Software Architectures and the Creative Processes in Game Development

Alf Inge Wang and Njål Nordmark,

Norwegian University of Science and Technology,

N7491 Trondheim, Norway

alfw@idi.ntnu.no / njaal.nordmark@gmail.com

ABSTRACT

Game development is different from traditional software engineering in that there are no real functional requirements and the customers buy and use the software only because it is engaging and fun. This article investigates how game developers think about and use software architecture in the development of games. Further, it looks at how creative development processes are managed and supported. The results presented in this article come from responses to a questionnaire and a survey among thirteen game developers. The research questions answered in this study are: what role does the software architecture play in game development, how do game developers manage changes to the software architecture, how are creative development processes managed and supported, and how has game development evolved the last couple of years. Some of our findings are that software architectures play a central role in game development where the focus is mainly 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 represented, and game developers 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.

1. Introduction

Game development can be incredibly challenging as game technology such as game engines and game platforms changes rapidly, and code modules crafted for specific games offer less than 30 percent reuse [1]. In the early days of the video games era, game development was carried out by small teams, where the software architectures was just made out of a few modules such as 2d graphics, simulation, sound, streaming of i/o and main. At this time, there was not much focus on software architecture and software engineering, but rather on how to how to create an interesting game with the limited hardware resources available. The success of the video game industry, the players' wish for more, and the advancement of hardware have resulted in large and complex games developed

by large teams of multiple professions. The evolution of games has also resulted in an evolution of game architectures that have grown in size and complexity [2]. Today, some game projects are very large, and the game software itself has a complex software architecture with many interconnected modules. 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]. Further, the development of AAA games (major game titles) requires 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 (e.g. the technical team vs. the creative team) poses challenges for how a game is developed. In addition to game development being a big and innovative industry, all the abovementioned challenges make game development in interesting area for software engineering research.

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 article presents a study on how game developers think about and manage software architecture and the creative processes that characterize game development. The study investigates the relationship between creative design and software development, and how the technical and creative teams collaborate. Our approach to research this field has been to contact game developers and ask them to respond to a questionnaire. Further, those who were willing were asked to do a follow-up survey where in-depth questions were asked. Along with our own experience from game development and with support from research literature, we draw a picture of how game developers work with and manage software architecture and the creative development process. To our knowledge, this is the first study of this kind within software engineering.

The rest of the article is organized as follows. Section 2 presents the research related to what being presented in this article as well as research goal, research questions and research method. Section 3 presents the results from the questionnaire and the survey. Section 4 discusses the validity of the research. Section 5 concludes the article.

2. Material and Method

In this section, we present other research relevant to this article as well as the research goal, the research questions and the research method used.

2.1 Related Work

As far as we know, there are no similar studies that focus on software architecture and creative processes in game development. 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 [6]: *Diverse Assets* – game development is not simply a process of producing source code, but involves assets such as 3D modes, textures, animations, sound, music, dialog, video, *Project Scope* – the scale of a video game can be massive and

the game industry is known for poorly describing and defining project scope, *Game Publishing* - brining a video game to market involves convincing a game publisher to back up financially that will affect the deliveries and development process, *Project Management* – project management in game development can be extra hard due to very tight schedules and involvement of many professions, *Team Organization* - involves building teams that enhances communication across disciplines, *Development Process* – the development process includes more than just software development, and *Third-Party Technology* – for many game developers third party software represents the core of the project due to rising development costs and increasing complexity. In this article we will mainly focus on project management, team organization, development process, and third-party technology.

Only one systematic literature review concerning game development was found [5]. This literature review assessed the state of the art on research concerning software engineering for video games. 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. Articles related to requirement engineering 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 [7]. The initial requirements for a game can be labeled emotional requirements that contain 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]. Research on coding tools and techniques include articles on development of game engines [8-10], component-based game development [11], the use of game engines [12], development of serious games [1], and challenges and solutions for networked multiplayer games [13-17]. There are also several articles that focus on software architectures for games [18-20], and design patterns for games [21-23]. These articles propose various software architectures and/or design patterns to solve problems in game development. However, unlike our article, they say very little about the processes in which the architectures and patterns are used, and how the various roles in game development affect them.

