

Bian Wu

Theoretical Foundation for Lecture Games

Thesis for the degree of Philosophiae Doctor

Trondheim, June 2013

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and
Electrical Engineering
Department of Computer and Information Science



NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

© Bian Wu

ISBN 978-82-471-4502-9 (printed ver.)

ISBN 978-82-471-4503-6 (electronic ver.)

ISSN 1503-8181

Doctoral theses at NTNU, 2013:195

Printed by NTNU-trykk

Abstract

Nowadays, computer games are played in a technology-rich environment equipped with laptops, smart phones, game consoles (mobile and stationary), set-top boxes, and other digital devices. It is believed that the intrinsic motivation for games in young people can be combined with educational content and objectives into what Prensky calls “digital game-based learning” [1]. In recent years especially, the innovative mobile electronic products, such as phones and Android phones, present new opportunities for Game-Based Learning (GBL). These devices can be combined with game content to be played in different locations such as classrooms, offices, homes, and outside, for formal and/or informal learning. Further, new game development tools, including some game editors, simplify the game development process and even let game players create their own games without programming. In this context, not only can a game be used for learning, but game development can also be used as assignments in education.

This thesis investigates how to apply games or game development as a motivation for lecture-based coursework learning using current computer technology. The term “lecture games” is defined and categorized in order to identify the research scope. Generally, games can be integrated in coursework in three ways. First, games can be used instead of traditional exercises motivating students to put more effort into the work, and giving the teacher and/or teaching assistants an opportunity to monitor how the students progress with the exercises in real-time. Second, games can be played within lectures to improve the participation and motivation of students. These two approaches presented above are categorized as “*Game as a motivation for lectures*”. The third way, categorized as “*Game development as a motivation for lectures*”, involves modification or development of a game as a part of coursework using a Game Development Framework (GDF) to learn specific skills. The latter method is termed “Game Development-Based Learning” (GDBL). This term is used to define a new research area. The GDF denotes the toolkits, which can be used to develop/build/modify games, e.g. game engines, game editors, game (simulation) platforms, or even an Integrated Development Environments (IDE) such as Visual C++. GDBL is typically used in computer-related courses, but can also be used in other fields, e.g., literacy in primary education [2]. Based on these concepts, a major challenge is to find a supportive theory from the perspective of game design and pedagogy, to guide the process of applying a game or game development in learning in the context of a technology-rich environment, and evaluate the results in education. In summary, the research goal is to use supportive theory and current computer technology as dual basis to facilitate lecture games in the contemporary technology-rich environment.

In order to describe the research questions and contributions clearly and systematically, the research questions have been grouped according to two topics. Topic 1 - “Games as a motivation for lectures”, deals with identifying supportive theory to guide the design and evaluation of lecture games, as well as application of current relevant technology and appropriate peripherals to provide various play experiences in the Lecture Games project. Topic 2 - “Game development as a motivation for lectures”, is concerned with game development based learning (GDBL), including the researchers’ views of GDBL,

and the GDBL characteristics in terms of supportive theory and the current technology-rich environment.

For the study of topic 1, the relevant literature review was undertaken to get an overview of the existing research, and four case studies were conducted with four multiplayer games, using quiz and other concepts, on various devices, including smart phones in lectures. In the study of topic 2, it was found that there were no existing literature reviews available, so a systematic literature review of GDBL was carried out. In addition, quasi-experiments were run integrating two GDFs, Microsoft XBOX New Architecture (XNA) Game Studio and Android Software Development Kit (SDK), in exercises for a software architecture course where students worked in teams to develop a game using their knowledge from this course. Then, based on the data and experiences in the above research, the supportive theory was identified for each topic to enrich the theoretical foundation for GBL. Further, the GBL field was extended in this study by including GDBL.

The main contributions for *game as a motivation for lectures* are:

C1: Identification of research topics and cases in regards to the recent technology-rich environment within the context of game as a motivation for lectures.

C2: An analysis chart of applying supportive theory and enabling technology to guide the study of educational games for lectures.

The main contributions for *game development as a motivation for lectures* are:

C3: Identification of a set of research themes and elements in GDBL.

C4: Identification of the factors contributing to the success or failure of GDBL.

C5: Framework of linked elements for the design of GDBL.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for partial fulfillment of the requirements for the degree of Philosophiae Doctor.

This doctoral work has been performed at the Department of Computer and Information Science (IDI), NTNU, Trondheim, Norway under the supervision of Professor Alf Inge Wang as the main supervisor, and Associate Professor Harald Øverby and Associate Professor Sara Brinch as co-supervisors.

This PhD thesis has been financed as an integrated PhD study by an internal scholarship from the Department of Computer and Information Science and the Faculty of Information Technology, Mathematics and Electrical Engineering at NTNU.

Acknowledgements

First of all, I would like to thank Professor Alf Inga Wang for his supervision, continuous support, and advice during my PhD studies. His inspiring thoughts and insightful remarks helped a lot at many stages in the course of this doctoral research. His careful editing contributed enormously to the production of my papers and this thesis. I would like to extend my thanks to my co-supervisors, Associate Professor Harald Øverby and Associate Professor Sara Brinch, for their encouraging guidance and suggestions during my time as a PhD student. Next, special thanks go to all members and participants in the Lecture Games project who provided valuable input for this research. I am also grateful to my colleagues and teachers who shared their knowledge and experience, and provided important feedback for my work on numerous occasions in the software engineering research group meetings.

I would like to express my gratitude to Associate Professor Hallvard Trætteberg, Meng Zhu, and Hong Guo for their suggestions during the project. I would also like to extend my gratitude to all stakeholders involved in the project, especially to some graduate students and Trygve Bragstad from Chamber of Commerce in Trondheim who provided background knowledge on Trondheim city in one case study.

I would also like to thank Alf Inge Wang, Jingyue Li, and Tor Stålhane for helping and cooperating in my teaching duties during these years. I would also like to take this opportunity to thank all my friends and colleagues who made my stay at NTNU memorable by sharing ideas, experiences, and good times. It is hard to mention everyone's name here, but it is even harder not to mention a few like Feng Luan, Chengzhi Liu, Huamin Ren, Shang Gao, Shengtong Zhong, Wei Wei, Jiangqiang Ma, Min Shi, Ellen Hoprekstad, Oskar Sündberg, Kirsti Elisabeth Berntsen, Birgit Krogstie, Salah Uddin Ahmed, Alfredo Perez, and Gry Seland.

Finally, I would like to thank my wife, Qiaoduo Shi, and my parents, Buyun Wu and Qingxiang Xiong in China, for providing inspiration, love, and enduring support during all these years.

Contents

Abstract.....	i
Preface.....	iii
Acknowledgements	iv
Contents	v
List of Figures.....	viii
List of Tables	ix
Abbreviations	x
1 Introduction.....	1
1.1 Problem Outline.....	1
1.2 Research Context.....	2
1.3 Research Questions	4
1.4 Research Design.....	5
1.5 Papers.....	7
1.6 Contributions	12
1.7 Thesis Structure.....	13
2 State of the Art	15
2.1 Lecture games scope and taxonomy.....	15
2.2 Understanding Learning Perspective towards Lecture Games	19
2.2.1 New approaches to collaboration for learning	22
2.2.2 Double stimulus from activity theory.....	23
2.2.3 Project-based learning.....	24
2.3 Game Design Theory for Lecture Games.....	25
2.3.1 GameFlow model from flow experiences.....	25
2.3.2 Motivational theory and intrinsic motivation theory about games	27
2.3.3 Other criteria: more than game features.....	28
2.4 Experimental Software Engineering	29
2.5 Enabling Technology.....	30
2.5.1 Prerequisites – the lecture environment.....	31
2.5.2 Mobile technology - games as motivation for lectures.....	31
2.5.3 Game development frameworks - game development as motivation for lectures	33
2.6 Summary.....	36
3 Research Methods.....	37
3.1 Research Design.....	37
3.1.1 Fixed design.....	38
3.1.2 Flexible design.....	38
3.1.3 Literature review	39
3.2 The Methods Selection for Data Collection	39
3.3 Dealing with the Data.....	40
3.3.1 Collecting data	40
3.3.2 Preparing for analysis	40
3.3.3 Quantitative analysis	40
3.3.4 Qualitative analysis	41

4 Research Process	42
4.1 Research Goal	42
4.2 Game as Motivation for Lectures - Case Study	44
4.2.1 Case study to answer RQ1 and RQ2.....	45
4.2.2 Data collection methods.....	47
4.2.3 Data analysis	47
4.3 Game Development as Motivation for Lectures - Literature Review.....	48
4.3.1 Systematic literature review to answer RQ3.....	48
4.3.2 Protocol development.....	49
4.3.3 Data source and search strategy.....	49
4.3.4 Data extraction with inclusion and exclusion criteria	49
4.3.5 Synthesis of findings	51
4.4 Game Development as Motivation for Lectures - Quasi-experiment.....	52
4.4.1 Quasi-experiment to answer RQ4	52
4.4.2 Experiment definition	53
4.4.3 Experiment planning	54
4.4.4 Data collection	57
4.4.5 Data analysis	57
5 Results	59
5.1 Summary of the Studies	59
5.2 Game as Motivation for Lectures	60
5.2.1 Case 1 - World of Wisdom	60
5.2.2 Case 2 - Lecture Quiz.....	61
5.2.3 Case 3 - Knowledge War.....	62
5.2.4 Case 4 - Amazing City Game.....	63
5.2.5 Summary for RQ1&RQ2.....	64
5.2.6 Contribution of the study.....	65
5.3 Game Development as Motivation for Lectures	71
5.3.1 Literature review of GDBL.....	72
5.3.2 Experiment 1 - XNA used as a GDF	73
5.3.3 Experiment 2 - Android SDK used as a GDF	73
5.3.4 Summary for RQ3&RQ4.....	74
5.3.5 Contribution of the study.....	75
6 Evaluation and Discussion.....	82
6.1 Evaluation of Research Questions	82
6.1.1 Evaluation criteria	82
6.1.2 Evaluation of the research questions.....	83
6.2 Evaluation of Contributions.....	85
6.2.1 Evaluation of contributions in game as motivation for lectures	85
6.2.2 Evaluation of contributions in GDBL.....	86
6.3 Evaluation of Validity Threats	87
6.3.1 Threats to validity of case studies	87
6.3.2 Threats to validity of literature review.....	88
6.3.3 Threats to validity of quasi-experiment	89
7 Conclusions	91
7.1 Contributions	91
7.2 Limitation and Future Work.....	92
7.3 Concluding Remarks	94

8 References	95
9 Appendix: Selected papers	106

List of Figures

Figure 1: Studies and their contribution connected with research questions and publications	7
Figure 2: Relations among lecture games, serious game, GBL and edutainment....	19
Figure 3: Framework of Research Design	37
Figure 4: Mapping of Research Design Framework into RQ1 and RQ2.....	44
Figure 5: Mapping of Research Design Framework into RQ3.....	48
Figure 6: Steps of the Study Selection Process.....	50
Figure 7: Mapping from Framework to Game Design to RQ4.....	52
Figure 8: the Architecture of World of Wisdom Game.....	60
Figure 9: System Overview of LQ 2.0.....	62
Figure 10: KnowledgeWar Architectural Overview.....	63
Figure 11: ACG User Interface	64
Figure 12: Relationships among Supportive Theory, Game Design and Evaluation	66
Figure 13: Research Issues Relations.....	66
Figure 14: Relations between Research Issues and Cases	67
Figure 15: A Practical Guidance of Analysis Chart for Educational Game Design....	71
Figure 16: Relations of Themes and Entities in GDBL	75
Figure 17: A Guideline for Technical and Pedagogical Co-design of GDBL.....	80

List of Tables

Table 1: Links between the Research Questions, Contributions and Papers.....	12
Table 2: Serious Game Taxonomy [31].....	16
Table 3: Taxonomy of Lecture Games	19
Table 4: Theoretical Context for Learning [4]	21
Table 5: Mapping the Elements from Games Literature to the Elements of Flow[86]	26
Table 6: Game Elements for Design Adapted from Malone and Lepper	28
Table 7: Questionnaires in SUS	30
Table 8: Examples of Apps that were used to Provide Game Play in our Games	32
Table 9: Study of GDFs for Novices.....	34
Table 10: Study of GDFs for Developers.....	35
Table 11: Relationship of each Entity in Research Process	44
Table 12: GQM Table for GDBL experiment	56
Table 13: A Brief Outline and Contribution of each Paper	60
Table 14: Characteristics of Good Educational Games	68
Table 15: Most Relevant Features for Design of an Educational MMORPG	69
Table 16: Study of GDFs	77
Table 17: Evaluation of Research Questions.....	85
Table 18: Strategies for Dealing with Threats to Validity	87
Table 19: Boote and Beile’s Literature Review Criterion	88

Abbreviations

ACG	Amazing City Game
AoC	Age of Computer
ATAM	Architecture Trade-off Analysis Method
COTS	Commercial, off-the-shelf
CS	Computer Science
CSCL	Computer-Supported Collaborative Learning
DGBL	Digital Game-Based Learning
GBL	Game-Based Learning
GDBL	Game Development-Based Learning
GDF	Game Development Framework
GQM	Goal/Question/Metric
HCI	Human Computer Interaction
IDE	Integrated Development Environment
IS	Information Systems
ISO	International Organization for Standardization
IT	Information Technology
KW	Knowledge War
LG	Lecture Quiz
MMORPG	Massively Multiplayer Online Role-Playing Game
NTNU	Norwegian University of Science and Technology
OS	Operating System
OSS	Open Source Software
PMA	Post-Mortem Analysis
QIP	Quality Improvement Paradigm
RQ	Research Question
SE	Software Engineering
SG	Serious Game
SPSS	Statistical Product and Service Solutions
SUS	System Usability Scale
SWA	Software Architecture

WoW World of Wisdom

1 Introduction

1.1 Problem Outline

The first successful commercial video game was developed about forty years ago [3]. Video games have quickly become one of the most pervasive, profitable, and influential forms of entertainment across the world. Further, it has been discussed for decades [4-6] how games can be integrated in education in order to stimulate students' interest for learning, to enhance the effectiveness of study, and to motivate the attendance in class.

In recent times, technology evolved into more diverse forms than before. Various educational goals can be achieved using advanced equipment and technologies, ranging from software applications, such as Wikipedia, YouTube, Facebook, and Twitter, to hardware platforms, such as Apple iPhone, Sony PlayStation, Nintendo DS, and Microsoft XBOX. All these technologies and devices can enrich the teaching and training environment, and provide better learning platforms through mobile network support such as Wi-Fi¹ or GSM². In this environment, many games emerged with the aim of improving learning, from playing educational games to other relevant game activities used in learning, e.g. using game development to teach programming. This phenomenon indicates that the appearance of educational game ideas is always accompanied by novel equipment or technology, e.g. sport games based on Wii-Fit³. Further, the advances in technology become a challenge to the educational games themselves, posing questions how can innovative game ideas be adapted to current technology in the context of a new generation of students, and how to design a game to get a good balance between entertainment and education. Some challenges and opportunities arise from establishing and enriching the theoretical base for Game-Based Learning (GBL). They stem from the necessity to combine pedagogical aspects and game technology in the design to improve the awareness of lecture games, and to enhance the knowledge construction process. The implementation of many educational games concentrates excessively on technological issues, missing the pedagogical and

¹ http://www.webopedia.com/TERM/W/Wi_Fi.html

² <http://www.webopedia.com/TERM/G/GSM.html>

³ <http://wiifit.com>

psychological context, or it focuses on the game idea itself and neglects the validation of the learning outcomes.

This thesis aims to further the research on these issues. It contributes to bridging the gap between the pedagogy and recent game technology by formulating the game design theory and evaluation criteria to enhance the foundation of GBL, especially for lectures in higher education.

1.2 Research Context

The *Lecture Games project*, which this PhD is a part of, is interdisciplinary by nature. It intersects the research fields of video games and learning. Lecture games are a sub-category of educational games and Game-Based Learning with the focus on games mainly related to a lecture in a classroom. The Lecture Games project research is carried out in the software engineering group at the Department of Computer and Information Science in cooperation with the Department of Telematics and the Department of Art and Media Studies at the Norwegian University of Science and Technology (NTNU). More specifically, the Lecture Games project aims to propose new game ideas to improve the traditional classroom teaching style in lectures for higher education. In addition, this thesis investigates how to integrate the popular technology into lectures, and how to validate the ultimate learning effectiveness through case studies and experiments. This project also involved other participants, working under supervision, such as graduate students developing games in sub-projects of the Lecture Games project. During the whole project process, the author was the main designer, executor, and evaluator of the project, while the games were implemented together with graduate students. Details of the author's work and contributions are described in Section 1.5.

The focus of the project was on the exploration of research issues at the intersection of lecture and games, and the intersection of pedagogy, technology, and game design methods. The ultimate objective is to propose, evaluate, and enrich teaching methods in lectures while facilitating GBL, and to formulate theory strengthening the theoretical foundation of GBL. The research background will be described in detail using the sub-categories listed below.

Digital games for learning: Video game development has now extended beyond pure entertainment to other areas. For example, the game "Driving theory training" on Nintendo DS, introduced in 2008, has been used to help learners study by simulating a realistic driving test. In addition to a complete test of theory based on the real life examination, learners can benefit from revision, graphs, mini-games, as well as questions and answers relating to driving vehicles. Another example is "Franklin: Birthday Surprise" on Sony PlayStation 2, which is a side-scrolling educational game starring Franklin the Turtle. Today, more people have an open mind about learning from games. However, in early 1990s, some educators feared that video games might foster violence, aggression, negative imagery of women, or social isolation, and they ignored the positive effects of using games in education [7]. Other educators saw video games as powerful motivating digital environments, and studied video games in order to determine how interesting factors in popular video games could be integrated into

instructional design [8-11]. This argument may still exist today, but by now, video games became a part of daily life. Indeed, video games have the potential to be valuable in schools, because much of the content students need to learn does not motivate them directly. The words “boring”, “dry”, or “technical” are often used to describe such situation whether the learners are in schools or universities. The attitude of today’s students toward the video games is the very opposite of the attitude that most of them have toward school, as outlined in “The Myth of the Educational Computer” [12]. Nowadays, games like SimCity are used in geography or urban planning classes, and Maxis (developer of SimCity) has published a set of resources for teachers on their website [13]. Further, the games can be interesting, competitive, cooperative and results-oriented, and most learners are motivated through such educational games in a variety of ways. Referring to lecture games, it could be considered to be a subset of GBL. The more understanding about GBL, the more helpful it will be for the Lecture Games project. Games, as a mediator tool, play a very important role in this project. Achieving the primary learning goal by this tool is the common aim in both GBL and lecture games. Since game research is not a mature and traditional research field, it does not have many systematic theoretical methods or foundations to guide the design process or evaluate results. In addition, how to find a useful theory to design GBL in the current technology-rich environment is still an open research question. This area deserves more research effort and this became a motivation for this study.

Gamification: “Gamification” as a term describing using game elements in non-game applications originated from the digital media industry. The first documented use dates back to 2008 [14], and the term became widely adopted after the second half of 2010. Similar terms are also introduced, such as “productivity games” [15], “surveillance entertainment” [16], “funware” [17], “playful design” [18], “behavioral games” [19], “game layer” [20], and “applied gaming”⁴. In the game industry and the game studies community, the term “gamification” is more widely used than in other fields. Vendors and consultants have tended to describe “gamification” practically in terms of client benefits, for example as “*the adoption of game technology and game design methods outside of the games industry*” [21], “*the process of using game thinking and game mechanics to solve problems and engage users*” [22], or “*integrating game dynamics into your site, service, community, content or campaign, in order to drive participation*”⁵. Gamification is the infusion of game mechanics, game design techniques, and/or game style into any other activity or product. It typically involves applying game design thinking into non-game applications to make them more enjoyable and engaging. The core idea is to extract the elements of game mechanics and apply them into a product to make it more interesting. The main feature of gamification is not to use games directly in a non-game field, but to apply game elements as an abstraction level into a product. For example, some online forums require users to get more skills to upgrade to a higher level, which allows them to get more functionality in the forum. Sebastian Deterding [23] proposed a definition of this phenomenon as “*the use of game design elements in non-game contexts.*” He explains further that “gamified” application refers to: the use (rather than the extension) of design (rather than game-

⁴ <http://www.natronbaxter.com>

⁵ <http://www.bunchball.com/nitro/>

based technology or other game-related practices) elements (rather than full-fledged games) related to characteristics of games (rather than play or playfulness) in non-game contexts (regardless of specific usage intentions, contexts, or media of implementation). A common example of gamification in the real world is “Frequent Flyer Programs”, offered by many airlines. Typically, airline customers enrolled in the program accumulate frequent flyer miles corresponding to the distance flown on that airline or its partners. Acquired miles can be redeemed for free air travel, for other goods or services. Another example of applying gamification and serious games in personalized health could be found in [24]. This suggests that by including game elements any process can potentially benefit motivating users to take part and find enjoyment, and this concept could be used in GBL.

The current technology environment impacts on both lectures and students:

Technology always has both positive and negative effects on human life. Whether the effects are positive or negative usually depends on how it is applied. This means that new technology brings challenges and opportunities to our life, including the game and learning fields. Traditional lectures have been undergoing a gradual change through constantly updated technology. Several years ago, there were no projectors or LCD screens in classrooms, but now not only they are being used routinely, but also most classrooms provide several ways of accessing computer networks. In this context, some exciting game ideas came into being. These new ideas are based on the recent technology-rich environment combined with new equipment in daily use, from software aids to hardware support, mentioned in Section 1.1. This phenomenon can become an inspiration for GBL, providing the possibility to broaden the teaching design involving recent technology in a positive way. Further, the combination of these technologies and lecture games matches the expectations of the new generation of students who grew up in a technology culture and are familiar with new devices and technical novelties, for example the touch screen found on smart phones and tablets.

The above discussion shows that new concepts, like gamification, change of learners’ attitude, and technology development play important roles in the Lecture Games interdisciplinary project helping to achieve the learning goal. Therefore, the Lecture Games project will consider using innovative computer technology for new generation gamers as the basic context for the design and study.

1.3 Research Questions

The main research goal for this PhD work within the Lecture Games project was to use *supportive theory* and *current computer technology* as dual basis to facilitate lecture games in the current technology-rich environment. It means that this PhD work focused on the survey and identification of supportive theory in areas such as pedagogy, game design, and evaluation criteria, in order to build a framework for the lecture games design and evaluation. Moreover, selecting and integrating relevant technology into lecture games design was essential to provide interesting play experiences for students. In summary, the lecture games should be designed based on both supportive theory and relevant technology. Specifically, games can be integrated in lectures in three ways as mentioned in the Abstract. The first two approaches, i.e. games, used in exercises and played within lectures, were part of research topic 1 - “Game as a motivation for

lectures”. The third approach, integrating game development in students’ exercises based on a Game Development Framework (GDF), was given a new name - Game Development Based Learning (GDBL). This latter approach belongs to research topic 2 - “Game development as a motivation for lectures”. Based on the description of the research context in Section 1.2 and in order to improve the lecture process through GBL with the supportive theory and innovative technology, the research questions corresponding to these two topics are described as follows:

Topic 1: Game as a motivation for lectures:

RQ1: How can supportive theory be identified to guide the design and evaluation of lecture games?

RQ2: How can current relevant technology and appropriate peripherals be used to provide various play experiences in new lecture games?

Topic 2: Game development as a motivation for lectures:

RQ3: What is game development based learning (GDBL) and what are the researchers’ views of GDBL?

RQ4: How can the GDBL be characterized in terms of supportive theory and the current technology-rich environment?

1.4 Research Design

The study described in this thesis aims towards building a theoretical knowledge base at the intersection of learning and games within the context of technology-rich environments. The specific research methods, which have been used in this thesis, are case study, quasi-experiment, and systematic literature review. The case study and quasi-experiment were exploratory in nature and have been conducted by following the strategy defined by Colin Robson [25]. The systematic literature review was carried out by following the methods described by Bryony Oates [26] and Bootes and Beile [27]. The case study was used in research topic 1 since literature reviews for this topic already exist. For topic 2, both a systematic literature review and quasi-experiments were used in the study.

The research design focused first on a thorough examination of the current state of knowledge, providing taxonomy of the different solutions, and pointing out the weaknesses or shortcomings existing in the research area. Afterwards, the focus of the research was on two topics presented above.

For topic 1: “Game as a motivation for lectures”, two case studies were conducted to identify the common issues in the game design and evaluation. One case was a multiplayer online game with quiz fights as an exercise, running on both Windows and Mac OS X. The design was based on the pedagogical theory and game design theory extracted from the literature survey. Another case was a multiplayer quiz game for the

lectures, running on mobile devices. This was the second version of the game - Lecture Quiz [28]. It was designed in terms of game design theory extended from Malone intrinsic motivation [29]. Both case studies aimed to fill the theoretical gap for lecture games. Data obtained from this study was used for finding answers to research question RQ1. In addition, two other case studies were conducted, and they showed that the recent technology may influence education and provides various ways to combine with learning. One was a social quiz for schools, running on iPhones. Another one was a pervasive educational game running on Android smart phones. Both took advantage of current popular features of smart phones, e.g. GPS and camera, and provided users with different play experiences. The experiences and evaluation data gained in these two case studies contributed to finding answers to research question RQ2.

For topic 2: “Game development as a motivation for lectures”, a systematic review of literature and two quasi-experiments were conducted to identify GDBL’s effectiveness - learning through a game development assignment in a technology-rich environment. The term GDBL was created to define this research area since no term existed to describe it. In addition, there has been no prior literature review work in this field. Therefore, a systematic literature study was carried out to validate the original GDBL method. It aimed at answering research question RQ3. In the meantime, two quasi-experiments used XNA and Android SDK as GDFs in a NTNU’s software architecture course. The course structure was changed to integrate game development as a basis for the exercises in the students’ project. The students worked in teams to develop a game using either XNA or Android SDK in order to apply the course content in practice. A non-GDBL project was also provided in this experiment in order to obtain evaluation data for both GDBL and non-GDBL methods. Comparing the results served to reveal the differences between GDBL and non-GDBL, and to find answers to RQ4.

The theories and enabling technology mentioned in the study of topic 1 and topic 2 will be described in detail in Chapter 2.

Figure 1 shows the relationship between the studies (research methods), contributions, papers, and research questions. The papers are listed in Section 1.5 and the contributions are listed in Section 1.6. A link between RQ-n (research questions), and G-n (papers on “Game as a motivation for lectures”) or GDF-n (papers on “Game Development as a motivation for lectures”) indicates that research question RQ-n was addressed in paper G-n or GDF-n. A link between Study A and paper G-n/GDF-n indicates that paper G-n/GDF-n describes the results of Study A. A contribution is represented as a circle C. The list of papers, which add to a particular contribution C-m, is positioned next to the circle C-m. In addition, the research goal converts into two factors in the figure. The horizontal line represents one factor that both research topics are driven by supportive theory. The supportive theory, adopted at the beginning of the study from areas of pedagogy and game design, should direct the design process for each topic. Each topic may involve independent supportive theories according to its features. The vertical line represents the fact that both topics depend on the recent widespread computer technology, especially mobile technology used mainly in topic 1 and GDFs used in topic 2.

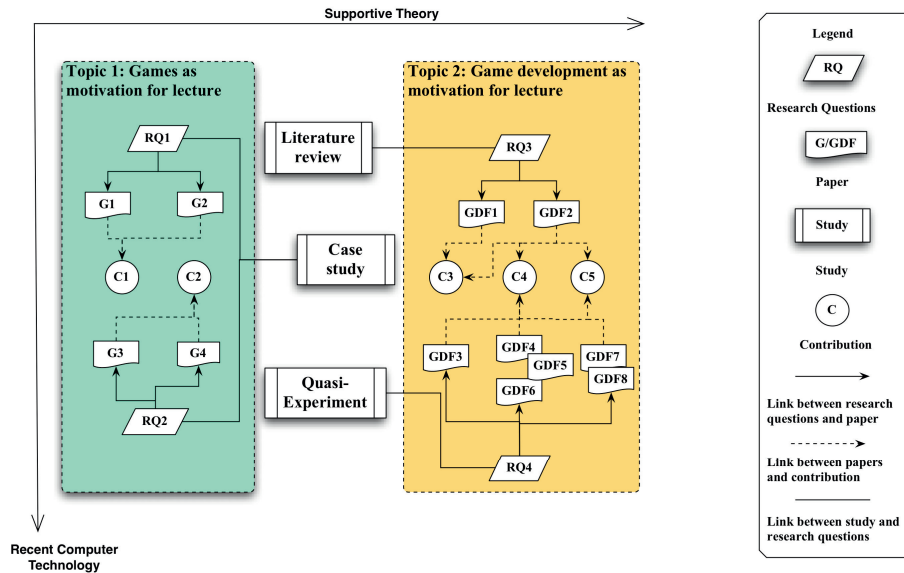


Figure 1: Relationship of studies and their contribution to research questions and publications

1.5 Papers

This thesis is based on a collection of published papers. A list of these twelve papers is provided below, divided into two research topics. There are four papers about the case studies for topic 1; the remaining eight papers are literature reviews and experiments related to topic 2. The author’s contribution to these papers is stated for each paper.

Topic 1: Game as a motivation for lectures - four case studies

G1: *Bian Wu*, Alf Inge Wang and Yuanyuan Zhang, "Experiences from Implementing an Educational MMORPG", *2nd International IEEE Consumer Electronics Society's Games Innovation Conference (GIC 2010)*, Hong Kong, 21-23 December 2010. ISBN: 978-1-4244-7178-2, DOI: 10.1109/ICEGIC.2010.5716896

Relevance to this thesis: This paper is mainly a tentative case study of using MMORPG in education. It gives answers to research question RQ1, and adds mainly to contribution C1.

Author’s contribution: This paper is the result of a two-year sub-project to implement a game style exercise as an alternative to the traditional paper exercise. The game content and architecture was designed, seven students were

supervised, and the development process was directed over two years. The author was the leading writer of this paper.

G2: *Bian Wu*; Alf Inge Wang; Erling Andreas Børresen; Knut Andre Tidemann: "Improvement of a Lecture Game Concept - Implementing Lecture Quiz 2.0", *3rd International Conference on Computer Supported Education*, 6-9 May 2011, Noordwijkerhout, The Nederland. ISBN: 978-989-8425-50-8

Relevance to this thesis: This paper describes a case study of using common mobile devices and the existing technology infrastructure in education. It gives answers to research question RQ1, and adds to contribution C1.

Author's contribution: The author reviewed and evaluated two versions of LQ, 1.0 and 2.0, and contributed to the introduction, related work, game design, evaluation, and conclusion. The author was the leading writer of this paper.

G3: Alf Inge Wang, ***Bian Wu***, Sveinung Kval Bakken, "Experiences from Implementing a Face-to-Face Educational Game for iPhone/iPod Touch", *2nd International IEEE Consumer Electronics Society's Games Innovation Conference (GIC 2010)*, 21-23 December 2010, Hong Kong. ISBN: 978-1-4244-7178-2. DOI: 10.1109/ICEGIC.2010.5716895

Relevance to this thesis: This paper presents a case study of using a popular mobile device, iPhone, in education. It gives answers to research question RQ2, and adds to contribution C2.

Author's contribution: This paper is the result of cooperation with the thesis supervisor and another student. The author performed the related work, data extraction and analysis, and further discussion of the results.

G4: *Bian Wu*, Alf Inge Wang, " A Pervasive Game to Know Your City Better", *2011 International IEEE Consumer Electronics Society's Games Innovation Conference (IGIC 2011)*, November 2011, Orange, California, USA.

Relevance to this thesis: This paper mainly describes a case study of using a popular mobile device, Android smart phone, as a tool for informal learning. It gives answers to research question RQ2, and it adds to contribution C2 and, to some degree, C1.

Author's contribution: This paper is the result of integrating pervasive game and popular technology in learning to construct a game concept supporting education. The author designed the game, conducted game evaluation, and was the leading writer of this paper.

Topic 2: Game development as a motivation for lectures - literature review

GDF1: *Bian Wu*, Alf Inge Wang, "Game Development Framework for Software Engineering Education", *2011 International IEEE Consumer Electronics Society's Games Innovation Conference (IGIC 2011)*, November 2011, Orange, California, USA.

Relevance to this thesis: This paper is based on earlier experimental results of integrating GDFs in a software architecture course. It is a survey of related research methods using game design or game development in the software engineering field. It gives answers to research question RQ3, and it adds to contribution C3.

Author's contribution: This paper is the result of literature review. The author conducted the literature survey work and carried out the summary with the supervisor's support. The author was the leading writer of this paper.

GDF2: *Bian Wu*, Alf Inge Wang, "A guideline for game development-based learning: A literature review", *Accepted by the International Journal of Computer Games Technology*.

Relevance to this thesis: This paper presents a systematic literature review and investigation of the GDBL method. The review collected data related to using the game development in all possible educational fields. It gives answers to research question RQ3 and adds to all the contributions in the GDBL field.

Author's contribution: The author conducted a systematic literature review with the supervisor's guidance. This included searching, collecting the results and data from different bibliographies, such as IEEE Xplore and ACM portal, the summary of the data and the analysis of their common characteristics, as well as creation of a valuable framework for the future work.

Topic 2: Game development as a motivation for lectures - quasi-experiments

- **Experiment preparation:**

GDF3: Alf Inge Wang, *Bian Wu*, "Using Game Development to Teach Software Architecture", *International Journal of Computer Games Technology*, vol. 2011, Article ID 920873, 12 pages, 2011. ISSN: 1687-7047 EISSN: 1687-7055. DOI: 10.1155/2011/920873

Relevance to this thesis: This paper presents a re-design of a traditional software architecture course to integrate game development in the coursework. It describes the experience of changing the course setting to apply the GDBL method. It gives answers to research question RQ4, and it adds to contribution C4.

Author's contribution: This paper is the result of cooperation with the supervisor. The author performed the role of a teaching assistant in the course and mainly worked on the exercise improvements.

- **Experiment 1:**

GDF4: Alf Inge Wang, *Bian Wu*, "An Application of a Game Development Framework in Higher Education", *International Journal of Computer Games Technology*, Special Issue on Game Technology for Training and Education, Volume 2009. ISSN: 1687-7047 EISSN: 1687-7055. DOI=10.1155/2009/693267

Relevance to this thesis: This paper presents the preliminary findings in the GDBL field. It provides an initial conceptual framework to integrate GDBL in a software architecture course. It gives answers to research question RQ4, and it adds to contribution C4 and, to some degree, C5.

Author's contribution: This paper is the result of cooperation with the supervisor. The author was a teaching assistant in the software architecture course using the GDBL method. This paper proposed an initial conceptual framework for the design of GDBL based on experiences in the software architecture course.

GDF5: *Bian Wu*, Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "An Evaluation of Using a Game Development Framework in Higher Education", *22nd IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2009)*, February 17-19, Hyderabad, India, 2009. ISBN: 978-0-7695-3539-5 DOI=10.1109/CSEET.2009.9

Relevance to this thesis: This paper presents the evaluation data of the initial experiment with using XNA in teaching a software architecture course. It follows the lecture design presented in the papers - GDF3 and GDF4, and presents a positive initial feedback to integrating GDBL in the course. It gives answers to research question RQ4, and it adds to contribution C4.

Author's contribution: This paper is the result of cooperation with the supervisor. The author contributed to the introduction, related work, data extraction and analysis, evaluation, and conclusion. The author was the leading writer of this paper.

GDF6: *Bian Wu*, Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "XQUEST used in Software Architecture Education", *IEEE Consumer Electronics Society's Games Innovation Conference*, August 25-28, 2009, London, UK. ISBN: 978-1-4244-4459-5, DOI: 10.1109/ICEGIC.2009.5293607

Relevance to this thesis: This paper presents the next step in research of the GDBL field. An extended Lib-XQUEST was provided, based on XNA, to simplify the students' effort in the game programming. It gives answers to research question RQ4, and it adds to contribution C4.

Author's contribution: This paper is the result of cooperation with the supervisor and two other students. The author contributed to the introduction, related work, data extraction and analysis, evaluation, and conclusion. The author was the leading writer of this paper.

- **Experiment 2:**

GDF7: *Bian Wu*, Alf Inge Wang; Anders Hartvoll Ruud; Wan Zhen Zhang: "Extending Google Android's Application as an Educational Tool", *the 3rd IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL)*, April 12-16 2010, Kaohsiung, Taiwan. ISBN: 978-1-4244-6433-3. DOI: 10.1109/DIGITEL.2010.38

Relevance to this thesis: This paper presents the second step in research of the GDBL field. It is based on the previous design and feedback to integrating XNA in the course. The Android operating system was used in the software architecture course as a game development tool. This paper gives answers to research question RQ4, and it adds to contribution C4.

Author's contribution: This paper reports the results of adding a new GDF to the software architecture course. The author supervised the 3rd author in the design and implementation of Sheep, an extended software library based on Android, to aid the game development. The author conducted the theoretical design, literature review, data extraction, analysis, and further discussion. The author was the leading writer of this paper.

GDF8: *Bian Wu*, Alf Inge Wang, "Comparison of Learning Software Architecture by Developing Social Applications vs. Games on the Android Platform", *International Journal of Computer Games Technology*, Volume 2012, Article ID 494232, 10 pages, 2012. ISSN: 1687-7047 EISSN: 1687-7055. DOI: 10.1155/2012/494232

Relevance to this thesis: This article presents the evaluation results of using Android as a development tool in the project in a software architecture course. The experiment aims to provide an answer to research question RQ4, it adds to contribution C4 and, to some degree, C5.

Author's contribution: The author performed the experiment with the supervisor's help, designed the questionnaire for the data collection, collected the data, and used SPSS to analyze the data. The collected data included students' feedback, complexity of the project, the effort students put into the

project, and project grades to determine the features of GDBL in the context of using Android as GDF in the software architecture course.

1.6 Contributions

The contributions of this thesis are as follows:

The main contributions to topic 1: game as a motivation for lectures:

- C1: Identification of research issues and cases in the contemporary technology-rich environment within the context of game as a motivation for lectures.
- C2: An analysis chart of applying supportive theory and enabling technology as a dual guideline in the study of educational games for lectures.

The main contributions to topic 2: game development as a motivation for lectures:

- C3: Identification of a set of research themes and elements in GDBL.
- C4: Identification of factors contributing to the success or failure of GDBL.
- C5: Framework of linked elements for the design of GDBL.

Table 1 maps the connections between research questions, papers, and contributions.

Table 1: Links between the research questions, contributions, and papers

Topic	Research questions	Contributions	Papers	Research Design	Focus
Game as a motivation for lecture	RQ1	C1	G1, G2	Case study	Supportive theory; Recent computer technology
	RQ2	C2	G3, G4		
Game development as a motivation for lecture	RQ3	C3, C4, C5	GDF1, GDF2	Literature review	
	RQ4	C4, C5	GDF3, GDF4, GDF5, GDF6, GDF7, GDF8	Quasi-experiments	

1.7 Thesis Structure

The structure of the remainder of this PhD thesis is as follows:

Chapter 2: State of the Art

This chapter gives an overview of games and learning through taxonomy, defines the research scope, and then gives a short introduction to four aspects relevant to this research: 1) Understanding learning perspective of lecture games, 2) Game design theory for lecture games, 3) Experimental software engineering, and 4) Technical issues related to lecture games. The first three aspects relate to the supportive theory survey and the last aspect relates to the current computer technology.

Chapter 3: Research Methods

This chapter presents the big picture of the research methods, including the ontological and epistemological views. Then, it briefly describes theoretical aspects of specific research methods and data analysis methods.

Chapter 4: Research Process

This chapter presents the complete research process based on the research methods in Chapter 3. The process of the studies is discussed showing how the research methods were applied in this research and how the research strategies were chosen for this study, from high level down to details.

Chapter 5: Results

This chapter describes the main results for topic 1 based on the case studies, and the results for topic 2 based on a literature review and quasi-experiments. In addition, it presents the answers to research questions and the final contributions.

Chapter 6: Evaluation and Discussion of Results

This chapter presents the evaluation and discusses the research results with respect to the research questions and claimed contributions. In addition, it discusses the validity of the research methodology.

Chapter 7: Conclusion

This chapter sums up the main findings of this study, presents the limitations of the work, and outlines possible future work as a continuation of this research in the interdisciplinary field of games and learning.

Appendix

Appendix A contains the twelve papers, which have been accepted or published in conferences and journals. These papers contain the material on which this thesis is based.

2 State of the Art

This chapter presents the context for the study in this thesis. An overview of the research background is given by introducing taxonomy from the perspective of games and learning. The lecture games are an inter-disciplinary field with no existing systematic theory for the design and evaluation of such games. Therefore, concepts are borrowed from various theoretical fields, such as learning theory to guide the design of lecture games, and experimental software engineering to guide the evaluation of lecture games. To summarize, three main aspects related to supportive theory of lecture games are discussed: pedagogical context, game design theory, and experimental software engineering. Finally, a survey of recent computer technology is presented.

2.1 Lecture games scope and taxonomy

After defining the main research goal and the research questions, the first challenge was to identify the research scope of the Lecture Games project and its relations to other fields, i.e. serious games, GBL, and edutainment. These three fields are sometimes overlapping. The term “serious game” came into wide use with the emergence of the Serious Games Initiative in 2002 (seriousgames.org). The website of the Serious Games Initiative provides the following description of serious games:

“The Serious Games Initiative is focused on uses for games in exploring management and leadership challenges facing the public sector. Part of its overall charter is to help forge productive links between the electronic game industry and projects involving the use of games in education, training, health, and public policy.”

Zyda [30] gave a more formal definition, “*Serious game: a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives.*” Commonly, these definitions describe serious games as attempting to achieve something more than entertainment, i.e. “*games that do not have entertainment, enjoyment or fun as their primary purpose*” [31].

GBL is described as “*a branch of serious games that deals with applications that have defined learning outcomes*” (en.wikipedia.org). Others consider GBL and serious games to be more or less the same (e.g., Corti [32]). According to Corti, GBL has the potential for improving training activities and initiatives by virtue of its engagement, motivation, role-playing, and repeatability (failed strategies can be modified and tried again). Game-

based education or educational games are another expression of GBL. Digital game-based learning (DGBL) is closely related to GBL, with the additional restriction that it concerns digital games. Edutainment, education through entertainment, was a popular field in the 1990s with the growing multi-media PC market [31]. Generally, edutainment refers to any kind of education that also entertains, even though it is usually associated with video games with educational aims. In general terms, serious games are associated with “*games for purposes other than entertainment*” [33]. Serious games and edutainment have the same aims, but serious game also involves all aspects of education (e.g. teaching, training, and informing) and all ages.

The taxonomy of serious games, described below, will help to clarify further the serious games characteristics. The book “Understanding Video Game” [34] contains one chapter on serious games. In this book, serious games are classified into commercial educational video games, often known as *edutainment*; commercial entertainment with basic educational purposes and research-based educational video games. These three types have different focuses. First category teaches the player certain specific skills, while the second category teaches the player freely without focusing on specific skills. The third category focuses on the innovative learning style and documentation about educational effectiveness.

Another serious game taxonomy was proposed by Ben Sawyer [35]. He refuted the notion that “*serious game equals to games for learning or training*” and stated that all games are serious in some aspects. He uses vertical and horizontal axes to determine the game type. The vertical axis corresponds to the purpose of the game, for instance games for health, games for training, games for education, and games as work. The horizontal axis corresponds to the application field of the game, for instance, defense, healthcare, education, corporate, industry, as shown in Table 2. In terms of placement on two axes, the final taxonomy is as follows:

Table 2: Serious Game Taxonomy [35]

	Game for health	Games for advertising	Games for training	Game for Education	Games for Science and Research	Production	Games as Work
Government & NGO	Public health education & mass casualty response	Political games	Employee training	Informing public	Data collection/planning	Strategic & policy planning	Public diplomacy, opinion research
Defense	Rehabilitation & wellness	Recruitment & propaganda	Soldier /support training	School or house education	Wargames /planning	War planning & weapons research	Command & control
Healthcare	Cybertherapy / exercise gaming	Public health policy & social awareness campaigns	Training games for health professionals	Games for patient education and disease management	Visualization & epidemiology	Bio-tech manufacturing & Design	Public health response planning & logistics
Marketing & Communications	Advertising Treatment	Advertising, marketing with games, product placement	Product use	Product Information	Opinion research	Use real-time 3D computer graphics rendering engines to create cinematic productions.	Opinion Research
Education	Information about	Social issue games	Training teachers/	Learning	Computer science &	Point-to-Point(P2P) learning,	Teaching, distance

	diseases/risks		training workforce		recruitment	constructivism, documentary	learning
Corporate	Employee health information & wellness	Customer education & awareness	Employee training	Continuing education & certification	Advertising/visualization	Strategic planning	Command & control
Industry	Occupational safety	Sales & recruitment	Employee training	Workforce education	Process optimization simulation	Nano/bio-tech design	Command & control

In addition, James [36] argues that educational researchers have much to learn about learning from good computer and video games. Such good games and game technologies can be used to enhance learning. One example is *Age of Mythology*⁶ used in primary school education. Students read about mythology inside the game and get to know mythology from outside of game or wrote stories connected with the game and other mythological themes. Similar examples could be *Neverwinter Nights*⁷ or *Civilization*⁸. These games were not originally designed for serious purpose, but resulting in educational features. The lecture games put education as the a priori design criterion. Accordingly, they are different from pure entertainment games featuring learning principles.

The above examples of taxonomies show different perspectives and rules to determine the category of a “serious game”, with GBL and edutainment as their sub-categories. This is a useful method to identify the research scope. In the light of early results of investigating the project context, the aim was to create taxonomy of lecture games in order to define this term and the research scope, as well as to describe game examples and related technology for each game genre.

In the Lecture Games project, the games can be integrated in the lecture in three ways, which were briefly introduced earlier in this thesis. Here is a more detailed description:

First, games can be used instead of traditional exercises motivating students to put more effort into the exercises, and giving a teacher and/or teaching assistants an opportunity to monitor how the students proceed with the exercises in real-time. A game such as the “Age of Computers” (AoC) takes a historical approach to computer science by combining collaborative possibilities, simulations, and quiz games framed in a massive multiplayer online role-play game [37]. Charge Master is a Windows compatible software package, which aids in visualizing equipotentials⁹ produced by systems of point charges [38]. It accomplishes this via an educational game with an option to plot equipotentials. The Schools Quiz [39] is an educational game based on the popular “Buzz!” series. The game’s 5,000 questions are based on the curriculum for UK pupils between the ages of 7 and 11 years.

Second, games can be used within lectures in a classroom to improve the participation and motivation of students. This includes a multiplayer quiz game called Lecture Quiz

⁶ http://www.microsoft.com/games/ageofmythology/norse_home.aspx

⁷ <http://nvwault.ign.com/>

⁸ <http://www.civilization.com/>

⁹ Equipotentials : composed of points all at the same potential of a surface or line

(LQ) [40], where multiple players can participate using their own mobile phones, and the teacher moderates the game using his own PC and a video projector. Lucas Arts has lesson plans on its website to help teachers use their games to teach critical thinking [41]. Microsoft has sponsored a “Games-to-Teach” project at MIT which is building games for learning difficult concepts in physics and environmental science on the XBOX and Pocket PC [42]. The teachers can use these games to explain relevant concepts in classroom. The first and second approaches are combined as topic 1 - “Game as a motivation for lectures”, already mentioned in Section 1.3.

Third, the students are required to develop a game as a part of a course using a GDF to learn skills in computer science, software engineering, or other relevant courses. This is known as GDBL. Recently, there has been an increase in the number of game editor environments and game engines, which allow users to customize their gaming experiences by modifying and building games. “Learning through game modding (modifying)” [43] describes the use of modifying existing games with game editors to learn computer science, mathematics, physics, and aesthetic principles. It describes two exploratory case studies of modifying games in classroom settings to illustrate skills learned by students. It also describes how game design motivates students to acquire and apply these skills and how different game engines can be used to let students focus on the acquisition of particular skills and concepts in the classroom. In addition, the ACM curriculum for computer science explicitly mentions using game development as a way of motivating software engineering students.¹⁰ Simon Egenfeldt-Nielsen refers to GDBL as learning by making games.¹¹ ITALICS e-journal published an issue on learning by making games.¹² In addition, computer game development as a literacy activity [2] considers computer game development as a pedagogical activity, which can motivate students to improve literacy. The students were asked to develop computer games using a game development shell to acquire literacy without programming. This case shows that GDBL can be used outside of both computer science field and higher education. Similar methods also appeared in [44-49]. All of these types of game related methods have one feature in common - they depend on a mediator to teach a subject. This mediator could be any GDF such as XNA [50], Java Instructional Game Engine [51], Scratch [52], Warcraft map editor, or Alice [53]. This third approach is covered by topic 2 - “Game development as a motivation for lectures”, which was mentioned in Section 1.3.

Based on the discussion above, the Lecture Games in this thesis are defined as “use of games and game development as a motivation for lectures and students’ exercises.” Lecture games can be categorized according to three main approaches. Table 3 shows how the two topics explored in this thesis map to the lecture games taxonomy.

¹⁰ <http://www.acm.org/education/curricula/ComputerScience2008.pdf> (Section 6.3)

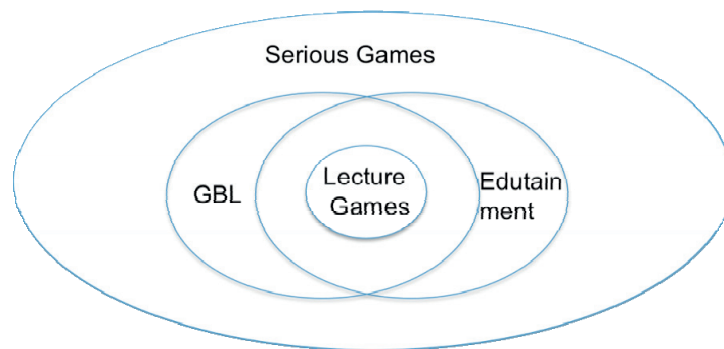
¹¹ <http://egenfeldt.eu/blog/2012/01/13/the-best-schools-if-you-are-looking-to-use-games/>

¹² <http://www.ics.heacademy.ac.uk/italics/vol5iss3.htm>

Table 3: Taxonomy of Lecture Games

Topic	Approach
Topic 1: Game as a motivation for lectures	Exercise games
	Classroom games
Topic 2: Game development as a motivation for lectures	GDBL

The above taxonomy provides a complete overview of the research scope for this thesis, further illustrated by Figure 2 showing the relations among the Lecture Games project, GBL, edutainment, and serious games.

**Figure 2: Relations among lecture games, serious games, GBL, and edutainment**

In addition, we defined a new research area, GDBL as an extension of the GBL field with lecture games being a subset of GBL. The next step was to identify relevant supportive theories and enabling technology for each topic.

2.2 Understanding Learning Perspective towards Lecture Games

In order to understand the potential role of games in support of learning, the term “learning” must be defined. There are multiple definitions of learning, with significant areas of disagreement as to both what it means to learn and what forms of learning are valuable.

The literature review of educational video game design [54] presented three aspects of the learning theory in relation to games:

1) *Constructivism*: Some researchers found that learning with well-designed video games adheres to constructivist principles [36, 55-58]. E.g. Corbit [59] presented a multi-user virtual world, SciCtr, and pointed out the merits of the constructivist approach used to analyze virtual environments. According to Corbit, the paths to navigate, and content embedded in these virtual worlds, are constructed by the developer/learner through meticulous research and thoughtful design.

2) *Constructionism*: Designing and developing computer games, rather than playing them, constitutes a constructionist approach to learning with games [60, 61]. El-Nasr and Smith [43] viewed game modding (one type of GDBL), i.e. developing new components inside of a game using toolkits within this game itself or other related game tools as a constructionist process of learning. It involves two activities: “*the construction of knowledge through experience and the creation of personally relevant products. The theory proposes that whatever the product, e.g. a birdhouse, computer program, or robot, the design and implementation of products are meaningful to those creating them and that learning becomes active and self-directed through the construction of artifacts*”. Steiner [62] concluded that, “*children as design partners improve the technologies they consume as well as gain educational benefits from the experience*”.

3) *Situated Cognition*: Learning theory for the analysis of educational video games, especially for simulation games, are combined with situate learning to provide an authentic context and involved players by allowing them to practice (play) again and again [63]. Lunce [64] argued that situated or contextual learning provides the rationale for simulation games because of their ability to provide a simulated context in which to situate learning. E.g. SimCity has ability to situate learning that players can run their own city as practice. Basically, the situated learning has better outcome in knowledge understanding than traditional learning [55, 57, 58, 65].

Ray Schroeder [66] presented four levels for learning theories:

1) Collected opinions about learning from different fields: Behaviorists (Thorndike, Pavlov, Skinner) focus only on the objectively observable aspects of learning. Cognitivists (Craik, Tulving, Ausubel) look beyond behavior to explain brain-based learning by including motivation, thinking, memory, and reflection. Constructivists (Piaget, Vygotsky, Bruner) view learning as a process in which the learner actively constructs or builds new ideas or concepts.

2) Mental Representations: Paivio’s Dual Coding Theory [67, 68], a theory of cognition, proposed that there are two ways a person could expand on learned material: verbal associations and visual imagery. Words and pictures together enhance cognitive coding more than one of them in isolation.

3) Cognitive theory of multimedia learning [69, 70]: Mayer [69] extends Paivio’s theory suggesting that pictures can be “animation” and text can be “narration”, with an emphasis on computer-based multimedia presentations. In addition, prior knowledge influences integration of pictures and text into “working memory” [71].

4) Connectivism [72, 73] is a recent theory of networked learning [74], which focuses on learning as making connections. Viewing personal knowledge as a “network” is a mark of “*a learning theory for the digital age*”. One aspect of connectivism is the use of a network with nodes and connections as a central metaphor for learning. Consequently, connectivism sees learning as a process of creating connections and developing a network. Finally, Ray Schroeder proposed “*just do it right*” as a learning theory. This

includes motivating learners, promoting meaningful learning, encouraging interaction and collaboration, giving timely constructive feedback, and looking at the whole person with the learner and learning process at the center.

There is a continuing discussion and debate about some of the learning theories. Table 4 below is adapted from [4] and it defines key “battle lines” in this debate. This table also shows an overview of learning theories.

Table 4: Theoretical Context for Learning [4]

Aspect	Behaviorist	Cognitivist	Humanist	Social and situational
View of the learning process	Changes behavior	Process entirely in the mind of the learner (including insight, information, processing, memory, and perception)	A development of personal potential	Interaction/observation in a group context, akin to an apprenticeship
Site of learning	External resources and tasks are what matters	Making connections in the learner’s mind is what really matters	Emotion, attitude, and thinking are important	Learning needs a relationship between people and environment
Purpose in education	Produce behavioral change in a desired direction	Develop capacity and skills to learn better	Become self-reliant, autonomous	Full participation in communities of practice, i.e. the learner graduates from apprentice to craftsman

As can be seen from the above schematic presentation, these aspects of learning involve contrasting ideas regarding the purpose and process of learning as well as the roles of educators. Considering different views on learning, it can be stated that it is a process, which leads to change in behavior, change in ways of thinking, achievement of personal potential, or development of capacity to operate within particular communities. These processes are not mutually exclusive. The understanding of learning from different perspectives helps to get an overview of the meaning of learning as an experience which could happen anywhere at anytime to anybody.

However, when learning is considered as a research area, from a general level down to specifics, the questions to answer are who is learning what, where, and why. This is a pragmatic approach, proposed by Prensky [1] and John Kirriemuir [4], from the perspective of games and learning. They argue that “*the model we apply to learning should depend on what is that we are trying to ensure people learning at any given time.*” These issues should be considered in reference to the Lecture Games project. For

instance, “who” is the new generation students, mentioned in Section 1.2; “what” depends on the course content; “where” points possibly to classrooms in schools where the lectures are delivered; “why” relates to a motivation to learn through games or game development.

Only those learning strategies are discussed here, which are relevant and were applied in this study to guide the game design process. In the study of topic 1, the ubiquitous technology and mobile devices provide the possibilities for the collaboration in both virtual world and real world. A multiplayer quiz-based game concept was chosen to match the collaborative learning features from a pedagogical point of view, and the specific games were designed based on the game design theory by Malone [29, 75]. Further, this game concept was extended into four games as a part of this PhD research, one running on stationary devices and the other three running on mobile devices, i.e. iPhone or Android smart phone. It provides various playing experiences in both real world and virtual worlds in order to show the impact of technology on the lecture games design. Finally, the evaluation framework based on game design theory and experimental software engineering was used to assess the game design and to direct improvements. In the study of topic 2, the design of GDBL was guided by the pedagogical concepts, e.g. double stimulus [76] and Project-based learning [77], combined with game design theory by Malone [29, 75]. This approach was applied in two quasi-experiments where XNA and Android SDK were used as GDFs in a software architecture course. The course assignment was a game development project designed to suit certain GDFs and course contents’ requirements, and students had to work together to finish the project. Finally, the methods were borrowed from experimental software engineering to design the experiment and to evaluate data in order to discover the differences between GDBL and non-GDBL. The following sections discuss in detail the theories used in this study.

There may be other theories or strategies, which could have been applied in the Lecture Games project. The chosen supportive theories and technologies appeared to be most relevant to demonstrating how to “use supportive theory and current computer technology as dual basis to facilitate lecture games in the current technology-rich environment.”

2.2.1 New approaches to collaboration for learning

Today’s students living in a technology-rich environment have changing preferences for education and work environments, which may negatively affect their focus on the traditional university course programs, e.g. enrolment and retention rates in these programs. One phenomenon is that students get used to collaboration with each other through social applications. To be more suitable and to improve such lecture’s educational situation, teaching methods and tools outside of the traditional lecture sessions and textbooks must be explored or implemented if needed. Currently, research on educational games and on collaborative classrooms benefit each other by focusing on this issue. The Lecture Games project deals with both educational games and students’ collaboration issues during playing games. “Literature review in games and learning” [4] demonstrates that a collaborative educational game has an advantage by increasing

learning gains and student engagement above that of individual learning game experiences. In addition, collaborative work in collaborative educational games may pose to course instructors, by helping to manage and evaluate student performance [78].

Further, in this underexplored area, the term “collaborative games for learning” is used to denote collaborative games combined with collaborative learning. Recently, studies on game design to support effective and engaging collaboration between students have been investigated by some computer-supported collaborative learning (CSCL) researchers in environments such as Second Life [79] and World of Warcraft [80, 81]. These studies reveal that the multiplayer games allow players to use in-game objects to create new activities collaboratively, and enhance social interaction in the games. These examples show how to create multi-player games, which effectively support collaboration between players.

Collaboration does not necessarily mean competition between teams [82]. In the real world, a goal which requires a collaborative process, like solving a problem, may create a conflict in the form of the interaction within the team cooperation [83], but it is not a contest amongst adversaries. The team has to cooperate to reach a common goal. Besides, the recent appearance of proper means of communication and interaction could easily support collaboration in computer games, but there are very few actual truly collaborative learning games on the market. Therefore, in the investigation of topic 1, this study uses collaborative learning as a tool to explore how students work closely together in a multiplayer quiz-based game.

2.2.2 Double stimulus as an element of activity theory

Another specific theory, which benefits the Lecture Games project, is double stimulation [76]. In schools, learners face a challenge, a problem, or a task, which was designed for a particular pedagogical purpose, or they face situations, which are likely to appear in work and public life. In these cases, exploiting tools can help learners to respond to such challenge/problems. Using these tools, learners can understand/solve the problems and better grasp the relevant knowledge. The construct of the relationship between the educational tasks and the material artifact is at the heart of Vygotsky’s notion of double stimulation [76], a method for studying cognitive processes and not just their effects. In a school setting, typically the first stimulus would be the problem, challenge, task, or assignment to which learners are expected to respond. The second stimulus would be the available mediating tools. However, it is important to note that Vygotsky described this relationship in dynamic terms, where the second stimulus is not a discrete end point for the process but, *“Rather, we simultaneously offer a second series of stimuli that have a special function. In this way we are able to study the process of accomplishing a task by the aid of specific auxiliary means.”* [76]. Note that Vygotsky identifies the second stimulus in the plural as a series. This is considered to be most important when providing the second stimulus in the form of digital tools [84].

The original “double stimulation” [76] experiment was conducted by Leontiev under Vygotsky’s supervision. Three age groups were presented with lists of words with the instruction to memorize the words. Each group was divided into two subgroups

corresponding to two experimental conditions. In one case, the words were the only stimuli presented. In the other case, the subjects were given a secondary set of stimuli, a stack of picture cards, which they could use as mnemonic tools. The results showed that among preschool children, the performance was rather poor and approximately at the same level in both cases. In middle-school children, the usage of cards resulted in a marked increase in performance level compared to the no-cards case. University students showed a high level of performance under both conditions and the difference between the cases was small. It was found that performance in each of two cases improved with age and that using cards as tools generally improved performance further. However, the difference between recalling words with or without cards has manifested differently in the three age groups (pages 44 in [85]).

In the research of topic 2, double stimulation was used to guide the design of GDBL in a higher education course. Specifically in this course, the first stimulus was a game development task the students had to undertake. The second stimulus was the available mediating tool, e.g. GDFs in GDBL. In the software architecture course, the second stimulus was the development kit, XNA or Android SDK. Although the Leontiev's 1978 double stimulation experiment shows small differences for university students, after decades, it can be argued that today's new generation of students and technology could affect the results more than before. For instance, if a GDF with no programming requirements is provided a child is motivated to learn literacy [2], which is different from Leontiev's experiment results. This is an evidence of changes in the new generation of students and development of technology. For this reason, different outcomes for university students can be expected in the current technology-rich environment. The evaluation data of GDBL and non-GDBL were compared to find the differences between them. The important aspect of this comparison was to discover the extent of the differences, and to determine whether GDBL showed a positive effect even for small differences. It was also investigated whether the differences were affected by the kind of tools chosen for a specific course. Using the results of this evaluation can assist in finding optimal GDFs to maximize the desired differences. Even if the difference is small, such an approach can improve the learning quality.

2.2.3 Project-based learning

Project-based learning is an approach to classroom activity emphasizing learning strategies, which are long-term, interdisciplinary, and student-centered. If a class applied the project-based learning, students usually organize their own project work and manage their own time. This makes classroom activities less structured than traditional, teacher-led classroom education.

Definitions of Project-based learning include features relating to the use of an authentic ("driving") question, a community of inquiry, and the use of cognitive (technology-based) tools [86, 87]. Expeditionary Learning adds features of comprehensive school improvement, community service, and multidisciplinary themes [88].

To be considered an instance of Project-based learning, a project should have the following characteristics [77]:

- Be central, not peripheral to the curriculum.
- Be focused on questions or problems that “drive” students to encounter (and struggle with) central concepts and principles of a discipline.
- Involve students in a constructive investigation.
- Be student-driven to some significant degree.
- Be realistic, not like schoolwork.

Project-based collaborative learning [89] is a further step to integrate the team element into the project work, and it emphasizes the factor of collaboration in a team during a learning process. The classroom settings combine the methods used to organize a collaborative learning group and a process of producing technology-related projects with instructional methodology. It also provides an analysis of the artifacts produced by the *pre-service* teachers as well as the feedback from the students and the *in-service* teachers involved in the project. Within the Project-based learning approach, students collaborate to make sense of their environment and achieve a realistic goal.

In the study of topic 2, the double stimulus and Project-based learning methods were combined in the GDBL design. The first stimulus is not a single problem, but a series of problems organized as a Project-based learning project, which needs four to five students to work together to complete. In view of this, Project-based learning is also an outcome of Problem-based learning [90], used in the design of GDBL [48, 89], but not, to a large degree, in this study.

Theories and strategies, mentioned in Section 2.2, were the foundation for the design of the Lecture Games project, but, in general, there are more than just three strategies to support the design of lecture games. Here, they are used as examples to show how to design a lecture game activity from a learning perspective. Further, the “New approaches to collaboration for learning” presented in Section 2.2.1 are mainly a guide to the study of topic 1. The double stimulus and Project-based learning presented in Section 2.2.2 and 2.2.3 are used as the theoretical context for the study of topic 2.

2.3 Game Design Theory for Lecture Games

In addition to the above theories for guiding the design of this study from the pedagogical perspective, inclusion of another field can be beneficial, namely game design theory. Since there are no systematic criteria to guide the GBL, this section discusses the following theories and methods that can direct the design of this research: GameFlow model, intrinsic motivation theory, and other related theories.

2.3.1 GameFlow model

The GameFlow model [91] of enjoyment in games was constructed based on the literature on the elements of flow and the evidence of flow experiences in games. Flow is an experience “*so gratifying that people are willing to do it for its own sake, with little concern for what they will get out of it, even when it is difficult or dangerous*” [92]. Csikszentmihalyi [92] conducted extensive research into what makes experiences

enjoyable, based on long interviews, questionnaires, and other data collected over twelve years from several thousand respondents, and he formulated seven elements in the flow theory. The GameFlow model consists of eight core elements: concentration, challenge, skills, control, clear goals, feedback, immersion, and social interaction. Each element includes a varying number of criteria, which relate to Csikszentmihalyi's elements of flow theory, as shown in Table 5, except for the social interaction added as a result of the game literature review.

Table 5: Mapping the elements in games literature to the elements of flow[91]

Game literature	Flow
<i>The Game</i>	A task to be completed
<i>Concentration</i>	Ability to concentrate on the task
<i>Challenge Player skills</i>	Perceived skills should match challenges and both must exceed a certain threshold
<i>Control</i>	Allowed to exercise a sense of control over actions
<i>Clear goals</i>	The task has clear goals
<i>Feedback</i>	The task provides immediate feedback
<i>Immersion</i>	Deep but effortless involvement, reduced concern for self and sense of time
<i>Social Interaction</i>	N/A

Sweetser [91] provided further explanation for each element of the GameFlow model. 1) Concentration - Games should require concentration and the player should be able to concentrate on the game. 2) Challenge - Games should be sufficiently challenging and match the player's skill level. 3) Player Skills - Games must support players' skill development and mastery. 4) Control - Players should feel a sense of control over their actions in the game. 5) Clear Goals - Games should provide the player with clear goals at appropriate times. 6) Feedback - Players must receive appropriate feedback at appropriate times. 7) Immersion - Players should experience deep but effortless involvement in the game. 8) Social Interaction - Games should support and create opportunities for social interaction. As described, the purpose of the GameFlow criteria is to build an understanding of enjoyment in games. In their current form, the criteria are not meant to be used as an evaluation tool for game developers. However, the expert evaluations [91] showed that the criteria are a useful tool for reviewing games and identifying issues, as well as the effect of these issues on player enjoyment. In addition, the criteria were used to develop a solid understanding of what constitutes good design and player enjoyment in real-time strategy games.

The GameFlow model was used as the main validating framework in the research of topic 1. It has several criteria to assess and review different aspects of the games. Typically, it can also serve as the design criteria to be used in advance for the game design. Further, an extended EGameFlow Scale [93] based on GameFlow model has one more element - Knowledge. The attributes of this new element reflect whether the game increases knowledge, whether the player can acquire the fundamentals of the knowledge taught, and whether the player wants to expand this knowledge. This is an extra criterion for the design and evaluation of an educational game. EGameFlow scale was used as an evaluation framework in one of case studies for topic 1.

2.3.2 Motivational theory and intrinsic motivation theory for games

The existing literature reviews reveals that there exist two opinions on the source of video games motivation. One side is the compelling nature of games to their narrative context [55, 57, 94, 95], while another side emphasis that motivation is implanted into game play's goals and rewards as intrinsic [96-98]. However, both sides agree that motivation to play is one of the important attributes of serious games and that it already brings effects on the educational game design. The motivation aspect is of particular interest to educational researchers because of the crucial role it plays in student learning. Motivational models are central to game design, because without motivation a player will not be interested in progressing further within a game [99]. Several models for game play motivation have been proposed, including one by Richard Bartle [100]. In addition, Jon Radoff proposed a four-quadrant model of game play motivation that includes cooperation, competition, immersion, and achievement [101]. Generally, the category of motivation is conceptualized as either intrinsic or extrinsic [102] usually expressed as intrinsic motivation and extrinsic motivation [103]. Following is a short description: *Intrinsic motivation* is an internal feeling, like bringing pleasure, importance or other things that motivate people to do something. *Extrinsic motivation* happens when people is compelled to do something due to external factors, e.g. good grades or prizes.

This project in an attempt to explore the intrinsic motivation related to the educational games field. According to the literature survey, one of the most valuable and classic theories for GBL stems from Malone's work. Malone and Lepper defined a taxonomy for learning using intrinsic motivation as a factor for encouraging learners' engagement [29]. Intrinsic motivation is defined more simply in terms of what people will do without an external inducement [29]. It is an activity for its own sake rather than to receive external rewards or avoid punishment. Such activities are engaging simply by being interesting, captivating, and/or enjoyable. Intrinsically motivating activities are those in which people will engage for no reward other than the interest and enjoyment, which accompanies them. Malone and Lepper integrated a large amount of research on motivational theory by means of a synthesis of approaches to design intrinsically motivating environments. They characterized them by seven factors: challenge, curiosity, control, fantasy, competition, cooperation, and recognition.

Further, as an extension of Malone's proposal, Nicoletta and Kelly [106] defined a set of game design criteria which are likely to promote user's interest, enjoyment, and learning. The elements, adapted in this study, are summarized in Table 6 in a pragmatic way. The features are derived from the elements of intrinsic motivation identified by Malone and Lepper [75, 107].

Table 6: Game elements for design adapted from Malone and Lepper

ID	Game elements which may promote engagement, motivation, and fun	Reference
1	A shared, imaginary story context which establishes and supports the activities	[108, 109]
2	An overarching goal	[75, 108, 110]
3	A gentle on challenge	[75, 108, 110]
4	Multiple levels with variable difficulty	[75, 111]
5	Uncertain outcomes	[75, 108, 110]
6	Various ways to win	[109]
7	A well defined advancement system	[75, 108, 109]
8	Rewards associated with advancement	[75, 108, 109, 112]
9	Opportunities to build new content	[108, 109, 113]
10	Ability to progress at the user's own rate	[109, 110]
11	Hints not answers	[110]

In this project, we mainly refer to the list in Table 6. The intrinsic motivation is used mostly in the research of topic 1 to guide the design of quiz game interface and content, as described in Section 5.2. Moreover, it is used as a reference in the selection of GDFs for the research of topic 2, described in detail in Section 5.3.

2.3.3 Other criteria: More than game features

In addition to the game design theories described above, there are other methods supporting educational game design. One example is by Nicoletta and Kelly, who used color psychology to guide the design of an educational game's interface [106] - *"The choice of the color and lighting schemes was based on research studies on the impact of color and light on learning, and on the association between colors and children's emotions. One study shows that de-saturated colors have a negative impact on stimulation while highly saturated colors increase alpha waves in the brain which are directly linked to awareness."* In another research field related to games, 3D virtual environment for learning, the 'exploratory learning model' (ELM) extends Kolb's experiential learning model by adopting the use of 3D applications. Examples of research and development projects are provided to demonstrate how the model works in practice [114].

The study of topic 1 included the design of an educational MMORPG game. This design was supported by the theories of the EGameFlow model and Intrinsic Motivation, as well as a literature survey on current popular elements in MMORPGs. These elements were then considered for an educational MMORPG game. In the study of topic 2, "Gamification" concept described in the Section 1.2 gave inspiration for using one of game elements, Game Development, in the education. Further, a software engineering course was adapted and implemented as an experiment using the GDBL method. There might be duplicating or conflicting aspects in these theories when they

are combined in a single application. If this is the case, it is necessary to make trade-offs between these theories during the design process.

2.4 Experimental Software Engineering

Experimental software engineering was useful for evaluating results in this study, and thus it became a part of the framework for evaluating GDBL as well as contributing to other evaluation processes for GBL. According to Claes Wohlin's "Experimentation in software engineering: an introduction" [115], experiments are appropriate to investigate different aspects, including:

- Confirming theory, i.e. to test existing theories.
- Confirming conventional wisdom, i.e. to test people's conception.
- Exploring relationships, i.e. to test that a certain relationship holds.
- Evaluating the accuracy of models, i.e. to test that the accuracy of certain models is as expected.
- Validating measures, i.e. to ensure that a measurement actually measures what it is supposed to.

Experimental software engineering is a valuable method for all software engineers who are involved in evaluating and choosing between different methods, techniques, languages, and tools. In the Lecture Games project, experimental software engineering methods were helpful in validating the emerging theory and conclusions through experiments. Such methods are mainly used in the research of topic 2 and, to some extent, in the study of topic 1. In addition, Claes Wohlin defines five steps for carrying out an experiment to be [115]:

- *Experiment definition*: This step helps to identify the object of study, purpose, quality focus, perspective view, and context.
- *Experiment plan*: This step is the foundation for the experiment. It includes the environment of the experiment, as well as inputs and output of the experiment. The subjects and instrumentation are carefully defined in the plan.
- *Experiment operation*: Preparation, execution, and data validation are the basic three steps in this period. The preparation step is concerned with preparing the subjects and material needed. The main concern in execution is to ensure that the experiment is conducted according to the plan and the design of the experiment. Finally, it has to be ensured that the data actually collected is accurate and provides a valid picture of the experiment.
- *Analysis and interpretation*: After data collection, descriptive statistics are used to gain insight into their meaning. A hypothesis test can also be applied here. The focus of interpretation should be based on the analysis and testing of results.
- *Presentation and package*: To describe primarily results, a research paper or a technical report is usually needed.

In the Lecture Games project, the evaluation of the topic 2 for this study was conducted through experiments. Since the major part of the development of a game is software

engineering, a lecture game itself is the result of software development. The concepts of experimental software engineering can be borrowed to provide guidelines and references for the evaluation of the Lecture Games project.

In addition, during the experimentation in software engineering, different empirical strategies can be applied in a software engineering context, i.e. Quality Improvement Paradigm (QIP) and Goal/Question/Metric paradigm (GQM). In this project, the GQM was chosen for the experiment design. The GQM approach [116] specifies a measurement model targeting a particular set of issues and a set of rules for the interpretation of the measured data. It defines a practical goal on a conceptual level, a set of research questions on an operational level, and a set of metrics to answer the defined research questions on a quantitative level. Additionally, the System Usability Scale (SUS) [117] was used in the experiment to measure the game usability, since usability and enjoyment of a game are two closely related concepts. SUS is a usability questionnaire consisting of ten generic Likert scale items. Responses to the questionnaire result in a score, called the SUS score, a single number between 0 and 100 indicating the overall usability of the system being studied. SUS has ten questions, listed in Table 7. Each question has a scale from 1 to 5. For items 1, 3, 5, 7, and 9, the total score is calculated by adding the points. For items 2, 4, 6, 8, and 10, the total score is calculated by subtracting the points from 5, and adding the difference. This implies that each question can contribute from zero to four points to SUS. Finally, the sum of the scores is multiplied by 2.5 and divided by the number of replies to obtain the SUS score. These questions can be used to test the games or tools and to get a final score to measure their usability. The higher score denotes the higher usability.

Table 7: The questionnaire in SUS

ID	Questions
1	I think that I would like to use this system frequently.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I think that I would need the support of a technical person to be able to use this system.
5	I found the various functions in this system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
9	I felt very confident using the system.
10	I needed to learn many things before I could get going with this system.

2.5 Enabling Technology

The contemporary computer technology improves many aspects of human life and communication, but it also affects the society in a negative way, and its impact on learning depends on how it is used.

2.5.1 Prerequisites – the lecture environment

In this research, the lecture hall is the main environment for using lecture games. Today, most universities provide a complete Wi-Fi coverage over the whole campus including the lecture halls, and most of the larger rooms have video projectors. This fulfills the basic requirement for using ubiquitous technology in the lecture games.

In addition, the use of smart phones and mp3-players has become a part of everyday life. In the early stage of the project, GSM networks with limited bandwidth and mobile phones with small screens were used as peripherals for the lecture games. Recently, iOS and Google Android platforms opened the possibility of utilizing mobile games on the mobile devices for lectures using Wi-Fi, 3G, or 4G. These attractive peripherals can fascinate new generation students and open new ways to learning.

2.5.2 Mobile technology - games as a motivation for lectures

Computer game technology has been developed for decades. Now, games can be played on various devices, such as personal computer, mobile devices, and game consoles. Mobile devices are most convenient for educational game play during the lectures. This section contains a brief introduction to relevant hardware and software for games utilizing mobile technology.

Since the computer was invented in mid-20th century (1940 - 1945)¹³, it has become the dominating device for electronic game playing. In 1979, the first handheld game console, Microvision¹⁴, was released, and it opened the door to mobile gaming. Through a successful launch of Game Boy by Nintendo in 1989, the mobile games became as popular as non-mobile games. In the 21st century, there are several choices of platform for the same game with a Windows version, a PSP version, a Nintendo DS version, and an iPhone version available. The following paragraphs discuss hardware related to lecture games and the corresponding software behind the game play.

The hardware supporting game play can be classified as stationary devices and mobile devices. Most of the common stationary devices are desktop computers and game consoles, such as Microsoft XBOX series, Sony Playstation series, and Nintendo series. Game companies, like Ubisoft, usually release games for multiple platforms. Mobile devices include laptops, smart phones, tablets (e.g. iPod), mp3-players (e.g. iPod touch), and mobile game consoles, such as Sony Portable Playstation (PSP), and Nintendo DS. The game play can be experienced either with full functionality on stationary devices or usually in a more limited version on portable mobile devices.

In the Lecture Games project, especially in the research of topic 1: “Games as a motivation for lectures”, the focus was mainly on the features of software on mobile devices, the most relevant technology for lecture games. Students usually bring their laptops and smart phones to lectures. The most fundamental level of software, the

¹³ <http://www.thepcmuseum.net/timeline.php>

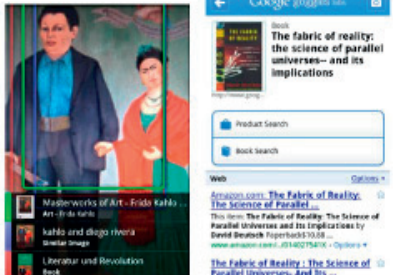

¹⁴ <http://www.engadget.com/2006/03/03/a-brief-history-of-handheld-video-games/>

Operating System (OS), is the first element to be taken into consideration. According to the survey of popular Ross in the recent mobile device market, Mac OS and Windows are typically used for laptops, while iOS, Android OS, Symbian OS, and Windows Mobile OS are used in most smart phones, tablets, and mp3-players. In addition, the smart phones are used every day as the major tool and they became the most convenient and versatile device for students to bring to school. Based on analysis of devices used by students, the most common OS at NTNU in early 2008, when this study started, was Symbian OS. Today, most students use devices running iOS and Android OS.

In this study, we chose these two common OSs, iOS and Android OS, to develop games to be used as an aid for lectures. In most smart phones, the most relevant features for the type of games to be developed are GPS, camera, Wi-Fi, Near Field Communication¹⁵, gyroscope, accelerometer, and digital compass.

Apart from the above features offered by recent mobile devices, the ideas in lecture games were inspired by widgets and apps already available on these devices. An ongoing survey of iOS and Android OS platforms was conducted to find applications with a potential to enrich game play experiences in the study of topic 1. Table 8 lists some collected examples.

Table 8: Examples of apps used to provide game play


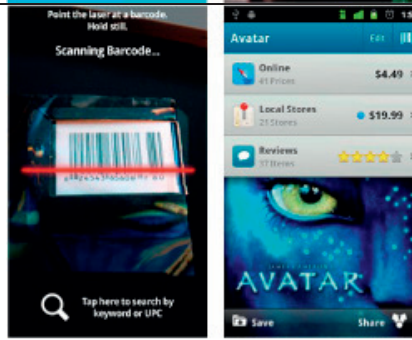
Name	Description	Interface
Google Goggles ¹⁶	A free image recognition application created by Google. Google Goggles enables the user to use photos taken with the mobile phone to search the web. This can be text, landmarks, books, contact info, artwork, wine, and logos ¹⁷ . Google Goggles currently does not work well with pictures of animals, plants, cars, furniture, or apparel.	
Layar ¹⁸	A mobile platform for discovering information about the world around us by using Augmented Reality technology. Layar displays digital information by using the camera of mobile phones. Layar supports a high level of interactivity, which includes audio and visual elements, 3D models, and social sharing capabilities.	

¹⁵ <http://www.nearfieldcommunication.org/>

¹⁶ <http://www.google.com/support/mobile/bin/answer.py?hl=en&answer=166331>

¹⁷ <http://www.google.com/mobile/goggles/#text>

¹⁸ <http://www.layar.com>

<p>Shazam¹⁹</p>	<p>An application for recognizing songs being played, for instance on the radio. The application receives music snippets through the microphone on the phone, and creates a fingerprint of the snippet based on a spectrogram, which it matches against fingerprints stored in a central database of music.</p>	
<p>ShopSavvy²⁰</p>	<p>An application for reading barcodes of products using the camera of the mobile phone. After reading the barcode, the application will identify the product and provide a list of online and local suppliers and prices.</p>	

Most of the applications in Table 8 combine more than two features of smart phones, e.g. GPS, camera, Wifi, gyroscope, and the digital compass. All of these features open exciting possibilities for the Lecture Games project.

2.5.3 Game development frameworks - game development as a motivation for lectures

This section introduces the relevant software used in the study of topic 2: “Game development as a motivation for lectures”.

As mentioned before, GDFs are the game development tools used for GDBL. They are the major tools supporting the study of topic 2. GDFs encompass the toolkits used to develop or modify games, such as game engines, game editors, game (simulation) platforms, or even any Integrated Development Environment (IDE), for example Visual C++, Eclipse, J2ME, or Android SDK, since any of them can be used to build games. GDFs are used in student exercises to learn skills, extending GDFs as a teaching aid. The motivation for teaching through game development is to utilize the students’ enthusiasm for games and creativity for learning. The GDBL method did not appear recently. The earliest similar application of learning by programming in a game-like environment took place in the early 1970s. The Logo [118], the turtle graphics, is one of the oldest libraries which was used to introduce computing concepts to beginners. The

¹⁹ <http://everythingelsematterstoo.blogspot.com/2010/11/how-shazam-works.html>.

²⁰ <https://market.android.com/details?id=com.biggu.shopsavvy>

concept was based on a “turtle” moved across a 2D screen with a pen, which could be positioned on or off the screen, and thus, leave a trace of the turtle’s movements. Programming the turtle to draw different patterns could be used to introduce general programming concepts, such as procedural operations, iteration, and recursion. Further, in 1987, Micco reported the experience of writing a tic-tac-toe game for learning [119].

The GDFs in the survey conducted for this study were classified as (a) *Game engines*: This category covers the commercial game engines as well as mature and well-known toolkits used to create games. (b) *Self-made GDF*: This category includes the game development frameworks created by researchers for usage in a specific course. (c) *Games or game editors*: This category contains editors or platforms, which can be used to modify games. (d) *Simulation platforms*: This category includes controllers to create a game-like system on a robotic or other simulation platform. (e) *Hardware platforms*: This category includes both game hardware and related software on this hardware to build games (laptops and computers are excluded), such as Wii remotes, Windows surface with XNA, and robotic hand. (f) *Others are general IDEs*, such as Visual C++, J2ME, or unspecified software creation toolkits, not specifically designed for game development.

From the perspective of application in learning, GDFs can be classified into *GDFs for novices* and *GDFs for developers*. The main focus of GDFs for novices, including non-programmers, is to provide visual methods for customizing game templates and to allow creation or design of games with little or no programming skills. The main focus of GDFs for developers is to offer toolkits which support development of high quality 2D/3D rendering, special effects, physics, animations, sound playback, and network communication in common programming languages, such as C++, C#, and Java. A list of some examples found by the literature survey is given in Tables 9 and 10 to show GDFs for novices and GDFs for developers in the context of GDBL. In the study of topic 2, XNA and Android SDK were chosen as GDFs in the software architecture course.

Table 9: GDFs for novices

GDFs	Features Description
Alice (http://alice.org)	Alice provides a point-and-click programming interface allowing creation of simple 3D games and animations. It is a tool for teaching object-oriented programming.
CeeBot Series (http://www.ceebot.com/ceebot/family-e.php)	The programming language in CeeBot is very similar to Java, C++, and C#. It was developed especially to make learning programming easier. “CeeBots4 School” is a programming course for middle and high school.
Scratch (http://scratch.mit.edu)	Scratch provides a point-and-click programming interface to create media-rich games, animations, and applications for the web. Scratch is suitable for teaching children basic programming (variables, arrays, logic, and user interface), and for creating simple 2D “quick-and-dirty” applications.
Greenfoot (www.greenfoot.o)	Greenfoot is a solid tool, which provides many of the constructs needed for creating 2D computer games at a level,

rg)	which is especially appropriate and enjoyable for novice programmers.
Maya/ Photoshop/Flash	These software products are mainly used for art design to create digital characters and animations for games. Flash can also create Flash-games.
Game maker (www.gamemaker.nl)	Game Maker is a rapid application development tool for young people at home and in schools to create two-dimensional and isometric games.
StarLogo TNG	StarLogo TNG is designed upon the basic framework of Logo. The programming is done with programming blocks instead of text commands, which moves programming from abstract to visual.
Game editor: Warcraft3 Editors/ NeverWinter Night toolsets	The editor provides a simple GUI for customizing game templates, and requires little or no programming skills to create interesting game designs. The editors are implemented as visual programming tools, which allow users to customize game behavior visually, including character behavior, game map, and game play.
Game platforms: Bomberman /Wu's Castle/ Critical Mass board game/quiz- based web game shell	These are individual games, but they provide visual interface for the users to modify or add basic code to change the game scenarios.

