Osborne-O'Hagan, A. and O'Connor, R. V., Towards an Understanding of Game Software Development Processes: A Case Study, In O'Connor, R.V. Akkaya, M., Kemaneci K., Yilmaz, M., Poth, A. and Messnarz R. (Eds), Systems, Software and Services Process Improvement, CCIS 543, Springer-Verlag, 2015.

Towards an Understanding of Game Software Development Processes: A Case Study

Ann Osborne O'Hagan[1], Rory V. O'Connor[2]

[1] Dundalk Institute of Technology, Ireland
ann.osborneohagan@dkit.ie
[2] Dublin City University, Ireland
roconnor@computing.dcu.ie

Abstract. This paper aims to fill the gap that exists about software development processes in game development in the research literature, and address the gap in the research literature by investigating and reporting information about the software development processes used in game development. To investigate the role of the software development process in relation to the game development process, and to better understand the processes and practices used in game software development, a single industrial based case study was undertaken and reported to investigate in a real world context the software development processes and practices used in game development. This research contributes to our knowledge of the field of game development and potentially forms the foundation for further research in the area.

Keywords: Game Development, Software Process, Software Process Improvement (SPI).

1 Introduction

Creating computer games is a complicated task that involves the expertise of many skilled professionals from many disciplines including computer science, art and media design and business. Mcshaffry et al. [1] state that game software development differs from classical software development in many aspects. Games are products that have much more limited life cycle than conventional software products. According to [2] games are usually developed in a shorter timescale and all phases of the life cycle need to be minimised. The main maintenance activity for most computer games is corrective such as bug fixing as the average lifespan is 6 months before a new version of a game is released. As successful games may lead to one or more sequels this could involve some perfective maintenance based on user feedback. The pressure on game development companies to get a product to market as quickly as possible means that there are often schedule over runs and poor time estimation is a problem. For these reasons game project management differs significantly from traditional software project management.

The games development process remains relatively unchanged from inception to consumption despite the fluidity of the industry. The main activities are development/production, publishing / commercialisation, distribution and customer engagement. The publishing role is constantly increasing and changing as the market is getting increasingly more crowded. Traditional distribution is increasingly being bypassed by developers and publishers and there are many intermediates that act as virtual shop windows for online and mobile games. The role of customer engagement is moving beyond that of technical support and involves assisting users with game play and strategy. The Development/production activity is at the core of the game industry, all other activities emanate from this. The game development process will be explored further.

There are various challenges in the game development process. A survey of actual problems in computer games development from analysing post-mortems by [3] affirms that both the traditional software industry and games industry have mainly management problems rather than technology problems, some examples are:

- An important problem specific to the game industry is the communication among the team members. In the electronic games industry, a multidisciplinary team includes people with distinct roles, such as artists, musicians, scriptwriters and software engineers. This mix of roles although being positive in terms of having a more creative work environment, causes a division, dividing it in to "the artists" and "the programmers". This division can be a source for communication problems;
- Within the game development process the game requirements elaboration is complex, subjective elements such as "fun" do not have sufficient/efficient techniques for its determination. It is necessary to extend the traditional techniques of requirement engineering to support the creative process of the electronic game development. A method currently used is to create an early prototype of the game and start people playing it. This helps establish the fun gameplay and once there is a prototype in place there can be an evolutionary approach to development [4]. To develop great games means that you have to design the software to accommodate nearly constant change;
- Transitioning from design to development where there are many defined and undefined requirements can be problematic; it can be hard to project manage unstable and volatile requirements. There can be legacy problems from the preproduction stage. A lot of the game play elements may not have been established and these can cause a much bigger workload in production. It is vital that there is constant user feedback so that the fun elements of the game are developed and that features that are not used or are not delivering user satisfaction are removed or changed.

The subjective nature of game development and the tendency for problems to be related to managerial challenges is making the software development process used in game development worthy of consideration.

1.1 Software Development Processes in Game Development

The over-arching phases of game development according to [5] are preproduction, production, and testing (often referred to in the literature as post production). Preproduction involves the conception of a game, and the construction of a Game Design Document (GDD). By the end of preproduction, the game design document GDD should be finished and will be updated during other phases. In the preproduction phase the game designers and developers do some game prototyping in order to establish the fun or innovative element of a game. This influences the next phase of production as actions in preproduction determine requirements and affect the production phase. Production is where the majority of assets are created, which includes software code. This is a challenging time in the life cycle of the game as a poorly managed production phase results in delays, missed milestones, errors, and defects. In the production phase, the developers can create prototypes, iterations and/or increments of the game. These changes in prototypes or iterations of the game can cause drastic changes to the GDD, with poorly managed changes causing widespread problems affecting functionality, scheduling, resources, and more. The testing phase is usually the last phase and involves stressing the game under play conditions. The testers, not only look for defects, but push the game to the limits for example the number of players could be set to the maximum and can be labelled stress testing (or load testing). These phases are more complicated than the overview given.