There are also articles that focus on the game development process and the involved roles. In [24], 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 take form of threaded messages or discussions on web sites. In [25], a survey of problems in game development is presented. The survey is 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 [26]. 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* – planned to create games that are beyond what is possible within resources and skills of the team, *Feature Creep* – new modules are added without planning during software construction, *Cutting Features During Development* – removing already developed features during the development process, *Problems in the Design Phase* – problems related to design document and game development/technology, *Delays* – delayed schedules, *Technological Problems* – problems with third-party APIs/platforms, *Crunch Time* – accepting unrealistic schedule and long working hours, *Lack of Documentation* – little or no up-to-date documentation, *Communication Problems* – mainly between technical and art teams, *Tool Problems* – lack of effective tools, *Test Problems* – insufficient testing, *Team Building* – caused communication and relationship difficulties, *Number of Defects* – great number of defects found in the development phase, *Loss of Professionals* – loosing key personnel during development, and *Over Budget* – development costs beyond budget [25]. The problems clearly differentiate game development from conventional software development are unrealistic scope, feature creep, lack of documentation, and crunch time. You can also find these problems in traditional software development, but not to such a great extent.

In another study, 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 [27]. 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 [28]. This is related to our study where we investigated how the creative team, the technical team and the management collaborate and coordinate.

2.2 Research Goal, Questions and Method

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) [29]. The metrics used in our study is a mixture of qualitative and quantitative data [30].

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

The purpose of this study was 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 survey and a literature study to back up the findings. The questionnaire consisted of 20 statements where the respondents should state how they agree using the Likert's scale [31]. In addition, we added an opportunity for the respondents to give a free text comment on every statement. The statements in the questionnaire were constructed on the basis of the research goal and the research questions presented above. The subjects of the study were recruited from the Nordic booth at Game Developer Conference (GDC 2012) 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 we had received the questionnaire responses, a survey with eight open-ended questions was constructed to get more detailed answers targeting the respondents from the questionnaire willing to give more detailed answers. The survey was conducted on the web only using surveymonkey.com.

The questionnaire and the survey is described in detail in [32].

3. Results

This section describes the results from the questionnaire and the survey.

3.1 Results from the Questionnaire

This section presents the quantitative results from the questionnaire, comments from the respondents, as well reflections from research literature. We received thirteen responses on this questionnaire from game companies. The distribution of number of employees in the thirteen companies is shown in Figure 1. 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 survey presented in next section can be found in [32].

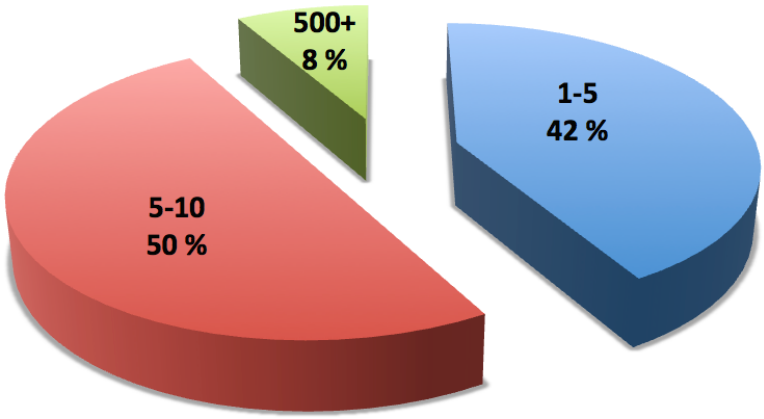

Figure 1. Distribution of Number of Employees

3.1.1 Design of Software Architecture (RQ1)

The statements Q1 to Q6 in the questionnaire were related to the design of software architecture in game development.