Table 10: GDFs for developers

GDFs	Features Description
FPS game engine: Torque game engine /Unreal Engine	These are the original commercial game engines already applied in popular commercial games. They are usually not free and provide some editing tools. They are more complex than an individual game editor.
XNA (www.xna.com)/ XNACS1Lib framework/ XQUEST/ BiMIP	These are game development tools based on MFC and DirectX on Windows platform and they have the same structure as the game loop concept. BiMIP is a self-made GDF similar to XNA developed by researchers. XNA is a GDF to develop cross-platform games for the Windows PC, Windows mobile phone, XBOX, and the Zune platform using C#. XNA features a set of high-level APIs targeted for 2D and 3D games. It consists of an IDE along with several tools for managing audio and graphics. XQUEST and XNACS1Lib are game libraries for XNA, which contain convenient game components.
Android/Sheep (www.android.com)	The Android mobile platform is an operating system issued by Google. The Sheep framework is an extended game library for Android.
Simulation	These are self-made simulation games or simulators, which

platforms: Spacewar simulator/ RoboRally/ JGOMAS MUPPETS/ SIMPLE framework	provide users with the controller to modify the parameters and control the avatar in the simulation platforms. They are used to teach programming and AI.
---	---

2.6 Summary

GBL is a relatively new research area compared to other traditional research fields. This means that there are more challenges related to its theoretical underpinnings. Due to lack of sufficient theoretical foundation, GBL design is usually directed by researchers' own experiences. This study indicates that GBL is inter-disciplinary in nature and could benefit from relevant research in mature fields such as education. In relation to the Lecture Game project, Chapter 2 investigated some classical theories, which can benefit game design. Further, related work in the fields of psychology and software engineering was reviewed to enrich the supportive theory and to become a resource pool for this research. Another area, investigated in this study, covered current popular devices and features of the technologies, which formed a base to construct the game concepts. The following chapters will deal with the challenges of selecting related theories and enabling technology to design and evaluate lecture games, and to enrich the theoretical base for both lecture games and GBL.

3 Research Methods

This chapter presents the philosophical view of the research method chosen for this study. First, the research design is briefly introduced. Second, data collection methods are discussed according to the corresponding research design. Finally, various data analysis methods are given.

3.1 Research Design

Colin Robson proposed a framework for research design (page 83 in [25]). The components are purpose(s), theory, research questions, methods, and sampling strategy, as showed in Figure 3. The research design turns research questions into a project.

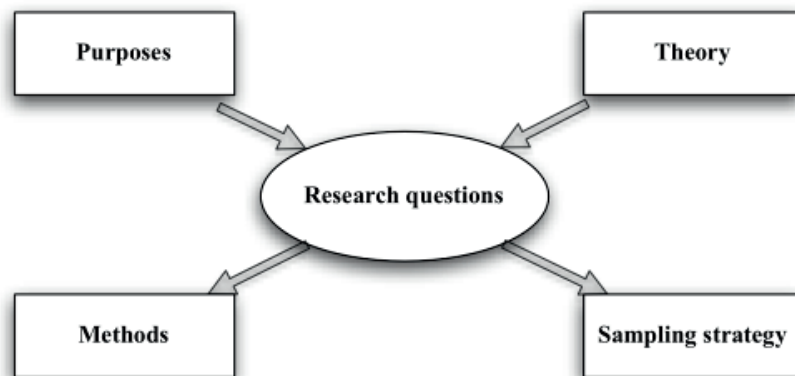


Figure 3: Framework of research design

As stated by Colin Robson [25]: “a good design framework has high compatibility among purposes, theory, research questions, methods and sampling strategy: 1) If the only research questions that we can get answers are not directly relevant to the purposes of the study, then something has to change - probably the research questions 2) If our research questions do not link to theory or it is unlikely that we will produce answers of value. In this case, theory needs developing or the research questions need changing. 3) If the method and/or the sampling strategy are not providing answers to

the research questions, something should change. Collect additional data, extend the sampling or cut down on or modify the research questions.”

A study design based on experiments and surveys is classified by Anastas and MacDonald [121] as a “fixed research design”. However, other research approaches are also recognized. For instance, most of researchers are strong advocates for qualitative design. It has many forms and arises from a variety of theoretical positions [122]. Anastas and MacDonald [121] refer to such designs as flexible. The two terms, “qualitative” and “flexible”, describe important features of such designs. Usually, *“flexible designs can include the collection of quantitative data. Fixed designs rarely include qualitative data (but could do)”* [25].

Apart from fixed design and flexible design, a *literature review* is also used in this research project. Conducting a literature review is a means of showing author’s knowledge about a particular field of study, including phenomena, history, development, theoretical basis, its key issues, and possible solutions. In addition, it also informs other interested researchers and research groups in the field. Finally, with some modification, the literature review is a *“legitimate and publishable scholarly document”* [123].

3.1.1 Fixed design

Fixed designs are usually concerned with aggregates, group properties, and general tendencies [25]. Traditional fixed design research strategies include both experimental strategy and non-experimental strategy [25].

The central feature of experimental strategy is that *“the research actively and deliberately introduces some form of change in the situation, circumstances or experience of participants with a view to produce a resultant change in their behaviors (skills, opinions)”* [25]. Usually, it is a measurement of the effects of manipulating one or more variables on another variable. The plans and preparations should be complete before the experiment begins. Quasi-experimental strategy shares many similarities with the experimental strategy, but it specifically lacks the element of random assignment. Instead, quasi-experimental strategy typically allows the researcher to control the assignment.

The overall approach of non-experimental strategy is similar to experimental strategy but the *“research does not attempt to change the situation, circumstances or experience of the participants”* [25]. It also requires that the details of preparation are fully specified before experiment begins.

3.1.2 Flexible design

Three traditional flexible design research strategies are case study, ethnographic study, and grounded theory study. Colin Robson explains them as follows [25]:

Case study is to develop detailed, intensive knowledge about a single “case”, or of a small number of related “cases”. This approach typically involves data collection and analysis.

Ethnographic study seeks to capture, interpret, and explain how a group, organization, or community lives, experiences, and makes sense of their lives and their world. It typically tries to answer questions about specific groups of people, or about specific aspects of the life of a particular group.

Grounded theory study is aimed at generating theory from data collected during the study. It is particularly useful in new, applied areas where there is no theory and concepts to describe and explain the observed phenomena. Data collection, analysis, and theory development and testing are interspersed throughout the study.

3.1.3 Literature review

In addition to the above research designs, a literature review was used independently as a research method in this project. Cooper suggests conducting a literature review in the following steps [124]:

- (1) Problem formulation
- (2) Data collection
- (3) Data evaluation
- (4) Analysis and interpretation
- (5) Public presentation

He also emphasizes the key components and strategies - “(a) a rationale for conducting the review; (b) research questions or hypotheses that guide the research; (c) an explicit plan for collecting data, including how units will be chosen; (d) an explicit plan for analyzing data; and (e) a plan for presenting data. Instead of human participants, for example, the units in a literature review are the articles that are reviewed” [124].

Further, a systematic literature review aims to provide an exhaustive summary of literature relevant to a defined field. It has more strict requirements for the number of the bibliographies in a field than a literature review, and it provides a relatively complete collection of information in a current research field.

3.2 Selection of Methods for Data Collection

The selection of methods is based on the kind of information sought, from whom and under what circumstances. It is decided at an early stage in a fixed design project. In flexible design projects, it is better to make some initial decisions on the methods to collect data, but these can be changed as the data collection progresses. Colin Robson [25] provides simple rules for selecting methods for both fixed designs and flexible designs: “1) To find out what people do in public use direct observation. 2) To find out what they do in private, use interviews or questionnaire. 3) To find out what they think,

feel and/or believe, use interviews, questionnaires or attitude scales. 4) To determine their abilities, or measure their intelligence or personality, use standardized tests.”

When using a literature review, the scope of the literature survey has to be decided, i.e. how much time and how much effort is to be expended on the search of a bibliography, especially considering the filtering workload with a large number of bibliographies, necessary to collect the most useful data.

3.3 Dealing with the Data

After identifying the methods to collect the data, the next step is to collect the data and prepare for the data analysis.

3.3.1 Collecting data

According to Colin Robson [25] there is no generally “best method” for data collection, and he states: “*We should do it properly using these methods in a systematic, professional fashion.*” The selection of methods depends on the research questions to be answered. This has to be adapted to what is feasible, in terms of available time and other resources or to the skills and expertise of the researcher.

3.3.2 Preparing for analysis

All data come in different forms, including sets of instrument readings or test results, responses to questionnaires, diary entries, reports of meetings, documents, and possibly audios or videos, etc. Many of them are either words or numbers. Qualitative analysis is used for words or other data, which come in a non-numerical form, and quantitative analysis for numbers.

3.3.3 Quantitative analysis

There are many tools for carrying out quantitative analysis, and it would be impossible to expect everyone conducting an enquiry to use all of them. There is a tendency to gain some familiarity with a narrow range of approaches and then be inclined to use them. One suggestion is to get advice from a consultant or other persons familiar with a wide range of approaches to the quantitative analysis of research data if possible. In addition, one can use the most popular software package for statistical analysis - SPSS (the Statistical Package for Social Science). However, for simple statistical data, such specialized software may not be required, and spreadsheet software such as Microsoft Excel can be used to perform a range of statistical tasks [125].

Descriptive statistical results can be obtained with the help of SPSS or Excel. These software products provide different ways of representing some important aspects of a dataset by a single number. The statistics can then be used to discover major tendencies. Two most common aspects, dealt with in this way, are the level of the distribution and

its dispersion. Depending on the complexity of data and the research purpose, measures such as mean, median, variance, or distribution can be used to evaluate the results.

3.3.4 Qualitative analysis

Text is by far the most common form of qualitative data. There are different approaches to qualitative analysis, summarized by the author from [126] as follows:

Quasi-statistical approaches: They use word or phrase frequencies and inter-correlations as key methods of determining the relevant importance of terms and concepts, and they are typified by content analysis.

Template approaches: The key codes are determined either according to an a priori criterion or by the initial perusal of the data. These codes serve as a template for data analysis but may be changed as the analysis continues. These approaches are typified by matrix analysis.

Editing approaches: They are more interpretive and flexible than the above methods. They have no (or few) a priori codes. The codes are based on the researcher's interpretation of the meanings or patterns in the texts. These approaches are typified by grounded theory methods.

Immersion approaches: They are the least structured and most interpretive, emphasizing the researcher's insight, intuition, and creativity, and as such they are fluid and not systematized. These approaches are close to literary/artistic interpretation and connoisseurship.

All of these research designs and methods guided the selection of methods for this study. The next chapter will explain in detail how they were integrated in the research process.

4 Research Process

This chapter begins by briefly introducing the research goal, relations between research questions, and the author's papers. This project consists of two topics, and although both topics have the same ultimate goal, each has its own characteristics and research process. The research process presented in this chapter is based on research methods discussed in Chapter 3, described here separately for each topic. Mainly, three research methods were used: the case study based on flexible design, the quasi-experiment based on fixed design, and the systematic literature review.

4.1 Research Goal

The research goal was already presented in Chapter 1 as follows: "*use supportive theory and current computer technology as dual basis to facilitate lecture games in the current technology-rich environment*". According to this goal, the Lecture Game project was divided into two topics. These two topics are independent but have the same goal. This section reiterates and discusses the research questions for each topic and divides them into sub-research questions.

Topic 1: Game as a motivation for lectures:

RQ1: How can supportive theory be identified to guide the design and evaluation of lecture games?

- RQ1.1 What is supportive theory within the context of lecture games?
- RQ1.2 How can a relevant theoretical framework be set up and applied in the design and evaluation of lecture games?

RQ2: How can current relevant technology and appropriate peripherals be used to provide various play experiences in the new lecture games?

- RQ2.1. What is current technology and peripherals relevant to GBL?
- RQ2.2. How can these technologies be integrated into the Lecture Games project and evaluated?

Topic 2: Game development as a motivation for lectures:

RQ3: What is game development based learning (GDBL) and what are the researchers' views on GDBL?

- RQ3.1 What is the definition and scope of GDBL?
- RQ3.2 What current technology and game development frameworks (GDF) are involved in GDBL?
- RQ3.3. What is the history and current position of GDBL in other researchers' opinion?

RQ4: How can the GDBL approach be characterized in terms of supportive theory and current technology-rich environment?

- RQ4.1 How can the GDBL method be adopted in a software architecture course and what is the students' perception of the GDBL method?
- RQ4.2 How can the framework and criteria in GDBL be characterized in terms of this study and other researchers' views?

The history of video games as a research field is much shorter than other traditional mature sciences. The literature review revealed a lack of theoretical constructs or systematic research methods in GBL research. Referring to the Lecture Games project, a cross-disciplinary approach was used combining games and learning, and the relevant theories from both the game and learning fields were identified. In addition, it should be noted that the rapid development of computer technology (e.g. smart devices and wireless networks) has a profound impact on the life of new generation students, and changes the way they acquire information. This phenomenon opens avenues for a possible evolution of GBL. The Lecture Games project is a result of the recognition of this phenomenon. The research questions are meant as a motivation for devising new game concepts, which, in combination with supportive theories and current enabling technology, will result in improved and innovative lecture games.

In Chapter 3, research methods were discussed at a high level of abstraction. Specific research methods, selected for this PhD work, are shown in Table 11. In the research of topic 1 (game as a motivation for lectures), the following case studies were carried out: "World of Wisdom" (WoW), "Lecture Quiz" (LQ), "Knowledge War" (KW), and "Amazing City Race" (ACG). Topic 2 (game development as a motivation for lectures), comprised 1) two GDBL quasi-experiments using XNA and Android SDK as GDFs in a software architecture course, and 2) a systematic GDBL literature review. The connections between the studies and the research questions are also revisited. In this section, the research design and data analysis process in the case studies, the experiments, and the literature review are described in detail, while the final results are presented in the next chapter.

Table 11: Relationship of entities in research process

Topic	RQ	Research Methods	Study	Examples	Papers
Game as a motivation for lectures	RQ1	Case study	Case 1	WoW	G1
			Case 2	LQ	G2
	RQ2	Case study	Case 3	KW	G3
			Case 4	ACG	G4
Game development as a motivation for lectures	RQ3	Literature review	-	-	GDF1, 2
	RQ4	Quasi-Experiment	Experiment 1	XNA	GDF3, 4, 5, 6
			Experiment 2	Android SDK	GDF7, 8

Based on Table 11, the following sections discuss which research methods were chosen, why they were chosen, and how they were applied in practice for the two topics.

4.2 Game as a Motivation for Lectures - Case Study

As mentioned before, questions RQ1 and RQ2 belong to topic 1 - game as a motivation for lectures. Their purpose is to identify the effectiveness of study of topic 1, and to establish how the supportive theory and enabling technology influenced the co-design of the project. According to three traditional flexible design research strategies, the case study was chosen for research of topic 1. Features of the case study match requirements of this research in terms of [25]: 1) Selection of a single case (or a small number of related cases) of a situation, individual or group of interest, or concern. 2) Study of the case in its context. 3) Using a range of data collection techniques including observation, interview, and document analysis.

Figure 4 shows how the topic of game as a motivation for lectures was decomposed based on the framework of research design presented in Figure 3 in Section 3.1.

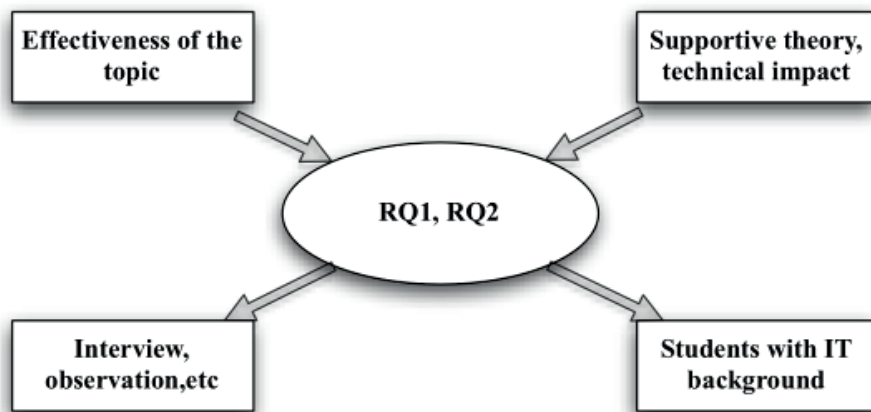


Figure 4: Mapping of research design framework into RQ1 and RQ2

The approach in this case study was to use the supportive theory to guide the design and evaluation in a technology-rich environment. The interviews, observations, and questionnaires are the methods for data collection. The four case studies were not limited to just a single concrete context of course content; this means that the cases are suitable to learn any subject. In this instance, we chose the students with computer science (CS) background since the challenges in the lecture games relate to the computer science field.

In addition, no systematic literature review of topic 1 was conducted because such reviews already exist [4, 128-130]. However, related works were surveyed and reviewed at the beginning of each case study. The case studies provided both quantitative and qualitative data.

The objective of this study was to seek answers to questions RQ1 and RQ2. The main purpose of this study was to identify critical success factors and design methods for games as a motivation for lectures. There were two case studies for each research question.

4.2.1 The case studies to answer questions RQ1 and RQ2

In order to answer question RQ1, two case studies, WoW and LQ, shown in Table 11, were carried out to explore the design issues and evaluation methods. These two case studies used supportive theories in the design and demonstrated how to apply them into the implementation and the evaluation of the game.

In order to answer question RQ2, two additional case studies were chosen, mainly to present the impact of technology on the lecture games. These two case studies introduced an interesting learning style using recent attractive peripherals, iPhone or Android smart phone.

Several data collection methods were used in order to get a wide range of data. The four case studies were carried out using the same steps, including background information investigation, procedures for the major tasks, questions, and reporting:

- i. Preparing the case studies for RQ1 and RQ2:
- ii. Participating in the development team meetings during the implementation.
- iii. Designing a questionnaire for users.
- iv. Conducting participatory observation of users during the game play.
- v. Conducting the user questionnaire.
- vi. Dealing with data and developers' documents.
- vii. Finishing reports and publishing the results.

- i. **Preparing case studies in the context of RQ1 and RQ2:** These research questions are concerned with the disciplines of game design and education. The game plot design in all four cases was based on theoretical support, technology

impact, and the developers' background. The four cases aim to answer questions RQ1 and RQ2. Further, each case study had its own focus: the case studies of WoW and LQ focus on the aspect of supportive theory construction, while the other two case studies, ACG and KW, emphasize the aspect of innovative game concept involving recent technology and peripherals. Specifically, WoW is an educational MMORPG game in which students can "play exercises" through a quiz in a virtual world. LQ is a multiplayer quiz game used in the classroom to let students review the course content and motivate the learning process in lecture. KW is a location-aware educational game for ions devices where students can play a quiz-based game to strengthen the social interactions at school. ACG is a pervasive game with an educational purpose based on the features of Android devices where the game plot is a competition to let players move around in a city and acquire and use knowledge such as city history.

- ii. **Participating in the development team meetings during the implementation:** At this stage the focus was on the implementation of educational games using the supportive theory and enabling technology. Mobile devices were used as the platform for development and testing. After the design was finished, the game developer teams usually met every two weeks with the supervisors (the author and his PhD supervisor). In these meetings, the group reported on their progress. At least one of the supervisors was present at all of the group meetings to follow closely the development process and to control the quality and progress of the project.
- iii. **Designing a questionnaire for users:** A set of questionnaires was designed, each with a specific purpose, in order to investigate the students' attitudes towards the lecture games. Some questions were more general, based on the supportive theory such as the GameFlow model, while others were designed specifically for each case.
- iv. **Conducting participatory observation of users during the game play:** The supervisors became observers and members of the observed group. During the game play, additional attention was paid to suitable/flexible game challenges and the balance of the entertainment and learning during the game play. Informal interviews, asking unobtrusive questions, and making notes were used to acquire relevant data for further analysis.
- v. **Questionnaire for users:** This step mainly involved giving the questionnaire to the individual/group players after they finished the game play. They were required to complete the questionnaire reporting their own playing experiences, and they were not allowed to discuss their experiences with each other during this phase.
- vi. **Dealing with data and developers' documents:** After collecting data from the above steps, the analysis and evaluation of the data in students' questionnaires and relevant records, namely observation notes and developers' documents, was conducted.
- vii. **Finishing reports and publishing the results:** The results of the case studies were published in the following four papers: G1, G2, G3, and G4.

4.2.2 Data collection methods

The following data collection methods were used in the case studies:

Self-completion questionnaire: The questionnaires were designed based on two sources. Firstly, the concepts were borrowed from the GameFlow model (including EGameFlow scale [93]), mentioned in Section 2.3.1, to measure the effectiveness of the educational games. This model uses a series of criteria to measure the enjoyment of video games, and it helps the game designer to understand the strengths and weaknesses of the game. EGameFlow has all the features of the GameFlow model with addition of knowledge improvement. Secondly, new questions were designed to investigate features and usefulness of certain game aspects pertinent to this study.

Scale: Usability of a software product and enjoyment of a game are two closely related concepts. According to the ISO 9241-11²¹ definition, usability is derived from three independent measures: efficiency, effectiveness, and user satisfaction.

- Effectiveness - The ability of users to complete tasks using the system, and the quality of the output of those tasks.
- Efficiency - The level of resource consumed in performing tasks.
- Satisfaction - Users' subjective reactions to using the system.

However, there are various methods to evaluate the usability. The method chosen in this study was the widely used System Usability Scale (SUS) [117] described Section 2.4, with a generic questionnaire of 10 questions for a simple indication of system usability represented by a number on a scale from 0 to 100 points.

Participant Observation: Project activities were closely observed by taking part in the project meetings and game testing. As a supervisor and manager in the project, the author took notes at the meeting and occasionally interacted with the project members as needed. Several documents, for example weekly reports or meeting minutes, were created by the project members. Further, the supervisor and the author carried out the evaluation of the results. During the game testing, the whole process was followed, photos were taken, videos of the game playing process were recorded, and players' activities were observed.

4.2.3 Data analysis

After collecting the data and filtering out the incomplete data, both quantitative data and qualitative data were available for analysis.

For the quantitative data analyzed with GameFlow Model and SUS, the required routines were used to identify the individual scores for these measurements. In addition, a general evaluation of students' attitudes towards the game's usability and engagement was performed. The scores were compared with other games using the same criteria for

²¹ http://www.iso.org/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16883

assessment. The weaknesses and strengths of the game itself could be identified by comparing the scores. For other quantitative data, Microsoft Excel and SPSS were used to identify the mean and variance for each individual score and to reach the conclusion.

For the qualitative data, for example data collected in the observation process, quasi-statistical approaches and template approaches, mentioned in Section 3.3.4, were mainly followed; the data was classified into categories and the implications interpreted.

4.3 Game Development as a Motivation for Lectures - Literature Review

During the study of topic 2, the term GDBL was proposed to define the scope and systemize the rules of this field. The research purpose was to find the theoretical context, research methods, the application process, and the evaluation criteria. It required following a detailed process for data analysis and theory generation in this very new field. In this context, the decision was taken to conduct a literature review to get an overview of the field and to find the useful data from bibliographies that can be extracted to generic guidelines for GDBL. The first step was a preliminary literature survey on the GDBL in the software engineering field. Further, a systematic literature review of all possible fields was performed. The following section discusses the systematic literature review, including the main results of the preliminary review.

4.3.1 The systematic literature review to answer RQ3

According to the framework of research design in Figure 3 in Section 3.1, the topic of game development as a motivation for lecture was decomposed in order to answer question RQ3, as shown in Figure 5.

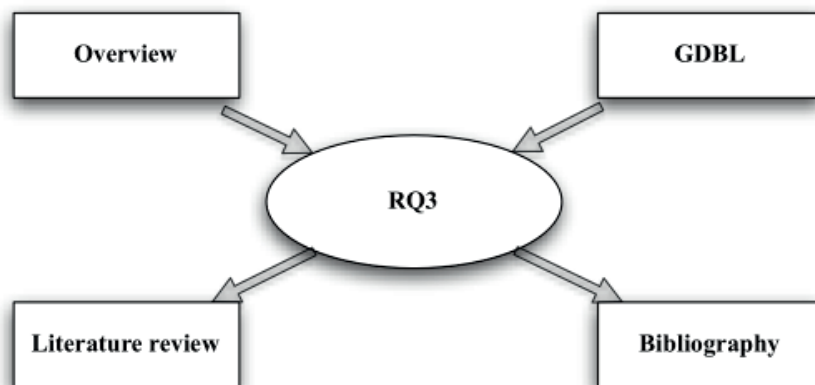


Figure 5: Mapping of research design framework for RQ3

In order to get an overview of GDBL and answers to RQ3, a systematic literature review of current main bibliographies was conducted, the collected data analyzed, and conclusions drawn.

Informed by the established method of systematic review [124, 131, 132], the review was undertaken in the following distinct stages: development of the review protocol, identification of the inclusion and exclusion criteria, a search for relevant studies, critical appraisal, data extraction, and synthesis.

4.3.2 Protocol development

A protocol for the systematic review was developed following the guidelines, procedures, and policies of the Campbell Collaboration (www.campbellcollaboration.org), the Cochrane Handbook for Systematic Reviews of Interventions [131], the University of York's Centre for Reviews and Dissemination's guidelines for carrying out or commissioning reviews [132], and the reviews of serious game research [4, 128]. This protocol specified the research aim, the search strategy, the inclusion and exclusion criteria, data extraction, and methods of the synthesis.

4.3.3 Data source and search strategy

For the purpose of this study, a literature search was undertaken between August and December 2010 in the following international online bibliographic databases: (a) ACM portal, (b) IEEE Xplore, (c) Springer, and (d) Science Direct. The search string used was: ("Game") AND ("Learning" OR "Teaching") AND ("Lecture" OR "Curriculum" OR "Lesson" OR "Course" OR "Exercise"). "Education" was not included in the keyword list since it was considered to be too general and it would not help minimize the searching scope. The search process was limited to titles and abstracts of articles published in journals and conference proceedings (some are book chapters), in English language, from year 2000 onwards. The latter limitation was imposed due to the rapid changes in ICT in general, and in computer game technologies in particular.

4.3.4 Data extraction with inclusion and exclusion criteria

Figure 6 shows the complete process of the data extraction. The first step was to identify relevant studies. Journal and proceedings articles related to GDBL were located during the search process in the afore-mentioned databases. The search resulted in 1155 articles. In step 2, the abstracts of the articles were searched for topics related to learning through game play or game development in curriculums. Most of the excluded articles were concerned with games used in classroom directly to motivate the students' interest and attendance rate, and using game play instead of traditional exercises to study or review the course content. For instance, these were articles, which addressed using virtual online multiplayer game environments to encourage a collaborative learning style, e.g. [133, 134], and articles which referred to games used in classroom to motivate attendance and to review the course knowledge, e.g.[28]. The articles related to the economics terms "game theory" and "business game" used as business terms were also excluded. Further, articles were excluded depicting novel game concepts,

which were not computer or video games but physical game activities without any technology support. For instance, article [135] used a self-made table card game in SE education. Mainly based on these three criteria, 1009 articles were excluded in this step.

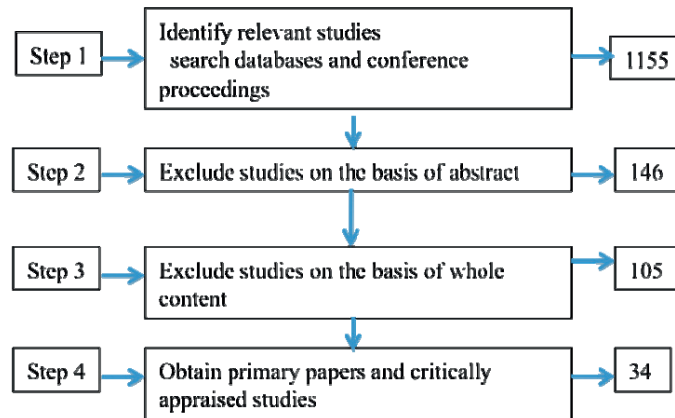


Figure 6: Steps of the article selection process

In step 3, the whole content of the articles was checked. The inclusion criteria were further limited to articles describing a case study or several case studies involving GDBL. In particular, it was required that the article contains:

- a) A relatively detailed description of lecture design process. The articles without detailed description of their teaching design or exercise process made it impossible to validate the process of integrating GDFs in lectures or exercises. According to this requirement, posters, tutorial presentations, and some short papers without detailed description of teaching process were excluded since they could not provide valuable data for this research and made it impossible to validate the effectiveness of the method, e.g. [136-141]. This was also a measure to ensure inclusion of high quality literature in the review.
- b) Articles using development toolkits in curriculum but did not aim to develop games were also excluded, e.g. [142].
- c) Articles emphasizing other aspects apart from GDBL were excluded as it was difficult to validate how game development was integrated in class, e.g. learning in a interactive e-lab [143]. Similarly, articles were excluded, which presented the development of an educational game framework, but did not mention how it was integrated in a specific curriculum, e.g. [144-147].
- d) Articles, which focused on changing the controller of the software or hardware, but without elements of computer game development, were also excluded, e.g. [148, 149]. Most of them focused on creating a robot controller to learn algorithms, or change some components of a robot to learn Artificial Intelligence (AI). In contrast, this study

included learning by modifying parts of a simulator to create the game elements or a game-like system, e.g. [120, 150]. Finally, a total of 105 articles remained after this step.

In step 4, the remaining articles were carefully studied and their topics, methods, teaching processes, and evaluation quality were compared. After the comparison, the following study requirements were included: 1) Evaluation data in these articles should be collected from assignments or scores after using the GDBL method. 2) Questionnaire should be converted to quantitative data and interviews or feedback should be converted to qualitative data. Based on the survey, it was found that empirical data were limited since GDBL is quite a new research area. In addition, diverse and innovative articles were also included; they used GDFs outside the scope of computer higher education, e.g. literacy for primary education [2]. However, papers reporting use of hardware tools to create game or game-like system, such as real robot hand [150], Wii remote [151], Microsoft surface [152], and a projector-camera system [153], to support teaching or learning environment were not included. Finally, a total of 34 articles were included in the review. This number appeared to be sufficient to create an extensive reference for explaining how to integrate the GDBL methods in curriculums.

4.3.5 Synthesis of findings

A typology to categorize the 34 articles had to be devised. The classification scheme proposed by [154] in their review of the general instructional gaming literature was adopted for the needs of the present study. This scheme, which was also used in [155], defines the following five categories [154]: (a) Research: systematic approaches in the study of gaming targeted at explaining, predicting, or controlling particular phenomena or variables; (b) Theory: articles explaining the basic concepts, aspects, or derived outcomes of gaming; (c) Reviews: syntheses of articles concerning general or specific aspects of gaming; (d) Discussion: articles describing experiences or stating opinions with no empirical or systematically presented evidence; and (e) Development: articles discussing the design or development of games or projects involving gaming.

The following criteria for classification of the articles were applied in this study. The articles were grouped into these five categories according to their primary focus. Of the 34 articles found in step 4, 20 were placed in the 'Research' category, one in the 'Theory' category, seven in the 'Discussion' category, and six in the 'Development' category, whereas no articles were appropriate for the 'Review' category, which underlines the usefulness and originality of the present study. As in other reviews of the general instructional gaming literature [129, 155], in this study there were fewer articles in the 'Theory' categories than in the 'Research', 'Discussion', and 'Development' categories. This can be explained by the fact that instructional gaming is a relatively new domain of educational technology, and that a substantial empirical base is needed to address relevant theoretical issues. The discussion above describes the research process of the systematic literature review of GDBL in this study.

4.4 Game Development as a Motivation for Lectures - Quasi-experiment

A “quasi-experiment” in fixed design is defined as a research design involving an experimental approach but where random assignment to treatment and comparison groups has not been used [25]. A quasi-experiment approach, using GDFs as tools in software architecture education in order to determine the effectiveness of GDBL in education, was adopted for two experiments conducted during the four years of this PhD project. Research methods presented in Chapter 3 were used to establish the core ideas of this research design. Further, these methods were combined with the experimental software engineering methods to guide the research process. The two quasi-experiments used both GDBL and non-GDBL in a software architecture course in order to measure the differences between them.

4.4.1 Quasi-experiment to answer RQ4

According to the research design in Figure 3, described in Section 3.1, the experiment was adapted to a software architecture course in order to provide a platform for using the GDBL method, as shown in Figure 7.

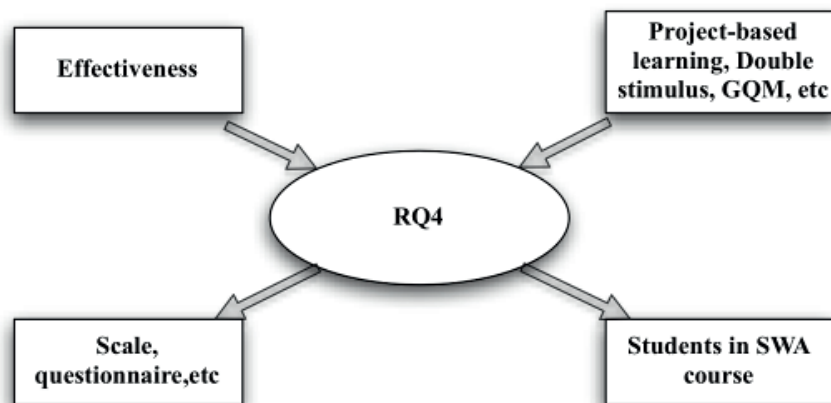


Figure 7: Mapping from framework to game design to RQ4

The purpose of the quasi-experiment was to identify the effectiveness of research on topic 2 in terms of 1) differences between using GDBL and non-GDBL within a software architecture course, and 2) the positive effects of using GDBL in a software architecture course.

This quasi-experiment included two experiments: 1) using XNA as GDF in a software architecture course, 2) using Android SDK as GDF in the same course. Based on own experiences in GDBL and literature review results, the aim was to identify the features

of GDBL. The two experiments included the same process steps: definition, planning, operation, data collection and analysis, and results reports. The results are summarized in the next chapter.

4.4.2 Experiment definition

The software architecture course is a post-graduate course offered to CS and SE students (not mandatory) at the NTNU. The course is taught every spring, its workload is 25% of one semester, and about 70-100 students attend the course every spring. The textbook used in this course is the “Software Architecture in Practice, Second Edition”, by Bass, Clements, and Kazman [156]. Additional papers are used to cover topics not sufficiently covered by the textbook, such as design patterns, software architecture documentation standards, view models, and post-mortem analysis [157-161].

The education goal of the course is as follows:

“The students should be able to define and explain central concepts in software architecture literature, and be able to use and describe design/architectural patterns, methods to design software architectures, methods/techniques to achieve software qualities, methods to document software architecture and methods to evaluate software architecture.”

The software architecture course at NTNU (course code TDT4240) is taught in a different way than at most other universities, as the students also have to implement their designed architecture in a project as an assignment. The motivation for doing so is to make the students understand the relationship between the architecture and the implementation, and to be able to perform a real evaluation of whether the architecture and the resulting implementation fulfill the quality requirements specified for the application. Throughout the project, the students have to use software architecture techniques, methods, and tools to succeed according to the specified project requirements and the document templates. The development process in the project is affected by the focus on software architecture, which prescribes how the teams should be organized and how they should work.

The grade awarded in the software architecture course is split, 30% is awarded for the software architecture project all students have to complete, while 70% is awarded for the results of a written examination. The goal of the project is for the students to apply the methods and theory in the course to design and document fully a software architecture, to evaluate the architecture and the architectural approaches (tactics), to implement an application according to the architecture, to test the implementation related to the functional and quality requirements, and to evaluate how the architectural choices affected the quality of the application.

The experiment includes the project exercises in both GDBL and non-GDBL. It consists of the following phases:

- 1) Commercial Off-The-Shelf (COTS): Learn the development platform/framework to be used in the project by developing some simple test applications.
- 2) Design pattern: Learn how to utilize design patterns by making changes in two architectural variants of an existing system designed with and without design patterns.
- 3) Requirements and architecture: Describe the functional and the quality requirements, describe the architectural drivers, and design and document the software architecture of the application in the project, including several viewpoints: stakeholders, stakeholder concerns, architectural rationale, etc.
- 4) Architecture evaluation: Use the Architecture Trade-off Analysis Method (ATAM) [156, 162, 163] to evaluate the software architecture in regards to the quality requirements.
- 5) Implementation: Design in detail and implement the application based on the designed architecture and the results of the evaluation. Test the application against both functional and quality requirements specified in phase 3, evaluate how well the architecture helped to meet the requirements, and evaluate the relationship between the software architecture and the implementation.
- 6) Project evaluation: Evaluate the project using the Post-Mortem Analysis (PMA) method [158]. In this phase, the students will elicit and analyze the successes and problems encountered during the project.

In the two first phases of the project, the students work on their own or in pairs. For the phases three to six, the students work in self-composed teams of four to six students, and they can decide whether their team focuses on game development or non-game development. In both cases, the students should apply what they have learned in the course. The students spend most time in the implementation phase (six weeks), and they are encouraged to start the implementation in earlier phases to test their architectural choices (incremental development). During the implementation phase, the students extend, refine, and evolve the software architecture through several increments.

4.4.3 Experiment planning

In previous years, the goal of the project was to develop a robot controller for a robot simulator in Java with emphasis on assigned quality attributes such as availability, performance, modifiability, or testability. The functional aim of this project was to develop a robot controller, which moves a robot in a maze collecting balls and bringing them to a light source. With several years of experience using the robot simulator in software architecture teaching, the course staff made the following changes in order to prepare the experiment:

- 1) Course preparations

The features of XNA and Android SDK were investigated and examples and self-learning materials were provided to the students. Further, the approach described below was used to integrate the game development project with the software architecture course.

2) Changes to the syllabus

It was rather difficult to change the syllabus of the software architecture course to include more literature on software architecture in games. Good books and papers with an insight into game architectures and game architecture patterns do not exist. There are several papers describing architectures of specific games such as [164, 165] or books giving a brief overview of game architecture [166, 167], but none of them covers the typical abstractions (architectural patterns) in game software development. In the end, the syllabus included some chapters from the book “Game Architecture and Design” [167] to study the initial steps of creating a game architecture, and two self-composed sets of slides on 1) software architecture and games, and 2) architectural patterns and games. The former was a one hour lecture on motivation in software architecture design in games [168], architectural drivers within game development [169], challenges related to software architecture in games [170], and the main components of game architectures [171]. The latter was a one-hour lecture describing architectural patterns, which are common and useful for games, such as model-view controller, pipe-and-filter, layered architecture, and hierarchical task trees.

3) Changes to the student project

The course staff decided to let the student teams themselves choose between the game project and the non-game project. This meant that the main structure of the project had to remain the same, and that two variants of the project had to be devised. For the non-game project, the students usually had fixed requirements; while for the game project the students were to define their own requirements (design their own game). However, the documents to be delivered were the same for both types of project, based on the same templates, and the development process was also the same. This ensured the same experiment conditions for GDBL and non-GDBL.

According to the above description, the main changes in project phases were as follows, as mentioned Section 4.4.2:

- Introduction to XNA game, Android game, and social application exercises.
- Requirements and architecture for the game/social application project.
- Evaluation of the game/social application project.
- Detailed design and implementation.

4) Changes to staff and schedule

The main change to staffing was that two last year master students were hired to give technical support for student during the project (both game and non-game project). The

main tasks of the technical support staff were to give lectures on the COTS, to be available on email for technical questions, to be available two hours a week in a lecture hall for questions, and to evaluate the implementation of the final project delivery (testing the games and the robots).

After changing the setting of the course, both game development project and non-game development project were used as student exercises. The comparison of the non-game and game project helped to discover the differences and reveal the positive and negative effects of introducing a game development project in the software architecture course. The experiment plan was designed based on the Goal Question Metrics (GQM) approach [116] where a research goal was first defined (conceptual level), then a set of research questions was defined (operational level), and finally a set of metrics was selected to answer the defined research questions (quantitative level). In this experiment, the GQM approach helped to identify each element during the research process and to ensure that this experiment properly planned and executed. Table 12 shows the experiment elements using GQM approach to evaluate GDBL in the software architecture course.

Table 12: GQM table for GDBL experiment

Goal	Analyze	GDBL used in the software architecture course			
	For the purpose of	Comparing game development with non-game development domains			
	With respect to	Differences between and effectiveness of two domains			
	From the point of view of	Researcher & Educator			
	In the context of	Students in the software architecture course			
Questions	Q1: Are there any differences in how the students perceive the project between students choosing a game project vs. students choosing a non-game project?	Q2: Are there any differences between the software architectures designed by students selecting a game project vs. students selecting a non-game project?	Q3: Are there any differences in the implementation effort in the project between students selecting a game project vs. students selecting a non-game project?	Q4: Are there any differences in the performance between students selecting a game project vs. students selecting a non-game project?	
Metric	M1: Number of students choosing game project vs. non-game project.	M3: Project reports	M4: Source code files	M6: Project score	
	M2: Questionnaire survey with 5 Level Likert Scale	-	M5: Time spent	-	

4.4.4 Data collection

There are six phases in the experiments, as mentioned in Section 4.4.2. In the first two phases, the students become familiar with the COTS. The real work on the project consisting of design and implementation starts in phase 3. The students are asked to report how much time they spent on the project when they were conducting design and implementation in phases 3 to 5. After the final implementation, the students answer a questionnaire survey and sit the final written examination. During the project, all groups (Robot, XNA, and Android) followed the same requirements and templates for the exercises in phases 3 to 6. The data for the final evaluation comes from these phases. The data collection methods include:

Questionnaire/Scale: After finishing the exercise, the students are required to answer the questionnaire designed by the author and his supervisor. The questionnaire is given to obtain the students' feedback and opinions on the game development projects. A reply to each question is on a scale from 1 - "completely disagree" to 5 - "completely agree" to identify the effectiveness of the experiments. The concrete scale, like SUS, was used to collect relevant data, but most of the questionnaires were created in this study for certain purposes.

Test/Score: The grades for the students' assignments and the final exam were also used as data to investigate results. The students' assignments were project based. The assessments can give insight into the effectiveness of GDBL compared with non-GDBL, and the examination scores can be used as supplementary evidence. Further, a comparison of the grades in the game projects and non-game projects can lead to a direct conclusion.

Observation/interview: The teachers and teaching assistants in the course can help the students with various difficulties and discuss topics of course setting, challenges, improvements, etc. Throughout this process, the teachers can observe student activities during the lectures and the exercises, and collect students' feedback during the course.

Documents: The student projects include several documents, which describe their software architecture or other attributes according to the requirements of the exercises. These documents are very valuable for a deeper analysis of how the students work and learn in GDBL. The project deliveries were analysed focusing on several key indicators which reflect the students' effort needed for the exercise.

Code: As a part of the final delivery, the students have to present the source code of their games and applications. The lines of source code, time spent and code structure was used to measure the effort the students put into the project to determine whether there were differences between the game project and the non-game project.

4.4.5 Data analysis

After using the above methods, the collected data have both quantitative data and qualitative data. Since the experiments were based on comparison, 95% of the data was

quantitative. Microsoft Excel and SPSS were mainly used to perform the quantitative data analysis. The analysis of data involved the following steps:

- i. Measure the central tendency using the mean, median, or mode.
- ii. Measure the variability by showing the range, inter-quartile range, variance, or standard deviation.
- iii. Measure two variables relations through comparing the game project data and the non-game project data.

This section described the research process, explained how the data were defined, prepared, generated, and acquired. Next chapter will present the results of the data analysis.

5 Results

This chapter presents the main research results of this PhD project. In topic 1, the experiences gained in four case studies are used to achieve the research goal. Topic 2 defines a new field “GDBL” and its research scope, supported by the results of the systematic literature review and two quasi-experiments. The results are expected to enrich the theoretical foundation for both lecture games and GBL.

5.1 Summary of the Studies

In the research of topic 1, game as a motivation for lectures, the focus was mainly on how to combine supportive theory and enabling technology with educational games in order to enhance the students’ motivation during learning and exercises. This guided the design of four case studies, which were conducted to show how the supportive theory is applied in the Lecture Games project and whether the design and concept can elicit positive feedback from students.

For topic 2, the game development as a motivation for lectures, the scope was defined and the research was conducted using two methods: literature review and quasi-experiments. Practical and concrete understanding of this topic was gained by conducting the experimental studies on GDBL in a software architecture course. The literature review on GDBL revealed a broad background of other research in this field resulting in a complete overview of GDBL.

Table 13 shows the focus of each of the papers published in the course of this study. For instance, paper G1 mainly contributes to theoretical grounding in “Flow theory”, and paper GDF2 is a systematic literature review with the main focus on a general level instead of a specific theory or technology application.

In addition, Table 13 shows the theoretical grounding and relevant technology applied in these studies, and the connections between the research questions and the papers. In the research on each topic, the contributions were systematic and spread across the papers, and the conclusions were based on case studies, literature review, and experiments.

Table 13: A brief outline and contribution of each paper

Paper	Theoretical Grounding	Enabling Technology	Research Question	Contribution	Research Method
G1	Flow theory	Features of mobile devices, including iPod Touch, Android Phone, etc	RQ1	C1, C2	Case Study
G2	Features of educational games, SUS		RQ2		
G3	SUS				
G4	EGameflow				
GDF1&GDF2	-	-	RQ3, RQ4.2	C3, C4, C5	Literature Review
GDF3	-	-	RQ4.1	C4, C5	Experiment
GDF4	Project-based learning, Double Stimulus	XNA as GDF	RQ3.2, RQ4		
GDF5&GDF6			RQ4.1		
GDF7		Android	RQ4.1		
GDF8		SDK as GDF	RQ4.1		

5.2 Game as a Motivation for Lectures

Four case studies were carried out to explain how to use both the technological aspects and the supportive theory to co-design games for lectures. There were two case studies emphasizing the technological impact of games more than the supportive theory. The other two emphasized the supportive theory more than technological aspects. This means that each case study targeted its primary goal, but neither of them neglected supportive theory or technological issues. Specifically, articles G1 and G2 used the game design theory and current technology platforms to co-design the cases, but the core idea was the theoretical foundation for lecture game design. Articles G3 and G4 reported studies involving both interesting game peripherals and relevant evaluation criteria as dual guidance for the design. The main focus of these articles was on the impact of recent technology on learning using games. The common theme of all four case studies was that they are based on quiz game play. However, they all ran on different platforms and provided different game concepts. A brief introduction of the four case studies is presented below in a chronological order:

5.2.1 Case 1 - World of Wisdom

This case study involves the implementation of an educational Massively Multiplayer Online Role-Playing Game (MMORPG), named World of Wisdom (WoW). Figure 8 shows the architecture of the WoW game. WoW design was guided by game design theory, and MMORPG's game features were based on surveys of popular MMORPGs. The game was built with an open source game engine, Golden T Game engine, as

shown in Figure 8. WoW is an open educational platform running on Windows and Mac OS where students can “play their exercises” instead of writing them on paper. It provides several kingdoms, where one specific kingdom represents one part of a curriculum or full curriculum for one course. Each kingdom has several zones, designated as a safety zone and battle zones. The safety zone is populated with re-spawn point, shops, and buildings. When players receive quests from a non-playable character (NPC) or the teacher, they go to various zones to complete these quests. If the quest involves some fighting, the players must go to the battle zones where they will fight monsters through answering various questions related to the curriculum. Further, the players can chat with each other or ask help from the teaching assistant inside the game. The game also provides an editor for teachers to create new game tasks and content without the need for programming. As an aid to learning, WoW can be a supplementary motivation for the students to do their exercises more thoroughly. This case was published in paper G1.

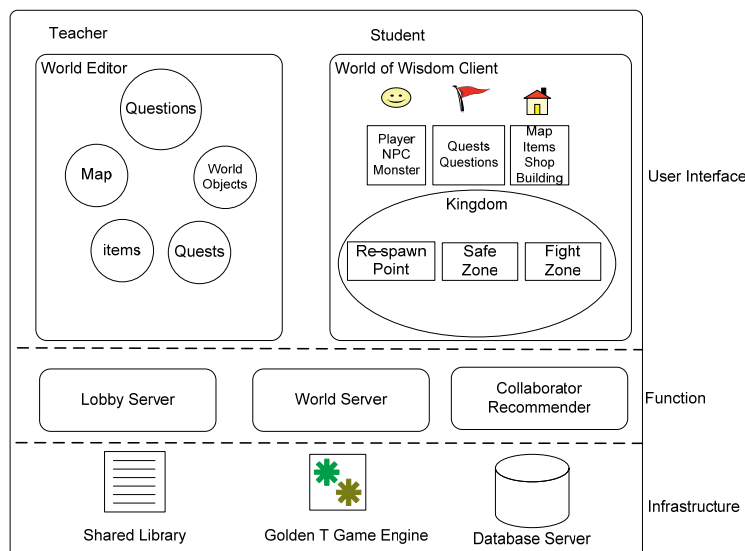


Figure 8: The architecture of World of Wisdom game

5.2.2 Case 2 - Lecture Quiz

A problem, when teaching in classrooms in higher education, especially in large classes, is the lack of support for interaction between the students and the teacher during the lectures. The lecture quiz concept was proposed, based on collected good educational game features, which can enhance the communication and motivate students through more interesting lectures. Figure 9 gives the system overview for Lecture Quiz. This game concept is based on the current technology-rich and collaborative learning environments. A SUS evaluation of the first version and the second version of Lecture Quiz was carried out, and it showed that the Lecture Quiz concept is a suitable game approach for improving most aspects of lectures. The second version of Lecture Quiz

was improved in several ways, such as addition of an editor for the teachers to update the questions, enhanced architecture easily extended to new game modes, web-based student clients to provide an easier start compared with the first version of Lecture Quiz, etc. The final results were published in paper G2.

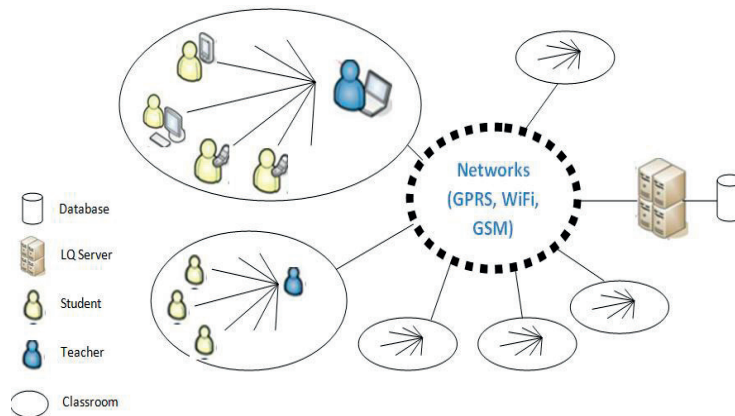


Figure 9: System overview of LQ 2.0

5.2.3 Case 3 - Knowledge War

This case study uses a location-aware educational game, KnowledgeWar, for the iPhone/iPod Touch platform. The initial idea behind the KnowledgeWar game was the notion that students spend much time walking around and socializing. The social interactions among students can be both face-to-face and electronic, using mobile devices such as smart phones and laptops. A major challenge for educational games is to create games, which can be used in several courses, but are still enjoyable. Single player quiz-games fit very well into this category, but they can be somewhat tedious and repetitive. By adding a social component, such games can be much more competitive and engaging. In this quiz game, students can challenge each other in face-to-face or remote knowledge battles. The game contains a game lobby where players can see all who are connected, and the physical distance to them. The implementation of the KnowledgeWar game is based on a service-oriented client-server architecture. An overview of the architecture is shown in Figure 10. A game server provides all services shared by users such as player profiles, question database, and game sessions. Apple's Push Notification Service is used to communicate events to the iPhone/iPod Touch clients. The results of a questionnaire survey, which included SUS questions, showed that the game has high usability, it is helpful for summarizing topics, and it can stimulate involvement through social interaction. It was also found that smart phones are well suited for such social games. However, the results also revealed that the game did not stimulate students to attend more lectures or pay more attention during lectures. This case was published in paper G3.

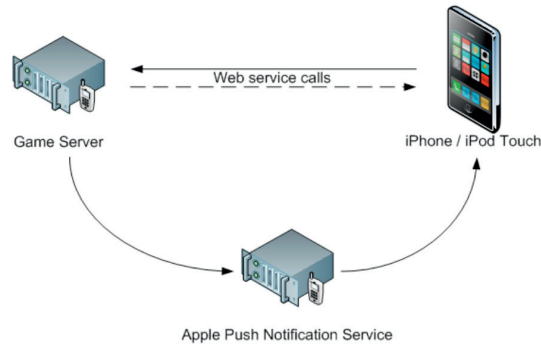
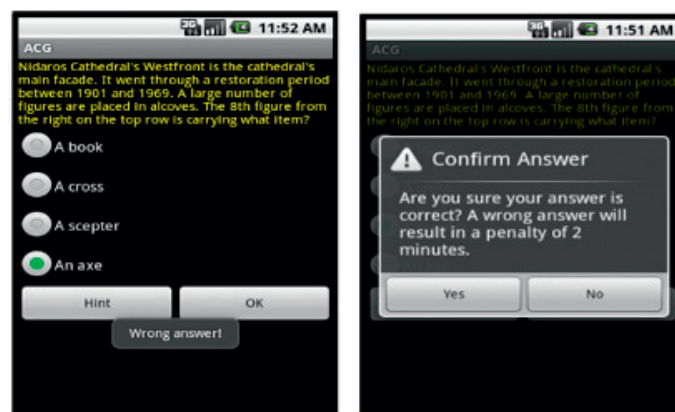


Figure 10: KnowledgeWar architectural overview

5.2.4 Case 4 - Amazing City Game

An integration of game, learning, and ubiquitous technology can result in a new and relaxing informal learning style. Inspired by this concept, an educational pervasive game was implemented on the Android platform where players can participate in a knowledge competition tour in groups in the city of Trondheim to gain better understanding of the city through solving various problems. This adventure game asks the contestants to undertake tasks at different locations by using relevant technologies available on Android smart phones. Each task uses one to three hints if the player is unable to proceed and not able to solve the problem. If a player uses a hint, a corresponding penalty time is added to the final score. Figure 11 shows the interface of the game running on an Android phone. The group to reach the final destination in the least amount of time is the winner. This study represents an innovative game concept for informal learning. The results of the EGameFlow framework [93] evaluation showed that the idea of using pervasive technology in a game and learning context is an interesting and innovative concept. The game scored high in the areas of feedback, immersion, and social interaction items in the EGameflow framework. The results were published in paper G4.



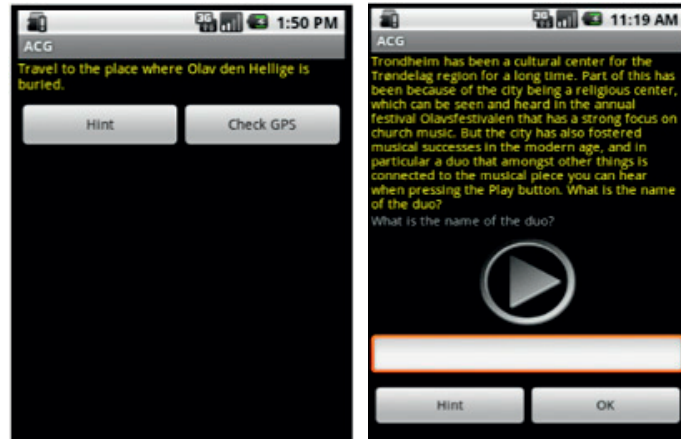


Figure 11: ACG user interface

5.2.5 Summary for questions RQ1 and RQ2

The aim of the above four case studies, related to topic 1 (game as a motivation for lectures), was to answer research questions RQ1 and RQ2.

RQ1: How can supportive theory be identified to guide the design and evaluation of lecture games?

In order to answer RQ1, the theoretical background was investigated in three areas: 1) Pedagogical theory; 2) Game design theory; and 3) Game evaluation criteria, which were already reviewed in Chapter 2, e.g. GameFlow model. After examining the supportive theories, the next issue was how to select and apply relevant theories to the case studies. It is not possible to provide a single theoretical framework for the design of lecture games based on the case studies. Usually, the first step is to create an overview of current or classic theories for the game design or evaluation, for example intrinsic motivation by Malone [29], GameFlow theory [91, 172] or flow experiences [92]. In addition, since the games are mainly used for lectures, the pedagogical aspect of the game design is also an important factor to consider, for example design of an educational game through collaborative learning [173]. In a specific case, some adjustment of the theory may be needed during the design process. In some cases, a tradeoff is necessary between different supportive theories. On the other hand, it is impossible to use/match all the elements from all the theories in each case. In the case studies, only the best matching elements were chosen to guide the design of lecture games. Further, the design theory was enriched based on a survey of game genres. Depending on a specific game genre, summary or review articles need to be located, which explain the features of this game genre, and these articles can be used as criteria for the design of such a game, as in the case study of WoW. The final answers and results are summarized in the next section.

RQ2: How can current relevant technology and appropriate peripherals be used to provide various play experiences in new lecture games?

In order to answer RQ2, relevant technologies and peripherals were reviewed. Secondly, guided by the available budget and the popularity of the devices, iOS/Android and iPod touch/Android phone were chosen as the main platforms. The iPod touch was chosen because it offers most of the functions of the iPhone at one-third of the price. Similarly, Android smart phones with comparable features are generally cheaper than the iPhone. Their attractive features are mentioned in Section 2.5.2. In terms of these features, a quiz concept was conceived for two popular mobile devices: iPod touch and Android phone. The similarity of both cases is that they both encourage the social interaction through the use of ubiquitous technology. Most of the features, for example the GPS function, the camera, the audio player, and interesting widgets, are applied in the games to provide diverse experiences and increase flexibility of game tasks for the players. The final evaluation of the cases shows that players gave positive feedback related to social interaction and immersion provided by the mobile technology used in the lecture games.

5.2.6 Contribution of the study

C1: Identification of research issues and cases related to recent technology-rich environment within the context of game as a motivation for lectures.

The research goal identifies two issues: the impact of supportive theory and recent technology on the design and evaluation of lecture games. Through the literature survey, it was found that the supportive theory is easy to neglect in the design and implementation of educational games. One possible explanation can be that no systematic supportive theory has yet been proposed in the relatively new field of GBL. A traditional design of an educational game usually comes from the developers' own experience and understanding. This is often limited in terms of the utilization of theory due to the developers' lack of relevant knowledge and experience. Typical problems can be that too much effort goes into innovative game ideas while the balance of learning and game play is neglected, or own subjective criteria are used to judge the play experiences. If a widely accepted theory could be found to guide the design of lecture games, it would not only ensure the game quality and learning goals, but also enrich the game content for learning. The foundation of supportive theory can be considered as a resource for the educational game design before the educational game is implemented. It provides a reference for the game design process. After the game has been implemented, the supportive theory provides a further reference for the evaluation and improvement of the game concept. Thus, the game concepts can be improved, re-designed, and re-developed based on the evaluation results. Their relationship can be explained as shown in Figure 12.

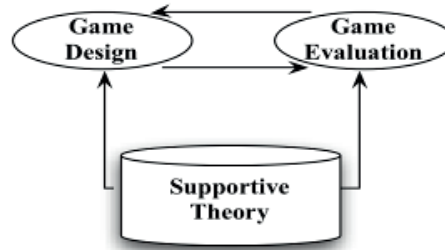


Figure 12: Relationships among supportive theory, game design, and evaluation

Apart from the supportive theory, another important aspect was the relevant computer technology and suitable peripherals. Most video games are popular within a limited timeframe since their graphics and game concepts get outdated. This is an inspiration to search for new ideas and technologies to be used in games. As mentioned earlier, new developments in technology and peripherals, from the Commodore 64 game system²² to the Wii with games controlled by body movement, from stationary devices to mobile devices (e.g. iPhone, Nintendo DS), always bring new playing experiences. This phenomenon shows a fast changing and improving technology environment, which encourages new game play experiences. In this study, instead of creating new technologies, close attention was paid to the current technologies and the ways to apply them to this research. Specifically, in order to identify the research scope, the focus was on two mobile device platforms popular in the project period of 2008-2012: iOS and Android platforms. The quiz style game play was utilized with various game plots. The supportive theory and enabling technology are believed to be the most important issues in this research. Their relations to game design and game evaluation are showed in Figure 13.

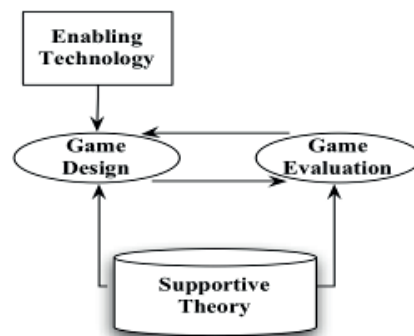


Figure 13: Research issues relations

²² <http://www.videogameconsolelibrary.com/pg90-64gs.htm#page=reviews>

Thirdly, four cases were implemented to explain how these two issues benefit game implementation and evaluation. These cases were already discussed in Sections 5.2.1 to 5.2.4. The four cases were the source of experience in design and evaluation of educational games based on the dual basis of supportive theory and current computer technology. Specifically, WoW and LQ focused on the supportive theory and balanced each element of the game using current mature technology. The evaluation of the design and implementation of these games included both positive and negative experiences. It was found that there are various theories, which can benefit the design of educational games. However, the key problem was how to choose and apply the relevant theory to support the design. This issue will be described later in this section. The experiences gained in the WoW and LQ cases are a good illustration of this key problem. The KW and ACG cases included innovative concepts illustrating how the newest computer technology at the time could be combined with a quiz. They provide a novel play experience in a real world in order to motivate learning. The supportive theory was also applied when designing both of these games, but SUS and EGameFlow were mainly used for game evaluation. Figure 14 illustrates the relationships between the cases and the research entities.

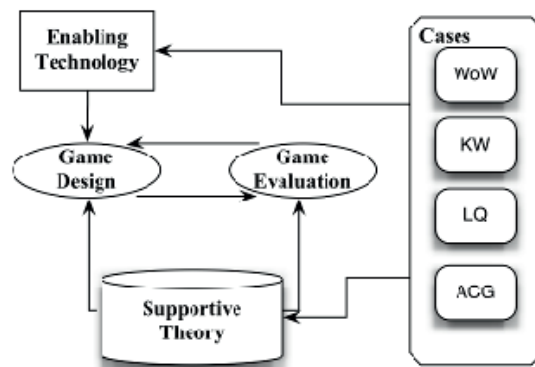


Figure 14: Relations between research issues and cases

C2: An analysis chart for applying supportive theory and enabling technology as dual guidance in the study of educational games for lectures.

Since research on lecture games is a cross-disciplinary field, the supportive theory framework is discussed and analyzed from three perspectives: 1) pedagogical theory, 2) game design theory, and 3) theory for evaluating games.

First, since the learning theory is an independent research area with various concepts to interpret the learning process such as behaviorist, cognitivist, and humanist, several interpretations exist related to the purposes of learning, as mentioned earlier. In this study, the learning process is understood from the perspective of an educational game technologist. At the beginning of the study, a connection between the possible learning theories and recent technology was examined to investigate the playing experience.

However, the selection of game technologies depends on the game content. After identifying the game purpose, which was to let students review the course content in a quiz game as exercises, the most relevant theory was chosen. This was the theory of *collaborative learning* as a learning strategy to match the game content and the enabling technology. One reason is that the four case studies have common features, i.e. multiplayer quiz game genre and the collaborative learning. The evaluation of results in the four case studies demonstrated that the social interaction in the class was improved by the lecture games. Most of the students expressed the opinion that this feature made the lectures more entertaining. Another reason is that this research includes just a few examples of how to apply the supportive theory and enabling technology together, instead of independently, in the game design. The process of combining learning theory is very complex and difficult. It is not possible simply to combine several learning theories and to utilize all the respective learning strategies in a game design. The process of combining learning theories and strategies is a separate research topic beyond the scope of this thesis. In general case, it is suggested that the most relevant and important learning strategies are selected to guide the game design process. These strategies should support the usage of the enabling technology to be exploited in a game. Further, the choice of the enabling technology always depends on the game content. It is recommended to start the game design process by examining the game concept, the enabling technologies, and the learning theory, and how the three can be combined. Finally, it should be measured how well these three components match each other through an evaluation of a game prototype.

Second, the game design theory was examined and it was found that there exists only limited literature on game design theories to guide the design of educational games. The most influential theory for such games is the “flow experience” theory and “intrinsic motivation” theory by Malone, as described in Section 2.3. However, there exists other literature to be considered when designing good educational games. Table 14 lists the characteristics of a good educational game according to the combined supportive game design theory.

Table 14: Characteristics of good educational games

ID	Educational game elements	Explanation	Reference
1	Variable instructional control	How the difficulty level is adjustable or adjusts to the skills of the player	[75, 174]
2	Presence of instructional support	The possibility to give the player hints when he or she is incapable of solving a problem	[174, 175]
3	Necessary external support	Providing appropriate computer equipment, technical support, time for the learner.	[174]
4	Inviting screen design	The feeling of playing a game and not operating a program	[174]
5	Practice strategy	The possibility to practice the game without affecting the user's score or status	[174, 175]
6	Sound instructional principles	How well the user is taught how to use and play the game	[174, 176-178]
7	Concept credibility	Abstracting the theory or skills to maintain	[179]

		integrity of the instruction	
8	Inspiring game concept	Making the game inspiring and enjoyable	[4, 75]

The above eight important characteristics of good educational games are proposed based on the combined theories and the experiences gained in the Lecture Games projects. The list of characteristics was compiled as a reference for a researcher/developer designing educational games. Note that missing one of the characteristics in the game does not mean that the game will be useless or unsuccessful, but including the missing characteristics in the game may improve it.

Thirdly, the WoW case study used an additional approach in the game design. Every game can be categorized according to its genre, besides serving an educational purpose. Since WoW is an educational MMORPG game, the literature on main features MMORPGs should be consulted in addition to theories on educational game design. The current trends in MMORPGs and their characteristics were surveyed. Achterbosch [180] listed the features which contribute to a successful MMORPG. The most of the relevant educational features were chosen for the WoW design. Table 15 lists the most relevant features for designing an educational MMORPG.

Table 15 Most relevant features for design of an educational MMORPG

ID	Existing features	Note
1	Three character development models	Skill points-based system; class-based system; combination of class/skill
2	Multiplatform support	-
3	Highly customizable characters	-
ID	Favorite Features	Note
4	Preferred character types in ranking	Combination of class/skill; skill points-based system; class-based system
5	Top 3 game setting	Fantasy, futuristic, post apocalyptic
6	Top 5 MMORPGs	Many class/skill options, graphics and effects, large world to explore, Player vs. Player, socialization
ID	Improving existing features	Note
7	Player versus player combat	Such as balancing between classes
8	The level grinding: process of engaging in repetitive tasks during video games.	Repeated battles to increase the level
ID	Anticipated new features	Note
9	Player created and controlled	-

	content	
10	Mini games	-
11	Dynamic content and quests	-

With the WoW design and implementation experience, it was found that features identified in certain types of games are also useful for educational games of the same type. It was noticed that in addition to learning theory and game design theory, there is still space for using other theories to support educational game design. This case shows that there exist other possible disciplines or areas contributing to the design of lecture games. It is better to consider them during the design process before presenting the final lecture game to students. Further, there might be duplicating or conflicting aspects in these theories. If this is the case, more experiments are needed to gain more experience in trade-off between these theories during the design process, selecting the most relevant theories for the game design.

Fourth, the criteria for the evaluation of games should be discussed. Basically, the GameFlow model derived from flow experience, described in Section 2.3.1, is the core criterion for the evaluation in these studies. Further, the GameFlow model was extended to create the EGameflow framework for assessing educational games. The difference between them is that “Knowledge Improvement” was added as a new element for the evaluation of the learning output. All of these evaluation models have detailed sub-criteria, which can serve as reference in designing the content of an evaluation questionnaire. According to the experience in this study, these models can be used as a reference and a starting point, but not all sub-criteria need to be included as not all of them might suit the game being evaluated. Some of the criteria had to be adapted to evaluate an educational game better. In addition, it was found necessary to add new sub-criteria to cover certain aspects of the research focus. Apart from the GameFlow model, the SUS method was also tailored to shift its focus from evaluation of system usability in experimental software engineering to evaluation of games. Another application of these evaluation criteria is to use them as design criteria for educational games in order to remind the game designer of potential weaknesses in the design. It is much better and cheaper to find these weaknesses during the design rather than after the implementation is completed.

Figure 15 shows an analysis chart of guiding principles for using supportive theory and technology as co-design of lecture games. This figure extends the chart in Figure 14 with four steps related to supportive theory. It is based on the case study experiences and the above analysis. This analysis chart should be used as a framework for good educational game design.

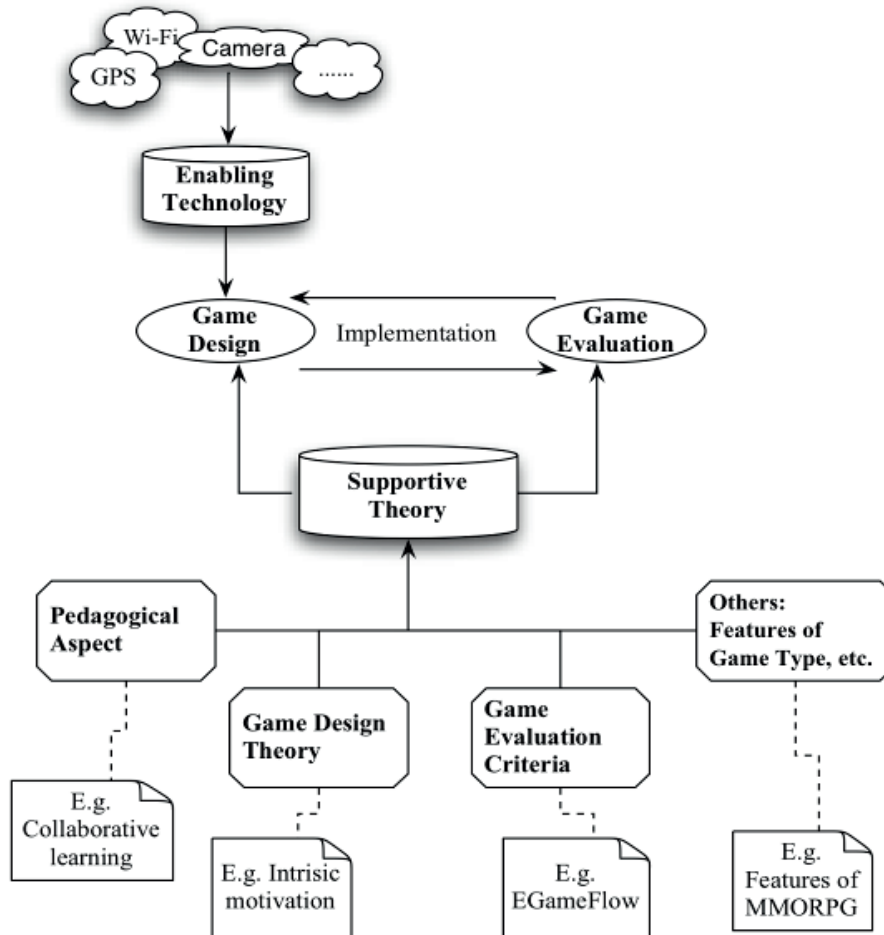


Figure 15: The analysis chart of guiding principles for educational game design

The above discussion suggests that there is no single method for educational game design. This chart helps to expose the issues in lecture game design from multiple angles. Game design has a large variety of aspects, depending on the specific case or game content, and a trade-off is often necessary for a good design. It is usually impossible to cover all of the aspects of the supportive theories.

5.3 Game Development as a Motivation for Lectures

The focus of GDBL is quite different from games used as a motivation for lectures, and it has its own supportive theory and enabling technology. A systematic literature review was conducted to obtain an overview of the GDBL field. Further, two quasi-experiments were carried out to discover how to apply the technological aspects and

supportive theory to co-design GDBL for lectures. First, a preliminary literature review was conducted, focusing on GDBL within the software engineering field. However, GDBL can also be applied in other domains. Therefore, another systematic literature review was carried out to cover all possible applications of GDBL. At the same time, two quasi-experiments were conducted. At the beginning of the first experiment, a traditional software engineering course had to be modified to adapt to the GDBL method, as mentioned in Section 4.4.3. After the preparations, one experiment was conducted where XNA was used as a GDF in a software architecture course. Another experiment focused on the use of Android as a GDF in the same course.

5.3.1 Literature review of GDBL

First, a tentative small-scale literature survey was conducted on the methods of creating/modifying a game using a game development framework (GDF) as an assignment to learn software engineering (SE). Based on the survey, the theoretical context was identified to guide the design of using GDF in SE. Second, relevant GDFs used in SE education were analyzed, namely Alice [139, 181-183], Scratch [184-186], CeeBot Series [120], Warcraft3 Editors [43], Never Winter Night Toolsets [187], Greenfoot [188], Game maker [189, 190], StarLogo TNG [191], and Wu's castle [192]. According to the analysis of these cases and the author's experiences, a general recommendation was formulated for choosing an appropriate GDF for the SE education. These results were published in paper GDF1.

Based on the above preliminary results, it was found that learning through creating/modifying a game on a GDF can be used in CS or SE to study topics such as data structures, development processes, artificial intelligence, graphics programming, and team management. It was also found that game development could be applied to fields other than CS and SE, such as literacy education [2]. In order to define the research field accurately, this style of learning was labeled as game development based learning – GDBL. The next step was to conduct a systematic literature review. Specifically, the study aimed at critically reviewing published scientific literature on the topic of the GDBL method utilizing game development frameworks (GDF). The systematic review of online bibliographic databases, mentioned in Section 4.3, resulted in selection of 34 relevant articles for the final study. Further, combined with the characteristics of the GDBL method, three aspects of the articles were analyzed: (a) pedagogical context and teaching process, (b) selection of GDFs, and (c) evaluation of the GDBL method. The overview of 34 articles suggests that GDFs offer many potential benefits as an aid to teaching computer science, software engineering, art design, and other disciplines, and that such GDFs combined with the motivation from games can improve students' knowledge, skills, attitudes, and behaviors compared to traditional classroom teaching methods. Through the systematic literature review, teaching strategies guidelines were compiled for using the GDBL method in a curriculum, identifying features of GDFs related to GDBL, and presenting a synthesis of the available empirical evidence and impact factors on the educational effectiveness of the GDBL method. The empirical evidence to support the educational effectiveness of GDBL is still rather limited, but current findings indicate a positive overall picture. The outcomes of the literature review were discussed in terms of their implications for future

research, and they can provide useful guidance to educators, practitioners, and researchers in the areas of GBL. These results can even be used by the GDF creators, as suggestions are given what functionality a GDF should have to serve pedagogical purposes. These results were described in the published paper GDF2.

5.3.2 Experiment 1 - XNA used as a GDF

In 2008, the students in the software architecture course could choose between two domains in their project: Khepera robot simulation in Java or XNA game development in C#. Independently of the domain chosen, the students had to go through the same phases, produce the same documents based on the same templates, and follow exactly the same process. Through the evaluation, the main conclusion was that game development projects could be used successfully to teach software architecture. Further, the results of the evaluation showed, among other things, that students who chose the game project produced software architectures of higher complexity, and put more effort into the project than the Robot project students. No significant statistical differences were found in final grades awarded to the game project students as compared with the Robot project students. However, the game project students obtained a higher grade in their project than in the written examination, whereas the Robot project students scored higher in the written examination than in their project. Finally, compared to the Robot project students, those that chose the game project had fewer problems with COTS affecting the architecture design and introducing technical challenges. In addition, in order to simplify the development process of using XNA in education, a Microsoft XNA extended library, XQUEST (XNA QUick & Easy Starter Template), was developed for the software architecture course. The evaluation of the results showed that XQUEST enhanced XNA suitability as a teaching aid in software engineering learning, and that it can be a useful and helpful tool for students to understand XNA. The detailed process and the results were published in papers GDF3, 4, 5, and 6.

5.3.3 Experiment 2 - Android SDK used as a GDF

This experiment focused on how to extend Google's Android platform as a game development tool to learn software architecture based on the double stimulation method, mentioned in Section 2.2.2. During 2010-2011, students in the software architecture course at NTNU could choose between four types of projects: development of a robot controller using Java on the Khepera Robot Simulator, development of a game on the XNA platform, development of a game on the Android platform, and development of a social application on the Android platform. Independently of the chosen type of project, all students had to go through the same phases, produce the same documents based on the same templates, and follow exactly the same process. This study focused on the Android projects, to determine how the application domain affects the course project independently of the chosen technology. The results revealed some positive effects for the students choosing game development compared to social application development on Android SDK, for example motivation to work with games, a better focus on quality attributes such as modifiability and testability during the development, production of software architectures of higher complexity, and higher productivity in writing lines of code. However, no significant differences were found in awarded grades between

students choosing the two different domains. In addition, in order to simplify the game development process using Android SDK in students' assignment project, an extended library, called "Sheep", was developed to help the students in the game development on Android platform. The final results were published in papers GDF7 and GDF8.

5.3.4 Summary for questions RQ3 and RQ4

This section is a summary of answers to questions RQ3 and RQ4 based on the experiments related to topic 2 - game development as a motivation for lectures.

RQ3: What is game development based learning (GDBL) and what are the researchers' views of GDBL?

This question consists of two parts. To answer the *first part*, the scope of GDBL, the term GDF and a taxonomy for GDBL must be defined. GDBL integrates game development and course knowledge in assignments based on the GDF. The GDF denotes the development tools or platforms for the game development purpose. This definition is based on the preliminary small-scale literature review and the first experiment. To answer the *second part*, a systematic literature review was conducted to identify research papers describing how GDBL was used in their studies. The most relevant cases were collected and their content examined in terms of supportive theory aspect, technical issues, teaching process, and evaluation results. Afterwards, through the compilation of data from the literature review, the results reported by different research papers were summarized. The data was extracted and summarized according to the most common views of researchers. These related mainly to the supportive theory for the design and evaluation process, as well as GDF features as the most important technical issue. Three aspects were discussed from the point of view of a researcher: 1) theoretical context to support the design of GDBL, 2) common technical issues in GDFs, and 3) impact factors, both positive and negative, based on results of experiments or evaluation of the teaching process.

RQ4: How can the GDBL be characterized in terms of supportive theories and current technology-rich environment?

In order to answer this question, quasi-experiments were carried out with a focus on using GDFs in GDBL based on the supportive theory such as the Project-based learning methodology, mentioned Section 2.2.3, for the design or using the SUS, mentioned in Section 2.4, for the evaluation. Specifically, the first experiment, conducted in 2008, was to use XNA in software architecture. The second experiment, conducted in 2010-2011, was to use Android in software architecture. Before the first experiment was carried out the research context was investigated and some parts of the course changed to match the GDBL. The experiment was prepared by designing questionnaires and choosing metrics to assess the results. During the experiment, students were interviewed to get feedback, the project implementation process was observed, source code and documents from the assignments were analyzed, etc. In both experiments, experiences using GDBL were accumulated and the issues, relevant to application of GDBL in software architecture course, identified. Most of the issues encountered in the

experiments were also mentioned and discussed in some of the reviewed literature, for example selection of a suitable GDF for the course. Finally, based on the results of the literature review and the analysis of data from quasi-experiments, a framework was proposed to guide the design of GDBL. The results are detailed in the description of contributions in the next section.

5.3.5 Contributions of the study

C3: Identification of a set of research themes and elements in GDBL.

First contribution to GDBL was to identify the research issues related to the research goal based on the literature review and the experiments. Many aspects of GDBL deserve to be studied, and it was not the intention to explore and solve all the problems exhaustively. The most interesting and important issues in terms of the research goal and experiments were investigated. Finally, four themes were selected and discussed in relation to GDBL: 1) pedagogical context support, 2) teaching process, 3) taxonomy of GDFs, and 4) selection criteria of GDFs for GDBL. For each research theme, some relevant entities, which deserve to be discussed, were identified. A map illustrating relations of themes and entities is shown in Figure 16.

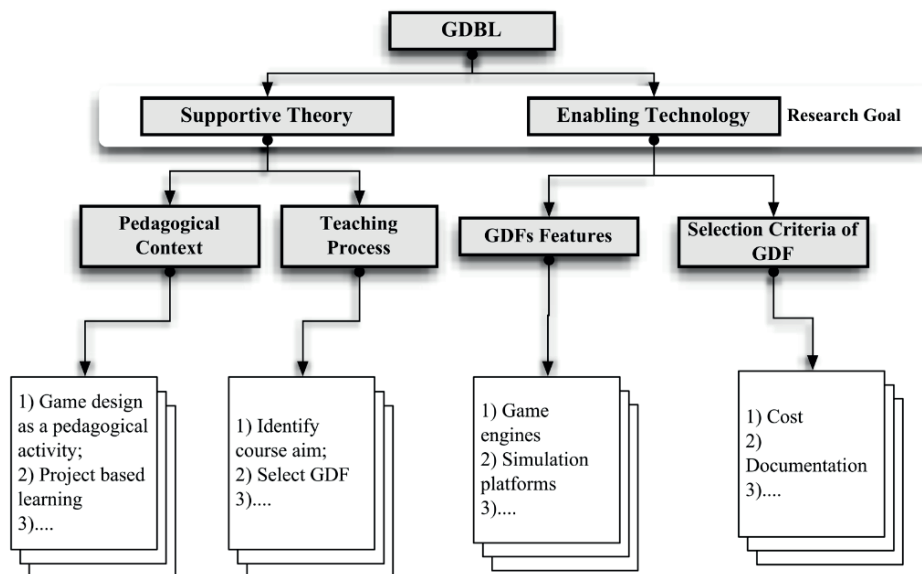


Figure 16: Relations of themes and entities in GDBL

For the pedagogical context in Figure 16, there exist literature considering game development - as opposed to game play - as a pedagogical activity in the classroom. Seymour Papert presents a relevant conclusion - programming as one example of the constructionist learning theory [43]. This can be a fundamental concept explaining the

pedagogical context of GDBL. Based on Seymour Papert's opinion, another question is how to use the pedagogical theory to support the design. A possible response is double stimulation [76] and Project-based learning to guide GDBL's design, as mentioned in Section 2.2. The above results support the validity of using a GDF in education from a pedagogical point of view. Basically, they show that students creating games by applying the course content on GDFs participate in a knowledge construction process, and this process can be integrated with pedagogical theory support, for example double stimulus or Project-based learning, to improve the learning process. For instance, when double stimulus is chosen as pedagogical support, the learning design can be decomposed into two main elements: one is a problem, task, or goal designed by the teacher, and the other is the corresponding learning activity undertaken by students. The first stimulus is the task or assignment, and the second stimulus is the tool chosen to suit the first stimulus. The outcome depends on teachers' capacity to keep the two elements matching each other. A good task (first stimulus) with inappropriate GDF (second stimulus) will not optimize the output. With this double stimulus support in mind, teachers should find an appropriate approach to connect tasks and GDFs instead of just focusing on one aspect. It is not recommended, for instance, to design the best possible task but neglect the effort of selecting the GDF. This would not be the correct way of applying double stimulus. Further, if the selected GDF always conflicts with the tasks, changing the tasks or tools should be considered, including application of a non-game tool. It implies that the double stimulus approach can support learning activity for both GDBL and non-GDBL methods. The teachers should keep this in mind when they apply double stimulus in teaching, and carefully analyze which tool is better for the course aim and for the students.

Based on the survey and the experiments in this study, the necessary common steps were identified for teaching based on the GDBL method:

The *first step* is to identify the explicit aims of a course. Since each course has its own educational goal, applying GDBL methods should not hinder reaching the course aim. When the course aim is clear, a common way to integrate GDBL in the course is that the teacher designs an assignment where game development is required. Then, the students develop a solution to this assignment in order to learn course content. When a teacher considers applying GDBL in a certain course, she or he should find an entry point to integrate GDBL with the course and its exercises.

The *second step* is the exercise design and the selection of one or more GDFs. When applying a GDF in a certain course, the selection usually depends on the course content and exercises types. Three types of game related exercises were recognized. The first type is to modify a game or add a component to a game or simulation platform to implement a complete game. The second type is to create a simple exercise using a GDF to study or practice one or two concepts in the course content. The third type is to carry out a complete game development project applying all concepts in the course. Usually, the first and second types can be used at the beginning of a course as a transition period when students are not familiar with the GDF environment, while the third type exercise can be used as a final project. The main driver of exercise design depends on the course

aim and students' background. The selection of GDFs is governed by separate criteria discussed later.

