Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2012, Article ID 494232, 10 pages
doi:10.1155/2012/494232

Research Article

Comparison of Learning Software Architecture by Developing Social Applications versus Games on the Android Platform

Bian Wu and Alf Inge Wang

Department of Computer Science, Norwegian University of Science and Technology, 7491 Trondheim, Norway

Correspondence should be addressed to Bian Wu, bian@idi.ntnu.no

Received 15 April 2012; Accepted 16 July 2012

Academic Editor: Daniel Thalmann

This paper describes an empirical study where the focus was on discovering differences and similarities in students working on development of social applications versus 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. 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 focuses on one of projects—Android project, to see how much the application domain affects the course project independently 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.

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 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, and play writer, and so forth. 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, 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, for example [6, 7]. Second, games can be played as a part of a lecture to improve the participation and motivation of students, for example [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, for example [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, for example, 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 paper focuses on an evaluation where we wanted to discover similarities and differences between making students learn software architecture through game development versus 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 an 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 Tradeoff Analysis Method (ATAM) [18–20] 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 Postmortem 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.

2.3. Previous Results. Previously, 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 is 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. Independently 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 affects the learning and the project experiences independently of the COTS. A detailed description was in following sections.

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 paper 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 versus 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 quasiexperiment, 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.

3.3. Procedures. When students start the project and follow 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 postmortem 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 versus 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.

The statistics of Figure 1 clearly reveal 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 marked.

In the first phase of the project, the students were asked to fill in a questionnaire on the reasons to choose the COTS and

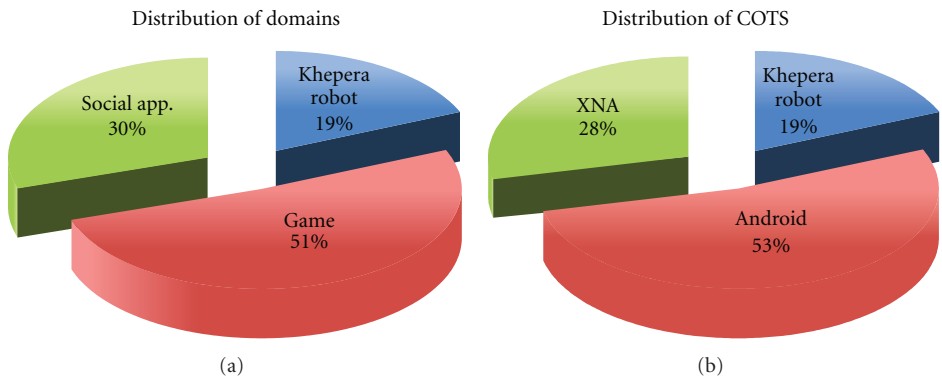

Figure 1: Distribution of selection of type of software architecture projects.

Table 1: GQM table.

Goal	Analyze	Software development project		
	For the purpose of	Comparing social application versus game application domain on same COTS		
	With respect to	Difference and effectiveness of two domains of the projects		
	From the point of view of	Researcher and 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 versus students choosing an Android social project?	Q2: Are there any differences in the software architectures designed by students doing an Android game project versus 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 versus students doing an Android social project?	Q4: Are there any differences in the performance of students doing an Android game project versus students doing an Android social project?
Metric	M1: Number of students choosing game project versus social project. M2: Questionnaire survey with 5-Level Likert Scale: Strong disagree (1), Disagree (2), Neutral (3), Agree (4), Strong Agree (5)	M3: Project reports	M4: Source code files M5: Time spent	M6: Project score

domain. The top reasons list was: (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 versus social causes 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 versus 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 nonparametric 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

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 another project if I could go back in time	Game	1.13	1	0.34	0.289
	Social	1.20	1	0.41	

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.

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 have significant difference on the students perceiving 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 termsmodifiability 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 statement have no significant difference from students perception.

Further, even there is no significant difference for the two other low P values, the average value of PS1 and PS4 still indicates that students from game project found it 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 more detailed statistics for it.