Q1 – Design of software architecture is an important part of our game development process:

Agree	Neutral	Disagree	Not Applicable
69%	15%	8%	8%

This statement focused on assessing the importance of software architecture among game developers today. The result shows that most of the game developers in our study considered software architecture to be important. 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, the software architectures of games were simple, and small teams of one to three persons typically developed the games. As game technology, user demands and the game industry have grown, careful design and planning of software architectures have become necessary to manage the complex software and large projects with many involved developers [2]. Although the majority of respondents to our questionnaire 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 [11]. 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) [18]. In addition, for massive multiplayer online games (MMOGs), quality attributes such as security (to avoid cheating) and availability in crucial for success [33]. Careful design and evaluation of a software architecture is the main approach to achieve predictable and acceptable quality attributes in software development [34].

Q2 – The main goal of our software architecture is performance:

Agree	Neutral	Disagree	Not Applicable
54%	15%	23%	8%

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/scriptability, and testability" (Agree)
- "Main goals are:

  1. Performance

  1.5. Memory consumption

  2. 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)

Q3 – Our game concept heavily influences the software architecture:

Agree	Neutral	Disagree	Not Applicable
69%	8%	15%	8%

Almost 3 out of 4 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 the boundary between the game and the game engine. Currently, game engines are targeting 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 [18]. Plummer tries to overcome this problem by proposing a flexible and expandable architecture for video games not specific to a genre [20]. 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 [18].

Q4 – The creative team is included in the design of the software architecture:

Agree	Neutral	Disagree	Not Applicable
69%	15%	8%	8%

The large majority of the respondents agreed that the creative team was 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". 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." 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 [25].

Q5 – Our existing software suite provides features aimed at helping the creative team do their job:

Agree	Neutral	Disagree	Not Applicable
92%	8%	0%	0%

The response concludes that the game engine and the supporting tools provide features that help the creative team. 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 that the creative teams have to use. Ideally, the creative team should only have to work with various GUI editor 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 usually carried out in collaboration with the technical team.

Q6 – Our existing software architecture dictates the future game concepts we can develop:

Agree	Neutral	Disagree	Not Applicable
15%	47%	38%	0%

Game development is all about creativity and coming up with new game concepts. The response from the respondents shows that this for the most true. This statement touches upon the very foundation of how game companies look upon themselves. Are they constrained by existing technology, or can they create whatever the creative team conjures up? The response also shows that almost half of the respondents are neutral to this statement. The only large game company also said to be neutral to this statement and further commented:

- "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)

This comment shows 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. Another comment from a company that was neutral to this statement was:

- "It may influence, but not dictate whenever possible" (Neutral)

A comment from one of the companies that disagreed to the statement was:

- "It makes it a bit more expensive to go to certain genres, but that's it." (Disagree)

These comments 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.

3.1.2 Changes to the Software Architecture during Development (RQ2)

The statements Q7 to Q12 in the questionnaire focused on how game developers cope with changes to the software architecture.

Q7 – The creative team has to adopt their ideas to the existing game engine:

Agree	Neutral	Disagree	Not Applicable
31%	46%	23%	0%

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 limitations 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.

Q8 – During development, the creative team can demand changes to the software architecture:

Agree	Neutral	Disagree	Not Applicable
69%	31%	0%	0%

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

- "Depends how far in development and how big of a changes, 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 from Q8, 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 [35]. 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 [36].

Q9 - 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 the decisions about change. More game developers have also started to adopt agile development practices, where it is more common to have frequent planning and decision meetings where different professions are present [37].

Q10 – The technical team implements all features requested by the creative team:

Agree	Neutral	Disagree	Not Applicable
69%	15%	8%	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)
- "Things just aren't segmented this way in our situation." (Not Applicable)
- "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)

Q11 – It is easy to add new gameplay elements after the core of our game engine has been completed:

Agree	Neutral	Disagree	Not Applicable
70%	15%	0%	15%

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 [38]: 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).