The *third step* is to include a tutorial lecture where the GDF is introduced to the students.

The *fourth step* is to run an initial exercise, which should be easy and motivating, and let the students become familiar with the development environment. If the allocated teaching time is limited, classroom guidance teaching over several hours can be used.

The *fifth and final step* is to complete exercises, typically integrated in a larger project, which includes the implementation of a game.

The GDFs can be classified as (a) game engines, (b) self-made GDF, (c) games or game editors, (d) simulation platform, and (e) others common tools, which were already mentioned in Section 2.5.3. To guide the choice of a GDF for GDBL, the GDFs are further classified into two categories: GDFs for novices, and GDFs for developers. The main focus of GDFs for novices, including non-programmers, is to provide visual methods for customizing game templates and to allow creation or design of games with little or no programming skills. The main focus of GDFs for developers is to offer toolkits supporting development of high quality 2D/3D rendering, special effects, physics, animations, sound playback, and network communication, in common programming languages such as C++, C#, and Java. Table 16 presents a GDFs resource pool based on the systematic literature review results. It includes XNA and Android used as GDFs in the quasi-experiments. Note that the list of GDFs is not complete. It mainly gives examples of GDFs from the literature review.

Table 16: Study of GDFs

	GDFs for novices	GDFs for developers
Game engine	Alice (http://alice.org); Scratch (http://scratch.mit.edu); Greenfoot (www.greenfoot.org); Game maker (www.gamemaker.nl)	FPS game engine: Torque game engine /Unreal Engine XNA (www.xna.com)/ XNACS1Lib framework/ XQUEST/ BiMIP
Self-made GDF	StarLogo TNG	-
Game or Game editor	Game editor: Warcraft3 Editors/ NeverWinter Night toolsets Game platforms: Bomberman /Wu's Castle/ Critical Mass board game/quiz-based web game shell	-
Simulation platforms:	-	Simulation platforms: Spacewar simulator/ RoboRally/ JGOMAS MUPPETS/ SIMPLE framework
Common	Maya/ Photoshop/Flash	Android/Sheep

tools	(www.android.com)
-------	-------------------

The following common guidelines, based on the literature review and experiences, can be followed when selecting a GDF: (a) *Technical environment including costs to use and acquire*: Technical environment requirements specify operating system and hardware, what tools are provided, what third-party tools are supported, and the difficulty to install the GDF. For instance, a typical problem is that XNA runs only on Windows, and many students now have PCs running Linux or Mac OS X. The technical requirements might also be an economical issue, as the choice of GDF might force hardware upgrades or paying for licenses. (b) *Sufficient documentation to guide the usage of GDFs*: Students need to explore the GDF as an additional task before they start game development on the GDF. If the resources and materials are sufficient and easy to acquire for beginners, it will help them shorten the time spent on getting to know this environment. (c) *Matching the students programming expertise - easy to learn and allowing rapid development*: This issue is also driven by time-constraints. Usually, if learning a GDF is not the main study aim in the course but rather an aid to learn something else, learning a new GDF requires additional effort and time in the course schedule. An easy and friendly environment is needed for the students to focus more on the course content and less on the GDF. (d) *Not in conflict with the educational goals of the course*: All GDFs have constraints related to course content in terms of how they were designed or how they are released. For example in SE education, open source GDFs make it possible to perform white-box testing on this GDF. Further, some GDFs might have constraints on games design in terms of design and architectural patterns, event-handling, ability to expand the functionality, and more. These constraints must be integrated in the SE teaching to introduce the students to the real world where software is rarely built from scratch. (e) *Using a common programming language*: This issue applies to the types of GDFs for developers using commercial game engines with widely known programming languages, like C#, Java, and C++, which are familiar to the students. A common language is not really needed for the GDFs for novices, if the course lets students know the data structures. In the long term, special programming languages are not as useful as widely accepted and used programming languages if the students will do more software programming in the future. (f) *Flexibility to combine a GDF with teaching materials and possibility to add/change libraries to be used within the GDF*: If GDFs are not easy to use, and not strongly relevant to the course content, an additional high-level library with APIs can be created along with a user guide to reflect course content in the GDF. (g) *Amusement and interactivity*: The GDF should provide a visual and stable development environment to encourage students' curiosity and imagination. A game development assignment in a user-friendly game development environment could be a good motivation for the students compared to traditional assignments. For example, most students think that it is more interesting to work on their own game project compared to a system for a bank.

C4: Identification of factors contributing to the success or failure of GDBL.

The systematic literature review and the experiments revealed the impact factors, which could cause positive or negative outcomes of GDBL. The factors are categorized as

positive, negative, or neutral (may cause both). The list below summarizes noteworthy factors in applying GDBL:

1) *Communication between the researcher and the teacher towards the understanding of the course content:* This item does not apply where the teachers and the researchers (including GDF's developers if any) are the same persons. If the method is designed by the researchers independently and the researchers invite the teachers to adopt it in schools, good communication and mutual trust are crucial to achieving the desired effect. For instance, in each case the teachers should become comfortable with using GDBL and spend more time on it compared to traditional method, otherwise a misunderstanding or bias against GDBL may develop. The researchers may be concerned that the teachers do not have a complete understanding how usage of games can improve education, and how motivation through games can be used to improve the course design. This indicates that researchers should help teachers in gaining self-confidence, and provide constant support while the decision is made to apply GDBL in a course.

2) *Teamwork:* This factor could have a positive or negative effect on the teaching results. The team size and working environment must be considered in advance. For example, laboratory environment with teamwork can help to improve the effectiveness of cooperative learning. In addition, instant communication in a team has significant impact. Group work can help weaker students, but unexpected situations can occur during the teamwork to hinder the instant communication, so students need some experience in working effectively in teams. Most of the case studies found in literature provide the evidence that teamwork can be used together with GDBL and the nature of teamwork is suitable for cooperative learning. However, a few articles described examples of applying cooperative and competitive learning in the exercises, with a positive feedback in both cases.

3) *GDF relevance:* There are three aspects which impact the outcomes: (a) advantages of using interactive graphical GDFs (graphics can provide instant feedback, making student engaged in programs), (b) a GDF can improve students' confidence in handling programming tasks, (c) it is necessary to analyze the features of GDF in the light the course content, and detailed GDF tutorials should be conducted before it is used in later exercises.

4) *Students' background:* The current students' background was presented in Section 1.2, showing that most of them played games as they were growing up. This is a suitable pre-requisite to GDBL. The negative aspect is addictiveness of games. Some students may focus too much on the game and game development thus losing focus on what they should learn in a course. This means that the design of the course and the project must be carried out in such a way that the students are forced to learn and use course content and theory to succeed in the project. It was also noticed that the diversity of student background causes some difficulty in using GDBL. The programming experience of the students strongly affects the choice of GDF between the ones for novices and the ones for developers. For instance, to use XNA/XQUEST or Android/Sheep for developers (Table 16), the students must know object-oriented (OO)

programming well and be familiar with OO design patterns and OO principles. Some other GDFs might require learning a specific simplified programming language for game creation, which is more suitable for students without programming experience.

5) *Teachers' requirements:* Teachers' attitude to applying the GDBL method in the course is the essential aspect of including GDBL in a teaching process. It is suggested that the faculty should master relevant technical knowledge and skills in the GDF before introducing GDBL. They should prepare and solve the anticipated problems they may face during teaching. It is essential that the course staff have technical experience in the selected GDF to provide help for students and to avoid the focus shifting from the course content to technical matters.

6) *Time-constraints and workload:* The experiments and the literature survey showed that limited time was a key problem when applying GDBL. It is suggested that students read the background material carefully before the class in order to save class time for actual exercises. Time constraints were particularly felt in the beginning phase. Some students complained about insufficient time to complete the project. To help in time management, a comprehensive time schedule should be prepared in advance for both the teacher and the students.

C5: Framework of linked elements for the design of GDBL.

After validating all the important entities in the design of GDBL in contributions C3 and C4, guidelines for integrating a GDF in learning and teaching strategies were created. Figure 17 shows a simplified high-level diagram giving an overview of the design process of applying GDBL. It contains four elements (course aim, pedagogical theory support, GDF resource pool, and impact factor), two methods (learning by creating and learning by modifying games), standard six steps in teaching process, and two roles (students and teachers).

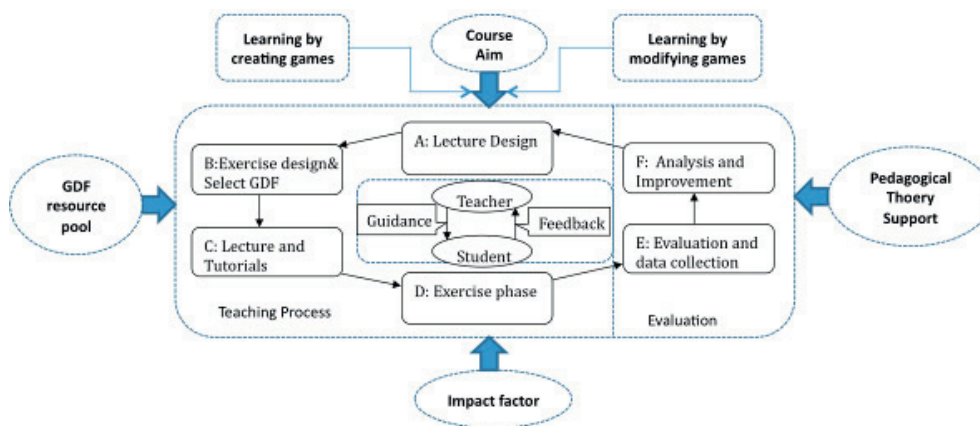


Figure 17: A Guideline for technical and pedagogical co-design of GDBL

The course aim has the fundamental impact on the selection of GDF and pedagogical theory to support the teaching design. The GDF resource pool in Table 16 could be the reference point for the selection of GDFs. Usually, during steps A and B in the teaching process, the pedagogical theory support and GDF resource pool play important roles. The impact factor of the GDBL should be considered for the whole process from the beginning. Based on the course aim, pedagogical theory support, and GDF resource pool, the teaching process starts with designing lectures and exercises using the selected GDF, then lectures and tutorials covering course content and GDFs are prepared. Finally, the course delivery starts and students begin the design and implementation of their projects. For the evaluation framework, it is suggested that teachers/researchers collect data using surveys. Based on the analysis of the collected data and teaching experiences, they can improve the teaching process. Here, a compact case is used to explain how each element in Figure 17 works in a certain course if the GDBL method is applied. The assumption is that the course aim is to teach basic programming for beginners. The choice should be made between “learning by modifying games” using a game editor with scripting, and “learning by creating games” in a GDF for novices. Then, the relationships between the problems and tools should be considered from the perspective of double stimulus, or other pedagogical support theory should be used to construct the learning process, for example Project-based learning. With this in mind and according to criteria for the choice of GDFs, commonly used tools can be selected from the GDF resource pool - GDFs for novices in Table 16 or another GDF if no suitable GDF is found in this table. After finishing steps A and B in the teaching process, the lectures start with the introduction to both exercises and GDFs. Later, the students commence the implementation individually or in groups. During the whole teaching process, from A to D, the impact factors are relevant but optional. For instance, a graphical interactive GDF can be chosen, time to be spent on lectures and on exercises should be estimated. Applying the impact factors in the teaching process depends on certain course situations. The evaluation and analysis steps E and F in Figure 17 serve this purpose. The feedback data can help to validate the choice in each step to determine whether the right task or the suitable GDF was chosen and whether the focus is on the most relevant impact factors in a specific course. In addition, because of the interaction between various elements in GDBL, a deeper analysis and evaluation must take place. Thus, an effective evaluation helps to validate the whole teaching process, and it is not judged by teachers’ own experiences only.

6 Evaluation and Discussion

This chapter revisits and justifies the choice of the research questions for this study (Section 6.1), evaluates the contributions (Section 6.2), and examines the issues related to internal, external, construct, and conclusion validity (Section 6.3).

6.1 Evaluation of Research Questions

6.1.1 Evaluation criteria

This section revisits the research questions and justifies why the particular research questions were chosen. According to [25], there are some basic characteristics of good research questions:

- Clear: They should be unambiguous and easy to understand.
- Specific: They should be sufficiently specific for it to be clear what constitutes an answer.
- Answerable: It should be possible to see what data are needed to answer them and how those data must be collected.
- Interconnected: The questions should be related in some meaningful way, forming a coherent whole.
- Substantively relevant: They should be worthwhile, non-trivial questions worthy of the research effort to be expended.

In order to evaluate each research question, they are classified according to the purposes of enquiry as “exploratory”, “descriptive”, “explanatory”, and “emancipatory”. According to [25]: *“The exploratory question is almost exclusively of flexible design. It is to find out what is happening, particularly in little-understood situations to seek new insights or to assess phenomena in a new light or to generate ideas and hypotheses for future research. The features of descriptive question are typically to portray an accurate profile of persons, events or situation and require extensive previous knowledge of the situation, to be researched or described, so that you know appropriate aspects on which to gather information. The explanatory question seeks an explanation of a situation or problem, traditionally but not necessarily in the form of causal*

relationships and explains patterns relating to the phenomenon being researched to identify relationships between aspects of the phenomenon. The features of emancipatory question are to create opportunities and the will to engage in social action.”

6.1.2 Evaluation of the research questions

Research questions in both topics - “game as a motivation for lectures” and “game development as a motivation for lectures” - relate to same research goal - “use supportive theory and current computer technology as dual basis to facilitate lecture games in the current technology-rich environment”. The evaluation of the research questions is discussed based on the criteria for a good research question in Section 6.1.1 except the item of ‘answerable’ since it was already addressed in Chapter 5.

The first two questions in topic 1 are exploratory enquiries. For RQ1 - “How can supportive theory be identified to guide the design and evaluation of lecture games?”, understanding of the phenomena was attempted, and useful distinctions were identified to clarify the understanding. In that sense, the first question is suitable for addressing the research themes and exploring various interesting issues within the themes. It is a further step to explore the research goal from the aspect of game as a motivation for lectures. As an exploratory question, it is not vague. It identifies the research context, research aim, research objects, and research orientation in a more relevant specific way than the research goal by itself. The sub-question for RQ1, RQ1.1, “What is supportive theory within the context of lecture games?” is a descriptive enquiry leading to an overview of relevant themes and gathering information. It prepares for sub-question RQ1.2 “How can relevant theoretical framework be setup and applied in the design and evaluation of lecture games?”. RQ1.2 is an exploratory question, which means that it proposes more specific tasks and orientation for the case study. Characteristically to exploratory and descriptive enquiry, a flexible design case study is usually chosen to obtain an answer. As a whole, the relationship between sub-questions is substantively relevant to answering the main question RQ1 in a specific and feasible way, because RQ1.1 is the preparation for RQ1.2. They constitute two steps for exploring RQ1. Therefore, the design of this research question followed the criteria for a good question.

RQ2 - “How can current relevant technology and appropriate peripherals be used to provide various play experiences in new lecture games?” - represents another aspect of the research goal. RQ1 focuses on the supportive theory aspect, while in RQ2 the emphasis is on the technical issues of the Lecture Games project. It is a research issue stemming from the research goal. RQ2 is still an exploratory question since most of technology appeared after 2007, e.g. iPhone, so there was little previous experience and research work to refer to. This research resembles an exploration through a tentative study. RQ2 identifies the important factor of how current popular devices relate to lecture games. Examination of this issue requires an overview of the current situation, addressed by a descriptive research question, RQ2.1 - “What current technology and peripherals are relevant to GBL?”. The specific research orientation is expressed by RQ2.2 - “How can these technologies be integrated into Lecture Games project and evaluated?”. RQ2.1 and RQ2.2 have the attributes similar to RQ1.1 and RQ1.2. It should be noted that a descriptive research question is essential for an unfamiliar or

unexplored topic. A preliminary overview of a field provides a foundation for further research. In this case, RQ2.1 provides a research base for RQ2.2.

The above analysis indicates that RQ1 and RQ2 are at the same level, and the structure of their sub-questions is the same. It is an explicit explanation of the research goal in terms of game as a motivation for lectures. The degree of clarity and specificity of the research goal and the research questions with their sub-RQs is being gradually strengthened. The questions are interconnected and compact, and no third research question is need for this research topic.

In topic 2, RQ3 - “What is game development based learning (GDBL) and what are the researchers’ views of GDBL?” is a *descriptive* question addressing the current status of GDBL and motivation for this research. All of its sub-RQs are also descriptive enquiries. It leads to the definition of GDBL (RQ3.1), explores the main technical issues (RQ3.2), and reveals the current status of GDBL (RQ3.3). All of these sub-questions are specific and clear, and they correspond to the different aspects of RQ3.

RQ3 is the preparation for RQ4: “How can the GDBL be characterized in terms of supportive theory and current technology-rich environment?” In order to answer this question, a literature review and experiments were conducted to collect sufficient data for extracting essential elements from the implementation of GDBL. It should be considered what kind of technical tools could be used, what kind of courses could be taught with GDBL, and what is the final feedback from the students. All of these issues are covered by RQ4.1: “How can the GDBL method be adopted in a software architecture course and what is the students’ perception of the GDBL method?” After gathering experiences in the experiments and the literature review, it was possible to establish a framework to guide the GDBL design at a high level to provide an answer to RQ4.2. All of these questions are linked in a logical sequence. The sub-RQs help to specify the steps of research work, they are sufficient to explain the study, and they are non-trivial questions worthy of the research effort. A summary of evaluation results based on the criteria in Section 6.1.1 is shown in Table 17.

Table 17: Evaluation of research questions

RQ	Type	Clear	Specific	Answerable	Interconnected	Relevant
RQ1	Exploratory	Not vague and easy to understand	Identify the research context, research aim, research objects, and research orientation to a more specific degree than the research goal	Already addressed in Chapter 5	Together with R2 to address research goal	One aspect of research goal in topic 1
RQ1.1	Descriptive				Prepare for RQ1.2	Two steps to answer RQ1
RQ1.2	Exploratory				Answer RQ1	
RQ2	Exploratory				Together with R1 to address research goal	Another aspect of research goal in topic 1
RQ2.1	Descriptive				Prepared for RQ2.2	Two steps to answer RQ2
RQ2.2	Exploratory				Answer RQ2	
RQ3	Descriptive				Prepare for answering R4	Overview of topic 2
RQ3.1	Descriptive				Overview of three aspects of R3	Three steps to answer R3
RQ3.2	Descriptive					
RQ3.3	Descriptive					
RQ4	Exploratory				Address research goal	Results of topic 2
RQ4.1	Exploratory				Prepare for answering R4.2	Another aspect of R4
RQ4.2	Exploratory				Answer R4	Together with R3 and R4.1 to address research goal in topic 2

6.2 Evaluation of Contributions

This section evaluates the contributions of this research with respect to the state of the art. The focus is on the novelty of these contributions and the impact they have in the research area. There are two contributions in topic 1 - game as a motivation for lectures, and three contributions in topic 2 - game development as a motivation for lecture.

6.2.1 Evaluation of contributions in game as a motivation for lectures

The study of topic 1 did not provide a single solution or method for the design of educational games. Instead, the design and implementation process was analyzed to identify the experiences contributing to the theoretical construction of lecture games.

The first contribution is the identification of the research issues, which intersect games and learning in the current technology-rich environment. With respect to the state of the art, few similar studies for systematic theoretical construction in this area took place so far. In this study, the focus is on the theoretical foundation and the impact of current

technology on the design of educational games defined as “supportive theory” and “enabling technology”. Four case studies were carried out to investigate how supportive theory and enabling technology enhanced and enriched the game play and learning process. Since GBL is a relatively new research field, the case studies helped to justify the need for more focus not only on the impact of technology, but also on the construction of theoretical basis for the design and evaluation of lecture games.

The second contribution goes one step further, proposing an analysis chart of applying supportive theory and enabling technology as dual guidance in the study of educational games for lectures. With respect to the state of the art, this is the first conceptual chart describing the intersection of different disciplines in GBL. This research work deals more specifically with lecture games being the intersection between lectures and games. The chart provides a knowledge base for understanding the issues in lecture games design. Further, the relevant theories and criteria were selected to enrich each element in the chart in order to provide more choices in a game play according to the specific game genre and research aims.

6.2.2 Evaluation of contributions to GDBL

The study of topic 2 led to a definition of a new term, GDBL, to denote the method of “learning through game design or game development.” It extends GBL with more variety from game to game development in order to motivate the learning. Since no complete description or reviews in this field exist, it was explored by conducting a systematic literature review and two quasi-experiments.

The third contribution is the identification of a set of research themes in GDBL. It reveals research issues from the perspective of supportive theory and enabling technology. With respect to the state of the art, several other cases of using GDBL in specific courses were examined, but neither of them involved a collection of all similar cases or identified common research elements in GDBL. In this study, these research issues were examined at a high level. All common issues were recognized by the systematic review of specific studies in GDBL and summarized in a systematic way. This is the first attempt to identify the research scope and research issues regarding GDBL in a scientific way. These research issues enrich the theoretical construction of GDBL, extending the GBL field by including GDBL.

The fourth contribution is the identification of the factors affecting the success or failure of usage of GDBL. These impact factors were discovered with the help of experiences gained in experiments and the systematic literature reviews since no prior studies existed with respect to the state of the art. This study should help researchers and educators to identify the weaknesses, to avoid unnecessary losses, and to increase the potential for success in the GDBL design.

The fifth contribution is a framework of linked elements for the design of GDBL. This is a high-level summary of the GDBL process. It explains how each research issue relates to the GDBL field. With respect to the state of the art and the systematic literature review, this is the first conceptual framework to describe GDBL. This

framework could be seen as an introduction to GDBL and it shows relationships among the elements of GDBL. It can help educators to understand this field and guide the usage of GDBL in their courses.

6.3 Evaluation of Validity Threats

Three research methods were used in this research: case study, literature review, and quasi-experiment. Each method is subject to validity threats, which must be considered during the research process.

In regards to the flexible design case study, Maxwell [193] identified the areas of threats to validity in qualitative research as description, interpretation, and theory. All of these are considered as internal threats to validity. The external validity may not be an issue [25]. The strategy for dealing with internal threats to validity can be prolonged involvement, triangulation, peer debriefing/support, member checking, negative case analysis, and audit trail.

Bootes and Beile (2005) [27] created a five-category rubric for evaluating internal validity threats to a literature review. They used this scoring rubric to rate a selected sample of 12 education-related academic dissertations. The external validity threats were discussed in terms of Dybå and Dingsøyr [194] proposition.

For the fixed design quasi-experiment, the threats to validity include construct validity, internal validity, and external validity. They will be discussed individually.

6.3.1 Threats to validity of case studies

Maxwell [193] identified the areas of threats to validity in qualitative research for a case study as: description, interpretation, and theory. Description and interpretation were investigated in this case study. According to Maxwell, *“the main threat to provide a valid description of what we have seen or heard lies in the inaccuracy or incompleteness of the data. The main threat to provide a valid interpretation is that of imposing a framework or meaning on what is happening rather than this occurring or emerging from what we learn during our involvement with the setting.”* Finally, Padgett [195] proposed strategies for dealing with such threats, as shown in Table 18.

Table 18: Strategies for dealing with threats to validity

Strategy	Threat to validity (reactivity)	Researcher bias	Respondent bias
Prolonged involvement	Reduces threat	Increases threat	Reduces threat
Triangulation	Reduces threat	Reduce threat	Reduces threat
Peer debriefing/support	No effect	Reduce threat	No effect
Member checking	Reduces threat	Reduce threat	Reduces threat
Negative case	No effect	Reduce threat	No effect

analysis			
Audit trail	No effect	Reduce threat	No effect

These strategies can serve as the evaluation criteria. In this case study, triangulation, peer debriefing, member checking, and negative case analysis were used since prolonged involvement, which is used in ethnography, was not suitable. In particular, triangulation is a valuable and widely used strategy. Denzin [196] distinguished four types of triangulation. For the first, “data triangulation”, more than one method was used for the data collection, namely: questionnaire, observation, and interviews. For the second, “observer triangulation”, there were more than two observers in the study. For the third, “methodological triangulation”, both quantitative and qualitative approaches were used. For the fourth, “theory triangulation”, more than one theory was used to support the case study design. The “peer debriefing and support” and “member checking” were applied in some instances in this case study. The “negative case analysis” was not used, but both positive and negative experiences in this study were described in detail in the published papers. Audit trail was not suitable for this case study research.

According to [25], the external validity, construct validity, and conclusion validity do not relate to case studies.

6.3.2 Threats to validity of literature review

Internal validity

As mentioned above, Bootes and Beile [27] created a five-category rubric for evaluating a literature review, shown in Table 19: coverage, synthesis, methodology, significance, and rhetoric.

Table 19: Boote and Beile’s literature review criteria

Category	Criterion
1 Coverage	A. Justified criteria for inclusion and exclusion from review.
2. Synthesis	B. Distinguished between what has been done in the field and what needs to be done.
	C. Placed the topic or problem in the broader scholarly literature.
	D. Placed the research in the historical context of the field.
	E. Acquired and enhanced the subject vocabulary.
	F. Articulated important variables and phenomena relevant to the topic.
3. Methodology	G. Synthesized and gained a new perspective on the literature.
	H. Identified the main methodologies and research techniques, which have been used in the field, and their advantages and disadvantages.
4. Significance	I. Related ideas and theories in the field to research methodologies.
	J. Rationalized the practical significance of the research problem.
5. Rhetoric	K. Rationalized the scholarly significance of the problem.
	L. Was written with a coherent, clear structure, which supported the

	review
--	--------

The literature review was conducted strictly following the process in Table 19. The literature review lasted three months and the whole process was described in detail in Section 4.3. The results of the review were presented with references to the literature. Search keywords were updated and modified in order to gather more relevant articles. Relevant articles from other search results were allowed as alternative sources. Searching the references cited in the relevant articles offered multiple data sources. The interpretation bias or subjectivity was removed by having two third parties read each paper before including it in the review.

External Validity

Dybå and Dingsøyrr [194] suggested that in the context of a systematic review, external validity is concerned with whether or not the study is posing an appropriate research question. They also said that the assessment depends on the purpose for which the study is to be used. External validity is closely connected with the generalizability and applicability of the study's findings. The framework shown in Figure 17 was extracted from a systematic literature review following a detailed design process, and similar conclusions were also found in GDBL experiments. In addition, a list of the articles included in the review and the conclusions reached were provided to readers in Microsoft Word format, thus the proposed interpretation could be checked.

Construct validity and conclusion validity do not relate to literature reviews.

6.3.3 Threats to validity of quasi-experiment

Internal Validity

For quasi-experiment as a fixed design, there are a few threats to its internal validity [197], e.g. testing, maturation or selection of subjects. Any of them may affect to the experimental evidence that supports the conclusion. In an experiment, it concerns "*the validity of inferences about whether observed co variation between A (the presumed treatment) and B (the presumed outcome) reflects a causal relationship from A to B as those variables were manipulated or measured*" [198]. So the conditions for this causality include that A and B must be related and there should not be other confounding, unwanted extraneous factors affecting the changes of B. In the light of such criteria, researchers can use experimental design to reduce the threats to this internal validity.

There are two main internal validity threats to this evaluation. The first internal threat is that the sample of the two groups used in the evaluation was not randomized. The students were allowed to choose freely either a game project or a non-game project. It does not appear that one specific type of student chose one project over the other, which could harm the evaluation results. The collected data did not reveal any major differences between the two groups. The second internal threat would exist if there were differences in how the students conducted the project depending on the domain chosen.

In this quasi-experiment, the students had to go through exactly the same phases in the project and deliver exactly the same documents based on the same document templates.

External Validity

External design validity denotes whether conclusions from an experiment can be generalized and if they are valid for cases or groups beyond the experiment at hand. Typical threats to external validity include pre-test effects, post-test effects, experimental setting or the subject's knowledge [199]. In relation to an experiment, the issue of external validity concerns "*whether a causal relationship holds (1) for variations in persons, settings, treatments, and outcomes that were in the experiment and (2) for persons, settings, treatments, and outcomes that were not in the experiment*" [198].

The results of the quasi-experiments are most relevant for teachers who consider introduction of game projects in their software architecture course. Further, the results are also relevant for teachers who want to introduce game projects in SE and CS courses, as many of these courses have similar characteristics. A limitation of this study is that the subjects in the evaluation are CS or SE students who have completed their first three years of study. It is not evident that the results are valid for students without any or less than three-year background in CS or SE.

Construct Validity

Construct validity concerns "*the degree to which inferences are warranted, from (1) the observed persons, settings, and cause and effect operations included in a study to (2) the constructs that these instances might represent. The question, therefore, is whether the sampling particulars of a study can be defended as measures of general constructs*" [198].

In the evaluation of using a game project in a software architecture course, the research goal was to investigate whether a game development project was suited for teaching this course. The GQM approach was chosen to decompose this goal into four research questions with supporting metrics, described in Section 4.3.3. The answers to these four research questions are based on the data sources and metrics collected in the software architecture course run at NTNU. It cannot be claimed that the selected data sources and metrics in the evaluation support all the conclusions, but they are all strong indicators contributing to interpretation of the differences between the two project types. In the evaluation, various methods were used for comparing the results. The choice of methods was based on the best way of describing and visualizing the differences between the two groups using the available data.

7 Conclusions

This chapter concludes this thesis based on the empirical data that were gathered. The conclusions confirm the validity of the idea of combining supportive theory and technology as dual basis to study lecture games, especially in the construction of the theoretical foundation for the design of lecture games. This project contributes to this aspect of the field further and thus constitutes a contribution to the GBL field.

7.1 Contributions

Five contributions were presented in this thesis. For an interdisciplinary field, it is important to achieve a good understanding of the research area and to position the research in respect to the state of the art. The contributions are based on the evaluation of the case studies in topic 1 as well as the literature review and the experiments in topic 2.

Regarding the contributions related to game as a motivation for lectures, a weakness was identified in the theoretical foundation of GBL, namely that educational games are mainly designed based on designer's personal ideas and experiences. The research goal was to enhance the theoretical foundation of the design process for GBL. The study started by taking concepts from pedagogical theory, then adding game design theory and other criteria, which can benefit the educational game design. This has been an interesting and encouraging process, which contributed to the theoretical foundation of GBL. The evaluation data of the case studies show that the games used as an aid to the formal lectures have a positive effect on students' motivation for exercises as well as enhancing the socialization of students in the lectures. However, it also shows a negative feedback from students, with excessive focus on the games, and not substantially increasing the attendance rate in the classrooms. This deserves further research and improvement. A practical process for the design of educational games was proposed based on experiences in the four case studies and an analysis chart for this design process was constructed. Researchers and educators can use these contributions to analyze various aspects of the design of lecture games, to remove weaknesses, and to identify the ignored parts when designing a lecture game. In a specific situation, many features of the supportive theory can come into play, e.g. a game concept or game genre may be a constraint to applying all aspects of supportive theories, where only some parts of them fit a certain lecture game. Additionally, designers who lack understanding of the supportive theory or lecture content may base the game design on wrong premises. Therefore, this contribution

can reduce the possibility of failure, although it cannot guarantee the success of every lecture game design process.

Regarding the contribution to the topic of game development as a motivation for lectures, the term “Game Development Based Learning”, GDBL, was proposed to define the research scope and its methods. With the unified research goal for both topics, the focus was on the supportive theory construction and technical issues in using GDBL. Two quasi-experiments were carried out and a new method, GDBL, proposed to teach software architecture. Based on the experiments, it was found that students could benefit from putting more effort into project exercises, and better understanding of course content. On a negative side, additional time must be spent on game development, which might not be directly related to the course aim. Further, a literature survey was performed to investigate possible fields where GDBL can be used. Through a systematic literature survey, it was found that GDBL could be used in fields other than software engineering and computer science, e.g. literacy education in primary school. Through combining the experiences in the experiments and the literature review, a framework was proposed to guide the design of GDBL. In addition, the supportive theory was compiled to support the design of GDBL, the criteria for selecting GDFs were proposed, and the GDBL output impact factors were identified as the final contribution.

7.2 Limitations and Future Work

This section briefly discusses the limitations of this work, and then it outlines how future work can be carried out based on this PhD research.

For the game as a motivation for lectures, the case studies were conducted to answer the research questions. Although four cases should be sufficient to find most of the answers, there are still limitations, which cannot be avoided.

Each research question is supported by two relevant case studies. The results are summarized based on these case studies. Most of the generalization for answering research questions is based on the experiences in the case studies and the existing theories. There may exist other circumstances not encountered in the case studies and, thus, ignored due to limited experiences. More case studies should be carried out to increase the precision and quality of the conclusions.

To reduce the limitations of the research on “game as a motivation for lectures”, further work may include:

- *Testing lecture games in various game genres with more supportive theories:* In the case studies, collaborative learning and intrinsic motivation were mainly used to guide a multiplayer quiz-based concept, and the concept was evaluated with the GameFlow model. However, other game genres with suitable learning strategies or game design theory support should be explored.
- *Exploring pervasive learning games:* Game technology develops very fast, and provides more ways to enhance playing experiences, such as pervasive learning games. One of the case studies, ACG, was a tentative study to integrate pervasive technology in a learning game, but it was far from exhausting the full potential of

pervasive learning games. It appears that this new research area is quite interesting, and thus deserves more exploration.

- *Systematic literature review*: GBL is a relatively large and new field, and there is no systematic literature review in this area. One serious obstacle for this kind review is the time and effort required filtering out thousands of irrelevant articles. A possible solution is to narrow down the topic to a specific discipline, such as physical education or primary education. Such work would help to form more concrete conclusions as the topic is narrowed down.
- *Establishing closer collaboration with related disciplines*. GBL is a cross-disciplinary field and the theoretical foundation for lecture games, as it was presented in this thesis, comes from related disciplines, e.g. pedagogical field and game design field. In order to apply these supportive theories correctly in design of games, it would be beneficial to establish a closer collaboration with the researchers in these fields and to get a more in-depth understanding of lecture games.

The purpose of all the above work was to enrich the theoretical foundation of GBL in order to formulate a systematic theory to guide the educational game design.

For the research of game development as a motivation for lectures, two quasi-experiments and a systematic literature review were carried out. Even though a systematic literature review can enhance the generalization of the results, there are still some limitations in this study.

With the experience of conducting this literature review, the following limitations were identified: (a) The scope of data search and collection from four scientific search engines is relatively limited; (b) Due to the fact that game research field is newer than other traditional research fields, the number of articles with empirical data is still limited in the survey. It may affect the evaluation results in terms of generalization; (c) Some topics deserve further discussion, for instance cross-disciplinary courses. A game development course covers programming and art of design, and a machine course focuses on 3D-animation and movie creation. Both of them could be further explored since the GDFs play different roles in these two courses. In the game development course, the GDF is used as the main tool for development, while in the machinima course the GDF is an innovative auxiliary tool.

In the experiments conducted for this study, the exact experimental process was not followed and this is why they were categorized as quasi-experiments. One explanation is that the course situation is not a real experiment environment, and it cannot be assumed that the students will follow the experimental process, e.g. with half of them choosing game project and half of them choosing the non-game project. In the actual situation, they were free to choose the project they liked, making the sample of subjects not random.

To reduce the limitations in research of “game development as a motivation for lectures”, further work could be enhanced in the following aspects:

- *Finding a suitable environment allowing for a controlled experiment, strictly following the experimental design to control all inputs of the experiments*: This

could increase the confidence in the results and reduce other negative factors affecting the output.

- *Setting different goals for the experiments:* For the double stimulus experiment, there are different outcomes depending on levels of education. In the experiments in this study, it was found that GDBL could have a positive effect in higher education. The difference of outcomes depends on what kinds of tools are used in the course. This means that researchers can use a different combination of GDF/non-GDF tools/no-tools, and a different educational level: primary education, secondary education, or higher education. For instance, it would be interesting to explore the combination of GDF/no-tools in primary education to compare the results of GDBL and non-GDBL.
- *Adding more resources to systematic literature reviews:* The systematic literature review covered papers up to the year 2010. Since that time, more articles have been published in this area. Inclusion of newer data would enhance the summarized conclusions regarding GDBL.
- *Enriching the GDF resource pools:* Teachers may need new GDFs for a specific course or find more GDFs, which can be used for GDBL. The continuous development of hardware and software technology means that old GDFs will be replaced.
- *Finding new fields for GDBL:* As the survey of the systematic literature review showed, GDBL can be applied to fields other than software engineering and computer science, and beyond higher education. There exist cases where GDBL was used in primary education or literacy education. These are indicators that GDBL can be applied in more fields.

GDBL is a new and original definition of a term as well as a research field established through this thesis. Further work should mainly focus on exploring many opportunities within GDBL including more evaluation data to improve this approach in the future.

7.3 Concluding Remarks

Finally, the contributions of this research add to the knowledge base in the interdisciplinary research area of *games and learning*. Researchers can get an overview of this area based on the investigation presented in this thesis and derive research directions from the research issues identified here. This study has shown that both games and game development do have the potential power to help students learn various curricula. It is hoped that this study will provide a useful guidance to educators, practitioners, and researchers in the area of GBL, including GDBL.

8 References

- [1] M. Prensky, "Digital game-based learning," *Computers in entertainment*, vol. 1, pp. 21- 24, 2003.
- [2] R. Owston, *et al.*, "Computer game development as a literacy activity," *Computers & Education*, vol. 53, pp. 977-989, 2009.
- [3] B. A. Myers, "A brief history of human-computer interaction technology," *Interactions*, vol. 5, p. 44, 1998.
- [4] J. Kirriemuir and A. McFarlane, "Literature review in games and learning," Report 8.2004.
- [5] S. I. d. Freitas, "Using games and simulations for supporting learning," *Learning, Media and Technology* vol. 31, 2006.
- [6] T. Vold and S. McCallum, "Gamers and learning," 2011, pp. 1-4.
- [7] E. F. Provenzo, *Video kids: Making sense of Nintendo*, 1991.
- [8] R. F. Bowman, "A Pac-Man theory of motivation. Tactical implications for classroom instruction.," *Educational Technology* 22(9), 14-17., 1982.
- [9] J. E. D. Driskell, D.J., "Microcomputer videogame based training.," *Educational Technology*, 24(2), 11-15., 1984.
- [10] G. W. Bracey, "The bright future of integrated learning systems.," *Educational Technology*, 32(9), 60-62., 1992.
- [11] S. De Freitas and M. Griffiths, "Online gaming as an educational tool in learning and training," *British Journal of Educational Technology*, vol. 38, pp. 535-537, 2007.
- [12] W. N. Holmes, "The myth of the educational computer," *Computer*, vol. 32, p. 36, 1999.
- [13] K. Squire, "Cultural Framing of Computer/Video Games," *the international journal of computer game research*, vol. 2, 2002.
- [14] R. Paharia. (Sep. 2012). *Who coined the term "gamification"?* *Quora*, . Available: <http://goo.gl/CvcMs>
- [15] M. McDonald, *et al.*, *Using Productivity Games to Prevent Defects*: Microsoft Press, Redmond, 2008.
- [16] M. V. Grace and J. Hall, "Projecting Surveillance Entertainment.," in *Presentation, ETech*, San Diego, CA, 2008.
- [17] D. Takahashi. (2008, *Funware's threat to the traditional video game industry.*) Available: Venturebeat <http://goo.gl/O9lSq>
- [18] J. Ferrara, "Playful Design. Creating Game Experiences in Everyday Interfaces," New York.
- [19] A. Dignan, "Game Frame: Using Games as a Strategy for Success ,," 2011.
- [20] S. Priebsch. (2011, *The Game Layer on Top of the World.* Available:) <http://goo.gl/DnwBH>
- [21] D. Helgason. (2010, *Trends. Unity Technologies Blog*,). Available: <http://goo.gl/AZ4vm>.
- [22] G. Zicherman. (2011, *A Long Engagement and a Shotgun Wedding: Why Engagement is the Power Metric of the Decade.* Available: <http://goo.gl/jlaO0>.
- [23] S. Deterding, *et al.*, "From game design elements to gamefulness: defining "gamification"," presented at the Proceedings of the 15th International

- Academic MindTrek Conference: Envisioning Future Media Environments, Tampere, Finland, 2011.
- [24] S. McCALLUM, "Gamification and serious games for personalized health," *Phealth 2012: Proceedings of the 9th International Conference on Wearable Micro and Nano Technologies for Personalized Health*, p. 85, 2012.
- [25] C. Robson, *Real world research*, 1997.
- [26] B. J. Oates, *Researching Information Systems and Computing*: SAGE Publications Ltd, 2006.
- [27] D. N. Boote and P. Beile, "Scholars before researchers: On the centrality of the dissertation literature review in research preparation.," *Educational Researcher*, vol. 34(6), pp. 3-15, 2005.
- [28] A. I. Wang, *et al.*, "LECTURE QUIZ - A Mobile Game Concept for Lectures," presented at the In 11th IASTED International Conference on Software Engineering and Application (SEA 2007), 2007.
- [29] T. W. Malone, & Lepper, M. R., "Making learning fun: A taxonomy of intrinsic motivation for learning," *Aptitude, Learning, and instruction.* , vol. Volume 3: conative and effective process analyses, pp. (223-253), 1987.
- [30] M. Zyda, "From visual simulation to virtual reality to games," *Computer.*, vol. 38(9), pp. 25-32, 2005.
- [31] D. Michael and S. Chen, "Serious games: Games that educate, train, and inform," 2006.
- [32] K. Corti. (2006, *Games-based Learning; a serious business application. PIXELearning Limited.* Available: www.pixelelearning.com/docs/games_basedlearning_pixelelearning.pdf
- [33] T. Susi, *et al.*, "Serious Games – An Overview," in *Technical Report 2007-02-05*.
- [34] S. Egenfeldt-Nielsen, *et al.*, *Understanding video games : the essential introduction*. New York: Routledge, 2008.
- [35] B. Sawyer and P. Smith., "Serious Games taxonomy," 2008.
- [36] J. P. Gee, "What video games have to teach us about learning and literacy," *Computers in entertainment*, vol. 1, p. 20, 2003.
- [37] L. Natvig, *et al.*, "Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals," in *Proceedings of the 2004 Frontiers in Education Conference*, 2004.
- [38] R. F. Lyvers, "A unique instructional tool for visualizing equipotentials and its use in an introductory fields course," *IEEE transactions on education*, vol. 36, p. 237, 1993.
- [39] R. S. Limited. (2008, 29th, March). *Buzz!: The Schools Quiz*. Available: <http://www.relentless.co.uk/games/buzz-schools-quiz>
- [40] T. Ø. Alf Inge Wang, and Ole Kristian Mørch-Storstein., "LECTURE QUIZ - A Mobile Game Concept for Lectures.," presented at the In 11th IASTED International Conference on Software Engineering and Application (SEA 2007),, 2007.
- [41] L. Arts. 8th, March, 2010). *Lucas Learning*. Available: <http://www.lucaslearning.com/edu/lesson.htm>
- [42] M. MIT. 8th, March, 2010). *Games-to-Teach*. Available: <http://www.educationarcade.org/gtt/index.html>

- [43] M. S. El-Nasr, "Learning through game modding," *Computers in entertainment*, vol. 4, 2006.
- [44] L. Joe and S. Amber, "Teaching game programming using XNA," presented at the Proceedings of the 13th annual conference on Innovation and technology in computer science education, Madrid, Spain, 2008.
- [45] M. Overmars, "Teaching computer science through game design," *Computer*, vol. 37, p. 81, 2004.
- [46] A. I. Wang, "An application of a game development framework in higher education," *International Journal of Computer Games Technology*, vol. 2009, p. 1, 2009.
- [47] W. K. Chen, "Teaching object-oriented programming laboratory with computer game programming," *IEEE transactions on education*, vol. 50, p. 197, 2007.
- [48] J. Ryoo, "Teaching object-oriented software engineering through problem-based learning in the context of game design," in *21st Conference on Software Engineering Education and Training*, 2008, p. 137.
- [49] S. McCallum, *et al.*, "Creating a Computer Game Design Course," *Proceedings of the New Zealand Game Developers Conference, (NZGDC)*, 2004.
- [50] B. Brown, MacColl, I., Chalmers, M., Galani, A., Randell, C., and Steed, A., "Lessons from the lighthouse: collaboration in a shared mixed reality system.," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Florida, USA, April 05 - 10, 2003, 2003, pp. p. 577-584.
- [51] M. J. McAlister and X. Peng Hui, "Using a PDA for mobile learning," in *Wireless and Mobile Technologies in Education, 2005. WMTE 2005. IEEE International Workshop on*, 2005, p. 3 pp.
- [52] G. Schwabe and C. Göth, "Mobile learning with a mobile game: design and motivational effects," *Journal of Computer Assisted Learning*, vol. 21, pp. 204-216, 2005.
- [53] R. E. Grinter, Aoki, P. M., Szymanski, M. H., Thornton, J. D., Woodruff, A., and Hurst, A. , "Revisiting the visit:: understanding how technology can shape the museum visit. ," in *Proc. ACM CSCW'02, 2002*, 2002, pp. p. 146-155
- [54] M. J. Dondlinger, "Educational Video Game Design: A Review of the Literature," *Journal of Applied Educational Technology*, vol. Volume 4, Number 1, 2007.
- [55] M. D. Dickey, ""Ninja Looting" for instructional design: The design challenges of creating a game- based learning environment," in *the ACM SIGGRAPH 2006 conference*, Boston, 2006.
- [56] C. Dede, *et al.*, "Design-based research strategies for studying situated learning in a multi-user virtual environment," in *the 6th international conference on Learning sciences*, Santa Monica, CA, 2004.
- [57] M. D. Dickey, "Three-dimensional virtual worlds and distance learning: Two case studies of Active Worlds as a medium for distance education," *British Journal of Educational Technology*, vol. 36(3), pp. 439-451, 2005.
- [58] K. Schrier, "Using augmented reality games to teach 21st century skills. ," in *the ACM SIGGRAPH 2006 Conference*, Boston, 2006.
- [59] M. Corbit, "Moving into cyberspace: Game worlds for learning. ," *Knowledge Quest*, , vol. 34(1), pp. 18-22, 2005.

- [60] J. Robertson and J. Good, "Story creation in virtual game worlds. ," *Communications of the ACM*, vol. 48(1), pp. 61-65, 2005.
- [61] J. Robertson, *et al.*, "Children's narrative development through computer game authoring: The untapped world of video games," in *the 2004 Conference on Interaction Design and Children: Building a Community*, Vienna, Austria., 2004.
- [62] B. Steiner, *et al.*, "When play works: Turning game-playing into learning.," in *the 2006 Conference on Interaction Design and Children*, Tampere, Finland, 2006.
- [63] R. Halverson, *et al.*, " Theorizing games in/and education," in *the 7th international conference on Learning Sciences*, Bloomington, IN., 2006.
- [64] L. Lunce, "Simulations: Bringing the benefits of situated learning to the traditional classroom.," *Journal of Applied Educational Technology*, vol. 3(1), pp. 37-45, 2006.
- [65] E. Klopfer and S. Yoon, "Developing games and simulations for today and tomorrow's tech savvy youth.," *Tech Trends*, vol. 49(3), pp. 33-41, 2005.
- [66] R. Schroeder, "Online Learning: Hyper Linking Higher Education to the Future," 2006.
- [67] S. K. Reed, "Cognition: Theories and application (8th ed.). ," *Belmont, CA: Wadsworth Cengage Learning.*, 2010.
- [68] A. Paivio, *Imagery and verbal processes*. New York: Holt, Rinehart, and Winston., 1971.
- [69] R. E. Mayer and M. R., "A Cognitive Theory of Multimedia Learning: Implications for Design Principles," 1998.
- [70] R. Moreno and R. Mayer, "Cognitive principles of multimedia learning: The role of modality and contiguity," *Journal of Educational Psychology* vol. 91 (2), pp. 358–368, 1999.
- [71] A. D. Baddeley and H. G.J., "Working Memory," *The psychology of learning and motivation: advances in research and theory.* , pp. pp. 47–89., 1974.
- [72] R. Kop and A. Hill., "Connectivism: Learning theory of the future or vestige of the past? ," *The International Review of Research in Open and Distance Learning*, vol. Vol 9, No 3, 2008.
- [73] S. Downes. (Accessed on 31th, Aug. 2012) *What Connectivism Is?* Available: <http://halfanhour.blogspot.no/2007/02/what-connectivism-is.html>
- [74] L. Dirckinck-Holmfeld, *et al.*, "Analysing Networked Learning Practices in Higher Education and Continuing Professional Development. ," 2009.
- [75] W. M. Thomas, "What makes things fun to learn? heuristics for designing instructional computer games," presented at the Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, Palo Alto, California, United States, 1980.
- [76] L. S. Vygotskiï, *Mind in society: The development of higher psychological processes*, 1978.
- [77] J. Thomas, *A review of research on project-based learning*. Novato, CA: The Buck Institute for Education, 2000.
- [78] A. Nickel and T. Barnes, "Games for CS education: computer-supported collaborative learning and multiplayer games," presented at the Proceedings of

- the Fifth International Conference on the Foundations of Digital Games, Monterey, California, 2010.
- [79] B. Brown and M. Bell, "CSCW at play: there as a collaborative virtual environment," presented at the Proceedings of the 2004 ACM conference on Computer supported cooperative work, Chicago, Illinois, USA, 2004.
- [80] S. Bardzell, *et al.*, "Blissfully productive: grouping and cooperation in world of warcraft instance runs," presented at the Proceedings of the 2008 ACM conference on Computer supported cooperative work, San Diego, CA, USA, 2008.
- [81] B. Nardi and J. Harris, "Strangers and friends: collaborative play in world of warcraft," presented at the Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work, Banff, Alberta, Canada, 2006.
- [82] T.Manninen and T.Korva, "Designing Puzzle for Collaborative Gaming Experience - CASE: eScape," in *DiGRA 2005 Conference: Changing Views - Words in play 2005*, Vancouver, Canada, 2005.
- [83] C. Crawford, *The Art of Computer Game Design: Osborne/McGraw Hill*, 1982.
- [84] A. Lund and I. Rasmussen, "The right tool for the wrong task? Match and mismatch between first and second stimulus in double stimulation," *International Journal of Computer-Supported Collaborative Learning*, vol. 3, pp. 387-412, 2008.
- [85] V. Kaptelinin, *Acting With Technology*, 2006.
- [86] J. S. Krajcik, *et al.*, "A collaborative model for helping middle-grade science teachers learn project-based instruction," *The Elementary School Journal*, vol. 94, pp. 483-497, 1994.
- [87] Ronald W. Marx, *et al.*, "Enacting project-based science: Experiences of four middle grade teachers," *Elementary School Journal*, vol. 94, pp. 517-538, 1994.
- [88] E. L. O. Bound, "A design for comprehensive school reform.," *Expeditionary Learning Outward Bound*, 1999.
- [89] J. Huang, "Improving undergraduates' teamwork skills by adapting project-based learning methodology," in *5th International Conference on Computer Science and Education (ICCSE)*, 2010, pp. 652-655.
- [90] A. Garrido, *et al.*, "Using graphics: motivating students in a C++ programming introductory course," in *EAAEIE Annual Conference*, 2009, pp. 1-6.
- [91] P. Sweetser, "GameFlow: a model for evaluating player enjoyment in games," *Computers in entertainment*, vol. 3, p. 3, 2005.
- [92] M. CSIKSZENTMIHALYI, "Flow: The Psychology of Optimal Experience.," 1990.
- [93] F.-L. Fu, *et al.*, "EGameFlow: A scale to measure learners' enjoyment of e-learning games," *Computers & Education*, vol. 52, pp. 101-112, 2009.
- [94] S. M. Fisch, "Making educational computer games "educational"," in *the 2005 Conference on Interaction design and children*, Boulder, CO., 2005.
- [95] A. Waraich, "Using narrative as a motivating device to teach binary arithmetic and logic gates," in *the 9th annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, Leeds, United Kingdom., 2004.
- [96] A. Amory, *et al.*, "The use of computer games as an educational tool: Identification of appropriate game types and game elements," *British Journal of Educational Technology*, vol. 30(4), pp. 311-321, 1999.

- [97] G. Denis and P. Jouvelot, "Motivation-driven educational game design: applying best practices to music education," in *the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, Valencia, Spain, 2005.
- [98] M. Jennings, "Best practices in corporate training and the role of aesthetics: Interviews with eight experts. ," in *the 2001 ACM SIGCPR Conference on Computer Personnel Research*, San Diego, CA., 2001.
- [99] J. Radoff, *Game On: Energize Your Business with Social Media Games*, April 2011.
- [100] R. Bartle, *Designing Virtual Worlds*, 2003.
- [101] J. Radoff, "Game Player Motivations. ," May 2011.
- [102] P. Alexander, *et al.*, "Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions," *Contemporary Educational Psychology*, January 01, 2000.
- [103] R. J. Vallerand, "The Academic Motivation Scale: A Measure of Intrinsic, Extrinsic, and Amotivation in Education," *Educational and Psychological Measurement*, March 08, 1993.
- [104] S. Harter, "A New Self-Report Scale of Intrinsic versus Extrinsic Orientation in the Classroom: Motivational and Informational Components," 1981.
- [105] M. L. Diana Cordova, "Intrinsic Motivation and the Process of Learning: Beneficial Effects of Contextualization, Personalization, and Choice," 1995.
- [106] A.-V. Nicoletta and W. Kelly, "SMILE: an immersive learning game for deaf and hearing children," presented at the ACM SIGGRAPH 2007 educators program, San Diego, California, 2007.
- [107] M. R. Lepper, "Motivational considerations in the study of instruction," *Cognition and Instruction*, pp. 289-309, 1988.
- [108] S. Barab, *et al.*, "Making learning fun: Quest Atlantis, a game without guns," *Educational Technology Research and Development*, vol. 53, pp. 86-107, 2005.
- [109] B. Shelley. (2006, *Guidelines for developing successful games*. Available: http://www.gamasutra.com/view/feature/3041/guidelines_for_developing_.php
- [110] C. Chuck, "An interpreted demonstration of computer game design," presented at the CHI 98 conference summary on Human factors in computing systems, Los Angeles, California, United States, 1998.
- [111] S. Ben, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*: Addison-Wesley Longman Publishing Co., Inc., 1997.
- [112] P. Bickford, "Interface design : The Art of developing easy - to - use software " 1997.
- [113] R. Maria, "Learning by doing and learning through play: an exploration of interactivity in virtual environments for children," *Comput. Entertain.*, vol. 2, pp. 10-10, 2004.
- [114] S. d. Freitas and T. Neumann, "The use of 'exploratory learning' for supporting immersive learning in virtual environments," *Computers & Education*, vol. 52, pp. 343-352, 2009.
- [115] C. Wohlin, *Experimentation in software engineering: an introduction*. Boston: Kluwer, 2000.
- [116] V. Basili, "Software modeling and measurement: the Goal/Question/Metric paradigm," 1992.

- [117] P. W. Jordan, *et al.*, *Usability Evaluation in Industry, chapter SUS - A quick and dirty usability scale*: CRC Press, 1996.
- [118] G. Lukas, "Uses of the LOGO programming language in undergraduate instruction," presented at the Proceedings of the ACM annual conference - Volume 2, Boston, Massachusetts, United States, 1972.
- [119] M. Micco, "An undergraduate curriculum in expert systems design or knowledge engineering," presented at the Proceedings of the 15th annual conference on Computer Science, St. Louis, Missouri, United States, 1987.
- [120] T. Phit-Huan, *et al.*, "Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception," in *International Conference on Computer Technology and Development, 2009(ICCTD '09)*, 2009, pp. 42-46.
- [121] J. W. Anastas and M. L. MacDonald, *Research design for social work and the human services*: New York: Lexington Books, 1994.
- [122] M. Hammersley, "The Relevance of Qualitative Research," *Oxford Review of Education*, vol. Vol. 26, No. 3/4,, pp. 393-405, 2000.
- [123] LeCompte, *et al.*, "Editor's introduction," *Review of Educational Research*, vol. 73(2), pp. 123-124, 2003.
- [124] H. M. Cooper, *The integrative research review: A systematic approach* vol. (Vol. 2). : Beverly Hills, CA: Sage., 1984.
- [125] Pelosi M.K., *et al.*, *Doing statistics with excel 97*: Chichester:Wiley, 1998.
- [126] B. F. Crabtree and W. F. Miller, *A Template Approach to Text Analysis: Developing and Using Codebooks.* : Newbury Park, CA, Sage Publications., 1992.
- [127] J. W. Drisko, "Qualitative data analysis: it's not just anything goes!," Charleston, SC: Society for Social Eork and Research Annual Conference, 2000.
- [128] M. Papastergiou, "Exploring the potential of computer and video games for health and physical education: A literature review," *Computers & Education*, vol. 53, pp. 603-622, 2009.
- [129] R. Hays, "The effectiveness of instructional games: A literature review and discussion.," Technical report 2005-004. Orlando, FL: Naval Air Warfare Center, Training Systems Division.2005.
- [130] A. Mitchell, *The use of computer and video games for learning: A review of the literature*, 2004.
- [131] J. P. Higgins and S. Green, *Front Matter*: John Wiley & Sons, Ltd, 2008.
- [132] K. S. Khan, *et al.*, *Undertaking systematic reviews of research on effectiveness: CRD's guidance for carrying out or commissioning reviews*: CRD report, Number 4, second ed., NHS centre for revies and lissemination, University of York, 2001.
- [133] Y. En, *et al.*, "Enhancing software engineering education using teaching aids in 3-D online virtual worlds," in *37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, (FIE '07) 2007*, pp. T1E-8-T1E-13.
- [134] B. Wu, *et al.*, "Experiences from Implementing an Educational MMORPG," in *International IEEE Consumer Electronics Society's Games Innovations Conference, 2010. GIC 2010.*, ed: IEEE conference proceedings, 2010.

- [135] A. Baker, *et al.*, "Problems and Programmers: an educational software engineering card game," in *Proceedings. 25th International Conference on Software Engineering*, 2003, pp. 614-619.
- [136] F. McCown, "Teaching a game programming class for the first time: tutorial presentation," *Journal of Computing Sciences in Colleges*, vol. 25, pp. 131-132, 2010.
- [137] C. Leska and J. Rabung, "Learning O-O concepts in CS I using game projects," *SIGCSE Bull.*, vol. 36, pp. 237-237, 2004.
- [138] E. Ferguson, *et al.*, "Video game development using XNA game studio and C#.Net," *Journal of Computing Sciences in Colleges*, vol. 23, pp. 186-188, 2008.
- [139] R. H. Seidman, "Alice first: 3D interactive game programming," *SIGCSE Bull.*, vol. 41, pp. 345-345, 2009.
- [140] F. Xiang, *et al.*, "Work in progress; A sandbox model for teaching entrepreneurship," in *2010 IEEE Frontiers in Education Conference*, 2010, pp. F2C-1-F2C-2.
- [141] M. Kolling, "Greenfoot: introduction to Java with games and simulations," *Journal of Computing Sciences in Colleges*, vol. 25, pp. 117-117, 2010.
- [142] A. Azemi and L. L. Pauley, "Teaching the introductory computer programming course for engineers using Matlab," in *38th Annual Frontiers in Education Conference (FIE 2008)*, 2008, pp. T3B-1-T3B-23.
- [143] A. Pardo and C. D. Kloos, "Deploying interactive e-labs for a course on operating systems," presented at the Proceedings of the 6th conference on Information technology education, Newark, NJ, USA, 2005.
- [144] P. Rooney, *et al.*, "Cross-Disciplinary Approaches for Developing Serious Games in Higher Education," in *Conference in Games and Virtual Worlds for Serious Applications, 2009 (VS-GAMES '09)* 2009, pp. 161-165.
- [145] A. W. B. Furtado, *et al.*, "Cegadef: a collaborative educational game development framework," presented at the Proceedings of the 2003 conference on Interaction design and children, Preston, England, 2003.
- [146] H. C. Yang, "A General Framework for Automatically Creating Games for Learning," in *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, 2005.
- [147] K. Kardan, "Computer role-playing games as a vehicle for teaching history, culture, and language," presented at the Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames, Boston, Massachusetts, 2006.
- [148] S. Arakawa and S. Yukita, "An Effective Agile Teaching Environment for Java Programming Courses," in *36th Annual Frontiers in Education Conference*, 2006, pp. 13-18.
- [149] W. W. Y. Lau, *et al.*, "Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students," *SIGCSE Bull.*, vol. 41, pp. 504-508, 2009.
- [150] S. v. Delden, "Industrial robotic game playing: an AI course," *J. Comput. Small Coll.*, vol. 25, pp. 134-142, 2010.
- [151] T. E. Daniels, "Integrating engagement and first year problem solving using game controller technology," in *Frontiers in Education Conference, 2009. FIE '09. 39th IEEE*, 2009, pp. 1-6.

- [152] A. Striegel and D. Van Bruggen, "Work in progress; Development of a HCI course on the Microsoft Surface," in *2010 IEEE Frontiers in Education Conference*, 2010, pp. S3F-1-S3F-6.
- [153] A. Wang, "Interactive Game Development with a Projector-Camera System," in *Technologies for E-Learning and Digital Entertainment*. vol. 5093, ed: Springer Berlin / Heidelberg, 2008, pp. 535-543.
- [154] J. Dempsey, *et al.*, "The instructional gaming literature: Implications and 99 sources. ," Technical report no. 96-1. University of South Alabama, College of Education.1996.
- [155] J. Dempsey, *et al.*, "Since Malone's theory of intrinsically motivating instruction: What's the score in the gaming literature?," *Journal of Educational Technology Systems*, 22(2), 173-183., 1993-1994.
- [156] Len Bass, *et al.*, *Software architecture in practice: Second Edition*: Addison-Wesley Professional, 2003.
- [157] J. O. Coplien, *Software Design Patterns: Common Questions and Answers*. . New York, : Cambridge University Press,, 1998.
- [158] A. I. Wang and T. Stalhane, "Using Post Mortem Analysis to Evaluate Software Architecture Student Projects," in *Software Engineering Education & Training, 18th Conference on*, 2005, pp. 43-50.
- [159] a. A. L. W. D. P. Perry, "Foundations for the Study of Software Architecture," *ACM Sigsoft Software Engineering Notes*, vol. 17(4), , pp. 40-52, 1992.
- [160] IEEE, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," Software Engineering Standards Committee of the IEEE Computer Society2000.
- [161] P. Kruchten, "The 4+1 View Model of Architecture," *IEEE Software*,, vol. 12, 6, , pp. 42 – 50, 1995.
- [162] B. Ahmed. and M. Steve., "Using ATAM to Evaluate a Game-based Architecture," in *Workshop on architecture-Centric Evolution(ACE 2006), hosted at the 20th European Conference on Object-Oriented Programming ECOOP.*, Nantes, France, 2006.
- [163] R. Kazman, *et al.*, "The architecture tradeoff analysis method," in *Fourth IEEE International Conference on Engineering of Complex Computer Systems*, 1998, pp. 68-78.
- [164] C. Vichido, *et al.*, "A constructivist educational tool: software architecture for Web-based video games," in *Proceedings of the Fourth Mexican International Conference on Computer Science, 2003. ENC 2003.* , 2003, pp. 144-150.
- [165] J. Krikke, "Samurai Romanesque, J2ME, and the battle for mobile cyberspace," *Computer Graphics and Applications, IEEE*, vol. 23, pp. 16-23, 2003.
- [166] S. Rabin, "Introduction to game development," in *Course Technology cengage learning*, ed: , 2008.
- [167] A. Rollings and D. Morris, *Game architecture and design - A new edition*: New riders publishing, 2004.
- [168] J. Blow, "Game Development: Harder Than You Think," *Queue*, vol. 1, pp. 28-37, 2004.
- [169] G. Booch, "Best practices in game development " IBM Presentation 2007.
- [170] A. Grossman, *Postmortems from game developer*: Focal Press, 2003.

- [171] R. Darken, *et al.*, "Projects in VR: the Delta3D open source game engine," *Computer Graphics and Applications, IEEE*, vol. 25, pp. 10-12, 2005.
- [172] B. Cowley, "Toward an understanding of flow in video games," *Computers in entertainment*, vol. 6, p. 1, 2008.
- [173] B. R. Gifford and N. D. Enyedy, "Activity centered design: towards a theoretical framework for CSCL," presented at the Proceedings of the 1999 conference on Computer support for collaborative learning, Palo Alto, California, 1999.
- [174] J. S. Lowe and E. F. Holton., "A Theory of Effective Computer-Based Instruction for Adults.," *Human Resource Development Review.*, pp. 4(2), 159-188. , 2005.
- [175] P. M. Privateer, "Academic Technology and the Future of Higher Education: Strategic Paths Taken and Not Taken," *Journal of Higher Education, Vol. 70* , 1999.
- [176] S. S. Boocock and J. S. Coleman, "Games with Simulated Environments in Learning," *Sociology of Education*, vol. 39, pp. 215-236, 1966.
- [177] J Kirriemuir and A. McFarlane, "Use of computer and video games in the classroom," in *Proceedings of the Level Up Digital Games Research Conference*, Universiteit Utrecht, Netherlands., 2003.
- [178] J. B. M. Schick, *The Decision to Use a Computer Simulation* vol. Vol. 27, No. 1 (Nov., 1993), pp. 27-36 Society for History Education, 1993.
- [179] C. D. Elder, "Problems in the Structure and Use of Educational Simulation," *Sociology of Education*, vol. 46, pp. 335-354, 1973.
- [180] L. Achterbosch, "Massively multiplayer online role-playing games: the past, present, and future," *Computers in entertainment*, vol. 5, p. 1, 2008.
- [181] E. W. Amerikaner, "Introduction to computer science using Alice 2.0: tutorial presentation," *J. Comput. Small Coll.*, vol. 25, pp. 141-141, 2010.
- [182] K. Anewalt, "Making CS0 fun: an active learning approach using toys, games and Alice," *J. Comput. Small Coll.*, vol. 23, pp. 98-105, 2008.
- [183] L. Werner, *et al.*, "Can middle-schoolers use Storytelling Alice to make games?: results of a pilot study," presented at the Proceedings of the 4th International Conference on Foundations of Digital Games, Orlando, Florida, 2009.
- [184] G. Fesakis and K. Serafeim, "Influence of the familiarization with "scratch" on future teachers' opinions and attitudes about programming and ICT in education," presented at the Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education, Paris, France, 2009.
- [185] P. A. G. Sivilotti and S. A. Laugel, "Scratching the surface of advanced topics in software engineering: a workshop module for middle school students," *SIGCSE Bull.*, vol. 40, pp. 291-295, 2008.
- [186] W. Jui-Feng, *et al.*, "Teaching Boolean Logic through Game Rule Tuning," *IEEE Transactions on Learning Technologies*, vol. 3, pp. 319-328, 2010.
- [187] J. Robertson and C. Howells, "Computer game design: Opportunities for successful learning," *Computers & Education*, vol. 50, pp. 559-578, 2008.
- [188] M. Al-Bow, *et al.*, "Using game creation for teaching computer programming to high school students and teachers," *SIGCSE Bull.*, vol. 41, pp. 104-108, 2009.

- [189] Y. Rankin, *et al.*, "The impact of game design on students' interest in CS," presented at the Proceedings of the 3rd international conference on Game development in computer science education, Miami, Florida, 2008.
- [190] Yulia and R. Adipranata, "Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment," in *2nd International Conference on Education Technology and Computer (ICETC)*, 2010, pp. V2-355-V2-359.
- [191] K. Wang, *et al.*, "3D game design with programming blocks in StarLogo TNG," presented at the Proceedings of the 7th international conference on Learning sciences, Bloomington, Indiana, 2006.
- [192] M. Eagle and T. Barnes, "Experimental evaluation of an educational game for improved learning in introductory computing," presented at the Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, USA, 2009.
- [193] J. A. Maxwell, "Understanding and Validity in Qualitative Research," *Harvard Educational Review*, vol. 62, pp. 279-279-300, 1992.
- [194] Tore Dybå and Torgeir Dingsøy, "Strength of evidence in systematic reviews in software engineering," presented at the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, Kaiserslautern, Germany, 2008.
- [195] D. K. Padgett, *Qualitative methods in social work research: challenge and rewards*: Thousand Oaks, Calif: Sage., 1998.
- [196] N. K. Denzin, *The research act: A theoretical introduction to sociological methods, 3rd edn. Englewood Cliffs, NJ: Prentice-Hall.*, 1988.
- [197] T. D. Cook and D. T. Campbell, *Quasi-experimentation: Design and Analysis Issues for Field Setting*. Chicago: Rand McNally., 1979.
- [198] W. R. Shadish, *et al.*, *Experimental and quasi-experimental designs for generalized causal inference*: Boston, MA, US: Houghton, Mifflin and Company, 2002.
- [199] M. D. LeCompte and J. P. Goetz, "Problems of reliability and validity in ethnographic research," *Review of Educational Research*, vol. 52, pp. 31-60, 1982.

9 Appendix: Selected papers

In this appendix we have included the twelve papers that have contributed the most towards the work presented in this thesis. The papers are included in chronological order.

Topic 1 - Game as motivation for lectures:

- Paper 1:** Experiences from Implementing an Educational MMORPG
- Paper 2:** Experiences from Implementing a Face-to-Face Educational Game for iPhone/iPod Touch
- Paper 3:** Improvement of a Lecture Game Concept - Implementing Lecture Quiz 2.0
- Paper 4:** A Pervasive Game to Know Your City Better

Topic 2 - Game development as motivation for lectures:

- Paper 5:** An Application of a Game Development Framework in Higher Education
- Paper 6:** An Evaluation of Using a Game Development Framework in Higher Education
- Paper 7:** XQUEST used in Software Architecture Education
- Paper 8:** Extending Google Android's Application as an Educational Tool
- Paper 9:** Using Game Development to Teach Software Architecture
- Paper 10:** Game Development Framework for Software Engineering Education
- Paper 11:** A guideline for game development-based learning: A literature review
- Paper 12:** Comparison of Learning Software Architecture by Developing Social Applications vs. Games on the Android Platform



Paper 1:

G1: Bian Wu, Alf Inge Wang and Yuanyuan Zhang, "Experiences from Implementing an Educational MMORPG", 2nd International IEEE Consumer Electronics Society's Games Innovation Conference (GIC 2010), Hong Kong, 21-23 December 2010. ISBN: 978-1-4244-7178-2, DOI: 10.1109/ICEGIC.2010.5716896

Experiences from Implementing an Educational MMORPG

Bian Wu¹, Alf Inge Wang²

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
bian@idi.ntnu.no¹, alfw@idi.ntnu.no²

Yuanyuan Zhang

Computer Teaching and Research Section
Institute of Chemical Defense of People Liberation Army
Beijing, China
yoyozhang-mail@163.com

Abstract— This paper describes the implementation of an educational Massively Multiplayer Online Role-Playing Game (MMORPG), named World of Wisdom (WoW). WoW is designed under the context of game design theory and game features extracted from surveys of popular MMORPGs. It is an open educational platform where students can “play exercises” instead doing them in the traditional paper way. Further, it provides an editor for teachers to create new game plots and content without the need of programming. As an aid for lectures, WoW can motivate the students to do the exercises more thoroughly. Finally, the paper presents both positive and negative experiences from the design and implementation of the game. We find that there are various theories that can benefit the design of educational MMORPGs. However, the key problem is how to choose and apply relevant theory to support the design, our experiences in the paper are examples that explore and explain this problem.

Keywords - MMORPG, Flow theory, educational game

I. INTRODUCTION

Massively Multiplayer Online Role-Playing Games (MMORPGs) have recently achieved tremendous success, and its characteristics can be beneficial for the learning purposes. Further, there are some cases studies [1-5] that describe how to apply the learning in MMORPGs, and evaluate its effectiveness for education. Most of these evaluations show positive results that such games can motivate the learners to study more actively. This also certifies that educational MMORPGs are fun to play and provide good learning experiences. Based on this context, our research has been to look into these kind games and identify how features from popular MMORPGs can be applied in educational MMORPGs to enrich useful design methods that guide the implementation of educational MMORPGs. In detail, we will present a short survey of recent educational MMORPG games, and investigate their design and implementations. And finally, we will propose an educational MMORPG design method to implement a MMORPG for learning, named World of Wisdom (WoW). This WoW prototype presents an open knowledge world for reviewing the contents of various courses. Finally, we discuss the positive and negative experiences we learned during the process of designing and implementing WoW.

The goals of WoW project are: 1) Develop an 2D educational MMORPG that can be used as an aid in lectures in

higher education instead of traditional paper exercises by providing “playable exercises, 2) Provide the toolsets for WoW that can be used to create new games for different courses without any programming requirements, and 3) Provide an example on how to design an educational MMORPG through supportive theories to make it enjoyable and effective for learning. WoW is mainly considered as a supplement to the formal classroom teaching in order to diversify lecture teaching. How to combine WoW with the course content depends on the lecture design of the teacher. We can use it in the classroom for a short time playing to review several knowledge points or let students play it on their own.

II. METHODS FOR DESIGN

A. Game design theory

Through our survey, we found that there are very limited game design theories that guide the design of educational MMORPG games. Most of common theories are from Malone. Here we give one example of a game design method from Nicoletta and Kelly [4] that have defined a set of game design criteria which are likely to promote user’s interest, enjoyment and learning; these elements are adapted by us and summarized in Table I. These features are from the three elements of intrinsic motivation (challenge, curiosity, and fantasy) identified by Malone and Lepper [6, 7].

TABLE I. GAME ELEMENTS FOR DESIGN ADAPTED FROM MALONE AND LEPPER

ID	Game elements that may promote engagement, motivation and fun	Reference
1	A shared, imaginary story context that establishes and support the activities	[8, 9]
2	An overarching goal	[6, 8, 10]
3	A gentle on ramp	[6, 8, 10]
4	Multiple levels with variable difficulty	[6, 11]
5	Uncertain outcomes	[6, 8, 10]
6	Various ways to win	[9]
7	A well defined advancement system	[6, 8, 9]
8	Rewards associated with advancement	[6, 8, 9, 12]
9	Opportunities to build new content	[8, 9, 13]
10	Ability to progress at the user’s own rate	[9, 10]
11	Hints not answers	[10]

To our knowledge, there are no papers that specifically describe examples of applying other game design theory beneficial for the design of MMORPGs. Thus, we would like to apply the flow theory from Csikszentmihalyi [14] for game design. He has conducted extensive research into what makes experiences enjoyable, based on long interviews, questionnaires, and other data collected over a dozen years from several thousand respondents. Flow is an experience “so gratifying that people are willing to do it for its own sake, with little concern for what they will get out of it, even when it is difficult or dangerous” [14]. Two individual papers by Sweetser and Cowley [15, 16] map the elements from games literature to the elements of flow, shown in Table II, adapted by us.

TABLE II. MAPPING THE ELEMENTS FROM GAMES LITERATURE TO THE ELEMENTS OF FLOW

ID	Flow theory	Games play elements
1	A task that can be completed;	The game
2	The ability to concentrate on the task;	Concentration
3	Concentration is possible because the task has clear goals;	Clear goals
4	Concentration is possible because the task provides immediate feedback;	Feedback
5	The ability to exercise a sense of control over actions;	Control
6	A deep but effortless involvement that removes awareness of the frustrations of everyday life;	Immersion, Flexible challenge
7	Concern for self disappears, but sense of self emerges stronger afterwards;	Immersion, Links between of virtual and real worlds.
8	The sense of the duration of time is altered.	Immersion
9	N/A	Social interaction

B. Characteristics from MMORPGs

Since we plan to implement a MMORPG for learning, we have surveyed current trends of MMORPGs and their characteristics. Finally, we would like to quote results from Achterbosch [17]. He attempts to determine the many aspects that make a successful MMORPG by a questionnaire survey. He also attempts to ascertain what new and innovative features are expected by the users from the next generation of MMORPGs. The research focuses on four areas of MMORPG: the social interactions between players in MMORPGs; the different architectures to build MMORPGs; the effects of latency on MMORPGs; and the problems that plague MMORPGs. We summarize the most popular and relevant features in Table III that could be used as indications for the design of educational MMORPGs. And some of the features were ignored, as they were only relevant to pure commercial MMORPGs, and to a less extent to the educational field. Such as some features feedback by the player, like “real world services”, “elaborate crafting system”, and “competition for resources”. These are not common issues relevant for learning games. Most relevant features for design of an educational MMORPG are shown in Table III.

TABLE III. MOST RELEVANT FEATURES FOR DESIGN OF AN EDUCATIONAL MMORPG

ID	Existing features	Remarks
1	Three character development models	Skill Points-Based System; Class-Based System; Combination of Class/skill
2	Multiplatform Support	-
3	Highly Customizable Characters	-
ID	Favorite Features	Remarks
4	Preferred Character Types in Ranking	Combination of Class/skill; Skill Points-Based System; Class-Based System
5	Top 3 Game setting	Fantasy, Futuristic, Post Apocalyptic,
6	Top 5 MMORPG	Lots of class/skill options, Graphics and effects, Large world to explore, Play vs Play, socialization
ID	Improving existing features	Remarks
7	Player versus Player combat	Such as balancing between classes
8	The level Grind	Repeated battles to level up
ID	Anticipating new features	Remarks
9	Player created and controlled content	-
10	Mini games	-
11	Dynamic content and quests	-

C. Final design for WoW

According to above preparation work, most of WoW’s features are based on what being described in Table I-III. Here we give examples of designs that are from above design methods, shown in Table IV. The left column is our design and right three columns are reference IDs for this design feature.

Such as *A* feature from Table IV (shorted as “A”), it mainly comes from the ID 1 in Table I, and we choose kingdom as a fantasy world since fantasy is top one game setting from ID5 in Table III. And the feature *B* is designed based on the ID 2 in Table I and relevant to ID 1,3,4,6 from the Table II when we implement it in detail. Also, the feature that is no need to struggle with level up is in *B*, which is from ID 8 in Table III. The feature *C* is designed mainly based on the ideas of flexible challenge to the different players. The features *D* and *E* are designed based on the idea of providing random rewards after killing the enemy non-playable character (NPC) or finishing the quest, making player feel encouragement and immerse in the game, but *E* also shows that the game is only one part of the lecture content. The feature *F* is hinted by the factors in the ID 1, 3, 4 in Table III. The toolset from feature *G* is inspired by the new content creating from ID 9 in Table I and ID 11 in Table III. The feature *H* mainly comes from ID 6 in Table II. Feature *I* has many functions, such as teacher can give hints (ID 11 in Table I), or discuss with the teacher for other help and make socialization (ID 9 in Table II, ID 6 in Table III). The feature *J* is an example that high score student will get bonus in real world by the teachers. The features *K* and *L* are two cases from Table III.

All of above features are general features, which are only a part of the whole WoW’s feature set, but we can design based on these features to make WoW more concrete and interesting. Besides of these features, we still have others issues, like

technical implementation, which describes in the following sections.

TABLE IV. FEATURES OF WoW AND ITS REFERENCES

ID	Features of WoW	ID in Table1	ID in Table2	ID in Table3
A	Consider each course as one imaginary kingdom, the kingdom have safe zone and battle zones. In each kingdom have different quests issued to the players	1	-	5
B	The goal is various and intact. Such as one quest requires you get level 2 from battles, but no more level up.	2	1,3,4,6	8
C	We classify the quests from easy levels to hard levels. Also we classify the questions from easy level to hard levels.	3, 4.	1, 2,	11
D	We have different random rewards (item, experiences) to the player if they win the battle.	5, 8	4, 6,	-
E	The exercises will have different definite way to win according the lecture design.	6	-	11
F	Character Development Models are mixed of class and skill-based	-	-	1,3,4
G	Use toolset to create new worlds, such as new and large battle zone, monsters or update questions	9	-	11
H	Have different level quests to challenge the newer player and skilled players.	-	6	-
I	Teaching assistant can log in world and help players. Players could chat each other or to teaching assistant.	11	9	6
J	Playing exercises is part of the lecture, having effect to the real world, Also best player gets bonus in real lectures.	-	7	-
K	Choose Java as main programming language	-	-	2
L	Mini game	-	-	10

III. FIRST PHASE: WORLD OF WISDOM

This section describes the prototype of WoW from aspects of the game plots and architectural design.

A. World of Wisdom introduction

World of Wisdom provides several kingdoms, where one specific kingdom focuses on one curriculum (one course). Each kingdom has several zones, mainly categorized as safety zone and battle zones. The safety zone has re-spawn point, shops, buildings, etc. If players get quests from a NPC or a teacher, they can go to the different zones to complete quests. If the quest involves some fighting, the players should go to the battle zones, and they will fight with monsters though answering different questions related to the curriculum. Further, the players and teaching assistants can chat with each other in WoW for socializing or help purposes.

Figure 1 shows the battle zone with different players and enemies. On the bottom of the figure are chatting window, player status, and system button from left to right. Figure 2 shows a screenshot of a question popping up during battle. Here the player has one minute to answer a question. The question is answered by selecting one of the alternatives below the time-left counter.



Figure 1. Battle zone from WoW



Figure 2. Questions during battle from WoW

B. Prototype overview

The prototype is divided into four applications: Client, Lobby Server, World Server and Database Server. The applications use a common package called Shared Library, which contains models used by several applications and the shared network implementation for sending messages between the applications. Figure 3 shows an architectural overview of the World of Wisdom prototype.

1) *Client*: The client is the main program for the user to log on to a kingdom. When starting the client, it connects to the Lobby Server, and the user creates or inputs a username and password. The Lobby Server checks if the information is correct via the Database Server. If the user is authorized to log on, the Lobby Server will return a list of World Server that the user may connect to. When the user has connected to a World Server, the user will be in the game world. Client will present the kingdom that user plays in. While in the kingdom, the user may fight with enemy or walk around and chat with other players. The user can also look at the states of the character,

attributes, attacks and inventory. In the inventory the user can drag-and-drop items from their bag onto their body to equip items.

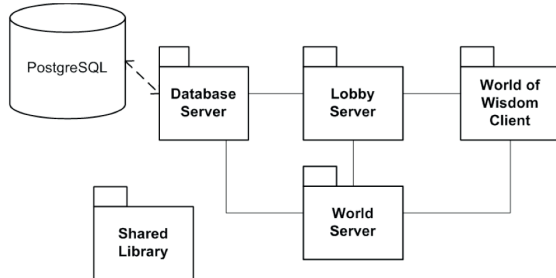


Figure 3. . Architectural overview of the WoW Prototype

2) *Lobby Server*: The Lobby Server handles the verification of players, and contains a list of the World Servers that are available online. The Lobby Server is a small but central part of the WoW prototype.

3) *World Server*: The World Server contains the game worlds that the users can move around in. When the world server starts, the teacher will choose a kingdom to register as an online state and informs the Lobby Server, so that users may connect to. When the user performs an action, like movement or attack, a message will be sent to the World Server with information about the action. The action will then be handled, and in case of movement, it will be broadcasted to the other players so that their worlds are updated.

4) *Database Server*: The Database Server receives and handles requests from the World Server and the Lobby Server for access to the PostgreSQL [18] database. It uses serializable objects to package the information and send it over the network using the TCP/IP protocol in Java.

IV. SECOND PHASE: WOW EDITOR

Before creating the WoW toolsets, we performed a survey on the features of different toolsets in MMORPGs and tried to integrate educational functionality when designing an editor that are helpful for the teachers to create their own kingdom (course) and input relevant questions and answers in the WoW educational MMORPG.

A. Preparation works

The earlier type of game editors were often simple, using text editors, or using simple primitives for representing game elements, such as the editor that was made for Wolfenstein 3D [19]. It was not an official editor, but shows how colors and letters can be used as representatives when creating a map. The letters represented object, like creatures or power ups, and colors represented floors and walls.

Recent game editors are more advanced and often feature in-game graphics that shows how it will look when you play in the maps. Examples are Trackmania [20], which is a arcade

game, and Farcry [21], which is a first person shooter. This allows quick testing of changes and also makes it faster to develop new maps (levels). Another example is the Adventure Construction Set [22] from 1985, which could be used to generate entire role-playing game. Trackmania from 2003 to 2008 offers an easy to use in-game level editor, whereas LittleBigPlanet [23] from 2008 contains a perfect example of a state-of-the-art game with an integrated editor which is also a part of the gameplay itself. Aurora [24] is the game engine developed by Bioware for the game Neverwinter Nights, and the Aurora Toolset is the world editor that comes with the game. With this editor the players may create their own worlds and alter most of the variables in the game, like spells, monsters, NPC dialog, etc.

