Survey on Software Architecture, Creativity, and Game Technology

ALF INGE WANG, Norwegian University of Science and Technology
NJÅL NORDMARK, Norwegian University of Science and Technology

Software engineering and game development are different as in game development it is very difficult to elicit functional requirements from the users, and the customers buy and use the software only because it is engaging and fun. This book chapter presents results from a survey on how game developers think about and use software architecture in the development of games, on how creative development processes are managed and supported, and on use of game technology. The results presented in this chapter are responses from thirteen game developers on a survey focusing on software architecture, creative development processes and game technology. The research questions answered are: What role the software architecture plays in game development; How game developers manage changes to the software architecture; how creative development processes are managed and supported; and How game development and technology have evolved the last couple of years. The results of the survey show among other things that software architectures play a central role in game development where the main focus is on achieving software with good performance and high modifiability, creative processes are supported through flexible game engines and tools, use of scripting and dynamic loading of assets, and feature-based teams with both creative and technical professions are incrementally using more game-specific engines, tools and middleware in their development now compared to earlier.

KEYWORDS

Software architecture, Software process, Game development, Creative software development, Game technology

1 INTRODUCTION

Game development can be challenging as game engines and game hardware and development platforms changes rapidly, and code modules crafted for specific games offer less than 30 percent reuse [1]. In first decades of the video game history, game development was carried out by small teams. The software architectures as these games were typically made out of a few modules such as 2d graphics, simulation, sound, streaming of i/o and a main module. At this time, the main focus was on how to create an exciting game with the limited hardware resources available rather than a focus on software architecture and software engineering. The growth of the video game industry along with the progress in hardware have resulted in vastly larger and more complex games rendering and simulation huge interactive virtual worlds developed by large multi-disciplinary teams. The growth in size and complexity of games have resulted in similar growth in size and complexity of game architectures [2]. The game projects producing a typical AAA game title today are very large, and the game software itself has a complex software architecture with many interconnected modules. Many quality attributes in game development are the same as in traditional software development such as modifiability, reliability, security and usability, but one aspect that makes software architectures for games challenging is the absolute real-time requirements and the need to support the creative processes in game development [1]. Game development requires often to a larger extent than traditional software development a multitude of computer science skills [3] as well as other disciplines as art, game design, and audio/music [4]. The direct involvement of professions with very different background, knowledge and skills (e.g. the technical team vs. the creative team) poses challenges for how a game is developed. Based on all the challenges described above, game development is an interesting domain for software engineering research, as well as the fact that game development is a very big, successful and innovative industry.

So far, most of the software engineering research related to game development has focused on requirement engineering, and there is a lack of empirical work [5]. This chapter presents a study on how game developers think about and manage software architecture, how the creative processes affect the development, and how game development and game technology has changed in the recent years. The study investigates the relationship between creative design and software development, and how the technical and creative teams collaborate. The results presented is based on responses from a survey aimed at game developers. The initial survey was extended with a follow-up survey where in-depth questions were asked. Along with our own experience from game development and with support from research literature, this chapter draws a picture of how game developers work with and manage software architecture, the creative development process the use and how game development and technology has changed the recent years. To our knowledge, this is the first study of this kind within software engineering. This book chapter is based a master thesis at the Norwegian University of Science and Technology [6], and parts of this study has previously been published in [7].

The rest of the chapter is organized as follows. Section 2 presents related work on software architecture and games, and software engineering and games. Section 3 describes the research goal, research questions and research method used. Section 4 presents the results from the initial survey addressed to game developers. Section 5 describes the results from the more focused follow-up survey. Section 6 discusses the validity of the research and results presented, and Section 5 concludes the chapter.

2 RELATED WORK

As far as we know, there are no similar studies that focus both on software architecture and creative processes in game development. However, there are studies that focuses on the software architecture in games and studies focusing on the creative processes. In this section, we will present work in the field of software engineering and game development.

As games over the years have grown into large complex systems, the video game industry is facing several software engineering challenges. Kanode and Haddad have identified the software engineering challenges in game development to be [8]: Diverse assets including both code and graphics and audio assets, a large project scope that can be difficult to define, high risk of game publishing, project management with a very tight schedules and involvement of many professions, inter-disciplinary team organization, development process that includes more than just software, and third-party technology. In this article we will mainly focus on project management, team organization, development process, and cost and complexity related to third-party technology. Similarly, Petrillo et al. found through a survey of game postmortems that game development and traditional software development suffers from similar main problems such as unrealistic scope, over budget, and loss of professionals [9]. A major difference found was that game development had a bigger problem with crunch time. Lewis and Whitehead describes the intersection of software engineering and game development and investigates four main areas: development of games, how games are designed, how middleware supports creative processes, and how games are tested [10]. This research focusing on two topics in our chapter: the difference between traditional and game development, and tool support for creative processes. As in traditional software development, a major challenge of game development is testing and dealing with bugs. Lewis, Whitehead and Wardrip-Fruin have established a taxonomy of video game bugs extracted from online user documentation [11]. The taxonomy covers bug areas like timing, position, graphical representation, change of value, artificial behavior, information, and action. There have also been several proposals from the research community to game developers to adopt software engineering practices, such as the use of ISO/IEC 29110 (lifecycle

profiles for very small entities) in game development [12]. So far, the impact of introducing new software engineering practices on the gaming industry has been limited.

To our knowledge, only one systematic literature review related to software engineering and game development has been conducted [5]. The goal of this review was to establish the state of the art on research concerning software engineering in the video games domain. The result of this literature review showed that the main emphasis in this research domain is on requirement engineering, as well as coding tools and techniques. Research related requirement engineering in games focuses on the problem of going from a game concept that should be fun and engaging to functional requirements, and software architectures and software designs that can produce game software realizing the game concept [13]. The initial requirements for a game can be labeled emotional requirements containing the game designer's intent and the means which the game designer expects the production team to induce that emotional state in the player [4]. Another area within software engineering and game development is research on coding tools and techniques including development of game engines [14-16], component-based game development [17], the use of game engines and tools [18-21], development of serious games [1, 22, 23], and challenges and solutions for networked multiplayer games [24-28]. Further, there are articles focusing on software architectures [29-36], and design patterns [37-39] for games. Such articles propose software architectures and/or design patterns to solve particular problems in game development. However, unlike our chapter, these articles say very little about the processes in which the architectures and patterns are used, and how the game development process is affected by various roles.