Current game development literature suggests that the traditional software development model, exemplified by the Waterfall Model that requires explicit requirement assessment followed by orderly and precise problem solving procedures is inadequate for the innovative and creative process of the videogame industry [6]. Agile development methodologies are less focused on documenting the pre-production phase and more focused on quickly getting a workable version of the game, by using iterative design and dynamic problem solving techniques that are facilitated through frequent and co located meetings. This goes some way towards easing the transition from preproduction to production. Shell [7] describes an iterative process that he calls looping, which essentially consists of an iterative cycle of design and test. Agile development methods are increasingly becoming the industry norm and according to [21] more agile practices should be incorporated into game development.

From an examination of the literature, most of the works relating to game software development focus on the design phase and design challenges, and on the problems associated with transitioning from preproduction to production. This is reiterated in [7] who state that the game development process literature mostly has design and design problems as a primary concern, as opposed to production and the issues that relate to production. A case study on a game development company reports on the organisational enablers for agile adoption [8]. Successful agile adoption requires project stakeholders to have common project objectives, employees having the ability to make decisions at relevant levels of abstraction, effective project management and a supportive learning environment.

The focus of this research is on the software development processes in game development and as such it would be beneficial to explore the SDLC used in the development phase of the game development process. The SDLC does not include all elements needed to create a game; it basically describes the steps and iterations needed to develop software. Overall there is a lack of published studies relating to the software development processes/methodologies used in game development and this gives supporting evidence for the proposition about the lack of research in the literature on the software development processes/methodologies used in game development.

In this research it was found that there is a lack of categorisation in the literature relating to game development processes and to lay a foundation it would be helpful to categorise and systematically analyse the literature in relation to game development processes in a scientific way. It is proposed that a Systematic Literature Review (SLR) would be a suitable method to do this and would also help establish a gap in the literature relating to the use of agile methods in the game development process.

A Systematic Literature Review of the software processes used in game development was conducted [9] where a total of 404 papers were analyzed as part of the review and the various process models that are used in industry and academia/research are presented. Software Process Improvement initiatives for game development are dis cussed. The factors that promote or deter the adoption of process models, and implementing SPI in practice are highlighted. Our findings indicate that there is no single model that serves as a best practice process model for game development and it is a matter of deciding which model is best suited for a particular game. Agile models such as Scrum and XP are suited to the knowledge intensive domain of game development where innovation and speed to market are vital. Hybrid approaches such as reuse can also be suitable for game development where the risk of the upfront investment in terms of time and cost is mitigated with a game that has stable requirements and a longer lifespan.

Given the above a set of four research questions were formulated as follows:

- What software processes are game development companies using and how are these software processes established?
- What phases/steps are involved in the software processes used in game development?
- How do the software processes, that game development companies are using, change and what causes these software processes to change?
- How do the operational and contextual factors, present in game development companies, influence the content of software processes?

2 Case Study Research Approach

It is proposed to conduct a single industrial case study using grounded theory data coding methods [10] to help with data analysis to develop theory about game development processes and to capture the best practices of a game development company.

An interview guide was developed as an instrument to help guide the interviewer in gathering specific data during an interview session, and to help the researcher collect data in a consistent and predefined manner. The interview guide included closed and open-ended questions and some related notes about ranges and samples of possible information appropriate to the research. Closed questions looked for specific information and open ended questions allowed scope for the participant to add contextual information that may be of importance to the research. The sample responses were included to help guide the interviewer and act as examples should they be required, these examples also helped keep the interviewer on the right track due to the fact that some of the terminology could have more than one meaning and therefore there could be misinterpretation.

The data analysis methods based on grounded theory coding were selected for use in this study as they were deemed to be more robust and traceable than qualitative data analysis and more explicit and systematic than content analysis. Coding can be described as the key process in grounded theory [11] and the three coding techniques proposed by Grounded Theory methodology are: open coding; axial coding; and selective coding[12]. The 4 main stages used in applying the grounded theory method that helped with data analysis are described:

- **Open Coding** - This involved identifying categories, properties, and dimensions.
- **Axial Coding** - This involved examining conditions, strategies, and consequences.
- **Selective Coding** - This involved coding around an emerging storyline.
- **The Conditional Matrix** – This involved reporting the resulting framework as a narrative framework or as a set of propositions.