B. WoW Editor Interface Design

The main function of the WoW editor is to provide convenience for the teachers to create game levels, and update questions databases (not available for the players/students). The design of the WoW editor GUI is shown in Figure 4:

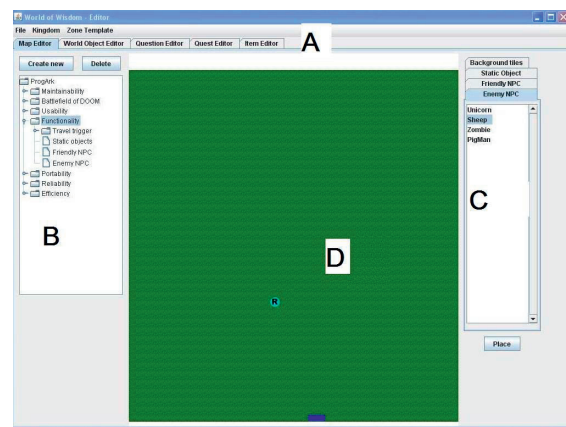


Figure 4. World of Wisdom Editor

We divide editor interface into four areas, as A, B, C, and D. Each area has its own function. A is the toolbar with major commands and preferences, including five panels: Map editor, World editor, Question editor, Quest editor and Item editor. B contains a list of the objects corresponding to each panel from A. C contains the content and its attribute that can be placed in the world. D is the main display that shows the world, and where most of the work is done.

Figure 5 is five screenshots of all panels, from the top to bottom is Map editor, World Object editor, Question editor, Quest editor and Item editor respectively.

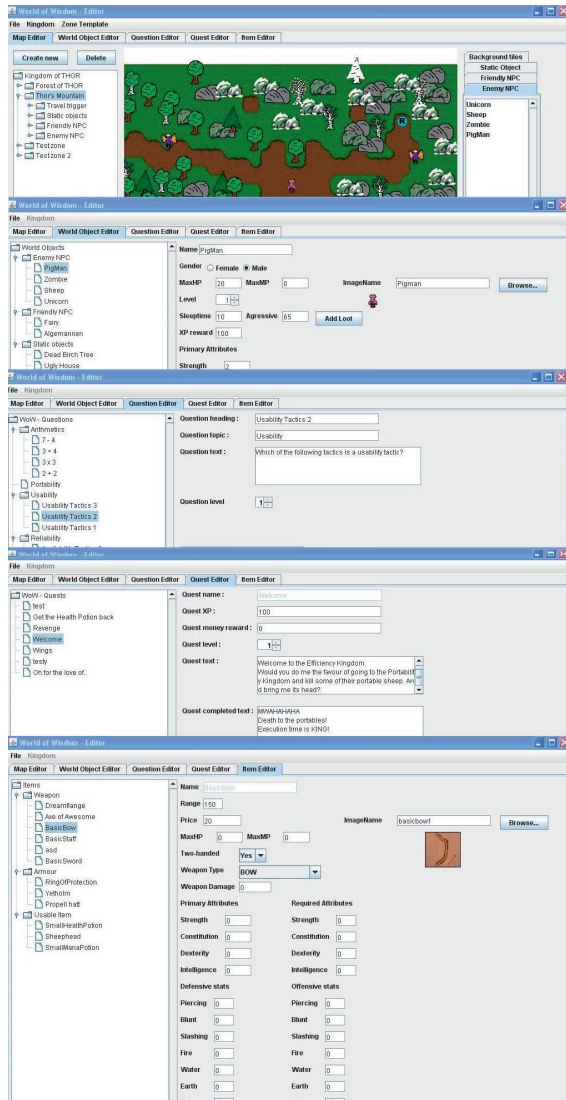


Figure 5. World of Wisdom Editor

1) *Map editor*: It is mainly used for creating different zones in a kingdom, such as add maps, trees, stones, friendly and enemy NPC in different zones, or put a trigger between two zones when a player wants to enter from one zone to another.

2) *World objects editor*: Teachers can add new world object and its attributes through world objects editor, such as new enemy NPC. It can also load this NPC images through external links.

3) *Questions editor*: Teachers can add or update questions through questions editor. And the questions can be classified

into various categories, and marked with varying levels of difficulty.

4) *Quest editor*: Teachers can add new or revise quests' content through the quest editor. The quests are marked with different difficulty levels with corresponding rewards.

5) *Item editor*: It can be used for adding or updating item attributes.

C. System overview

Figure 6 is an updated overview of the WoW system including the WoW Editor. Compared with Figure 3, WoW editor, shorted as world editor in the Figure 6, communicate directly with the Database server.

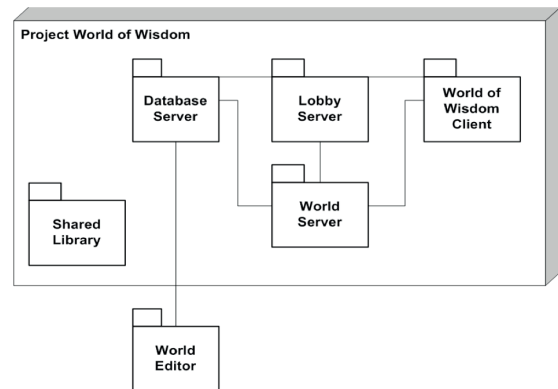


Figure 6. Architecture of World of Wisdom Editor

D. How to create a kingdom and exercises for students

Since WoW is an open educational platform, it can serve different courses. Here are the steps the teacher needs to go through to create the game world without any programming requirements:

First, the teacher uses the WoW editor to create a kingdom for a specific course. We can use the WoW editor to paint the maps or load an existing map template directly. *Second*, the teacher can use the Map editor to place the NPCs and items or he/she can create new NPC and items through the world object editor and the item editor. *Third*, a teacher will create the quests and questions in the kingdom. For the questions used in the battle, a teacher can input questions of varying difficulty levels related the course, and link these questions to different levels enemy NPCs. Similarly, quests can be created and issued to the friendly NPCs. *Finally*, students can log into the world and find friendly NPCs to get quests and go to the battle zone to perform challenging tasks. Figure 7 shows an example of a final kingdom ready to be played.

V. DISCUSSION: EXPERIENCES FROM DESIGN AND IMPLEMENTATION

This section presents experiences we learned during developing an educational MMORPG.

A. Positive experiences

We would like to share our positive experiences that could be useful as reference for others wanting to design and implement an educational MMORPG.

- Use game design methods to guide the design and implementation.** When preparing to develop an educational MMORPG game, we had limited supportive theories to use. Most of the examples describe a direct way to implement MMORPGs for learning from their own experiences, but without any support in theory. Here we propose to apply suitable theories for this genre in the game design. Further, we provide an example of how to apply the combination methods of Malone, flow theory and features of MMORPG in the integrated design of the WoW framework. We find it quite useful to use this approach during the process of design and implementation. Most of the functions and scenarios become more and more concrete and interesting through the interwoven design method.

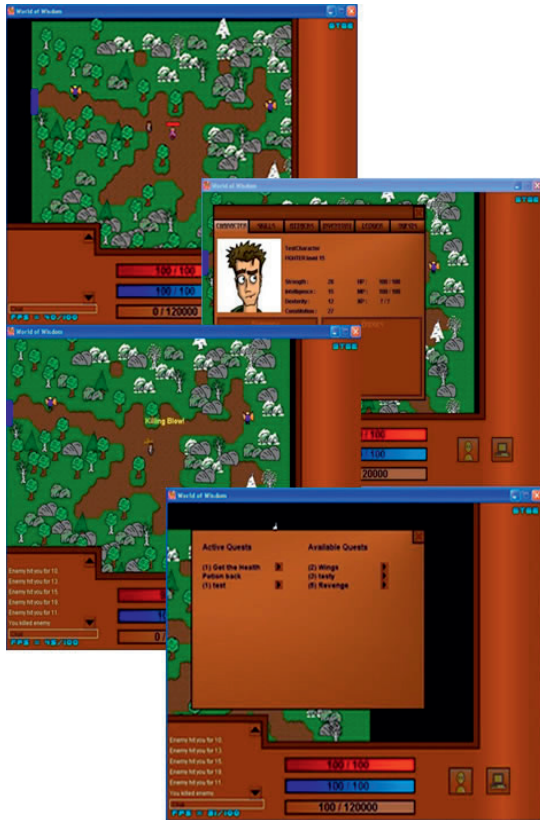


Figure 7. World of Wisdom

- Use toolsets to create games based on the WoW framework.** Our toolset was inspired by some open editors from existing computer games, which can be used to create new maps and scenarios for game players without requiring any programming. This makes possibility for users to create their own imaginative worlds and plots from the existing game frameworks. We extend this type of editor to provide not only the creation of traditional game plots, but also related educational functionality. Thus the teachers can create new games for courses, and update questions databases and quests using the provided editors.
- Massive Multiplayer Foundation.** The system is designed with several servers (world server, database server and lobby server) to support the client. This is an important part of a MMORPG architecture since these server are implemented with some specified functionality. The client must log in with the lobby server before being allowed to join one of the potentially many world servers. Both the lobby server and world servers talk to the database server to get information from the database. From our experiences, this foundation, while somewhat time-consuming to develop, proved to be reliable and effective. We had little problem with this aspect of the system, other than it consumed quite a lot of development effort.
- Good Teamwork.** The developers are all last year master students from Norwegian University of Science and Technology. It is positive experiences that students can works in pairs or groups to implement the projects. They can cooperate with each other to solve the coding problems and use their personal advantages to help teamwork. Another benefit of using students to implement such games is that the students know the game genre well and how such games should behave since they are regularly playing such games.

B. Negative experiences

These negatives experiences need to be taken care of and could be seen as improvement reference for the educational MMORPGs.

- Lack of pedagogical background or learning theory to support the design.** Even we use game design methods to implement the WoW, adding learning theory should improve the prototype and make it more effective for learning. Sancho describes how MMORPG can be applied to problem-based learning [5], while Economou shows how MMORPG can be applied to collaborative learning [25]. These papers can be a starting-point to improve our WoW design methods. Since the mini-games in WoW are still under development, we will consider applying the learning theory in the mini-games.
- Lack of MMORPG features.** There are still other interesting or anticipating features from Achterbosch's survey of MMORPGs [17], such as "Technical enhancement", "Item crafting and Player Economy", to have a game master (an intermediary) between the

developers and players [26]. Due to our limited resources and time and that these are not highly relevant educational factors, such features are not implemented in our prototype.

- **Use of other theory.** Besides of learning theory and game design theory, there is still room for using other theories to support educational MMORPGs design. One example is from Nicoletta and Kelly [4], using color psychology to guide the design: “The choice of the color and lighting schemes was based on research studies on the impact of color and light on learning [27, 28], and on the association between colors and children’s emotions [29]. One study shows that desaturated colors have a negative impact on stimulation while highly saturated colors increase alpha waves in the brain which are directly linked to awareness.” It will demand a huge amount of work to do experiments of different supportive theories to design educational MMORPGs. Further, there might be duplicating or conflicting parts in these theories. If this is the case, we need to conduct more experiments to gain more experiences to make trade-off between these theories during the design process.
- **Limited help from a game engine.** We chose to use an existing game engine to develop WoW to save time. Since our game should be cross-platform game, we only considered Java game engines. Based previous experiences, the most parts of the Golden T Game Engine (GTGE) [30] works well, but suffer from some existing bugs. Such as, when the file could not be found on the hard drive by using the URL, it returns a null object, and this caused a null pointer exception in GTGE. The problem was that the GTGE framework returned an error message, but didn’t throw an exception, so the line where the error occurred was not specified as usual with error traces. As a whole, the GTGE is a decent game development framework for Java, but we did encounter a few issues while using it. In the end we are not sure if we really saved much time by using the GTGE, since of the time demanded to learn the parts we used, and fix bugs that we found. In the end, we did not end up using much of the GTGE framework and had to implement most part of WoW from scratch.
- **Lack of intact documentation for a long project.** This WoW project has lasted more than one and half year. We developed the WoW prototype first, and later continued to add the editor functions and make some changes. Since the developers are all students with half year projects, it is necessary to keep the intact logs and complete development documentation useful for the new comers to the project. Even we predicted this problem, we encountered problems of effective software management. We cannot predict which logs are important for the new incoming coming students since they can choose their own focus on the project. We try to put everything in logs, but it costs a lot of time and the work might be useless. To overcome this problem, we make the student do the most important

key documentation. This is not always easy, as the students have different background in programming, experiences in Java and MMORPG, and what one student find is sufficient documentation is not always sufficient for another. Another aspect is that we suggest making a classification of the documentation to enable quick search and identify the information we need if the logs become too large and too cluttered.

As a summary, it is still harder than we think to use an interwoven design to implement an educational MMORPG, but we think it will pay off in the long run. Such a design approach covers various design methods that exceed the field of learning, and games design theory. Our experiences presented in this paper are examples that explore and explain this problem.

VI. CONCLUSIONS

This paper describes the implementation of an educational MMORPG game and shares related experiences, including positive and negative aspects. Most of the features of our WoW game come from existing game design methods. But based on our experiences, we find that there exists a cross-topic of applying design methods to educational MMORPGs. From the case study, we find that more research and experiments are needed to find a set of criteria or a framework to guide the design of the educational MMORPGs. This is not end, it is just a beginning of research in this area.

ACKNOWLEDGMENT

We would like to thank Thor Grunde Krogsæter, Henrik Halvorsen, Esben Andre Føllesdal, Andreas Johnsen, Lawrence Valtola and Sondre W. Bjerkaug for the works of implementing of WoW.

REFERENCES

- [1] Martin, V.S.: ‘Online Videogames in an Online History Class’, Second conference on Digital Games and Intelligent Toys Based Education, 2008 , pp. 146-148
- [2] Chang, M.: ‘Web-Based Multiplayer Online Role Playing Game (MORPG) for Assessing Students’ Java Programming Knowledge and Skills, 2010 Third IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning’ (2010. 2010)
- [3] L. Natvig, S.L., and A. Djupdal: ‘Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals’, 34th Annual frontiers in education, 2004.
- [4] Nicoletta, A.-V., and Kelly, W.: ‘SMILE: an immersive learning game for deaf and hearing children’. Proc. ACM SIGGRAPH 2007 educators program, San Diego, California 2007
- [5] Sancho, P.: ‘Multiplayer role games applied to problem based learning Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts - DIMEA 08’ (2008)
- [6] Thomas, W.M.: ‘What makes things fun to learn? heuristics for designing instructional computer games’. Proc. Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, Palo Alto, California, United States 1980 pp.
- [7] Lepper, M.R.: ‘Motivational considerations in the study of instruction’, Cognition and Instruction, 1988, pp. 289-309

- [8] Barab, S., Thomas, M., Dodge, T., Carteaux, R., and Tuzun, H.: 'Making learning fun: Quest Atlantis, a game without guns', Educational Technology Research and Development, 2005, 53, (1), pp. 86-107
- [9] http://www.gamasutra.com/view/feature/3041/guidelines_for_developin_g_.php
- [10] Chuck, C.: 'An interpreted demonstration of computer game design'. Proc. CHI 98 conference summary on Human factors in computing systems, Los Angeles, California, United States 1998 pp.
- [11] Ben, S.: 'Designing the User Interface: Strategies for Effective Human-Computer Interaction' (Addison-Wesley Longman Publishing Co., Inc. 1997)
- [12] Bickford, P.: 'Interface design : The Art of developing easy - to - use software ', 1997
- [13] Maria, R.: 'Learning by doing and learning through play: an exploration of interactivity in virtual environments for children', Comput. Entertain., 2004, 2, (1), pp. 10-10
- [14] CSIKSZENTMIHALYI, M.: 'Flow: The Psychology of Optimal Experience.' 1990
- [15] Sweetser, P.: 'GameFlow: a model for evaluating player enjoyment in games', Computers in entertainment, 2005, 3, (3), pp. 3
- [16] Cowley, B.: 'Toward an understanding of flow in video games', Computers in entertainment, 2008, 6, (2), pp. 1
- [17] Achterbosch, L.: 'Massively multiplayer online role-playing games: the past, present, and future', Computers in entertainment, 2008, 5, (4), pp. 1
- [18] <http://www.postgresql.org/download/>, accessed 30-Aug 2010
- [19] <http://www.idsoftware.com/games/wolfenstein/wolf3d/>, accessed 22-Jan 2009
- [20] <http://www.trackmania.com/tm/index.php?rub=united>, accessed 22-Jan 2009
- [21] <http://www.crytek.com/games/far-cry/overview/>, accessed 22-Jan 2009
- [22] http://www.spillverket.no/artikler/den_kreative_revolusjonen/66786/1, accessed 22-Jan 2009
- [23] <http://www.littlebigplanet.com/> accessed 22-Jan 2009
- [24] <http://nwn.bioware.com/builders/index.html> accessed 22-Jan 2009
- [25] Economou, D.: 'User centred virtual actor technology, Proceedings of the 2001 conference on Virtual reality archeology and cultural heritage - VAST 01' (2001)
- [26] Alexander, T.: 'The Massively Multiplayer Game Development 2 (Game Development)' (Charles River Media, 2005)
- [27] Engelbrecht, K.: 'The impact of color on learning', NeoCON2003, 2003
- [28] Duke, D.L.: 'Does it matter where our children learn? White paper for the national academy of sciences and the national academy of engineering.' 1998
- [29] Boyatzis, C.J.: 'Children's emotional associations with colors', The Journal of Genetic Psychology, 1994, 155, (1), pp. 77
- [30] <http://www.goldenstudios.or.id/products/GTGE/>, accessed June 2009

Paper 2:

G2: Alf Inge Wang, Bian Wu, Sveinung Kval Bakken, "Experiences from Implementing a Face-to-Face Educational Game for iPhone/iPod Touch", 2nd International IEEE Consumer Electronics Society's Games Innovation Conference (GIC 2010), 21-23 December 2010, Hong Kong. ISBN: 978-1-4244-7178-2. DOI: 10.1109/ICEGIC.2010.5716895

Experiences from Implementing a Face-to-Face Educational Game for iPhone/iPod Touch

Alf Inge Wang¹, Bian Wu²

Dept. of Computer and Information Science,
Norwegian University of Science and Technology
Trondheim, Norway

¹alfw@idi.ntnu.no, ²bian@idi.ntnu.no

Sveinung Kval Bakken

Dept. of Telematics,
Norwegian University of Science and Technology
Trondheim, Norway

sveinung.bakken@gmail.com

Abstract—This paper presents a location-aware educational game for the iPhone/iPod Touch platform. The game, KnowledgeWar, is a quiz game where students can challenge each other in face-to-face or remote knowledge battles. The game contains a game lobby where players can see all who are connected, and the physical distance to them. The paper describes our experiences from developing KnowledgeWar and results from a user test followed by a questionnaire. The user test focused on usability, and how well the game was suited for learning. The results showed among other things that the game had high usability, it is helpful for summarizing topics, it can stimulate involvement and social interaction, and that smartphones are well suited for such games. The results also revealed that our game did not to stimulate students to attend more lectures or pay more attention during lectures.

Keywords—component; Educational games, Mobile games, Location-awareness, iPhone development

I. INTRODUCTION

In recent years, smart phones have now become increasingly popular. In Norway with a population of only 4.8 million, close to 400,000 iPhones have been sold. Smart phones are no longer only seen in the hands of businessmen, but more and more students use such phones everyday. The motivation for students to get these phones is in addition to impressing their friends to have better access to mobile Internet and to enjoy a richer mobile gaming experience.

In the Lecture Games project, we want to explore how to use games in higher education to provide variation in teaching and new ways of promoting learning through interaction between teacher and students, and interaction between fellow students. Smart phones open new opportunities to be explored for educational games, including the utilization of location.

Games in education have also become increasingly popular in recent years, especially for children and have proven to be beneficial for academic achievement, motivation and classroom dynamics [1]. Teaching methods based on educational games are not only attractive to schoolchildren, but can also be beneficial for university students [2]. Research on games concepts and game development used in higher education is not unique, e.g. [3-5], but there is an untapped potential that needs to be explored.

By introducing games in higher education teachers can access teaching aids that promote more activity among students, provide alternative teaching methods to improve variation in lectures, enable social learning through multiplayer learning games, and motivate students to work harder on their projects and exercises.

Games can mainly be integrated in higher education in three ways. *First*, traditional exercises can be replaced by games motivating the students to put extra effort in doing the exercises, and giving the course staff an opportunity to monitor how the students work with the exercises in real-time [6, 7]. *Second*, games can be used within a traditional classroom lecture to improve the participation and motivation of the students through knowledge-based multiplayer games played by the students and the teacher [8, 9]. *Third*, game development projects can be used in computer science (CS) or software engineering (SE) courses to learn specific CS or SE skills [10, 11]. This paper focuses on a presentation of experiences from implementing a game that can be used in the first two ways described above. The KnowledgeWar game described in this paper can be used as an exercise to make the students rehearse the theory in a more interesting way. It can also be used as a part of a lecture, where the students get a few minutes to play a game trying to remember what they have been taught during the lecture. We believe that it is important to incorporate games and game technologies into teaching, as it gets more common to also use game technology in serious applications [12-15].

This paper describes the architecture and the design of an iPhone game where the players challenge each other in a quiz-battle. The paper shares experiences from working with the iPhone platform as well as results from a user test that focused on usability and on how the students perceived learning from playing the KnowledgeWar game.

The paper is organized as follows. Section II describes related works. Section III describes the KnowledgeWar game including its architecture and design. Section IV shares some experiences we gained from working on the iPhone platform, and describes the user test we performed to assess usability and whether the game was successful in an educational setting. Section V presents an evaluation and discussion of the results, and Section VI concludes the paper.

II. RELATED WORK

In this section we present some educational games and applications for iPhones and for other mobile devices.

Statecraft X is on a mobile learning game for iPhone, designed and developed to enact a program for citizenship education undertaken by 15-year-old students [16]. Located in the Social Studies curriculum, the game represents one component of a broader learning environment that includes in-class dialogic activity to facilitate student sense-making and identity construction.

The paper “Using a PDA for Mobile Learning” [17] provides a learning space based on a Role Play Game (RPG) and quiz model which is good at supporting high level social interaction, progression by incremental tasks, continuous player feedback, and reward systems. The architecture consists of two PDAs running the game application. Information from one application is passed to another via an infrared connection. An exchange manager defines how data objects are passed between two Palm OS handhelds. The user model as defined in the database is used to control the operation of the game from the learner’s perspective by relating information from a number of components with which the learner interacts.

Schwabe and Göth used handheld computers running a mobile learning game to support the orientation days at a university [18]. The orientation rally is a fun event intended to get to know the university and its surroundings through doing various tasks at certain spots. The students play individually or in small groups against other players. Each group receives a handheld computer. During the orientation rally, each group gets different tasks referring to significant places, people and events. The handheld device shows the current position of the group on the digital map of the university. When the group enters a building the outdoor map switches to an indoor map of the building. The whole rally is structured as a cooperative and competitive game. The architecture integrates three components: a mobile PDA client, a web browser client, and a server. The architecture provides clients with their own private state of the ongoing game so the game also works offline. The server works as the game’s central coordination point. Any changes in the game are transferred to the server, which broadcast the data to all clients.

The Sotto Voce project [19] is a PDA mobile companion for Museum co-visiting that provides audio content of artwork descriptions and acts as an audio media space between visitors providing a mean for awareness and sociability. The authors have identified four kinds of activity: (i) *shared listening* to promote interaction and communication between companions; (ii) *independent use* to enable temporarily or entirely the switching off of the shared listening for visitors that do not want to engage social interactions; (iii) *following*, when a companion is in charge of driving, implicitly or explicitly, the tour; and (iv) *checking in*, which is a short activity to maintain and update the shared context.

The City project [20] takes place at the lighthouse in Glasgow. The system considers three kinds of technologies: (i) a real visit using a PDA; (ii) a virtual reality visit in a 3D world; and (iii) a Web visit. With this system, visitors are able

to share their museum experience visit and navigate jointly through mixed realities: the Web, the virtual and physical reality. Information is provided about each visitor location and orientation. In addition, they may communicate through audio channels. The authors have observed that voice interaction, location and orientation awareness, and mutual visibility are essential to the success of co-visiting between remote users.

The paper “Using mobile phones in English education in Japan” [21] proposes an application that create a Web site explaining English idioms. Student-produced animation shows each idiom’s literal meaning; a video shows the idiomatic meaning. Textual materials include an explanation, script, and quiz. Thirty-one Japanese college sophomores evaluated the site using video-capable mobile phones, finding few technical difficulties, and rating highly its educational effectiveness.

TAMALLE (television and mobile phone assisted language learning environment) [22] describes the development processes for a cross-platform ubiquitous language learning service via interactive television (iTV) and mobile phone. The aim of the system is to support advanced learners of English as a second language in their television viewing, as just one element in their language learning activities. As the focus of the learners will be on media consumption rather than on conscious language learning, this environment is designed to be as discreet and non-intrusive as possible. The system provides support for captions and other on-screen displays for comprehension of specific language (or sometimes cultural) items for viewers as they watch English language programs. The mobile phone can further support learners’ understanding of the program by enabling them to access the summary of program as well as difficult language and cultural items that may appear. Viewers are also able to add, search for and remove items from/into their personal spheres. Even without television, the mobile service is useful for learning the new language items and as a tool for managing personal knowledge.

Lecture Quiz is a multiplayer game where students can play a quiz game using their own mobile phone and the teacher moderates the game using his PC and a video projector [8, 9]. The game provides two game modes: *score distribution* – the 3D animated presentation of the students answers distributed on the various alternatives, and *last man standing* where the players have to answer correctly to make it to the next round.

Mobile Game-Based Science Learning [23] describes a pedagogical methodology based on interactive games for mobile devices (PDAs). The methodology is oriented to developing problem-solving skills in science classes for 8th graders, by including pre-classroom activities with the teacher, classroom activities, and a central activity using an interactive game for a mobile device. The core problem they have to solve through the game consists in preserving and evolving different biological species from the animal kingdom, in an unknown and varying environment, by modifying some key factors for evolution of the species.

III. THE KNOWLEDGEWAR GAME

This section presents the KnowledgeWar iPhone/iPod Touch game as well describing the main architecture and the design of the game.

A. The Game

The initial idea behind the KnowledgeWar game is the notion that students spend a lot of time walking around socializing. Sometimes students even skip classes to just spend some time together. The social interaction among students can be both face-to-face and electronically using mobile devices such as smart phones and laptops. With the KnowledgeWar game, we would like to offer the students an opportunity to spend this social time both entertaining and educational.

A major challenge for educational games is to create games that can be used in several courses but still can be fun. Single player quiz-games fits very well into this category, but they can be a bit tedious and repetitive. By adding a social component, such games can be much more competitive and engaging. Figure 1 shows four screenshots from KnowledgeWar.

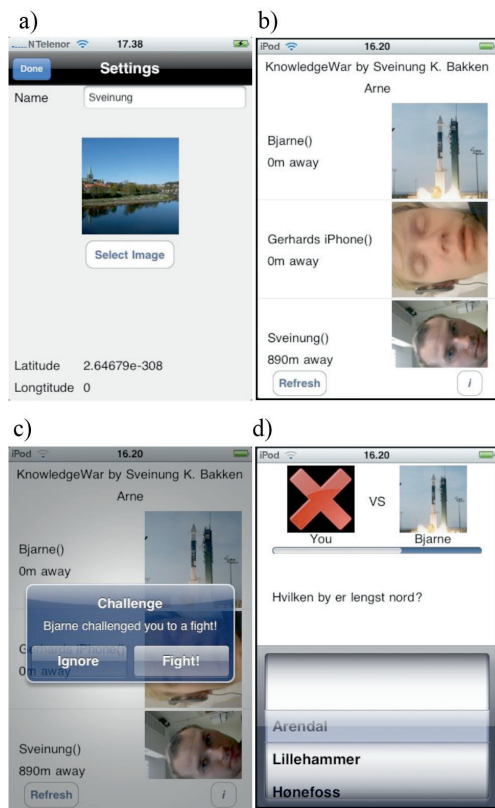


Figure 1. Screenshots from the KnowledgeWar game

The game is interesting, but not very complex. When the user starts the game application on his iPhone/iPod Touch, the player can set a nickname and choose a picture if it has not been done before (see Figure 1a). The lobby screen of the game shows all the available players with names and pictures, and how far away they are (see Figure 1b). In this way, a player can choose to have a face-to-face game or to play remote. Since the

game does not enforce the players to be on the same spot, our game supports players with different social preferences. To play the game, the user simply touches the player to be challenged (see Figure 1c). If the other player agrees to start a knowledge fight, the game is on (see Figure 1d). The game itself is a quiz game where the player can choose among a set of alternatives using an iPhone selector (roll selector). The alternative chosen when the timer runs out will be evaluated, and the players get points if their choice is correct. After playing through several questions, the winner is announced to both players. The game application will then take the user back to the game lobby, where the user can challenge more players to fight. The quiz questions used for the game are stored in a database on the game server. The main restrictions regarding the questions are that there must be two or more answer alternatives that can be described in a short sentence (maximum 30 characters) for every question and that the time limit for giving an answer and the correct alternative must be specified. This format makes the game perfect for rehearsing theory in a course to test students' theoretical knowledge. The game is not suitable for testing certain skills or techniques. Many textbooks provide teachers with multiple-choice assignments ready-to-be used for testing students' knowledge. These multiple-choice assignments can directly be used in the game as long as the description of alternatives is not too long. When a player challenge another player in the KnowledgeWar game, the game server will randomly pick five questions from the database to be used in a game session. A natural extension of the game would be for the teacher to be able to bundle questions at different difficulty levels, to let students get questions that are appropriate for their level of knowledge. The difficulty level could also be related to the score you get from winning a knowledge fight.

The main challenge when implementing the iPhone client was to learn Objective C and the interfaces in the iPhone OS.

B. The Architecture

The implementation of the KnowledgeWar game is based on a service-oriented client-server architecture. An overview of the architecture is shown in Figure 2. A game server takes care of all services shared by users such as player profiles, question database and game sessions. Apple's Push Notification Service is used to push events to the iPhone/iPod Touch clients.

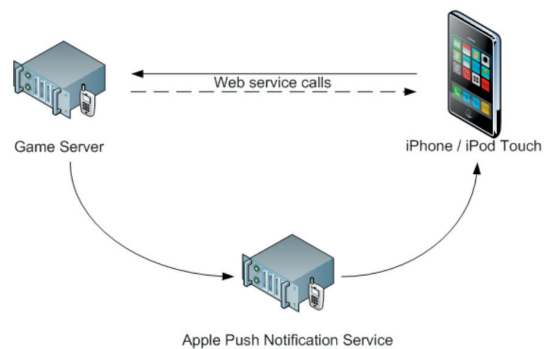


Figure 2. KnowledgeWar architectural overview

Figure 3 shows the game server architecture. We used several free and open source libraries for providing the main services of the server to make the server as flexible as possible. The server is Java-based and all of the components are tied together with Maven, Spring XMLs and custom Java code. This enables a flexible plugin interface to the server. The architecture shown in Figure 3 is divided into three main parts.

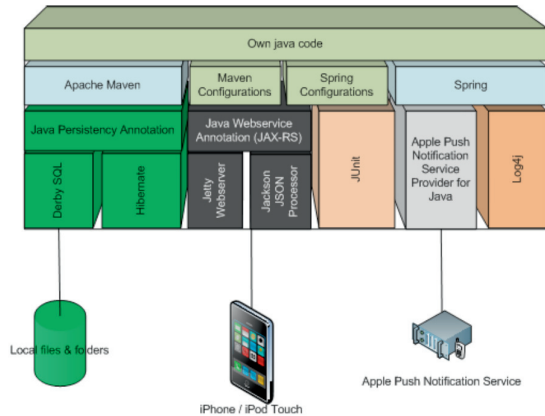


Figure 3. KnowledgeWar Game Server Architecture

The left part in Figure 3 takes care of the data management and data persistence. The middle part consists of a web server and a framework for providing web services to the client. The right part takes care of pushing events to the client via Apple's Push Notification Service. The architecture also includes JUnit use for testing and Log4j for providing logging when running the server. The domain model shared between the server and the client consists of *Challenge* – a challenge from one player to another, *Heartbeat* – position processing, *Opponent* – contains a *Player* object and the distance to this player, *Player* – containing unique identification, nickname and avatar image, and *Round* – containing questions, possible answer and correct answer. Services provided by the server are a game service, a location service, a player service, a persistence service, and a push notification service.

Figure 4 shows the client architecture. When the client is launched, the application delegate is initiated. The other parts of the architecture is the persistency class providing persistency locally on the device, the domain model similar to the server (*Heartbeat*, *Challenge*, *Player*, *Opponent* and *Round*), the backend integration, view controllers consisting of four views (see Figure 1), the view definitions used by the view controllers, and the Utility class providing shared attributes and methods.

The third author of this paper implemented the KnowledgeWar game over 4 months in his master project.

IV. EXPERIENCES AND USER TEST

In this section we will share some experiences we learned from developing an iPhone application, describe how we

performed the user tests of the game, and present the results from the user tests and the questionnaire.

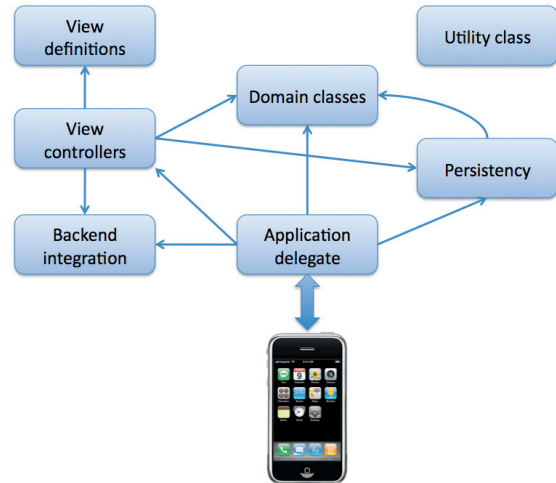


Figure 4. KnowledgeWar Client Architecture

A. Provision of iPhone Apps

The iPhone OS has a built-in security mechanism to ensure that applications have not been tampered with before they are installed. The security mechanism consists of a chain of certificates that must be signed by all applications to be installed on the device. In practice, this means that researchers cannot simply compile and build the code and then deploy it directly on a device. It is important for researchers wanting to use iPhone/iPod Touch as an exploration platform to know about the limitations related to deployment and what options are available. There are four methods for deploying an iPhone/iPod Touch application: 1) *Use App IDs* that will identify your application and make it related to your provisioning profile; 2) *Use ad hoc distribution* that will allow the developer to distribute an un-approved application to up to 100 pre-registered devices through iTunes; 3) *Use a debug install* where the application is deployed to a physically connected device by launching the application directly from the development tool; and 4) *Use the AppStore distribution channel* where the application must be approved by Apple before it is made available on Apple's AppStore.

For many developers, including ourselves, we were not used to the approval process enforced by Apple. Even though we did not plan to release our KnowledgeWar game on AppStore, we were curious on how hard it was and how much time was required to get an application approved and ready for download/sale at AppStore. To test the entire app approval lifecycle, we submitted a four unrelated apps to the AppStore. The applications we submitted were two variants of a time tracking application and two applications for efficient emergency event management. Three out of the four apps were

payable apps. The results of our test on approval time are shown in Table I.

TABLE I. TIME FROM APPSTORE SUBMISSION TO READY FOR SALE

Application	Approval time
App1	4 days
App2	4 days
App3 version 1	5 days
App3 version 2	6 days
App4 version 1	5 days
App4 version 2	6 days

Our experiences from submitting apps to AppStore approval were that we did not encounter any problems, and that the processing time was acceptable. For larger apps, for apps that challenges the technical constraints or for apps that challenges the moral constraints of Apple, longer processing time must be expected.

B. The User Test of the KnowledgeWar Game

To get the users' verdict of what they thought of the KnowledgeWar game we conducted a user test. The purpose of the test was two-fold. First, we wanted to test the usability of the application, as high usability is expected on iPhone/iPod Touch applications. Second, we wanted to see if the students found the game useful for learning and fun to play.

The user test was held on April 30th, 2010 in a reserved auditorium at the university campus. We recruited students from 1st, 2nd and 5th year of the Master of Science (MSc) in Communication Technology to do drop-in sessions during a two-hour time window. In total eight subjects participated. All the subjects had a technical background and were familiar with smart phones. The game was pre-installed on six iPhones and iPod Touches before the test began. The students played against each other with mainly technical questions from a software architecture course in the questions pool. The available wifi network on campus was used as the carrier for the communication, and the game server was hosted on a laptop on the same wifi network.

The users had to go through the following steps:

1. Start the application
2. Input a nickname and select an image for the player profile
3. Challenge another player
4. Play the game
5. Repeat steps 3 and 4 a couple of times
6. Fill inn the questionnaire

The success criteria for the KnowledgeWar game were:

- H1: The game application has high usability
- H2: The game is a fun way of practicing knowledge

- H3: The KnowledgeWar game has a positive effect on learning

C. The Results

To assess the usability of our KnowledgeWar game we used the System Usability Scale (SUS) [24]. SUS measures usability through ten statements, which the subject is to state a degree of agreement by using the Likert scale (from Strongly disagree=1 to Strongly agree=5). Odd statements contribute with their "average value - 1", and the even statements with "5 - average value". These contributions are multiplied by 2.5 to get a score between 0 and 100 points where higher is better. Table II shows the SUS statements and the scores we got from the student questionnaires.

TABLE II. SUS STATEMENTS AND SCORES

Statement	Average	SUS Contribution
Q1 I think that I would like to play this game frequently.	3.50	6.25
Q2 I found the game unnecessary complex.	1.88	7.81
Q3 I thought the game was easy to use.	4.13	7.81
Q4 I think that I would need the support of a technical person to be able to use this game application.	1.50	8.75
Q5 I found the various functions in this game were well integrated.	3.63	6.56
Q6 I thought there was too much inconsistency in this game.	1.86	7.86
Q7 I would imagine that most people would learn to use this game application very quickly.	4.38	8.44
Q8 I found the game application very cumbersome to use.	1.88	7.81
Q9 I felt very confident using the game application.	4.25	8.13
Q10 I needed to learn a lot of things before I could get going with this game.	1.38	9.06
	SUS score	78.48

KnowledgeWar received a SUS score of 78 points (see Table II) out of 100. According to Bangor, Kortum and Miller [25], our score is well within the acceptable range on their SUS score, and is about in the middle between the markers for good and excellent. The three biggest contributors to the SUS score was:

- Disagree with the statement: "I needed to learn a lot of things before I could get going with this game". This statement contributed 9.06 pts.
- Disagreement with the statement: "I think that I would need the support of a technical person to be able to use this game application". This statement contributed 8.75 pts.
- Agreement with the statement: "I would imagine that most people would learn to use this game application very quickly". This statement contributed 8.44 pts.

The statement with the lowest contribution was the agreement with the statement: "I think that I would like to play this game frequently" (only 6.25 pts). Hopefully it is possible to improve this part by improving the graphical presentation of the game, introducing new game modes, and populating the database with more entertaining and engaging questions. Another possible explanation for the low score could be that

the test was not carried out as a part of a course, and the questions were not taken from their current courses.

Although the number of participants in this study was fairly low (n=8) and there are some sources of errors in how the subjects perceived the game, the SUS score strongly indicates that a quiz game like KnowledgeWar is suitable on the smart phone platform.

The next part of the assessment was to investigate the subjects' perceptions of using a game like KnowledgeWar as a part of a class or for teaching purposes. This assessment was made through ten additional (to SUS) statements in the questionnaire. The results are shown in Table III.

TABLE III. KNOWLEDGEWAR GAME AND LEARNING ASSESSMENT

Statement	Average	% (Strongly) Agree
N1 This game can create healthy competition in a class.	4.25	75%
N2 This game can stimulate to involvement in a class.	4.13	88%
N3 I would attend more lectures, if they were supported by such a game.	3.13	38%
N4 This game would help summarize a topic after a lecture.	4.38	100%
N5 The use of mobile smartphones is a good platform for increased motivation in a class.	4.13	88%
N6 I liked the idea of challenging other fellow students.	4.25	75%
N7 I would like to challenge the teacher through this game.	3.75	75%
N8 This game was social.	4.25	88%
N9 Every course at NTNU should have a game like this.	3.75	63%
N10 I would pay more attention in the lecture, if I could play such a game after the lecture.	3.38	50%

Table III shows the results of the additional non-SUS statements in the user test. These statements were added to the questionnaire specifically to address if “*The game would be a fun way of practicing knowledge*” (success criteria H2) and if “*The KnowledgeWar game has a positive effect on learning*” (success criteria H3). We will first look into the former. For the statements 1, 6 and 7, 75% of the participants check the strongly agree box, “This game can create healthy competition in a class”, “I like the idea of challenging other fellow students” and “I would like to challenge the teacher through this game”, which clearly support H2. 88% strongly agreed to statement 8 that “This game was social”. Improved social integration in a class is desirable from both teacher and students support that also support H2.

If we consider success criteria H3, we find that only 38% strongly agreed to statement 3, that “They would attend more lectures if it was supported by a game like KnowledgeWar”. It seems that the game in its current version is not likely to improve lecture attendance. However, every subject strongly agreed to that “This game would help summarize a topic after a lecture” (statement 4), which is very encouraging. 88% of the subjects strongly agreed to “This game might stimulate to better involvement” (statement 2) and “The use of mobile smart phones as platform” (statement 5). These are very positive results that encourage us to continue the development of such games in the future.

V. EVALUATION AND DISCUSSION

This section presents evaluation and discussion of using iPhone as a development platform for lecture games and how smart phone affects the usability of lecture games.

A. Developing for the iPhone

Developing for the iPhone proved to be an interesting learning experience. The platform has a number of good characteristics that aid the development of rich and powerful mobile applications. Some of the good features of the platform that deserve to be highlighted are:

- Excellent development tools for both code and GUI.
- Good run-time environment, desirable features at place for the developer.
- Well-documented and mature platform.
- Good process support for certificate generation and provisioning profiles with web portal.
- Provided service for standardized push communication to the mobile device.

What might be left as arguments against choosing the iPhone as platform for a mobile application might include:

- Application guidelines restrict the use of undocumented system libraries, which exclude certain hardware and system information (like visible wifi access points).
- The lack of automatic garbage collector and not so widespread programming language (object C).
- Intel Mac is required to run the development toolkit Xcode & Interface Builder. No support for Windows or Linux operating systems.

The KnowledgeWar game was developed for the iPhone OS version 3.2.3. A major restriction for this version was the lack of support for multitasking. This problem has been solved in the iOS4 (iPhone OS version 4) release where multitasking was introduced. However, the multitasking is still limited as it can only be used to execute some specified services. Here are some new features in OS4 that could have improved our game:

Text messaging - A pre-filled type and send Short Message Service (SMS) screen could have been included in one of the application views. This feature could have been used to improve the social aspects of the KnowledgeWar game.

Background processes – multitasking – is one of the most important contribution of the iOS4. Instead of completely exiting and destroying applications when pressing the Home button, applications can continue to run in a background context. An event is raised informing the application about the state change. Applications that require to prevail can do so by three techniques: (i) *Schedule local notifications* to alert the user of activity, basically a local, scheduled version of the Apple Push Notification Service. Possible usage scenario could be alerting the user about it is time to

play another round of quiz game; (ii) *It can request to run for more time* to complete some important task that will take more time than what is allowed during shutdown, e.g. writing large amounts of data to disk; and (ii) *It can become a background service* that will be awaked at specific intervals or at a specific time. This will allow e.g. background updates to a web service or similar.

The communication and back-end solution was designed and implemented with flexibility and agility as one of the main criteria. Best practices from the respective developer communities have been adopted into this solution's requirements. The flexibility and agility has been provided by using the Spring architecture and the Maven build system. By following Java standards, the solution should also be ready for future changes in the technologies and libraries. Implementing an additional web service or a new server with this solution as a mold, serving a completely different purpose, with its own domain model and another transportation format is trivial and would require only a minimal effort. We have not executed any substantial performance and load tests, but we do not believe that this will be a major problem as it is not likely that hundreds of players will play the game simultaneously. If an alternative component emerge who offers speed, functionality or other improvements, it can be swapped with the original component with no or little source code changes.

Based on experience from this project and others, we find that the iPhone platform is an extensive and mature platform for application and game development. No major technical issues were encountered during the implementation that was impossible to overcome. The solutions provided to the developers for technical issues, like the addressing and delivery of server-initiated notifications, further elaborated the maturity and quality of the frameworks and tools provided for applications and application developers.

Most of the limitations we encountered such as no support for background process have been addressed in iOS4. Other issues like undocumented system libraries and time-consuming application approval process still put extra burden on the developer, but these restrictions can be beneficial for the end-user with a more homogeneous and reliable end product. In some applications these properties can be worked around, but in others they are deal breakers that will force the use of another platform.

Research projects that do not demand, or will pass an application approval process have alternative means of distribution which will lift some of the restrictions as presented in this paper, but imply other restrictions like maximum one hundred pre-registered ad hoc users.

The push functionality in our game was implemented by using Apples Push Notification Service (APNS). The APNS was selected for addressing and routing simplicity as a server-initiated channel and it proved to be easy to use, reliable and fast. The second communication channel was a lightweight REST/JSON web service, which was designed and implemented with flexibility and extensibility in mind. This worked also very well in our architecture.

B. Usability and Learning

The KnowledgeWar game was tested by a small number of subjects during a usability experiment before they answered a questionnaire about usability and the use of games in an educational context. Overall, the feedback from the experiment was very positive, especially on the general usability. We found the smart phone platform in general and iPhone/iPod Touch platform specifically to be well suited for lecture games. The main benefits of using this platform is that the platform is popular with the students, it provides high usability if the applications are designed correctly, it has a large screen that is well suited for quiz games with texts of varying lengths, and it is a cheap platform for doing large scale tests with own equipment. A major headache when doing large-scale user tests with mobile equipment is to provide enough sim cards for several mobile phones or smart phones. It might be possible to get a limited set of non-functional sim cards to opens the phones so they can be used in tests where only the wifi network is used. However, if applications are developed for the iPhone/iPod Touch platform, iPod Touch devices can be used for testing. These devices cost about the half or less than smart phones, they have all the functionality of smart phones apart from being able to make calls, and they can be used for testing right out of the box. The major limitation of using iPod Touch as a deployment platform is that the device does not come with a built-in GPS. This means that the iPod Touch works well for testing location-aware applications that do not require very accurate positioning (accuracy of 15-20 meters and better) or positioning without wifi coverage.

The location-awareness functionality in the KnowledgeWar game is designed not to be intrusive by showing the physical distance to other players connected to the game server. This makes it possible for players to choose to play the game face-to-face or remote. With iOS4, we could have improved the location-awareness functionality of the game. One improvement could be to let the game run in background and notify the user when other players are nearby. Similarly, we could have extended the game lobby to allow users to create friend lists. This could stimulate players to build a community of players that enjoy playing against each other.

The KnowledgeWar game it self is not a very complex game. What we would like to add in the future is a role-playing aspect where the players level up and get new features and game modes as they progress. The goal of the game will then be to level up and be the king of the hill (the best overall player). Another feature to make it a more interesting experience is to add AI-controlled knowledge monsters that players have to beat to level up. Some of these monsters might be very powerful and skillful and could involve knowledge fights where several players have to collaborate to beat the monster (similar to quests in MMORPGs). There are limitless opportunities of expanding the game. In this design process, the main drivers is to make the game more social and hook the players into the game by providing new and interesting features as they level up (rewards). Further, we could provide every player with an avatar that levels up as the game progress, which will also improve players vs. player encounters. In this way, we hope that this will be a game motivating students to spend more time studying, just to beat the game.

VI. CONCLUSION

The KnowledgeWar game was designed with three success criteria in mind: H1: "The game application should have high usability", H2: "The game should be a fun way of practicing knowledge", and H3: "The KnowledgeWar game should have a positive effect on learning".

The results from our user test shows that the game has high usability (close to 80% SUS score). However, we identified some areas that must be improved. The statements with lowest scores were "I would like to play this game frequently" (6.25 point) and "I found the various functions in the game well integrated" (6.56 points). These statements indicate that the coherence of the game application must be improved along with providing more engaging and variable gameplay. We believe adding role-play elements to the game including leveling up, new game modes, and challenges against AI knowledge monsters can solve this problem. Further, we believe that improving the social aspects of the game will also make students to play the game more frequently. This can be done by adding friend lists and opening for collaborative gameplay.

The scores from the non-usability statements were overall positive in relation to the game being a fun way of practicing knowledge and to have a positive effect on learning. The students liked that the game was social and that this was an engaging way of rehearsing theory. The statements that scored the lowest were "I would attend more lectures, if they were supported with such a game" (38%) and "I would pay more attention in the lecture, if I could play such a game after a lecture" (50%). We believe that these statements scored low as the user test was not carried out in the context of a course, and this issue can be improved if we make the game more competitive and introduce a clearer overall goal of the game (e.g. to be the king of the hill). Especially, if we introduce a prize for the best player of the semester, and if the questions are directly linked to the lecture, we believe that lecture attendance and paying attention during lectures will improve.

REFERENCES

- [1] R. Rosas, M. Nussbaum, P. Cumsille, V. Marianov, M. Correa, P. Flores, V. Grau, F. Lagos, X. López, V. López, P. Rodriguez, and M. Salinas, "Beyond Nintendo: design and assessment of educational video games for first and second grade students," *Computer Education*, vol. 40, pp. 71-94, 2003.
- [2] M. Sharples, "The design of personal mobile technologies for lifelong learning," *Comput. Educ.*, vol. 34, pp. 177-193, 2000.
- [3] A. Baker, E. O. Navarro, and A. v. d. Hoek, "Problems and Programmers: an educational software engineering card game," in *Proceedings of the 25th International Conference on Software Engineering* Portland, Oregon: IEEE Computer Society, 2003.
- [4] L. Natvig, S. Line, and A. Djupdal, "Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals," *Proceedings of the 2004 Frontiers in Education Conference*, 2004.
- [5] E. O. Navarro and A. v. d. Hoek, "SimSE: an educational simulation game for teaching the Software engineering process," in *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* Leeds, United Kingdom: ACM, 2004.
- [6] B. A. Foss and T. I. Eikaas, "Game play in Engineering Education - Concept and Experimental Results," *The International Journal of Engineering Education* vol. 22, 2006.
- [7] G. Sindre, L. Nattvig, and M. Jahre, "Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals," *IEEE Transaction on Education*, vol. 52, pp. 10-18, 2009.
- [8] A. I. Wang, T. Øfsdal, and O. K. Mørch-Storstein, "Lecture Quiz - A Mobile Game Concept for Lectures," in *IASTED International Conference on Software Engineering and Application (SEA 2007)* Cambridge, MA, USA: Acta Press, 2007, p. 6.
- [9] A. I. Wang, T. Øfsdal, and O. K. Mørch-Storstein, "An Evaluation of a Mobile Game Concept for Lectures," in *Proceedings of the 2008 21st Conference on Software Engineering Education and Training - Volume 00*: IEEE Computer Society, 2008.
- [10] M. S. El-Nasr and B. K. Smith, "Learning through game modding," *Computer Entertainment*, vol. 4, p. 7, 2006.
- [11] B. Wu, A. I. Wang, J.-E. Strøm, and T. B. Kvamme, "An Evaluation of Using a Game Development Framework in Higher Education," in *Proceedings of the 2009 22nd Conference on Software Engineering Education and Training - Volume 00*: IEEE Computer Society, 2009.
- [12] N. Holmes, "Digital Technology, Age, and Gaming," *Computer*, vol. 38, pp. 108-107, 2005.
- [13] A. Sliney, D. Murphy, and J. Doc, "A Serious Game for Medical Learning," *First international Conference on Advances in Computer-Human interaction*, February 10-15 2008.
- [14] F. Mili, J. Barr, M. Harris, and L. Pittiglio, "Nursing Training: 3D Game with Learning Objectives," *Proceedings of the First international Conference on Advances in Computer-Human interaction*, pp. 10-15, 2008.
- [15] L. v. Ahn, "Games with a Purpose," *Computer*, vol. 39, pp. 92-94, 2006.
- [16] Y. S. Chee, E. M. Tan, and Q. Liu, "Statecraft X: Enacting Citizenship Education Using a Mobile Learning Game Played on Apple iPhones," in *Proceedings of the 2010 6th IEEE International Conference on Wireless, Mobile, and Ubiquitous Technologies in Education*: IEEE Computer Society.
- [17] M. J. McAlister and P. H. Xie, "Using a PDA for Mobile Learning," in *Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education*: IEEE Computer Society, 2005.
- [18] G. Schwabe and C. Göth, "Mobile learning with a mobile game: design and motivational effects," *Computer Assisted Learning*, vol. 21, pp. 204-216, 2005.
- [19] R. E. Grinter, P. M. Aoki, M. H. Szymanski, J. D. Thornton, A. Woodruff, and A. Hurst, "Revisiting the visit: understanding how technology can shape the museum visit," in *Proceedings of the 2002 ACM conference on Computer supported cooperative work* New Orleans, Louisiana, USA: ACM, 2002.
- [20] B. Brown, I. MacColl, M. Chalmers, A. Galani, C. Randell, and A. Steed, "Lessons from the lighthouse: collaboration in a shared mixed reality system," in *Proceedings of the SIGCHI conference on Human factors in computing systems* Ft. Lauderdale, Florida, USA: ACM, 2003.
- [21] P. Thornton and C. Houser, "Using mobile phones in English education in Japan," *Computer Assisted Learning*, vol. 21, pp. 217-228, June 2005.
- [22] S. Fallahkhair, L. Pemberton, and R. Griffiths, "Development of a cross-platform ubiquitous language learning service via mobile phone and interactive television," *Computer Assisted Learning*, vol. 23, pp. 312-325, August 2007.
- [23] J. Sánchez, A. Salinas, and M. Sáenz, "Mobile game-based methodology for science learning," in *Proceedings of the 12th international conference on Human-computer interaction: applications and services* Beijing, China: Springer-Verlag, 2007.
- [24] J. Brooke, "SUS - a quick and dirty usability scale," in *Usability Evaluation in Industry*: Taylor and Francis London, 1996, pp. 189-194.
- [25] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *Usability Studies*, vol. 4, pp. 114-123, 2009.

Paper 3:

G3: Bian Wu; Alf Inge Wang; Erling Andreas Børresen; Knut Andre Tidemann: "Improvement of a Lecture Game Concept - Implementing Lecture Quiz 2.0", 3rd International Conference on Computer Supported Education, 6-9 May 2011, Noordwijkerhout, The Nederland. ISBN: 978-989-8425-50-8

IMPROVEMENT OF A LECTURE GAME CONCEPT

- Implementing Lecture Quiz 2.0

Bian Wu, Alf Inge Wang

Dept. of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway
bian@idi.ntnu.no, alfw@idi.ntnu.no

Erling Andreas Børresen, Knut Andre Tidemann

Dept. of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway
erling.andreas.borresen@stud.ntnu.no, knut.andre.tidemann@stud.ntnu.no

Keywords: Educational game, Multiplayer game, Computer-supported collaborative learning, Software engineering education, Evaluation, System Usability Scale (SUS).

Abstract: A problem when teaching in classrooms in higher education is lack of support for interaction between the students and the teacher during the lecture. We have proposed a lecture game concept that can enhance the communication and motivate students through more interesting lectures. It is a multiplayer quiz game, called Lecture Quiz. This game concept is based on our current technology rich and collaborative learning environment and was proved as a viable concept in our first prototype evaluation. But based on our previous implementing experiences and students' feedbacks about this game concept, it was necessary to improve this first lecture quiz prototype in four aspects: 1) Provide a more extensible and stable system; 2) Easier for students to start and use; 3) Easier for the teachers to use; and 4) Good documentation to guide the further development. According to these aims, we developed the second version of Lecture Quiz and carried out an evaluation. Through comparing the evaluation data from second version with first version of Lecture Quiz, we found that both surveys show that the Lecture Quiz concept is a suitable game concept for improving lectures in most of aspects and that Lecture Quiz have been improved in several ways, such as editor for the teachers to update the questions, improved architecture that could be easy to extend to the new game modes, web-based student clients to get an easier start than first version of lecture quiz, etc. The results are encouraging for further development of the Lecture Quiz platform and for exploring more in this area.

1 INTRODUCTION

Traditional educational methods may include lecture sessions, lab sessions, and individual and group assignments, in addition to exams and other standard means of academic assessment. From experiences at our university, we acknowledge that today's lectures mostly use slides and electronic notes and can still be classified as one-way communication lectures. In a typical lecture the teacher will talk about a subject, and the students will listen and take notes.

However, the exclusive use of such methods may not be ideally suited to today's students, particularly those in the generation born after 1982, or "Millennial students," as termed by education researchers (Raines; Oblinger and Oblinger, 2005; D. Oblinger, 2003). Millennial students prefer

hands-on learning activities, and collaboration in education and the workplace. Female, African American, Hispanic, and other underrepresented students may also be inclined toward ways of learning and working that involve more group work and social interaction than traditional university education provides (Williams et al., 2007).

The technology has now evolved and smart phones, laptops and wireless networking have become an integrated part of students' life. These technologies open new opportunities for interaction during lectures. As game technology is becoming more important in university education, we proposed a way to make the lecture more engaging and interactive. In 2007 we developed Lecture Quiz, an educational multiplayer quiz game prototype (Wang et al., 2007; Wang, 2008) denoted as LQ 1.0 in this paper. It provides a possibility for the students to

participate in a group quiz using their mobile phone or laptop to give an answer. The questions are presented on a big screen and the teacher has the role as host of a game show. This prototype was created in a hastily manner to prove that the game concept was viable for educational purposes.

The implementation of LQ 1.0 was clearly a prototype that was made as a primary proof of concept. It lacked a good structure to serve as a platform for various lecture quiz games and was not designed with extendibility and modifiability in mind. As everything was hard-coded, it was difficult to extend this prototype to be used at a larger scale. There were also issues with an unstable application and the architecture itself was not built for large-scale usage. This could be easily identified by some limitations, such as that only one session was allowed per server and no ability for the teacher to edit quiz data without hacking into the database. We wanted to extend its structure to support playing many lectures at the same time. Besides of these limitations of the software architecture, we also wanted to add new features to save preparation time to make a quick and easy start of the game, provide guidance for the new developers, and quiz editor tools for teachers. In the light of this, the main aim was to develop second version of Lecture Quiz, denoted LQ 2.0, providing the following features: 1) A more extensible and stable system with a suitable architecture; 2) Easier to start and use; 3) Easier for the teachers to use; and 4) Good documentation as reference for the further development in future, such as add new game modes to the system.

The final goal was to give a good and solid base as an extendable lecture game platform, thus hopefully making it a regular part of university lectures.

2 RELATED WORKS

Here we will present a survey of similar approaches for lectures, the design criteria for Lecture Quiz and introduce the previous version and the improvements of second version.

2.1 Literature Review

There was no paper describing exactly the same game concept using the technical infrastructure in lecture halls for higher education when we implemented LQ 1.0. During implementing LQ 2.0 in 2010, we found some new similar quiz games used in education in different ways, but excluding the quiz used without any technology, such as

(Schuh et al., 2008) or the quiz development frameworks, such as Quizmaker (Landay, 2010).

Using a game in a portable console (Larrazza-Mendiluze and Garay-Vitoria, 2010) describes an educational strategy that directly situates students in front of a game console, where the theoretical concepts will be learned collaboratively through a question and answer game. PCs and Nintendo DS consoles were compared.

Moodle (Daloukas et al., 2008) is an online open source software aiming at course management. It focuses on a game module consisting of eight available games, which are "Crossword", "Hangman", "Snakes and Ladders", "Cryptex", "Millionaire", "The hidden picture", "Sudoku" and "Book with questions". Their data are derived from question banks and dictionaries, created by users, both teachers and students. The rationale behind the design is to create an interactive environment for learning various subjects. Since learners are accustomed and attracted to gaming as well as they are able to gain immediate feedback on their performance, they should be easily engaged in them.

The baseball game (Han-Bin, 2009) implements an learning platform for students by integrating a quiz in virtual baseball play. Students can answer questions to get higher possibilities to win the game. The higher percentage they made right decisions, the better performance can be made in the baseball game. By integrating authoring tools and gaming environment, students will be focused in the contents provided by teachers.

Also, we found some related approaches prior to 2008 based on technology rich environment, described in LQ 1.0 (Wang, 2008). Such as, the Schools Quiz (Boyes, 2007), Quiz game for Medical Students (Roubidoux et al., 2002), the TVREMOTE Framework (Bar et al., 2005), Classroom Presenter (Linnell et al., 2007), WIL/MA(Lab), ClassInHand (UNIV.), ClickPro (AclassTechnology). Only the first two cases are designed as games.

2.2 Criteria for the Game design

Our lecture game concept intends to improve the non-engaging classroom teaching by collaborative gaming. And its design is based on the eight elements that make the games more fun to learn.

2.2.1 Collaborative Gaming for Learning

Today's Millennial students (Raines; Oblinger and Oblinger, 2005; D. Oblinger, 2003) have changing preferences for education and work environments that negatively affect their enrolments and retention rates into university course programs. To better suit these preferences, and to improve the lecture's

educational techniques, teaching methods and tools outside of the traditional lecture sessions and textbooks must be explored and implemented. Currently, both work on serious games and collaborative classrooms focus on this issue. The proposed lecture game concept deals with both serious games and student collaboration research, proposing that educational games with collaborative elements (multiplayer games) will take advantage of the benefits offered by each of these areas. The result is an educational game that demonstrates increased learning gains and student engagement above that of individual learning game experiences. Collaborative educational games and software also have the potential to solve many of the problems that collaborative work may pose to course instructors in terms of helping to regulate and evaluate student performance (Nickel and Barnes, 2010).

Currently, research into the combination of serious games and collaborative work (for example, collaborative, or multiplayer educational games) is an underexplored area, although recently, computer-supported collaborative learning (CSCL) researchers have begun investigation how games are designed to support effective and engaging collaboration between students. Studies on social interaction in online games like Second Life (Brown and Bell, 2004), or World of Warcraft (Bardzell et al., 2008; Nardi and Harris, 2006) reveal how multiplayer games allow players to use in-game objects to collaboratively create new activities around them, and how social interaction in the games is facilitated and evolving. From these studies, we learn how to create multi-player games that effectively support, or even require, collaboration between players.

Collaboration does not necessarily mean competition between teams, or otherwise an adversarial approach (Manninen and Korva, 2005) in the virtual environment, like above online multiplayer games. In the real world, a goal that requires a collaborative process, like solving a puzzle does create a conflict in the form of the interaction within the game (C. Crawford, 1982), but it is not a contest amongst adversaries. The team has to cooperate to reach a common goal. Up until recently, the lack of proper means of communication and interaction has made it difficult to support collaboration in computer games, and there exist few actual true collaborative games on the market. So we would like to explore this issue by a case study of using multiplayer quiz game in lecture to see what will happen when combining serious game with collaborative works in the physical world.

2.2.2 Characteristics of Good Educational Games

This section presents eight important characteristics of good educational games based on computer supported collaborative learning and Malone's statements of what makes games fun to learn. The following list of characteristics is extracted as a reference for people designing educational games, shown in Table 1. Note that missing one of the characteristics may not mean that the game will be unpopular or unsuccessful, but including the missing characteristics in the game concept may make it better.

Our lecture games concept, both in LQ 1.0 and 2.0 are designed based on these characteristics.

Table 1: Characteristics of good educational games

ID	Educational Game elements	Explanation	Reference
1	Variable instructional control	How the difficulty is adjustable or adjusts to the skills of the player	(Thomas, 1980; Lowe and Holton., 2005)
2	Presence of instructional support	The possibility to give the player hints when he or she is incapable of solving a task	(Lowe and Holton., 2005; Privateer, 1999)
3	Necessary external support	The need for use of external support	(Lowe and Holton., 2005)
4	Inviting screen design	The feeling of playing a game and not operating a program	(Lowe and Holton., 2005)
5	Practice strategy	The possibility to practice the game without affecting the users score or status	(Lowe and Holton., 2005; Privateer, 1999)
6	Sound instructional principles	How well the user is taught how to use and play the game	(Lowe and Holton., 2005; Boocock and Coleman, 1966; J Kirriemuir and McFarlane, 2003; Schick, 1993)
7	Concept credibility	Abstracting the theory or skills to maintain integrity of the instruction	(Elder, 1973)
8	Inspiring game concept	Making the game inspiring and fun	(Thomas, 1980; Kirriemuir, 2004)

2.3 Lecture Quiz 1.0

The developed prototype of LQ 1.0 consisted of one main server, a teacher client and a student client. To begin a session the students had to download the student client to their phone using Wifi, Bluetooth or the mobile network (GPRS/EDGE/3G). After the download was finished, the software had to be installed before the students were ready to participate. This was seen as a bit of a cumbersome process. The teacher client was implemented in Java and used OpenGL to display graphics on a big screen.

The prototype implemented two game modes. In the first game mode, all the students answered a number of questions. Each question had its own time limit, and the students had to answer within that time. After all the students had given their answers, a screen with statistics was displayed providing information on how many students that answered on each option. At the end of the quiz, the teacher client displayed a high-score list.

The other game mode was named “*last man standing*”. The questions were asked in the same fashion as with the plain game mode, but if a student answered incorrectly, he or she was removed from the game. The game continued until only one student remained and was crowned as the winner.

One of the main drawbacks of LQ 1.0 was that it lacked a good architecture, making it hard to extend, modify and maintain. It also lacks good documentation, and there was not quiz editor to add a new quiz or a question. The teacher had to manually edit the data in the database. The time spent on downloading and installing the software on the students’ devices also made it less interesting for regular use in lectures.

2.4 Improvements of Lecture quiz

Firstly, LQ 2.0 was based on above design methods and lecture quiz concept. But according to previous experiences and students’ feedback, we made some improvements on these aspects. Table 2 shows the additional functional requirements in LQ 2.0.

Table 2: List of added new functional requirements

Functional requirement
A developer can extend the game with new game modes
A teacher can update the question through a quiz editor
A teacher can tag questions for easier reuse and grouping
A server should be able to run several quiz sessions at the same time

In addition to functional requirements, we defined some non-functional requirements for LQ 2.0 described as quality scenarios (Len Bass et al., 2003). Table 3, 4, and 5 shows three quality scenarios for modifiability respectively.

Table 3: Modifiability scenario 1.

M1 - Deploying a new game mode for a client	
Source of stimulus	Game mode developer
Stimulus	The game mode developer wants to deploy a new game mode for one of the Lecture Quiz clients or the server
Environment	Design time
Artefact	One of the Lecture Quiz clients or the game server
Response	A new game mode is deployed and should be ready for use
Response measure	The new game mode should be possible to be deployed in few hours

Table 4: Modifiability scenario 2.

M2 - Creating a new client	
Source of stimulus	Client developer
Stimulus	The client developer wants to create a new client for the Lecture Quiz game
Environment	Design time
Artefact	The Lecture Quiz service
Response	A new client supporting to play the Lecture Quiz game.
Response measure	The server communication part of the client should be complete within two days

Table 5: Modifiability scenario 3.

M3 - Adding support for a new database back-end	
Source of stimulus	Server developer
Stimulus	Server developer wants to add support for another database back-end
Environment	Design time
Artefact	The Lecture Quiz server
Response	A new option for database storage in the server
Response measure	The new back-end should be finished in two hours

Also we required including a guide explaining how to create a new game mode for the Lecture Quiz server as well as for the clients. Such a guide would make it possible for an external developer to create new game modes with minimal effort.

As the ability for further development and expansion of the Lecture Quiz framework was an

important part of our work, we decided to include detailed information on how this could be done. This information was intended for new developers wanting to pick up the Lecture Quiz system and continue development on the many aspects of it.

3 IMPLEMENTATION

In this section we describe how we have implemented the architecture for LQ 2.0. The main component in this architecture is the Lecture Quiz Game Service. The clients are implemented as flexible components that are easy to extend and improve. Figure 1 gives the system overview.

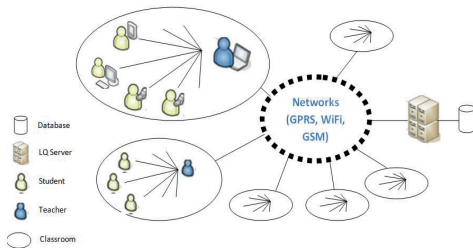


Figure 1: System overview of LQ 2.0

3.1 Lecture Quiz Game Service

The Lecture Quiz Game Service is the server component that handles all the game logic. Both teacher and student clients connect to this server through its web service API. The server itself is implemented in Java EE 6 and was running on the Apache Tomcat application server during development, but should be able to run on any Java web container.

3.2 Database Design

The database design is given in Figure 2. Based on five main tables in the database, we have added two

reference tables that help provide the needed relations between quizzes and questions, as well as the tags that could be as a new function for teachers to search certain questions.

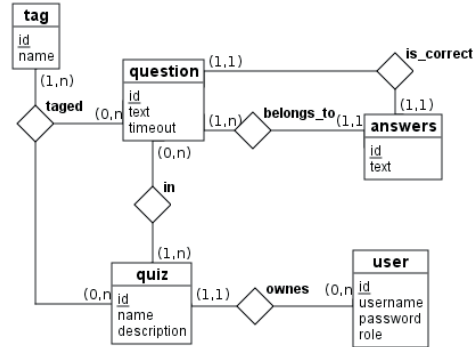


Figure 2: ER diagram of database

3.3 Student Client

The student client was developed in Java using the Google Web Toolkit4 (GWT) and the AJAX5 framework. As with the teacher client, the main focus of this implementation has been on functionality and providing a reference as of how a client can be implemented. Hence, the graphical design is minimalistic that also fits the small screens and easy to download content for mobile phones.

3.4 Teacher Client

The teacher client is developed in Java SE 6. The development mainly focused on the functional parts of the client. Implemented in the teacher client is a simple menu system, a quiz editor to create and edit quizzes and questions, and a single game mode. When the teacher client is started, a connection check is performed to make sure the application can reach the Lecture Quiz web service. Figure 3 shows the interface of teacher clients.

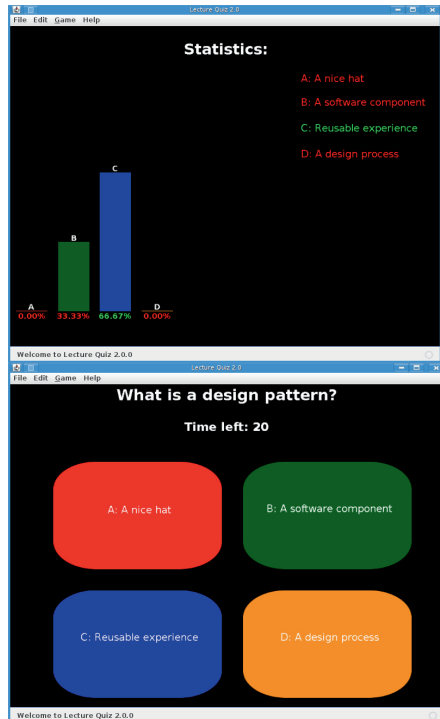


Figure 3: Screenshots from teacher client

4 EVALUATION

In this section we will present an empirical experiment where our system was tried out in a realistic environment and the findings we found.

4.1 Experiment Delimitation

The goal of this experiment was to get an overall picture of how the Lecture Quiz service and clients worked in a real life setting, and comparing it to the similar experiment conducted for LQ1.0 in 2007 (Wang et al., 2007; Wang, 2008). We will point out and discuss trends based on these results and our experiences. Statistical analysis and thorough psychological analysis are out of the scope of the current aim.

4.2 Experiment Method

The goal of the formative evaluation was to assess engagement and usability of lecture quiz concept

with a group of target users. The group of subjects included 21 students with average age of 22. The minimum number of participants was determined using the Nielsen and Landauer formula (Nielsen and Landauer, 1993) based on the probabilistic Poisson model:

$$\text{Uncovered problems} = N (1 - (1 - L)^n)$$

Where: N is the total number of usability issues, L is the percentage of usability problems discovered when testing a single participant (the typical value is 31% when averaged across a large number of projects), and n is the number of subjects.

Nielsen argues that, for web applications, 15 users would find all usability problems and 5 participants would reveal 80% of the usability findings. Lewis (Nielsen and Landauer, 1993) supports Nielsen but notes that, for products with high usability, a sample of 10 or more participants is recommended. For this study it was determined that testing with 15 or more participants should provide meaningful results.

Usability and enjoyment of a game are two closely related concepts. According to the ISO 9241-11 (Jordan et al., 1996) definition, usability is derived from three independent measures: efficiency, effectiveness, and user satisfaction.

- **Effectiveness** - The ability of users to complete tasks using the system, and the quality of the output of those tasks
- **Efficiency** - The level of resource consumed in performing tasks
- **Satisfaction** - Users' subjective reactions to using the system.

Also, there are various methods to evaluate the usability. To measure usability we chose the System Usability Scale (SUS) (Jordan et al., 1996), which is a generic questionnaire with 10 questions for a simple indication of the system usability as a number on a scale from 0 to 100 points. Each question has a scale position from 1 to 5. For items 1,3,5,7 and 9, the score contribution is given by subtracting from the scale position. For item 2,4,6,8 and 10, the contribution is 5 minus the scale position. This implies that each question has a SUS contribution of 0-4 points. Finally, the sum of the scores are multiplied by 2,5 and divided by the number of replies to obtain the SUS score. The questionnaire is commonly used in a variety of research projects.

4.3 Experiment

This experiment tested the usability and functionality of LQ 2.0. The experiment took place on May 2010.

The purpose of this experiment was to collect empirical data about how well our prototype worked in a real life situation, especially regarding usability and functionality.

4.3.1 Participants and Environment

The experiment was conducted in a lecture in the Software Architecture course at our university, and all the participants were students taking this course. 21 students took part of this experiment, where 81% were male and 19% were female. As the test was conducted in a class of computer science students, most of the students consider themselves to be experienced computer users, but none of the participants had tried the software before the experiment. The test was lead by the teacher, and he controlled the progress of the game with the teacher client running on a laptop and displayed the quiz on a big screen by a video projector. The students used own mobile phones or laptops to participate through a web browser supporting java script. The Lecture Quiz server was running on a computer located outside of the lecture room.

4.3.2 Experiment Execution

21 of the students in class agreed to participate in the experiment. The lecture was a summary lecture in the Software Architecture course. In the first part of the lecture, theory from current semester was summarized and discussed. The students were allowed to ask questions. The experiment took place in the second part of the lecture, after a short break.

The teacher client was started on a laptop, and an URL to the student client was shown on the projector. Each student logged in on the web client using a desired username and the quiz code

displayed on the large screen processed by the teacher's client.

The experiment was executed without any problems. Everyone was able to answer the questions using their own mobile clients, and there was a relaxed atmosphere in the room. Some of the answer options made the students laugh a bit. In one of the questions, the teacher client was not able to display the statistics and correct answer. But this was displayed correctly on the student client. The problem was solved by the next question, and all the software seemed to handle this issue well. All of the 21 students that took part of the experiment did also answer the questionnaire.

4.4 Results and Findings

We will present our results mainly on two aspects: the usability and usefulness of the LQ 2.0.

4.4.1 SUS Score and Student Feedbacks

In this section we will present the results of the questionnaire. First we explain how the SUS score was calculated. Most of statements had five choices for the user to answer. From strongly disagree to strongly agree. These choices were displayed in the graphs as values from 1 to 5 respectively, and -1 means that the user did not answer this question.

To calculate our SUS score we had to discard 6 of the 21 returned questionnaires, as they had not answered all of the questions included in the SUS part of the questionnaire. That made it 15 valid questionnaires for our SUS calculation.

Our software got a SUS score of 84 out of 100. This is displayed in Table 6, and shows how Lecture Quiz scored on each question along with the results from LQ 1.0 (the previous experiment).

Table 6: Lecture Quiz 1.0 and 2.0 SUS Scores

ID	Question	LQ 2.0		LQ 1.0	
		Avr	Score	Avr	Score
1	I think that I would like to use this system frequently	3.53	3.53	3.6	2.6
2	I found the system unnecessarily complex	1.40	3.6	1.85	3.15
3	I thought the system was easy to use	4.53	3.53	4.02	3.05
4	I think that I would need support of a technical person to be able to use this system	1.13	3.87	1.35	3.65
5	I found the various functions in this system were well integrated	3.73	2.73	3.2	2.2
6	I thought there was too much inconsistency in this system	1.73	2.73	1.95	3.05
7	I would imagine that most people would learn to use this system very quickly	4.73	3.27	4.35	3.35
8	I found the system very cumbersome to use	1.73	3.27	1.95	3.05
9	I felt very confident using the system	4.33	3.33	3.55	2.55
10	I needed to learn a lot of things before I could get going with this system	1.27	3.73	1.95	3.05
--	SUS score		84.00		74.25

LQ 2.0's SUS score of 84 shows that it has high usability. The SUS score of the experiment in LQ 1.0 was 74.25. LQ 2.0 does mainly the same things from the students' aspect, except that the student client is web-based. Thus we conclude that the web-based approach was a success.

Also, if we look closely to the questions: 3, 4, 7, 10 from Table 6, it shows that LQ 2.0 has the scores of 4.53, 1.13, 4.73 and 1.27 respectively. We find the results relatively clear. The people that answered our questionnaire found LQ 2.0 both easy to use and easy to getting started with. All of these results are somewhat better compared to previous LQ 1.0. It shows *the system is easy to getting started with and use*.

This was an encouraging result, but we still had to face some negative feedback from students on the LQ 2.0 experiment. Some of the students commented that the graphical design of the software was not good. Many students complained that the answer buttons were too small, although this could be solved using the zoom function in their web browser. We are fully aware that we are not graphical designers, and that major improvements could be done on this area. But our main focus in this system was to get the technical issues on the back-end done right.

There were also some complaints about the colour chosen as a background about option two on

the teacher client. This colour was displayed differently from the projector than on a standard computer screen and this made the text almost unreadable. In the experiment, the teacher read out all the choices, so that all the students did get the information they needed. The colour problem was corrected after the experiment by choosing a darker background colour for the teacher client to improve readability.

From the teacher's perspective, LQ 2.0 was clearly an improvement over LQ 1.0 as the time to start a quiz was shortened dramatically and there were no technical issues the teacher had to attend. The teacher only needed to put a URL on the blackboard or on the large screen, and then let the students log into the system. This meant that Lecture Quiz did not introduce a break during the lecture.

4.4.2 Results from Usefulness Questions

Our questions and results regarding usefulness of using Lecture Quiz both in LQ 1.0 and 2.0 are shown in Table 7. In this part of the survey, we looked at the students' attitude towards the game compared to the previous version. We also had an open question part where the students could come with their comments.

Table 7: Usefulness questions

ID	Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Version
1	I think that I am an experienced computer user	-	-	-	-	-	LQ 1.0
		0	0	5%	19%	76%	LQ 2.0
2	I think I paid closer attention during the lecture because of the system	10%	10%	30%	40%	10%	LQ 1.0
		5%	0	42%	32%	21%	LQ 2.0
3	I found the system had a distracting effect on the lecture	35%	35%	15%	10%	5%	LQ 1.0
		60%	25%	5%	5%	5%	LQ 2.0
4	I found the system made me learn more	5%	5%	40%	50%	0	LQ 1.0
		0	15%	25%	50%	10%	LQ 2.0
5	I think I learn more during a traditional lecture	5%	55%	25%	10%	5%	LQ 1.0
		15%	25%	40%	15%	5%	LQ 2.0
6	I found the system made the lecture more fun	0	0	5%	35%	60%	LQ 1.0
		0	0	10%	30%	60%	LQ 2.0
7	I think regular use of the system will make me attend more Lectures	15%	0	15%	45%	25%	LQ 1.0
		10%	15%	30%	20%	25%	LQ 2.0
8	I feel reluctant to pay 0.5 NOK in data transmission fee per lecture to participate in using the system	35%	15%	30%	10%	10%	LQ 1.0
		20%	25%	5%	20%	30%	LQ 2.0
-	LQ 2.0 question	Yes	No	If no, please describe the problem			
9	Did the client software work properly on your mobile/laptop?	90%	10%	Totally we got 20 responses, only two have problems.			

From question 2 and 3 in Table 7, we can find that *most students did not find the system intrusive in*

the lecture. Question 2 shows that most of the students (53%) thought they paid closer attention

during the lecture because of the system. We find this as a positive result, as this was more evenly distributed in LQ 1.0. And question 3 shows that over 80% disagreed in some way that the system had a distracting effect on the lecture, where 60% strongly disagreed. This is a slightly better result than survey data from LQ 1.0, where 70% disagreed to this statement in some way. We guess that having the quiz at the end of the lecture, and not having to change lecture room as in 2007, may be factors changing this result.

From question 4 and 5, we found that *lecture quiz have positive effect to the learning*. Over half students agree that they learned more from the system and the lecture quiz at least do not have negative effective on learning compared to traditional lectures.

Also from question 6 we found that *the students found the system inspiring and fun*. From both surveys of LQ 1.0 and LQ 2.0, we see a clear trend that students (over 90%) think using the lecture quiz system in lectures make them more fun.

From question 7 in the LQ 1.0 survey, the majority thought that regular use of the system would make them attend more lectures. But in LQ 2.0 survey, the distribution of answers was more even. We guess there are more factors that affect the attendance rate, and maybe game factor is not the biggest one. This proves that more research is necessary before we can make a valid result on this question,

From question 9, we found that *the system worked as it should*. Out of the 21 returned questionnaires, 18 reported that the software worked as it should on their devices. One did not answer, one meant that the software did not work because of the problem with small buttons in the mobile screen; this could be solved when he zoomed in the mobile browser, and one complained that the software did not work in Opera Mini. The reason for the problem in Opera Mini is that LQ 2.0 is based on AJAX and therefore needs java script support in the browser. In Opera Mini the requests are compressed and handled on a central server before being sent to the mobile device, and thus java scripts do not work. And this student switched the browser before the formal experiment starts.

During the experiment the teacher client failed to show the statistics for one of the questions once. But the statistics were displayed correctly on all the student clients, and all answers were stored as they should. The quiz continued as usual when the teacher pressed the button to start the next question. This is only a minor bug in the teacher client and that the rest of the system works as expected. We were not able to reproduce this bug later.

As a whole, we had less technical problems than the comparable experiment in 2007, thus probably resulting in the users to be friendlier in their evaluation of the system. The results of this experiment are mostly positive and in most areas better than for the previous version of the system.

5 CONCLUSION

Through the data from the evaluation and by comparing with the first version of lecture quiz, we found that lecture quiz is a suitable game concept to be used in lecture from both evaluation data.

And LQ 2.0 improved lecture game quiz concept in several ways. The main feature of building a strong and easily modifiable web-based architecture is extendable game modes, the ability to run multiple game servers on the same database and run many different quiz sessions on the same server. The new student web-based client reaches more students as close to 100% of students have access to a web-browser using a laptop or a mobile phone. In addition, the quiz editor makes it easy for teachers to maintain the question database, and it is easy to extend the game with the new game modes through the architecture. All of these features can be the factors that made the survey and evaluation better than the last version in most of aspects. More elaborate experiments must be conducted to find whether Lecture Quiz improves how much the students actually learn.

REFERENCES

- Raines, C. Managing Millennials; [cited October, 2010]. Available from: <https://www.cpcc.edu/millennial/presentations-workshops/faculty-or-all-college-workshop/9%20-%20managing%20millennials.doc>
- Oblinger, D.; Oblinger, J. Educating the Net Generation. Boulder, CO: Educause, 2005.
- D. Oblinger. Boomers, Gen-Xers, and Millennials: Understanding the New Students. Educause Review, July/August 2003, ; vol. 38, no. 4, :pp. 37-47.
- Williams, L.; Layman, L.; Slaten, K.M.; Berenson, S.B.; Seaman, C. On the Impact of a Collaborative Pedagogy on African American Millennial Students in Software Engineering. Proceedings of the 29th international conference on Software Engineering: IEEE Computer Society; 2007. p. 677-687.
- Alf Inge Wang; Terje Øfsdahl; Mørch-Storstein., O.K. LECTURE QUIZ - A Mobile Game Concept for Lectures., In 11th IASTED International Conference on Software Engineering and Application (SEA 2007).; Acta Press; 2007. p. pages 128-142.