There are articles discussing the game development process and the involved roles. In [40], Scacchi presents how the free and open source development practices in the game community differs from traditional software engineering practices in that the process does not fit into a traditional life-cycle model or partially ordered as a spiral process. Also, the requirements are not found in requirement specification documents, but they are extracted from threaded messages or discussions on web sites. In [41], a survey of problems in game development is presented based on an analysis of postmortems (summaries of what went right and what went wrong in completed projects) written by various game developers. According to Flood, all game development postmortems say the same things: the project was delivered behind schedule; it contained many defects; the functionalities were not the ones that had originally been projected; and it took a lot of pressure and an immense number of development hours to complete the project [42]. This description also fits well with typical problems within conventional software engineering. Petrillo et al. further details the specific problems found in game development postmortems to be unrealistic scope, feature, cutting features during development, problems in the design phase, delays, technological problems, crunch time, lack of documentation, communication problems, tool problems, test problems, team building, number of defects, loss of professionals, and over budget [41]. The problems clearly differentiate game development from conventional software development are more issues related to unrealistic scope, feature creep, lack of documentation, and crunch time.

Several studies that have examined the tension between being creative and working structured and goal-oriented in game development [43]. In their study, Tschang and Szczypula found that game design results from individuals' creative actions, idea creation, constructivism, and evolution [44]. Another study of 65 project reports by Tschang showed that the same processes are used in game development as those used in the creation of other creative products [45]. A case study of how game studios produce games by Stacey and Nandhakumar, revealed that the game development process was a alternation of routine and improvisation [46]. The need for improvisation in game development is routed in the fact that game development is all about innovation and about producing a product the user will first of all enjoy and have a great experience with. The unique and close relationship between the product and the user in game development, produce a strong dependency between the game developer and the player community,

which is very different from traditional software development [47]. Although game development always has been strongly influenced by creativity and artistic freedom, a study based on interviews with the game industry by Kultima and Alha indicate a rise of instrumentalist views within game industry opposed to more artistic and personal view [48].

A study by Petrillo and Pimenta investigates if (and how) principles and practices from Agile Methods have been adopted in game development by analyzing postmortems of game development projects [49]. The conclusion of this study was that game developers are adopting a set of agile practices, although informally. This means that game developers can easily start using agile methods like Scrum and XP, since they have already several agile practices in place. One aspect of agile methods that is very relevant to our research is the emphasis on frequently gathering relevant stakeholders to bridge the gap between of all involved in the project [50]. This is related to our study where we investigated how the creative team, the technical team and the management collaborate and coordinate.

3 RESEARCH GOAL, QUESTIONS AND METHODS

The research method used in case study is based on the Goal, Question, Metrics (GQM) approach 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) [51]. The metrics used in our study is a mixture of qualitative and quantitative data [52].

The research goal of this study was defined as the following using the GQL template:

> The purpose of this study is to *examine how software architecture is used and how creative processed are managed* from the point of view of a *game developer* in the context of *video game development*.

The following research questions were defined by decomposing the research goal:

RQ1: What role does software architecture play in game development?
RQ2: How do game developers manage changes to the software architecture?
RQ3: How are the creative processes managed and supported in game development?
RQ4: How has game development evolved the last couple of years?

To find answers to the research questions, we used a combined approach that included a questionnaire, a follow-up survey and a literature study to support the findings. The questionnaire consisted of 20 statements where the respondents state whether they agree or not, using the Likert's scale [53]. In addition, the questionnaire provided a free text comment for every statement. The statements in the questionnaire were constructed from the research goal and the research questions presented above. The subjects of the study were recruited from the Nordic booth at the Game Developer Conference in San Francisco, as well as direct emails sent to game developers. The questionnaire was answered both on paper and using web-forms created using surveymonkey.com.

After receiving questionnaire responses, a follow-up survey with eight open-ended questions were sent out to the respondents from the questionnaire willing to give more detailed answers. The survey was conducted on the web only using surveymonkey.com.

4 RESULTS FROM QUESTIONNAIRE

This section presents the quantitative results from the questionnaire, comments from the respondents, as well reflections from the research literature. Responses from thirteen game companies were received. Figure 1 shows the distribution of number of employees of the thirteen companies. Only one big game developer responded (with 500+ employees), half of them (50%) had 5-10 employees, and the rest (42%) had between 1 and 5 employees. None of the game companies wanted their name to be public. The complete results from the questionnaire and the follow-up survey presented in next section can be found in [6].

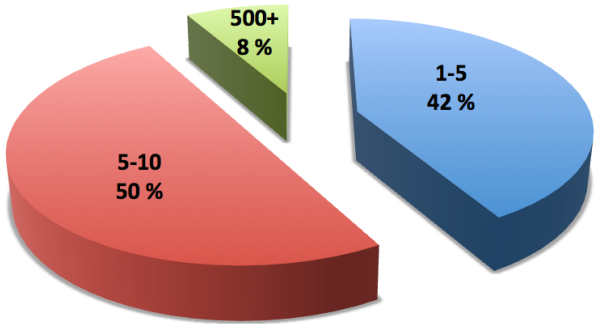

Fig. 1. Distribution of number of employees.

4.1 Design of Software Architecture (RQ1)

The first part of the questionnaire focused on the design of software architecture in game development, and the responses from the first six statements are shown in the Tables 1-6.

Table 1. "Design of software architecture is an important part of our game development process".

Agree	Neutral	Disagree
75%	16%	9%

This statement assessed the importance of software architecture among game developers. The result shows that most of the game developers in our study considered software architecture to be important part of game development. One comment from the respondents clearly illustrate the importance of software architecture in games (the given response on Likert's scale is shown in parenthesis):

- *"Oversight in the game software architecture may lead to serious dead ends, leading to a need to rewrite the entire system"*. (Agree)

In the early years of the video game industry, simple game software architectures were made by small teams of one to three persons. As game technology, user demands and the game industry have grown, careful design and planning of software architectures have become necessary to manage the complexity and size, and the involvement of many involved developers [2]. Although the majority of respondents represent smaller game developers, the complexity of game engines and use of other libraries and APIs demands a focus on software architectures to create platforms that can cope with changing requirements during the project [17]. Another reason software architecture has become very important in game development is the fact that the quality attributes are so important. It is impossible to have success with a game that suffers from bad performance (low and/or unstable framerates), bad usability, poor portability, poor testability (resulting in many bugs), and limited modifiability (hard to extend after release) [29]. In

addition, for massive multiplayer online games (MMOGs), quality attributes such as security (to avoid cheating) and availability in crucial for success [54]. Careful design and evaluation of the software architecture is the main approach to achieve predictable and acceptable quality attributes in software development [55].