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 receive more positive feedback than the social teams on how they perceived the project.

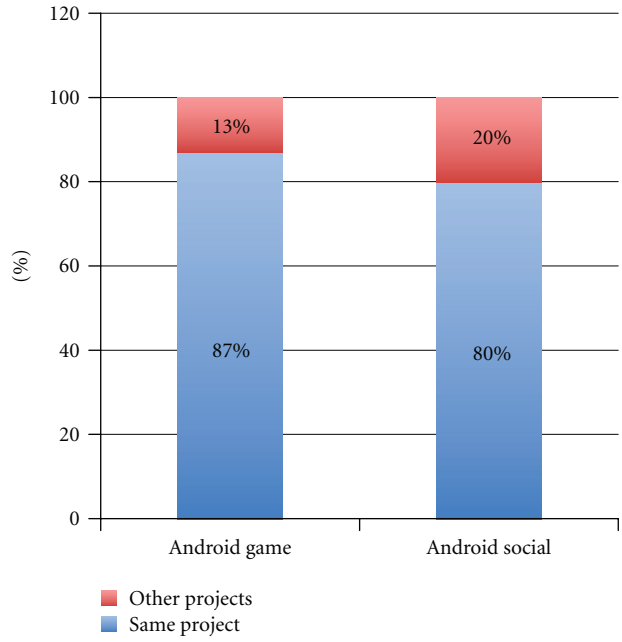

Figure 2: Reponses to PS6: would you have chosen the same project if you could go back in time.

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 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.

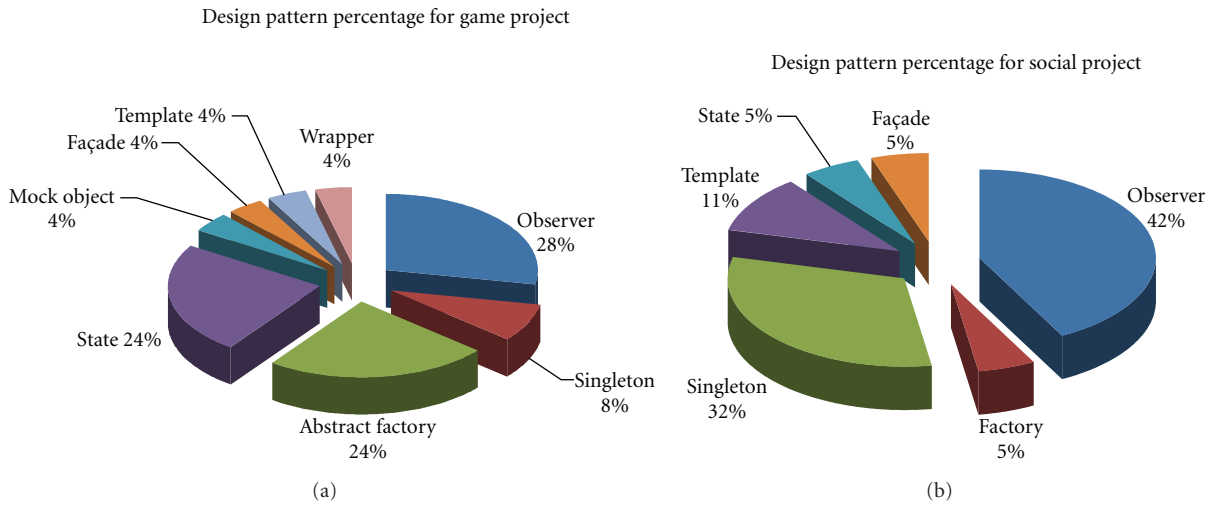

Figure 3: Distribution of usage of design patterns for game and social projects.

Table 3: Number of design patterns used.

		Average	Standard deviation	Max	Min
Design Patterns	Game	2.67	1.92	7	1
	Social	1.56	0.73	3	1

4.2.1. Use of Design Patterns. Table 3 presents the descriptive statistics of the number of architectural and design patterns used in the Social and the Game projects. The results in Table 3 indicate that there are some differences in how patterns are used in the two types of projects.