- A.I. Wang, An Evaluation of a Mobile Game Concept for Lectures Software Engineering Education and Training, 2008 IEEE 21st Conference on; 2008. 197 p.
- Schuh, L.; Burdette, D.E.; Schultz, L.; Silver, B. Learning Clinical Neurophysiology: Gaming is Better than Lectures. *Journal of Clinical Neurophysiology*. 2008;25(3):167-169
110.1097/WNP.1090b1013e31817759b31817753.
- Landay, S. Online Learning 101: Part I: Authoring and Course Development Tools. *eLearn*. 2010;2010(6).
- Larraza-Mendiluze, E.; Garay-Vitoria, N. Changing the learning process of the input/output topic using a game in a portable console. Proceedings of the fifteenth annual conference on Innovation and technology in computer science education. Bilkent, Ankara, Turkey: ACM; 2010. p. 316-316.
- Daloukas, V.; Dai, V.; Alikanioti, E.; Sirmakessis, S. The design of open source educational games for secondary schools. Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments. Athens, Greece: ACM; 2008. p. 1-6.
- Han-Bin, C. Integrating baseball and quiz game to a learning platform. *Pervasive Computing (JCPC)*, 2009 Joint Conferences on; 2009. p. 881-884.
- Boyes, E. Buzz! spawns School Quiz.; [modified 2007]. Available from: <http://www.gamespot.com/news/6164009.html>
- Roubidoux, M.A.; Chapman, C.M.; Piontek, M.E. Development and Evaluation of an Interactive Web-Based Breast Imaging Game for Medical Students. *Academic Radiology*. 2002;9(10):1169-1178.
- Bar H.; Tews, E.; G. Robling. Improving Feedback and Classroom Interaction Using Mobile Phones. In *Proceedings of Mobile Learning*; 2005. p. 55-62.
- Linnell, N.; Anderson, R.; Fridley, J.; Hinckley, T.; Razmov, V. Supporting classroom discussion with technology: A case study in environmental science. *Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*, 2007 FIE '07 37th Annual; 2007. p. F1D-4-F1D-9.
- Lab, L. Wireless Interactive Lecture in Manheim: UCE Servers & Clients, Lecture Lab. WIL/MA; [cited 2007]. Available from: <http://www.lecturelab.de/>
- UNIV., I.A.W.F. ClassInHand: Wake Forest University; [cited 2007]. Available from: <http://classinhand.wfu.edu>
- AclassTechnology. EduClick-Overview; [cited 2010]. Available from: <http://www.aclasstechnology.co.uk/eduClick/index.html>
- Nickel, A.; Barnes, T. Games for CS education: computer-supported collaborative learning and multiplayer games. Proceedings of the Fifth International Conference on the Foundations of Digital Games. Monterey, California: ACM; 2010. p. 274-276.
- Brown, B.; Bell, M. CSCW at play: there as a collaborative virtual environment. Proceedings of the 2004 ACM conference on Computer supported cooperative work. Chicago, Illinois, USA: ACM; 2004. p. 350-359.
- Bardzell, S.; Bardzell, J.; Pace, T.; Reed, K. Blissfully productive: grouping and cooperation in world of warcraft instance runs. Proceedings of the 2008 ACM conference on Computer supported cooperative work. San Diego, CA, USA: ACM; 2008. p. 357-360.
- Nardi, B.; Harris, J. Strangers and friends: collaborative play in world of warcraft. Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work. Banff, Alberta, Canada: ACM; 2006. p. 149-158.
- T.Manninenand; T.Korva. Designing Puzzle for Collaborative Gaming Experience - CASE: eScape. DiGRA 2005 Conference: Changing Views - Words in play 2005. Vancouver, Canada, 2005.
- C. Crawford. *The Art of Computer Game Design*: Osborne/McGraw Hill; 1982.
- Thomas, W.M. What makes things fun to learn? heuristics for designing instructional computer games. Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems. Palo Alto, California, United States: ACM; 1980.
- Lowe, J.S.; Holton., E.F. A Theory of Effective Computer-Based Instruction for Adults. *Human Resource Development Review*., 2005;4(2), 159-188. .
- Privateer, P.M. Academic Technology and the Future of Higher Education: Strategic Paths Taken and Not Taken. *Journal of Higher Education*, Vol 70, . 1999.
- Boocock, S.S.; Coleman, J.S. Games with Simulated Environments in Learning. *Sociology of Education*. 1966;39(3):215-236.
- J Kirriemuir; McFarlane, A. Use of computer and video games in the classroom. Proceedings of the Level Up Digital Games Research Conference. Universiteit Utrecht, Netherlands.; 2003.
- Schick, J.B.M. The Decision to Use a Computer Simulation. Vol. Vol. 27, No. 1 (Nov., 1993), pp. 27-36 Society for History Education; 1993.
- Elder, C.D. Problems in the Structure and Use of Educational Simulation. *Sociology of Education*. 1973;46(3):335-354.
- Kirriemuir, J.M., A. Literature review in games and learning. Report 8.; 2004.
- Len Bass; Paul Clements; Kazman, R. *Software architecture in practice: Second Edition*: Addison-Wesley Professional; 2003.
- Nielsen, J.; Landauer, T.K. A mathematical model of the finding of usability problems. Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems. Amsterdam, The Netherlands: ACM; 1993. p. 206-213.
- Jordan, P.W.; Thomas, B.; Weerdmeester, B.A.; McClelland, A.L. Usability Evaluation in Industry, chapter SUS - A quick and dirty usability scale: CRC Press; 1996. pages 189-194. p.

Paper 4:

G4: Bian Wu, Alf Inge Wang, " A Pervasive Game to Know Your City Better", 2011 International IEEE Consumer Electronics Society's Games Innovation Conference (IGIC 2011), November 2011, Orange, California, USA.



A Pervasive Game to Know Your City Better

Bian Wu, Alf Inge Wang, Norwegian University of Science and Technology

Abstract— This paper presents a pervasive game on Android platform where players can play a knowledge competition tour in groups in the city of Trondheim, and gain better understanding of the city through solving different tasks. From the evaluation, the result shows that the concept of using pervasive game in a learning context is an interesting concept that should be explored.

I. INTRODUCTION

During recent years, there is a growing trend that can be referred to as pervasive and social games, which brings more physical movement and social interactions into game world [1]. Concretely, smart phones with Internet, GPS and other capabilities have become increasingly common, making mobile phone-based pervasive games easy to play and more interesting. Inspired by the game-based learning [2], one possible research area is to provide learning platform through pervasive games. In this context, we have a tentative case study to explain how learning is perceived and integrated in pervasive game.

There are two main inspirations for this case study: one is about the game plot, and another is about the new and interesting applications of mobile technology. The first is the American television series “The Amazing Race” (http://en.wikipedia.org/wiki/The_Amazing_Race), a reality show where contestants compete to be the first to reach different checkpoints all over the world. Similarity, the other two are: 1) a treasure hunt called “The Game” ([http://en.wikipedia.org/wiki/The_Game_\(treasure_hunt\)](http://en.wikipedia.org/wiki/The_Game_(treasure_hunt))), *Shelby Logan’s Run* is the 2002 edition of “The Game”, a Seattle-based yearly puzzle hunt. 2) A pervasive learning space called *Heroes of Koskenniska* [3], it combined mobile and sensor technologies with environmental education. Another motivation is popularity of Android platform and its applications. Its features can meet our requirements in different technology demanding scenarios in our case study. By getting contestants to travel to several different locations, we can thoroughly put the GPS-unit, Wi-Fi or 3G into work. In addition, recent interesting applications based on the camera, microphone and headphone from Android Market provide a multitude of other technologies that can be integrated in a pervasive game. For instance, 1) QR code and barcode can be scanned through phone’s camera, and we can use barcode generator to output clues for game tasks. 2) Google Goggles (<http://www.google.com/mobile/goggles>) is a free image recognition application. It enables the player to use pictures taken from the mobile phone to search on web resource; these pictures could come from text, landmarks, books, contact information, artwork, wine, or logo. 3) Layar (<http://www.layar.com/>) is a mobile platform for discovering

information about the world around us by using augmented reality technology, 4) Shazam (<http://www.shazam.com/>) is an application for recognizing songs that are playing, the application listens to music snippets through the microphone, and search the songs information. 5) ShopSavvy (<http://shopsavvy.mobi/>) is an extensive application from barcode category to scan the information of products using the camera of the mobile phone. After reading the barcode, the application will identify the product information and provide a list of online and local prices for it. In this context we introduced Trondheim city through a knowledge race called “The Amazing City Game” (ACG). The game is an adventure game where the contestants have to solve tasks at different locations by using relevant technologies from the Android phone. The group that reaches the final destination in the least amount of time is the winner.

II. DESIGN AND IMPLEMENTATION

The game has three main goals: 1) to integrate ubiquitous technologies from the Android platform in games, 2) to give the contestant knowledge about the city of Trondheim, and 3) to let the contestants have fun while playing the game.

Based on the above goals, we have constructed the following types of tasks for ACG. Each task may have 1-3 hints. If player uses a hint, a responding penalty time will be counted in the final score.

A. Tasks Design

Location Task: The player has to find a specific location and confirm it with the use of the GPS.

Scan Task: The player has to scan a barcode, text, figures or audio in order to get assigned a route or answer.

Open Task: The player is given a question and has to type answer into the answer text box.

Multiple Choice Tasks: The player is given a question and has to select the right answer out of the possible solutions.

Checkbox Task: The player is given a question and has to select the right answer out of the possible choices, where multiple answers might be correct.

Further, some tasks are combinations of above two or three types of tasks. E.g. Shazam Challenge: The player is given a question and has to type answer into the answer text box. The difference between this task and the Open Task is that the player is given an audio clue in the task description, and can be recognized by Shazam. ShopSavvy Challenge: The player is given a series of multiple-choice questions. The difference from other tasks is that the alternatives for each question are printed as barcode on a sheet of paper at the location, where each corresponding answer is next to a commercial product

picture that can be found by scanning the barcode. By taking the first letter of the each product name and grouped letters in correct order to be a word, and then they will give this word that is typed into a text box and get relevant information about the city.

For the final game play, one task is to know the city flag and city flower. Players will find the left picture from Figure 1, and they can open the Goggles application from the phone to scan this figure and search and find the web resource link. The correct link information shows that the figure is the city flag. If they read the information carefully, they will know the flower in the flag is *rosa canina*. Another task example related to the right side of Figure 1 is that audio will be broadcast and searched by Shazam and get the songs information and find famous musical writer from the city. To all tasks, another possible solution is that they can ask local passengers for help or search information from city library.

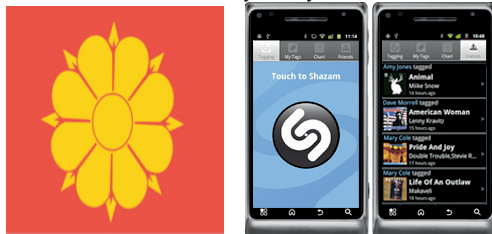


Figure 1: Trondheim flag (left) and Shazam (right)

B. User Interface

The user interface is clean and simple. Figure 2 shows examples of the interface: left top is the “Wrong Answer” pop up on a single choice task; right top is the Confirmation Box after choosing the answer. Left bottom is the GPS task interface and right bottom is the Shazam task.

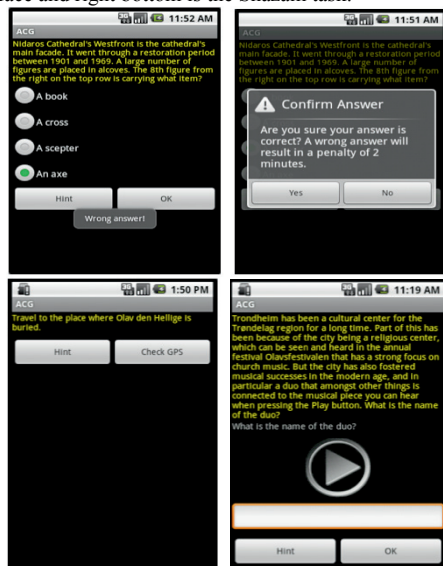


Figure 2: ACG user interface

III. RESULTS

A. Participants and Execution

The contestants were students with computer science background. There were four groups with two students in each group, totally eight students. Four of them were Norwegian, two were Spanish, one was Chinese and the one was Lithuanian. Each group had one Android phone with ACG installed. The game play was set from 1:15pm to 4:15pm on 3rd of May, and took place in Trondheim city of Norway. All groups started at meeting point. When all groups were ready, a brief introduction was given, and the first location disclosed. Immediately after this, everyone raced off to this location.

Upon arriving at the location, each group received different routes according to their arrival time. The groups started solving the tasks, and observed and recorded closely by the tutors. When the task at first location was finished, the groups continued with unique routes. From this point, each group was alone with their tutor for the rest of the game play (Tutor followed the group and recorded the video about the group's activity for the later observation). To the left in Figure 3 is a group is using the camera of an Android phone to scan a barcode. To the right a group is asking for help from a person working in a Tourist Information Center.

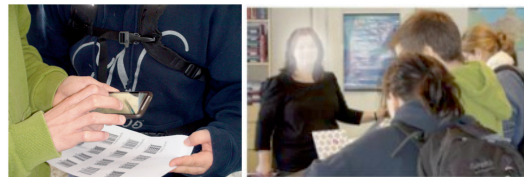


Figure 3: ACG play process

B. Results

Most of the groups spent 2-3 hours in the city tour game. From several observations, the GPS accuracy did not reach participants' expectation. Also the participants' background was not at the same level for the competition: E.g. some tasks were difficult for the foreigners since they did not have relevant culture background, while other participants were unfamiliar with the android applications. Overall, participants thought the tasks were a bit challenge but interesting. They claimed to have gained a better understanding of the city and more interested in android technology.

IV. SURVEY AND EVALUATION

A survey was conducted to evaluate our game system. The survey includes two parts: 1) System usability, and 2) Enjoyment of an educational game.

The System Usability Scale (SUS) [4] has previously been used to evaluate the usability of games, e.g. [5-7]. SUS is a generic questionnaire with 10 questions for a simple indication of the system usability as a number on a scale from 0 to 100 points. Each question has a scale position from 1 to 5. For items 1,3,5,7 and 9, the score contribution is given by subtracting 1 from the scale position. For item 2,4,6,8 and 10,

the contribution is 5 minus the scale position. This implies that each question has a SUS contribution of 0-4 points. Finally, the sum of the scores are multiplied by 2,5 and divided by the number of replies to obtain the SUS score.

We used the EGameFlow scale to measure the enjoyment of our educational game [8]. It is a scale that measures the enjoyment offered by E-learning games, and helps the game designer to understand the strengths and weaknesses of the game efficiently from the learner's point of view. EGameFlow consists of a number of questions in eight areas. The eight areas of EGameFlow are:

- **Concentration:** Games must provide activities that encourage the player's concentration while minimizing stress.
- **Goal Clarity:** Tasks should be clearly explained from the beginning.
- **Feedback:** Feedback allows a player to determine the gap between the current stage of knowledge and the knowledge required for completion of the task.
- **Challenge:** The game should offer challenges that fit the player's skill level, the difficulty of these challenges should change in accordance with the increase in the player's skill level.
- **Autonomy:** The learner should enjoy taking the initiative in game-playing and asserting total control over his or her choices in the game.
- **Immersion:** The game should lead the player into a state of immersion.
- **Social Interaction:** Tasks in the game should become a mean for players to interact socially.
- **Knowledge Improvement:** The game should increase the player's level of knowledge and skills while meeting the goals of the curriculum.

To answer the questions or statements in each area, the respondents have to express their degree of agreement or disagreement. Each item in the questionnaire is responded to by assigning a scale value from 1 to 7, where 1 indicates strong disagreement and 7 indicates strong agreement.

A. The results from the SUS survey

The number of survey respondents was eight. This gives us a small sample size, and thus the results are seen as useful indications rather than definite results.

TABLE 1 SUS SCORE FOR AMAZING CITY GAME

ID	Question	Avr	Score
1	I think that I would like to use this system frequently	2.63	1.63
2	I found the system unnecessarily complex	2.13	2.88
3	I thought the system was easy to use	3.88	2.88
4	I think that I would need support of a technical person to be able to use this system	2.13	2.88
5	I found the various functions in this system were well integrated	3.38	2.38
6	I thought there was too much inconsistency in this system	2.13	2.88
7	I would imagine that most people would learn to use this system very quickly	3.88	2.88
8	I found the system very cumbersome to use	2.00	3.00
9	I felt very confident using the system	3.75	2.75

10	I needed to learn a lot of things before I could get going with this system	1.75	3.25
--	SUS score		68.44

Six of the respondents were students from computer science. All of the respondents therefore have a high technical competence. The SUS score for our game was 68.44, which is a bit below the mean score of 70.14 taken from 2324 surveys of other systems [9]. For a game, this score is a bit low meaning that the user-interface of the game was a bit difficult to use. In the debrief of the participants several challenging areas of the usability were identified. First of all, the participants were not sure about the overall goal of the game through the introduction and how the game should be used. Further, the users had to switch between several applications in order to solve the challenges (QR bar code scanner, Googles, Layer, Shazam, and ShopSavvy). The ACG application was open-ended and it was left very much in the hand of the user how it should be used. This made it a bit difficult for the players what to do next. An identified improvement would have been to integrate all the needed extra applications into ACG to avoid switching between applications. We also noticed that users with prior Android experience had far less usability problems compared to those unknown to Android. Our SUS score suffers also from users that both had to learn the application as well as Android.

B. EGameflow survey

Table 2 shows a comparison of the ACG EGameFlow results compared to four other games found in [8].

TABLE 2 EGAMEFLOW GAMES VS. AMAZING CITY GAME

Category	Game1	Game2	Game3	Game4	ACG
Concentration	5.118	5.225	5.214	5.153	5.22
Goal Clarity	4.180	5.360	5.048	5.306	5.03
Feedback	4.890	4.950	5.230	5.149	6.22
Challenge	4.654	4.880	5.019	4.764	4.22
Autonomy	4.686	4.880	5.019	4.764	4.38
Immersion	4.686	4.378	4.651	4.265	5.44
Social Interaction	3.163	3.250	3.365	2.826	5.38
Knowledge Improvement	4.985	5.420	5.171	5.055	5.21

Table 3 shows detailed feedback for each area:

TABLE 3 EGAMEFLOW SCALE FOR AMAZING CITY GAME

Concentration	Mean
Most of the gaming activities are related to the learning task	5.13
Generally speaking, I can remain concentrated in the game	5
I am not distracted from tasks that the player should concentrate on	5.13
Workload in the game is adequate	5.63
Average	5.22
Goal Clarity	Mean
Overall game goals were presented in the beginning of the game	4.13
Overall game goals were presented clearly	4.63
Intermediate goals were presented in the beginning of each scene	5.75

Intermediate goals were presented clearly	5.63
Average	5.03
Feedback	Mean
I receive feedback on my progress in the game	5.75
I receive immediate feedback on my actions	6
I am notified of new tasks immediately	6.63
I receive information on my success (or failure) of intermediate goals immediately	6.5
Average	6.22
Challenge:	Mean
The game provides "hints" in text that help me overcome the challenges	5.38
The game provides video or audio auxiliaries that help me overcome the challenges	4.13
The game provides new challenge with an appropriate pacing	5
The game provides different levels of challenges that is tailored to different players	2.38
Average	4.22
Autonomy:	Mean
I feel a sense of control and impact over the game	4.25
I know the next step in the game	4.5
Average	4.38
Immersion	Mean
I forget about time passing while playing the game	5.88
I become unaware of my surroundings while playing the game	4.63
I temporarily forget worries about everyday life while playing the game	5.13
I experience an altered sense of time	5.25
I can become involved in the game	5.88
I feel emotionally involved in the game	5.88
Average	5.44
Social Interaction	Mean
I feel cooperative toward other classmates	5.38
I strongly collaborate with other classmates	5.13
The cooperation in the game is helpful to the learning	5.63
Average	5.38
Knowledge Improvement	Mean
The game increases my knowledge	5.5
I catch the basic ideas of the knowledge taught	5.38
I want to know more about the knowledge taught	4.75
Average	5.21

Basically, from the survey, we found this pervasive educational game have high quality in feedback, immersion, social interaction since their average score is much higher than the other games' score shown in Table 2. It indicates that advantage to implant pervasive elements into an educational game. Although we can not make it as a general conclusion due to the limitation of total amount of participants, our results shows that the idea of using pervasive game in a learning context is an interesting concept that should be explored.

For the *concentration* and *knowledge improvement*, the score is similar to the other games' score shown in Table 2. But if we look further in Table 3 for knowledge improvement

area, we get high marks on first two items: *game increases participant knowledge* and let them *catch the basic knowledge*. For the third item, it seems that this *game's motivation* is not as strong as we thought.

The rest of Goal clarity, *challenge* and *autonomy* are a bit lower score that the other games' score shown in Table 2. For the Goal clarity, we found intermediate goals are clear in Table 3, but *overall goal* is not clearly present. We thought that lack of detailed instruction in the beginning of game maybe the reason. For the Challenge area, two groups meet troubles in the *video and auxiliaries* maybe cause a low score in second item. For fourth item, it reminds us that more *resources and plots* should be input to create challenges to match different levels. For Autonomy area, it indicates that autonomy of the game could be improved to let participant to feel freer to control the game. For the second item, the participants are not supposed to know the next step of the game until they have arrived at it. Although, this item has a low score, it is exactly a positive feedback from our aspect.

V. CONCLUSION

From our experiences we acknowledge that pervasive games for learning purposes need more exploration. Our study shows that pervasive educational games could be an informal learning environment and could be an interesting supplement to the formal and traditional education.

ACKNOWLEDGMENT

We thank Runar Os Mathisen, Lawrence Alexander Valtola, Sondre Wigmostad Bjerkaug and Trygve Bragstad for providing content and implementing the prototype of the amazing city game.

REFERENCE

- [1] H. Guo, *et al.*, "TeMPS: A Conceptual Framework for Pervasive and Social Games," in *Third IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL)*, 2010, pp. 31-37.
- [2] M. Prensky, "Digital game-based learning," *Computers in entertainment*, vol. 1, pp. 21- 24, 2003.
- [3] T. H. Laine, *et al.*, "Viable and Portable Architecture for Pervasive Learning Spaces," in *9th International Conference on Mobile and Ubiquitous Multimedia*, Limassol, Cyprus., 2010.
- [4] P. W. Jordan, *et al.*, *Usability Evaluation in Industry, chapter SUS - A quick and dirty usability scale*: CRC Press, 1996.
- [5] B. Wu, *et al.*, "IMPROVEMENT OF A LECTURE GAME CONCEPT- Implementing Lecture Quiz 2.0," in *Proceedings of the 3rd International Conference on Computer Supported Education*, 2011, pp. 26-35.
- [6] A. I. Wang, "An Evaluation of a Mobile Game Concept for Lectures," presented at the IEEE 21st Conference on Software Engineering Education and Training, 2008.
- [7] B. Wu, *et al.*, "XQUEST used in software architecture education," in *International IEEE Consumer Electronics Society's Games Innovations Conference (ICE-GIC 2009)*, 2009, pp. 70-77.
- [8] F.-L. Fu, *et al.*, "EGameFlow: A scale to measure learners' enjoyment of e-learning games," *Computers & Education*, vol. 52, pp. 101-112, 2009.
- [9] A. Bangor, *et al.*, "An Empirical Evaluation of the System Usability Scale," *International Journal of Human-Computer Interaction*, vol. 24, pp. 574-594, 2008.

Paper 5:

GDF1: Alf Inge Wang, Bian Wu, "An Application of a Game Development Framework in Higher Education", International Journal of Computer Games Technology, Special Issue on Game Technology for Training and Education, Volume 2009. ISSN: 1687-7047 EISSN: 1687-7055. DOI=10.1155/2009/693267



An Application of Game Development Framework in Higher Education

Alf Inge Wang and Bian Wu

Dept. of Computer and Information Science
Norwegian University of Science and Technology
alfw/bian@idi.ntnu.no

ABSTRACT

This paper describes how a game development framework was used as a learning aid in a software engineering course. Games can be used within higher education in various ways to promote student participation, enable variation in how lectures are taught, and improve student interest. In this paper, we describe a case study at the Norwegian University of Science and Technology (NTNU) where a game development framework was applied to make students learn software architecture by developing a computer game. We provide a model for how game development frameworks can be integrated with a software engineering or computer science course. We describe important requirements to consider when choosing a game development framework for a course, and an evaluation of four such frameworks based on these requirements. Further, we describe some extensions we made to the existing game development framework to let the students focus more on software architectural issues than the technical implementation issues. Finally, we describe a case study of how a game development framework was integrated in a software architecture course, and the experiences from doing so.

KEY WORDS

Game development framework, Software architecture, Software engineering education

1 Introduction

Games have been used in schools for many years to help children learn skills in math, language, geography, science and other domains in an interesting and motivating way. Research shows that integrating games within a classroom with children can be beneficial for academic achievement, motivation and classroom dynamics [24]. There is also evidence that the teaching methods based on educational games are not only attractive to schoolchildren, but also to university students [19]. There have been conducted research on games concept and game development used in higher education before, e.g. [3, 16, 11], but we believe there is an untapped potential that needs to be explored. Games can provide teachers in higher education teaching aids that can promote more active students, provide alternative teaching methods to improve variation, and enable social learning through multiplayer learning games.

Games can be integrated in higher education in three ways. *First*, games can be used instead of traditional exercises motivating students to put extra effort in doing the exercises, and giving the teacher and/or teaching assistants an opportunity to monitor how the students work with the exercises in real-time [29, 30]. *Second*, games can be used within lectures to improve the participation and motivation of students [1, 31]. In this approach, the students and the teacher participate in knowledge-based games. *Third*, the students are required to develop a game as a part of a course using a game development framework (GDF) to learn skills within computer science or software engineering [32]. This paper focuses on the latter, where game development and a GDF is used in student projects to learn software engineering skills, extending the use of games as a teaching aid in higher education. The motivation of making students develop games to learn software engineering is to bring the students' enthusiasm from playing games to learn to courses through game development. In addition, we wanted to investigate if the specific features of a GDF are suitable for teaching software engineering, and how game development can be integrated with the education process. More specifically, we wanted to explore how the use of game development and the GDF would affect the learning of software architecture with focus on the technical aspects of the GDF.

This paper focuses on how the technical aspects of a GDF affect the learning of software architecture, the selection of appropriate GDF for a software architecture course, and how a GDF can be applied in a software engineering course. The main contribution of this paper is a presentation of a novel GDF concept that can be used in courses that includes software development, experiences from actual usage of the GDF, and some course design considerations.

The rest of the paper is organized as follows. Section 2 describes and motivates for how a GDF can be used in higher education and what criteria should be considering when choosing one. Section 3 describes a case study of applying a GDF in a software architecture course. Section 4 describes experiences from using a GDF in a software course. Section 5 describes similar approaches, and Section 6 concludes the paper.

2 Game Development Frameworks in Higher Education

This section presents the motivation for applying GDFs in higher education, a model for how GDFs can be integrated with a course, and requirements for how to choose the appropriate GDF for educational purposes.

2.1 GDF and Education

The main motivation for introducing GDF in software engineering (SE) or computer science (CS) courses is to motivate students to put more effort into software development project in order to improve software development skills. Game development offers an interesting way of learning and applying the course theory. By introducing a game development project in a course, the students have to establish and describe most of the functional requirements themselves (what the game should be like). This can be a motivating factor especially for group-based projects, as each group will develop a unique application (the game), it will encourage creativity, and it will require different skills from the group members (art, programming, story, audio/music). The result will be that the students will have a stronger feeling of ownership to the project. Furthermore, students also could learn about game development technology. The main disadvantages by introducing a game development project and a GDF into a SE or CS course is that the student might spend too much time on game-specific issues and that the project results might be difficult to compare. It is critical that the students get motivated applying a GDF in a course, and that they get increased motivation for learning and applying course theory through a game development project.

Tom Malone has listed three main characteristics that make things fun to learn: they should provide the appropriate level of challenge, they should use fantasy and abstractions to make it more interesting, and they should trigger the player's curiosity [26]. These characteristics can directly be applied when developing a game for learning purposes. However, we can also consider these characteristics when introducing a GDF in a SE or CS course. By allowing the students to develop their own games using a GDF, such projects are likely to trigger students' curiosity as well as provide a challenge for students to design fun games with their knowledge, skills, imagination and creativity. The level of the challenge can be adjusted according to the project requirements given in courses by the teacher. Thus, the challenge level can not only be adjusted to the right level for most participants, but also tailored for individual differences. As the students will work in groups, group members helping other group members can compensate for the individual differences. An open platform and agile courses requirements should be provided for students to design their own games, combined with their ability, fantasy and comprehension of lecture content.

The main benefit of using a GDF as a teaching aid is that it can be a motivating initiative in courses to learn about various topics such as software requirements, software design, software

architecture, programming, 2D and 3D graphic representation, graphic programming, artificial intelligence, physics, animation, user interfaces, and many other areas within computer science and software engineering. It is most useful for learning new skills and methods within a specific domain but also useful for testing and rehearsing theory by applying know skills and knowledge in a project using a GDF.

2.2 Circulatory Model of Applying a GDF in a Course

There are several good reasons for introducing a GDF and game development projects in CS and SE courses as described in previous section, but in order to make it a success it is important that the GDF is well integrated with the course. Based on our experiences, we have developed a circular model for how to apply a GDF in a CS or SE course through six steps (see Figure 1). The model is intended for courses where a software development project is a major part of the course.

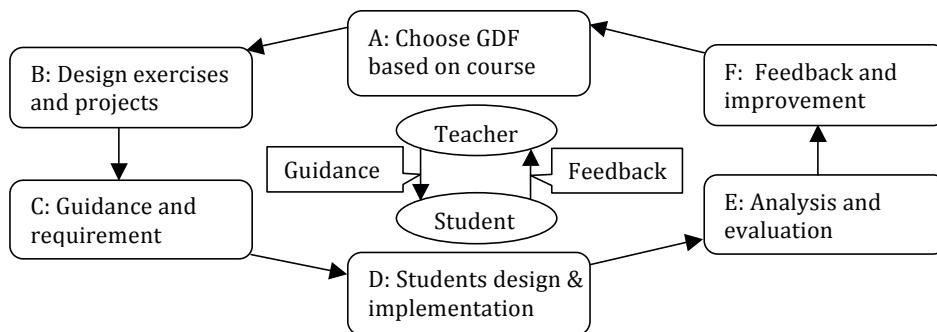


Figure 1. Circulatory model of GDF's application in courses

To choose one appropriate development platform according to the course content, it is important to consider the process of the course related to the development project. This process starts with choosing an appropriate GDF (step A) for the course related to some requirements (described in the next section). Next, the design of exercises and projects (step B) must reflect the limitations and constraints of the chosen GDF. In the initial phase of the student project, it is important that the students get the required technical guidance and appropriate requirements (step C) related to the GDF. It is important that the students get to know the GDF early, e.g. by introducing an exercise to implement a simple game in the GDF. It is critical that there is sufficient course staff that knows the GDF well enough to give the required feedback. The next step is for the students to start designing and implementing (step D) their own game according to the constraints within the course and the GDF. After the students have delivered their final version of their project implementation and documentation, the students should get the chance to evaluate and analyse (step E) their own projects to learn from their successes and mistakes. This information should then be used to provide feedback in order to improve the course (step F). The feedback from the students might indicate that another GDF should be used or that the course constraints on the projects should be altered. The core of this model is that the teacher should encourage the students to explore the course theory through a game development project using a GDF, and give the opportunity to improve the game development project through feedback from the students.

2.3 Criteria for choosing the right GDF

How to choose an appropriate GDF that easily can be integrated with course content should be based on the educational goals of the course, the technical level and skills of students, and the time available for projects and/or exercises. Based on experiences from using GDFs and from student projects in CS and SE courses, we have come up with the following requirements for choosing a GDF for a CS or SE course:

- 1) *It must be easy learn and allow rapid development.* According to Malone's recommendation of how to make things fun to learn, it is crucial that we provide the appropriate level of challenge. If the GDF is too much of a challenge and requires too much to learn before becoming productive, the whole idea of game development will be wasted as the student will lose motivation. An important aspect of this is that the GDF offers high-level APIs that makes it possible for the students to develop impressive results without writing too many lines of code. This is especially critical in the first phase of the project.
- 2) *It must provide an open development environment to attract students' curiosity.* Malone claims that fantasy and curiosity are other important factors that make things fun to learn. By providing a relatively open GDF without too many restrictions on what you can produce, the students get a chance to realize the game of their dreams. This means that the GDF itself should not restrict what kind of game the students can make. This requirement would typically rule out GDFs that are tailored for producing only one game genre such as adventure games, platform games or board games. In addition, ideally an open development environment should offer public and practical interfaces for developers to extend their own functions. In this respect, open source game development platforms are preferred.
- 3) *It must support programming languages that are familiar to the students.* The students should not be burdened to have to learn a new programming language from scratch in addition to the course content. This would take away the focus of the educational goals of the course. We suggest to choose GDFs that support popular programming languages that the students know like C++, C# or Java. It is also important that the programming languages supported by the GDF have high-level constructs and libraries that enable the programmers to be more productive as less code is required to produce fully functional systems. From an educational point of view, programming languages like Java and C# are better suited than C and C++, as they have more constraints that force the programmers to write cleaner code and there is less concern related to issues like pointers and memory leakage. From a game development perspective, programming languages like C and C++ are more attractive as they generally produce faster executables and thus faster games.
- 4) *It must not conflict with the educational goals of the course.* When choosing a GDF it is important that the inherent patterns, procedures, design and architecture of the GDF are not in conflict with the theory taught in the course. One example of such a conflict could be that the way the GDF enforces event handling in an application is given as an example of bad design in the textbook.
- 5) *It must have a stable implementation.* When a GDF is used in a course, it is essential that the GDF has few bugs so the students do not have to fight a lot of technical issues instead of focusing on the course topics. This requirement indicates that it is important that the GDF is supported by a company or a development community that have enough resources to eliminate serious technical insufficiencies. It is also important that the development of the GDF is not a dead project, as this will lead to compatibility issues for future releases of operating systems, software components and hardware drivers.
- 6) *It must have sufficient documentation.* This requirement is important both for the course staff and the students. The documentation should both give a good overview of the GDF as well as document all the features provided. Further, it is important that the GDF provides tutorials and examples to demonstrate how to use the GDF and its features. The frameworks should provide documentation and tutorials of high quality enabling self-study.
- 7) *It should be inexpensive (low costs) to use and acquire.* Ideally, the GDFs should be free or have very low associated cost to avoid extra costs running the course. This requirement also involves investigating additional costs related to the GDF such as requirements for extra or more powerful hardware, and/or requirements for additional software.

The goal of the requirements above is to save the time and effort the students have to spend on coding and understanding the framework, making them concentrate on the course content and software design. Thus, an appropriate GDF could provide the students exciting experiences and offer a new way of learning through a new domain (games). The requirements above are also important for the course staff, as they will help to find a GDF that would cause less effort spent on technical issues, and incompatibility between GDF and the course contents.

From the requirements above, we acknowledge that there is a conflict between requirement one and two. The level of the freedom the developer is given to make whatever game he likes could be in conflict with providing a development environment that allows rapid development and is easy to learn. A more open GDF usually means that the developer must learn more APIs as well as the APIs themselves usually are lower level, and thus harder to use. However, it is possible to get a bit of both worlds by offering high-level APIs that are relatively easy to use, but still allow the developer to access underlying APIs that gives the developer the freedom in what kind of games that can be made. This means that the GDF can allow inexperienced developers to just modify simple APIs or example code to make variants of existing games, or to allow more experienced developers make unique games by using more of the provided underlying APIs. How hard the GDF is to use will then really depend on the ambition of the game developer and not on the GDF itself. This can also be a motivating factor to learn more about the GDF's APIs.

3 Case Study: Applying a GDF in a Software Architecture Course

This section describes a case study of a software architecture course at the Norwegian University of Science and Technology (NTNU) where a GDF was introduced.

3.1 The Software Architecture Course

The software architecture course is a post-graduate course offered to CS and SE students at NTNU. The course is taught every spring, its workload is 25% of one semester, and about 70 postgraduate students attend the course every semester. The students in the course are mostly of Norwegian students (about 80%), but there are about 20% foreign students mostly from EU-countries. The textbook used in this course is the “Software Architecture in Practice, Second Edition”, by Bass, Clements and Kazman [23]. Additional papers are used to cover topics that are not sufficiently covered by the book such as design patterns, software architecture documentation standards, view models, and post-mortem analysis [2, 8, 6, 22]. The education goal of the course is:

“The students should be able to *define and explain* central concepts in software architecture literature and be able to *use and describe* design/architectural patterns, methods to design software architectures, methods/techniques to achieve software qualities, methods to document software architecture, and methods to evaluate software architecture.”

The course is taught in four main ways:

- 1) Ordinary lectures given in English
- 2) Invited guest lectures from the software industry
- 3) Exercise in design patterns
- 4) A software development project with emphasis on software architecture

30% of the grade is based on an evaluation a software architecture project that all students have to do, while 70% is given from the results of a written examination. The goal of the project is for the students to apply the methods and theory in the course to design a software architecture and to implement a system according to the architecture. The project consists of the following phases:

- 1) COTS (Commercial Off-The-Shelf) exercise: Learn the development platform to be used in the project by developing some simple test applications.
- 2) Design pattern: Learn how to utilize design pattern by making changes in an existing system designed with and without design patterns.
- 3) Requirements and architecture: Describe the functional and the quality requirements, and design the software architecture for the application in the project.
- 4) Architecture evaluation: Use the Architecture Trade-off Analysis Method (ATAM) [23, 36] to evaluate the software architecture in regards to the quality requirements. Here one student group will evaluate another student group's project.
- 5) Implementation: Do detailed design and implement the application based on the created architecture and based on the results from previous phase.
- 6) Project evaluation: Evaluate the project after it has been completed using a Post-Mortem Analysis (PMA) method.

In the two first phases of the project, the students work on their own or in pairs. For the phases 4-6, the students work in self-composed groups of four students. The students spend most time on the implementation phase (6 weeks), and they are also encouraged start the implementation in earlier phases to test their architectural choices (incremental development). In previous years, the goal of the project has been to develop a robot controller for a robot simulator in Java with emphasis on an assigned quality attribute such as availability, performance, modifiability or testability.

3.2 Choosing a GDF for the Software Architecture Course

Fall 2007, we started to look for appropriate GDFs to be used in the software architecture course spring 2008. We looked both for GDFs where the programmer had to write the source code as well as visual drag-and-drop programming environments. The selection of candidates was based on GDFs we were familiar with and GDFs that had developer support. Further, we wanted to compare both commercial and open source GDFs. From an initial long list candidate GDFs, we chose to evaluate the following GDFs more in detail:

- **XNA:** XNA is a GDF from Microsoft that enables development of homebrew cross-platform games for Windows and the XBOX 360 using the C# programming language. The initial version of Microsoft XNA Game Studio was released in 2006 [18], and in 2008 Microsoft XNA Game studio 3.0 was released that includes support for making games for XBOX Live. XNA features a set of high-level API enabling the development of advanced games in 2D or 3D with advanced graphical effects with little effort. The XNA platform is free, and allows developers to create games for the Windows, Xbox 360 and Zune using the same GDF [20]. XNA consists of an integrated development environment (IDE) along with several tools for managing audio and graphics.
- **JGame:** JGame is a high-level framework for developing 2D games in Java [33]. JGame is an open source project and enables developers to develop games fast using few lines of code as JGame will take care of typical game functionality such as sprite-handling, collision detection, and tile handling. JGame games can be run as stand-alone Java-games, Java applets games running in a web-browser or on mobile devices (Java ME). JGame does not provide a separate IDE, but is integrated with Eclipse.
- **Flash:** Flash is a high-level framework for interactive applications including games developed by Adobe [34]. Most programming in Flash is carried out in Action script (a textual programming language), but the Flash environment also provides a

powerful graphical editor for managing graphical objects and animation. Flash applications can run as stand-alone applications or in a web-browser. Flash applications can run on many different operating systems like Windows, Mac OS X and Linux as well as on mobile devices and game consoles (Nintendo Wii and Sony Playstation 3). Programming in Flash is partly visual by manipulating graphical objects, but most code is written textually. Flash supports development of both 2D and 3D applications.

- **Scratch:** Is a visual programming environment developed by MIT Media Lab in collaboration with UCLA that makes it easy to create interactive stories, animations, games, music and art – and share the creations on the web [17]. Scratch works similar to Alice [5] allowing you to program by placing sprites or objects on a screen and manipulate them by drag-and-drop programming. The main difference between Scratch and Alice is that Scratch is in 2D while Alice is in 3D. Scratch provides its own graphical IDE that includes a set of programming primitives and functionality to import various multimedia objects.

An evaluation of the four GDF candidates is shown in Table 1. From the four candidates, we found Scratch to be the least appropriate candidate. The main disadvantage with Scratch was that it would be very difficult to teach software architecture using this GDF as the framework did not allow exploring various software architectures. Further, Scratch was also very limited in what kind of games that could be produced, limiting the options for the students. The main advantage using Scratch is that it is very easy to learn and use. JGame suffered also from some of the same limitations as Scratch, as it put some restrictions on what software architecture that could be used and it had little flexibility in producing a variety of types of games. The main advantage using JGame was that it was an open source project with access to the source code and that all the programming was done in Java. All CS and SE students at NTNU learn Java in the two first introductory programming courses. An attractive alternative would be to use Flash as a GDF. Many developers use Flash to create games for kids as well as games for the Web. Flash puts little restrictions on what kind of games you can develop (both 2D and 3D), but there are some restrictions on what kind of software architecture that you can use in your applications. The programming language used in Flash, Action Script, is not very different from Java so it should be rather easy for the students to learn. The main disadvantage using Flash in the software architecture course was the licence costs. As the computer and information science department does not have a site licence for the Flash development kit, it would be too expensive to use. XNA was found an attractive alternative for the students, as it made it possible for them to create their own XBOX 360 games. XNA puts little restrictions on what kinds of software architectures you apply in your software, and it enables the developers to create almost any game. XNA has strong support from its developer (Microsoft) and has a strong community of developers along with a lot of resources (graphics, examples, etc). The main disadvantages using XNA as a GDF in the course were that the students had to learn C# and that the software could only run on Windows machines. Compared to JGame and other Java-based GDFs, XNA has a richer set of high-level APIs and a more mature architecture.

Table 1 Evaluation four GDF candidates

Selection requirement	XNA	JGame	Flash	Scratch
1 Easy to learn	Relatively easy to learn, but requires to learn several core concepts to utilize the offered possibilities.	Easy to learn, but requires to learn a small set of core concepts.	Relatively easy to learn, but requires to learn several core concepts to utilize the offered possibilities.	Very easy and intuitive to learn and supports dynamic changes to the game in run-time.
2 Open development environment	XNA puts little restrictions on what kind of games that can be developed and supports development	JGame supports a limited set of games mainly classical 2D arcade games. Open source project.	Flash puts little restrictions on what kind of games that can be developed and supports development of both 2D	Scratch limits the options of what kind of games the user can make through the limited options provided in the graphical programming

	of both 2D and 3D games. Not open source project.		and 3D. Not open source project.	environment. Not open source project.
3 Familiar programming language	All programming is done in C#.	All programming is done in Java	Some programming can be done using drag-and-drop, but most will be written in Action Scripts.	All programming is done in the visual drag-and-drop programming language Scratch.
4 Not in conflict with educational goals	XNA puts <i>little restrictions</i> on what kinds of software architectures that can be used.	JGame puts <i>some restrictions</i> on what kinds of software architecture that can be used.	Flash puts <i>some restrictions</i> on what kinds of software architectures that can be used.	Scratch puts <i>strict restrictions</i> on what kinds of software architectures that can be used.
5 Stable implementation	XNA has a very stable implementation and is updated regularly.	JGame has a relatively stable implementation and is updated regularly.	Flash has a very stable implementation and is updated regularly.	Scratch has a relatively stable implementation and is updated regularly.
6 Sufficient documentation	XNA is well documented and offers several tutorials and examples. Many books on XNA are available.	JGame is not well documented, but some examples exist.	Flash is well documented and offers several tutorials and examples. Many books on Flash are available.	Scratch is ok documented and has some examples and tutorials available.
7 Low costs	XNA is free to use. A \$99 for a year of membership is required to develop games for XBOX 360.	JGame is free to use.	The Flash development kit costs \$199 per licence (university licence).	Scratch is free to use.

Based on the evaluation described above, we chose XNA as a GDF for our course. From previous experience we knew that it does not require much effort and time to learn C# for students that already know Java.

3.3 XQUEST – An Extension of the Chosen GDF

After we had decided to use XNA as a GDF in the software architecture course, we launched a project to extend XNA to make XNA even easier to use in the student project. This project implemented XQUEST (XNA QUick & Easy Starter Template) [27], which is a small and lightweight 2D game library/game template developed at NTNU that contains convenient game components, helper classes, and other classes that can be used in the XNA game projects (see Figure 2). The goal of XQUEST was to identify and abstract common game programming tasks, and create a set of components that could be used by students of the course to make their life easier. We choose to focus only on 2D. There are a few reasons for this. *First*, the focus of the student projects is software architecture, not making a game with fancy 3D graphics. *Second*, students unfamiliar with game programming and 3D programming may find it daunting to have to learn the concepts needed for doing full-blown 3D in XNA, such as shader programming and 3D-modelling, in addition to software architectures. To keep the projects in 2D may reduce the effect of students focus only on the game development and not on the software architecture issues.

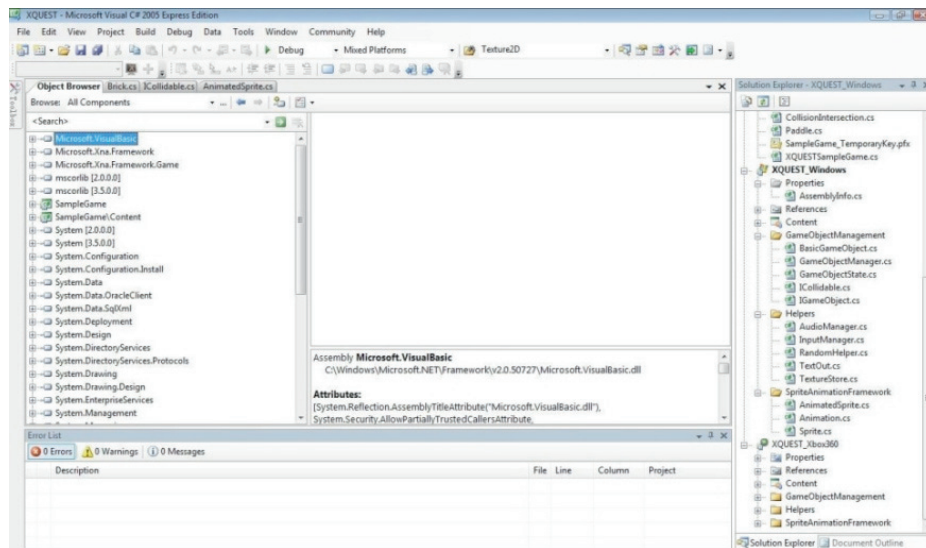


Figure 2. The XQUEST library shown in the XNA development environment

3.4 Teaching Software Architecture using XNA

XNA was introduced in the software architecture course to motivate students to put extra effort in the student project with the goal to learn the course content such as attribute driven design, design and architectural patterns, ATAM, design of software architecture, view points and implementation of software architecture. This section will go through the different phases of this project and describe how XNA affected these phases.

3.4.1 Introduction of XNA Exercises

In the start of the semester the course staff gave an introduction to course where the software architecture project was presented. Before the students started with their project, they had to do an exercise individually or in pairs where they got to choose their own partner. The goal of the first exercise was to get familiar with the XNA framework and environment, and the students were asked to complete four tasks:

- 1) Draw a helicopter sprite on the screen and make it move around on its own.
- 2) Move around the helicopter sprite from previous task using the keyboard, change the size of the sprite when a key was pressed, rotate the sprite when another key was pressed and write the position of the sprite on the screen.
- 3) Animate the helicopter sprite using several frames and do sprite collision with other sprites.
- 4) Create the classical Pong game in XNA.

Before the students started on their XNA introduction exercise, they got a two-hour technical introduction to XNA. During the semester, two technical assistants were assigned to help students with issues related to XNA. These assistants had scheduled two hours per week to help students with problems, in addition to answer emails about XNA issues.

3.4.2 Requirement and Architecture for the Game Project

After the introduction exercise was delivered, the students formed groups of four students. Students that did not know anyone, were assigned to groups. The course staff then issued the project task where the goal was to make a functioning game using XNA based on students' own defined game concept. However, the game had to be designed and implemented according to their specified and designed software architecture. Further, the students had to

develop a software architecture that focused on one particular quality attribute assigned by the course staff. We used the following definitions for the quality attributes in the game projects: *Modifiability*, the game architecture and implementation should be easy to change in order to add or modify functionality; and *Testability*, the game architecture and implementation should be easy to test in order to detect possible faults and failures. These two quality attributes were related to the course content and the textbook. A perfect implementation was not the ultimate quest of this XNA game project, but it was critical that the implementation reflected the architectural description. It was also important that the final delivery was well-structured, easy to read, and made according to the template provided by the course staff.

The first phase of the project was the requirement and architecture phase where the students should delivery requirements and the software architecture of the game along with a skeleton code reflecting the architecture. The requirements document focused on a complete functional requirement description of the game and several quality requirements for the game described as scenario focusing on one particular quality attribute. The architectural description was the most important part of the final delivery of for the game project, and the students had to document their architecture according to IEEE 1471-2000[14]. The architecture documentation could be altered several times before its final delivery. Table 2 lists main attributes required in the architectural description in the game projects.

Table 2 List of architecture description for the game project

#	Architectural Description Attributes	Details of the Implementation
1	Architectural Drivers	The main drivers that affect the system mostly, including the attribute on which the students focus.
2	Stakeholders and Concerns	Stakeholders of the system, and their concerns.
3	Selection of Architectural Viewpoint	A list of the viewpoints used, their purpose, target audience and from of description. Places to look for possible viewpoints include the book [23], and the 4+1 article by Kruchten [15].
4	Quality Tactics	Including all attributes and more detailed for the focused ones.
5	Architectural Patterns	The major patterns of your architecture, both architectural and major design ones.
6	Views	A separate section for each required views: logic, process and development views or other views added by students.
7	Consistency Among Views	Discuss the consistency between each described view.
8	Architectural Rationale	In this section and sub-sections, add why things are chosen.

We also required that the students wrote the code skeleton for the architecture they had designed. This was done to emphasize the importance of starting the implementation early, and to ensure that students designed an architecture that was possible to implement.

3.4.3 Evaluation of the Game Project

After the requirements, the architecture and the code skeleton were delivered, the student groups were assigned to evaluate each other's architecture using ATAM. The whole idea was for one project group to evaluate the architecture of the other group's game to give feedback on the architecture related to the quality focus of the software architecture [37]. It included attribute utility tree, analysis of architectural approach, sensitivity points, trade-off points, risks and non-risks, and risk themes.

3.4.4 Detailed Design and Implementation

The focus of implementation phase was to design, implement and test the game application. The documentation delivered in this phase focused on the test results from running the game related to the specified requirements, and the discussion of the relationship between the implemented game and the architectural documentation [8, 6]. Table 3 lists what should be delivered in the implementation phase:

Table 3 Design & Implementation phase description

#	Implementation	Details of Implementation
---	----------------	---------------------------

	Deliverables	
1	Design and Implementation	A more detailed view of the various parts of the architecture describing of game design.
2	User's Manual	To guide the users the steps to compile and run the game.
3	Test report	Contain both functional requirements and quality requirements (quality scenarios).
4	Relationship with the architecture	List the inconsistencies between the game architecture and the implementation and the reasons for these inconsistencies.
5	Problems, Issues and Points learned	Listing problems and issues with the document or with the implementation process.

For the test report part in the Table 3, the functional requirements and quality requirements had the attributes like shown in List 1, 2. The test reports should also include a discussion about the observation of the test unless there was nothing to discuss about the test results.

F1: The role in game should be able to jump along happily	
Executor:	Super Mario III
Date:	23.3.2005
Time used:	5min
Evaluation:	Fail: White role cannot jump!

List 1 Attributes of functional requirements

A1: The role in game should not get stuck	
Executor:	Snurre Sprett
Date:	24.3.2005
Stimuli:	The role should be able to move around for 10 min
Expected response:	Success in 8 of 10 executions
Observed response:	Success in 3 of 10 executions
Evaluation:	Fail

List 2 Attributes of quality requirements

At the end of this phase, the students had to submit their final delivery of their projects that included all documents, code and other material from all project phases. The course staff evaluated all the groups' deliveries and gave grades by judging document and implementation quality, document and implementation completeness, architecture design, and readability and structure of code and report.

3.4.5 The Game Project Workshop

In this workshop, selected groups had to give short presentations about the project goal, quality attribute focus, proposed architectural solution with some diagrams or explanations, and an evaluation of how well did the solution worked related to functional requirements and quality focus. Further, the selected groups ran demos of their games and it was opened for questions from the audience.

The workshop provided an open mind environment to let students give each other feedback, brainstorm about improvements and ideas, and to discuss their ideas to give a better understanding of the course content and game architecture design.

3.4.6 Post-Mortem Analysis

In the final task in the project, every group had to perform a post-mortem analysis of their project. The focus of the PMA was to analyse successes and problems of the project. The PMA was documented in a short report that included a positive (successes) and a negative (problems) KJ-diagram (structured brainstorm map); a positive and a negative causal map (a diagram that shows cause-effect relationships), and experiences from using PMA [2]. The PMA made the students reflect on their performance in the project and gave them useful feedback to improve in future projects and inputs for the course staff to improve the course. The main topics analysed in the PMA were issues related to group dynamics, time management, technical issues, software architecture issues, project constraints, and personal conflicts.

4 Experiences of using GDF in Software Architecture

The experiences described in this section are based on the final course evaluation, feedback from the students during the project, and the project reports.

The final course evaluation made all students (mandatory) taking the course answer three questions. The results reported below are a summary of the students' responses related to the project and the GDF.

1) What have been good about software architecture course?

- **About the project itself:** "Cool project", "Really interesting project", "We had a lot of fun during the project", "It is cool to make a game", "Fun to implement something practical such a game", "Videogame as an exercise is quite interesting", "I really liked the project", "The game was motivating and fun".
- **Project and learning:** "Good architectural discussion in the project group I was in", "Learned a lot about software architecture during the project", "The project helped to understand better the arguments explained in the lectures, having fun at the meantime", "Fun project where we learned a lot", "I think that the creation of a project from the beginning, with the documentation until the code implementation, was very helpful to better understand in practice the focus of the course", "The game project was tightly connected to the syllabus and lectures and gave valuable experience. The main thing I learned was probably how much simpler everything gets if you have a good architecture as a basis for your system", "The interplay of game and architectural approaches".
- **The project being practical work:** "I think it was pretty good that you guys made us do a lot of practical work", "To choose C# as a platform is a good idea as it is used a lot in the software industry, at the same time it is very similar to Java so it is rather easy to learn the language.
- **Interplay between groups:** "It was also good to see the results of the others' projects in the final presentation".

2) What have been not so good about the course software architecture?

- **XNA support:** "The way the student assistants were organized, during the implementation periods at least they should be available in a computer lab and not just in the classroom", "Maybe the use of XNA Framework XQUEST was very difficult because I never use it. Maybe some extra lecture focus on the use of XQUEST Framework was better", "We didn't have lectures on XNA, could have got some more basic info...Hmm..."
- **XNA vs. software architecture:** "Took a lot of time getting to know c#, I liked it, but I did not have the time to study architecture", "The use of game as a project may have removed some of the focus away from the architecture. XNA and games in general limits the range of useful architectures."

3) What would you have changed for next year's course?

- **Project workload:** "Maybe just little more time to develop the game", "I would change the importance of the project. I think that the workload of the project was very big end it can matter the 50% of the total exam."
- **XNA support:** "Perhaps have some c# intro?", "It would be helpful to have some lab hours".
- **Project constraints:** "Maybe more restrictions on game-type, to ensure that the groups choose games suited for architectural experimentation."

The responses from the students were overall very positive. In the previous years, the students in the software architecture course had to design the architecture and implement a robot controller for a robot simulator in Java. The feedback from the XNA project was much more positive than the feedback from the robot controller project. Other positive feedback we got from the students was that they felt they learned a lot from the game project, that they liked the practical approach of the project and having to learn C#, and the interaction between the groups (both ATAM and the project workshop).

The negative feedback from the course evaluation was focusing on lack of XNA support and technical support during the project, and that some student felt that there was too much focus on C#, XNA and games and too little on software architecture.

The suggestions to improve the course was mainly according to the negative feedback, namely to improve XNA support and to adjust the workload of the project. One student also suggested limiting the types of games to be implemented in project to ensure more focus on software architectural experimentation.

4.2 Snapshots from some Student Projects

Figure 3 shows screenshots from four student game projects. The game at upper left corner is a racing game, the game at the upper right corner is a platform game, and the two games below are role-playing games (RPGs). Some of the XNA games developed were original and interesting. Most games were entertaining, but were lacking contents and more than one level due to time constraints.

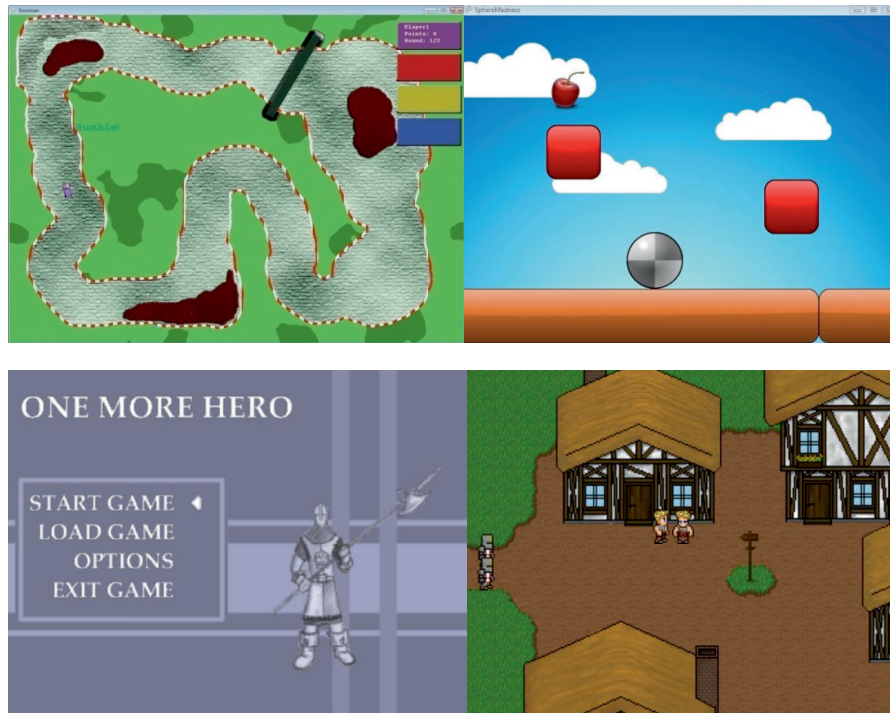


Figure 3. Game based on XNA framework
(Top left: Racing; Top right: Codename Gordon; Bottom: RPG)

5 Related Work

This paper describes experiences from utilizing the special features of a GDF in a software architecture course. The main benefits from applying a GDF in a CS or SE course is that the students get more motivated during the software development project. As far as we know, there are few papers that describe the usage of a professional GDF concept applied in universities courses that is not directly target for learning game development, especially no papers about usage of XNA in higher education. However, there are some related approaches in education described in this section.

El-Nasr and Smith describes how the use of modifying or *modding* existing games can be used to learn computer science, mathematics, physics and ascetic principles [32]. The paper describes how they used modding of the WarCraft III engine to teach high school students a class on game design and programming. Further, they describe experiences from teaching university students a more advanced class on game design and programming using the Unreal Tournament 2003 engine. Finally, they present observations from student projects that involve modding of game engines. Although the paper claims to teach students other things than pure game design and programming, the GDFs were used in the context of game development courses.

The framework Minueto [4] is implemented in Java and it is used by students in their second year of undergraduate studies at McGill University in Montreal, Canada. The framework encapsulates graphics, audio and keyboard/mouse inputs to simplify Java game development. It allows development of 2D games, such as card games and strategy games, but it lacks in support for visual programming and suffers from limited documentation.

The Labyrinth [9] is implemented in Java and it is a flexible and easy-to-use computer game framework. The framework enables instructors to expose students to very specific aspects of computer science courses. The framework is a finished game in the Pac-Man genre, highly modular, and it lets the students change different aspects of the game. However, it cannot be used to develop different genres of game and there is little room for changing the software architecture of the framework.

The JIG (Java Instructional Gaming) Project [28] is a collaborative effort between Scott Wallace (Washington State University Vancouver) and Andrew Nierman (University of Puget Sound) in conjunction with a small group of dedicated students. It has three aims: 1) to build a Java Instructional Game Engine suitable for a wide variety of students at all levels in the curriculum; 2) to create a set of educational resources to support the use of the game engine at small, resource-limited, schools; and 3) to develop a community of educators that use and help improve these resources. The JIG Project was proposed in 2006, after a survey of existing game engines revealed a very limited supply of existing 2D Java game engines. JIG is still in development.

GarageGames [12] offers two game engines written in C++. The Torque Game Engine targets 3D games, while the Game Builder provides a 2D API and encourages programmers to develop using a proprietary language (C++ can also be used). Both engines are aimed at a wide audience, including students and professionals. The engines are available under separate licenses (\$50 per licence per year for each engine) that allow full access to the source code. Documentation and tutorials cover topics appropriate for beginners and advanced users.

The University of Michigan's DXFramework [10] game engine is written in C++. The current version is targeted specifically for 2D games, although previous versions have included a 3D API as well. This engine is designed for game programming education and is in its third major iteration. The DXFramework is an open source project. Compare to XNA,

DXFramework has no competitive advantage as it has limited support for visual programming and it is not easier than XNA to learn.

The University of North Texas's SAGE [21] game engine is written in C++ and targets 3D games, not 2D. Like the DXFramework, SAGE is targeted specifically for game programming educational usage. The source code can be downloaded and is currently available without license.

Marist College's GEDI [7] game engine provides a second alternative for 2D game design in C++, and is also designed with game programming educational use in mind. Source code can be downloaded and is currently available without license, but GEDI is still in the early phases of development. Only one example game is distributed with the code, and little documentation is available.

For business teaching, Arena3D [25] is a game visualization framework with its animated 3D representations of the work environments, it simulates patients queuing at the front desk, and interacts with the staff. IBM has also produced a business game called INNOV8 [13] which is "an interactive, 3-D business simulator designed to teach the fundamentals of business process management and bridge the gap in understanding between business leaders and IT teams in an organization".

6 Conclusion and Future Work.

In this paper we have presented a case study of how a GDF was evaluated, chosen and integrated with a software architecture course. The main goal of introducing a GDF and a game development project in this course was to motivate students to learn more about software architecture during the game development project. The positive feedback from the students indicate that this was a good choice as the student really enjoyed the project and learn software architecture from carrying out the project.

We will continue to explore the area of using games, games concept and game development in CS and SE education and evaluate how this affects the students' motivation and performance. The choice of XNA as a GDF proved to be a good choice for our software architecture course. The main disadvantage using XNA is the lack of support for non-Windows operating systems like Linux and Mac OS X. Mono.XNA is a cross platform implementation of the XNA game framework that allows XNA to run on Windows, Mac OS X and Linux using OpenGL [35]. The project is still in an early phase. An alternative to solve this problem is to let the students choose between different GDFs, e.g., XNA and a Java-based GDF. The main challenge for this approach is the course staff needs to know all the GDFs offered to the students to give proper technical assistance. Based on the feedback from the students, the technical support is very important and must be considered before providing choices of more GDFs.

Acknowledgement

We would like to thank Jan-Erik Strøm and Trond Blomholm Kvamme for implementing XQUEST and for their inputs for this paper. We would also like to thank Richard Taylor and Institute for Software Research (ISR) at University of California, Irvine (UCI) for providing a stimulating research environment and for hosting a visiting researcher.

Reference

- [1] A. I. Wang, O. K. Mørch-Storstein, T. Øfsdahl, "Lecture quiz - a mobile game concept for lectures", The 11th IASTED International Conference on Software Engineering and Application (SEA 2007), November 19-21, 2007.
- [2] A. I. Wang, T. Stålhane, "Using Post Mortem Analysis to Evaluate Software Architecture Student Projects", In Proceedings of the 18th Conference on Software Engineering Education & Training, April 18 - 20, 2005.
- [3] A. Baker, E. O. Navarro, and A. Hoek, "Problems and Programmers: an Educational Software Engineering Card Game", In Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), pages 614–619, 2003.
- [4] A. Denault. Minueto, "An undergraduate teaching development framework", Master's thesis, School of Computer Science McGill University, 2005.
- [5] Carnegie Mellon University, "Alice.org", Web: <http://www.alice.org/>, Retrieved June 2008.
- [6] A. Rollings and D. Morris, "Game Architecture and Design - A New Edition", New Riders Games, pages 462-500, 2003. Web: <http://www.k-team.com>
- [7] R. Coleman, S. Roebke, L. Grayson, "GEDI: a game engine for teaching videogame design and programming", Journal of Computing Science in Colleges, 21(2), 72–82, 2005.
- [8] J. O. Coplien, "Software Design Patterns: Common Questions and Answers", The Patterns Handbook: Techniques, Strategies, and Applications, Cambridge University Press, New York, pp. 311-320, 1998.
- [9] J. Distasio and T. Way, "Inclusive computer science education using a ready-made computer game framework", ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, pages 116-120, 2007.
- [10] C. Johnson and J. Voigt, "DXFramework", Web: <http://www.dxfamework.org>, Retrieved June, 2008.
- [11] A. O. Navarro and A. Hoek, "SimSE: an Educational Simulation Game for Teaching the Software Engineering Process", In ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, pages 233–233, New York, NY, USA, 2004. ACM Press.
- [12] GarageGames, "GarageGames", Web: <http://www.garagegames.com>, Retrieved June, 2008.
- [13] IBM, "INNOV8 – a BPM Simulator", Web: <http://www-304.ibm.com/jct03001c/software/solutions/soa/innov8.html>, Retrieved June 2008.
- [14] IEEE, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", Software Engineering Standards Committee of the IEEE Computer Society, 2000.
- [15] P. Kruchten, "The 4+1 View Model of Architecture", IEEE Software, 12, 6, Pp. 42 – 50, 1995.
- [16] L. Natvig, S. Line, and A. Djupdal, "Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals", In FIE 2004: Proceedings of the 2004 Frontiers in Education Conference, 2004.
- [17] Lifelong Kindergarten Group, MIT Media Lab, "Scratch | Home | imagine, program, share", Web: <http://scratch.mit.edu/>, Retrieved June.2008.
- [18] Microsoft corporation, "XNA developer center", Web: <http://msdn.microsoft.com/en-us/xna/aa937794.aspx>, Retrieved June,2008
- [19] M. Sharples, "The design of personal mobile technologies for lifelong learning", Computer & Education, 34(3-4):177–193, 2000.
- [20] B. Nitschke, "Professional XNA Game Programming: For Xbox 360 and Windows", Wiley Publishing, Inc.,2007.
- [21] I. Parberry, "SAGE: a simple academic game engine", Web: <http://larc.csci.unt.edu/sage>, Retrieved June 1, 2008.
- [22] D. P. Perry, and A.L. Wolf, "Foundations for the Study of Software Architecture", ACM Sigsoft Software Engineering Notes, 17(4), Pp. 40-52, 1992.
- [23] P. Clements L. Bass and R. Kazman, "Software Architecture in Practice Second Edition", Addison-Wesley, 2003.
- [24] R. Rosas, M. Nussbaum, P. Cumsille, V. Marianov, M. Correa, P. Flores, V. Grau, F. Lagos, X. Lopez, V. Lopez, P. Rodriguez, and M. Salinas, "Beyond Nintendo: design and assessment of educational video games for first and second grade students", Computers & Education, 40(1): 71–94, 2003.
- [25] Rockwell Automation Inc, "Arena Simulation Software", Web: <http://www.arenasimulation.com/>, Retrieved June 2008.

- [26] T. W. Malone, "What makes things fun to learn? Heuristics for designing instructional computer games", In SIGSMALL '80: Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, pages 162–169, New York, NY, USA, 1980. ACM Press.
- [27] T. Blomholm Kvamme and J.-E. Strøm, "Evaluation and Extension of an XNA Game Library used in Software Architecture Projects", Master thesis at NTNU, June 2008.
- [28] Washington State University Vancouver and University of Puget Sound, "The Java Instructional Gaming Project", Web: <http://ai.vancouver.wsu.edu/jig/>, Retrieved June. 2008
- [29] G. Sindre, L. Nattvig, M. Jahre, "Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals", to appear in IEEE Transaction on Education.
- [30] B.A. Foss and T.I. Eikaas, "Game play in Engineering Education - Concept and Experimental Results", The International Journal of Engineering Education 22(5), 2006.
- [31] A. I. Wang, T. Ø. and O. K. Mørch-Storstein: "An Evaluation of a Mobile Game Concept for Lectures", 21st IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2008), Charleston, S. Carolina, USA, April 14-17, 2008,.
- [32] M. S. El-Nasr and B. K. Smith, "Learning through game modding", ACM Computer Entertainment 4(1), Jan. 2006.
- [33] JGame project, "JGame: a Java game engine for 2D games", Web: <http://www.13thmonkey.org/~boris/jgame/>, Retrieved November 2008.
- [34] Adobe, "animation software, multimedia software – Adobe Flash CS4 Professional", Web: <http://www.adobe.com/products/flash/>, Retrieved November 2008.
- [35] Monoxna, "monoxna – Google Code", Web: <http://code.google.com/p/monoxna/>, Retrieved November 2008.
- [36] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere, "The Architecture Tradeoff Analysis Method," Engineering of Complex Computer Systems, IEEE International Conference on, vol. 0, no. 0, pp. 0068, Fourth IEEE International Conference on Engineering Complex Computer Systems (ICECCS'98), 1998.
- [37] A. BinSubaih, S.C. Maddock (2006), "Using ATAM to Evaluate a Game-based Architecture", Workshop on Architecture-Centric Evolution (ACE 2006), Hosted at the 20th European Conference on Object-Oriented Programming ECOOP 2006, July 3-7, 2006, Nantes, France.

Paper 6:

GDF2: Bian Wu, Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "An Evaluation of Using a Game Development Framework in Higher Education", 22nd IEEE-CS Conference on Software Engineering Education and Training (CSEET 2009), February 17-19, Hyderabad, India, 2009. ISBN: 978-0-7695-3539-5 DOI=10.1109/CSEET.2009.9



An Evaluation of Using a Game Development Framework in Higher Education

Bian Wu , Alf Inge Wang , Jan-Erik Strøm , and Trond Blomholm Kvamme
Dept. of Computer and Information Science
Norwegian University of Science and Technology
Bian/alfw@idi.ntnu.no, janerist/trondblo@stud.ntnu.no

Abstract

This paper describes an application of a Game Development Framework (GDF) - Microsoft XNA in software architecture (SA) course at Norwegian University of Science and Technology (NTNU) and evaluates how well the GDF is to use and integrate in a software engineering (SE) course. The result of the evaluation is based on the questionnaire with 9 types of general questions related to SE learning. In most aspects, the result shows that XNA is a suitable teaching aid in SE learning and can be used to teach SA. It is easy to use and save students time in development, thus let them have more time focusing on the course theory.

Keyword: *Game development framework, XNA, Software architecture, Software engineering education, Evaluation.*

1. Introduction

Research on games concept used in higher education has been done before, e.g. [2, 4, 3], but we believe there is an untapped potential that needs to be explored. This paper will change the angle from using games to teach to applying GDFs in student projects for learning computer skills, extending its application as a teaching aid in higher education. The GDF can be integrated mainly in three ways with a university course. *First*, it can be used to develop games that can replace traditional exercises. *Second*, it can be used to develop games that can be integrated in lectures to improve the participation and motivation of students. *Third*, the students can use a GDF in projects to develop software to understand the courses content related to computer science.

This paper will focus on GDF's application in higher education and evaluate its application in an existing course. The evaluation focuses the suitability of a specific GDF to be used by students, and whether the GDF is useful for the teaching and understanding course theory.

2. Application of a GDF in Higher Education

This section is a case study of Master course of SA at Norwegian University of Science and Technology (NTNU) to elaborate the application of a GDF used as a teaching aid in higher education.

2.1. Choice of the GDF

The course staff started searching for any GDF that provided high-level APIs to ease game development and that was easy to learn. As many GDFs were immature, we ended up with choosing Microsoft XNA (Xbox/DirectX New Generation Architecture) framework [5]. It was the most suitable framework for fast game development at that time. Another reason for choosing it was that support for developing game for XBOX 360 would be a motivation factor for students to put an extra effort into projects.

2.2. Student projects based on XNA

In NTNU's SA course, the goal of the project is for the students to apply the methods and theory from the course to design a SA and to implement a system based on XNA framework. The project consists of the following phases: 1) *COTS (Commercial Off-The-Shelf) exercise*: Learn the technology to be used through developing a simple application. 2) *Design pattern*: Learn how to use and apply design pattern by making changes in an existing system. 3) *Requirements and architecture*: List functional and quality requirements and design the SA for the application (a game). 4) *Architecture evaluation*: Use the ATAM (Architecture Tradeoff Analysis Method) evaluation method to evaluate the SA of project in regards to the quality requirements. 5) *Implementation*: Do a detailed design and implement the application based on the created architecture and on the changes from the evaluation. 6) *Project evaluation*: Evaluate the project as a whole using a PMA (Post-Mortem Analysis) method [1].

The course staff issues the task to make a functioning game using XNA. The game has to be designed according to a specified SA. Further, the students had to develop an architecture where they had to focus on one particular quality attribute: *Modifiability*, the game architecture and implementation should be easy to change in order to add or modify functionality; or *Testability*, the game architecture and implementation should be easy to test in order to detect possible faults and failures. The course's workload was 25% of one semester and students were grouped in 3-4 persons and spent most time on the implementation phase (6 weeks).

3. Evaluation of XNA used in a software architecture course

This paper investigates the XNA framework's usefulness for teaching students SA based on book "Software Architecture in Practice" [6]. Concretely, we investigate the following research questions:

- R1: To what degree does the COTS influence the learning process?
- R2: How much time is spent on technical matters and on architectural matters?
- R3: How difficult is it to integrate known architectural and design patterns and learn the necessary prerequisite skills to be able to develop programs?
- R4: How much do the students feel they have learned about SA through the game development project?

3.1. Questionnaire and Result

The participants of our survey were postgraduate students of NTNU's SA course. We published a questionnaire using the existing e-learning platform (It's Learning) three days after the delivery deadline of the students' projects, and received a total of 46 responses to the general questionnaire. Table 1 shows statistical results of quality attributes from students' projects. Table 2 lists the 9 general items and students responses.

Table 1: Distribution of responses related to assigned quality attributes

46 Responses in XNA	55% Testability
	45% Modifiability

Table 2: The 9 general questions labeled Q1-Q9

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly agree
----------	-------------------	----------	---------	-------	----------------

Q1: I found it hard to come up with good requirements	5%	30%	40%	20%	5%
Q2: I think the COTS did not hinder the design of a good architecture(Total)	5%	20%	35%	35%	5%
Q2.1 I think the COTS did not hinder the design of a good architecture(Testability)	10%	25%	30%	35%	0%
Q2.2 I think the COTS did not hinder the design of a good architecture(Modifiability)	0%	10%	45%	35%	10%
Q3: I found it difficult to evaluate the other group's architecture in the ATAM	0%	20%	15%	45%	20%
Q4: I think the COTS made it easier to identify architectural drivers(Total)	10%	15%	55%	20%	0%
Q4.1 I think the COTS made it easier to identify architectural drivers(Testability)	15%	15%	60%	10%	0%
Q4.2 I think the COTS made it easier to identify architectural drivers(Modifiability)	0%	15%	50%	35%	0%
Q5: I found it difficult to focus on our assigned quality attributes(Total)	10%	25%	10%	25%	30%
Q5.1 I found it difficult to focus on our assigned quality attributes(Testability)	10%	0%	0%	30%	60%
Q5.2 I found it difficult to focus on our assigned quality attributes(Modifiability)	10%	50%	20%	20%	0%
Q6: I found it easy to integrate known architectural or design patterns(Total)	0%	10%	40%	40%	10%
Q6.1 I found it easy to integrate known architectural or design patterns(Testability)	0%	10%	35%	45%	10%
Q6.2 I found it easy to integrate known architectural or design patterns (Modifiability)	0%	10%	45%	35%	10%
Q7: I spent more time on technical matters than on architectural matters	5%	20%	30%	30%	15%
Q8: I spent too much time trying to learn the COTS in the start of the course	5%	35%	30%	25%	5%
Q9: I have learned a lot about software architecture during the project(Total)	10%	15%	30%	40%	5%
Q9.1: I have learned a lot about software architecture during the project(Testability)	10%	10%	35%	45%	0%
Q9.2: I have learned a lot about software architecture during the project(Modifiability)	10%	20%	25%	35%	10%

3.2. Analysis of questionnaire results

Here we will evaluate the results against the stated problems.

R1: To what degree does the COTS influenced the learning process?

- *The requirements gathering and specification:* Reflected in Q1 of Table 2, the result did not show that the COTS made a significant impact on requirements phase of the project.
- *The design of the architecture:* From Q2, most of students agreed that the COTS did not hinder design of a good architecture. The major reason is that XNA supports different types of games development with flexible architecture. There was a tendency that modifiability groups thought the COTS was a less hindrance than the testability groups.
- *The ATAM evaluation:* Reflected in Q3, most of students found it difficult to evaluate another group's ATAM document. The main reason is because of XNA's open environment and different types of games, all of them have their own structure and playing style.
- *Architectural Drivers:* From Q4, it seems safe to say that the COTS did not influence the difficulty of identifying architectural drivers. And students who focused on modifiability

tend to agree more that the COTS made it easier to identify architectural drivers, while students who focused on testability tend to disagree more.

- *Quality Attribute Focus*: From Q5, the results indicate that choice of COTS does not have much influence on difficulty of focusing on the assigned quality attribute. And generally students with modifiability focus found it easy and students with testability focus found it difficult. The probably reason is that students have a much better understanding of modifiability, but creating a program that makes testing easier is a whole new area.

R2: How much time is spent on technical matters and on architectural matters?

From Q7, the choice of COTS influences the time the students have at their disposal to focus on architectural matters. From our own experience, the XNA environment is much more user-friendly with a high-level API, the probable reason was that many students were completely new to both C# and XNA.

R3: How difficult was it to integrate known architectural and design patterns and learn the necessary prerequisite skills to be able to develop programs?

- *Integrate known architectural and design patterns*: From Q6, the result shows that generally the majority of students thought it was easy. Also, we found when looking at the quality attribute distribution, there was several suitable patterns presented, such as Model-View-Controller, Pipe and filter, Layered, Task Control and so on.

- *Learn the necessary prerequisite skills to be able to develop programs*: From Q8, results show that majority students disagree that they spent too much time learning the COTS. Also, from our own experience, we received almost no help requests from students' groups.

R4: How much did the students feel they have learned about SA through the project?

From the results of Q9, most of the students feel they have learned a lot about SA throughout the project. For the negative attitude in result, probably due to the first time to use XNA in teaching and some of students might have become spellbound by the fun of creating a game, thus focusing more on game play than architecture.

5. Conclusion

In this paper we have presented an evaluation of the XNA integrated into a SA lecture. The result shows that XNA is easy to use, requires little time to develop, and supports different types of game development. Further, the students claimed that XNA framework contributed to increased learning and motivation.

6. References

- [1] A. I. Wang, T. Stålhane. Using Post Mortem Analysis to Evaluate Software Architecture Student Projects, Conference on Software Engineering and Training 2005, 8 pages.
- [2] Alex Baker, Emily Oh Navarro, and Andr'e van der Hoek. Problems and Programmers: an Educational Software Engineering Card Game. In ICSE '03: Proceedings of the 25th International Conference on Software Engineering, pages 614–619, Washington, DC, USA, 2003. IEEE Computer Society.
- [3] Emily Oh Navarro and Andr'e van der Hoek. SimSE: an Educational Simulation Game for Teaching the Software Engineering Process. In ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, pages 233–233, New York, NY, USA, 2004. ACM Press.
- [4] Lasse Natvig, Steinar Line, and Asbjørn Djupdal. Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals. In FIE 2004: Proceedings of the 2004 Frontiers in Education Conference, 2004.
- [5] Microsoft corporation. XNA developer centers. <http://msdn.microsoft.com/en-us/xna/aa937794.aspx>, Retrieved June, 2008
- [6] P. Clements L. Bass and R. Kazman. Software Architecture in Practice Second Edition, 2003. Addison-Wesley.

Paper 7:

GDF3: Bian Wu, Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm
Kvamme: "XQUEST used in Software Architecture Education", IEEE
Consumer Electronics Society's Games Innovation Conference, August 25-28,
2009, London, UK. ISBN: 978-1-4244-4459-5, DOI:
10.1109/ICEGIC.2009.5293607



XQUEST used in Software Architecture Education

Bian Wu

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
Bian@idi.ntnu.no

Jan-Erik Strøm

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
Janerist@stud.ntnu.no

Alf Inge Wang

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
Alfw@idi.ntnu.no

Trond Blomholm Kvamme

Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
trondblo@stud.ntnu.no

Abstract— This paper describes the motivation and application of a Microsoft XNA extended library- XQUEST (XNA Quick & Easy Starter Template) in a software architecture course. Further, it presents the evaluation of the usability and usefulness of the XQUEST library in the context of a software architecture course. XQUEST was designed and implemented to save students' time in development projects offering flexible components. The evaluation was based on the survey of students questionnaires. Finally, the questionnaire results were analyzed in relation to three aspects: suitability, usefulness and usability. In many aspects, the results show that XQUEST enhances XNA in suitability as a teaching aid in software engineering learning, and can be a useful and helpful extension to understand XNA. The results also show that XQUEST is easy to use and save students time in development, thus giving students more time to focus on the practice of course theory.

Keywords- XNA; Software architecture; Software engineering education; Evaluation; Games

I. INTRODUCTION

Research on games concept and game development used in higher education has been done before, e.g. [1, 2, 3], but we believe there is an untapped potential that needs to be explored due to development of new technologies. After some commercial SDKs have come out in recent years, such as XNA [6], iPhone SDK [19] or Android [20], we had considered how to use new technology and devices in the higher education to enrich the learning environment. This paper will focus on how to use a game development environment to teach software architecture or related courses. The motivation is to bring the same enthusiasm from playing games to learn to courses' contents through game development. The specific features of a game SDK can give new insights and provide support for the educational process used in the teaching directly, providing an open platform for students during teaching. The games and game development frameworks can be integrated mainly in three ways with a university course. *First*, they can be used to replace traditional exercises. This approach would motivate students to put extra effort into exercises and give teachers

and/or teaching assistants an opportunity to monitor how the students work with the exercises in real-time [22, 23]. *Second*, they can be integrated in lectures to improve the participation and motivation of students [24, 25]. The goal of proposed game concept is to prompt students and increase students' attendance in lectures. *Third*, the students can use them in projects to develop software to understand the courses' content related to software engineering or computer science [21, 26, 32].

This paper focuses on the latter, where students do game development to learn software engineering skills. Concretely, we focus on one specific game SDK, XNA. Our idea was to extend the XNA game libraries to improve it and make it more suitable for higher education in two ways: shorten students' development time, and improve the content and structure of XNA to fit certain course. Further, an evaluation and analysis of extension of XNA game libraries' application is presented.

The rest of the paper is organized as follows. Section 2 is an introduction of XNA and its application in software architecture course. Section 3 describes the XQUEST design and its structure. Section 4 describes an assessment of XQUEST application, Section 5 describes related work, and Section 6 concludes the paper.

II. XNA USED IN HIGHER EDUCATION

This section is a detailed discussion of XNA structure and its application in a software engineering course at Norwegian University of Science and Technology (NTNU).

A. XNA Structure

XNA is a game development platform developed by Microsoft, which includes a programming framework and a set of tools to offer a complete game development package [4]. The overview architecture of XNA consisting of four layers is shown in Fig. 1. Based on the .NET platform, XNA offers game development for the PC, the Xbox 360, and more recently the Zune [5] media player. Further, XNA uses the C# programming language. XNA mainly targets students,

hobbyists, and independent game developers. XNA is free to use, but to deploy games on the Xbox 360, a subscription to the XNA Creators Club [6] is required. XNA was motivated by an earlier attempt at bringing the DirectX C++ multimedia API [7] over to the .NET platform, called Managed DirectX [8]. It was essentially a 1:1 mapping of the DirectX API onto .NET. XNA took the idea one step further and provides a complete game development solution, not just the programming API. First released version 1.0 was shipped in December 2006, and the latest version of XNA is 3.0, released in October 2008 [6].