The second statement in the questionnaire addressed the purpose of software architecture in game development (see Table 2).

Table 2. "The main goal of our software architecture is performance".

Agree	Neutral	Disagree
59%	16%	25%

The majority of the companies in the survey agreed that the main goal of the software architecture for them was to ensure sufficient performance. However, the respondents clarified that there are other quality attributes than performance that must be taken into account. Here are some comments regarding this statement:

- *"Performance plus functionality"*. (Agree)
- *"Also, future change, ability to be data-driven, optimized deployment processes, ease [of] automation/script-ability, and testability"*. (Agree)
- *"Main goals are: Performance, Memory consumption and Actual purpose of the software. Real time software as games must perform according to the platform requirements in order to see the light of the day regardless of the content"*. (Agree)

The next statement in the questionnaire investigated the relationship between the game concept and the software architecture (see Table 3).

Table 3. "Our game concept heavily influences the software architecture".

Agree	Neutral	Disagree
75%	9%	16%

Over three out of four game developers agreed that the game concept heavily influences the software architecture. This result was a bit surprising, as usage of game engines should ideally make the software architecture less dependent on game concept and game design. One respondent provided the following comment:

- *"Entirely depends on the game concept requirements, but in general: more generic – within boundaries – the better. This highlights that the importance of separating generic modules (core) with modules specific for a game (gameplay). Such an approach will allow reuse of core components, and at the same time provide sufficient freedom in development of game concept".* (Agree)

How much the game concept will influence the game software architecture is really a question about where to draw the line between the game and the game engine. Currently, most game engines target one or few game genres, such as real-time strategy games (RTS) or first-person shooters (FPSs). As there is

yet no taxonomy that can be used to specify all types of games, there exist few game engines that are independent of genres [29]. Plummer tries to overcome this problem by proposing a flexible and expandable architecture for video games not specific to a genre [31]. However, too general game engines will most likely have poor performance and heavy usage of memory due to overhead in the code. This means that the design of game engines must balance performance and use of resources vs. modifiability. Thus, games that stretches game genres will result in software architectures that deviate from the architecture of the game engine [29].

Statement four in the questionnaire asked whether the creative team is included in the design of the software architecture (see Table 4).

Table 4. "The creative team is included in the design of the software architecture".

Agree	Neutral	Disagree
75%	16%	9%

The large majority of the respondents agreed that the creative team is included in the design of the software architecture. It should be noted that the majority of the game developers in this survey were small companies, which makes it easier for the whole team to be involved in the whole development process. In smaller companies, many employees play several roles and work both with software development as well as creative design. A comment from one company with 5-10 employees highlights how the creative team can be involved in the software architecture:

- *"Only because I am a programmer and also the lead designer. Other creative people don't know enough to be productively included"*. (Agree)

It is interesting to note that the only large developer with 500+ employees said that they were neutral to this statement. We have seen at least three ways the creative team can contribute to the software architecture:

1) **Which game to make:** The decision of the game to make will give the foundation for the main constraints of the software architecture.
2) **New in-game functionality:** The creative team might request new in-game functionality that changes the software architecture.
3) **New development features:** Request for new development features (e.g. tool support and/or tool integration) might lead to changes in the software architecture.

Another comment related to this statement was:

- *"This is mostly true when working on the tools the creative team will be using. It rarely applies to in-game specific features."* (Agree)

Experiences from postmortems of game development projects show the importance of making the technical and creative team overlap going from game concept into developing the actual game software [41].

The next statement in the questionnaire is related to how well the creative team is supported by the tools provided for development (see Table 5).

Table 5. "Our existing software suite provides features aimed at helping the creative team do their job".

Agree	Neutral	Disagree
92%	8%	0%

The response concludes that the game engine and the supporting tools provide features helping the creative team in their work. This is further supported and refined in the comments:

- *"Our third-party tools do not do this, but we've developed in-house extensions that do."* (Agree)
- *"Use two software tiers that aims at very different levels of artist integration: Visual Studio and Unity3D"*. (Agree)

The latter comment describes the situation that it is not always the ideal tools the creative teams have to use. Ideally, the creative team should be supported by various GUI editors and high-level scripting. However, in practice, it might be necessary to dive into the source code to get the game where the creative team wants it to go. This process is normally carried out in collaboration with the technical team.

Table 6 shows the response to statement 6 regarding whether existing software architectures put restrictions on future game concepts.

Table 6. "Our existing software architecture dictates the future game concepts we can develop".

Agree	Neutral	Disagree
15%	47%	38%

A strong drive for game development is creativity and coming up with innovative fresh game concepts. The response shown in Table 6 shows that this is for the most part true. This topic relates to the core of how game companies see themselves and if they are constrained by existing technology, or can they create whatever the creative team comes up with. Half of the respondents are neutral to this statement, which show that in practice they have to go for the middle ground. Here are some comments to this statement:

- *"We have engines that gives us a great benefit when building new games and we would prefer to continue on the same engines. However, it doesn't fully dictate the games we will make in the future. This is primarily market-driven."* (Neutral)
- *"It may influence, but not dictate whenever possible."* (Neutral)
- *"It makes it a bit more expensive to go to certain genres, but that's it."* (Disagree)

The first two comments show that game developers want to be free to create whatever they want, but at the same time they are constrained by market, convenience and budget. The third comment (disagree) indicate that the influence exerted by the existing software architecture is a direct result of a cost-benefit trade-off. The higher cost of change, the more influence the existing software architecture exert on the game concepts.

4.2 Changes to the Software Architecture (RQ2)

The section presents on results from the questionnaire on statements on how game developers cope with changes to the software architecture shown in the Tables 7-12. Table 7 shows the response to a statement on whether the creative team's ideas are restricted by existing game engine.

Table 7. "The creative team has to adopt their ideas to the existing game engine".