Q12 – During development, the creative team has to use the tools and features already available:

Agree	Neutral	Disagree	Not Applicable
47%	15%	38%	0%

There is not possible to draw a conclusion based on the results from statement Q12. 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.

3.1.3 Supporting the Creative Processes

The statements Q13 to Q16 relates to how creative processes are supported through technology and processes.

Q13 – Our game engine supports dynamic loading of new content:

Agree	Neutral	Disagree	Not Applicable
92%	8%	0%	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 in 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 game play and changes of the game engine itself usually cannot be changed in run-time.

Q14 – Our game engine has a scripting system the creative team can use to try out and implement new ideas:

Agree	Neutral	Disagree	Not Applicable
70%	15%	15%	0%

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 uses with their 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 [39]. Until the game engines can optimize the scripts automatically, the technical team often must assist the creative team with scripting.

Q15 – The creative team is included in our development feedback loop (e.g., scrum meetings):

Agree	Neutral	Disagree	Not Applicable
86%	8%	0%	8%

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. The large game developer in the survey (500+ employees) also said that the creative team was included in development feedback loops. This is in alignment with what has been found in other studies [24, 27, 37]. The only comment related to this statement was:

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

Q16 – Our game engine allows rapid prototyping of new levels, scenarios, and NPC's/behavior:

Agree	Neutral	Disagree	Not Applicable
86%	8%	0%	8%

The statement is related to Q14 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 requires time consuming tweaking tasks" (Agree)

3.1.4 Changes over Time

The statements Q17 to Q20 investigate how game development has changed the last couple of years.

Q17 – Today our company uses more 3rd party modules than 3 years ago:

Agree	Neutral	Disagree	Not Applicable
46%	15%	8%	31%

The response to this statement seamed 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 [40]. The only comment to this statement was "It is about time …" for more usage of third party modules.

Q18 – It is easier to develop games today than it was 5 years ago:

Agree	Neutral	Disagree	Not Applicable
77%	8%	15%	0%

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)

Q19 – Middleware is more important to our company today than 3 years ago:

Agree	Neutral	Disagree	Not Applicable
55%	15%	15%	15%

The majority of respondents agreed that middleware is more important to the company today than 3 years ago.

Q20 – Game development is more like ordinary software development today than 5 years ago:

Agree	Neutral	Disagree	Not Applicable
38%	24%	38%	0%

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 [41]. 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 [7]. 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 [7]. 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 [42]. Usability tests and frameworks are also used within game development, but they are tailored specifically for the game domain [43, 44].

3.2 Results from the 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.

3.2.1 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.

3.2.2 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 [7]. 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 [45].

3.2.3 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.

4. 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" [46]. 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 out to many game developers. We would have liked to have more respondents and also 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 [46]. 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 [46]. 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.

5. Conclusion

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

The *first* research question (RQ1) asked about the role software architecture plays in game development. The response from the game developers was that software architecture is important in game development, and it is important as it helps at managing the complexity of game software as well as achieving the quality in performance and modifiability in games such as performance. We also found 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, an existing software architecture may or may not dictate future game concepts depending on a cost/benefit analysis (reuse of the software architecture if possible).

The *second* research question (RQ2) asked how game developers manage changes to the software architecture. The game developers said that the creative team has to some degree adjust their game play 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 involving personnel from technical team, creative team and management, but the management has the final word. Further, we found that 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 game play elements after the core game engine was complete (although not recommended late in the project). The literature highlighted two approaches to deal with adding game play 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 game play.

The *third* research question asked about how the creative processes are managed and supported in game development. Almost all 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 enabled 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, dynamic and loading of element.