Table 4 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 4 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, the Abstract Factory and 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.

Even there is no significant difference, but the low P value is close to 0.1. The median in Table 4 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 shown in Figure 3.

4.2.2. 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 5 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 5 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 6 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 the fact that they used more patterns to implement 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 it, 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.

4.3.1. Time Spent. We have asked students to estimate on how many hours the project teams worked in the software

Table 4: 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 5: 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 6: 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 7: 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

architecture project during the phases 3–5 (core phases of the project). Table 7 shows the estimated number of hours given by each team.

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.

From previous results, there is no statistically significant difference on time spent on the project for game teams and social teams. On contrary, the time spending distribution in both projects is quite similar.

4.3.2. 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 9 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 10 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.

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 9 indicates game teams put more effort on the implementation, like coding, making comments, structure codes into more files during the project.

From the Tables 7, 8, 9, and 10, we can find the game project teams have produced on average almost one third as much code (133% more) in similar time spending (334 versus 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 and 30 points and takes 30% of the final grade. The project grades intervals are classified as: A: score ≥90%; B: score ≥80% and score <90%; C: score ≥60% and score <80%; D: score ≥50% and score <60%; E: score ≥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

Table 8: Hypothesis on hours spending.

Hypothesis	COTS	N	Median	P
No difference in time spending for each team	Game	12	362	0.889
	Social	16	334	

Table 9: 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 10: 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 two groups is different. Table 11 presents the results of the Kruskal-Wallis Test on the difference in project grades for each game and social student.

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 versus social).

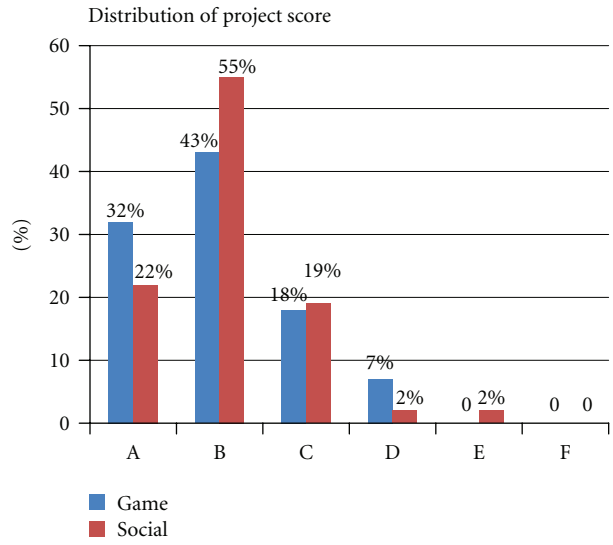

Figure 4: Grades distribution on project.

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 an Android game or an 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 were 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, that we noticed and excluded of them.

Table 11: Kruskal-Wallis Test on different in project score.

Hypothesis	COTS	N	Median	P
No difference in project score groups get from doing Game versus Social project	Game	44	26	0.997
	Social	58	26	

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 paper 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-year 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 surrounding interesting peripherals are one of the most attractive factors. Besides the introduction of a new COTS-Android in a software architecture course, the goal of this paper is to identify the difference output of same COTS and get evaluation result to answer the four research questions.

The first research question asked is if there are any differences in how students choosing Android game versus 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 is if there are any differences in how students choosing Android game versus social projects designed their software architectures (RQ2). Even the analysis of the project reports concludes that no significant difference on the used design patterns, 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 is 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 is if there are any differences in the performance of students doing a Game project versus 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 to the previous 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.

Referring to Android COTS specifically, the main differences from Android game projects could be used as an interesting and effective tool in software architecture teaching aspect to motivate students on design of complex architecture with applying 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 encourages us

to conduct more practices on improvement of using Android as a development tool in software engineering practices and inspires us the possibility to bring more choices, like iPhone SDK into COTS domains.