Agree	Neutral	Disagree
31%	46%	23%

No clear conclusion could be drawn based on how the game companies responded. However, here are some comments might explain the divergence in the responses:

- *"Technical realities are always something the creative side has to work around."* (Agree)
- *"Depending on structure. For assets handling, yes, but creatively, not so much. In latter case, the challenge is put to programmers to extend usage."* (Neutral)
- *"Most of the time, the creative team is not fully aware of the game engine limitation's, so it is not their job to make it work by locking the creativity to things known to have been done with the engine before, the people who implements just need to make the ideas work one way or another."* (Disagree)
- *"That is not the way we do it here. The game design comes first, then we build what is necessary to make it happen."* (Disagree)

These comments indicate a trade-off between creative freedom and the technical limitations. It is axiomatic that if an idea not supported in the current technology should be implemented, either the idea has to be adapted to the existing technology, the technology adapted to the idea, or something in between. Which one to be chosen depends on a cost-benefit analysis.

The next statement asked if changes of the software architecture can be demanded by the creative team (see Table 8).

Table 8. "During development, the creative team can demand changes to the software architecture".

Agree	Neutral	Disagree
69%	31%	0%

The majority of the respondents agree that the creative team can demand changes to the software architecture and none disagreed to this statement. There were two comments to this statement:

- *"Depends how far in development and how big of a change, the odds of re-factoring an entire system late in production are close to nil, but the development team keeps an open mind at all times."* (Neutral)
- *"But again, only because the head of the creative team is president of the company and also wrote the original version of the game engine. If someone who doesn't know how to program were to come to me and demand changes to the software architecture, I would probably not listen very seriously."* (Agree)

Based on the comments two statement in Table 8, game developers are inclined to prioritize the wants and needs of the creative team, given that the cost-benefit trade-off is favorable. Another important issue is the phase the project is in. The later in the project (production), less changes and request from the creative team is possible. Boehm and Basili estimate that requirements error can cost up to 100 times more after delivery if caught at the start of the project [56]. A possible solution to this problem is to spend more time in the preproduction phases (25%-40% of the project time) before moving to production, as it would leave relatively few surprises in the production phase [57]. In practice, this might be very difficult as the majority of playtesting of the game happens in the production phase, which might reveal major problems with the core gameplay.

Table 9 shows the response to decides if change-request from creative team should be implemented.

Table 9. "Who decides if change-requests from the creative team are implemented?"

Technical team	Management	Creative team
10%	40%	50%

The responses to this question were mainly divided between management and the creative team. Here are the respondents' comments to this statement:

- *"Ultimately, the management can overrule everybody, but I would like to check the 3 options here, the creative team judges how important the change is, the technical team decides if it is realistic, and the management makes sure it can be afforded. So mostly, it is a team decision."* (Management)
- *"Actually, it is all of the above, but the question would not let me put that as an answer."* (Management)
- *"Sort of. The technical team advices what is possible, and as such has the final word. If it is possible, the decision falls on management, as it is usually related to economic costs."* (Technical team)
- *"Depends very much on the scale of change, we try as much as possible to keep this within and as a dialogue between the tech/creative teams, but if it means major change it goes to management. We also aim to be as much product/feature driven as possible, as the primary owner is in the creative team."* (Creative team)

The responses from the developers indicate that all three branches (administration, technical and creative) are involved in change decisions. More game developers have also started to adopt agile development practices, where it is more common to have frequent planning and decision meetings where various professions are involved [58].

The next statement asks whether the technical team implements all features required by the creative team (see Table 10).

Table 10. "The technical team implements all features requested by the creative team".

Agree	Neutral	Disagree
75%	15%	8%

The majority of the game companies agreed that the technical team implements all features requested by the creative team. There were several comments to this statement that provide more details:

- *"It can happen that the creative team contributes on technical aspects during prototyping phase. Production quality code is however left to the technical people."* (Agree)
- *"Of course, if the requests are decided to be implemented in the first place."* (Agree)
- *"It's very much a dialogue, we try not to have too formal split between tech and creative team when thinking about this but prioritize what the user experience should be and when we can ship at target quality."* (Agree)
- *"Some requested features are not tech. feasible."* (Disagree)

Table 11 shows the results from the statement regarding adding features to a near complete game engine.

Table 11. "It is easy to add new gameplay elements after the core of our game engine has been completed".

Agree	Neutral	Disagree
82%	18%	0%

The majority of the respondents agree that is easy to add new gameplay elements after the core game engine has been completed. However, the comments related to this statement suggest that adding new gameplay elements after completing the core game engine is often not possible, recommended or wanted:

- *"It is simple during prototyping phase, technology-wise. However, from a game concept point of view, it is highly dis-recommended and the fact it is simple does not motivate the team to stack up features because the existing one are just not convincing enough :)"* (Agree)
- "*This really depends a lot and can only be answered on a case to case effect."* (Neutral)
- *"Depends on the type of element – some may require significant underlying engine changes."* (Neutral)

One of the most common motivations for designing of software architecture is to provide a system that is easier to modify and maintain. In game development, modifiability must be balanced with performance. There are mainly two contrasting approaches to design modifiable game environments [59]: 1) *Scripting* that requires developers to anticipate, hand-craft and script specific game events; and 2) *Emergence* that involves defining game objects that interact according to rules to give rise to emergent gameplay. The most common approach is to create or acquire a game engine that provides a scripting language to create a game with predefines behavior. The emergence approach involves creation of a simulation of a virtual world with objects that reacts to their surroundings. The use of scripting makes it complex to add new gameplay elements, as everything is hardwired. The emergence approach makes it much easier to add new gameplay elements later in the project, with the price of being harder to test (large number of possible game object interactions).

The next statement focused on the creative team's use of existing tools and features during development (see Table 12).

Table 12. "During development, the creative team has to use the tools and features already available".

Agree	Neutral	Disagree
47%	15%	38%

There is not possible to draw a conclusion based on the results from the statement above. The comments from the respondents give more insights:

- *"The ones already available and the ones they request along the way."* (Agree)
- *"New tools can be made. However, it is certainly best to keep within the suite offered."* (Disagree)
- *"Our current engine (Unity) is easily extensible."* (Disagree)

This statement is really about cost. Adding new tools and features during development is costly and might also add risk to the project. However, in some cases new tools and features must be added to get the wanted results. The only large company with 500+ employees responded neutral to this statement to indicate that it depends on the circumstances.

4.3 Supporting the Creative Processes (RQ3)

The responses on statements presented in Table 13-16 relate to how creative processes are supported through technology and processes. Table 13 shows the responses to a statement whether the game engines used by the game companies support dynamics loading of new content (not require recompiling or build).

Table 13. "Our game engine supports dynamic loading of new content".