The *forth* research question asked how game development has evolved the last couple of years. This question can be summarized with the following: There has been an increased use of third-party software, middleware has become more important, and it has become technically easier to develop games. Although the majority of respondents said 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 also 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, 2005, pp. 25-32.
2. J. Blow, "Game Development: Harder Than You Think," *Queue*, vol. 1, no. 10, 2004, pp. 28-37; DOI http://doi.acm.org/10.1145/971564.971590.
3. C.E. Crooks, *Awesome 3D Game Development: No Programming Required*, Cengage Learning, 2004.
4. D. Callele, et al., "Emotional Requirements," *IEEE Softw.*, vol. 25, no. 1, 2008, pp. 43-45; DOI http://dx.doi.org/10.1109/MS.2008.5.
5. A. Ampatzoglou and I. Stamelos, "Software engineering research for computer games: A systematic review," *Information and Software Technology*, vol. 52, no. 9, 2010, pp. 888-901.
6. C.M. Kanode and H.M. Haddad, "Software engineering challenges in game development," *Proc. Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, IEEE, 2009, pp. 260-265.

7. D. Callele, et al., "Requirements engineering and the creative process in the video game industry," *Proc. Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, IEEE, 2005, pp. 240-250.
8. L. Bishop, et al., "Designing a PC Game Engine," *IEEE Comput. Graph. Appl.*, vol. 18, no. 1, 1998, pp. 46-53; DOI http://dx.doi.org/10.1109/38.637270.
9. T.C. Cheah and K.-W. Ng, "A practical implementation of a 3D game engine," *Proc. Computer Graphics, Imaging and Vision: New Trends, 2005. International Conference on*, IEEE, 2005, pp. 351-358.
10. R. Darken, et al., "The Delta3D Open Source Game Engine," *IEEE Comput. Graph. Appl.*, vol. 25, no. 3, 2005, pp. 10-12; DOI http://dx.doi.org/10.1109/MCG.2005.67.
11. E. Folmer, "Component Based Game Development–A Solution to Escalating Costs and Expanding Deadlines?," *Component-Based Software Engineering*, Springer, 2007, pp. 66-73.
12. C.A.M. Antonio, et al., "Using a Game Engine for VR Simulations in Evacuation Planning," *IEEE Comput. Graph. Appl.*, vol. 28, no. 3, 2008, pp. 6-12; DOI http://dx.doi.org/10.1109/MCG.2008.61.
13. C. Bouras, et al., "Networking Aspects for Gaming Systems," *Book Networking Aspects for Gaming Systems*, Series Networking Aspects for Gaming Systems, ed., Editor ed.^eds., IEEE Computer Society, 2008, pp.
14. J. Smed, et al., *A review on networking and multiplayer computer games*, Citeseer, 2002.
15. T. Hampel, et al., "A peer-to-peer architecture for massive multiplayer online games," *Book A peer-to-peer architecture for massive multiplayer online games*, Series A peer-to-peer architecture for massive multiplayer online games, ed., Editor ed.^eds., ACM, 2006, pp.
16. T. Triebel, et al., "Peer-to-peer infrastructures for games," *Book Peer-to-peer infrastructures for games*, Series Peer-to-peer infrastructures for games, ed., Editor ed.^eds., ACM, 2008, pp.
17. W. Cai, et al., "A scalable architecture for supporting interactive games on the internet," *Proc. Proceedings of the sixteenth workshop on Parallel and distributed simulation*, IEEE Computer Society, 2002, pp. 60-67.
18. E.F. Anderson, et al., "The case for research in game engine architecture," *Book The case for research in game engine architecture*, Series The case for research in game engine architecture, ed., Editor ed.^eds., ACM, 2008, pp.
19. S. Caltagirone, et al., "Architecture for a massively multiplayer online role playing game engine," *J. Comput. Small Coll.*, vol. 18, no. 2, 2002, pp. 105-116.
20. J. Plummer, "A flexible and expandable architecture for computer games," Arizona State University, 2004.
21. P.V. Gestwicki, "Computer games as motivation for design patterns," *SIGCSE Bull.*, vol. 39, no. 1, 2007, pp. 233-237; DOI http://doi.acm.org/10.1145/1227504.1227391.
22. A. Ampatzoglou and A. Chatzigeorgiou, "Evaluation of object-oriented design patterns in game development," *Information and Software Technology*, vol. 49, no. 5, 2007, pp. 445-454.
23. D. Nguyen and S.B. Wong, "Design patterns for games," *Book Design patterns for games*, Series Design patterns for games, ed., Editor ed.^eds., ACM, 2002, pp.