References

[1] S. M. Dorman, "Video and computer games: effect on children and implications for health education," *Journal of School Health*, vol. 67, no. 4, 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, C. Hillyard, R. L. Angotti, M. W. Panitz, D. S. Goldstein, and J. Nordlinger, "Game-Themed Programming Assignment Modules: a pathway for gradual integration of gaming context into existing introductory Programming Courses," *IEEE Transactions on Education*, vol. 54, no. 3, pp. 416–427, 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, no. 1, Article ID 693267, 12 pages, 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, no. 5, pp. 1043–1052, 2006.

[7] G. Sindre, L. Natvig, and M. Jahre, "Experimental validation of the learning effect for a pedagogical game on computer fundamentals," *IEEE Transactions on Education*, vol. 52, no. 1, pp. 10–18, 2009.

[8] A. I. Wang, "An evaluation of a mobile game concept for lectures," in *Proceedings of the IEEE 21st Conference on Software Engineering Education and Training*, 2008.

[9] A. I. Wang, T. Øfsdahl, and O. K. Mørch-Storstein, "LECTURE QUIZ—a mobile game concept for lectures," in *Proceedings of the 11th IASTED International Conference on Software Engineering and Application (SEA '07)*, 2007.

[10] M. S. El-Nasr and B. K. Smith, "Learning through game modding," *Computers in Entertainment*, vol. 4, no. 1, pp. 45–64, 2006.

[11] G. Lukas, "Uses of the LOGO programming language in undergraduate instruction," in *Proceedings of the ACM Annual Conference*, vol. 2, Boston, Mass, USA, 1972.

[12] M. Micco, "An undergraduate curriculum in expert systems design or knowledge engineering," in *Proceedings of the 15th Annual Conference on Computer Science*, St. Louis, Mo, USA, 1987.

[13] M. Habgood, S. Ainsworth, and S. Benford, "The educational and motivational content of digital games made by children," in *Proceedings of the Virtual Learning (CAL '05)*, 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 *Proceedings of the 2nd International Conference on Education Technology and Computer (ICETC '10)*, pp. V2355–V2359, June 2010.

[15] L. Werner, J. Denner, M. Bliesner, and P. Rex, "Can middle-schoolers use Storytelling Alice to make games? Results of a pilot study," in *Proceedings of the 4th International Conference on the Foundations of Digital Games (ICFDG '09)*, pp. 207–214, Orlando, Fla, USA, April 2009.

[16] J. Robertson and C. Howells, "Computer game design: opportunities for successful learning," *Computers and Education*, vol. 50, no. 2, pp. 559–578, 2008.

[17] S. Kurkovsky, "Can mobile game development foster student interest in computer science?" in *Proceedings of the 1st International IEEE Consumer Electronic Society's Games Innovation Conference (ICE-GiC '09)*, pp. 92–100, August 2009.

[18] B. Ahmed and M. Steve, "Using ATAM to evaluate a game-based architecture," in *Proceedings of the 20th European Conference on Object-Oriented Programming ECOOP, Workshop on Architecture-Centric Evolution (ACE '06)*, Nantes, France, 2006.

[19] L. Bass, P. Clements, R. Kazman et al., *Software Architecture in Practice*, Addison-Wesley Professional, 2nd edition, 2003.

[20] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere et al., "The architecture tradeoff analysis method," in *Proceedings of the 4th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '98)*, pp. 68–78, 1998.

[21] 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 and Training (CSEE & T '05)*, pp. 43–50, April 2005.

[22] WSU, Download WSU_KSuite_1.1.2., 2009.

[23] 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 22nd Conference on Software Engineering Education and Training (CSEET '09)*, pp. 41–44, February 2009.

[24] A. I. Wang, "Extensive evaluation of using a game project in a software architecture course," *ACM Transactions on Computing Education*, vol. 11, no. 1, article 5, 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, T. D. Cook, and D. T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*, Houghton, Mifflin and Company, Boston, Mass, USA, 2002.