Agree	Neutral	Disagree
92%	8%	0%

The response from the game developers shows that current game engines allow dynamic loading of new content. However, the comments to this statement show that there are some restrictions in terms of when and how it can be done:

- *"At some extent, in editor mode yes, at run-time only a subset of it."* (Agree)
- *"With some constraints, content must be properly prepped of course."* (Agree)

Different game engines provide different flexibility regarding changes that can be carried out at run-time. Most game engines support changes to the graphic as long as the affected graphical structures are the similar. Similarly, many game engines allow run-time changes using a scripting language that can change the behavior of the game. However, substantial changes to the gameplay and changes of the game engine itself usually cannot be changed in run-time.

The next statement asked if the game engine used had a scripting system than can be used by the creative team (see Table 14).

Table 14. "Our game engine has a scripting system the creative team can use to try out and implement new ideas".

Agree	Neutral	Disagree
70%	15%	15%

Most of the respondents say they have a scripting system that can be used by the creative team. However, there are also game developers in the survey that use own game engines without scripting capabilities. Especially for small game developers, it can be too expensive and too much work to create support for

scripting in their own game engine. In addition, small game developers do not necessarily have to competence to develop such flexible game engines. The comments related to this statement were:

- *"Yes, but could be better and more flexible (as always...)"* (Agree)
- *"Our 'scripting system' is typing in C++ code and recompiling the game."* (Disagree)

A recognized problem of letting the creative team script the game engine, is that they usually do not understand the underlying low-level mechanisms related to performance [60]. Until the game engines can optimize the scripts automatically, the technical team often must assist the creative team with scripting.

Table 15 shows the results to a statement that asked whether the creative team was included in the company's development feedback loop or not.

Table 15. "The creative team is included in our development feedback loop (e.g., scrum meetings)".

Agree	Neutral	Disagree
91%	9%	0%

As the majority of the game developers in this survey are rather small organizations, it is naturally that the creative team is included in the development feedback loop. However, even the large game developer in the survey (500+ employees) said that the creative team was included in development feedback loops. This is in alignment with what has been found in other studies [40, 49, 58]. The only comment related to this statement was:

- *"Depends on the phase of the project."* (Neutral)

The next statement focused on rapid prototyping using game engines (see Table 16).

Table 16. "Our game engine allows rapid prototyping of new levels, scenarios and NPCs/behavior".

Agree	Neutral	Disagree
91%	9%	0%

This statement is related to the statement in Table 14, and the response was also the same. Game engines supporting scripting normally provide rapid prototyping. There was only one comment related to this statement:

- *"While most of the systems are designed with simplicity and fast iteration time in mind, certain things still require time consuming tweaking tasks."* (Agree)

4.4 Changes over Time (RQ4)

Tables 17-20 shows the response to statements that investigate how game development has changed the last couple of years. Table 17 shows the results from statement on the change of usage of 3rd party modules.

Table 17. "Today our company uses more 3rd party modules than 3 years ago".

Agree	Neutral	Disagree
67%	22%	12%

The response to this statement seemed to be that the majority uses more third-party modules than 3 years ago. This confirms the predictions that buying a good middleware will provide a better result than what an organization can produce at the same prize [61]. The only comment to this statement was "It is about time …" for more usage of third party modules.

Table 18 shows the results from statement on whether game developer has become easier the recent years.

Table 18. "It is easier to develop games today than it was 5 years ago".

Agree	Neutral	Disagree
77%	8%	15%

The vast majority in the survey agrees that it is easier to develop games today than it was 5 years ago. The complexity of games and the players' expectations have increased over the years [2], but the tools and the engines have also made it easier to manage complexity as well as achieving higher fidelity. The comments from the respondents highlight that the technical part has probably become easier, but the overall challenge of game development probably not:

- *"The challenges have changed, and the quality bar has risen, it is more accessible to people less interested in nerdy things nowadays (engines like Unity reduced/removed the low-level aspect of the development), but developing a great game is still as challenging as before, the problems to solve just have evolved."* (Disagree)
- *"Technically and graphically, yes. Conceptually, no."* (Agree)

The next statement investigates how the importance of middleware has changed over time (see Table 19).

Table 19. "Middleware is more important to our company today than 3 years ago".

Agree	Neutral	Disagree
65%	18%	18%

The majority of respondents agreed that middleware is more important to the company today than 3 years ago. Table 20 shows the results from a statement comparing how game development and software development has changed over time.

Table 20. "Game development is more like ordinary software development today than 5 years ago".

Agree	Neutral	Disagree
38%	24%	38%

The feedback on this statement was mainly divided into two camps. It is interesting to note that the large game developer with 500+ employees answered neutral to this statement. The only comments to this

statement came from those disagreeing that game development is becoming more like ordinary software development:

- *"Game development requires a more eccentric creative problem solving than development in most of other industries and this will probably remain true forever ;)"* (Disagree)
- *"Nope. It was software development then, and still is now."* (Disagree)
- *"I think the tools available today moves game development further away from 'ordinary software development'.)"* (Disagree)

Several differences between game development and conventional software development have been identified in the literature. One example is that games usually have more limited lifecycle than conventional software products and that the maintenance of games mainly only focuses on bug fixing without charging the end-user [62]. Another example is that game development does not include functional requirements from the end-users. Typical end-user requirements to a game is that the game must be fun and engaging [13]. The latter poses a challenge of going from preproduction phase that produces a game design document (and maybe a prototype), to the production phase where all the software, game design, art, audio and music will be produced [13]. From a software engineering point of view, a challenge in game development is to create functional requirements from a game design document that describes the game concept. Another difference between conventional software systems and games is the importance of usability. A software system might be used if it provides much needed functionality even if the usability is not the best. However, a game with low usability is very unlikely to survive [63]. Usability tests and frameworks are also used within game development, but they are tailored specifically for the game domain [64, 65].

5 RESULTS FROM FOLLOW-UP SURVEY

In this section you can find a summary of the results from a more in-depth survey with free text questions targeted six of the thirteen game developers from that responded to the questionnaire in previous section.