The researcher investigated various tools which are used for data management in qualitative research and selected Atlas Ti [13] a tool designed specifically for using with grounded theory. This tool enabled the researcher to: store and keep track of interview scripts; to code and, manage codes and related memos; to generate families of related codes; and to create graphical representation for codes, concepts and categories. Atlas TI provided support for axial and selective coding used in this study. Overall Atlas TI supported data storage, analysis and reporting.

2.1 Case Study Company

The game development company was chosen based on the fact that it was in close proximity to the researcher and is representative of a typical case for game software development in a small start-up company which is representative of many indigenous Irish game companies. The researcher proactively studied the game development project during the production/implementation phase of development. The case study research was initiated in June 2014 and was executed over a three month period.

The game development company was founded in September 2012 with the goal of developing a Massively Multi-Player Online (MMO) game for the seven to twelve year old demographic with an educational aspect. The company is developing games for the mobile and online platforms. The company can be described as a VSE with an official employee count of 5. At the time of the case study the company had no games published and can be described as a start-up company. That company had one game development project in the production phase that had commenced in September 2013

2.2 Data Collection

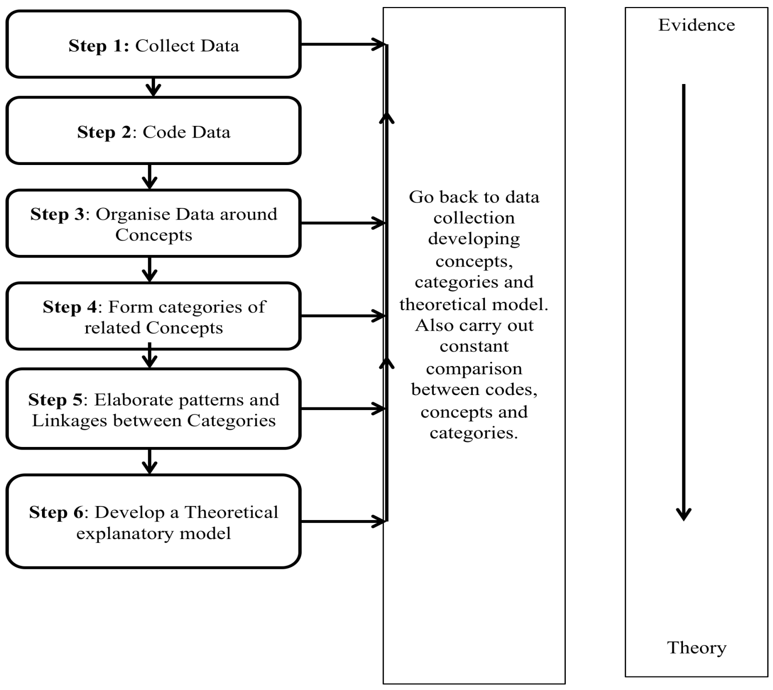
Data collection involved the use of semi-structured interviews. To support the semi-structured interviewing process the researcher developed an interview guide with a formal question set. The interview guide contained 24 questions and these were divided into the following 7 sections: (i) General company and job description (ii) Process Establishment [14] (iii) How the process works [15] (iv) Software Process Improvement [16] (v) Project Success factors [17] (vi) Operational and contextual Factors [18] that affect the process [19] (vii) Ending. The use of an interview guided allowed the researcher collect data in a consistent and unbiased fashion. The questions were based on the researcher's prior knowledge and were proof read by someone external to the case study that had an expertise in software process. When conducting interviews it is desirable to have different viewpoints that can be analysed and compared and that are complimentary to each other. The researcher proposed to have two viewpoints: managers; and developers. Therefore interviews were conducted on company employees from both management and development roles in the game development company and as such allowed a complimentary analysis based on both viewpoints. The subject sampling strategy was to interview all employees currently employed at the game development company. The animator who was a member of the software development team was not available at the time of the study. Each participant was given an information sheet describing the research project and was asked to sign a consent form regarding the recording of the interview. All participants attended the interview voluntarily and data collected was treated in strict confidence. The three interviewees were from various roles within the game development company: (i) The general manager, (ii) a member of the development team and (iii) a senior technical member of the company.

The interviews varied between 20 minutes and 60 minute duration. The reason for the variation in the interview times was that the CEO who was the first interview only wanted to partake in the first section of the interview. This interviewee maintained that she had no knowledge of process and did not want to answer the remaining questions. The CEO had valuable information pertaining to the general company and was the main access to the company for carrying out further interviews. Some notes were taken by the researcher during the interviews. All interviews were audio recorded for later transcription and analysis. A session summary sheet was completed after each interview. This described who was involved, the issues covered, their relevance to the research questions, and any implications for subsequent data collection.