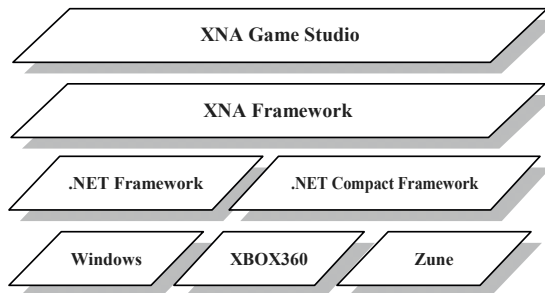


Figure 1. The Deployment View of XNA

B. XNA used in Software Architecture Course

The software architecture course is a post-graduate course offered to computer science and software engineering students at NTNU. The course is taught every spring based on the book *Software Architecture in Practice* [9], and its workload is 25% of one semester. In the software architecture course, 30% of the grade is based on an evaluation of a software architecture project all students have to do. The rest 70% is given from a written examination. The goal of the project is for the students to apply the methods and theory in the course to design software architecture and to implement a system based on XNA framework according to the architecture. The project consists of the following phases [32]:

- 1) *COTS (Commercial Off-The-Shelf) exercise*: Learn the technology to be used through developing a simple application.
- 2) *Design pattern*: Learn how to use and apply design pattern by making changes in an existing system.
- 3) *Requirements and architecture*: List functional and quality requirements and design the software architecture for the application (a game).
- 4) *Architecture evaluation*: Use the ATAM (Architecture Tradeoff Analysis Method) evaluation method to evaluate the software architecture of project in regards to the quality requirements.
- 5) *Implementation*: Do a detailed design and implement the application based on the created architecture and on the changes from the evaluation.
- 6) *Project evaluation*: Evaluate the project as a whole using a PMA (Post-Mortem Analysis) method [10].

The course staff issued the tasks to make a functioning game using XNA, based on students' own defined game concept. However, the game had to be designed according to a specified and designed software architecture. Further, the students had to develop an architecture where they had to focus on one particular quality attribute. We used following definitions for the quality attributes in the game projects: *Modifiability*, the game architecture and implementation should be easy to change in order to add or modify functionality; and *Testability*, the game architecture and implementation should be easy to test in order to detect possible faults and failures. These two quality attributes also were related to the course content.

III. MOTIVATION AND OVERVIEW OF XQUEST

XQUEST (XNA QUick & Easy Starter Template) [11] is a small and lightweight 2D game library/game template developed by the two master students Strøm and Kvamme at NTNU (co-authors of this paper) that contains convenient game components, helper classes, and other classes that can be used in the XNA game projects. The goal of the XQUEST project was to identify and abstract common game programming tasks and create a set of components that could be used by students of the course to make their programming life easier. We chose to focus mainly on 2D. There were a few reasons for this. *First*, the focus of the student projects is software architecture, not making a game with fancy 3D graphics. *Second*, students unfamiliar with game programming and 3D programming may find it daunting to have to learn the concepts needed for doing full-blown 3D in XNA, such as shade programming and 3D-modelling, in addition to software architectures. To keep the projects in 2D may reduce the effect of students only focusing on the game development instead of focusing on the software architecture issues. However, we still consider to implement basic 3D components in XQUEST and help documentation for the students interested in 3D gaming programming, but it is not a mandatory for students to use them.

A. Motivation for XQUEST

From the feedback of using XNA in the software architecture course [21], the majority of the students thought there was much time focus on game issues and little time on software architecture, even the XNA environment is very developer-friendly with a high level graphic API. The following Table I is a collection data from 46 students' replies in software architecture course. From the table we can see that a great percentage of the students (55%) claim that too much time is spent on developing game play compared to time spent on the software architecture.

Apart from the negative feedback above, we also consider to enrich educational features of XNA for the software architecture course:

- Save time in C# learning and programming and add appropriate guidance and cases on mini 3D game programming.
- Provide cases of good designed software architecture based on XNA and XQUEST.

TABLE I. COLLECTION DATA ABOUT DEVELOPING TIME ON XNA FROM STUDENTS

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Q: I spent too much time developing the game play and not enough time on the architecture	5%	30%	10%	40%	15%

- Provide some documentation to explain the trade-off between architecture design and COTS, especially XQUEST components.

B. Design Principles for XQUEST

Here were the general principles used to design XQUEST:

1) *Flexible structure*: Due to XQUEST is used for teaching software architecture, and students will design game projects based on suitable software architectures like Model-View-Controller, Pipe and filter, Layered, Task Control and etc., we tried to provide flexible components for the students that would not hinder the students to design their own software architecture.

2) *Easier to use*: From our teaching experiences, 90% students had programming skills in Java, but not in C#. XQUEST should help them to learn XNA in an easy and quick way with good comments on code and supported documentation. And XQUEST is based on XNA and it should make it easier for students to use and save development time. Students should learn it quickly even they only have experience in Java programming.

3) *3D guidance*: We intended to lead students into the world of 3D, and give them the basic ideas of 3D programming. But 3D programming needs more time on 3D models and some basic 3D transformations to 2D on screen. We should provide several demos in XQUEST to show some 3D technologies and give the students some sensorial 3D concepts in mind and to get a quick entrance into the 3D world. Thus, they did not need to know the Math basics of how to do 3D transformations into 2D.

4) *Providing tutorials to investigate the software architecture in game*: Very little literature have been published on the subject of software architecture in game development, although some attempts had been done [12, 13, 14]. However, these attempts have failed to deliver a general high-level presentation on software architecture topics in games, and tend to focus more on the design and implementation of software modules common in games. We have looked into the differences between traditional software development and game development, as well as identifying different architectural and design patterns that were useful for game development. Also, portraying the challenges of designing and implementing game architectures could be proved useful for determining the scope of such an endeavor.

5) *Reflect the quality attributes in a game architecture*: Quality attributes should be the driving force behind every big decision in the development process, and had to be considered

at all times. We identified some quality attributes that were most relevant for game development, such as modifiability, testability, availability or usability. We would look at structures and patterns that underline certain quality attributes, and to use these elements in a game architecture.

C. XQUEST Structures and Components

The XQUEST library is presented component by component as shown in Fig. 2. The XQUEST functionality is split into eight components described with their relationships. We put the XQUEST.GameObjectManagement component in the middle purposely to indicate its importance. It contains the game object system, which is at the heart of XQUEST.

Here is a brief description of each component:

1) *XQUEST.GameObjectManagement* contains the game object system, responsible for handling game objects such as players, enemies, power ups, etc. The object system allows for many different types of objects, 2D or 3D, and provides state tracking and a flexible collision detection system.

2) *XQUEST.CameraSystem* provides functionality for setting up both 2D and 3D cameras to view a scene or track game objects in multiple perspectives.

3) *XQUEST.GameStateManagement* handles state and state transitions in the game. It uses the concept of game screens. A game screen can be a menu screen, an inventory screen, a combat screen, etc.

4) *XQUEST.Audio* handles audio-related functionality like playback of sound effects and music, adjustment of volume levels, grouping into categories, etc.

5) *XQUEST.Misc* contains miscellaneous components of utility that do not fit into any other namespace.

6) *XQUEST.Input* handles querying and interpretation of keyboard, mouse, and Xbox 360 game pad input.

7) *XQUEST.Helpers* contains convenient helper classes for common tasks.

Besides of above components, we also provided some demos and directions to illustrate how to use these components, what kind of architecture used in one demo and what types of attributes (modifiability, testability or others) it focuses on.

IV. EVALUATION OF XQUEST USED IN A SOFTWARE ARCHITECTURE COURSE

One goal of this paper was to investigate how successfully the XQUEST could be applied in a software architecture course. Concretely, we investigated the following research questions:

- RQ1: What is the usability of XQUEST?

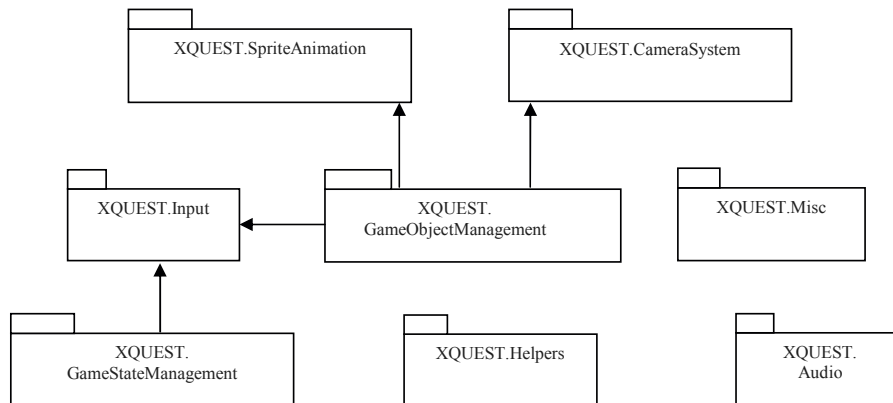


Figure 2. Structure View of XQUEST

- RQ2: What is the suitability and usefulness of XQUEST?
- RQ3: What is the usefulness of the specific components in XQUEST?
- RQ4: What other positive or negative issues are related to XQUEST?

A. Preparation for Evaluation

In this part, we present our research methods and research context of the evaluation process.

1) The research methods:

a) *System Usability Scale*: The System Usability Scale [17], hereafter SUS, is a usability questionnaire consisting of ten generic Likert items. Responses to the questionnaire result in a score, called the SUS score. The SUS score is a single number between 0 and 100 indicating the overall usability of the system being studied. The SUS is used for subjectively measuring usability of a system at a high level. The outcome of a SUS questionnaire is a score within the range of 0 to 100, where higher values indicate a higher measured usability of the system. Each item in the SUS is responded to by assigning a scale value from 1 to 5, where 1 indicates strong disagreement and 5 indicates strong agreement. To calculate the SUS score, we first sum together the score contributions for each question. Each question's score contribution is a number in the range 0 to 4. There are totally 10 questions, for odd-numbered questions (1, 3, 5, 7, 9), the score contribution is given by the scale position minus 1. For even-numbered questions (2, 4, 6, 8, 10), the score contribution is 5 minus the scale position. The sum of the score contributions are then multiplied by 2.5 and divided by the number of replies to the survey to obtain the final SUS score. We had incorporated the SUS items in our XQUEST questionnaire.

b) *Empirical investigation*: The survey was based on the method of conducting an empirical investigation [15]. We had applied the recognized methods combined with our own

subjective assessments to implement the survey. Measurable items in the questionnaires had been formed as Likert [16] items. These were not questions, but statements that the respondents responded to by specifying their agreement to the statements. We had used 5-level Likert items, where the levels of agreement were: Strongly Disagree; Disagree; Neutral; Agree and Strongly Agree. The items from the questionnaires were assessed subjectively. And these subjective analyses were based on our teaching experiences.

2) The participants and Environments.

The participants of our survey were postgraduate students of the software architecture class spring 2008 at NTNU. They used an online e-learning platform during the course. The questionnaire was published three days after the students had delivered their projects on the e-learning platform by using its survey functionality. Each question was prefixed with a context name indicating which section it belonged to. The participants and environment was authentic in the sense that the students of the course were the intended users of XNA and XQUEST.

B. Results from System Usability Scale (SUS) Questions

Here we present the results from the SUS part of the XQUEST questionnaire.

The result of SUS score is shown in Table II. Our system achieved a score of 60.53 out of a possible 100. It is above the average usability, which indicates that the system is not difficult to use. The main challenge for some students was to spend required time on becoming familiar with the 2D/3D structure in XQUEST. This process can be improved by giving an introduction lecture about 2D/3D concept after first simple exercise when the students had already setup some experiences and context of XNA programming environment. We also need to improve 2D/3D components into two separate components in next XQUEST version.

C. Results from Suitability and Usefulness Questions

The results from the questions about suitability and usefulness of XQUEST are shown in Table III. This

questionnaire was only for students using both XNA and XQUEST in project. We received a total of 19 responses from the students using XQUEST out of the 46 students that worked on an XNA game project.

The results showed that students could use XQUEST as a template or referring to it as a library, 40% modified the code of XQUEST, and 30% kept it unchanged. This reflects of the fact that one successful design philosophy of XQUEST was to create it as abstract and reusable as possible, enabling the students to choose their own ways of using XQUEST that helped the project design.

Q2 shows a positive result that XQUEST prepared most commonly used components for students, and it saves the time in game development.

We found that 60% students disagreed that they spent too much time looking into the source code of XQUEST. This positive result indicates our good documentation work and self-explanatory public interfaces. This positive result is also caused by Visual Studio's functionality to show comments in source code through the IntelliSense [18] tooltips that pop up while the programmer is typing in code. For example, when creating a new instance of a class, the IntelliSense will display any comments available for the different parameters that the constructor accepts.

It is also inavoidable that students should both focus on architectural matters and on technical matters. However, from the Q4 result, to a certain extent, XQUEST still could help one third of the students more on architectural matters than on technical matter.

D. Results from Usefulness of Every Component Questions

From Table IV result, we could find out how the students used XQUEST in their projects related to the offered components, and where we should focus on to improve XQUEST.

As we expected, the most popular component was the Animated Sprite Framework. Since all groups worked on 2D games, this is understandable since sprite rendering is the easiest way to output graphics in a 2D environment. Another popular component was the Game Object Management component. From the results going through the XNA deliveries, we were surprised to find that most groups did not create their own implementation of the `IGameObject` interface, but rather used the standard `BasicGameObject`. Using `BasicGameObject` has some limitations, as it is tightly interwoven with the Sprite Animation Framework for representing the game object using sprites. This implies that the groups that used this approach, also needed to use the Sprite Animation Framework. Looking at the high percentage of students who responded to have used both of these components, there is no doubt that the use of `BasicGameObject` is the main reason for this. This finding shows that the students need to pay attention during lectures.

In third place comes the `InputManager` component. This is probably the most useful component in XQUEST, since every game needs to handle input in some forms. It contains several

methods for supporting all the XNA input devices such as keyboard, mouse, and up to four Xbox 360 game pads. By looking at the deliveries, we found that most games were single-player games played with a keyboard, or hot-seat multiplayer games where all players shared the same keyboard. Some games used the mouse as the primary input device, but very few implemented gamepad support, since they did not have access to Xbox 360 game pads during the project unless they brought one themselves. The input needs may therefore have not been so great that it required a component like the `InputManager`. We will therefore simplify the functions in `InputManager` component to minimize the amount of code the students needs to read to save total time of code reading, such as delete input support for XBOX360 according to the practical application from students.

The least used components were the `AudioManager` and `TextOut` components. Having music and sound effects in your game may not take the greatest priority in a school project, where the evaluation criteria leans more towards software architecture and fulfilling an assigned quality attribute. For this reason, many groups decided not to implement audio features in their games to save development time, and hence no need for the `AudioManager` component. Still, almost half of the games used this component. The `TextOut` component was a component we thought would be more popular. It is really simple to use, and has features that makes it very convenient for text display. It may be the fact that text display is so simple that the students did not see the need for using it. The standard way of displaying text with `SpriteBatch` may fulfill all the desired text rendering needs. In this way, we could also cut some functions in `TextOut` to save coding reading workload.

E. Open Questions Analysis

Table V is the collection of main feedback from students to the open question. 20% of the students agreed that some components were missing in XQUEST. Pixel-perfect collision detection is a very performance-intensive operation that we described as not suitable for a multi-purpose game object system such as the one in XQUEST. However, we decided to include support for it in next XQUEST version. It is disabled by default, but can be enabled on a per-object basis, meaning the user is in total control of how the collision detection should be executed for every object in the scene.

`BasicGameObject` is per definition not supposed to be flexible. The flexibility of the game object management system in XQUEST lies in the `IGameObject` interface, of which `BasicGameObject` is an implementation. As expressed above, we were surprised that so few groups did not take advantage of this flexibility by providing their own implementation of the `IGameObject` interface. By doing so, they could have tailored it for their game. Instead, they chose to use the standard implementation in `BasicGameObject`, which of course also constrained them to using the `Sprite` class for the graphical representation.

TABLE II. RESULTS FROM THE SUS

Question	Sum score contribution of 19 students
1 I think that I would like to use this system frequently.	35
2 I found the system unnecessarily complex.	52
3 I thought the system was easy to use.	50
4 I think that I would need the support of a technical person to be able to use this system.	55
5 I found the various functions in this system were well integrated.	47
6 I thought there was too much inconsistency in this system.	48
7 I would imagine that most people would learn to use this system very quickly.	44
8 I found the system very cumbersome to use.	45
9 I felt very confident using the system.	40
10 I needed to learn a lot of things before I could get going with this system.	44
Sum:	460
SUS Score:	$460 * 2.5 / 19 = 60.53$

TABLE III. THE 5 GENERAL QUESTIONS LABELED Q1-Q5

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
Q1: I found that I could use XQUEST as is without modifications	15%	25%	30%	25%	5%
Q2: I think XQUEST saved me a lot of time and effort by providing components and functionality that I otherwise would have had to create myself	10%	10%	15%	40%	25%
Q3: I spent too much time looking into the XQUEST source code	10%	50%	25%	15%	0
Q4: I think XQUEST helped me focus more on architectural matters and less on technical matters	5%	35%	30%	30%	0

TABLE IV. QUESTIONS ABOUT EVERY COMPONENT IN XQUEST

#	I used the following components of XQUEST		
	Sprite/ AnimatedSprite	GameObject Manager	InputManager
Component			
Percentage	90%	85%	75%
Component	TextureStore	AudioManager	TextOut
Percentage	65%	40%	30%

TABLE V. OPEN QUESTION COLLECTION

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Q: I think there were components missing that most students could benefit from	0	25%	55%	10%	10%
Q: If you felt there were components missing, which ones would you like to see in a future version of XQUEST?					
A1. Sprite layers and pixel collision detection.					
A2. Pixel-based collision detection, system for being called with certain intervals, better modifiability.					
A3. More flexible BasicGameObject, allowing non-sprite objects.					

V. RELATED WORK

This paper described experiences how to improve and enhance the XNA for teaching purposes in a software architecture course. As far as we know from the literature, XNA is always *directly* used in education without any modifications. There are only few papers describing its application in education and no paper goes further to describe the idea to extend the XNA's structure to enhance its features as a teaching aid for certain course. However, there are some related approaches used in education described in this section.

Joe Linhoff describes a game development course that uses the XNA platform to allow a heterogeneous group of students to gain experience in all aspects of console game creation [31]. It uses the features of XNA directly for the teaching, such as Pipeline or console that could be XBOX360 to activate students' programming interesting.

Youngblood describes how XNA game segments can be used to engage students in advanced computer science education [27]. Game segments are developed solution packs providing the full code for a segment of a game with a clear element left for implementation by a student. The paper describes how XNA was used in an artificial intelligence course

where the students was asked to implement a chat bot, motion planning, adversarial search, neural networks and flocking. Finally the paper describes seven design principles for using game segments in CS education based on lessons learned.

Oliver Denninger and Jochen Schimmel present their experiences utilizing game programming for project courses based on XNA [30]. Game programming usually involves many repetitive and time consuming tasks such as accessing hardware resources and managing game content. Since XNA framework relieves programmers from many of the tedious tasks and allows them to develop a feature complete game and to gain experience with the process of software development, students were so fascinated by the subject that they prefer to spend more time on the courses.

El-Nasr and Smith describes how the use of modifying or modding existing games can be used to learn computer science, mathematics, physics and ascetic principles [26]. The paper describes how they used modding of the WarCraft III engine to teach high school students a class on game design and programming. Further, the describe experiences from teaching university students a more advanced class on game design and programming using the Unreal Tournament 2003 engine. Finally, they present observations from student projects that involve modding of game engines. Although the paper claims to teach students other things than pure game design and programming, the game engine is used in the context of game development courses.

The Labyrinth [28] was implemented in Java and it is a flexible and easy-to-use computer game framework. The framework enables instructors to expose students to very specific aspects of computer science courses. The framework is a finished game in the Pac-Man genre, highly modular, and it lets the students change different aspects of the game. However, it cannot be used to develop different genres of game and there is little room for changing the software architecture of the framework.

The JIG (Java Instructional Gaming) project [29] is a collaborative effort between Scott Wallace (Washington State University Vancouver) and Andrew Nierman (University of Puget Sound) in conjunction with a small group of dedicated students. It has three aims: 1) to build a Java instructional game engine suitable for a wide variety of students at all levels in the curriculum; 2) to create a set of educational resources to support the use of the game engine at small, resource-limited, schools; and 3) to develop a community of educators that use and help improve these resources. The JIG project was proposed in 2006 and the JIG engine 1.0 is available now.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the principles to design XQUEST to improve XNA teaching functions for students in the exercise of the software architecture course. Furthermore, we evaluated the XQUEST application and analyzed several aspects of XQUEST's suitability, usefulness and usability based on questionnaires. In many aspects, the results show that XQUEST enhances XNA in suitability as a teaching aid in software engineering learning, and that it can be a useful and helpful extension to understand XNA. The results also show

that it is easy to use and save students time in development, and let students have more time to focus on the practice of course theory.

This paper describes results from the first time we have used XQUEST in the software architecture course. Based on our experiences and evaluations so far we acknowledge that more work needs to be done in improving the components in XQUEST to make them more useful, and updating the documentation due to updated XNA versions. We will go further to extend game library and enriching help resources of XQUEST in 3D development.

ACKNOWLEDGMENT

We would like to thank Jan-Erik Strøm and Trond Blomholm Kvamme for implementing XQUEST and for their inputs to this paper.

REFERENCES

- [1] Alex Baker, Emily Oh Navarro, and André van der Hoek. Problems and Programmers: an Educational Software Engineering Card Game. In ICSE '03: Proceedings of the 25th International Conference on Software Engineering, pages 614–619, Washington, DC, USA, 2003. IEEE Computer Society.
- [2] Emily Oh Navarro and André van der Hoek. SimSE: an Educational Simulation Game for Teaching the Software Engineering Process. In ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, pages 233–233, New York, NY, USA, 2004. ACM Press.
- [3] Lasse Natvig, Steinar Line, and Asbjørn Djupdal. Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals. In FIE 2004: Proceedings of the 2004 Frontiers in Education Conference, 2004.
- [4] N. Landry, "Microsoft XNA: Ready for Prime Time?," in CoDe Magazine, vol. Sept/Oct, 2007.
- [5] Microsoft; "Zune.net", <http://www.zune.net/>. Retrieved April 24, 2008.
- [6] Microsoft; "XNA Creators Club Online", <http://creators.xna.com/>. Retrieved May 21, 2008.
- [7] F. Luna, Introduction to 3d Game Programming with Direct X 9.0c. Plano: Wordware Publishing, Inc, 2006.
- [8] T. Miller, "Managed DirectX 9 Graphics and Game Programming," Sams Publishing, 2004.
- [9] P. Clements L. Bass and R. Kazman. Software Architecture in Practice Second Edition, 2003. Addison-Wesley.
- [10] A. I. Wang, T. Stålhane. Using Post Mortem Analysis to Evaluate Software Architecture Student Projects , Conference on Software Engineering and Training 2005, 8 pages.
- [11] Trond Blomholm Kvamme and Jan-Erik Strøm, "Evaluation and Extension of an XNA Game Library used in Software Architecture Projects", Master thesis in NTNU, June 2008.
- [12] A. Rollings and D. Morris, Game Architecture and Design, 2 ed.: New Riders Publishing, 2003.
- [13] D. Eberly, 3d Game Engine Architecture. Amsterdam: Morgan Kaufman Publishers, 2005.
- [14] R. Rucker, Software Engineering and Computer Games. Boston: Addison-Wesley, 2003.
- [15] V. R. Basili, "The Experimental Paradigm in Software Engineering," in Dagstuhl Workshop. vol. Experimental Software Engineering Issues: Critical Assessment and Future Directives, H. D. Rombach, V. R. Basili, and R. W. Selby, Eds. Dagstuhl Castle, Germany: Springer-Verlag, 1992, pp. 3-12.
- [16] Microsoft, "Shader Series Primer: Fundamentals of the Programmable Pipeline in XNA Game Studio Express," 2007. <http://creators.xna.com/downloads/?id=128>

- [17] J. Brooke, "SUS - A quick and dirty usability scale," in *Usability Evaluation in Industry* London: Taylor and Francis, pp. 189-194.
- [18] Wikipedia; "IntelliSense", <http://en.wikipedia.org/w/index.php?title=IntelliSense&oldid=208720089>. Retrieved May 7, 2008.
- [19] Apple. "iPhone Dev Center", <http://developer.apple.com/iphone/>, Retrieved February 2, 2009.
- [20] Google. "Android - An Open Handset Alliance Project", <http://code.google.com/intl/en/android/documentation.html>. Retrieved February 2, 2008.
- [21] Bian Wu, Alf Inge Wang, Jan-Erik Strøm, Trond Blomholm Kvamme, "An Evaluation of Using a Game Development Framework in Higher Education," CSEET, pp.41-44, 2009 22nd Conference on Software Engineering Education and Training, 2009
- [22] G. Sindre, L. Nattvig, M. Jahre, "Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals", to appear in *IEEE Transaction on Education*.
- [23] B.A. Foss and T.I. Eikaas, "Game play in Engineering Education - Concept and Experimental Results", *The International Journal of Engineering Education* 22(5), 2006.
- [24] A. I. Wang, O. K. Mørch-Storstein, T. Øfsdahl, "Lecture quiz - a mobile game concept for lectures", *The 11th IASTED International Conference on Software Engineering and Application (SEA 2007)*, November 19-21, 2007.
- [25] A. I. Wang, T. Ø. and O. K. Mørch-Storstein: "An Evaluation of a Mobile Game Concept for Lectures", 21st IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2008), Charleston, S. Carolina, USA, April 14-17, 2008.
- [26] M. S. El-Nasr and B. K. Smith, "Learning through game modding", *ACM Computer Entertainment* 4(1), Jan. 2006.
- [27] Youngblood, G. M. 2007 Using XNA-GSE Game Segments to Engage Students in Advanced Computer Science Education. In *The 2nd Annual Microsoft Academic Days Conference on Game Development*, February 22-25.
- [28] Distasio, J. and Way, T. 2007 Inclusive computer science education using a ready-made computer game framework. In *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, 116-120.
- [29] Washington State University Vancouver and University of Puget Sound. 2008 The Java Instructional Gaming Project. Web: <http://ai.vancouver.wsu.edu/jig/>, Retrieved June 2008.
- [30] Oliver Denninger, Jochen Schimmel, *Game Programming and XNA in Software Engineering Education*, *Proceedings of Computer Games and Allied Technology (CGAT08)*, 2008.
- [31] Joe, L. and S. Amber (2008). Teaching game programming using XNA. *Proceedings of the 13th annual conference on Innovation and technology in computer science education*. Madrid, Spain, ACM.
- [32] Wang, Alf Inge; Wu, Bian. An Application of a Game Development Framework in Higher Education. *International Journal of Computer Games Technology* 2009 ;Volume 2009.

Paper 8:

GDF4: Bian Wu; Alf Inge Wang; Anders Hartvoll Ruud; Wan Zhen Zhang: "Extending Google Android's Application as an Educational Tool", the 3rd IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), April 12-16 2010, Kaohsiung, Taiwan. ISBN: 978-1-4244-6433-3. DOI: 10.1109/DIGITEL.2010.38

Extending Google Android's Application as an Educational Tool

¹Bian Wu,¹Alf Inge Wang,¹Anders Hartvoll Ruud

¹Norwegian University of Science and Technology,
¹Norway

¹bian,alfw,anderru@idi.ntnu.no

²Wan Zhen Zhang

²Guilin University of Electronic Technology,
²China

²zwan_zer@163.com

Abstract—This paper introduces how to extend Google android platform as a game development tool to learn software architecture based on the double stimulation method. It starts with the motivation to choose the android platform since most of students in software architecture course from NTNU (Norwegian University of Science and Technology) have experiences of using java and eclipse platform before they starts this course. And then it describes the design and construct of extended android platform, called “Sheep” framework. Further, it presents the application of the Sheep framework as second stimulus means integrated in the game exercises in the software architecture course. Finally, the paper discusses the contribution from the aspects of technology, game ideas and pedagogy.

Keywords- Google Android; Higher Education; Double Stimulation; Software Architecture; XNA; iphone SDK

I. INTRODUCTION

The rapid development of electronic devices and network communication provides a foundation for improving the learning and teaching environments through technology. A common phenomenon is that new game ideas grow up with distinctive technology or novel equipments used in learning, and it also brings a challenge to educational games, how we could integrate games in lectures, exercises, day life with recent technologies, such as 3G[11], PSP [10], iphone [3] or youtube, etc, to enrich the teaching or training environment and achieve better learning life.

However, when we live in and start to deliberate our learning and teaching in technology rich learning environments, we are facing some challenges and opportunities that arise from introducing technology into learning and teaching. Most of the theoretical literature on learning and teaching has not yet incorporated a perspective on technology as to how perceive learning and teaching, especially on game-based learning. As such, we would discuss it by present cases of how game technology to perceive the learning in this paper.

This paper's idea was inspired by the work on using XNA [13] successfully in teaching software architecture through students' teamwork on game projects [1]. Similarly we want to see if we could use other development frameworks than XNA for teaching software engineering and computer science. Currently, most attractive choices are the Google android [2] and iphone SDK [3], both are issued in 2007 and free to download from their official websites. After two years, these SDKs become more matured, the newest

version of iphone SDK is 3.1, and android is 1.6. Both of them have potential power to enrich the learning life through diverse ways based on the various educational purposes.

This paper is organized as follows: Section 2 describes the theoretical context, previous works and investigates the features of Google android and iphone SDK. Section 3 introduces why and how to extend android platform as a game development tool for teaching purpose. Section 4 explains design issues and results. Section 5 presents how to integrate game development into teaching context based on our extended android platform for software architecture course. Section 6 presents a discussion of the teaching method from different aspects, and Section 7 concludes the paper.

II. RELATED WORKS

A. Theoretical Context

In schools, learners face a challenge, a problem, or a task that has been designed for a particular pedagogical purpose or they face situations that are likely to appear in work and public life. In both cases the purpose of exploiting tools is for learners to respond to such diverse challenges. Our focus is on the construct of the relationship between the educational tasks and the material artefact. This relationship is at the heart of Vygotsky's notion of double stimulation [14], a method for studying cognitive processes and not just results. In a school setting, typically the first stimulus would be the problem, challenge, task, or assignment to which learners are expected to respond. The second stimulus would be the available mediating tools. However, it is important to note that Vygotsky described this relationship in dynamic terms and where the second stimulus is not a discrete end point for this process but, “Rather, we simultaneously offer a second series of stimuli that have a special function. In this way we are able to study the process of accomplishing a task by the aid of specific auxiliary means” (p. 74, emphasis in the original). Note that Vygotsky identifies the second stimulus in the plural—a series. We take this to be most important when approaching the second stimulus in the form of digital tools [15].

Based on this point, we have a case design of learning environment present as below to describe how to construct the double stimulation in software architecture course. Also the design of the first stimulation (tasks) and criteria to

choose second stimulation (game development tools) are also given.

B. Previous works – Student projects based on XNA in software architecture course

In NTNU (Norwegian University of Science and Technology), the software architecture course is a post-graduate course offered to computer science students for one semester. Students were grouped in 3-4 persons and spent most time on the implementation phase (6 weeks) to finish game projects. The goal of the project is for the students to apply the methods and theory from the course to design software architecture and to implement a system (game) based on Microsoft XNA framework [4, 8]. The project consists of the following tasks:

- 1) *COTS (Commercial Off-The-Shelf) exercise*: Learn the technology to be used through developing a simple game.
- 2) *Design pattern*: Learn how to use and apply design pattern by making changes in an existing system.
- 3) *Requirements and architecture*: List functional and quality requirements and design the software architecture for a game.
- 4) *Architecture evaluation*: Use the ATAM (Architecture Trade off Analysis Method) evaluation method to evaluate the software architecture of project in regards to the quality requirements.
- 5) *Implementation*: Do a detailed design and implement the game based on the created architecture and on the changes from the evaluation.
- 6) *Project evaluation*: Evaluate the project as a whole using a PMA (Post-Mortem Analysis) method [12].

The second stimulus is chosen based on Malone's "What makes things fun to learn?" [16] and our own teaching experiences. The following are the criteria:

- Easy to learn and allow rapid development;
- Providing an open development environment to attract students' curiosity;
- Supporting programming languages familiar to the students;
- Not in conflict with the educational goals of the course;
- A stable implementation;
- Have sufficient documentation;
- Low costs to use and acquire. From our previous experiences, we found XNA to be a suitable tool in the software architecture course according to overall positive feedback from the students [1].

C. Features of the Google android and iphone SDK

This section compares the differences between android and iphone SDK. Table 1 is the summary features of the Google android and iphone SDK. Also, in order to get the overall understanding of game development platform, we also list the XNA's features for the comparison in the Table.

Both android and iphone SDK have strong support and market share. From technical perspective, they all have the potential value that could be extended as game development tools since most of their applications in the market are about games.

From the evaluation of the two SDKs [17], we decided to go for the android SDK to be used in the software architecture course to learn the syllabus through a project. However, we found that before android could be used, we had to tailor it for our educational purpose.

III. EXTENSION OF ANDROID FOR AN EDUCATIONAL PURPOSE

This section gives motivation of improving the android for teaching purpose and direction of how to extend it.

Due to the different educational environments and teaching aims, there could be various extending methods and directions. Under this situation, we will give a case study on how to improve Google android platform under the direction of learning software architecture through game projects.

A. Motivation

From our experiences of using XNA in software architecture course, the survey of students' context in NTNU are nearly 90% have background of java programming [4], and less than 20% have the background of C# or even more less have the Objective C experiences. Most of students face the time consuming of learning new programming languages if they choose the game project in software architecture course. This point is very important, since they have only 6 weeks for the implementation, and they also involve in other courses. As such, Google android could give one more choice for students with java background and enrich the resources for the second stimulus (game development tools) during teaching process. Moreover, using android to develop mobile games also could attract students' attention. So, our goal is as follows:

- Extend the android platform as a game development framework to match the first stimulus (tasks) based on the double stimulation;
- Save students game programming time, let them have more time focus on the course theory.

B. Direction of improving android

From our knowledge, there is no paper that describes extending android or iphone SDK's application as a game development tool for an educational purpose, so not much previous experiences are available. We must start from validating which of the desired characteristics are present in the extended android platform--we called it "Sheep" framework. According to our goal, Sheep both enhances the students' learning experience and helps them achieving their goal faster by saving game development time.

While the android development kit provides a huge programming interface for general application development, the Sheep framework should not only focus on game

TABLE I. FEATURES OF GOOGLE ANDROID, IPHONE SDK AND XNA

Criteria		Google Android	iPhone SDK	XNA
Development Environment		Eclipse recommended by Google	Xcode provided by Apple	Visual Studio and XNA Game Studio provided by Microsoft
Operating Systems for Development		Windows, Mac OS X, Linux	Mac OS X	Windows
Documentation		Official developer website provided	Official developer website provided	Official developer website provided
Emulator		Provided	Provided	Provided
Programming Language		Java	Objective-C	C#
Mobile Devices	Phone	Google phone is available in most countries.	iPhone is available in most countries.	No mobile phones type.
	Digital player	No digital player type.	iPod touch is a great developer device, no SIM card required.	ZunePlayer accepts partly XNA games, no SIM card required.
Programming Interface		API contains key high-level abstractions which short development time.	Mainly rely on the low-level standards, like OpenGL ES and OpenAL.	Contains high level abstractions to ease the game programming.
Share of Applications		Publish/sell the applications on Google android Market.	Publish/sell the applications on iTunes apple store.	Publish on the XNA creator club websites.

development, but also on game development for the purpose of learning software architecture to meet the tasks design in the first stimulus in further.

C. Method

The method we used to get inputs for the required features of the Sheep framework was a survey of previous students' exercises in the software architecture course.

This section presents a survey of student projects based on XNA submitted in the software architecture course in spring 2008, and investigates games types the students made, game components they used in their games projects, and architectural patterns and design pattern they used the most. In total, 15 projects are analyzed. The use frequency of game components are list as follows:

- 100% of the groups chose to make a 2D game. The complexity of a game can be significantly reduced by developing a 2D game rather than a 3D game. Many of the architectural challenges which are present in 3D game creation still apply in 2D games.
- 100% of the groups used fonts and text to some extent in their game.
- 93% of the projects utilize collision detection. Many groups use simple rectangle or circle collision detection, some use per-pixel collision maps, and a few use collision detection with advanced geometry.
- 93% of the games contained graphics in multiple layers.
- 87% used graphical user interfaces to some extent.
- 87% also used game state logic in their games. Most games had at least a initial state with a menu, and a running state.
- 87% used sound and two variants are relevant: background music, which enhances the atmosphere, and sound effects, which are triggered when some events occurs in the game.

- 40% of the projects used tiled graphics. Tiled graphics makes sense in many contexts, especially role-playing games, strategy games and platformers.
- 27% of the projects used frame-by-frame animations.
- 20% used persistent data storage in their game, such as saving/loading of progress, or a simple high-score.
- Only 7% used particle effects, which are used to achieve certain visual effects like fire, smoke, snow, and so forth.

Certain elements, which we take for granted in any game have been omitted from above lists, such as input. This is simply to avoid inflating the list with entries of 100% frequency. These omitted parts will not be neglected in the design of the framework.

Also, the patterns that students used in game projects are also useful references for the requirements and design for the Sheep framework. The Model-View-Controller (MVC) [5, 6] is by far the most popular architectural pattern, with 46% of the groups use it. Other favourites include Pipes and Filters (23%), Layered (11%), Strategy (8%) and Client-Server (8%), showing in Figure 1. And the Figure 2 shows that Observer, Abstract factory, State and Singleton pattern are the most popular design pattern. All these patterns are the key concept in practice of software architecture.

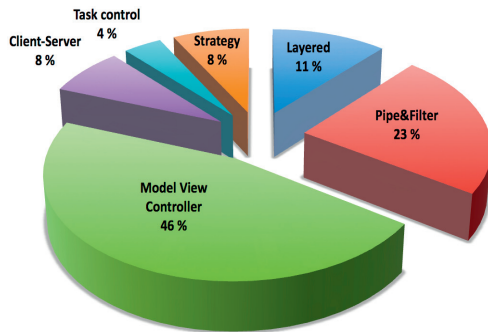


Figure 1. The distribution of chosen architectural patterns for game projects

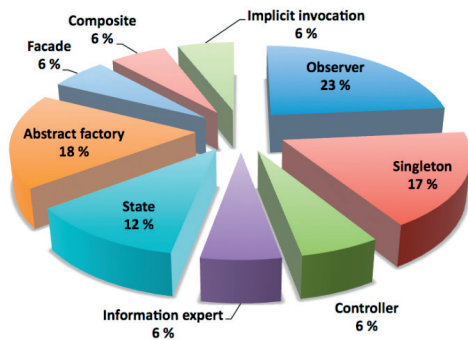


Figure 2. Distribution of usage of design patterns for game groups

D. Student expectations — requirements for sheep framework

As main criteria, the components with higher frequency will take a higher priority than the ones with lower frequency, but we must also take the usage frequency of patterns into account.

Under this point, we formed requirements as following for what the students expected to be able to use with the Sheep framework:

- 1) *Graphics*. The framework should be able to draw images, primitives and text on screen.
- 2) *Math*. The framework should be able to perform collision detection, and transformations on the graphics.
- 3) *Audio*. The framework should be able to play sound effects (non-streaming) and music (streaming).
- 4) *Timing*. The framework should be able to determine the timing of frames.
- 5) *Storage*. The framework should provide means to store persistent data.
- 6) *Networking*. It should be possible to transfer data over the network.

7) *Resource Management*. The framework should provide classes, which makes resource management as easy as possible.

8) *Input*. The framework should provide means of accessing input information.

Under these requirements, we also investigate what is available in the bare Android API. It is expected that some of the students' requirements will be satisfied fully by the bare Android API, such as networking.

IV. THE "SHEEP" FRAMEWORK

This section introduces the structure of the Sheep framework and the key components' value.

A. Design goals

From our previous experiences on XNA, students should not be involved in the programming too much time and cause less time on software architecture study, so the main goal of the Sheep framework is to allow the students to save time in game programming. In a nutshell, the two overall goals for all major components in the Sheep framework are:

- Simplify a common task in game development, so the students can spend more time on structure or course theory and less time on technical issues.
- Use known patterns to interact with client code, as to teach students these patterns, let them to perceive the course theory through using this framework.

According to these goals, we classify the components values in the Sheep framework as:

1) *Practical value* means that components which simplify common tasks without requiring the use of any particular patterns. The primary goal of these components is to allow faster development, and save time for student to focus on the course content.

2) *Academic value* means that components which require the use of certain patterns. The primary goal of these components is to illustrate the usefulness of a certain technique, let students could handle or use this pattern.

Not all components achieve both goals. Some may be of no direct academic value to the students, and may simply exist as a convenience, some components may be of great academic value, but may not be practical in a certain game genre or specific game design.

B. Structure of Sheep

According to our design goals, we could describe Sheep structure in two ways in which the Sheep framework makes the android platform more feasible for game development to learn the software architecture.

From aspect of time saving goal for game programming, the Sheep structure is organized as packages as follows:

- Sheep.audio provides components for loading and playback of sound.
- Sheep.collision contains collision detection and spatial partitioning components.

- Sheep.game assists in structuring the game logic (the model) of the game.
- Sheep.graphics contains components for loading images and fonts.
- Sheep.gui holds the graphical user interface system.
- Sheep.input contains the input devices, and the interfaces needed to subscribe to events.
- Sheep.math contains some math classes which aren't directly related to collision detection.
- Sheep.util is meant to contain miscellaneous components, but for now it only contains a singleton which keeps track of time between frames.

From aspect of encouraging or requiring the use of patterns, the components in the framework are:

- Sprites, which uses the Model-View-Controller.
- Game states, which uses the State pattern.
- Collision detection, which uses the Observer pattern and the Template pattern.
- Spatial partitioning, which uses the Visitor pattern.
- Graphical user interface system, which uses the Observer pattern and Chain of command.
- Other components without expected patterns.

All above pattern concepts are from the software architecture course, and students should master them during the process of using this framework.

C. Packages analysis

Three packages will be examples to explain the components values.

1) *Sheep.game package*. It provides components, which help organize the game model. Game State pattern is one of the main design concepts in this package. It keeps track of the high-level states of the game. Its main controller object contains methods for loading content, updating its internal state, drawing itself, and responding to input events. The practical value is that having a complete state system in place is beneficial because it allows relevant input events to be presented more clearly and quickly to the students. And its academic value is that State pattern is a well-known pattern, which allows an object partially changes its class at run-time. Specific game behaviour should be implemented via subclasses of the State class. Each state represents a different view of the game, when students use it in programming, they probably would understand it clearly.

2) *Sheep.collision package*. It provides functionality for detecting interactions between objects in the game world, and generates collision events, which may be subscribed by observers.

The practical value is that collision detection was used in most of the student projects, and getting the details of such collision systems to work right can be incredibly time consuming. When providing the students with a full collision detection system, they could use it directly to work efficiently.

The academic value is that two patterns will be visible from the perspective of the student, the Template pattern and Observer pattern. The Template pattern is in the Shape class, where the overall algorithm is fixed, but some sub-parts are modifiable by derived classes. The Observer pattern will be used for custom collision responses. As an example, perhaps a player should lose health when it is hit by another object, A "Lose Health Listener" could then be attached to listen for collision events occurring to the player object.

3) *Sheep.gui package*. It provides a graphic user interface system and can be used to create complex windowed menus or simple buttons. The practical value is that a few buttons were present to allow the user to start and quit the game, and also provide functional kit for the extensibility. The academic value is that Observer pattern is used to listen for events. The Chain of command pattern is used to control how input events are passed through the widget class hierarchy.

V. INTEGRATE SHEEP IN THE SOFTWARE ARCHITECTURE COURSE

This section presents how patterns work in Sheep framework, and how students interact with the framework based on the double stimulation. We choose two design cases to explain it.

A. First Task: Sheep and patterns

When the students start to use Sheep, they are inevitable to get into the code of Sheep. So the first task is to let the students become familiar with Sheep by list some patterns that they could find (or construct) in Sheep framework.

Here we give three exercise examples to explain the patterns that the Sheep framework used. Also, other patterns, such as *Template*, *Visitor*, *Singleton*, etc also can be found in Sheep, but not list here.

1) *Exercise 1: Model-View-Controller*. Student should find a Model View Controller design in the Sheep.

In the Sheep framework, the Sprite class acts as the superclass for all models. When a method on the Sprite itself is called to draw a sprite, this call is either redirected to the associated SpriteView, or ignored in case a SpriteView is not set. If the client code wishes to change the way that a Sprite is represented, for instance an animation instead of a static image, the client can simply create a new SpriteView subclass. The logic in the Sprite remains the same.

Figure 3 is one example to the exercise: a PlayerController listen for events on the keyboard. This controller can for example cause the Player to shoot bullets when a certain key is pressed.

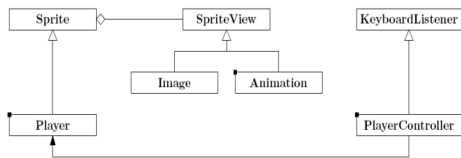


Figure 3. MVC pattern in Sheep framework

2) *Exercise 2: State*. Students should find an example of State pattern used in Sheep.

State pattern causes an object to appear as if it has changed its class. It can be used in the Sheep framework by adding subclasses of State to the instance of the Game class. Figure 4 is an example of State pattern used in Sheep.

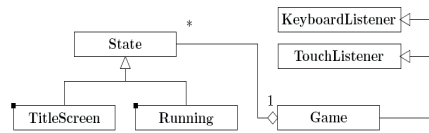


Figure 4. State pattern in Sheep framework

3) *Exercise 3: Observer*. Students should find an example of Observer pattern used in Sheep.

The observer pattern could be found under some conditions, such as, a) When listening to input devices, either via the keyboard or touch singletons or via a State; b) When listening to events from the collision detection system. Events are issued when Sprite objects collide; c) When listening to events from the graphical user interface system. Events are issued for various reasons, for instance when a button is pressed.

Figure 5 is an example in Sheep that PlayerController responds to events from the keyboard, touch or collision detection system.

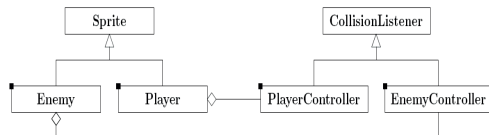


Figure 5. Observer pattern example in Sheep framework

B. Second Task: Patterns design and game implementation

We propose three exercises to implement small games by apply the patterns design through Sheep.

1) *Exercise 1: Moving Sprite*. The requirement is to make a simple game where a Sprite is controlled by the user. This could be done by subscribing to events issued by the Touch singleton.

The purpose of this task is to show how the observer pattern can be used to respond to input events in a way which is familiar to gamers.

In a solution, the students could create a subclass of Sprite, which listens to events directly; or simply instantiate Sprite and use the main state class as the controller; or they could create a separate controller class (Figure 6). There are also other possibilities of the solutions.

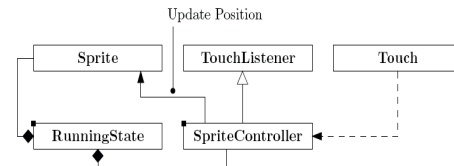


Figure 6. Possible solution to the exercise 1

2) *Exercise 2: Game States*. This task is to make a game with at least three States: a title screen, a main running state, and an in-game menu. The game can be as simple as Pong or Tic-Tac-Toe, as long as these states are present.

This exercise shows how an object may change its perceived class using the state pattern. A solution would consist of three (or more) subclasses of State, and some mechanism for transition between the states.

3) *Exercise 3: Racing Game*. This task is to make a racing game architecture with following characteristics:

a) It should be possible to change between two sets of graphics in the middle of the game; one with graphical sprites, and the other using primitive shapes only, for instance rectangles for cars and lines for the racetrack.

b) There should be more than one car; all cars except the player's car are controlled by the computer.

c) It should be possible to click on other cars and take control of them. In so doing, the computer should take control of your old car.

The racing game does not need good artificial intelligence or realistic car simulation, but these characteristics should be evident in the game.

This exercise shows how to decouple the visual representation, input handling, from the Model of Racecar. The solution here is to use the Model-View-Controller (Figure 7).

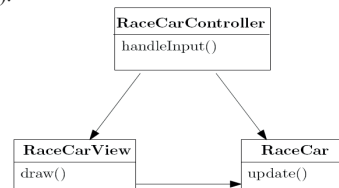


Figure 7. Solution to the exercise 3

VI. DISCUSSION

The software architecture course at NTNU is taught in an untraditional way, in that the students in addition to designing and evaluating their software architecture have to implement the architecture in a game project as well. The

main advantage with this approach is to let the students feel the “pain” of making their design decisions, as complicated and look-nice-on-paper architectures can be very difficult and time-consuming to implement.

From Sheep’s application view, the Sheep framework is a very useful tool to help the students with the transmission between the design and implementation by offering high-level components based on architecture and design patterns. The most difficult task for the students when implementing a software architecture is to decompose a high-level architecture into classes and design the interaction between these classes. The Sheep framework will make this transition easier, as the built-in architecture and design patterns is the first step in decomposing a high-level architecture. Due to this type of design in Sheep, the students could find appropriated available components to start with. The Sheep framework also enables the students to focus more on the architecture and less on issues related to the programming.

From a view of edutainment, the android platform was chosen and extended based on the Malone’s “What makes things fun to learn?” [16] and our own experiences [1]. We believe that it is useful to teach the students about design and architecture patterns in a practical way through the suitable game exercises proposed. This game domain is likely to motivate the students to put an extra effort when learning the patterns through various exercises. The students are motivated by learning how to program on the android platform as well as programming interesting games.

Game development for devices like android phones and iPhone/iPod Touch can also be motivating for the students from a business point of view, as development of games on these platforms can be low-cost and low risk. The result of a game project in a software architecture course might end up as a continuing hobby game project uploaded to android Market or AppStore to sell or as a student start-up game company.

From the pedagogical point of view, the design and application of the Sheep framework is also an example how to bridge pedagogy, technology and game ideas to enhance teaching in a reasonable way. During this double stimulation process, students seek to align their continuous interpretation of a task and tools. Also, the second stimulus is provided with series of tools that can be classified in horizontal-vertical orientations. From horizontal aspect, XNA and android are provided in parallel for the same task; from vertical aspect, XNA is used with other related tools, such as XNA club website for the help and sharing games, Zuneplayer for testing game demos, and PowerPoint for the final presentation. Corresponding, android is used with android Market, android phone and PowerPoint too.

We believe our analysis points to the necessity for further pedagogical and technological co-design to better facilitate awareness of game-based learning, better conduct the direction of how to design the knowledge construction process of involving individuals and small groups to

stimulate their initiative and creativity in game related activities. This indicates that future evaluation of using the Sheep framework for teaching the course is also beneficial, it reveals not only the efficiency of using the framework along with how much the students actually learn from game projects, but also the social relationships of learner-learner and learner-teacher. We also need to further investigate the relationship between games, tasks, and tools in technology-rich and collectively oriented knowledge construction in order to better understand and support the game-based learning.

VII. CONCLUSIONS

From our previous experiences in the software architecture course, we would like to offer a new choice for the double stimulation in this course. And we found that Google android is a suitable tool for the educational use. In this way, we extended the android platform mainly based on the requirements from previous students’ projects. Further, we have developed a game development platform called Sheep based on android. We also described how to integrate the game technology and software architecture learning in Sheep framework to explain one perspective of how technology perceives learning.

From the discussion, we found that there are various orientations to apply or extend a tool according to the previous experiences, context of students, local environment and technology and teaching aims. Based on these conditions, the game ideas, technology and learning should be integrated in a reasonable way to let the second stimulus match the first stimulus. This paper is an example from this idea that applied theoretical and empirical context to support the design process of game-based learning.

ACKNOWLEDGMENT

We would like to thank Anders Hartvoll Ruud for implementing Sheep framework and for his inputs to this paper.

REFERENCES

- [1] Wang, Alf Inge; Wu, Bian. “An Application of a Game Development Framework in Higher Education.” *International Journal of Computer Games Technology* 2009 ;Volume 2009.(2)
- [2] Google. “Android developers” <http://developer.android.com/index.html>, Retrieved September 22, 2009.
- [3] Apple. “iPhone Dev Center”, <http://developer.apple.com/iphone/>, Retrieved September 22, 2009.
- [4] Bian Wu, Alf Inge Wang, Jan Erik Strøm and Trond Blomholm Kvamme: “XQUEST used in software architecture Education”, *IEEE Consumer Electronics Society’s Games Innovation Conference* , August 25-28, 2009, London, UK.
- [5] Trygve M. H. Reenskaug, MVC, XEROX PARC, 1978-79. Accessed March 16th, 2009
- [6] Koen Witters, Game Architecture: Model-View-Controller, 2008. <http://dewitters.koonsolo.com/gamemvc.html>, Accessed March 16th, 2009.

- [7] Gamma et.al, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Co, 1994.
- [8] Bian Wu, Alf Inge Wang, Jan-Erik Strøm, Trond Blomholm Kvamme, "An Evaluation of Using a Game Development Framework in Higher Education," 22nd Conference on Software Engineering Education and Training, pp.41-44, 2009.
- [9] T. Blomholm Kvamme and J.-E. Strøm, "Evaluation and Extension of an XNA Game Library used in Software Architecture Projects", Master thesis at NTNU, June 2008.
- [10] PSP, "Sony PlayStation Portable" <http://www.us.playstation.com/psp>, Retrieved September 24, 2009
- [11] 3G "Definition" <http://en.wikipedia.org/wiki/3G>, Retrieved September 24, 2009
- [12] A. I. Wang, T. Stålhane, Using Post Mortem Analysis to Evaluate Software Architecture Student Projects, Conference on Software Engineering and Training 2005, 8 p.
- [13] XNA, "Microsoft XNA" <http://www.xna.com>, Retrieved October 5, 2009
- [14] Vygotsky, L. S.. Mind in society: The development of higher psychological processes. Cambridge, MA: Harvard University Press, 1978
- [15] Lund, A., & Rasmussen, I. (2008). The right tool for the wrong task? Match and mismatch between first and second stimulus in double stimulation. *International Journal of Computer-Supported Collaborative Learning*, 3(4), 387–412.
- [16] T. W. Malone, "What makes things fun to learn? Heuristics for designing instructional computer games", In SIGSMALL '80: Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, pages 162–169, New York, NY, USA, 1980. ACM Press.
- [17] Anders Hartvoll Ruud, "Designing a Game Development Framework for Teaching Software Architecture on the Android Platform", Master thesis at NTNU, June 2009.

Paper 9:

GDF5: Alf Inge Wang, Bian Wu, "Using Game Development to Teach Software Architecture", *International Journal of Computer Games Technology*, vol. 2011, Article ID 920873, 12 pages, 2011. ISSN: 1687-7047 EISSN: 1687-7055. DOI: 10.1155/2011/920873



Using Game Development to Teach Software Architecture

Alf Inge Wang

Norwegian University of Science and Technology
Sem Sælandsv. 7-9,
N-7491 Trondheim, Norway
+47 7359 4485
alfw@idi.ntnu.no

Bian Wu

Norwegian University of Science and Technology
Sem Sælandsv. 7-9
N-7491 Trondheim, Norway
+47 7359 1726
bian@idi.ntnu.no

ABSTRACT

This paper describes a case study of how a game project using the XNA Game Studio from Microsoft was implemented in a software architecture course. In this project, university students have to construct and design a type of software architecture, evaluate the architecture, implement an application based on the architecture, and test this implementation. In previous years, the domain of the software architecture project has been a robot controller for navigating a maze. *Robot controller* was chosen as the domain for the project, as there exist several papers and descriptions on reference architectures for managing mobile robots.

This paper describes the changes we had to make to introduce an XNA game development project to the software architecture course, and our experiences from running a software architecture project focusing on game development and XNA. The experiences described in this paper are based on feedback from the course staff, the project reports of the students, and a mandatory course evaluation. The evaluation shows among other things that the majority of the students preferred the game project to the robot project, that XNA was considered to be suitable platform for a software architecture project, that the students found it useful to learn XNA and C#, and that some students were carried away when developing the game in the software architecture project.

Key words

Software architecture, Game Development, Software Engineering Education, XNA.

1. INTRODUCTION

Games have been used in education for many years mainly focusing on teaching children in an interesting and motivating way. Research shows that integrating games within children's classroom can be beneficial for academic achievement, motivation, and classroom dynamics [1]. Teaching methods based on educational games are not only attractive to schoolchildren, but can also be beneficial for university students [2]. Research on game concepts and game development used in higher education is not unique, e.g. [3, 4, 5], but we believe there is an untapped potential that needs to be explored. By introducing games in higher education lecturers can access teaching aids that promote active students, provide alternative teaching methods to improve variation, enable social learning through multiplayer learning games, and motivate students to work harder on projects and exercises.

Games can mainly be integrated in higher education in three ways. *First*, traditional exercises can be replaced by games motivating the students to put extra effort in doing the exercises, and giving the course staff an opportunity to monitor how the students work with the exercises in real-time [6, 7]. *Second*, games can be used within a traditional classroom lecture to improve the participation and motivation of the students through knowledge-based multiplayer games played by the students and the teacher [8, 9]. *Third*, game development projects can be used in computer science (CS) or software engineering (SE) courses to learn specific CS or SE skills [10, 11]. This paper focuses on the latter, where a game development project was introduced in a course to teach CS and/or SE skills. The motivation for bringing game development into a CS or SE course is to utilize the students' fascination for games and game development to stimulate the students to put extra effort in the course project. Many students dream of making their own games, and game development projects allow the students to use their creativity in contrast to e.g. developing a more traditional web-based application. Game technologies and game user interfaces are now being more commonly used in serious applications [12, 13, 14], and the market for serious games is

growing. This makes it important for students to learn how to develop games even the students do not target to work in the game industry.

In this paper we describe a case study of how a game project was integrated with a software architecture course. From the perspective of a game developer, knowledge and skills about how to develop appropriate software architectures are becoming increasingly important. As games are growing bigger and becoming more complex, well-designed software architectures are needed to cope with variations in hardware configurations, functional modifications, and network real-time constraints [15]. From the perspective of a software architect, games are interesting due to the inherent characteristics of the domain including real-time constraints, changing and varying functionality, and user-friendliness. In addition, games are interesting from the perspective of a software architect, as there exist no real functional requirements that stem from the users. Typical user requirements for games are that the game should be fun to play, it should have enough variety, and it should be engaging.

The case study presented in this paper describes how a software architecture course was adapted to include a game development project. The paper describes the parts of the course and syllabus that had to be changed to make game development a natural part of the course, and how XNA was used as a game development platform in the course. Further, we present an evaluation of how the game development project was perceived by the students and the course staff compared to the robot project. The data of this evaluation is based on the students' responses to the final course evaluation, the feedback from the students during the project, and the student project reports.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 describes the software architecture course. Section 4 describes how the course was changed to adapt to the game project. Section 5 presents experiences we learned from running a game development project along with the robot development project in a software architecture course, and Section 6 concludes the paper.

2. RELATED WORK

This paper describes experiences from introducing an XNA game development project in a software architecture course. The main benefits from using XNA to teach software architecture is that the students get more motivated during the software development project. As far as we know, there are only few papers (presented here) that describe usage of XNA to teach CS or SE, and only few papers that contain case studies of games used in CS and SE education (also described here). In this section we will also briefly describe alternative game development frameworks to XNA that can be used in CS and SE education.

Youngblood describes how XNA game segments can be used to engage students in advanced CS education [16]. Game segments are developed solution packs providing the full code for a segment of a game with a clear element left for a student to implement. The paper describes how XNA was used in an artificial intelligence course where the students were asked to implement a chat bot, motion planning, adversarial search, neural networks and flocking. Finally the paper describes seven design principles for using game segments in CS education based on lessons learned. The approach described by Youngblood could also be used in a software architecture course, where the students can put together parts of the game (game segments) based on their designed architecture. However, this approach is very limiting as the architectural freedom will be very restricted and the students will not get the chance to design their own software architecture of their own game.

El-Nasr and Smith describe how modifying or modding existing games can be used to learn CS, mathematics, physics and ascetic principles [10]. The paper describes how modding of the WarCraft III engine was used to teach high school students a class on game design and programming. Further, they describe experiences from teaching university students a more advanced class on game design and programming using the Unreal Tournament 2003 engine. Finally, they present observations from student projects that involve modding of game engines. Although the paper claims to teach students other things than pure game design and programming, the focus is on game development in contrast to CS or SE. Modding existing games is not very useful in a software architecture course, as the focus of the course is the structure of software components and not game content nor game engine scripts.

Sweedyk and Keller describe how they have introduced game development in an introductory SE course [17]. The students learn principles, practices and patterns in software development and design through three projects. In the *first* project, the students develop a campus life 2D arcade game over four weeks with the educational focus on gaining familiarity with UML tools, learn and use a variety of development tools and gain understanding of game architecture and the game loop. In the *second* project, the students should build a one-hole miniature golf game over five weeks with the educational focus on learning and practicing evolutionary design, prototyping and refactoring, usage of UML design tools, usage of work management tools and design and implementation of a test plan. In the *third* and final project, the students can develop a game of their own choice over five weeks with educational focus on reinforcing the practices and principles learned in two previous projects, learn to apply design patterns and practice management of complex software projects. The students' response to this SE course has according to the authors been extremely positive. They argue that game projects allow them to better achieve the learning objectives in the SE course. Their main concern is related to gender, as women are less motivated to learn SE through game development projects. The main difference with Sweedyk and Keller's approach and ours is that they have introduced three projects instead of one, and the SE focus is different. For our purpose, more than one project would take away the focus on the software architectural educational goals and miss the opportunity to follow the evolution of the software architecture through a complete development cycle.

Kajal and Calypool describe another SE course where a game development project was used to engage the students and make the course more fun [18]. In this course, the students worked with one game project where the students had to go through all the phases in a software development process. The preliminary results of comparing the game-based SE course with a traditional SE course showed that the game version had higher enrollment, resulted in average higher grades, a higher distribution of A grades, and had a lower number of dropouts. The feedback from the students was also very positive. The approach described in this paper is very similar to our approach. The main difference is that in our course the students carry out the various phases in a software process from a software architecture perspective focusing on quality attributes, software architecture design and software architecture evaluation.

Volk describes how a game engineering course was integrated into a CS curriculum [19] motivated by the fact that game development projects are getting more and more complex and have to deal with complex CS and SE issues. The experiences from running this course showed that it was a good idea handle the game engineering course more in a form of a real project, that the students were very engaged in the course and the project, that the lack of multidisciplinary teams did not hinder the projects, that the transition from pre-production to production was difficult (extracting the requirements), and that some student teams were overambitious for what they wanted to achieve in their project. In our software architecture course we experienced some of the same issues as described in this paper, namely difficult extraction of requirements and overambitious teams.

Linhoff and Settle describe a game development course where the XNA platform was used to allow the students gain experience in all aspects of console game creation [20]. The course focuses on creating of fonts, icons, 3D models, camera and object animation paths, skeletal animations, sounds, scripts and other supporting content to the XBOX 360 game platform. In addition, the students are required to edit the source code of a game to change variables, and copy-and-paste code. The student response to the course was positive. The results also showed that students with programming background did better in the class. The students did not learn any CS or SE skills.

Zhu, Wang and Tan describe how games can be introduced in SE courses to teach typical SE skills [21]. The paper describes how the two games SimSE and MO-SEProcess were used to give students an opportunity to practice SE through simulations to learn the complex cause and effect relationships underlying the process of SE. MO-SEProcess is a multiplayer online SE process game based on the SimSE in 3D implemented in Second Life. In this game, the players should collaborate with other developers to develop a system by giving out tasks and following up tasks. Although the models and simulations in SimSE are much more extensive than the ones in MO-SEProcess, the usage of Second Life bring some advantages such as better support for group sharing and collaboration, and the possibility to create interactive learning experiences that would be hard to duplicate in real life. This approach is very different from ours and does not fit with our educational goals.

Rankin and Gooch describe a study on how game design project impact on students' interest in CS [22]. In a Computer Science Survey course, the students are given the task to apply SE principles in the context of game design. The pre and post survey results reveals that game design project can have both a positive and a negative impact on students' attitudes about enrollment in a game design course, pursuit of a CS degree, further development of programming skills and enrollment in additional CS courses.

Leutenegger and Edgington argue that the course assignment and example content is more important than whether a introductory programming course should focus on procedural vs. object-oriented approach [23]. Their paper describes an introductory programming course focusing on game programming. The results showed that the students improved their understanding basic programming concepts, and the students were satisfied with the course.

Coller and Scott describe an interesting approach for teaching mechanical engineering through game programming [24]. In a numerical methods course, the students are asked to program the behavior of a car in the Torcs open racing car simulator. The students must use numerical methods to program acceleration, steering, gearshifts, and breaking. A comparison with a traditional version of the course showed that for the game-based course the students on average spent roughly twice as much time on the course, and that the students achieved deeper learning as the students were more interested, more engaged and invested more in learning the material.

We have found the XNA was a perfect fit for our game project as it provides a high-level API, the framework is mature and well supported, and the students are motivated by the fact that XNA makes it easy to develop for XBOX 360. There are also other alternative game frameworks that can be used. The *Labyrinth* [25] is implemented in Java and is a flexible and easy-to-use computer game framework. The framework enables instructors to expose students to very specific aspects of CS courses. The framework is a finished game in the Pac-Man genre, highly modular, and it lets the students change different aspects of the game. The *JIG (Java Instructional Gaming)* project [26] has the aims to build a Java Instructional Game Engine suitable for a wide variety of students at all levels in the curriculum, to create a set of educational resources to support the use of the game engine at small, resource-limited, schools, and to develop a community of educators that use and help improve these resources. The *DXFramework* [27] is a game engine written in C++ targeted specifically for 2D games to be used in game programming education. The *SAGE* [28] game engine is also written in C++ and is targeted for game programming educational use focusing on 3D games. *GEDI* [29] game engine is another alternative for 2D games in C++ designed with game programming educational use in mind. For business teaching, *Arena3D* [30] is a game visualization environment with animated 3D representations of the work environments, simulation of patients queuing at the front desk, and interacts with the staff. IBM has also produced a business game called *INNOV8* [31], which is "an interactive, 3D business simulator designed to teach the fundamentals of business process management and bridge the gap in understanding between business leaders and IT teams in an organization".

Of the related work described in this section, the work by Kajal and Calypool is closest to the work described in this paper. The main difference with our approach is that we focus on software architecture methods and processes and not only software engineering topics in general. The students' responses to our course are very similar to the studies described in this section, characterized by higher motivation, higher enrollment and more effort spent on the course.

3. SOFTWARE ARCHITECTURE COURSE

The software architecture course is a post-graduate course offered to CS and SE students (not mandatory) at the Norwegian University of Science and Technology (NTNU). The course is taught every spring, its workload is 25% of one semester, and about 70-80 students attend the course every spring. The students in the course are mostly of Norwegian students (about 80%), but there are also 20% foreign students mostly from EU-countries. There are about 10% female students. The textbook used in this course is the "Software Architecture in Practice, Second Edition", by Bass, Clements and Kazman [32]. Additional papers are used to cover topics that are not

sufficiently covered by the book such as design patterns, software architecture documentation standards, view models, and post-mortem analysis [33, 34, 35, 36, 37].

The education goal of the course is:

“The students should be able to define and explain central concepts in software architecture literature, and be able to use and describe design/architectural patterns, methods to design software architectures, methods/techniques to achieve software qualities, methods to document software architecture and methods to evaluate software architecture.”

The course is taught in three main ways:

- 1) Ordinary lectures given in English
- 2) Invited guest-lectures from the software industry
- 3) A software development project with emphasis on software architecture

The software architecture course at NTNU (course code TDT4240) is taught in a different way than at most other universities, as the students also have to implement their designed architecture in a project. The motivation for doing so is to make the students understand the relationship between the architecture and the implementation, and to be able to perform a real evaluation of whether the architecture and the resulting implementation fulfill the quality requirements specified for the application. The architecture project in the course has similarities with projects in software engineering courses, but everything in the project is carried out from a software architecture perspective. Throughout the project, the students have to use software architecture techniques, methods, and tools to succeed according to the specified project requirements and the document templates. The development process in the project will also be affected by the focus on software architecture, as the development view of the architecture will specify how the teams should be organized and how they should work. The main disadvantage of this approach is that the students get less time dedicated to do the architectural design, as they have to spend time on the implementation. The main advantage is that the students are learning software architecture through doing a whole project where they can see the results of their architectural design as a product.

The TDT4240 software architecture course has been rated as one of the most useful and practical courses offered at the Dept. of Computer and Information Science in surveys conducted among ex-students now working in the IT industry. The course staff has also seen the benefits of making the students implement the architecture, as the students have to be aware of the developing costs of fancy and complicated architectural designs.

30% of the grade awarded to the software architecture course relate to the evaluation of the software architecture project all students have to do, while 70% is awarded for the results of a written examination. The goal of the project is for the students to apply the methods and theory in the course to design and fully document a software architecture, to evaluate the architecture and the architectural approaches (tactics), to implement an application according to the architecture, to test the implementation related to the functional and quality requirements, and to evaluate how the architectural choices affected the quality of the application. The main emphasis when grading the projects is on the quality of the software architecture itself, but the implementation should also reflect the architecture and the architectural choices.

The project consists of the following phases:

- 1) *Commercial Off-The-Shelf (COTS)*: Learn the development platform/framework to be used in the project by developing some simple test applications.
- 2) *Design pattern*: Learn how to utilize design patterns by making changes in two architectural variants of an existing system designed with and without design patterns.
- 3) *Requirements and architecture*: Describe the functional and the quality requirements, describe the architectural drivers, and design and document the software architecture of the application in the project including several view points and views, stakeholders, stakeholder concerns, architectural rationale, etc.

- 4) *Architecture evaluation*: Use the Architecture Trade-off Analysis Method (ATAM) [32, 38, 39] to evaluate the software architecture in regards to the quality requirements.
- 5) *Implementation*: Do detailed design and implement the application based on the designed architecture and based on the results from the evaluation. Test the application against both functional and quality requirements specified in phase 3, evaluate how well the architecture helped to meet the requirements, and evaluate the relationship between the software architecture and the implementation.
- 6) *Project evaluation*: Evaluate the project using a Post-Mortem Analysis (PMA) method [34]. In this phase, the students will elicit and analyze the successes and problems during the project.

In the two first phases of the project, the students work on their own or in pairs. For the phases 3-6, the students work in self-composed teams of four students. The students spend most time in the implementation phase (6 weeks), and they are also encouraged start the implementation in earlier phases to test their architectural choices (incremental development). During the implementation phase, the students continually extend, refine and evolve the software architecture through several increments.

In previous years, the goal of the project has been to develop a robot controller for a robot simulator in Java with emphasis on an assigned quality attribute such as availability, performance, modifiability or testability. The functional aim of this project was to develop a robot controller that moves a robot in a maze collecting balls and bringing them to a light source. Robot controller was chosen as a case for the software architecture project, as the problem of software architecture is well defined within this domain. For the robot controller domain there exist several examples of software architecture patterns or reference architectures that can be applied, such as Control loop [40], Elfes [41], Task Control [42], CODGER [43], Subsumption [44], and NASREM [45].

4. HOW THE COURSE WAS CHANGED

This section presents the changes we made to the course to integrate an XNA game development project with the software architecture course.

4.1 Course Preparations

Half a year before we integrated the game development project with the software architecture course, we initiated a master research project, named XQUEST, to explore how XNA could be used and integrated with the course. The goal of this project was to answer the following questions:

- Q1) How well is the XNA framework suited for teaching students software architecture?
- Q2) What resources must be in place to quickly get up to speed developing games using the XNA framework?
- Q3) How should XNA be introduced to the students?

The first question (Q1) was decomposed into three sub-questions. *First*, the XQUEST project investigated which software/game components were required to allow the students to stay focused on the software architecture during the their project. This work resulted in an implementation of a game library named XQUEST framework [46] to provide a high-level sprite animation framework, a game object management framework, and some additional helper classes (audio, input, text out and texture store) on top of XNA to ease the development. *Second*, the XQUEST project investigated how difficult it was for the students that only knew Java to learn the C# programming language. They found that it took about three days to learn the most essential features of C# for a postgraduate student with average Java skills. *Third*, the XQUEST project investigated what limitations or restrictions that should be put on a game development project in a software architecture course. The conclusion was to limit the projects to 2D games, and only to focus on the two quality attributes modifiability and testability. 2D games were preferred to 3D games, as the students should not spend too much time on 3D graphics and focus on the structure of the software. We also considered the quality attributes performance and usability for the

project. *Performance* was dropped because the XNA framework handles most of the performance issues and it is hard to make architectural design that actually will affect this quality attribute. Further, *usability* was dropped because this quality attribute is rather hard to measure without extensive usability tests (not within the scope of the software architecture course).

The necessary resources to quickly develop games in XNA (Q2) was found to be C# and XNA tutorials, XNA examples, XNA documentation, libraries of graphical art (sprites, tiles, etc.), a high-level API on top of XNA, and making course staff available that could answer specific XNA or C# questions. Although XNA provides a high-level API, the XQUEST framework was found necessary to provide an even higher API to help the students get going faster.

The conclusion of final question (Q3) was that XNA should be exposed to the students through a mixture of lectures, an XNA resource webpage and continues technical support through the semester. It was found to be very important to give an introductory lecture in XNA to learn the tools, environments and the core concepts of XNA, and give an overview of the differences between Java and C#.

4.2 Changes to the Syllabus

It was rather difficult to change the syllabus of the software architecture course to include more literature about software architecture in games. Good books and papers that give an in-depth insight into game architectures and game architecture patterns are to our knowledge non-existent. There are several papers that describe architectures of specific games such as [47, 48] or books that give a brief overview of game architecture [49, 50], but none that looks at the typical abstractions (architectural patterns) you can observe in game software development. The syllabus ended up with including some chapter from the book “Game Architecture and Design” [50] to describe the initial steps of creating a game architecture, and two self-composed sets of slides on 1) *software architecture and games*, and 2) *architectural patterns and games*. The *former* was a one hour lecture on motivation software architecture design in games [15], architectural drivers within game development [51], challenges related to software architecture in games [52], and the main components of game architectures [53]. The *latter* was a one-hour lecture describing architectural patterns that are common and useful for games, such as model-view controller, pipe-and-filter, layered architecture, and hierarchical task trees.

4.3 Changes of the Project

The course staff decided to let the student teams themselves choose between the robot and the game project. This meant that the main structure of the project had to remain the same, and that we had to make two variants of the project. For the robot project the students had fixed requirements, while for the game project the students should define their own requirements (design their own game). However, the documents to be delivered were the same for both types of projects based on the same templates, and the development process was also to be the same.

To evaluate and grade the software architecture project, we posted some project evaluation criteria in the beginning of the semester that stated *how* the project should be documented, *what* should be *documented*, *what* should be *delivered* (such as documents, source code, complied code etc.), *completeness* of robot controller or game, and an implementation that *reflects* the architecture. The main difference between the game and the robot versions of the evaluation criteria was how the implementation was to be evaluated. For the XNA projects we required the game to have a certain level of complexity (at least five classes organized in a structure), the game should be easy to install and run. For a top grade (A), the game should be impressive in some way (fun, nice, creative, or original). For the robot controller, the implementation should similarly have a certain level of complexity, but it had to adhere to the given functional requirements. For a top grade (A), the robot should be able to solve the task efficiently.

Another thing we had to change was the quality attributes the various teams should focus on during the project. The teams that chose the robot projects were assigned to focus on safety of the robot (not get stuck in the maze), modifiability (easiness of changing the robot controller software), and testability (easiness of testing the robot

software). For the game projects we ended up with modifiability (easiness of changing the game software) and testability (easiness of testing the game software).

The main change of the project assignments was to add XNA game variant of the COTS intro exercise (phase 1, see Section 2). The COTS intro exercise for the robot controller asked the students to do simple navigation and make to robot pick up balls. In the XNA game variant of this exercise, the students were asked to perform the following four tasks:

- 1) Draw a helicopter sprite on the screen and make it move around on its own (computer controlled);
- 2) Move around the helicopter sprite from previous task using the keyboard or a game controller, change the size of the sprite, rotate the sprite, and write the position of the sprite on the screen;
- 3) Animate the helicopter sprite using several frames and do sprite collision with other sprites; and
- 4) Create the classical Pong game (2D from-above tennis game).

4.4 Changes of the Staff and the Schedule

The main change to staffing was that two last year master students were hired to give technical support for student during the project (both robot and XNA). The main tasks of the technical support staff were to give lectures on the COTS, to be available for technical questions on email, to be available two hours a week in a lecture halls for questions, and to evaluate the implementation of the final project delivery (testing the games and the robots).

The main changes that were made to the course schedule were:

- Changed the motivation of the software architecture project to also include the game project. An extra bonus for the teams that chose the game project was that they could register for the Norwegian Game Awards competition [54]. This is an open national game developer competition for the all universities and colleges in Norway.
- Added an extra two-hour COTS introduction lecture to give an introduction to the robot simulator, C#, and XNA.
- Added an extra two-hour technical support lecture on COTS every week (both for robot and XNA).
- Changed a one-hour lecture on architectural patterns to also include architectural patterns on games.
- Added a one-hour lecture on software architecture in video games.
- Changed the project workshop where selected teams presented their work to give room to show more demos (mostly games and some demos of robots).

5. EXPERIENCES AND RESULTS

This section presents experiences and results from running the course. The experiences presented here are collected from course staff interviews and notes, final course evaluation, the project reports, student feedback by email, and feedback during lectures. The students doing game development projects used version 2.0 of XNA Game Studio (the most recent version at that time).

5.1 Staff Experiences

In the first weeks of the semester we were faced with a problem introduced by allowing students to choose between a robot and a game project. In previous years, the students did not have to make any decisions (e.g. forming teams etc.) regarding the project before week 7, as this was the start of the main project (phase 3, see Section 3). By introducing two variants of the project, the students had to choose in week 3 if they were going to do the robot or the game project (before they had formed the teams) due to the two variants of the COTS exercise. As a result, some students ended up doing an exercise on the robot and later did the game project and vice versa.

The course staff was excited to see the distribution the number of student that chose the robot vs. the game project. When we introduced the project to the students in the beginning of the semester, we admitted that this was the first time running a game project in the software architecture course, and that the robot version of the project was better supported through previous experience, examples, literature, and software architecture patterns. The result was that 6 teams chose the robot project while 16 teams chose the game project (see the distribution in Figure 1). The percentage of teams choosing the game project was much higher than we expected (almost 3 out of 4). The results show that students are attracted to games and it indicates that games can be a motivation for choosing a course or for putting extra effort into projects.

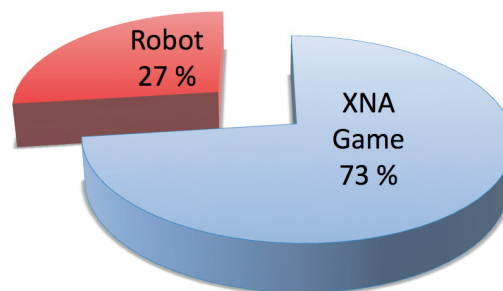


Figure 1 Distribution of Project Selection

During the semester, the students receive feedback on their part-deliveries from the course staff. The most notably difference between the part-deliveries made by robot and game project teams were found in phase 3 of the project (Requirements and Architecture, see Section 3). For many game project teams, it was hard to create proper requirements documentation. This was not unexpected, as these teams first had to specify some gameplay element and then translate these into functional requirements. The course staff suspected that it also would be harder to specify the software architecture in the game projects due to less available literature and architectural patterns. This was, however, not the case. For the final delivery of the project, there was no noticeable difference in the quality of documentation, requirements, design, architecture and implementation between the two variants (robot vs. game). The implementation of some teams (both robot and game) suffered for being too ambitious resulting in unfinished implementations. For teams implementing a robot controller, the main challenge was to implement an intelligent maze navigator. For teams implementing a game, the main challenge was to implement advanced game logic.

The educational approach for our software architecture course is to force the students to use the theory described in the textbook during the project by applying the methods and theoretical framework described. To make this work, the course schedule is heavy on theoretical presentations in the first part of the semester. At the same time, the students have to learn the COTS through exercises (phase 1 and 2). Phase 3 is really the start of the project, where the students will document the requirements and do the architectural design. Although the students at this stage should know the COTS and all the software architectural theory required to describe the requirements and to the design, we discovered that the students were lacking both knowledge of the COTS and the theory. This was true for both types of projects and we did not discovery any differences between robot and game teams. Based on feedback from the course staff and from another student team evaluating the project using ATAM, the software architectures improved significantly in terms of quality and quantity in the implementation phase of the project. The teams discovered problems with their architectural design mainly due to wrong assumptions about the COTS. Both XNA and Khepera put constrains on how to design the architecture, and the students discovered this through trial and error. The XNA teams struggled to make this work due to the complexity of the COTS, while for the Khepera simulator the main problem was lack of documentation. The students learned most during the implementation phase of the project, as they in this phase had to put everything together, reflect on their choices,

make changes to make it work, and do the final documentation including updating documentation from previous phases. The course staff also noticed that the students worked a lot the last couple of weeks to be able to finish in time, and put everything together.

One noticeable difference for the course staff after introducing the game project was that the software architecture workshop, where a selected number of teams presented their work, was much more interesting and exciting. In previous years, these workshops have not been very interesting, since most all the students had worked with the same domain (robot). The game projects brought new life to the workshop and it was very interesting to learn from creative game projects.

5.2 The Games Developed

In total, 16 different 2D games were developed. The type of games varied in several dimensions like number of players, game genre, network support, real-time vs. turn-based games, etc. The distribution of the game genres implemented by the students is shown in Figure 2. From the figure we can see that most students chose to implement a variant of a shooter game including a bee-shooter, space shooters, balloon-shooter, tank-shooter etc. The other major game genre was the strategy games that included trading games, and turn-based worm clones.

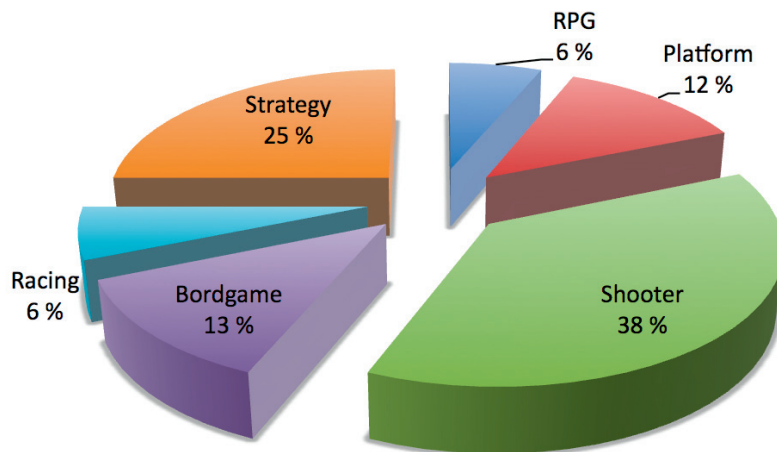


Figure 2 Distribution of Game Genres in Student Projects

The student projects also varied in support for multiplayer and network, and usage of the XQUEST-framework as shown in Figure 3. More than 56% of the games developed supported multiplayer, 31% were turn-based, and only two games supported playing over network. About 44% of the games used the XQUEST framework that was developed for this course to simplify the development in XNA.

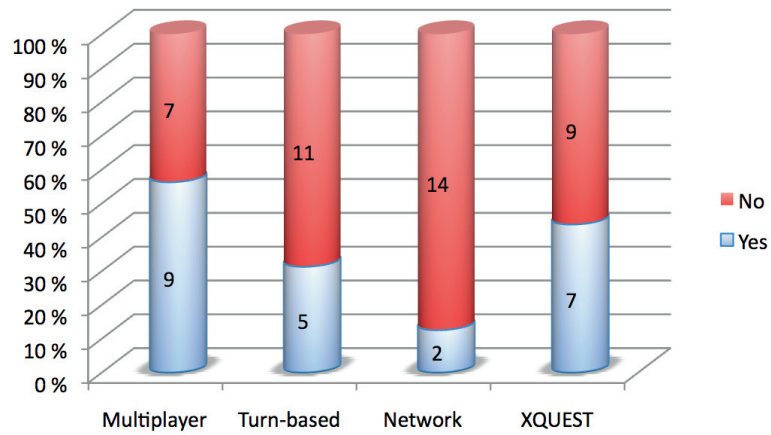


Figure 3 Distribution of Game Characteristics

None of the games developed were groundbreaking in terms of gameplay or graphics, but several of the games had new twists in gameplay or graphics (like including the two most known buildings in the local city – Trondheim). The most novel game was a two-player split screen death-match shooting game, where two players were navigating in an environment that was hand-drawn using colored pencils. One of the levels in the game was actually architectural drawings of the implementation of the game itself. Figure 4 shows a screenshot this game named BlueRose.