5.1 Game Engines and Middleware

Four out of the six respondents said they use external game engines where two use custom-made or their own. External game engines used by the respondents were Away3D (3D engine for Flash/ActionScript), Unity 3D and Unreal Engine 3. In addition to these game engines, a variety of external tools are being used such as Autodesk Beast (lightning), Autodesk Scaleform (user interfaces), Bink (video codec for games), Box2D (physics), DirectX (multimedia API), FMOD (audio), libvorbis (audio codec), NVIDIA PhysX (physics), SpeedTree (plugin/tools for tree and plants), Substance (texture designer), Flash (Web-platform), and Umbra (rendering optimization).
In the survey we asked about where game engines are heading in the future, and the following key-points summarize their responses:

- **Multi-platform:** The ability to create a game once and build it to run on different platforms allows game developers to reach a much larger audience, and at the same time being able to focus on the work of creating the game without always considering porting.
- **Quality of features:** Whilst most game engines today frequently present new features, the quality of the feature is more important than the quantity. Even if a really impressive, bleeding-edge feature is included in the game engine, most game developers will not use it until it works properly and is simple to integrate in the game.

- **Simplicity:** The usability of a game engine has improved rapidly since the earliest game engines to those who dominate the market today. The replies indicate that this trend will only continue, and that game engines, which are difficult to use, will fall behind in the competition. However, ease of use must not be at the expense of freedom. As there are limits to how much freedom a point-and-click interface can provide, the companies should still be able to edit the source code, allowing them to develop new and novel features.
- **Completeness:** A game engine today must present more than "just" a rendering engine, which accepts input data, and produces a game. The game engine needs to have a host of supporting features and tools, relieving the individual organizations from tasks like taking models from modeling tools and converting them to game engine-compatible data formats, or handling save games.

5.2 Software Architecture and Creative Team

Two recurring themes were recognized in how the creative team contributes to the design of the software architecture. *Firstly*, the creative team affects the software architecture indirectly through working with the technical team. *Secondly*, the main areas they affect relate to how tools interact with the game. This can be a result of discussions regarding workflow issues or based on the functional needs of the creative teams. Thus, the creative team does not affect the software architecture directly, but through requests made to the technical team.

To specify more in detail how the creative team affects the software architecture, we asked about which features is needed to help the creative team do their job. The responses showed that all companies desire functionality letting the creative team import new assets and try them out in-game. This allows rapid prototyping of new ideas, which again demands a software architecture that can provide such run-time flexibility. The goal for many is to achieve a more data- and tool-driven development that empowers the creative team. A part of this process is to achieve automatic transition from tools to the game. In practice, this means that the creative team can test out new ideas faster and more frequently, and thus produce better and more original games. Additionally, if the creative team possesses some programming skills, they could alter the source code on their own copy of the game project. This allows for a more fundamental approach to implement new features. Alternatively, "featured-oriented teams" can be used in game development, as suggested by one of the respondents. Such a team consists of at least one coder, one artist and one designer. The composition of roles allows them to focus on particular features represented as a single unit, allowing work to progress quickly without having to wait for any external resources. The use of feature-oriented teams is also a way of reducing the problems related the transition from preproduction to production in game development [13]. An overlap in roles in technical and creative teams is also recommended to bridge the code/art divide that many game development projects suffer from [66].

5.3 Implementing Changes

From the feedback from the respondents we recognized a pattern for the decision process on how companies are reasoning about implementing changes. *Firstly*, the importance of the feature from a user experience perspective must be assessed through asking how much better will the game be with this feature or how much will be lost if it is not implemented? *Secondly*, it must be asked how much it will cost in terms of time and resources to implement this feature, and can the added workload and extra use of resources be justified? If both parts evaluate positively, the organization will start considering how the features should be implemented. This process starts in a discussion involving both the creative and technical team. Here the initial goal as seen from the creative team, is subjected to technical

considerations. Based on this feedback and feed-forward, the creative team ends up with a specification of the features. The technical team will produce a prototype based on this specification. When both teams are happy with the prototype, it is fixed into production quality code.

The last topic we touched upon in this survey was about who are involved in the decision process and how important are the opinions of the creative team, the technical team, and the management. The responses from the survey gave some indicators for how these decisions take place in game development companies. *Firstly*, management has the final say if the change significantly alters budget or time estimates. This is not to say that it is not done without involvement from either the creative team or the technical team, but in the end, management decides. *Secondly*, for the companies that replied in our survey, all three groups (management, creative and technical) seem to be treated equally in the decision process. This makes sense as these three groups have three different responsibilities. Management should get the game launched on time and budget, the creative team should produce a game which is fun and involving, and the technical team should enable the technology to drive the creative team's content through in a reliable way.

6 THREATS TO VALIDITY

This section addressed the most important threats to the validity. There are mainly three validities that must be discussed: intern, construct, and external.

The *intern 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" [67]. If changes in B have causes other than the manipulation of A, there is a threat to the internal validity. The main internal threat in our study is that the sample of subjects in the experiment was not randomized. The respondents to the questionnaire were recruited in two ways. The first group was game developers that visited the Nordic booth at the Game Developer Conference that volunteered to fill out a paper questionnaire at the booth. The second group consisted of game developers that responded to emails we sent to many game developers. We would have preferred more respondents and especially more larger game developers. However, we have learned that it is very difficult to get game developers to respond to questionnaires as they are always behind schedule and overworked, so we were pleased getting thirteen responses in the end.

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 [67]. Our approach was to first create a questionnaire with the goal of answering our research questions that were decomposed from our research goal. The goal of our research was not only to get quantitative responses, so we encouraged the respondents to comment on how they answered the twenty statements. Further to get more qualitative data, we conducted a survey to those game companies who were willing to go more into details.

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 [67]. The results found in this study can mainly be generalized to smaller game companies (from 1 to 10 employees), since we only had one large game developer among the respondents. Also, since we only received thirteen responses to the questionnaire, the quantitative results must only be seen as indicators on how game developers think about the various

statements. The qualitative data in the questionnaire and survey along support from research literature strengthen the results and its validity.

7 CONCLUSIONS

This chapter presents the findings from a questionnaire and the follow-up survey on how game developers use and manage software architecture and how creative development processes are managed. The results presented are a combination of the response from thirteen game companies and findings in research literature.

The *first* research question (RQ1) addressed the role software architecture plays in game development. The game developers in our survey stated that software architecture is important in game development, especially to manage complexity and achieve quality attributes such as modifiability and performance. Another finding was that the game concept heavily influences the software architecture mainly because it dictates the choice of game engine. Further, the ways the creative team can affect the software architecture is through the creation of the game concept, by adding in-game functionality, and by adding new development tools. Finally, a cost/benefit analysis will decide whether an existing software architecture may dictate future game concepts or not. Whenever it is possible, reuse of the software architecture is wanted.