24. W. Scacchi, "Free and Open Source Development Practices in the Game Community," *IEEE Softw.*, vol. 21, no. 1, 2004, pp. 59-66; DOI http://dx.doi.org/10.1109/MS.2004.1259221.
25. F. Petrillo, et al., "What went wrong? A survey of problems in game development," *Computer Entertainment (CIE)*, vol. 7, no. 1, 2009, pp. 1-22; DOI http://doi.acm.org/10.1145/1486508.1486521.
26. K. Flood, "Game unified process," *GameDev. net*, 2003.
27. F. Petrillo and M. Pimenta, "Is agility out there?: agile practices in game development," *Proc. Proceedings of the 28th ACM International Conference on Design of Communication*, ACM, 2010, pp. 9-15.
28. K. Schwaber and M. Beedle, "gilè Software Development with Scrum," 2002.
29. V.R. Basili, *Software modeling and measurement: the Goal/Question/Metric paradigm*, University of Maryland for Advanced Computer Studies, 1992.
30. C. Wohlin, et al., *Experimentation in software engineering*, Springer, 2012.
31. R. Likert, "A technique for the measurement of attitudes," *Archives of psychology*, 1932.
32. N. Nordmark, *Software Architecture and the Creative Process in Game Development*, Master Thesis, Norwegian University of Science and Technology, 2012.
33. T.-Y. Hsiao and S.-M. Yuan, "Practical Middleware for Massively Multiplayer Online Games," *IEEE Internet Computing*, vol. 9, no. 5, 2005, pp. 47-54; DOI http://dx.doi.org/10.1109/MIC.2005.106.
34. L. Bass, et al., *Software Architecture in Practice*, Addision-Wesley, 2012, p. 624.
35. 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.
36. E. Bethke, *Game Developer's Guide to Design and Production*, Wordware Publishing Inc., 2002.
37. P. Stacey and J. Nandhakumar, "Opening up to agile games development," *Communications of the ACM*, vol. 51, no. 12, 2008, pp. 143-146.
38. 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, 2005, pp. 1-9.
39. W. White, et al., "Better scripts, better games," *Communications of the ACM*, vol. 52, no. 3, 2009, pp. 42-47.
40. A. Rollings and D. Morris, *Game Architecture and Design - A New Edition*, New Riders Publishing, 2004.
41. M. McShaffry, *Game coding complete*, Cengage Learning, 2013.
42. J.L.G. Sánchez, et al., "From usability to playability: Introduction to player-centred video game development process," *Human Centered Design*, Springer, 2009, pp. 65-74.
43. H. Desurvire and C. Wiberg, "Game usability heuristics (PLAY) for evaluating and designing better games: The next iteration," *Online Communities and Social Computing*, Springer, 2009, pp. 557-566.
44. S. Laitinen, "Better games through usability evaluation and testing," *Gamasutra. URL:* http://www*. gamasutra. com/features/20050623/laitinen_01. sht ml*, 2005.

45. J. Hayes, "The code/art divide: How technical artists bridge the gap," *Game Developer Magazine*, vol. 14, no. 7, 2007, pp. 17.
46. W.R. Shadish, et al., *Experimental and quasi-experimental designs for generalized causal inference*, Wadsworth Cengage learning, 2002.