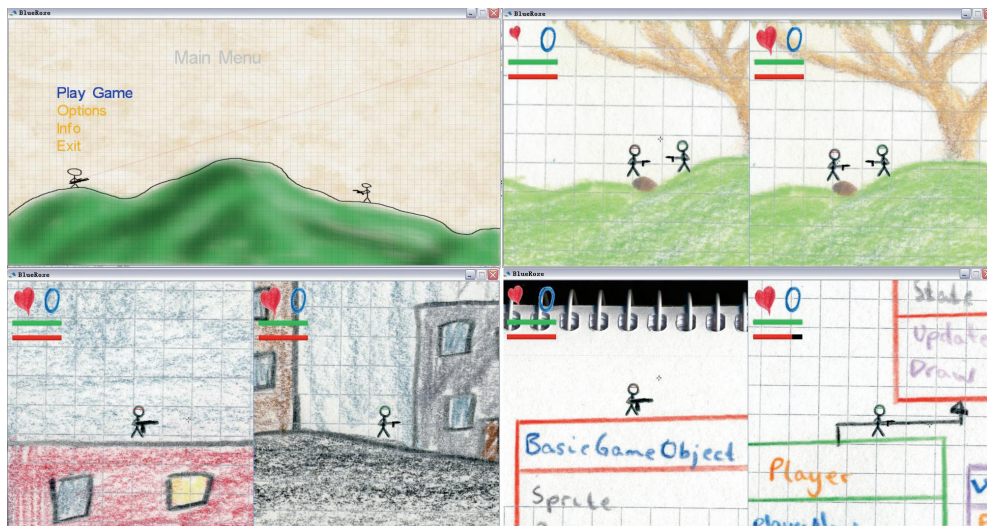


Figure 4 Screenshots from the BlueRose XNA game

Some of the teams have continued to develop their games after the course ended.

If we look further into differences between how the robot teams and the game teams in terms of the implementation, we found that the projects varied in complexity and size. Although the APIs of XNA and Khepera framework is about at the same abstraction level, the game projects on average had more complex architectures. The architecture of game teams on average consisted of 12 classes compared to 9 for robot teams. We also noticed that the robot teams had a standard deviation of about 3 classes compared to 4 classes for game teams. We found the same tendency for lines of code where robot teams wrote in average 1800 lines of code (without comments), while game teams wrote 3400 (about 90% more lines of code). Another finding was that there was much more variation in number of lines code in game teams compared to robot teams. For robot teams, the most productive team wrote about 2500 lines of code (less than the average for game teams), and the least productive 850 lines of code. For game teams, however, the most productive team wrote about 12000 lines of code and the least productive about 800 lines of code. From analyzing the code, we found that the game teams that produced most lines of code really got carried away with programming the game with less attention to the software architecture. We also compared the final grade of students doing game projects vs. students doing robot projects and did not find any significant difference in the final grade. However, we noticed a tendency that students from game teams got a better grade on the project compared to the final written examination, and the students from the robot teams the opposite. An extensive analysis of the differences between the two projects is described in [55].

5.3 Lessons Learned from the Students

This section describes experiences described in the students' lessons learned section of the teams' final reports.

A striking difference between students that did a game vs. students that did a robot project was how they experienced using the COTS. None of the robot students said anything positive about the Khepera framework. The students that did the game projects described XNA and C# to be easy to learn and work with, that the tools were user-friendly and helpful, that the XNA framework provided the most important functionality including the game loop, and that the game project was very interesting. The students also wrote that it was very valuable to learn XNA and C#, and that XNA and the XQUEST library let them focus on the logic of the video game thus saving a lot of time.

There were several comments both from robot and game teams about the negative experiences from using the chosen COTS. For the students working with the robot simulator the main problems were related to random and unpredictable behavior of the robot, that the robot simulator performed differently on different PCs, that it was difficult to implement the designed architecture using the API, and that the implementation forced the students to think too much on AI issues instead of software architecture. The random and unpredictable behavior of the robot simulator is a built-in feature to simulate unpredictable sensors in the real worlds. This issue caused a lot of frustration among the students. The different performance of the robot simulator on different PCs is due to problems of real-time execution in Java and real-time performance on different virtual machines. The negative experiences from using XNA was insufficient audio support (only support uncompressed audio files), no support for network testing of two instances on the same machine, limitations of the provided network API in XNA, and that more knowledge of the XNA framework was required to do a good architectural design.

Another topic that was covered by many teams in the lessons learned was their experience with the software architecture domain. Both robot and game teams found that they had learned a lot about software architecture through the design and implementation of the software architecture. One game team said that especially the XQUEST put some major restrictions on the architecture as it was tightly coupled to XNA. This made it difficult to implement a layered architectural pattern. Their conclusion was that the team should have spent more time in the beginning discovering the architectural limitations of the COTS. Another XNA team found that the COTS enabled a proper balance between the game functionality and the software architecture, which resulted in a smooth implementation. Finally, an XNA team described that they did not do an attempt to separate game logic and graphics beyond what was done in XNA, and that this was a big mistake that cause a lot of problems later in the project. For the robot teams, one team said that they used an inappropriate amount of time on the implementation and that the software architecture was therefor put in the background. One robot team discovered

that having a well-planned architecture before starting to implement made it a lot easier to divide the work and make changes during the project. Another robot team explained that they in the beginning only had considered the top-level architecture without examining the architecture of the major modules, which caused a lot of problem. Finally, yet another a robot team admitted that they should had thought more about splitting different classes into packages, as they ended up with code that was hard to modify and manage.

The overall lessons from the students doing a robot project were a mixture of positive and negative issues. The robot simulator itself frustrated the students and they had nothing positive to say about the COTS. Many students found the robot simulation domain to be fascinating, but they thought it was too difficult to implement the logic of the robot. However, the students had many positive comments about learning software architecture through such a project and designing a software architecture for a robot controller. They also mentioned that they had many reference architectural patterns they could use as a starting point. The hard part was implementing the architecture and the logic for the robot controller.

The overall lessons learned from the students doing an XNA game project were very positive about introducing a game project in a software architecture course. Some students felt that learning C# and XNA in addition to the syllabus was a bit too much, but generally most students said that to learn XNA and C# did not take much time. Some students said that the XNA architecture put major restrictions on their architecture. This is of course true, but this is also the case in most commercial software development projects, as they often use some kind of framework that the architecture must adhere to. The main challenges of using XNA in the software architecture project was to spend enough time learning the framework before designing the architecture, and doing the design and implementation. The identified issue of lacking support for other audio format than wav was resolved in XNA Game Studio 3.0. From the reports we could also see that our own XNA extension (XQUEST) limited the choices of architecture more than only using XNA. The main benefit of using XQUEST was a simpler interface to some of the most useful game functionality.

5.4 Student Evaluation Feedback

After completing the project, all students had to fill in a final course evaluation and write responses to three questions: What has been good about the course, what has been not so good, and what would you like to change to next year?

The responses regarding *what had been good* about the course can be categorized into main areas the project, learning, practical work, and group dynamics. Both students from robot and game teams stated that the project had been good, but students from game teams were overall happier with the project and described it to be cool, interesting, fun and motivating. Also both categories of students described that they learned a lot from the project in that they got to try out the theory from the lectures in practice. They also gave concrete example of theory that they got to try out in the project such as architectural and design patterns and how the software architecture is represented in code. Many students from game teams also wrote that the project was a fun way of learning software architecture and that it was useful to learn about the interplay of game and architectural approaches. Regarding the practical work, students from game teams mentioned that it was really useful to learn C# as it is commonly used in industry and that it was easy to learn because of its similarities with Java. Both robot and game students gave positive comments about the fact that the course forced the students to do practical work. Finally, it was mentioned that it was useful to learn from other teams through the final workshop. The responses from the students taking the course were overall very positive. The feedback from game team students was generally more positive than the feedback from the robot controller projects. Typical positive feedback we received from students doing a game project was that they felt they learned a lot from the game project, that they liked the practical approach of the project and having to learn C#, and the interaction between the teams (both ATAM and the project workshop). The students doing a robot project were pleased with learning software architecture through practical work, and thought it was very interesting to learn about software architecture in general.

The responses regarding *what had been not so good* about the course mainly concerned the COTS. Both students from robot and game teams complained about the lack of technical support during the project and sufficient

introductory lecture on the COTS in the beginning of the course. Further, both categories of students complained that the COTS took away focus from software architecture in the course. Few students on game teams complained that learning C# took so much time that they did not have enough time to study software architecture. Some other students on game teams said that the focus on the game itself keep them from focusing on the software architecture, and that the game domain limits the choice of architecture too much. Students on robot teams complained that the difficulty of implementing the robot controller took the focus away from architectural design, and that the workload of the project was way too high. The main negative feedback from students doing game projects focused on the lack of XNA technical support during the project, and that some student felt that there was too much focus on C#, XNA and games and too little on software architecture. The students doing a robot project also complained about not sufficient technical assistance, and that the robot simulator and the robot domain were very difficult to master.

On the final question in the course evaluation, *what would you have changed* for next year's course; we received various course improvement suggestions. Game team students suggested to allocate more time to develop the game, to make the project count 50% of the grade, to give a better C# introduction, to provide better technical support, and to put more restrictions on game-type to ensure that the teams choose games suited for the course. The robot team students suggested to either give better information on how to program the robot or drop the robot project all together, provide better technical support during the project, and split the project into several smaller exercises. One robot team student said that he rather would choose the game project if he could start all over again. The suggestions to improve the course were mainly according to the negative feedback namely to improve teaching and technical support related to the COTS (XNA and robot simulator), and to adjust the workload of the project.

6. CONCLUSION

In this paper we have described how we changed a software architecture course to include a game development project. The main motivation for introducing such a project was to motivate the students to put extra effort into the project and motivating for higher course enrollment. Some parts of the syllabus were changed to include game development as a natural part of the software architecture course. A challenge we discovered was to find appropriate literature on design of software architecture for the game domain, which we are still looking for. It is not very hard to motivate for why game developers can benefit from learning more about software architectures as games are becoming increasingly more complex (especially massively multiplayer online games). From a software architecture perspective, games are interesting since they introduce relevant challenges such as dealing with continues changes of functional requirements (modifiability), and hard real-time requirements both for hardware and network.

Our experience from running a game development project in a software architecture course is very positive. The course staff noticed an increasing interest and motivation for the project in the course. From the course evaluation, we also notice that students choosing the game project were more positive towards the project compared to those who chose the robot project. Robot team students complained more about the project while game team students generally expressed that the project was fun and engaging. Game development projects are also very positive for the group dynamics, as other that CS and SE skills are required (e.g., creative and artistic skills). The main negative effect of introducing a game development project was that some teams focus more on developing the game than on the software architecture of the game. This effect was not a major issue, as most teams did a good job of designing the architecture and then implementing it. There will always be some students that do not like to do a project on games. When we looked at the demographics to see if there were any various in choosing game projects, we only found minor variations between male (73%) and female (71%). Actually, the difference was larger between Norwegians (74%) and foreign students (70%). One challenge for some students was that they had to learn C#. Most students did not think this issue was negative thing, as to know C# is useful for later in the career and it is not very different from Java. Another challenge using XNA as a development platform was that it only runs on the Microsoft Windows platform. This is a major problem as more and more students have laptops

running Mac OS X and Linux. To compensate for this problem we provided a computer lab where 10 PCs running Microsoft Windows with XNA Game developer studio 2.0 installed. Unfortunately these PCs did not have proper graphics cards, making game development slow and tedious. To compensate for this problem in the future, we might offer game projects on other platforms such as Android and iPhone. Apart from the lack of support for other operating systems, we were very pleased with using XNA as a game developer platform. The high-level APIs in XNA makes it possible to be productive with little effort. Also XNA is flexible in terms of what games can be implemented and how the architecture can be designed. For the students, the opportunity to develop XBOX 360 games is very tempting. Only few of the teams tried to run their games on the XBOX 360 mainly due to time pressure. In XNA Game Studio 4.0 it is also possible to develop for Windows Phone, extending the target platform even more. This can give more variety of what kind of projects the students can develop in future projects.

ACKNOWLEDGMENTS

We would like to thank Jan-Erik Strøm and Trond Blomholm Kvamme for implementing XQUEST and for their inputs to this paper. We would also like to thank Richard Taylor at the Institute for Software Research (ISR) at University of California, Irvine (UCI) for providing a stimulating research environment and for hosting a visiting researcher from Norway. The Leiv Eriksson mobility program offered by the Research Council of Norway has sponsored this work.

REFERENCES

- [1] R. Rosas, M. Nussbaum, P. Cumsille, V. Marianov, M. Correa, P. Flores, V. Grau, F. Lagos, X. Lopez, V. Lopez, P. Rodriguez, and M. Salinas, Beyond Nintendo: design and assessment of educational video games for first and second grade students. *Computers & Education*, 40(1): 71–94, 2003.
- [2] M. Sharples, The design of personal mobile technologies for lifelong learning. *Computer & Education*, 34(3-4): 177–193, 2000.
- [3] A. Baker, E. O. Navarro, and A. Hoek, Problems and Programmers: an Educational Software Engineering Card Game. In *Proceedings of the 25th International Conference on Software Engineering (ICSE 2003)*, pages 614–619, 2003.
- [4] L. Natvig, S. Line, and A. Djupdal, Age of Computers: An Innovative Combination of History and Computer Game Elements for Teaching Computer Fundamentals. In *FIE 2004: Proceedings of the 2004 Frontiers in Education Conference*, 2004.
- [5] A. O. Navarro, and A. Hoek, SimSE: an Educational Simulation Game for Teaching the Software Engineering Process. In *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 233–233, New York, NY, USA. ACM Press, 2004.
- [6] G. Sindre, L. Nattvig, and M. Jahre, Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals. To appear in *IEEE Transaction on Education*.
- [7] B.A. Foss, and T.I. Eikaas, Game play in Engineering Education - Concept and Experimental Results. *The International Journal of Engineering Education* 22(5), 2006.
- [8] A.I. Wang, O.K. Mørch-Storstein, and T. Øfsdahl, Lecture quiz - a mobile game concept for lectures. In *The 11th IASTED International Conference on Software Engineering and Application (SEA 2007)*, November 19-21, 2007.
- [9] A.I. Wang, T. Øfsdahl, and O.K. Mørch-Storstein, An Evaluation of a Mobile Game Concept for Lectures. In *21st IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2008)*, April 14-17, 2008.

- [10] M. S. El-Nasr, and B.K. Smith, Learning through game modding. In ACM Computer Entertainment 4(1), Jan 2006.
- [11] B. Wu, and A.I. Wang, An Evaluation of Using a Game Development Framework in Higher Education. In 22nd IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2009), February 17-19, Hyderabad, India, 2009.
- [12] A. Sliney and D. Murphy, JDoc: A Serious Game for Medical Learning. In Proceedings of the First international Conference on Advances in Computer-Human interaction, February 10 – 15, 2008.
- [13] F. Mili, J. Barr, M. Harris, and L. Pittiglio, Nursing Training: 3D Game with Learning Objectives. In Proceedings of the First international Conference on Advances in Computer-Human interaction, February 10 – 15, 2008.
- [14] L.v. Ahn, Games with a Purpose. IEEE Computer Magazine: 39(6), June, 92-94, 2006.
- [15] J. Blow, Game Development: Harder Than You Think. In Queue: 1(10), February 28-37, 2004.
- [16] G.M. Youngblood, Using XNA-GSE Game Segments to Engage Students in Advanced Computer Science Education. In The 2nd Annual Microsoft Academic Days Conference on Game Development, February 22-25, 2007.
- [17] E. Sweedyk and R.M. Keller, Fun and games: a new software engineering course. ACM SIGCSE Bulletin, 37(3), 138-142, September 2005.
- [18] K. Claypool and M. Claypool, Teaching software engineering through game design. In Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in Computer Science Education (ITiCSE '05), Caparica, Portugal, 123-127, June 27 - 29, 2005.
- [19] D. Volk, How to embed a game engineering course into a computer science curriculum. In Proceedings of the 2008 Conference on Future Play: Research, Play, Share, 192-195, Toronto, Ontario, Canada, November 3 - 5, 2008.
- [20] J. Linhoff and A. Settle, Teaching game programming using XNA. In Proceedings of the 13th Annual Conference on innovation and Technology in Computer Science Education (ITiCSE '08), 250-254, Madrid, Spain, June 30 - July 02, 2008.
- [21] Q. Zhu, T. Wang, and S. Tan, Adapting Game Technology to Support Software Engineering Process Teaching: From SimSE to MO-SEProcess. In Proceedings of the Third international Conference on Natural Computation (ICNC 2007) - Volume 05, 777-780, August 24 - 27, 2007.
- [22] Y. Rankin, A. Gooch, and B. Gooch, The impact of game design on students' interest in CS. In Proceedings of the 3rd international Conference on Game Development in Computer Science Education (GDCSE '08), 31-35, Miami, Florida, February 27 - March 03, 2008.
- [23] S. Leutenegger and J. Edgington, "A games first approach to teaching introductory programming," SIGCSE Bull., vol. 39, pp. 115-118, 2007.
- [24] B. D. Collier and M. J. Scott, "Effectiveness of using a video game to teach a course in mechanical engineering," Comput. Educ., vol. 53, pp. 900-912, 2009.
- [25] J. Distasio and T. Way, Inclusive computer science education using a ready-made computer game framework. In ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, 116-120, 2007.
- [26] Washington State University Vancouver and University of Puget Sound. 2008 The Java Instructional Gaming Project. Web: <http://ai.vancouver.wsu.edu/jig/>, Retrieved June 2008.
- [27] C. Johnson and J. Voigt, DXFramework. Web: <http://www.dxframework.org>, Retrieved June 2008.
- [28] I. Parberry, SAGE: a simple academic game engine. Web: <http://larc.csci.unt.edu/sage>, Retrieved June 1, 2008.

- [29] R. Coleman, S. Roebke, and L. Grayson, GEDI: a game engine for teaching videogame design and programming. *Journal of Computing Science in Colleges*: 21(2), 72–82, 2005.
- [30] Rockwell Automation Inc, Arena Simulation Software. Web: <http://www.arenasimulation.com/>, Retrieved June 2008.
- [31] IBM, INNOV8 – a BPM Simulator. Web: <http://www-304.ibm.com/jct03001c/software/solutions/soa/innov8.html>, Retrieved June 2008.
- [32] P. Clements, L. Bass, and R. Kazman, *Software Architecture in Practice Second Edition*. Addison-Wesley, 2003.
- [33] J.O. Coplien, *Software Design Patterns: Common Questions and Answers. The Patterns Handbook: Techniques, Strategies, and Applications*. Cambridge University Press, New York, 311-320, 1998.
- [34] A.I. Wang, and T. Stålhane, Using Post Mortem Analysis to Evaluate Software Architecture Student Projects. In *Proceedings of the 18th Conference on Software Engineering Education & Training*, April 18 – 20, 2005.
- [35] D. P. Perry, and A.L. Wolf, *Foundations for the Study of Software Architecture*. ACM Sigsoft Software Engineering Notes: 17(4), 40-52, 1992.
- [36] IEEE, “IEEE Recommended Practice for Architectural Description of Software-Intensive Systems”, Software Engineering Standards Committee of the IEEE Computer Society, 2000.
- [37] P. Kruchten, The 4+1 View Model of Architecture, *IEEE Software*, 12, 6, Pp. 42 – 50, 1995.
- [38] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere, "The Architecture Tradeoff Analysis Method," *Engineering of Complex Computer Systems*, IEEE International Conference on, vol. 0, no. 0, pp. 0068, Fourth IEEE International Conference on Engineering Complex Computer Systems (ICECCS'98), 1998.
- [39] A. BinSubaih, S.C. Maddock (2006), "Using ATAM to Evaluate a Game-based Architecture", Workshop on Architecture-Centric Evolution (ACE 2006), Hosted at the 20th European Conference on Object-Oriented Programming ECOOP 2006, July 3-7, 2006, Nantes, France.
- [40] T. Lozano-Pérez, In *Preface to Autonomous Robot Vehicles*, Springer Verlag, New York, NY, 1990.
- [41] A. Elfes, *Sonar-Based Real-World Mapping and Navigation*. In *IEEE Journal of Robotics and Automation*, no.3, 249-265, 1987.
- [42] R. Simmons, *Concurrent Planning and Execution for Autonomous Robots* In *IEEE Control Systems*, no. 1, 46-50, 1992.
- [43] S.A. Shafer, A. Stentz, and C.E. Thorpe, *An Architecture for Sensor Fusion in a Mobile Robot*. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 7-10, 2002-2011, 1986.
- [44] D. Toal, C. Flanagan, C. Jones, and B. Strunz, *Subsumption architecture for the control of robots*, In *13th Irish Manufacturing Conference (IMC-13)*, 703-711, 1996.
- [45] R. Lumia, J. Fiala, and A. Wavering, *The NASREM Robot Control System and Testbed*. In *International Journal of Robotics and Automation*, no.5, 20-26, 1990.
- [46] A.I. Wang and B. Wu, *An Application of Game Development Framework in Higher Education*, Submitted to *International Journal of Computer Games Technology*, 2008.
- [47] C. Vichoido, M. Estranda and A. Sanchez, *A constructivist educational tool: Software architecture for web-based video games*, 4th Mexican International Conference on Computer Science (ENC 2003), 8-12 September, Apizaco, 2003.
- [48] J. Krikke, *Samurai Romanesque, J2ME, and the Battle for Mobile Cyberspace*, *IEEE Computer magazine*, 23(1), 2003.
- [49] S. Rabin, *Introduction to Game Development*, Course Technology Cengage Learning, 2008.
- [50] A. Rollings and D. Morris, *Game Architecture and Design - A New Edition*. New Riders Publishing, 2004.

- [51] G. Booch, Best Practices in Game Development. IBM Presentation March 12, 2007.
- [52] A. Grossman, Postmortems From Game Developer. Focal Press, January 2003.
- [53] R. Darken, P. McDowell, and E. Johnson, The Delta3D Open Source Game Engine. In IEEE Computer Magazine, May/June 2005.
- [54] NGA, Norwegian Game Awards 2011 – Home, Web: <http://www.gameawards.no> , Accessed April 7th 2011.
- [55] A. I. Wang, "Extensive Evaluation of Using a Game Project in a Software Architecture Course," Trans. Comput. Educ., vol. 11, pp. 1-28, 2011.

Paper 10:

GDF6: Bian Wu, Alf Inge Wang, "Game Development Framework for Software Engineering Education", 2011 International IEEE Consumer Electronics Society's Games Innovation Conference (IGIC 2011), November 2011, Orange, California, USA.



Game Development Frameworks for SE Education

Bian Wu, Alf Inge Wang, Norwegian University of Science and Technology

Abstract—This paper presents a literature survey about the method of creating/modifying a game on a game development framework (GDF) as an assignment to learn software engineering (SE), and we share our recommendation for choosing an appropriate GDFs.

I. INTRODUCTION

Games have been used in schools for many years to help students learn skills in math, language, science, engineering and other domains in an interesting and motivating way. Another innovative way is to provide exercises that require students to work individually or in groups to modify or develop a game as a part of a course using a game development framework (GDF) to learn skills within computer science or software engineering (SE) [1-3]. GDF denotes all toolkits used to develop games. This paper focuses on criteria for selecting appropriate GDFs that can be used in student exercises to learn SE skills. The motivation for teaching SE through game development is to utilize the students' enthusiasm for game creation. More specifically, we wanted to investigate how GDFs are used in SE education through our own experiences and a literature survey.

II. EXPERIENCES

We present our experiences as an example to explain how we apply XNA as a GDF in software architecture course in 2008 [1]. In this course, 30% of the grade is based on an evaluation of a software architecture project all students have to do. The rest 70% is given from a written examination. The goal of the project is to let students work in groups and apply the methods and theory from the course to design a software architecture for a game and implement it based on the XNA framework. The project consists of the following phases:

- 1) COTS (Commercial Off-The-Shelf) exercise: Learn the technology to be used through developing a simple game.
- 2) Design pattern: Learn how to use and apply design pattern by making changes in an existing game.
- 3) Requirements and architecture: List functional and quality requirements and design the software architecture for a game.
- 4) Architecture evaluation: Use the ATAM (Architecture Tradeoff Analysis Method) evaluation method to evaluate the software architecture of game project in regards to the quality requirements.
- 5) Implementation: Do a detailed design and implement the game based on the created architecture and on the changes from the Architecture evaluation.
- 6) Project evaluation: Evaluate the project as a whole using

a PMA (Post-Mortem Analysis) method.

The course staff issued the tasks of making a functioning game using XNA, based on students' own defined game concept. However, the game had to be designed according to a specified and designed software architecture. Further, the students had to develop an architecture where they had to focus on one particular quality attribute. We used following definitions for the quality attributes in the game projects: Modifiability, the game architecture and implementation should be easy to change in order to add or modify functionality; and Testability, the game architecture and implementation should be easy to test in order to detect possible faults and failures. These two quality attributes also were related to the course content. Finally, we got positive feedback from students' survey [1-3].

III. RESEARCH CONTEXT SURVEY

The scope of this paper is limited to the selection of GDFs only used in SE education, as SE is the major teaching field where GDFs applied. The survey is based on literature from IEEE Xplore and ACM digital library.

When looking into the background of how GDFs are used in SE education, we focus on why apply a GDF in a SE course in the first place. It is common to describe the teaching design using a GDF from the angle of teachers previous experiences from the course, not explaining its learning theory context [4, 5]. However, we still can find literatures that explain this learning activity, especially in SE education field.

For example, the paper "Learning Through Game Modding" [2] presents its experiences of using a GDF to teach students SE. It considers the learning activity of modifying/creating a game in a GDF in SE education as a design activity that has educational benefits such as learning content, skills, and strategies [6]. Design activities are meaningful and engaging to students for exploring skills (analysis, synthesis, evaluation, revision, planning and monitoring) and concepts to understand how they can be applied in the real world. Further, learning by modifying/creating games can be considered as variant of several available construction activities.

Seymour Papert presents programming as one example of the constructionism learning theory [2]. Constructionism involves two activities [7]. The first is the mental construction of knowledge that occurs with world experiences, a view borrowed from Jean Piaget's constructivist theories of learning and development. The second is a more controversial belief that new knowledge can be constructed with particular effectiveness when people engage in constructing products

that are personally meaningful. The important issue is that the design and implementation of products are meaningful to those creating them, and that learning becomes active and self-directed through the construction of artifacts. In SE education, creating games on GDFs could be this artifact.

A similar positive response to above is [8]. It presents a case study to use double stimulation [9] to guide the exercise designs based on a GDF. It also considers that using a GDF in SE education could be a knowledge construction process. It describes how to use double stimulus to guide a teaching activity, including the learning activity from creating a game. In schools, learners face a challenge, a problem, or a task that has been designed for a particular pedagogical purpose or they face situations that are likely to appear in work and public life. In both cases the purpose of exploiting tools is for learners to respond to such challenges. Based on constructionism, it constructs the relationship between the educational tasks and the material artifacts. This relationship is at the heart of Vygotsky's notion of double stimulation [9], a method for studying cognitive processes and not just results. In a school setting, typically the first stimulus would be the problem or challenge to which learners are expected to respond. The second stimulus would be the available mediating tools, like GDFs.

Similarity, using GDFs in SE education is related to Problem-Based Learning (PBL) [10, 11]. PBL is a pedagogical model that emphasizes the role of a real-life problem and a collaborative discovery process in learning [12]. Within a typical PBL setting, students are first given a challenging but realistic problem of significant size, relevant to the learning objectives of a given course. They are then encouraged to solve the problem in a group throughout the semester as independently as possible with minimum help from the instructor of the course. Apart from the traditional lecture-oriented teaching approach, PBL puts more emphasis on the instructors' role as facilitators, to prepare meaningful and interesting problems, and to create and organize course materials in a manner that students have a just right dose of information in each class to incrementally develop a final solution based on a GDF to the primary problem of the semester.

IV. SURVEY OF GDFs USED IN SE EDUCATION

In order to identify the main feature of several GDFs, we classify them according to two categories: GDFs for novices, and GDFs for developers.

The focus of GDFs for novices is to provide visual interface for customizing game templates and to allow creating or designing games with little or no programming skills. Here are examples of GDFs used in assignments to learn SE from literature survey and its resource link: Alice [13-16]; Scratch [17-19]; CeeBot Series [20]; Warcraft3 Editors [2]; Never Winter Night Toolsets [21]; Greenfoot [22]; Game maker [23, 24]; StarLogo TNG [25]; and Wu's castle [26]. The way these GDFs are used in SE education varies. E.g., Alice and Scratch are typically used for introducing programming or object-

orientation concept to students where the students get introduced to programming concepts through visually manipulating objects in order to implement some simple game behaviors from scratch. Other GDFs are mainly editors or modifiers for existing games, such as the Warcraft3 editor or the Never Winter Night toolsets. The educational approach when using such GDFs are totally different, as the focus is on tailoring or modifying existing behavior in the game instead of building everything from scratch.

The focus of GDFs for developers is to offer toolkits that support development of high quality 2D/3D rendering, special effects, physics, animations, sound playback, and network communication in common programming languages such as C++, C# and Java. Most of the commercial game engines belong in this category. Here are examples of such GDFs used in SE education: BiMIP [27]; Unreal Engine [2, 5]; XNA [28, 29]; XQUEST[30]; XNACS1Lib framework [31]; Android/Sheep [8]; MUPPETS framework [32]; and SIMPLE framework [33]. When using GDFs such as XNA, XQUEST and Android/Sheep, the students will mainly develop everything from scratch and follow the whole software cycle. But for other GDFs, such as Unreal game engine, the basic game functionality is in place and the programming will focus on the game instance. This is a more restrictive approach in what you can learn and the application of the software development process. If the goal of the SE course is to go through the whole software cycle, game engines are not usually suitable GDFs.

V. RECOMMENDATIONS

From both of our experiences and literature survey, introducing a GDF in a SE course can have positive effects such as higher enrollment, improved student motivation and project group dynamics, and more effort put into projects/ assignments [34]. The higher enrollment is mainly due to most of students think it is more interesting to work on a game project than e.g. a banking system. The improved student motivation and group dynamics is mainly due to collaboration of the teamwork provides the possibility of creating their own imaginative games and game development require other than pure technical skills.

However, there are also some obvious disadvantages. The most evident one is that some students will focus too much on the game development thus losing focus on what they shall learn in SE. This means that the design of the course and the project must be carried out in such a way that the students are forced to learn and use the SE methods and disciplines being taught in the course. One approach to enforce SE elements in exercises and projects is to require documentation during the whole project focusing on the SE learning goals and emphasize that the evaluation of the exercise and project will mainly focus on the quality of these SE deliverables and less on the game being produced. This is from our experiences on using XNA in the software architecture course. To ensure the SE focus, the students had to deliver part-deliveries focusing on different areas of software architecture, such as design and

architectural patterns, functional and quality requirements, a software architecture for the game described through several views, an architectural evaluation, and an implementation of the game where the students had to adhere to their quality requirements, their chosen patterns and their designed software architecture.

Further, it is really important to choose the appropriate GDF to be used in a SE course. There are many factors that come into play when conceiving an assignment based on a GDF:

Educational goal: The educational goal of the SE course will greatly affect the choice of GDF, e.g. if the focus of the course will be on requirements, software architecture, design, implementation, testing, maintenance, project management or the software process. As mentioned before, SE courses focusing on the whole development cycle should use GDFs that allow the students to develop a game from scratch such as XNA. However, if a SE course only focuses on testing or quality assurance, a game engine can be very effective for the education goals such as Unreal can work very well. Another important factor is whether course's focus on procedural programming vs. Object Oriented (OO) programming. For SE courses with more technical requirements, GDFs such as XNA, XQUEST or Android/Sheep are more appropriate. In other courses, the most important goal is not to learn programming, but rather to learn the SE principles such as requirements, design, and the project management. For such courses, GDFs with visual programming such as Alice, Scratch or the Warcraft3 editor can be used.

SE constraints: All GDFs have constraints related to SE in how they have been designed or how they are released. One example is open source GDFs that make it possible to do white-box testing on the GDF, while for other GDFs the source code is not available for the students. Open source GDFs are also important in courses where it is necessary to understand the details of the components used in students' game creation. Further, some GDFs might constrain how you can design your games, what design and architectural patterns you can use, how event handling must be managed, the freedom of expanding the GDFs functionality and more. These constraints must be integrated in the SE teaching to introduce the students to the real world where software never is built from scratch. Another important issue is the openness of the GDF to other tools. This issue could be very important e.g. the integration of test tools.

Programming experience: The programming experience of the students will highly affect the choice of GDF between the ones for novices and the ones for developers. Another factor is what programming languages the students know, such as Java, C#, C, C++ etc. E.g. to use XNA/XQUEST or Android/Sheep, the students must know OO programming well and be familiar to design patterns and OO principles in addition to C# and Java. And some GDFs offer their own programming languages to simplify the game programming (scripting). From our own experience, the hardest part for the students is not the programming language itself but rather the libraries and APIs they have to learn.

Staff expertise: It is essential that the course staff have technical experience in a GDF used in a SE course to provide help to students to avoid having them focusing on only the technical matter and not the SE challenges. From our own experiences on running a software architecture course, it is necessary to have dedicated staff to provide technical GDF support. Although it is important that the teacher teaching the SE course knows the basics of the GDFs, it is not necessary for this teacher to have a complete technical insight of the GDF. However, it is critical to have course staff available that can help the student with technical problems during the exercises or project.

Usability of the GDF: To avoid too much focus on technical matters and problems, it must be possible to learn the GDF quickly without too much of a hassle. In practice this means that the GDF must be well-designed, have a logical structure, provide high-level APIs, provide correct, updated and available documentation, provide helpful and many examples, and have many available tutorials. It is also a huge advantage if an active developer community supports the GDF. XNA is a good example of a GDF, which is well designed with high-level APIs, well documented and supported, and an active community. It is recommended to establish a GDF community within a course e.g. using a web forum, as well as encouraging the students to use external web resources.

Technical environment: Technical considerations must be taken into account when selecting a GDF. Typical technical considerations include operating system and hardware compatibility, license policies, tool support, support for third-party tools, and how difficult the software is to install on the students' PCs. The technical requirements might also be an economical issue, as the choice of GDF might force hardware upgrades or paying for expensive licenses. A typical problem is e.g., that XNA runs only on Windows, and many students now have PCs running Linux or Mac OS X. As our experiences on using XNA in a software architecture course, many of the students did not have a Windows PC at first and these students were told to use the available computer labs. Soon, however, we discovered that the existing computer labs running thin-clients were insufficient for running XNA. The problem was partly solved by the students themselves as many of the Mac OS X and Linux users installed Windows on their PCs (dual boot). In addition, our department gave access to a computer lab with stand-alone PCs powerful enough to run XNA.

The list of considerations above should be included in the process of finding the appropriate GDF for a SE course. If an appropriate GDF is chosen and the project or exercises "force" students to provide SE deliveries through the semester, the result is likely to be improved project results as the students are better motivated and put more effort into the work.

VI. CONCLUSIONS AND FURTHER WORK

Through our experiences and literature survey on the theoretical context and various GDFs used in SE education, it

has shown that this method has potential motivation to help students to learn SE courses. In order to select an appropriate GDF, we also identify the impact factors that play important roles on design process for the course when using GDFs in SE education. We believe that our study can provide the guidance for the teachers or researchers in the area of SE education, even for the GDFs' designers in the aspect of the enhancement of GDFs' educational features.

However, time, cost and expertise are significant barriers to experimenting with GDFs in educational settings, and there are limitations to what skills can be acquired using GDFs [2]. Based on our initial survey, this area deserves more research on the applications of GDF for SE education and how to design and improve the teaching process to maximize the effectiveness of using GDF in education.

REFERENCE

- [1] A. I. Wang and B. Wu, "An Application of a Game Development Framework in Higher Education," *International Journal of Computer Games Technology*, vol. 2009, 2009.
- [2] M. S. El-Nasr, "Learning through game modding," *Computers in entertainment*, vol. 4, 2006.
- [3] B. Wu, *et al.*, "An Evaluation of Using a Game Development Framework in Higher Education," *Proceedings / Conference on Software Engineering Education and Training*, 2009.
- [4] S. v. Delden, "Industrial robotic game playing: an AI course," *J. Comput. Small Coll.*, vol. 25, pp. 134-142, 2010.
- [5] E. L. Wynters, "3D video games: no programming required," *J. Comput. Small Coll.*, vol. 22, pp. 105-111, 2007.
- [6] S. Puntambekar and J. L. Kolodner, "Toward implementing distributed scaffolding: Helping students learn science from design," *Journal of Research in Science Teaching*, vol. 42, pp. 185-217, 2005.
- [7] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*. New York, 1980.
- [8] B. Wu, *et al.*, "Extending Google Android's Application as an Educational Tool," presented at the The 3rd IEEE Information Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGTEL 2010), Kaohsiung, Taiwan, April 12-16, 2010. . 2010.
- [9] L. S., *Mind in society: The development of higher psychological processes*, 1978.
- [10] A. Garrido, *et al.*, "Using graphics: motivating students in a C++ programming introductory course," in *EAAEIE Annual Conference*, 2009, pp. 1-6.
- [11] J. Ryoo, "Teaching object-oriented software engineering through problem-based learning in the context of game design," in *21st Conference on Software Engineering Education and Training*, 2008, p. 137.
- [12] H. S. Barrows, "A taxonomy of problem-based learning methods," *Medical Education*, vol. 20, pp. 481-486, 1986.
- [13] R. H. Seidman, "Alice first: 3D interactive game programming," *SIGCSE Bull.*, vol. 41, pp. 345-345, 2009.
- [14] E. W. Amerikaner, "Introduction to computer science using Alice 2.0: tutorial presentation," *J. Comput. Small Coll.*, vol. 25, pp. 141-141, 2010.
- [15] K. Anewalt, "Making CS0 fun: an active learning approach using toys, games and Alice," *J. Comput. Small Coll.*, vol. 23, pp. 98-105, 2008.
- [16] L. Werner, *et al.*, "Can middle-schoolers use Storytelling Alice to make games?: results of a pilot study," presented at the Proceedings of the 4th International Conference on Foundations of Digital Games, Orlando, Florida, 2009.
- [17] G. Fesakis and K. Serafeim, "Influence of the familiarization with "scratch" on future teachers' opinions and attitudes about programming and ICT in education," presented at the Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education, Paris, France, 2009.
- [18] P. A. G. Sivilotti and S. A. Laugel, "Scratching the surface of advanced topics in software engineering: a workshop module for middle school students," *SIGCSE Bull.*, vol. 40, pp. 291-295, 2008.
- [19] W. Jui-Feng, *et al.*, "Teaching Boolean Logic through Game Rule Tuning," *IEEE Transactions on Learning Technologies*, vol. 3, pp. 319-328, 2010.
- [20] T. Phit-Huan, *et al.*, "Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception," in *International Conference on Computer Technology and Development*, 2009(ICCTD '09), 2009, pp. 42-46.
- [21] J. Robertson and C. Howells, "Computer game design: Opportunities for successful learning," *Computers & Education*, vol. 50, pp. 559-578, 2008.
- [22] M. Al-Bow, *et al.*, "Using game creation for teaching computer programming to high school students and teachers," *SIGCSE Bull.*, vol. 41, pp. 104-108, 2009.
- [23] Y. Rankin, *et al.*, "The impact of game design on students' interest in CS," presented at the Proceedings of the 3rd international conference on Game development in computer science education, Miami, Florida, 2008.
- [24] Yulia and R. Adipranata, "Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment," in *2nd International Conference on Education Technology and Computer (ICETC)*, 2010, pp. V2-355-V2-359.
- [25] K. Wang, *et al.*, "3D game design with programming blocks in StarLogo TNG," presented at the Proceedings of the 7th international conference on Learning sciences, Bloomington, Indiana, 2006.
- [26] M. Eagle and T. Barnes, "Experimental evaluation of an educational game for improved learning in introductory computing," presented at the Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, USA, 2009.
- [27] A. Garrido, *et al.*, "Using graphics: motivating students in a C++ programming introductory course," in *EAAEIE Annual Conference*, 2009, 2009, pp. 1-6.
- [28] B. Wu, *et al.*, "An Evaluation of Using a Game Development Framework in Higher Education," *22nd Conference on Software Engineering Education and Training*, 2009, pp. 41-44, 2009.
- [29] K. Sung, *et al.*, "Game-Themed Programming Assignment Modules: A Pathway for Gradual Integration of Gaming Context Into Existing Introductory Programming Courses," *IEEE Transactions on Education*, 2010.
- [30] B. Wu, *et al.*, "XQUEST used in software architecture education," in *International IEEE Consumer Electronics Society's Games Innovations Conference (ICE-GIC 2009)*, 2009, pp. 70-77.
- [31] R. Angotti, *et al.*, "Game-themed instructional modules: a video case study," presented at the Proceedings of the Fifth International Conference on the Foundations of Digital Games, Monterey, California, 2010.
- [32] K. J. Bierre and A. M. Phelps, "The use of MUPPETS in an introductory java programming course," presented at the Proceedings of the 5th conference on Information technology education, Salt Lake City, UT, USA, 2004.
- [33] H. C. Jiau, *et al.*, "Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics," *IEEE Transactions on Education*, vol. 52, pp. 555-562, 2009.
- [34] A. I. Wang, "Extensive Evaluation of Using a Game Project in a Software Architecture Course," *Transactions on Computing Education (ACM)*, vol. Volume 11., February 2011. 2011.

Paper 11:

GDF7: Bian Wu, Alf Inge Wang, “A guideline for game development-based learning: A literature review”, Accepted by the International Journal of Computer Games Technology.

A guideline for game development-based learning: A literature review

Bian Wu, Alf Inge Wang, *Norwegian University of Science and Technology*

Abstract— This study aims at reviewing published scientific literature on the topics of game development-based learning (GDBL) method using game development frameworks (GDFs) with the perspective of: (a) summarizing a guideline for using GDBL in a curriculum; (b) identifying relevant features of GDFs; and (c) presenting a synthesis of impact factors with empirical evidence on the educational effectiveness of the GDBL method. After systematically going through available literature on the topic, 34 relevant articles were selected for the final study. We analyzed the articles from three perspectives: 1) Pedagogical context and teaching process, 2) Selection of GDFs, and 3) Evaluation of the GDBL method. The findings from the 34 articles suggests that GDFs have many potential benefits as an aid to teach computer science, software engineering, art design and other fields, and that such GDFs combined with motivation from games can improve students' knowledge, skills, attitudes and behaviors in contrast to traditional classroom teaching. Furthermore, based on the results of literature review, we extract a guideline of how to apply the GDBL method in education. The empirical evidence of current findings gives a positive overall picture and can provide a useful reference to educators, practitioners and researchers in the area of game-based learning.

Index terms — Game based learning, Game development-based learning, Game development framework, Teaching design, Literature review

1 INTRODUCTION

Computer games and video games have become very popular in children and adolescents' life and play a prominent role in the culture of young people [1]. Games can now be played everywhere in technology-rich environments equipped with laptops, smart phones, game consoles (mobile and stationary), set-top boxes and other digital devices. From this phenomenon, it is believed that the intrinsic motivation that young people shows towards games can be combined with educational content and objectives into what Prensky calls "digital game based learning" [2].

Besides of an abundant appearance of games in young students life, game development technology has matured and become more advanced than before [3]. Based on various existing game development software, the whole duty of game development process can be divided into several domains and roles such as game programmers, 3D model creators, game designers, musicians, animators, play-writers, etc. Under this situation, some web-resources and game engines can simplify the game development process. For instance, Microsoft's XNA game development kit provides the game loop function to draw and update the game contents, and it also provides convenient game development components to load the different format of graphics, audio, and videos. This makes it possible for students to modify existing games or develop own new games with or without programming. They can design and implement their own game concepts with these game creation tools, learn the developing skills and relevant knowledge, and accumulate related practical experience.

In this context, not only can a game be used for learning, but also the game development tools be used for studying relevant topics within computer science, software engineering (SE) or game programming through motivating assignments. Generally, games can be integrated in education in three ways [4, 5]. First, games can be used instead of traditional exercises motivating students to put extra effort in doing the exercises, and giving the teacher and/or teaching assistants an opportunity to monitor how the students work with the exercises in real-time, e.g. [6, 7]. Second, games can be played within lectures to improve the participation and motivation of students, e.g. [8, 9]. Third, the students are required to modify or develop a game as a part of a course using a Game Development Framework (GDF) to learn skills within computer science and SE, e.g. [10]. And we label this third as Game Development-Based Learning (GDBL). And the GDFs denote the toolkits that can be used to develop or modify games, e.g. game engine, game editors, or game (simulation) platforms, or even any Integrated Development Environment (IDE), like Visual C++, Eclipse, J2ME, and Android SDK since all of them can be used to build games. This literature review focuses on using the GDBL method in education, where GDFs are used in student exercises to learn skills, extending the use of GDFs as a teaching aid. The motivation for teaching through game development is to utilize the students' enthusiasm for games. This GDBL method is not new. The earliest similar application of learning through programming in a game-like environment was in early 1970s. The Logo [11], the turtle graphics, is one of the oldest libraries that was used to introduce computing concepts to beginners. The concept was

based on a “turtle” that could be moved across a 2D screen with a pen, which could be positioned on or off the screen, and thus, may leave a trace of the turtle’s movements. Programming the turtle to draw different patterns can be used to introduce general computing skill, such as procedural operations, iteration, and recursion. Further, in 1987, Micco presented the usage of writing a tic-tac-toe game for learning [12]. After several years of development, we believe that GDBL methods have been improved through the development of technology. Thus, we investigate how GDFs are being used in education through a literature survey and investigate how traditional lectures can become more dynamic, collaborative and attractive to the students utilizing the current technology rich environment. However, this assertion needs to be further supported by relevant theory, application experiences, evaluation results, and empirical evidence. Nevertheless, to the best of the authors’ knowledge, there does not exist any comprehensive literature reviews on application of the GDBL method so far.

The aim of the study is to review recently published literature on the use of GDFs in education to:

- (a) Summarize a guideline for how to use GDBL in a curriculum.
- (b) Identify the features of GDFs related to GDBL.
- (c) Present a synthesis of impact factors with the empirical evidence on the educational effectiveness of the GDBL method.

The study is unique in that it presents an overview of the recently published literature on the use of GDFs in education, while taking into account both game engines and relevant toolkits to create/modify games or game-like systems (e.g. simulators). The study can provide useful guidance to teachers at different educational levels or areas, as well as to educators, practitioners and researchers in the areas of game-based education.

The paper is organized as follows. Section 2 describes the method used for carrying out the systematic review of articles, Section 3 presents the results from the literature review, Section 4 extracts a guideline for GDBL according to existing literature, and finally Section 5 concludes the paper.

2 METHOD

Informed by the established method of systematic review [13, 14], the review was undertaken in distinct stages: the development of review protocol, the identification of inclusion and exclusion criteria, a search for relevant studies, critical appraisal, data extraction, and synthesis.

2.1 Protocol development

We developed a protocol for the systematic review by following the guidelines, procedures and policies of the

Campbell Collaboration¹, the Cochrane Handbook for Systematic Reviews of Interventions [13], the University of York’s Centre for Reviews and Dissemination’s guidance for those carrying out or commissioning reviews [14], and also refer to reviews on serious game research [15, 16]. This protocol specified the research aim, search strategy, inclusion, exclusion criteria, data extraction, and methods of synthesis.

2.2 Data source and search strategy

For the purpose of the study, a literature search was undertaken in December 2010 in the following international online bibliographic databases: (a) ACM portal, (b) IEEE Xplore, (c) Springer, (d) Science direct. The search string used was: (“Game”) AND (“Learning” OR “Teaching”) AND (“Lecture” OR “Curriculum” OR “Lesson” OR “ Course” OR “ Exercise”). And “education” was not included in the keyword list since we considered that education was a quite general word and did not help minimize the searching scope. Searches were limited to titles and abstracts of articles published in journals, and conference proceedings (some are book chapters), in English, from 2000 and onwards. The latter limitation was posed due to the rapid changes in ICT (Information and Communications Technology) in general, and in computer game development technologies in particular.

2.3 Data extraction with inclusion and exclusion criteria

Figure 1 shows the complete process of the data extraction. The first step was to identify relevant studies. A number of journal and proceedings articles about GDBL were located during searches in the aforementioned databases. The articles were examined and the search resulted in 1155 articles. In the step 2, from abstracts of each article, we distinguished learning through game play or game development. And most of the excluded articles were using games directly in classroom to motivate the students’ interest and attendance rate, and using game play instead of traditional exercises to study or review the course content. For instance, these were articles generally addressing using virtual online multiplayer game environments to provide a collaborative learning style, e.g. [17, 18], articles which referred to games used in classroom to motivate attendance and to review the course knowledge, e.g. [8]. In addition, the articles related to the economics terms “game theory” and “business game” used as business terms were also excluded from this category. Besides, we excluded articles that depicted novel game concepts that were not computer or video games but physical game

¹ www.campbellcollaboration.org

activities without any technology support for the lecture. For instance, article [19] used a self-made table card game in SE education. Mainly based on these three criteria, a total of 1010 articles were excluded after this step.

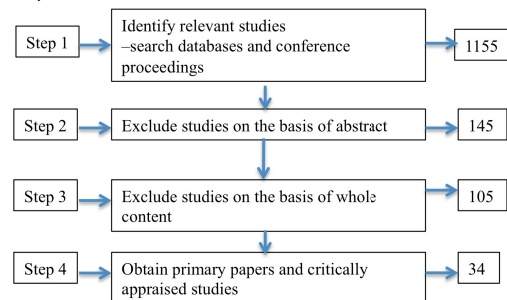


Fig. 1. Steps of the study selection process

In the step 3, the whole content of the articles was checked. The inclusion criteria were further limited to the scope: a case study or several case studies in the article to describe GDBL. In particular, it required a) A relatively detailed description of lecture design process. The articles without a detailed description of their teaching design or exercise process made it impossible to validate their process of how to integrate GDFs in lectures or exercises. According to this requirement, posters, tutorial presentations and some short papers without detailed description on teaching process were excluded since they could not provide valuable data for our research aim and made it impossible to validate the effectiveness of the method, e.g. [20-25]. This was also a measure to ensure inclusion of high quality literature in the review. b) Articles using development toolkits in the curriculums but did not aim to develop games were also excluded, e.g. [26]. c) Articles emphasizing on other aspects apart from GDBL were excluded as it was difficult to validate how game development was integrated in class, e.g. learning in a interactive e-lab [27]. Similarly, articles that presented the development of an educational game framework but did not mention how it was integrated in a specific curriculum were excluded, e.g. [28-31]. d) Articles, which focused on changing the controller of the software or hardware, but without elements of computer game development were also excluded, e.g. [32, 33]. Most of them focus on creating a robot controller to learn algorithms, or changing some component of a robot to learn Artificial Intelligence (AI). In contrast, we included learning from modifying parts of a simulator to create the game elements or a game-like system, e.g. [34, 35]. Finally, a total of 105 articles were remaining after this step.

In the step 4, we carefully looked through the remaining articles and compared their topics, methods, teaching process, and evaluation quality from the presentation of their concepts. After the comparison,

the following studies criteria were included: 1) Articles that had collected data from assignments or scores after using GDBL method. 2) Articles that had questionnaires with quantitative data and interviews or feedback with qualitative data. 3) Detailed discussion of the collected data and conclusion. In addition, diverse and innovative articles were not neglected, in order to show the various ways to integrate GDFs in education. However, articles reporting on use of hardware tools to create game or game-like system, such as real robot hand [34], Wii remote [36], Microsoft surface [37], and a projector-camera system [38] to support teaching or learning environment were not included. Finally, a total of 34 articles were included in the review. And we believe these articles were sufficient to get a complete guideline to explain how to integrate the GDBL method in the curriculums.

2.4 Synthesis of findings

A typology to categorize the 34 articles has to be devised. The classification scheme proposed by [39] in their review of the general instructional gaming literature was adopted for the needs of the present study. This scheme, which was also used in [40], defines the following five categories [39]: (a) Research (systematic approaches in the study of gaming targeted at explaining, predicting or controlling particular phenomena or variables), (b) Theory (articles explaining the basic concepts or aspects or derived outcomes of gaming), (c) Reviews (syntheses of articles concerning general or specific aspects of gaming), (d) Discussion (articles stating or describing experiences or opinions with no empirical or systematically presented evidence), and (e) Development (articles discussing the design or development of games or projects involving gaming).

Specifically, for the categorization of the articles, the following criteria were applied in this study. Articles comprising empirical research related to GDBL were assigned to the 'Research' category. Articles comprising theoretical analyses of concepts, aspects or outcomes of GDBL were placed in the 'Theory' category. Articles presenting syntheses of articles concerning GDBL conducted according to explicit methodology were placed in the 'Review' category. Articles reporting on opinions and experiences regarding GDFs used in teaching, with no empirical or systematically presented evidence, were assigned to the 'Discussion' category. Finally, articles mainly reporting on the design or development of GDFs used in the GDBL method were assigned to the 'Development' category. The articles were grouped into these five categories according to their primary focus. Of the 34 articles found after the step 4, 20 were placed in the 'Research' category, 1 in the 'Theory' category, 7 in the 'Discussion' category and 6 in the 'Development' category, whereas no articles fit the "Review" category, which highlights the usefulness and originality of the present study. Like

results from other literature reviews on instructional games [40, 41], in this study there were fewer articles in the 'Theory' categories than in the 'Research', 'Discussion' and 'Development' categories. This can be explained by the fact that instructional games, including GDBL are a relatively new domain of educational technology.

3 RESULTS

This section presents an overview of the studies of the GDBL method based on the results after step 3 and step 4 in Figure 1.

3.1 Overview of the study after the step 3

In order to have a complete overview of GDBL, we chose the results from step 3 mainly due to: (a) They covered a more complete variation of types of GDFs and contained more information than the 34 articles from step 4. (b) They provided more cases in the diversity of GDFs methods used in teaching, which also presents the potential advantages of using GDF in education. (c) They showed the development tendency of GDF related to other factors (e.g. times and technology). We had a study of 105 articles from step 3 representing use of GDBL method spanning over 11 years. Figure 2 presents the distribution of these articles related to publishing year after step 3. The result after step 4 is also presented for reference.

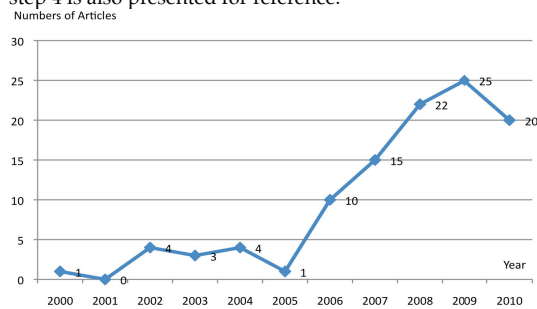


Fig. 2a. Study of each year on using GDBL method (Step 3)

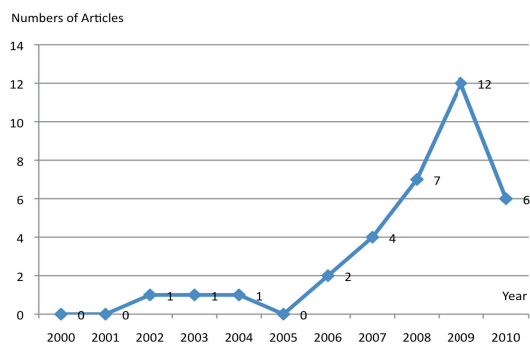


Fig. 2b. Study of each year on using GDBL method (Step 4)

The types of GDF are classified as (a) Game engines: It mainly covers the commercial game engines and mature and well-known toolkits mainly to create games. (b) Self-made GDF: It mainly includes the game development frameworks that were made by the authors of the articles for usage in a specific course. (c) Games or game editors: It mainly contains editors or platforms that can be used to modify games. (d) Simulation platform: It mainly includes controllers to create a game-like system for robots or other simulation platforms. (e) Hardware platform: It mainly includes both game hardware and related software to build games (laptops and computers are excluded), like Wii remotes, windows surface with XNA, robotic hand. (f) Others are general IDEs, like Visual C++, J2ME, or unspecified game creation toolkits with no specific requirement for learning. For some articles that covers more than one attribute like self-made GDF and simulation platform, we choose priority adhering the following sequence: game engine, self-made GDF, game editor, simulation platform, hardware, and others. Figure 3 shows distribution of types of GDFs applied in GDBL articles in percentage. Further, the top five in game engine subcategory are: XNA (9 articles); First Person Shooter (FPS) game engines (Unreal: 2 articles, Torque: 2 articles, Half-life: 1 article), Flash (4 articles), Alice (4 articles), Scratch (3 articles).

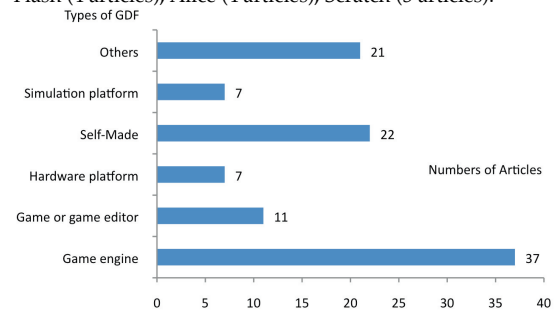


Fig. 3. Study about types of GDF.

From statistics shown in Figure 2 and 3, we discovered the following clues:

1) *Tendency of popularity.* Figure 2a and 2b present the tendency of increasing number of publications of GDBL articles from 2000, especially from 2006. Between 2006 and 2009, the number of GDBL publications grew with 3-7 articles per year, up to max of 25 articles in Fig 2a. Figure 3 shows the distribution of the types of GDFs. From the statistics, game engines are most frequently used in GDBL method. We can infer that the continuous development and improvement of game engines will drive the GDBL's development further in near future.

2) *Technology changes the ways of learning.* After 2006, there was a rapid increase in the number of GDBL articles published. We have analyzed possible reasons

concerning to this phenomenon from three perspectives: (a) Frequent release of new commercial GDFs free of charge, like XNA (2007), Android SDK (2008), and evolution of software development environments, like Flash (acquired by Adobe in 2007) made game development easier than before. Technology changes or enriches the ways of learning and teaching. (b) Cross-disciplinary curriculum started to be used after 2006, e.g. [42, 43]. It provides the possibility to use game development in these topics. c) The up-growing generation of students is a part of a game accepting culture where the public has an open mind towards games. This culture does not only focus on negative effects of video games such as violence and sex, but embraces the positive aspects of games such as social integration, various improved skills, and usage of games for educational purposes, such as Sim-city and Civilization. Furthermore, students that grew up with games have become teachers in schools and may use games in their teaching. They show how technology changes the learning style. Whether it has

positive and negative impact on learning depends on how we adopt the technology (game) and how it is used in teaching and learning.

3.2 Overview of the study after the step 4

In terms of classification method used in e-learning literature [44], a subcategory was iteratively developed based on the thematic topics found in the articles. Each subcategory was labeled with the disciplinary area – programming, SE, art and other topic areas. As already mentioned in the introduction, the intended target audiences of the present study are educators, practitioners, researchers and game designers that use GDFs in learning. The thematic subcategories should help the readers review teaching design, benefits, empirical findings and future research topics in own topic of interest. A similar thematic sub-categorization of research articles was also performed in review of the general instructional games literature [41]. The overview of 34 articles after the step 4 is shown in Table 1 grouped in four categories and labeled with course topics.

TABLE 1
OVERVIEW OF ARTICLES

Category	Item	Article	Major topics	Course topic
Research	R1	[45]	Students develop games on Torque game engine to learn game development.	Game development
	R2	[46]	Undergraduate and graduate build games by adding code in Spacewar simulator to learn artificial intelligence.	AI
	R3	[47]	Undergraduates develop games on XNACS1Lib framework to learn programming.	Programming
	R4	[48]	Students develop games on Scratch to learn basic programming.	Programming
	R5	[49]	Students develop games on Game maker platform to learn software engineering.	SE
	R6	[50]	Students develop games using Greenfoot to learn programming.	Programming
	R7	[51]	Students build games by adding code in Wu's Castle to learn programming.	Programming
	R8	[42]	Students build 3D movies on First person shooting game engine, Maya, Photoshop to learn Digital Character Production and Machinima.	Art
	R9	[10]	Students develop or modify Warcraft3 game editor, unreal game engine, etc. to learn software development, programming, project management, artistic concepts, etc.	Mixed topics
	R10	[43]	Undergraduates develop games to learn outsourcing and software engineering.	SE
	R11	[52]	Students develop games on self-made toolsets to learn programming.	Programming
	R12	[53]	Students develop games on GameMaker to learn programming.	Programming
	R13	[54]	Undergraduates develop Critical Mass board game on web-based platform to learn data structure.	Data structure
	R14	[55]	Undergraduates develop games to learn programming.	Programming
	R15	[5]	Undergraduates develop mini-games on XNA to learn programming.	Programming
	R16	[56]	Graduate develop games on XNA to learn software architecture.	SE
	R17	[57]	Students build games on Scratch to learn Boolean logic.	Boolean logic
	R18	[58]	Pupils build games by adding quiz to a web-based game shell platform to learn literacy.	Literacy
	R19	[59]	Students build games by adding code to a board game: RoboRally to learn artificial intelligence.	AI
	R20	[60]	Middle-school students build games on Storytelling Alice to learn information technology.	Mixed topics
Discussion	D21	[4]	Graduate Students develop games on XNA to learn software architecture.	SE
	D22	[61]	Middle school Students build games on adding code in StarLogo TNG to learn 3D programming.	3D programming
	D23	[62]	Art design students develop games on Flash to learn programming.	Programming
	D24	[63]	Electronics design field Students build game-like system to learn programming, distributed system, etc.	Mixed topics
	D25	[64]	Undergraduate Students develop games to learn programming.	Programming
	D26	[65]	Pupils develop games on NeverWinter Night toolsets to learn basic ICT curriculum.	Mixed topics
	D27	[66]	Students build games by adding code to Bomberman game to learn programming.	Programming
Theory	T28	[67]	Survey of mobile game development for different learning purposes.	Mixed topics
Development	Dev29	[68]	Develop MUPPETS that students could use it for game development to learn programming.	Programming
	Dev30	[69]	Develop XQUEST based on XNA that graduate could use it for game development to learning software architecture.	SE

Dev31	[70]	Develop Sheep based on Android that graduate could use it for game development to learn software architecture.	SE
Dev32	[71]	Design and develop SIMPLE framework that students could use it for game development to learn programming.	Programming
Dev33	[72]	Develop BiMIP framework that undergraduate could use it for game development to learn programming.	Programming
Dev34	[73]	Develop JGOMAS framework that undergraduate could use it for game development to learn artificial intelligence.	AI

These articles presents various GDFs used in GDBL and the covered course topics are summarized in the Figure 4. The article T28 in Table 1 presents a study for using mobile game development as a motivational tool and a learning context in the computing curriculums. From their survey, the game development process can be used in the study of AI, database, computer networks, SE, human-computer interaction, computer graphics, algorithms, programming, computer architectures, and operating systems.

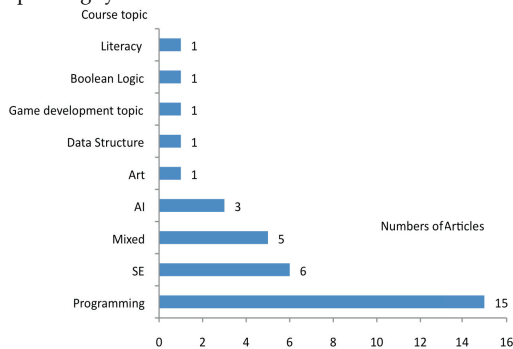


Fig. 4. Distribution of the course topic

Both the data from the Figure 4 and article T28 can validate that the GDBL method can be used to teach various topics. Most applications are in the field of computer science, electronic, and basic IT learning. However, there are some innovative examples of other applications as well: Article R18 presents how a web-based game-shell platform is used to create quiz game to teach pupils literacy with no programming requirement. Article R8 presents how Maya and Photoshop are used to create the digital character and movies that could be used as a video inside of a game.

From Table 1, it also shows that GDBL not only can be used in higher education, but also for basic IT education for kids in middle schools. The article D26 presents how pupils are taught basic ICT (Information and Communications Technology) curriculum by creating games. And the articles R20 and D22 describe how middle school students are taught IT and basic 3D programming by building games. The common GDFs used in the primary and middle schools are some GDFs that do not require much programming experiences for pupils, e.g. the game editor. This will be further discussed in Section 4.

4 FINDINGS

The articles collected after the step 4 are further discussed in this section to serve the purpose of helping to identify and extract the significant elements to meet our aims, like elements to be used to guide the teaching design process when using GDFs in education. Findings are further presented as three aspects: 1) pedagogical context and teaching process, 2) technical aspects, and 3) evaluation results in relation to the aims of this study.

4.1 Pedagogical context and teaching process

This section focuses on the current design process of integrating GDFs in courses or exercises to make the traditional teaching style become more engaging and diversified. This section also provides the detailed steps of how pedagogical theory can be used to guide the teaching design as well as strategies to aid the teaching.

The articles collected in this section are mainly from "Discussion" part in Table 1, and the rest is from the "Research" and "Development" categories. The "Discussion" articles usually have a more complete description than the articles of other categories and include: student background, GDF analysis, course setting and background, and teaching design with strategies. We are also concerned with the diversity and flexibility of using GDBL. The diversity shows not only that standard game engines or game frameworks are used in teaching, e.g. XNA, but also that GDFs that are adapted or extended for teaching, e.g. in article Dev31 they developed an extended library for the Android platform as a GDF for a specific course. Flexibility shows that: (a) the same GDF can be used in different situations, e.g. article D22 use XNA to teach software architecture and article R15 use XNA to teach programming, (b) the teaching process can be flexible to include other strategies than just integrating GDFs in the learning. For instance, article R13 adds the competition in game development for the assignments.

4.1.1 Pedagogical context

Integrating game developments in a course study can provide increased motivation and attractiveness for the students. What is behind this motivation and can any theoretical context explain why GDBL can support learning? We investigated this question in the literature review, mainly focusing on a) why apply the GDBL method in education, and b) how to apply it in a course

in the first place. We found it was common to present the teaching design using a GDF in articles from the perspective of a teacher's experiences from the course, not thinking this process from a learning theory perspective.

Apart from the fact that games motivate for learning, we do not have strong evidence from pedagogical theory to explain why it is a good idea to apply game development in education yet. However, there exists literature that explains game development, opposed to game play, as a pedagogical activity in the classroom. M.S. El-Nasr mentioned that Seymour Papert presented a relevant conclusion that programming is one example of the constructionism learning theory [10]. Constructionism involves two activities [74]. The *first* is the mental construction of knowledge that occurs with world experiences, a view borrowed from Jean Piaget's constructivist theories of learning and development. The *second* is a more controversial belief that new knowledge can be constructed with particular effectiveness when people engage in constructing products that are personally meaningful. The important issue is that the design and implementation of products are meaningful to those creating them, and that learning becomes active and self-directed through the construction of artifacts. In the GDBL method, creating games with GDFs could be this artifact. This could be the fundamental concept to explain the pedagogical context of the GDBL. We can find support for this view from the articles in Table 1. For instance, article R9 considers the learning activity - modifying/creating a game using GDFs as a design activity that has educational benefits such as learning content, skills, and strategies [75]. Design activities are meaningful and engaging to students for exploring skills (analysis, synthesis, evaluation, revision, planning and monitoring), and concepts to understand how they can be applied in the real world. Further, GDBL can be considered as variant of several available construction activities. Similarity, for "learning by design", the article D23 presents using Flash for students from aspect of "learning by doing"--Dewey's theory [76, 77]. The article D26 uses "learning by making" to learn basic ICT (Information and Communications Technology) knowledge by making games, and it describes that "game making" has the potential to be a powerful learning environment according to attributes identified by Smeets [78]. These contexts are the evidence to explain the GDBL method as a constructionism activity from a theoretical aspect.

Based on Seymour Papert's opinion, another question pops up: how to use the pedagogical theory to support the design? A positive response is the article Dev31. It presents a case study on the use of double stimulation [79] to guide the exercise design. It considers that using a GDF in education could be a knowledge construction process and describes how to

use double stimulus to guide a teaching activity. In schools, learners face a challenge, a problem, or a task that has been designed for a particular pedagogical purpose or they face situations that are likely to appear in work and public life. In both cases the purpose of exploiting tools is for the learners to respond to such challenges. Based on constructionism, it constructs the relationship between the educational tasks and the material artifacts. This relationship is at the heart of Vygotsky's notion of double stimulation [79], a method for studying cognitive processes and not just results. In a school setting, typically the first stimulus would be the problem or challenge to which learners are expected to respond to. The second stimulus would be the available mediating tools, like GDFs. Similarity, other pedagogical strategies are also found to support for the GDBL's teaching design. Problem-Based Learning (PBL) presented in the articles R6, R14, D25 and Dev33 are also considered as theoretical reference when using GDBL methods. PBL is a pedagogical model that emphasizes the role of a real-life problem and a collaborative discovery process in learning [80]. Within a typical PBL setting, students are first given a challenging but realistic problem of significant size, relevant to the learning objectives of a given course. They are then encouraged to solve the problem in a group throughout the semester as independently as possible with minimum help from the instructor of the course. Even further, article D25 classified the process into the inception phase of PBL by giving game development requirements; the elaboration phase of PBL by building a rapid game prototype; the construction phase of PBL by implementing a game in a project; and the transition phase of PBL by a results evaluation. Apart from the traditional lecture-oriented teaching approach, PBL puts more emphasis on the instructor's role as a facilitator, to prepare meaningful and interesting problems, and to create and organize course materials in a manner that students have a just right dose of information in each class to incrementally develop a final solution based on a GDF to the primary problem of the semester. In addition, the articles R12 and Dev29 proposed to use collaborative learning together with the game creation process, and article D24 proposed using "old model of Aristotle" [81] in the teaching design. All of them are helpful support for the understanding of the teaching process.

The collections of above results explain the validity of using a GDF in education from a pedagogical angle. Basically, it explains that applying the course content on GDFs by creating games fits well into a knowledge construction process, and it can be integrated with the pedagogical theory supports, like double stimulus or PBL to achieve an improved learning process and outcome. For instance, when we choose double stimulus as a pedagogical theory support, the learning design can be decomposed into two main elements: one

is a problem, task or goal that is designed by the teacher, and the other is a responding learning activity that is implemented by students. From the double stimulus perspective, the first stimulus is tasks or assignments and second stimulus can be chosen as a corresponding tool based on the first stimulus. Its outcome depends on teachers' capacity to keep the two elements match each other. A good task (first stimulus) with inappropriate GDF (second stimulus) will not optimize the output. With this double stimulus support in mind, teachers should find an appropriate match between tasks and GDFs instead of just focusing on one aspect more than the other, like over focus on the design of task but neglect the effort of selecting the GDF. This is not a correct way for applying double stimulus. Further, if the selected GDF always conflicts with the tasks, we should re-consider changing the tasks or GDFs, or even apply a non-game tool. It implies that double stimulus can support learning activity for both GDBL and non-GDBL methods. The teachers should realize it and analyze which tool is better for the course aim and for the students when they apply double stimulus in teaching.

The number of case studies shows that only 30% of 34 articles include both pedagogical and technological design when applying GDBL. This phenomenon reminds us to improve the teaching process with relevant theoretical support. We believe our analysis points to the necessity for further pedagogical and technological co-design to better facilitate awareness of GDBL, thus better conduct the teaching process.

4.1.2 Teaching process

How to integrate GDFs in teaching and exercises is a very important process when applying GDBL. This section analyzes the teaching process and exercise designs on various GDFs to achieve learning by implementing/modifying a game using GDFs. From our survey, we found necessary and common steps for integration of GDFs in a course from selected articles:

The *first step* is to identify explicit course aims. Figure 3 and Figure 4 show relationships between GDBL and other fields, and provides the case studies of how to integrate GDBL into different courses. After the course aim is clear, a common way to integrate GDBL in the course is that the teacher can design an assignment asking to develop a game. The students should then find a solution to this assignment that is in alignment with the course content. When facing such situation, the teacher should find an entry point in how to integrate GDBL with the course and exercises. If this is not possible, we recommend reading articles about similar courses from the selected examples in Table 1 and getting some inspiration. The *second step* is the exercise design and selection of GDFs. When applying a GDF in a certain course, the selection usually depends on the course content and exercises types, etc. We have recognized three types of exercises: One type

is to modify the game or adding component to game platform or simulation platform to achieve a complete game, like in the articles D26 or Dev29. The second type is to create a simple game as an exercise to study or practice one or two concepts from the course content, like in the article R9. The third type is to do a complete game development project applying all concepts from the course. Usually, the first and second types can be used in the beginning of a course as a transition period when students are not familiar with the GDF environment, while the third type exercise can be used as a final exercise. However, there are other special cases, like in article R2 where only one type exercise were selected and applied in the whole process. The main driver of exercise design depends on the course aim and students' background. Selection of GDFs is separately discussed in Section 4.2. The *third step* is to do a tutorial lecture where the GDF is introduced to the students. The *fourth step* is to run an initial exercise, which should be easy to do and let the students get familiar with the development environment. The *fifth and final step* is to do exercises that include implementation of a game. Usually, it is accompanied with some suggestions that were applied in most of the literature: (a) Collaborative learning: the student groups range from 2-6 students in our statistics, further article R12 has some discussions about how to locate student members in groups such as regular meetings with instructors and flexible meetings among group members. It is important to keep instant communication with the exercise requirements, which would be positive to the students' learning towards the GDBL method. For each group member, it would be a tradeoff between cooperative and individual work during the work duty allocation. Further, a workshop is suggested to be held at the end of course. (b) Support: Technical support to help students overcome the technical difficulties they face. It is helpful to give examples in the beginning such as to provide optional examples codes and exercise examples to explain the exercise's complexity. Also there are other strategies like conducting a pilot study before the formal application of GDBL. This approach only appeared in two articles. After the whole teaching process is completed, usually a survey to evaluate both the teaching process and the used strategies is conducted and a more detailed analysis is performed considering the impact factors described in Section 4.3 based on the evaluation from the literature.

4.2 Technical solution

The technical aspect of the GDBL method is mainly about GDFs' features described in the 34 articles. And this section will not go into technical details of development of GDFs due to out of the scope of this paper. On contrary, we mainly analyze the GDFs features in the context of GDBL based on our aim of this study.

4.2.1 GDFs survey

In order to provide a guide to choose a GDF for GDBL, we classify GDFs into two categories: GDFs for novices, and GDFs for developers. The main focus of GDFs for novices, including non-programmers, is to provide visual methods for customizing game templates and to allow creating or designing games with little or no programming skills. The main focus of GDFs for developers is to offer toolkits that support development of high quality 2D/3D rendering, special effects, physics, animations, sound playback, and network communication in common programming languages, such as C++, C#, and Java. The 34 articles are classified into Table 2 and Table 3 according to the GDFs used in the study. The unspecified GDFs or general SDK have been excluded, e.g. the articles R10, R11, R14, D24, D25, T28.

TABLE 2.
STUDY OF GDFs FOR NOVICES

GDFs	Features Description	Origin
Alice (http://alice.org)	Alice provides a point-and-click programming interface allowing creation of simple 3D games and animations. It is a tool for teaching object-oriented programming through creating simple games or animations.	R20
Scratch (http://scratch.mit.edu)	Scratch provides a point-and-click programming interface to create media-rich games, animations and applications for the Web. Scratch is suitable for teaching children basic programming (variables, arrays, logic, and user interface), and for creating simple 2D quick-and-dirty applications.	R4, R17
Greenfoot (www.greenfoot.org)	Greenfoot is a solid tool that provides many of the needed constructs for creating 2D computer games at a level that is especially appropriate and fun for novice programmers.	R6
Maya/ Photoshop/Flash	They are mainly used for art design to create digital characters and animations for games. Flash could also create Flash-games.	R8, D23
Game maker (www.gamemaker.nl)	Game Maker is a rapid-application development tool for young people at home and in schools to create two-dimensional and isometric games.	R5, R12
StarLogo TNG	StarLogo TNG is designed upon the basic framework of Logo. The programming is done with programming blocks instead of text commands, and moved programming from abstract to visual.	D22
Game editor: Warcraft3 Editors/ NeverWinter Night toolsets	The editor provides a simple GUI for customizing game templates, and requires little or no programming skills to create interesting game designs. The editors are implemented as visual programming tools that allow users to visually customize game behavior, including character behavior, game map, and game play.	R9, D26
Game platforms: Bomberman /Wu's Castle/ Critical Mass board game/quiz- based web game shell	These are concrete games, but provide visual interface for the users to modify or add basic code to change the game scenarios.	R7, R13, R18, D27

TABLE 3.

STUDY OF GDFs FOR DEVELOPERS

GDFs	Features Description	Origin
FPS game engine: Torque game engine /Unreal Engine	These are original commercial game engines and already have applied in commercial and popular games. They are usually not free and provide with some edit tools. And more complex than a concrete game editor.	R1, R8, R9,
XNA (www.xna.com/)/ XNACSILib framework/ XQUEST/ BiMIP	These are game development tools based on MFC and DirectX from windows platform and have same structure on game loop concept. BiMIP is a self-Made similar to XNA. And XNA is a GDF to develop cross-platform games for the Windows PC, Windows mobile phone, XBOX and the Zune platform using the C#. XNA features a set of high-level APIs targeted for 2D and 3D games. It consists of an integrated development environment (IDE) along with several tools for managing audio and graphics. XQUEST and XNACSILib are game library for XNA that contains convenient game components.	R3, R15, R16, D21, Dev30, Dev 33
Android/Sheep (www.android.com)	The Android mobile platform is a mobile application development platform issued by Google. And Sheep framework is an extended game library for Android.	Dev31
Simulation platforms: Spacewar simulator/ RoboRally/ JGOMAS MUPPETS/ SIMPLE framework	There are self-made simulation game or simulator that provide the controller for the users to modify the parameters and control the avatar in these simulation platforms, they usually to teach the programming and AI field.	R2, R19, Dev29, Dev32, Dev 34

In addition, one mature GDF selected from step 3 in Figure 1 could be a backup for novices -- CeeBot Series² [35]. The programming language in CeeBot is very similar to Java, C++ and C#. It has been developed especially to make learning programming easier. "CeeBots4 School" is a programming course for middle and high school.

4.2.2 Criteria for selection of suitable GDFs

Choosing a GDF is considered to be an important procedure during the preparation work for teaching. This process can be described by the following steps: a) Finding various GDF candidates. b) Analyze each GDF's features. c) Make criteria to filter GDF candidates, and choose one or more GDFs that fit best with the course content. Although our literature survey shows that different course aims have different requirements for the selection of the GDFs, there are still some common points to share. The article D21 presents a general criteria to choose a suitable GDF for the education in terms of theory - "What makes learning to be fun" by Malone [82]: e.g. easy to learn, allow rapid development, and provide an open development environment to attract students' curiosity. R1 presents that the GDFs should be chosen based on its cost and license, quality, difficulty, textbooks for guidance, and

² <http://www.ccebot.com/ccebot/family-e.php>

its main functionality. The article D26 explains that their students were not to become experts in programming, and thus they chose GDFs for novices. The article D27 introduces their self-made GDF and assess their own GDF by comparing it with other GDFs in terms of interactive, amusement, easy to use, using official program language, combine with teaching materials, evolutionary learning mode, census analysis, and storylines. The article Dev31 chose the GDF based on analysis of development environment, tutorial documents, emulator, programming language requirements, test devices, interface of the GDF, and possible ways to share games. Further both articles R3 and Dev31 developed a library for the GDF to make it more suitable for the course context. If we face the condition of only one choice, article D22 presents their effort to improve the only GDF. The article D23 presents how they compared different versions of same GDF, and made a choice between the newest version with powerful functions or old version but more stable.

To summarize, there are common and essential guidelines when selecting the GDFs: (a) *Technical environment and inexpensive (low costs) to use and acquire*: The technical environment requirements include required operating system and hardware, what tools are provided, are third-party tools supported and how difficult it is to install GDF. A typical problem can be e.g., that XNA runs only on Windows, and many students now have PCs running Linux or Mac OS X. The technical requirements might also be an economical issue, as the choice of GDF might force hardware upgrades or paying for licenses. (b) *Sufficient documentation to guide the usage of the GDF*. Students need to explore the GDF as an extra task before they start game development on the GDF. If the resources and materials are sufficient and easy to acquire for beginners, it will help them shorten the time spent on learning the technical environment. Time is an important factor during the whole teaching process, which will be further discussed in Section 4.3. (c) *Meets the students programming technique contexts*. The GDF must be easy to learn and allow rapid development. This issue is also driven by time constraints. Usually, if learning the GDF is not the major educational goal in the course and only an aid to learn something else, learning a new GDF will steal time from the course schedule. An easy and friendly environment is welcome in order to save time for the students and to keep the focus on the course content, and less on the GDF. (d) *Not in conflict with the educational goals of the course, flexibility to combine a GDF with teaching materials and possible to add/change libraries that can be used within the GDF*. All GDFs have constraints related to course content in how they have been designed or how they are released. One example is in SE education where open source GDFs make it possible to do white-box testing on the GDF, while the source code for other

GDFs might not be available. Further, some GDFs might have constraints on how you can design your games, what design and architectural patterns you can use, how event-handling must be managed, the freedom of expanding the GDFs functionality and more. These constraints must be integrated in the SE teaching to introduce the students to the real world where software rarely is built from scratch. In addition, if GDFs are not easy to use, and not strongly relevant to the course content, we can add/change a library with a user guide to apply course content in the GDF. (e) *Using an official programming language*. Conditionally, it applies to the types of GDFs for developers using commercial game engine with widely known programming languages, like C#, Java and C++, which are familiar to the students. But for the types of GDF for novices, if the course just lets students know the data structure, an official language is not really needed. But special programming languages are not widely accepted and as useful as official programming languages in a long run if the students will do more software programming in the future. (f) *Amusement and interactive*. The GDF should provide a visual and stable development environment to attract students' curiosity and engagement. A game development assignment in a user-friendly game development environment could be a good motivation for the students compared to traditional assignments. For example, most students think it is more interesting to work on a game project than e.g. a system for a bank. (g) *Ability to develop games in a cross-platform environment*. Conditionally, it applies to the types of GDFs for developers. One good example is XNA where the students can choose developing their games either in PC, mobile (Windows Phone 7), and/or console (Xbox360). Other game engines such as Unity3D also allow developing the game in multiplatform. The advantages are: (1) Provide students degrees of freedom in developing their games for the platform of their choice, and (2) Learn about the strengths and constraints of different platforms (e.g. user interface, viewing screen size, resolutions, resources such as memory and processor power, storage for saving/loading the game, and etc.) in game development.

We consider the above to be the most important criteria to guide the teachers in selecting one or more GDFs for their courses. And some criteria could be changed according to the specific context of the teaching environments. For instance, the target students are middle school pupils and the course goal is to let students familiarize themselves with information technology, it is not necessary to choose (e). In principle, the course aims and students context are the two fundamental and prioritized attributes to decide the selection of GDFs.

4.3 Evaluation

Besides the pedagogical analysis and the GDFs'

analysis in GDBL, this section summarizes the evaluation data from the articles mainly in the "Research" category. Furthermore, we hope to find empirical evidence to support the effectiveness of GDBL. Specifically, in order to approach the third aim of the study, the following information was drawn from each article (if provided in the article): (a) Major empirical findings related to the actual effectiveness of GDFs used as an aid in teaching, and (b) Factors that impact the teaching outcome in terms of the experiment data from the articles are also posed. It is not a simple process to assess the effectiveness of GDBL, and it covers at least two aspects: Teachers' and students' satisfaction of using this method. The teachers' concerns are the researcher's understanding of the course (not applied where the researchers and the teachers are the same), the GDFs' features, matching between the selected GDFs and the course content, and teachers' expertise on games. The students' concerns involve having interesting exercises and the difficulty of learning extra content – the GDF. Our literature review focuses on these aspects and Table 4 shows a summary of the evaluation process of GDBL in each article. Comparing the students' and teachers' satisfaction, students' satisfaction could be the most important result since directly relates to teaching effectiveness. And the following results are extracted from the literature and used to validate the effectiveness of GDBL and the impact factors related to it. The results in Table 4 are mainly shown in three categories: (a) Experiment Data that describes the collected data and materials for the measurement of the effectiveness of the results, (b) Conclusion of effectiveness of GDBL, and (c) Impact Factor that describes the elements that effect outcomes, and is classified into positive, neutral and negative categories based on the articles' data and conclusions.

From evaluation data in Table 4, the common expressions of measurements are: (a) Students' grade or score on the course exam. (b) Project results, including analysis of project size and classes they used in game programming; obtaining certain requirements of exercises by percentage; length of codes; percentage completed of the projects and time spent on the projects or the GDF, etc. (c) Questionnaire surveys to measure following aspects: students' satisfaction about the exercise, course and GDF; students background; students' interest in game development topic; course and exercise learned and open questions to get suggestions for the improvement of a course, etc. (d) Observation and feedback to perceive the fluency of the teaching process and interaction between students and the teacher.

From Table 4, the effectiveness of each article is collected. Generally, 22 of 23 articles have positive conclusion about using game development in a course in most of aspects, e.g. student motivation, engagement in lectures and exercises. Only the article R5 presents that

learning by game design did not have the expected outcome, and that the time constraint was a critical issue. Students indicated that they needed more time than two weeks to write a satisfactory 2D game. And finally, it explained that they did not have an adequate number of participants to have an accurate picture about the effects of game design on students' motivation and attitudes.