The *second* research question (RQ2) investigated how game developers manage changes to the software architecture. We found that the creative team has to adjust some extent their game ideas to existing software architecture based on a cost/benefit analysis. The creative team can demand changes to the software architecture during development, but this decision depends on how far the project has progressed and the cost and benefit of making the change. Decisions on change-requests are usually made by personnel from technical team, creative team and management, but the management has the final word due to economical justifications. Further, the technical teams to a large extent implement all features and tools requested by the creative team (within reasonable limits), and that most developers said it was easy to add new gameplay elements after the core game engine was complete (although not recommended late in the project). The literature highlighted two approaches to deal with adding gameplay elements to a game: Scripting – where the behavior of the game is pre-deterministic and acting according to a script, and Emergence - where the behavior is non-deterministic, and a virtual world is created by game objects that reacts the environment around them. The former has the advantage of being easier to test, and the latter has the advantage of being easier to extend gameplay.

The *third* research question (RQ3) addressed how the creative processes are managed and supported in game development. Most of the game developers in this study said they used game engines that support dynamic loading of new game elements (although not everything in run-time). The majority of the respondents use game engines that support scripting. Only game developers with own developed game engines did not support scripting. Finally, the majority of the developers said they used game engines that enable rapid prototyping of new ideas. The conclusion of this research question is that current game engines enable creative processes through support of GUI tools, scripting, and dynamic loading of elements.

The *forth* research question asked how game development has evolved the last couple of years. This question can be summarized as follows: There has been an increased use of third-party software, middleware has become more important, and it is technically easier to develop games. Although the majority of respondents said that the technical aspects of game development have become easier, game development in itself has not become easier due to higher player expectations and higher game

complexity. Similarly, there was no clear conclusion whether game development has become more like conventional software development. The main differences were identified to be that in game development there are no real functional requirements, the quality attributes performance and usability are more important, and game development has its own set of tools and engines.

ACKNOWLEDGEMENTS

We would like to thank Richard Taylor and Walt Scacchi at the Institute for Software Research (ISR) at the University of California, Irvine (UCI) for providing a stimulating research environment and for hosting a visiting researcher from Norway.

REFERENCES

[1] M. Zyda, "From visual simulation to virtual reality to games," *Computer,* vol. 38, no. 9, pp. 25-32, 2005.
[2] J. Blow, "Game Development: Harder Than You Think," *Queue,* vol. 1, no. 10, pp. 28-37, 2004.
[3] C. E. Crooks, *Awesome 3D Game Development: No Programming Required*: Cengage Learning, 2004.
[4] D. Callele, E. Neufeld, and K. Schneider, "Emotional Requirements," *IEEE Softw.,* vol. 25, no. 1, pp. 43-45, 2008.
[5] A. Ampatzoglou, and I. Stamelos, "Software engineering research for computer games: A systematic review," *Information and Software Technology,* vol. 52, no. 9, pp. 888-901, 2010.
[6] N. Nordmark, *Software Architecture and the Creative Process in Game Development,* Master Thesis, Norwegian University of Science and Technology, 2012.
[7] A. I. Wang, and N. Nordmark, "Software architectures and the creative processes in game development." pp. 272-285.
[8] C. M. Kanode, and H. M. Haddad, "Software engineering challenges in game development." pp. 260-265.
[9] F. Petrillo, M. Pimenta, F. Trindade, and C. Dietrich, "Houston, we have a problem...: a survey of actual problems in computer games development." pp. 707-711.
[10] C. Lewis, and J. Whitehead, "The whats and the whys of games and software engineering." pp. 1-4.
[11] C. Lewis, J. Whitehead, and N. Wardrip-Fruin, "What went wrong: a taxonomy of video game bugs." pp. 108-115.
[12] J. Kasurinen, R. Laine, and K. Smolander, "How applicable is ISO/IEC 29110 in Game Software Development?." pp. 5-19.
[13] D. Callele, E. Neufeld, and K. Schneider, "Requirements engineering and the creative process in the video game industry." pp. 240-250.
[14] M. Shantz, "Designing a PC game engine," 1998.
[15] T. C. Cheah, and K.-W. Ng, "A practical implementation of a 3D game engine." pp. 351-358.
[16] R. Darken, P. McDowell, and E. Johnson, "The Delta3D Open Source Game Engine," *IEEE Comput. Graph. Appl.,* vol. 25, no. 3, pp. 10-12, 2005.
[17] E. Folmer, "Component Based Game Development–A Solution to Escalating Costs and Expanding Deadlines?," *Component-Based Software Engineering*, pp. 66-73: Springer, 2007.
[18] C. A. M. Antonio, C. A. F. Jorge, and P. M. Couto, "Using a Game Engine for VR Simulations in Evacuation Planning," *IEEE Comput. Graph. Appl.,* vol. 28, no. 3, pp. 6-12, 2008.
[19] J. Kasurinen, J.-P. Strandén, and K. Smolander, "What do game developers expect from development and design tools?." pp. 36-41.
[20] M. Zhu, and A. I. Wang, "RAIL: A Domain-Specific Language For Generating NPC Behaviors In Action/Adventure Games," in 14th International Conference on Advances in Computer Entertainment Technology (ACE 2017), London, UK, 2017.
[21] M. Zhu, A. I. Wang, and H. Trætteberg, "Engine-cooperative game modeling (ecgm): Bridge model-driven game development and game engine tool-chains." p. 22.
[22] A. I. Wang, "The wear out effect of a game-based student response system," *Computers & Education,* vol. 82, pp. 217-227, 2015.
[23] A. I. Wang, and J. d. J. L. G. Ibánez, "Learning Recycling from Playing a Kinect Game," *International Journal of Game-Based Learning (IJGBL),* vol. 5, no. 3, pp. 25-44, 2015.
[24] C. Bouras, V. Poulopoulos, I. Sengounis, and V. Tsogkas, "Networking Aspects for Gaming Systems," in Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services, 2008.
[25] J. Smed, T. Kaukoranta, and H. Hakonen, *A Review of Networking and Multiplayer Computer Games,* TUCS Technical Report 454, Turku Centre for Computer Science, 2002.
[26] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, Singapore, 2006.
[27] T. Triebel, B. Guthier, R. Süselbeck, G. Schiele, and W. Effelsberg, "Peer-to-peer infrastructures for games," in Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Braunschweig, Germany, 2008.
[28] W. Cai, P. Xavier, S. J. Turner, and B.-S. Lee, "A scalable architecture for supporting interactive games on the internet." pp. 60-67.

[29] E. F. Anderson, S. Engel, P. Comninos, and L. McLoughlin, “The case for research in game engine architecture,” in Proceedings of the 2008 Conference on Future Play: Research, Play, Share, Toronto, Ontario, Canada, 2008.
[30] S. Caltagirone, M. Keys, B. Schlief, and M. J. Willshire, “Architecture for a massively multiplayer online role playing game engine,” *J. Comput. Small Coll.,* vol. 18, no. 2, pp. 105-116, 2002.
[31] J. Plummer, “A flexible and expandable architecture for computer games,” Arizona State University, 2004.
[32] W. Piekarski, and B. H. Thomas, "An object-oriented software architecture for 3D mixed reality applications." p. 247.
[33] A. I. Wang, “Extensive Evaluation of Using a Game Project in a Software Architecture Course,” *Trans. Comput. Educ.,* vol. 11, no. 1, pp. 1-28, 2011.
[34] W. Cai, M. Chen, and V. C. Leung, “Toward gaming as a service,” *IEEE Internet Computing,* vol. 18, no. 3, pp. 12-18, 2014.
[35] M. Zhu, A. Wang, H. Guo, and H. Trætteberg, “Graph of Game Worlds: New Perspectives on Video Game Architectures,” *Manuscript submitted for publication*, 2012.
[36] M. Zhu, A. I. Wang, and H. Guo, “From 101 to nnn: a review and a classification of computer game architectures,” *Multimedia systems,* vol. 19, no. 3, pp. 183-197, 2013.
[37] P. V. Gestwicki, “Computer games as motivation for design patterns,” *SIGCSE Bull.,* vol. 39, no. 1, pp. 233-237, 2007.
[38] A. Ampatzoglou, and A. Chatzigeorgiou, “Evaluation of object-oriented design patterns in game development,” *Information and Software Technology,* vol. 49, no. 5, pp. 445-454, 2007.
[39] D. Nguyen, and S. B. Wong, “Design patterns for games,” in Proceedings of the 33rd SIGCSE technical symposium on Computer science education, Cincinnati, Kentucky, 2002.
[40] W. Scacchi, “Free and Open Source Development Practices in the Game Community,” *IEEE Softw.,* vol. 21, no. 1, pp. 59-66, 2004.
[41] F. Petrillo, M. Pimenta , F. Trindade, and C. Dietrich, “What went wrong? A survey of problems in game development,” *Computer Entertainment (CIE),* vol. 7, no. 1, pp. 1-22, 2009.
[42] K. Flood, “Game unified process,” *GameDev. net*, 2003.
[43] F. T. Tschang, “Balancing the tensions between rationalization and creativity in the video games industry,” *Organization science,* vol. 18, no. 6, pp. 989-1005, 2007.
[44] F. T. Tschang, and J. Szczypula, “Idea creation, constructivism and evolution as key characteristics in the videogame artifact design process,” *European management journal,* vol. 24, no. 4, pp. 270-287, 2006.
[45] F. T. Tschang, “Videogames as interactive experiential products and their manner of development,” *International Journal of Innovation Management,* vol. 9, no. 01, pp. 103-131, 2005.
[46] P. Stacey, and J. Nandhakumar, “A temporal perspective of the computer game development process,” *Information Systems Journal,* vol. 19, no. 5, pp. 479-497, 2009.
[47] T. Burger-Helmchen, and P. Cohendet, “User communities and social software in the video game industry,” *Long Range Planning,* vol. 44, no. 5-6, pp. 317-343, 2011.
[48] A. Kultima, and K. Alha, ""Hopefully everything I'm doing has to do with innovation": Games industry professionals on innovation in 2009." pp. 1-8.
[49] F. Petrillo, and M. Pimenta, "Is agility out there?: agile practices in game development." pp. 9-15.
[50] K. Schwaber, and M. Beedle, “Agile Software Development with Scrum,” 2002.
[51] V. R. Basili, *Software modeling and measurement: the Goal/Question/Metric paradigm*, University of Maryland for Advanced Computer Studies, 1992.
[52] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*: Springer, 2012.
[53] R. Likert, “A technique for the measurement of attitudes,” *Archives of psychology*, 1932.
[54] T.-Y. Hsiao, and S.-M. Yuan, “Practical Middleware for Massively Multiplayer Online Games,” *IEEE Internet Computing,* vol. 9, no. 5, pp. 47-54, 2005.
[55] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed.: Addision-Wesley, 2012.
[56] B. Boehm, and V. R. Basili, “Software defect reduction top 10 list,” *Foundations of empirical software engineering: the legacy of Victor R. Basili,* vol. 426, 2005.
[57] E. Bethke, *Game Developer's Guide to Design and Production*: Wordware Publishing Inc., 2002.
[58] P. Stacey, and J. Nandhakumar, “Opening up to agile games development,” *Communications of the ACM,* vol. 51, no. 12, pp. 143-146, 2008.
[59] P. Sweetser, and J. Wiles, “Scripting versus emergence: issues for game developers and players in game environment design,” *International Journal of Intelligent Games and Simulations,* vol. 4, no. 1, pp. 1-9, 2005.
[60] W. White, C. Koch, J. Gehrke, and A. Demers, “Better scripts, better games,” *Communications of the ACM,* vol. 52, no. 3, pp. 42-47, 2009.
[61] A. Rollings, and D. Morris, *Game Architecture and Design - A New Edition*: New Riders Publishing, 2004.
[62] M. McShaffry, *Game coding complete*: Cengage Learning, 2013.
[63] J. L. G. Sánchez, N. P. Zea, and F. L. Gutiérrez, "From usability to playability: Introduction to player-centred video game development process," *Human Centered Design*, pp. 65-74: Springer, 2009.
[64] H. Desurvire, and C. Wiberg, "Game usability heuristics (PLAY) for evaluating and designing better games: The next iteration," *Online Communities and Social Computing*, pp. 557-566: Springer, 2009.

[65] S. Laitinen, "Better games through usability evaluation and testing," *Gamasutra. URL: http://www.gamasutra. com/features/20050623/laitinen_01. sht ml*, 2005.
[66] J. Hayes, "The code/art divide: How technical artists bridge the gap," *Game Developer Magazine,* vol. 14, no. 7, pp. 17, 2007.
[67] W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and quasi-experimental designs for generalized causal inference*: Wadsworth Cengage learning, 2002.