Apart from validating the effectiveness of using GDBL methods, the impact factors that could cause positive or negative outcome deserve to be analyzed. From Table 4, we have summarized what should be noticed when applying GDBL. The following items are the most common issues that appeared repeatedly in our survey:

1) *Communication between the researcher and teacher towards the understanding of the course content*: This item is not applied to the condition where the teacher and the researcher is the same person. If the researcher designs the method and the researcher invites the teachers to adopt it in schools, good communication and mutual trust between them are crucial to achieving the desired effect. The article R15 states that the teachers should become comfortable with using GDBL and spend a bit more time on it compared to traditional method in a certain course, otherwise it may cause a misunderstanding or bias against GDBL. Another aspect is that the researcher may worry about the teacher not totally understands the game effectiveness in education, and how game motivation can be successfully be used to improve the course design, which is mentioned in the articles R3, R6, R15, and R18. This indicates that the researchers should help teachers in gaining self-confidence, and provide constant support while the decision is made to apply GDBL in the curriculum.

2) *Teamwork*: This factor could have both positive and negative effects on the teaching results if students work in groups. First, the team size and working environment must be considered in advance. For instance it was found in article R10 that a big team size could have positive impact on outsourcing course teaching, and article R1 claims that Lab environment with teamwork could help improving the effectiveness of cooperative learning. On contrary, as the team gets larger, it becomes more difficult to set the time for general meetings and joint work hours. Further, it also means complex relations in a large group. A serious issue - bottleneck could happen in the game development process. If one member of the team does not perform, then the entire game development process slows down. Second, instant communication in a team has significant impact. Article R2 mentions that group work can help weaker students. Article R12 also agrees with this statement, but it describes that unexpected situations can occur during the teamwork to hinder the instant communication which the teacher should take care of. Third, the R14 article concludes that students need more experience in working effectively in teams. Most of the case studies

found in the articles provide the evidence that teamwork can be used together with GDBL and the nature of teamwork is suitable for cooperative learning and teacher should take care of the issues that may happen during teamwork. However, most of articles did not mention the strategy of competitive learning in GDBL. Only the articles R1 and R13 apply both cooperative and competitive learning in the exercises with a positive feedback in both cases.

3) *GDF relevance*: The most mentioned aspects related to GDFs that impact the outcome are: (a) The articles R2, R3, R12, and R15 present the advantages of using interactive graphical GDFs. It shows that visual graphics can provide instant feedback, making student engaged in programs, (b) The articles R3 and R4 describe how a GDF can improve students' confidence in programming tasks, and (c) The articles R1, R8, R9 R17, and R19 emphasize the need to analyze of the GDF's features in the light the course content, and detailed GDF tutorials should be conducted before it is used in the later exercises.

4) *Students' background*: In the article T28 surveys, the students' background was that most of them had played games as they were growing up. This is a suitable prerequisite to apply GDBL. But a negative aspect is the addictiveness to games, as mentioned in the articles R16 and R17. Some students may focus too much on the game and game development thus losing focus on what they shall learn in the course. This means that the design of the course and the project must be carried out in such a way that the students are forced to learn and use course content. From the articles R5 and R11, it was also noticed that the diversity of student background causes some difficulty of using GDBL. For instance, the programming experience of the students strongly affects the choice of GDF between the ones for novices and the ones for developers. For instance, to use XNA/XQUEST or Android/Sheep from Table 3 for developers, the students must know Object-Oriented (OO) programming well and be familiar with OO design patterns and OO principles. And some other GDFs require learning a specialized and simplified programming language for game creation, which is more suitable for students without programming experiences.

5) *Teachers' requirements*: Teachers' attitude of applying the GDBL method in the course is an essential aspect in a teaching process. The articles R3, R6, and R15 suggest that the faculty should have relevant technical background about the applied GDFs. The article R14 also mentions they should prepare and solve the anticipated

problems they may face during teaching. It is essential that the course staff have technical experience in the selected GDF to provide help for students and to avoid the focus shifting from the course content to technical matters.

6) *Time constraints and workload*: This problem has been stressed repeatedly in several articles. Most of articles found that the time was limited. For instance, the article R5 mentions that time constraint caused to cut down the time in beginning phase. The article R13 reports that some students complain about insufficient time to complete the project. So there are some advices correspondingly, like the article R18 proposes some suggestions on the time-consumption, and the article R3 suggests reading the background material better before the class in order to save class time for students. To help with the time management, a comprehensive time schedule should be prepared in advance for both the teacher and the students. Specifically, a series countermeasures can be: 1) Make sure that the students learn, understand, and apply the GDBL-project process; 2) Force students to set a mandatory rule for teams to create the schedule (strict milestones and deadlines); 3) Get involved with the students early to make sure that they make a realistic goal; 4) Teacher continuously monitor their progress and guide them to make adjustments, if needed, in order for them to complete their projects.

Other atypical factors could be found in Table 4. Further, Section 4.3 also provides a reference of how to assess the GDBL method. This indicates that future evaluation data of using GDBL is also beneficial, e.g. [83]. As it not only reveals the efficiency of using the framework along with how much the students actually learn from game projects, but also the social relationships' investigation of learner-learner, learner-teacher and teacher-researcher.

5 CONCLUSIONS

From the above findings, we summarize a guideline for integrating a GDF in learning with teaching strategies. Figure 5 shows a simplified diagram that gives an overview of the design process of applying GDBL (adapted from article D21 and Section 4.1.2). It contains four elements (Course aim, Pedagogical theory support, GDF resource pool, and Impact factor), two methods (learning by creating and learning by modifying games) and six steps in the teaching process and two subjects (students and teachers).

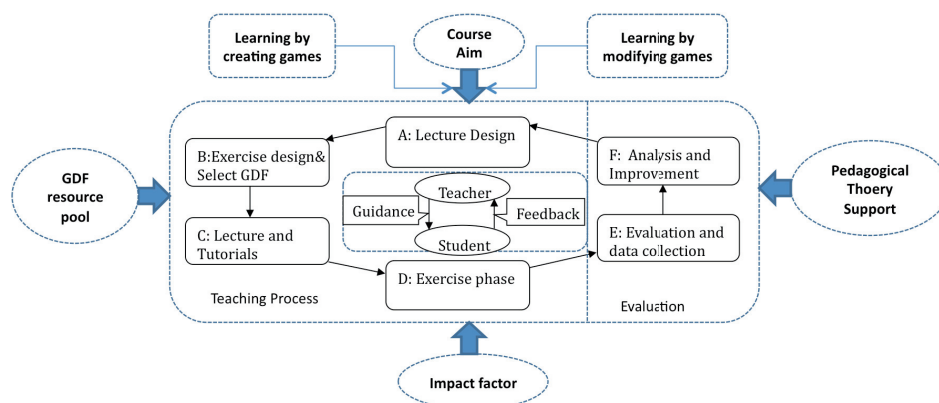


Fig. 5. A guideline for technical and pedagogical co-design of GDBL

Basically, the course aim has the fundamental affects on the selection of GDF. And the pedagogical theory (Section 4.1.1) could support the teaching design. The GDF resource pool (Section 4.2.2) could be the reference for the selection of GDFs. Usually, during steps A to B in the teaching process in the Fig. 5, pedagogical theory support and GDF resource pool play important roles in these two initial steps. Impact factors concern the whole process, but we suggest considering them at beginning as well. In terms of the course aim, pedagogical theory support and GDFs resource pool, the teaching process (Section 4.1.2) starts with designing the lectures and exercises with the selected GDF. After the lectures and tutorials, the course delivery starts and students begin the design and implementation of their projects. For the evaluation framework (Section 4.3), teachers/researchers are suggested to collect data using surveys. Based on the analysis of collected and teaching experiences, they can improve the teaching process framework. Here, we use a compact case to explain how each element in Figure 5 works in a certain course if the GDBL method is applied. The assumption is that the course aim is to teach basic programming rules for beginners. The choice could be made between "learning by modifying games" using a game editor with scripting, or "learning by creating games" using a GDF for novices. Then, we should consider the relationships between the problems and tools from the perspective of double stimulus or use other pedagogical theories to construct the learning process, for example PBL. With this in mind and according to criteria in Section 4.2.2, commonly used tools can be selected from the GDF resource pool - GDFs for novice in Table 2 or use another GDF if no suitable GDF is found in Table 2. After finishing steps of A to B in teaching process we start the lecture and the introduction of both exercises and GDFs. Later, students commence the implementation individually or in groups. During the whole teaching process from A to D, the impact factors

are relevant but optional. For instance, we can choose a graphical interactive GDF, and estimate time to be spent on lectures and exercises. Applying the impact factors in the teaching process depends on the courses' situations. That is why we have the evaluation and analysis steps E and F in Figure 5. The feedback data can help to validate the choice in each step - whether we choose a right task or a suitable GDF or focus on the most relevant impact factors in a course. In addition, since many elements interact in GDBL, which makes the real situation more complex to analyze and evaluate. Thus, an effective evaluation helps to validate the whole teaching process, and it is not only judged by teachers' own experiences, but also get opinions from students' aspect.

From the experience of accomplishing this literature review, we still have the following limitation: (a) The scope of data search and collection from four scientific search engines is relative limited; (b) Due to the game research field is younger than other traditional research fields, amount of articles with empirical data is still limited in our the survey, it maybe cause the pitfall of the evaluation results, e.g. generalization; (c) Some topics deserve further discussion. E.g. Cross-disciplinary courses, like game development course in article R1 covers programming and art design, and machinima course in article R8 have 3D animation and movie creation. Both of them could be further discussed since GDFs plays different roles -- main tool in article R1 and an innovative auxiliary in article R8.

This study has shown that GDBL do have the potential power to help students to learn different curriculums. We hope that the study will provide useful guidance to educators, practitioners and researchers in the area of GDBL, as well as to GDF designers, and that it will inform their future professional practices and research.

REFERENCES

- [1] S. M. Dorman, "Video and Computer Games: Effect on Children and Implications for Health Education," *Journal of School Health*, vol. 67, pp. 133-138, 1997.
- [2] M. Prensky, "Digital game-based learning," *Computers in entertainment*, vol. 1, pp. 21-24, 2003.
- [3] J. Blow, "Game Development: Harder Than You Think," *Queue*, vol. 1, pp. 28-37, 2004.
- [4] A. I. Wang and B. Wu, "An Application of a Game Development Framework in Higher Education," *International Journal of Computer Games Technology*, vol. 2009, 2009.
- [5] K. Sung, et al., "Game-Themed Programming Assignment Modules: A Pathway for Gradual Integration of Gaming Context Into Existing Introductory Programming Courses," *IEEE Transactions on Education*, 2010.
- [6] G. Sindre, "Experimental validation of the learning effect for a pedagogical game on computer fundamentals," *IEEE transactions on education*, vol. 52, p. 10, 2009.
- [7] B. A. Foss and T. I. Eikaas, "Game Play in Engineering Education Concept and Experimental Results," *International Journal of Engineering Education*, vol. 22, pp. 1043-1052, 2006.
- [8] A. I. Wang, et al., "LECTURE QUIZ - A Mobile Game Concept for Lectures," presented at the 11th IASTED International Conference on Software Engineering and Application (SEA 2007), 2007.
- [9] A. I. Wang, "An Evaluation of a Mobile Game Concept for Lectures," presented at the IEEE 21st Conference on Software Engineering Education and Training, 2008.
- [10] M. S. El-Nasr, "Learning through game modding," *Computers in entertainment*, vol. 4, 2006.
- [11] G. Lukas, "Uses of the LOGO programming language in undergraduate instruction," presented at the Proceedings of the ACM annual conference - Volume 2, Boston, Massachusetts, United States, 1972.
- [12] M. Micco, "An undergraduate curriculum in expert systems design or knowledge engineering," presented at the Proceedings of the 15th annual conference on Computer Science, St. Louis, Missouri, United States, 1987.
- [13] J. P. Higgins and S. Green, *Front Matter*: John Wiley & Sons, Ltd, 2008.
- [14] K. S. Khan, et al., *Undertaking systematic reviews of research on effectiveness: CRD's guidance for carrying out or commissioning reviews: CRD report, Number 4, second ed.*, NHS centre for reviews and dissemination, University of York, 2001.
- [15] M. Papastergiou, "Exploring the potential of computer and video games for health and physical education: A literature review," *Computers & Education*, vol. 53, pp. 603-622, 2009.
- [16] J. Kirriemuir and A. McFarlane, "Literature review in games and learning," Report 8, 2004.
- [17] Y. En, et al., "Enhancing software engineering education using teaching aids in 3-D online virtual worlds," in 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, (FIE '07) 2007, pp. T1E-8-T1E-13.
- [18] B. Wu, et al., "Experiences from Implementing an Educational MMORPG," in International IEEE Consumer Electronics Society's Games Innovations Conference (GIC 2010), 2010.
- [19] A. Baker, et al., "Problems and Programmers: an educational software engineering card game," in Proceedings. 25th International Conference on Software Engineering, 2003, pp. 614-619.
- [20] F. McCown, "Teaching a game programming class for the first time: tutorial presentation," *Journal of Computing Sciences in Colleges*, vol. 25, pp. 131-132, 2010.
- [21] C. Leska and J. Rabung, "Learning O-O concepts in CS I using game projects," *SIGCSE Bull.*, vol. 36, pp. 237-237, 2004.
- [22] E. Ferguson, et al., "Video game development using XNA game studio and C#.Net," *Journal of Computing Sciences in Colleges*, vol. 23, pp. 186-188, 2008.
- [23] R. H. Seidman, "Alice first: 3D interactive game programming," *SIGCSE Bull.*, vol. 41, pp. 345-345, 2009.
- [24] F. Xiang, et al., "Work in progress: A sandbox model for teaching entrepreneurship," in 2010 IEEE Frontiers in Education Conference, 2010, pp. F2C-1-F2C-2.
- [25] M. Kolling, "Greenfoot: introduction to Java with games and simulations," *Journal of Computing Sciences in Colleges*, vol. 25, pp. 117-117, 2010.
- [26] A. Azemi and L. L. Pauley, "Teaching the introductory computer programming course for engineers using Matlab," in 38th Annual Frontiers in Education Conference (FIE 2008), 2008, pp. T3B-1-T3B-23.
- [27] A. Pardo and C. D. Kloos, "Deploying interactive e-labs for a course on operating systems," presented at the Proceedings of the 6th conference on Information technology education, Newark, NJ, USA, 2005.
- [28] P. Rooney, et al., "Cross-Disciplinary Approaches for Developing Serious Games in Higher Education," in Conference in Games and Virtual Worlds for Serious Applications, 2009 (VS-GAMES '09) 2009, pp. 161-165.
- [29] A. W. B. Furtado, et al., "Cegadef: a collaborative educational game development framework," presented at the Proceedings of the 2003 conference on Interaction design and children, Preston, England, 2003.
- [30] H. C. Yang, "A General Framework for Automatically Creating Games for Learning," in Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05), 2005.
- [31] K. Kardan, "Computer role-playing games as a vehicle for teaching history, culture, and language," presented at the Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames, Boston, Massachusetts, 2006.
- [32] S. Arakawa and S. Yukita, "An Effective Agile Teaching Environment for Java Programming Courses," in 36th Annual Frontiers in Education Conference., 2006, pp. 13-18.
- [33] W. W. Y. Lau, et al., "Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students," *SIGCSE Bull.*, vol. 41, pp. 504-508, 2009.
- [34] S. v. Delden, "Industrial robotic game playing: an AI course," *J. Comput. Small Coll.*, vol. 25, pp. 134-142, 2010.
- [35] T. Phit-Huan, et al., "Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception," in International Conference on Computer Technology and Development, 2009 (ICCTD '09), 2009, pp. 42-46.
- [36] T. E. Daniels, "Integrating engagement and first year problem solving using game controller technology," in Frontiers in Education Conference, 2009. FIE '09. 39th IEEE, 2009, pp. 1-6.
- [37] A. Striegel and D. Van Bruggen, "Work in progress: Development of a HCI course on the Microsoft Surface," in 2010 IEEE Frontiers in Education Conference, 2010, pp. S3F-1-S3F-6.
- [38] A. Wang, "Interactive Game Development with a Projector-Camera System," in Technologies for E-Learning and Digital Entertainment. vol. 5093, ed: Springer Berlin / Heidelberg, 2008, pp. 535-543.
- [39] J. Dempsey, et al., "The instructional gaming literature: Implications and 99 sources.," Technical report no. 96-1. University of South Alabama, College of Education, 1996.
- [40] J. Dempsey, et al., "Since Malone's theory of intrinsically motivating instruction: What's the score in the gaming literature?," *Journal of Educational Technology Systems*, 22(2), 173-183, 1993-1994.
- [41] R. Hays, "The effectiveness of instructional games: A literature review and discussion," Technical report 2005-004. Orlando, FL: Naval Air Warfare Center, Training Systems Division, 2005.
- [42] M. C. v. Langeveld and R. Kessler, "Two in the middle: digital character production and machinima courses," *SIGCSE Bull.*, vol. 41, pp. 463-467, 2009.
- [43] W. L. Honig and T. Prasad, "A classroom outsourcing experience for software engineering learning," *SIGCSE Bull.*, vol. 39, pp. 181-185, 2007.
- [44] S. Hrastinski, "What is online learner participation? A literature review," *Computers & Education*, vol. 51, pp. 1755-1765, 2008.
- [45] A. D. Ritzhaupt, "Creating a Game Development Course with Limited Resources: An Evaluation Study," *ACM Transactions on Computing Education*, vol. 9, pp. 1-16, 2009.
- [46] A. McGovern and J. Fager, "Creating significant learning experiences in introductory artificial intelligence," *SIGCSE Bull.*, vol. 39, pp. 39-43, 2007.
- [47] R. Angotti, et al., "Game-themed instructional modules: a video case study," presented at the Proceedings of the Fifth International Conference on the Foundations of Digital Games, Monterey, California, 2010.

- [48] G. Fesakis and K. Serafeim, "Influence of the familiarization with 'scratch' on future teachers' opinions and attitudes about programming and ICT in education," presented at the Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education, Paris, France, 2009.
- [49] Y. Rankin, et al., "The impact of game design on students' interest in CS," presented at the Proceedings of the 3rd international conference on Game development in computer science education, Miami, Florida, 2008.
- [50] M. Al-Bow, et al., "Using game creation for teaching computer programming to high school students and teachers," SIGCSE Bull., vol. 41, pp. 104-108, 2009.
- [51] M. Eagle and T. Barnes, "Experimental evaluation of an educational game for improved learning in introductory computing," SIGCSE Bull., vol. 41, pp. 321-325, 2009.
- [52] W. K. Chen, "Teaching object-oriented programming laboratory with computer game programming," IEEE transactions on education, vol. 50, p. 197, 2007.
- [53] Yulia and R. Adipranata, "Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment," in 2nd International Conference on Education Technology and Computer (ICETC), 2010, pp. V2-355-V2-359.
- [54] R. Lawrence, "Teaching data structures using competitive games," IEEE Transactions on Education, vol. 47, pp. 459-466, 2004.
- [55] J. Huang, "Improving undergraduates' teamwork skills by adapting project-based learning methodology," in 5th International Conference on Computer Science and Education (ICCSE), 2010, pp. 652-655.
- [56] B. Wu, et al., "An Evaluation of Using a Game Development Framework in Higher Education," Proceedings / Conference on Software Engineering Education and Training, 2009.
- [57] J.-F. Weng, et al., "Teaching Boolean Logic through Game Rule Tuning," IEEE Trans. Learn. Technol., vol. 3, pp. 319-328, 2010.
- [58] R. Owston, et al., "Computer game development as a literacy activity," Computers & Education, vol. 53, pp. 977-989, 2009.
- [59] I. Timm, et al., "Teaching Distributed Artificial Intelligence with RoboRally," in Multiagent System Technologies. vol. 5244, ed: Springer Berlin / Heidelberg, 2008, pp. 171-182.
- [60] L. Werner, et al., "Can middle-schoolers use Storytelling Alice to make games?: results of a pilot study," presented at the Proceedings of the 4th International Conference on Foundations of Digital Games, Orlando, Florida, 2009.
- [61] K. Wang, et al., "3D game design with programming blocks in StarLogo TNG," presented at the Proceedings of the 7th international conference on Learning sciences, Bloomington, Indiana, 2006.
- [62] H. Chun-Hsiung, et al., "Computer Game Programming Course for Art Design Students by Using Flash Software," in 2008 International Conference on Cyberworlds, 2008, pp. 710-713.
- [63] B. Lennartsson and E. Sundin, "Experience from a course aiming at understanding system development with focus on system design and integration," in Frontiers in Education, 2002. FIE 2002. 32nd Annual, 2002, pp. T3G-1-T3G-6 vol.1.
- [64] J. Ryoo, "Teaching object-oriented software engineering through problem-based learning in the context of game design," in 21st Conference on Software Engineering Education and Training, 2008, p. 137.
- [65] J. Robertson and C. Howells, "Computer game design: Opportunities for successful learning," Computers & Education, vol. 50, pp. 559-578, 2008.
- [66] W.-C. Chang and Y.-M. Chou, "Introductory C Programming Language Learning with Game-Based Digital Learning," in Advances in Web Based Learning - ICWL 2008. vol. 5145, ed: Springer Berlin / Heidelberg, 2008, pp. 221-231.
- [67] S. Kurkovsky, "Can mobile game development foster student interest in computer science?," in International IEEE Consumer Electronics Society's Games Innovations Conference, (ICE-GIC 2009), 2009, pp. 92-100.
- [68] K. J. Bierre and A. M. Phelps, "The use of MUPPETS in an introductory java programming course," presented at the Proceedings of the 5th conference on Information technology education, Salt Lake City, UT, USA, 2004.
- [69] B. Wu, et al., "XQUEST used in software architecture education," in International IEEE Consumer Electronics Society's Games Innovations Conference, (ICE-GIC 2009), 2009, pp. 70-77.
- [70] B. Wu, et al., "Extending Google Android's Application as an Educational Tool," presented at the The 3rd IEEE Information Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL 2010), 2010.
- [71] H. C. Jiau, et al., "Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics," IEEE Transactions on Education, vol. 52, pp. 555-562, 2009.
- [72] A. Garrido, et al., "Using graphics: motivating students in a C++ programming introductory course," in EAEEIE Annual Conference,, 2009, pp. 1-6.
- [73] A. Barella, et al., "JGOMAS: New Approach to AI Teaching," IEEE Transactions on Education, vol. 52, pp. 228-235, 2009.
- [74] S. Papert, Mindstorms: Children, Computers, and Powerful Ideas. New York, 1980.
- [75] S. Puntambekar and J. L. Kolodner, "Toward implementing distributed scaffolding: Helping students learn science from design," Journal of Research in Science Teaching, vol. 42, pp. 185-217, 2005.
- [76] J. Dewey, Democracy and education: An introduction to the philosophy of education. New York, 2005.
- [77] J. Dewey, Experience and education. New York, 1997
- [78] E. Smeets, "Does ICT contribute to powerful learning environment in primary education," Computer and Education, pp. 343-355, 2005.
- [79] L. S. Vygotskii, Mind in society: The development of higher psychological processes, 1978.
- [80] H. S. Barrows, "A taxonomy of problem-based learning methods," Medical Education, vol. 20, pp. 481-486, 1986.
- [81] B. Lennartsson and E. Sundin, "Fronesis-the third dimension of knowledge, learning, and evaluation," in 31st Annual Frontiers in Education Conference, 2001, pp. T2B-14-19 vol.1.
- [82] W. M. Thomas, "What makes things fun to learn? heuristics for designing instructional computer games," presented at the Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, Palo Alto, California, United States, 1980.
- [83] A. I. Wang, "Extensive Evaluation of Using a Game Project in a Software Architecture Course," Transactions on Computing Education (ACM), vol. Volume 11,, February 2011. 2011.

TABLE 4
EVALUATION DATA COLLECTION AND IMPACT FACTOR

Title	Sample	Comparison	Experiment Data	Conclusion of Effectiveness	Outcome of Impact factor
R1	22 students	No	Quantitative data: 1) Questionnaire of student background 2) Survey project results of student game play preferences 3) Questionnaire on student satisfactory 4) Questionnaire on interest level in game development careers 5) Questionnaire on student assessment of gains 6) Questionnaire on helpful course elements	Students generally satisfied the elements in the course and resources (including teamwork).	Positive: 1) Lab environment and teamwork helped to archive the effectiveness of cooperative learning. 2) Teaching game development required a shift from teacher-centered to student-centered learning environment. 3) GDFs provided an environment that students could integrate wide variety of skills and knowledge. 4) Motivation factor: competition. Negative:

			7) Student Peer-Evaluation		5) Poor textbook for GDF provided negative effect.
R2	33 students (28 undergraduates and 5 graduate)	No	Quantitative data: 1) General questionnaire Qualitative data: 2) Students feedback about the course	Generally, students enjoyed the project and it fulfilled all of the criteria of a successful project outlined at the beginning plan.	Positive: 1) Flexible and interactive simulation platform. 2) Providing examples for difficulty part in the project that was out of the course aim. 3) Group project and discussion helped weaker students. Neutral: 4) Difficulties at first year, but smoothed out by get more teaching experiences and previous evaluation for the improvement.
R3	21 undergraduates	No	Quantitative data: 1) General questionnaire Qualitative data: 2) Video recording about course process 3) Faculty feedback	The GDF was excellent catalysts, enabling faculty to begin exploring teaching with game topics and help students to be more engaged.	Positive: 1) Because of the immediate interactive graphical feedback, students were engaged and motivated to experiment with the programs. Neutral: 2) Instructor's attitude toward the interest in GDF. Negative: 3) Visual feedback, although a powerful learning tool, could also be a source of distraction for students. 4) Time spending should not involve the reading of background material in class (better before class). 5) Limited classroom time was challenging for students.
R4	35 female students from both preschool and university	No	Quantitative data 1) Questionnaire: student opinions about GDF 2) Questionnaire: effect of students familiarization with scratch in using of ICT education 3) Questionnaire (pre and post-test) attitudes against internet in education and application development	Scratch was user friendly and satisfied by the students, and it also has a rather positive effect on students' opinions and attitudes towards computer programming and ICT educational value in education.	Positive: 1) Scratch helped to setup confidence of students in exploration of ICT in education.
R5	20 undergraduates	No	Quantitative data: 1) Questionnaire: Likert-scale (pre and post survey in game design course)	Game design had both positive and negative impact on students' attitudes about computer science, game design and further development of programming skills.	Positive: 1) Students who had prior programming experience can express interest in game design. Negative: 2) Time constraints: assignment might be better received and increase student interest if students were given more time and equal emphasis on other phases. 3) Game design topic course had a negative impact on students' interest in pursuing a CS degree. 4) Not adequate number of participants to have an accurate picture of true effects of game design on students' motivation and attitudes.
R6	26 high school students and 8 teachers	Yes	Quantitative data: 1) Questionnaire: Assignment survey 2) Questionnaire with pre and post survey: self-assessment on art and design 3) Questionnaire survey on teachers' attitude	It showed great promise for engaging high school students programming and increasing interest in computer related fields of study. Both teachers and students felt a significant improvement in computer programming and self-confidence.	Positive: 1) Researchers trained both students and teachers by applying GDBL. Neutral: 2) Teacher attitudes and self-confidence about GDBL's effect the teaching process.
R7	26 students in experimental group. 29 in control group	Yes	Quantitative data: 1) Each phase of study 2) Pre and post-test score 3) Learning difference between groups and subgroups 4) Game statistics 5) Questionnaire survey of each task	Students in the game-first group felt they spent less time on assignments and all students preferred the learning game to the program.	Positive: 1) "Wu castle" was more effective than a traditional programming assignment for learning, and could help prepare students to create deeper, more robust understanding of computing concepts and improving their perceptions of homework.
R8	--	No	Quantitative data: 1) Questionnaire to survey students feedback 2) Compared with whole school average score	Students got higher score in this course than school's average score.	Positive: 1) Assessing the GDF in the starting.
R9	26 students	No	Quantitative data: 1) Questionnaire to survey assignment	Using game development motivated students to learn	Positive: 1) GDBL could learn several subjects and concepts.

R9	26 students	No	Quantitative data: 1) Questionnaire to survey assignment difficulty with Likert-scale Qualitative data: 2) Observation of students progress	Using game development motivated students to learn and allowed them to apply and visualize the utility and application of the concepts.	Positive: 1) GDBL could learn several subjects and concepts. Neutral: 2) Different game engines implicitly stressed the use and development of certain skills.
R10	40 undergraduate students	No	Quantitative data: 1) Pre and post-test questionnaire to survey: Changed perception of outsourcing concept 2) Questionnaire: SE outcomes Qualitative data: 3) Observation: Discoveries in communications	Students improved their understanding of outsourcing, developed better appreciation for the importance of SE techniques, and created ad-hoc communication protocols between teams.	Neutral: 1) Enlarging the teams' sizes to other universities to create an inclusive teaching environment, which had limitation that only applied in outsourcing teaching.
R11	38 students (19 teams)	No	Quantitative data: 1) Length of codes according to grade 2) Project size and classes 3) Methods used in programming 4) Weekly working hours 5) Proportion of work: discussion, coding, thinking, graphics, audio 6) Object-Oriented skills applied in code.	Positive experience had been gained in teaching the topic by using game framework.	Neutral: 1) To keep the students motivated, and teachers tailored the course for each student. 2) Using game development to achieve depth of objects and object interactions training.
R12	124 students	No	Quantitative data: 1) Grade 2) Questionnaire to survey students attitude	Learning by creating game was able to improve the student grades largely.	Positive: 1) Object-Oriented programming concept became easier to understand after seeing object design visually in the GDF. 2) Students felt happy with using cooperative learning system, games development and visual design. Negative: 3) The group members' communication was hindered by the in front of computers. 4) GDBL could help with the passing rate, but still have improving space for graduation aim.
R13	55 students	No	Quantitative data: 1) User survey of game project: percentage completed 2) Login times 3) Questionnaire with Likert-scale: Student satisfaction 4) Questionnaire with Likert-scale: Tournament features	Combination of game development and friendly student competition was a significant motivator for increased student performance.	Positive: 1) Tournament could increase student participation and motivation. Negative: 2) Students' common complaint of not having adequate time to complete the project.
R14	--	No	Quantitative data: 1) Individual and group creativity levels perceived by students 2) Students' perception of abilities developed at intermediate or high levels Qualitative data: 3) Future career survey	Game project development with collaborative learning was manageable and effective for increasing students' teamwork capability and increase the employability confidence.	Positive: 1) Project (game project development) based learning motivated their team collaboration. Negative: 2) Teacher attitudes: Initial resistance for problems that students teams faced could be discouraging to faculty members who did not expect it. 3) Teamwork: students were not born knowing how to work effectively in teams. A Flawed team-based instructional model had negative effect.
R15	CS1:22 in GTA and 10 in Console CS2 : 18 in GTA and 9 in Console	Yes	Quantitative data: 1) Success rate (Passing rate) 2) Assignment score 3) Self-reported time spent on assignment 4) Post Assignment Survey 5) Pre and Post course survey Qualitative data: 6) Feedback from faculty	Interactive graphical assignments could be a good tool for teaching CS1 students. The success of GDBL hinged on the instructor's expertise and enthusiasm.	Positive: 1) GDF feature: interactive graphical application supported experimentation and visualization. Negative: 2) Teacher's background and attitudes towards the games impacted the output of a lecture, faculty "dropped" GDBL in the end at first experiment, but became more comfortable later.
R16	46 students	No	Quantitative data: 1) Questionnaire about learning process, trade-off between technical and architecture problems, integration of game development and course, learning outcome.	GDF was easy to use and not conflict with course aim. A good GDF could save development time.	Neutral: 1) GDF selection influenced learning process and extra technical issues, but students could learn a lot through a game project.
R17	27 in control group, 43 in experimental group	Yes	Quantitative data: 1) score of the pre- and post-test by a test sheet.	Results showed the proposed game development activity could have higher learning achievements compared to the traditional lecturing.	Positive: 1) GDF issues: choosing modifying game according to course topic with simple scenario. And tutorials for GDF were prepared well. Understanding game topic could make engage learning.

R18	125 experimental students and 186 control group students	Yes	Quantitative data: 1) GRADE test scores (pre-test, post-test) Qualitative data: 2) Interviews on teacher's feedback	Game development helped to improve student content retention, etc.	Positive 1) Optimum amount of time to spend at a sitting on game development activities was about 45 min by observation. Negative: 2) Too little time allotted to the development of game and insufficient gaps between each game creation activities.
R19	33 undergraduate	No	Quantitative data: 1) Questionnaire with Likert in general	Using GDBL indicated the motivation of the students was higher and they understand complex problems easier and exercise could be done more rapidly.	Positive: 1) GDF was searched and chose based on the requirements.
R20	22 middle-schoolers	No	Quantitative data: 1) Questionnaire: pre and post surveys of participants information 2) Programs analysis. Qualitative data: 3) Daily log 4) Interviews on students	Findings suggested the middle school students could use Alice to make games to build information technology fluency.	Neutral: 1) To provide proper challenge in class. 2) Difficulty in using GDF to finish the assignment
T28	NA	No	Quantitative data: 1) Survey of students background 2) Relevant application about Mobile GDBL	Mobile game development could be successfully integrated into computer science education.	Positive: 1) Student background: student lived in game environment and game development exercise could be a good motivation.
Dev30	19 graduate	No	Quantitative data: 1) Questionnaire survey with Likert scale, and System Usability Scale survey	XQUEST enhanced XNA in suitability as a teaching aid in SE learning.	Positive 1) To design the XQUEST from the previous assessment experiences.
Dev32	57 in group1, 45 in group2	Yes	Quantitative data: 1) Questionnaire result of student user experience 2) Score for pre/post test	SIMPLE improved both learning motivation and programming skills for the students.	Positive: 1) Use GQM approach in developing game metrics for students' exercise.

Paper 12:

GDF8: Bian Wu, Alf Inge Wang, "Comparison of Learning Software Architecture by Developing Social Applications vs. Games on the Android Platform", *International Journal of Computer Games Technology*, Volume 2012, Article ID 494232, 10 pages, 2012. ISSN: 1687-7047 EISSN: 1687-7055. DOI: 10.1155/2012/494232

Comparison of Learning Software Architecture by Developing Social Applications vs. Games on the Android Platform

Bian Wu, Alf Inge Wang

Dept. of Computer and Science

Norwegian University of Science and Technology

Bian@idi.ntnu.no, Alf@idi.ntnu.no

Abstract: This article investigates how much the chosen application domain in a development project affects the learning and perception of a software architecture course. Specifically, it describes an empirical study where the focus was on discovering differences and similarities in students working on development of social applications vs. students working on development of games using the same Android development platform. In 2010-2011, students attending the software architecture course at the Norwegian University of Science and Technology (NTNU) could choose between four types of projects: Development of a robot controller using Java on the Khepera Robot Simulator, development of a game on the XNA platform, development of a game on the Android platform, and development of a social application on the Android platform. Independent of the chosen type of project, all students had to go through the same phases, produce the same documents based on the same templates, and follow exactly the same process. This study focuses on the Android projects, to see how much the application domain affects the course project independent of the chosen technology. Our results revealed some positive effects for the students doing game development compared to social application development to learn software architecture, like motivated to work with games, a better focus on quality attributes such as modifiability and testability during the development, production of software architectures of higher complexity, and more productive coding working for the project. However, we did not find significant differences in awarded grade between students choosing the two different domains.

Keywords: Game based learning, Game development based learning, Android, Evaluation, Software engineering education, Software Architecture

1 Introduction

Computer games and video games have become very popular for children and youths, and play a prominent role in the culture of young people [1]. Games can now be played everywhere in technology-rich environments equipped with laptops, smart phones, game consoles (mobile and stationary), set-top boxes and other digital devices. From this phenomenon, it is believed that the intrinsic motivation that young people show towards games could be combined with educational content and objectives into what Prensky calls “digital game based learning” [2].

Besides of an abundant appearance of games in young students life, game development technology has matured and become more advanced [3]. Based on various existing game development environments, the whole duty of game development process can be divided into several expert domains and roles such as game programmer, 3D model creator, game designer, musician, animator, play-writer, etc. The process of integrating game content with technology can be simplified through the usage of game engines and available information on the web from various user and expert communities. For instance, Microsoft’s XNA game development kit provides the game loop function to draw and update the game contents, and it also provides convenient game development components to load the different format of graphics, audio, and videos. This makes it possible for game fans such as students with or without programming background to modify existing games or develop new games. They can design and implement their own game concepts with these game creation tools, and learn the developing skills and relevant knowledge, and accumulate related practical experience.

In this context, not only can games be used for learning, but also the game development tools can be used for studying relevant topics within computer science (CS), software engineering (SE) and game programming through motivating assignments. Generally, games can be integrated in education in three ways [4, 5]. First, games can be used instead of traditional exercises motivating students to put extra effort in doing the exercises, and giving the teacher and/or teaching assistants an opportunity to monitor how the students work with the exercises in real-time, e.g. [6, 7]. Second, games can be played as a part of a lecture to improve the participation and motivation of students, e.g. [8, 9]. Third, the students are asked to modify or develop a game as a part of a course using a Game Development Framework (GDF) to learn skills within CS and SE, e.g. [10]. We label the latter learning approach Game Development-Based Learning (GDBL). And the GDF denotes the toolkits that can be used to develop or modify games, e.g. game engine, game editors, or game (simulation) platforms, or even any Integrated Development Environment (IDE), like Visual C++, Eclipse, J2ME, and Android SDK since all of them can be used to develop games.

This article focuses on an evaluation where we wanted to discover similarities and differences between making students learn software architecture through game development vs. social application development

(e.g. Weather Forecast, chatting software) using the Android platform. The motivation for bringing game development into a CS or SE course is to exploit the students' fascination for games and game development to stimulate them to work more and better with course material through the project.

2 Related works

This section describes the research context and previous results about using GDBL method in software engineering field.

2.1 Research contexts

The earliest similar application of learning by programming in a game-like environment was in early 1970s. The Logo [11], the turtle graphics, is one of the oldest libraries that was used to introduce computing concepts to beginners. The concept was based on a "turtle" that could be moved across a 2D screen with a pen, which could be positioned on or off the screen, and thus, may leave a trace of the turtle's movements. Programming the turtle to draw different patterns could be used to introduce general computing skill, such as procedural operations, iteration, and recursion. Further, in 1987, Micco presented the usage of writing a tic-tac-toe game for learning [12]. Afterwards, other studies have been conducted using specialist game programming toolkits such as Stage Cast Creator [13], Gamemaker [14], Alice [15] and Neverwinter Nights [16]. Besides, article [17] presents a investigation for using mobile game development as a motivational tool and a learning context in computing curriculum. From their survey, it shows the relation between game programming and other computer science fields – Game development can be used in study of Artificial intelligence (AI), database, computer networks, SE, human-computer interaction, computer graphics, algorithms, programming, computer architecture, and operating system.

These studies indicate that making games is motivating and develops storytelling as well as technical programming skills. The nature of the task of making games is slightly different in purpose-built environments, and the balance of the roles assumed by the learner shifts accordingly. More recent game programming toolkits tend to have a stronger visual aspect than Logo, either in the sense that they enable designers to easily create graphical games or because they have a visual programming language, or both. This shifts the emphasis away from low-level programming, enabling learners to focus on the other roles as designers or writers. Thus, we investigate how GDFs are used in education through an experiment study and explore the evolution of the traditional lecture to be dynamic, collaborative and attractive to the students under current technology rich environment. However, this assertion needs to be further supported by relevant theory, application experiences, evaluation results, and empirical evidence. This is one motivation for sharing our experiences and empirical results in field of GDBL on using Android in a software architecture course.

2.2 Course and Project setting

The software architecture course at Norwegian University of Science and Technology (NTNU) (course code TDT4240) is taught in a different way than at most other universities, as the students also have to implement their designed architecture in a project. The motivation for doing so is to make the students understand the relationship between the architecture and the implementation, and to be able to perform a real evaluation of whether the architecture and the resulting implementation fulfill the quality requirements specified for the application. The architecture project in the course has similarities with projects in other software engineering courses, but everything in the project is carried out from a software architecture perspective. Throughout the project, the students have to use software architecture techniques, methods, and tools to succeed according to the specified project.

The software architecture project consists of the following phases:

- i. COTS (Commercial Off-The-Shelf) exercise: Learn the technology to be used through developing a simple game.
- ii. Design pattern: Learn how to use and apply design pattern by making changes in an existing system.
- iii. Requirements and architecture: List functional and quality requirements and design the software architecture for a game.
- iv. Architecture evaluation: Use the Architecture Trade off Analysis Method (ATAM) [18] [19] [20] evaluation method to evaluate the software architecture of project in regards to the quality requirements.
- v. Implementation: Do a detailed design and implement the game based on the created architecture and on the changes from the evaluation.

- vi. Project evaluation: Evaluate the project as a whole using a Post-Mortem Analysis (PMA) method [21].

In the first two phases of the project, the students work on their own or in pairs. For Phases 3-6, the students work in self-selected teams of 4-5 students. Meantime, students have one fixed primary assigned quality attribute to focus on during the project. For the secondary quality attribute, students can choose the quality attribute they like. The students spend most time in the implementation phase (six weeks), and they are also encouraged to start the implementation in earlier phases to test their architectural choices (incremental development). During the implementation phase, the students continually extend, refine, and evolve the software architecture through several iterations.

2.4 Previous results

Previous, the goal of the project has been to develop a robot controller for the WSU Khepera robot simulator (Robot) in Java [22] with emphasis on an assigned quality attribute such as availability, performance, modifiability, or testability. The students were asked to program the robot controller to move a robot around in a maze, collect four balls and bring them to a light source in the maze. In 2008, the students were allowed to choose between a robot controller project and a game development project. The process, the deliverables and the evaluation of the project were the same for both types of projects - only the domain was different. In the Game project, the students were asked to develop a game using the Microsoft XNA framework and C#. Finally, an evaluation about software architecture course are conducted [23, 24]. The evaluation is based on data from a project survey, the project deliverables from the students and other accessible course information. The main conclusion from study was that game development projects can successfully be used to teach software architecture if we consider Robot as an evaluation benchmark.

Integrating our experiences on running of game project in software architecture course in 2008, we conducted a new option to add one more COTS - Android in software architecture course project during 2010-2011. The students could now in addition to the Java Robot project and the XNA Game project, choose to develop a social application or a game in Android. Independent of the COTS and the domain chosen, the students had to focus on the same software architecture issues during the project and follow the same templates. The introduction of game and social Android projects allowed us to compare how the domain the students work on in the project affect the learning and the project experiences independent of the COTS. A detailed description was in following chapters.

3 Method

This section describes the research method to get the relevant data for our experiment of using Android development in software architecture projects.

3.1 Aim

This article focuses on using the same COTS but with different development domains to investigate whether the different domains produce different output. In our previous research, the effectiveness of GDBL conclusion was based on the different COTS - Robot and XNA. This paper excludes game developed in XNA and robot controller developed in Java, and only focuses on the Android platform and development of social application vs. game application. Our evaluation covers five topics: distribution of chosen domain, students' perception of the project, project deliveries and code quality and complexity, students' Effort, and awarded project grades.

3.2 GQM approach

The comparison of the social and game project should help to discover the differences and reveal the effects of introducing a project on the Android platform. This evaluation is a quasi-experiment, not a controlled experiment. The research method used is based on the Goal, Question Metrics (GQM) approach [25] where we first define a research goal (conceptual level), then define a set of research questions (operational level), and finally describe a set of metrics to answer the defined research questions (quantitative level). In our case, the metrics used to give answers to the research questions are a mixture of quantitative and qualitative data. Table 1 shows the GQM approach used to analyze game development project in software architecture course.

Table 1. GQM Table

Goal	Analyze	Software development project
	For the purpose of	Comparing social application vs. game application domain on same COTS
	With respect to	Difference and effectiveness of two domain of the projects

	From the point of view of	Researcher & Educator		
	In context of	Students in software architecture course		
Questions	Q1: Are there any differences in how the students perceive the project for students choosing an Android game project vs. students choosing an Android social project?	Q2: Are there any differences in the software architectures designed by students doing an Android game project vs. students doing an Android social project?	Q3: Are there any differences in the implementation effort in the project by students doing an Android game project vs. students doing an Android social project?	Q4: Are there any differences in the performance of students doing an Android game project vs. students doing an Android social project?
Metric	M1: Number of students choosing game project vs. social project.	M3: Project reports	M4: Source code files	M6: Project score
	M2: Questionnaire survey with 5-Level Likert Scale: Strong disagree (1)- Disagree (2)- Neutral (3)-Agree (4)-Strong Agree (5)		M5: Time spent	

3.3 Procedures

When students start the project and follows the projects phases, they should report the time they spend on each phase of the project. The first two phases allow the students individually or in pairs to get familiar with the COTS and architectural and design patterns. The main work of the project is carried out in the phases 3-5 and includes requirement specification, architectural design, architectural evaluation, implementation and testing. The students produce a delivery for each phase, which is evaluated by the course staff, and feedback is given to improve before the final delivery. At the end of phase 5, the students will produce a final delivery, which is evaluated and graded by the course staff. After completing phase 5, the students have to answer a questionnaire that focuses on how the students perceive the project. In phases 6, the students must carry out a post-mortem analysis of their project as a whole to reflect on their successes and their challenges.

4 Results

In 2010 and 2011, the students could choose to do the project using three COTS: Robot (Java), XNA (C#) and Android (Java). The students' selection of COTS is shown in Figure 1, where 36 students chose Khepera robot (19%), 55 students chose XNA (27%) and 102 students (54%) chose Android. Of the students that chose Android, 58 students (57%) chose social application vs. 44 students (43%) game. If we look at the domains the students chose we see that 51% chose game development, 30% chose social applications and 19% chose robot controller.

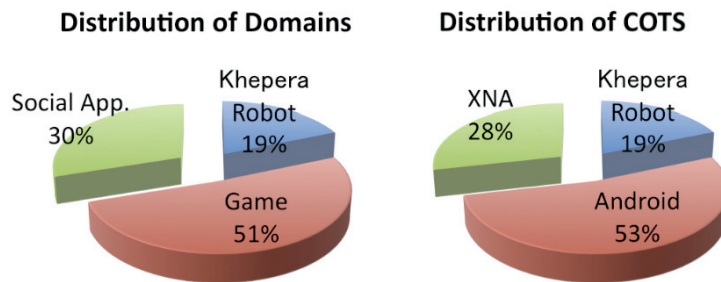


Figure 1. Distribution of selection of type of software architecture projects

The statistics of figure 1 clearly reveals that the majority of students prefer game development compared to other domains. And Android is the most popular COTS by far, and we believe this is due to its openness for developers, development in Java, attractive devices, innovative features and development, and a new way of sharing developed applications through Android market.

In the first phase of the project, the students were asked to fill in a questionnaire on the reasons to choose the COTS and domain. The top reasons list were: 1) Programming reason (familiar with Java or C#) (70.7%), 2) To learn about the COTS (Robot, XNA, Android) (59.5%), 3) Games motivation or amusement

reasons (40.1%), 4) Social application motivation (39.5%), 5) To learn about the domain (Robot, Game, Social) (34.2%), 6) Hardware motivation, running games on Android phone, Zuneplayer (33%), and 7) Make games for Android Market or XNA club (24.5%). From above data, we found that the game domain has advantages in drawing students' attention and its attractive peripherals, like hardware or software markets, and so does android social domain. This was not the case for the Robot domain.

The following subsections focus on the analysis of whether the domain game vs. social cause any significant different output in the following four aspects: 1) Students perception of the project, 2) The design complexity of software architectures, 3) Students' implementation effort in the project, and 4) Students' score in projects.

4.1 Differences in how students perceived the project

A project survey was conducted one week after the students completed their software architecture project. The goal of this survey was to reveal possible differences in the students' perception of the project between teams working with social projects vs. teams working with game projects on the same COTS - the Android platform. Statements in the survey made the students reflect on how the project helped them to learn software architecture.

The hypothesis defined for this survey was the following:

H_0 : There is no difference in how students doing game project and social project on the same COTS - Android perceive the software architecture project.

To test hypothesis we used Kruskal-Wallis Test [26] since it is a non-parametric method for testing equality of population medians among groups [24]. This test is usually for: 1) users cannot assume a normal population and 2) the sample sizes of the two groups are different. Table 2 shows the results of Kruskal-Wallis Test on the statements PS1-PS6. 38 of 44 game project students replied while 35 out of 58 social project students replied the questionnaire. Each item in the questionnaire is responded to by assigning a scale value from 1 to 5, where 1 indicates strong disagreement and 5 indicates strong agreement.

Table 2. Wilcoxon Test of the statements PS1-PS11

Statement	COTS	Average	Median	Standard deviation	P
PS1: I found it difficult to evaluate the other group's architecture in the ATAM?	Game	3.45	4	1.06	0.178
	Social	3.77	4	0.91	
PS2: I found it difficult to focus on our assigned quality attributes	Game	3.05	3	1.09	0.024
	Social	3.57	4	0.85	
PS3: I found it easy to integrate known architectural or design patterns	Game	3.21	3	0.93	0.332
	Social	2.94	3	1.03	
PS4: I spent more time on technical matters than on architectural matters?	Game	3.71	4	1.20	0.175
	Social	4.06	4	1.03	
PS5: I have learned a lot about software architecture during the project.	Game	3.50	4	0.86	0.552
	Social	3.31	4	0.99	
PS6: I would have chosen other project if I could go back in time	Game	1.13	1	0.34	0.289
	Social	1.20	1	0.41	

From the test results, the lowest significant difference ($P \leq 0.05$) in questionnaire's response is PS2 ($P=0.024$). We conclude that the Android game and Android social has significant difference on the students perceived the difficulty to focus on the assigned quality attributes in the project. The median of Likert scale score is 3 for android game, but 4 for android social. It indicates that android game project students were neutral on this PS5, but social project students have a tendency on the agreement of PS5. One possible explanation is that quality attribute, like terms - modifiability or testability linked to a game concept is easier to imagine and catch the students' attention to look into it. But social applications may have more fixed impression in students' life and cause less deep effect than games to motivate students to think. Others statements have no significant difference from students perception.

Further, even there is no significant difference for the two other low P-value, the average value of PS1 and PS4 still indicates that students from game project found less difficult to evaluate the other group's architecture in the ATAM and spent less time on technical matters than the students from social projects. In

addition, PS6: the students had to answer whether they would have chosen another project if they could go back in time. Figure 2 shows a more detailed statistics for it.

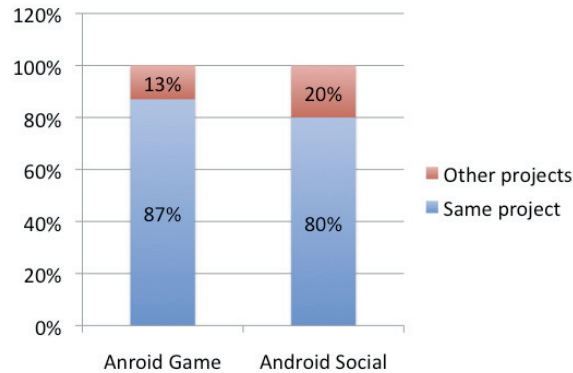


Figure 2. Responses to PS6: Would you have chosen the same project if you could go back in time.

Figure 2 shows that there is a higher percentage of the social project students that would have chosen another project (20%) compared to the game project students (13%).

As an overall, the survey reveals one significant difference that students from game projects have a better focus on quality attributes. Statements got low p-values (P1, P2, P4) that revealed the tendency that game teams were more positive feedback than the social teams on how they perceived the project.

4.2 Differences in the design of software architecture

It is difficult to evaluate software architectures empirically, but we have chosen to do so by comparing the number of design patterns the students used, the number of main modules/classes identified in the logical view of the software architecture, and the number of hierarchical levels in the architecture. We admit that there are many sources of errors in this comparison, as the two domains are so different. However, the emphasis in this course is on using software design patterns and presenting the different views of the software architecture in sufficient detail with emphasis on the logical view. The empirical data should highlight the differences between the two types of projects if any. The empirical data has been collected by reading through and analyzing the final project reports from 12 game project teams and 16 social project teams.

1) Use of design patterns

Table 2 presents the descriptive statistics of the number of architectural and design patterns used in the Social and the Game projects. The results in Table 2 indicate that there are some differences in how patterns are used in the two types of projects.

Table 2. Number design patterns used

		Average	Standard deviation	Max	Min
Design Patterns	Game	2.67	1.92	7	1
	Social	1.56	0.73	3	1

Table 3 presents Kruskal-Wallis Test results and shows that there are no statistically significant differences in the number of design patterns produced by the two different project types.

Table 3: Hypothesis tests on number of design patterns used

Hypothesis	COTS	N	Median	P
No difference in number of used design patterns	Game	12	2	0.111
	Social	16	1	

Table 3 indicates no statistically significant difference for the number of design pattern used for the two types of projects. From reading through the projects reports, Figure 3 presents the distribution of design patterns used by social teams and by game teams. The charts show that the Observer was the most popular for both types of project. Further, that the Abstract Factory, State pattern was among the top three for Game teams, singleton and template pattern was among the top three for social teams. The Game projects had more diversity in applying architecture and design patterns than social project. For instance, game projects used eight design patterns compared to six design patterns in social projects as shown in Figure 3.

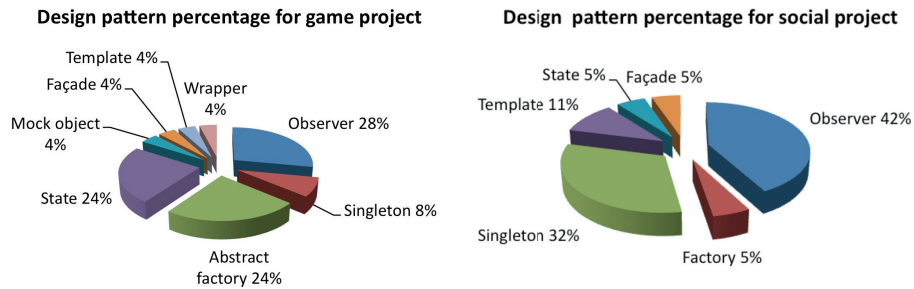


Figure 3. Distribution of usage of design patterns for game and social projects

Even there is no significant difference, but the low P-value is close to 0.1. The median in table 3 implies that game teams used more design patterns in their projects, it may cause that game projects used more types of patterns than social projects in an overall statistics showed in figure 3.

2) Software Architecture Complexity

Two metrics were chosen to indicate the complexity of the software architecture [24]: (1) The number of main modules or main classes described in the logical view of the software architecture, and (2) The number of hierarchical levels in the model presented in the logical view of the software architecture. The reason the logical view was chosen for computing complexity is that the logical view is the main one that gives the best overview of the designed architecture. Table 4 lists the measurements of the number of main modules/classes and the number of hierarchical levels in the logical view of the software architecture for social and game projects.

Table 4. Measurement of software architecture complexity

	Numbers of Main Modules/classes		Number of Levels in architecture	
	Game	Social	Game	Social
Average	14	9.7	3	1,75
Standard deviation	4.9	6.6	0.6	0,77
Max	21	28	4	3
Min	7	3	2	1

Table 4 shows that the game project teams on average have almost four more main modules/classes (28%) than the social teams and the standard deviation is lower. Further, the number of levels in the architecture in game projects can be decomposed into almost twice as many levels compared to social projects.

Table 5. Hypothesis tests on architectural complexity

Hypothesis	COTS	N	Median	P
No difference in number of main modules/classes	Game	12	14	0.021
	Social	16	7	
No difference in number of levels in architecture	Game	12	3	0.000
	Social	16	2	

Table 5 gives the results from Kruskal-Wallis Test on a number of main modules/classes and numbers of levels in the architecture. Both of the tests give low P-values ($P < 0.05$). Specifically, the tests show that there is statistically significant difference on the number of main classes and levels in architecture. From this result, it implies game project has more complexity in architecture levels than social projects, it may be due to they used more patterns to implement in their game projects that cause this difference.

4.3 Differences in the Effort Put into the Project

To evaluate the effort of each project that students put into, two indicators are used as the measurement criteria: 1) Time spent on the project, and 2) Structure and size of project files and number of lines of code.

1) Time spent

We have asked students to estimate on how many hours the project teams worked in the software architecture project during the phases 3-5 (core phases of the project). Table 6, shows the estimated number of hours given by each team.

Table 6. Time spent on the project for each team

Time per team (Hours)	Game	Social
Average	334	338
Standard Deviation	133.7	114.7
Max	520	535
Min	110	183

Based on each team's time effort, we ran the Kruskal-Wallis Test on the difference on hours spending in the project for each team.

Table 7. Hypothesis on hours spending

Hypothesis	COTS	N	Median	P
No difference on time spending for each team	Game	12	362	0.889
	Social	16	334	

From above results, there is no statistically significant difference on time spent on the project for game teams and social teams. On contrary, the time spending is quite similar.

2) Project analysis

Further, we chose to look at metrics from the implementation to give an estimate on how much was produced during the project. It can give a good indication of the complexity of the software architecture and the resulting implementation of the application [24]. Since both types of teams used Android and the domains are comparable in terms of complexity, we expected to find difference in productivity. During the development process, they were free to use online resource or other open source libraries for Android to save coding time for the software architecture design.

The following metrics were chosen to compute the effort of the student teams: 1) Number of source Files (NoF); 2) Number of Comments in code (NoC); 3) Lines of source Code not counting empty lines or comments (LoC).

Table 8 presents a comparison of the implementation metrics for the game projects and social projects, only java code files to be counted in the table, and the external library code files and resource files are excluded.

Table 8. Implementation metrics from the architecture projects

	NoF		NoC		LoC	
	Game	Social	Game	Social	Game	Social
Average	37	24	1016	536	2585	1949
Standard deviation	13	13	807	755	1172	1368
Max	54	45	2571	2886	4173	5082
Min	15	5	206	37	844	390

Table 9 shows the results from Kruskal-Wallis Test on the difference in the number of files and the number of lines of code produced by the two different types of project.

Table 9. Hypothesis tests on project implementation codes

Hypothesis		N	Median	P
No difference in number of lines of code	Game	12	2672	0.114
	Social	16	1523	

The results from the Kruskal-Wallis Test indicate that there is no statistically significant difference in LoC between the two types of project. But the low P-value is close to 0.1. The average value from Table 8 indicates game teams put more effort on the implementation, like coding, making comments, structure codes into more files during the project.

From the Table 6-9, we can found: the game project teams have produced on average almost one third as much code (133% more) in similar time spending (334 vs. 338). It implies that game project teams are more productive to put effort in coding, comments to construct a complex game software architecture in similar time spending than social project teams.

4.4 Difference in the project grades

The project score is between 0-30 points and takes 30% of the final grade. The project grades interval are classified as: A: Score $\geq 90\%$; B: Score $\geq 80\%$ and score $< 90\%$; C: Score $\geq 60\%$ and score $< 80\%$; D: Score $\geq 50\%$ and score $< 60\%$; E: Score $\geq 40\%$ and score $< 50\%$; F: Score $< 40\%$ (fail).

In order to investigate if there were any differences in how the group scored (0-30 points) on the project for students that has chosen game and social projects on Android. The Kruskal-Wallis Test was used to test this hypothesis, as we cannot assume a normal population and the sample size of the two groups is different. Table 10 presents the results of the Kruskal-Wallis Test on the difference in project grades for each game and social student.

Table 10. Kruskal-Wallis Test on different in project score

Hypothesis	COTS	N	Median	P
No difference in project score groups get from doing Game vs. Social project	Game	44	26	0.997
	Social	58	26	

There is no significant difference in the project score using same COTS for development. We run the social project in 2010 and game project in 2011 separately. The project implementation requirements and templates are keeping the same from phase 3 to 6 in two years and evaluation process and persons are the same, we can identify that students accomplished both projects under the same conditions. It reflects the difficulty could be similar. So, we only make a conclusion on the project score has no significant difference, In order to get an overview of the scores, Figure 4 gives the distribution of grades on the project for the two types of projects (game vs. social).

Distribution of Project Score

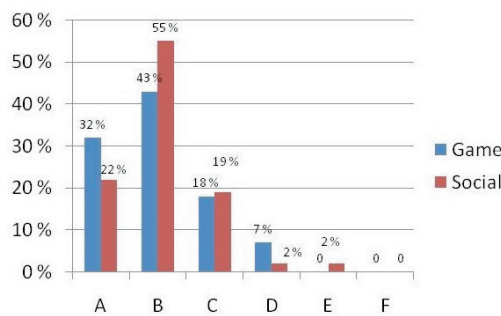


Figure 4. Grades distribution on project

5 Validity Threats

We now turn to what are considered to be the most important threats to the validity of this evaluation.

5.1 Internal Validity.

The internal validity of an experiment concerns “the validity of inferences about whether observed covariation between A (the presumed treatment) and B (the presumed outcome) reflects a causal relationship from A to B as those variables were manipulated or measured” [27]. If changes in B have causes other than the manipulation of A, there is a threat to internal validity.

There are two main internal validity threats to this evaluation. The first internal threat is that the sample of two groups used in the evaluation is not randomized. The students were allowed to choose either a Android game or a Android social project. We do not believe that one specific type of student chose one project over the other, thus harming the evaluation results. The second internal threat is if there were any differences how the students had to perform the project independently of the domain chosen. Independently of doing a social or a game project, the students had to go through exactly the same phases in the project and deliver exactly the same documents based on the same document templates in both 2010 and 2011. We have identified one difference in how the two types of projects were carried out. The 1- 2 phases of the project phase was different for the game and social projects students. These two phases are not a part of inclusive data and material used to evaluate the project. We do not believe that these differences have had any major impact in the way the students did or performed in their projects since it is the preparation phases, we noticed and excluded of them.

5.2 Construct Validity.

Construct validity concerns the degree to which inferences are warranted, from (1) the observed persons, settings, and cause and effect operations included in a study to (2) the constructs that these instances might represent. The question, therefore, is whether the sampling particulars of a study can be defended as measures of general constructs [27].

In the evaluation of using Android project in a software architecture course our research goal was to investigate the difference and similarity of game project and social project on Android platform. The GQM approach was chosen to detail this goal into four research questions with supporting metrics. In order to give answers to these four research questions the data sources and metrics available from our software architecture course were chosen. It cannot be claimed that the selected data sources and metrics in our evaluation give evidence for all the conclusions, but they are all strong indicators contributing to a picture that describes the differences between the two project types. Through the evaluation we have used various methods for comparing the results. The choice of methods is based on the best way of describing and visualizing the differences between the two groups using the available data.

5.3 External Validity.

The issue of external validity concerns whether a causal relationship holds (1) for variations in persons, settings, treatments, and outcomes that were in the experiment and (2) for persons, settings, treatments, and outcomes that were not in the experiment [27].

The results reported in this article are most relevant for other teachers thinking of introducing game projects as a part of their software architecture course. Further, the results are also relevant for teachers that want to introduce game projects in SE and CS courses, as many of these courses have similar characteristics. A limitation of this study is that the subjects in the evaluation are CS or SE students who have completed their first three years. It is not evident that the results are valid for students without any or less than three years background in CS or SE.

6 Conclusions

Based on our previous experiment of using XNA and current experiment of using Android in software architecture, we found game motivation and surround interesting peripherals are one of most attractive factor. Besides of the introduction of a new COTS – Android in a software architecture course, the goal of this article is to identify the difference output of same COTS and get evaluation result to answer the four research questions.

The first research question asked if there are any differences in how students choosing Android game vs. Android social projects perceived the software architecture project (RQ1). The statistically significant finding is that social project students found it more difficult to focus on the assigned quality attributes than game project ($P = 0.024$). Other data from lower P-value also reflect that game teams have more positive attitudes towards project requirements than the social team. In addition, the results show that 20% of the

students doing an Android social project would have chosen the other projects if they had to do the project again, which is more than the android game project students.

The second research question asked if there are any differences in how students choosing Android game vs. social projects designed their software architectures (RQ2). Even the analysis of the project reports concludes that no significant difference on design pattern used, but the low P-value close to 0.1 reveals that game teams applied more diverse patterns in their projects than social team. Further, the statistically significant difference shows that the software architectures produced in game projects were on average more complex than the architectures produced in social projects ($p < 0.05$).

The third research question asked if there were any differences in the effort the students put into the project when they worked with an Android game or an Android social project (RQ3). The results show that in similar time spending, teams working with game projects produced on average almost 133% as much code as teams working with Android social projects and game project students had customs to make twice detailed comments on the codes and organized codes into more files than social projects students.

The fourth and final research question asked if there are any differences in the performance for students doing a Game project vs. students doing a social project (RQ4). The comparison of the two types of projects showed that there was no statistically significant difference in the project.

According above conclusion and compared with previous research on XNA and Robot project used in software architecture course [24], we found that there exist quite similar conclusions for both game domain (XNA and Android game) in respect to: 1) Stable popularity of game domain; 2) Better perception of project from students aspect. 3) More design patterns used and high complexity of software architecture. 4) Same output in project score as social project.

Refer to Android COTS specifically, the main differences to Android game projects could be used an interesting and effectiveness tool in software architecture teaching in aspect to motivate students on design of complex architecture with applied more patterns and more productive coding work than Android social projects. Further, compared to XNA and Robot simulator, Android is an attractive platform to the students from the students' survey, that encourage us to conduct more practices on improvement of using Android as a development tool in software engineering practices, and inspire us the possibility to bring more choices, like iPhone SDK into COTS domains.

Reference

- [1] S. M. Dorman, "Video and Computer Games: Effect on Children and Implications for Health Education," *Journal of School Health*, vol. 67, pp. 133-138, 1997.
- [2] M. Prensky, "Digital game-based learning," *Computers in entertainment*, vol. 1, pp. 21- 24, 2003.
- [3] J. Blow, "Game Development: Harder Than You Think," *Queue*, vol. 1, pp. 28-37, 2004.
- [4] K. Sung, et al., "Game-Themed Programming Assignment Modules: A Pathway for Gradual Integration of Gaming Context Into Existing Introductory Programming Courses," *IEEE Transactions on Education*, 2010.
- [5] A. I. Wang and B. Wu, "An Application of a Game Development Framework in Higher Education," *International Journal of Computer Games Technology*, vol. 2009, 2009.
- [6] B. A. Foss and T. I. Eikaas, "Game Play in Engineering Education Concept and Experimental Results," *International Journal of Engineering Education*, vol. 22, pp. 1043-1052, 2006.
- [7] G. Sindre, "Experimental validation of the learning effect for a pedagogical game on computer fundamentals," *IEEE transactions on education*, vol. 52, p. 10, 2009.
- [8] A. I. Wang, "An Evaluation of a Mobile Game Concept for Lectures," presented at the IEEE 21st Conference on Software Engineering Education and Training, 2008.
- [9] A. I. Wang, et al., "LECTURE QUIZ - A Mobile Game Concept for Lectures," presented at the In 11th IASTED International Conference on Software Engineering and Application (SEA 2007), 2007.
- [10] M. S. El-Nasr, "Learning through game modding," *Computers in entertainment*, vol. 4, 2006.
- [11] G. Lukas, "Uses of the LOGO programming language in undergraduate instruction," presented at the Proceedings of the ACM annual conference - Volume 2, Boston, Massachusetts, United States, 1972.

- [12] M. Micco, "An undergraduate curriculum in expert systems design or knowledge engineering," presented at the Proceedings of the 15th annual conference on Computer Science, St. Louis, Missouri, United States, 1987.
- [13] M. Habgood, et al., "The educational and motivational content of digital games made by children," in CAL' 05: Virtual Learning, Bristol, UK., 2005.
- [14] Yulia and R. Adipranata, "Teaching object oriented programming course using cooperative learning method based on game design and visual object oriented environment," in 2nd International Conference on Education Technology and Computer (ICETC), 2010, pp. V2-355-V2-359.
- [15] L. Werner, et al., "Can middle-schoolers use Storytelling Alice to make games?: results of a pilot study," presented at the Proceedings of the 4th International Conference on Foundations of Digital Games, Orlando, Florida, 2009.
- [16] J. Robertson and C. Howells, "Computer game design: Opportunities for successful learning," Computers & Education, vol. 50, pp. 559-578, 2008.
- [17] S. Kurkovsky, "Can mobile game development foster student interest in computer science?," in International IEEE Consumer Electronics Society's Games Innovations Conference, (ICE-GIC 2009), 2009, pp. 92-100.
- [18] B. Ahmed. and M. Steve., "Using ATAM to Evaluate a Game-based Architecture," in Workshop on architecture-Centric Evolution(ACE 2006), hosted at the 20th European Conference on Object-Oriented Programming ECOOP., Nantes, France, 2006.
- [19] Len Bass, et al., Software architecture in practice: Second Edition: Addison-Wesley Professional, 2003.
- [20] R. Kazman, et al., "The architecture tradeoff analysis method," in Engineering of Complex Computer Systems, 1998. ICECCS '98. Proceedings. Fourth IEEE International Conference on, 1998, pp. 68-78.
- [21] A. I. Wang and T. Stalhane, "Using Post Mortem Analysis to Evaluate Software Architecture Student Projects," in Software Engineering Education & Training, 18th Conference on, 2005, pp. 43-50.
- [22] WSU. (2009, Download WSU_KSuite_1.1.2.
- [23] B. Wu, et al., "An Evaluation of Using a Game Development Framework in Higher Education," Proceedings / Conference on Software Engineering Education and Training, 2009.
- [24] A. I. Wang, "Extensive Evaluation of Using a Game Project in a Software Architecture Course," Transactions on Computing Education (ACM), vol. Volume 11., February 2011. 2011.
- [25] V. Basili, "Software modeling and measurement: the Goal/Question/Metric paradigm," 1992.
- [26] W. H. Kruskal and W. A. Wallis, "Use of Ranks in One-Criterion Variance Analysis," Journal of the American Statistical Association, vol. 47, pp. 583-621, 1952.
- [27] W. R. Shadish, et al., Experimental and quasi-experimental designs for generalized causal inference: Boston, MA, US: Houghton, Mifflin and Company, 2002.

Declarations on Co-author Consensus

Co-Authors	Papers
<i>Alf Inge Wang</i>	Paper 1-Paper 12
<i>Yuanyuan Zhang</i>	Paper 1
<i>Sveinung Kval Bakken</i>	Paper 2
<i>Erling Andreas Børresen</i>	Paper 3
<i>Knut Andre Tidemann</i>	Paper 3
<i>Jan-Erik Strøm</i>	Paper 6 and Paper 7
<i>Trond Blomholm Kvamme</i>	Paper 6 and Paper 7
<i>Anders Hartvoll Ruud</i>	Paper 8
<i>Wan Zhen Zhang</i>	Paper 8

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)

As co-author on the following joint publications in Bian Wu's PhD-thesis:

1. Bian Wu, Alf Inge Wang and Yuanyuan Zhang, "Experiences from Implementing an Educational MMORPG", 2nd International IEEE Consumer Electronics Society's Games Innovation Conference (GIC 2010), Hong Kong, 21-23 December 2010. ISBN: 978-1-4244-7178-2, DOI: 10.1109/ICEGIC.2010.5716896

Details: This study is a result from a case study of World of Wisdom education game. It was conducted by 1st author under the supervision of 2nd author. The 1st author contributed in the research design, implementation, discussion and conclusion. The 2nd author helped with guiding, correcting and reviewing the article. The 3rd author proposed some suggestions on research design and theoretical context.

2. Alf Inge Wang, Bian Wu, Sveinung Kval Bakken, "Experiences from Implementing a Face-to-Face Educational Game for iPhone/iPod Touch", 2nd International IEEE Consumer Electronics Society's Games Innovation Conference (GIC 2010), 21-23 December 2010, Hong Kong. ISBN: 978-1-4244-7178-2. DOI: 10.1109/ICEGIC.2010.5716895

Details: This study was done by all three authors. Contribution to introduction, game design, data collection, analysis and writing was also done all. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author contributed in the introduction, related works, evaluation and conclusion. The 3rd author was a student supervised by 1st author and contributed in the game design, data analysis, and evaluation.

3. Bian Wu; Alf Inge Wang; Erling Andreas Børresen; Knut Andre Tidemann: "Improvement of a Lecture Game Concept - Implementing Lecture Quiz 2.0", 3rd International Conference on Computer Supported Education, 6-9 May 2011, Noordwijkerhout, The Netherlands. ISBN:978-989-8425-50-8

Details: This article is a result from the case study of a lecture game project, called Lecture Quiz (LQ). The LQ was conducted by 1st author under the supervision of 2nd author. The 1st author summarized the two versions of LQ 1.0 and LQ 2.0, and contributed in the introduction, related works, game

design, evaluation and conclusion. And 2nd author helped in the correction and reviewing and overall supervising the case study as a whole. And 3rd and 4th authors were two students supervised by 2nd author, and they contributed in LQ 2.0 in the section 3 and section 4 partly.

4. Bian Wu, Alf Inge Wang, " A Pervasive Game to Know Your City Better", Accepted for 2011 International IEEE Consumer Electronics Society's Games Innovation Conference (IGIC 2011), November 2010, Orange, California, USA.

Details: This study was conducted by the 1st author with the support from the 2nd author. 1st author contributed in the introduction, game content and results. The 2nd author helped with guiding, correcting and reviewing the article.

5. Alf Inge Wang and Bian Wu, "An Application of a Game Development Framework in Higher Education", International Journal of Computer Games Technology, Special Issue on Game Technology for Training and Education, Volume 2009. ISSN: 1687-7047 EISSN: 1687-7055. DOI=10.1155/2009/693267

Details: This study was planned and designed by all two authors. Contribution to data collection, analysis and writing was also done all. The 1st author contributed in the introduction, course design, data extraction and analysis, evaluation and conclusion. The 2nd author contributed in the introduction, related works, a proposed framework for the applying game development in education, course summaries and conclusion.

6. Bian Wu and Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "An Evaluation of Using a Game Development Framework in Higher Education", 22nd IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2009), February 17-19, Hyderabad, India, 2009. ISBN: 978-0-7695-3539-5 DOI=10.1109/CSEET.2009.9

Details: This study was planned and designed by all four authors. Contribution to data collection, analysis and writing was also done all. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author supervised and helped the 1st author during the process. The 3rd and 4th authors were two students supervised by the 2nd author; they contributed in the survey and evaluation.

7. Bian Wu, Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "XQUEST used in Software Architecture Education", IEEE Consumer Electronics Society's Games Innovation Conference, August 25-28, 2009, London, UK. ISBN: 978-1-4244-4459-5, DOI: 10.1109/ICEGIC.2009.5293607

Details: This study was planned and designed by all four authors. Contribution to data collection, analysis and writing was also done all. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author supervised and helped the

1st author during the process. The 3rd and 4th authors were two students supervised by the 2nd author; they contributed in design, data analysis and evaluation.

8. Bian Wu; Alf Inge Wang; Anders Hartvoll Ruud; Wan Zhen Zhang: "Extending Google Android's Application as an Educational Tool," the 3rd IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), April 12-16 2010, Kaohsiung, Taiwan. ISBN: 978-1-4244-6433-3. DOI: 10.1109/DIGITEL.2010.38

Details: This study was an extended case study of using Android in software architecture course. It was mainly conducted by 1st author. The 2nd author helped with guiding, correcting and reviewing the article. And 3rd author was a student supervised by the 1st author with the data collection and the implementation of the Sheep framework. The 4th author gave suggestion on the theoretical context in section 2.

9. Alf Inge Wang and Bian Wu, "Using Game Development to Teach Software Architecture", International Journal of Computer Games Technology, vol. 2011, Article ID 920873, 12 pages, 2011. ISSN: 1687-7047 EISSN: 1687-7055. DOI: 10.1155/2011/920873

Details: This article was conducted by two authors. 1st author is a teacher of the course and mainly design the course and exercise. 2nd author is the teaching assistant of the course and mainly work on the exercise improvements. 1st author contributed in the course design, data analysis and discussion. 2nd author contributed in the related work and exercise data extraction and discussion.

10. Bian Wu, Alf Inge Wang, " Game Development Framework for Software Engineering Education", Accepted for 2011 International IEEE Consumer Electronics Society's Games Innovation Conference (IGIC 2011), November 2011, Orange, California, USA.

Details: This article is the result of the literature review. This review was conducted by the 1st author with the support from the 2nd author. 1st author contributed in the introduction, literature review, data extraction and results. The 2nd author helped with guiding, reviewing the data and results.

11. Bian Wu, Alf Inge Wang, "A guideline for game development-based learning: A literature review", in second revision for IEEE Transactions on Learning Technologies (TLT).

Details: This article is the result of the systematic literature review. This review was conducted by the 1st author with the support from the 2nd author. 1st author contributed in the introduction, literature review, data extraction and results. The 2nd author helped with guiding, reviewing the data and results.

12. Bian Wu, Alf Inge Wang, "Comparison of Learning Software Architecture by Developing Social Applications vs. Games on the Android Platform", submitted to the International Journal of Computer Games Technology.

Details: This quasi-experiment was planned and designed by two authors. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author supervised and helped the 1st author during the process.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date: 26/4-2012
Location: Trondheim


Alf Inge Wang

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)

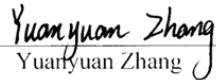
As co-author on the following joint publications in Bian Wu's PhD-thesis:

1. Bian Wu, Alf Inge Wang and Yuanyuan Zhang. "Experiences from Implementing an Educational MMORPG", 2nd International IEEE Consumer Electronics Society's Games Innovation Conference (GIC 2010), Hong Kong, 21-23 December 2010. ISBN: 978-1-4244-7178-2. DOI: 10.1109/ICEGIC.2010.5716896

Details: This study is a resulted from case study of World of Wisdom education game. It was conducted by 1st author under the supervision of 2nd author. The 1st author contributed in the research design, implementation, discussion and conclusion. The 2nd author helped with guiding, correcting and reviewing the article. The 3rd author proposed some suggestion on research design and theoretical context.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date: 2011-9-15
Location: Beijing, China


Yuanyuan Zhang

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)

As co-author on the following joint publications in Bian Wu's PhD-thesis:

1. Alf Inge Wang, Bian Wu, Sveinung Kval Bakken, "Experiences from Implementing a Face-to-Face Educational Game for iPhone/iPod Touch", 2nd International IEEE Consumer Electronics Society's Games Innovation Conference (GIC 2010), 21-23 December 2010, Hong Kong. ISBN: 978-1-4244-7178-2. DOI: 10.1109/ICEGIC.2010.5716895

Details: This study was done by all three authors. Contribution to introduction, game design, data collection, analysis and writing was also done all. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author contributed in the introduction, related works, evaluation and conclusion. The 3rd author was a student supervised by 1st author and contributed in the game design, data analysis, and evaluation.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date: 26-08-2011
Location: OSLO, NORWAY


Sveinung Kval Bakken

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)

As co-author on the following joint publications in Bian Wu's PhD-thesis:

1. Bian Wu; Alf Inge Wang; Erling Andreas Børresen; Knut Andre Tidemann: "Improvement of a Lecture Game Concept - Implementing Lecture Quiz 2.0", 3rd International Conference on Computer Supported Education, 6-9 May 2011, Noordwijkerhout, The Nederland. ISBN:978-989-8425-50-8

Details: This article is resulted from the case study of a lecture game project, called Lecture Quiz (LQ). The LQ was conducted by 1st author under the supervision of 2nd author. The 1st author summarized the two versions of LQ 1.0 and LQ 2.0, and contributed in the introduction, related works, game design, evaluation and conclusion. And 2nd author helped in the correction and reviewing and overall supervising the case study as a whole. And 3rd and 4th authors were two students supervised by 2nd author, and they contributed in LQ 2.0 in the section 3 and section 4 partly.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date: 6/9-2011
Location: Trondheim


Erling Andreas Børresen

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)

As co-author on the following joint publications in Bian Wu's PhD-thesis:

1. Bian Wu; Alf Inge Wang; Erling Andreas Børresen; Knut Andre Tidemann: "Improvement of a Lecture Game Concept - Implementing Lecture Quiz 2.0", 3rd International Conference on Computer Supported Education, 6-9 May 2011, Noordwijkerhout, The Nederland. ISBN:978-989-8425-50-8

Details: This article is resulted from the case study of a lecture game project, called Lecture Quiz (LQ). The LQ was conducted by 1st author under the supervision of 2nd author. The 1st author summarized the two versions of LQ 1.0 and LQ 2.0, and contributed in the introduction, related works, game design, evaluation and conclusion. And 2nd author helped in the correction and reviewing and overall supervising the case study as a whole. And 3rd and 4th authors were two students supervised by 2nd author, and they contributed in LQ 2.0 in the section 3 and section 4 partly.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date: 30/6/2011
Location: Sandefjord


Knut Andre Tidemann

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)

As co-author on the following joint publications in Bian Wu's PhD-thesis:

1. Bian Wu and Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "An Evaluation of Using a Game Development Framework in Higher Education", 22nd IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2009), February 17-19, Hyderabad, India, 2009. ISBN: 978-0-7695-3539-5 DOI=10.1109/CSEET.2009.9

Details: This study was planned and designed by all four authors. Contribution to data collection, analysis and writing was also done all. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author supervised and helped the 1st author during the process. The 3rd and 4th authors were two students supervised by the 2nd author; they contributed in the survey and evaluation.

2. Bian Wu, Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "XQUEST used in Software Architecture Education", IEEE Consumer Electronics Society's Games Innovation Conference, August 25-28, 2009, London, UK. ISBN: 978-1-4244-4459-5, DOI: 10.1109/ICEGIC.2009.5293607

Details: This study was planned and designed by all four authors. Contribution to data collection, analysis and writing was also done all. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author supervised and helped the 1st author during the process. The 3rd and 4th authors were two students supervised by the 2nd author; they contributed in design, data analysis and evaluation.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date: 29/08/2011
Location: Troadheim


Jan-Erik Strøm

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)

As co-author on the following joint publications in Bian Wu's PhD-thesis:

- Bian Wu and Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "An Evaluation of Using a Game Development Framework in Higher Education", 22nd IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2009), February 17-19, Hyderabad, India, 2009. ISBN: 978-0-7695-3539-5 DOI=10.1109/CSEET.2009.9


Details: This study was planned and designed by all four authors. Contribution to data collection, analysis and writing was also done all. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author supervised and helped the 1st author during the process. The 3rd and 4th authors were two students supervised by the 2nd author; they contributed in the survey and evaluation.

- Bian Wu, Alf Inge Wang, Jan-Erik Strøm and Trond Blomholm Kvamme: "XQUEST used in Software Architecture Education", IEEE Consumer Electronics Society's Games Innovation Conference, August 25-28, 2009, London, UK. ISBN: 978-1-4244-4459-5, DOI: 10.1109/ICEGIC.2009.5293607

Details: This study was planned and designed by all four authors. Contribution to data collection, analysis and writing was also done all. The 1st author contributed in the introduction, related works, data extraction and analysis, evaluation and conclusion. The 2nd author supervised and helped the 1st author during the process. The 3rd and 4th authors were two students supervised by the 2nd author; they contributed in design, data analysis and evaluation.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date:
Location:


Trond Blomholm Kvamme

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)

As co-author on the following joint publications in Bian Wu's PhD-thesis:

- Bian Wu; Alf Inge Wang; Anders Hartvoll Ruud; Wan Zhen Zhang: "Extending Google Android's Application as an Educational Tool," the 3rd IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), April 12-16 2010, Kaohsiung, Taiwan. ISBN: 978-1-4244-6433-3. DOI: 10.1109/DIGITEL.2010.38

Details: This study was an extended case study of using Android in software architecture course. It was mainly conducted by the 1st author. The 2nd author helped with guiding, correcting and reviewing the article. And 3rd author was a student supervised by the 1st author with the data collection and the implementation of the Sheep framework. The 4th author gave suggestion on the theoretical context in section 2.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date: August 26th, 2011
Location: Oslo, Norway


Anders Hartvoll Ruud

To Whom It May Concern,

Statement of authorship on joint publications to be used in Bian Wu's PhD-thesis

(Cf. NTNU PhD-regulation §7.4, section 4 and dr.philos regulation §3, section 5)


As co-author on the following joint publications in Bian Wu's PhD-thesis:

1. Bian Wu; Alf Inge Wang; Anders Hartvoll Ruud; Wan Zhen Zhang: "Extending Google Android's Application as an Educational Tool," the 3rd IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), April 12-16 2010, Kaohsiung, Taiwan. ISBN: 978-1-4244-6433-3. DOI: 10.1109/DIGITEL.2010.38

Details: This study was an extended case study of using Android in software architecture course. It was mainly conducted by the 1st author. The 2nd author helped with guiding, correcting and reviewing the article. And 3rd author was a student supervised by the 1st author with the data collection and the implementation of the Sheep framework. The 4th author gave suggestion on the theoretical context in section 2.

I declare that the candidate's contribution to these works are correctly identified and that I consent that the works are done to be used as part of the thesis.

Date: 2011.9.14
Location: Guilin, China


Wanzhen Zhang