Data collection and analysis were conducted concurrently. Each interview was transcribed by the researcher. The transcribed files and any additional collected data were stored in the qualitative analysis tool Atlas Ti.

2.3 Data Analysis

The grounded theory coding analysis method was used to inductively generate theory about game development processes. The researcher used Glaser's[20] non-linear method of theory generation as guidance for the data analysis as illustrated in figure 1.



Step 1: Collect Data
Step 2: Code Data
Step 3: Organise Data around Concepts
Step 4: Form categories of related Concepts
Step 5: Elaborate patterns and Linkages between Categories
Step 6: Develop a Theoretical explanatory model
Go back to data collection developing concepts, categories and theoretical model. Also carry out constant comparison between codes, concepts and categories.
Evidence
Theory

Fig 1. Grounded Theory Data Analysis Steps

All interviews were recorded and transcribed, and the analysis was conducted with rigour using open coding, axial coding and selective coding techniques. The open coding technique: involved the following 2 steps: Step 1- The researcher assigned codes to various quotes in the transcript to classify or categorise it. A code can represent a certain theme. One code can be assigned to many quotes, and onequote can be assigned to more than one code. Codes can contain sub-codes. There was an initial code approach using gerunds and in vitro coding approaches. Each statement of interest in the transcribed material was coded and the next step (step 2) involved sorting the codes into categories based on how the codes are related and linked. The emerging categories were used to organise and group the initial codes into meaningful clusters. This involved breaking the interview data into discrete parts based on similarities and differences; the researcher went through the material to identify any phrases that are similar in different parts of the material, patterns in the data or variances of any kind. These code categories were then used in subsequent data collection. The open codes that were conceptually similar were grouped into more abstract categories based on their ability to explain the sub units of analysis.

The researchers analysed the three interviews and during this iterative process a small set of generalizations were formulated. Diagrams in the form of flow charts were produced to help focus what was emerging from the data and network charts of codes were generated to help link concepts to categories. The transcripts were re-read and re-coded in a different order to see if any new themes emerged and when no new themes emerged this suggested that the major themes had been identified. From the data collected, the key points were marked with a series of 220 codes, which were extracted from the text. The codes were grouped into similar concepts in order to make the data more workable, this grouping was facilitated using Atlas Ti. From these concepts, 25 categories were formed, which were the basis for the creation of a theory.

3 Case Study Findings

The theory is based on two conceptual themes, *Process of Game Development* and Game Software Development Process, and four core theoretical categories, *Project Management*, *Contextual Factors*, *Operational Factors* and *End Product*. The axial coding role identified the categories into which the discovered codes and concepts could be placed and selective coding was used to explain the relationships between the categories to provide the overall theoretical picture. The objective of selective coding was to identify a key category or theme that could be used as the fulcrum of the study results. In this research, the analysis showed that there was one central category to support and link the two theoretical themes. The final list of themes, the core categories and the main categories identified by the study are shown in Table 1. Each category and code can be linked to quotations within the interviews and these are used to provide support and rich explanation for the results. The saturated categories and the various relationships were then combined to form the theoretical framework.

Table 1. Themes, Core Category and Main Categories

Theme	**Main Categories**
Process of Game Development	Company Profile Market Sector Business Drivers Company Formation
Game Software Development Process	As-is Process Drivers for Change Process improvement Process problems Process Strength Ideal process
Project management	Planning / Prioritise Tools
Contextual Factors	Team Size/Experience/Motivation

	Subjectivity Technology/Resources
Operational Factors	Up-capturing the intention Create the Right Working Environment Injection of Confidence Adequate Resources Capacity to Get a Good Review Vendor Requirements
End Product	Re-Use End-User Schedule Revenue

3.1 The Theoretical Framework

The emergent grounded theory was summarised in terms of themes, core categories and main categories. This summary is shown as a network diagram in Figure 2 which identifies the relationships between the major themes, core category, linked categories, and associated attributes. Within the theoretical framework, each node is linked by a precedence operator with the node attached to the arrowhead denoting the successor. All of the relational types within the framework are precedence and the network is read from left to right.

The root node of the framework, *Process of Game Development*, is a conceptual theme and is a predecessor of its four categories, *Company Profile, Market Sector*, *Company Formation* and *Business Drivers*.

The *Business Drivers* and the *Role and Experience of Employees* contribute to the *Process Origin* used as the basis for the company's software development activity and the *Process Model in use*. The *Role and Experience of the Employees* coupled with the *Background of Founder* of the company creates an associated *Management Style* and this, in conjunction with the adapted process model, creates the company's initial *As-is Software Development Process*.

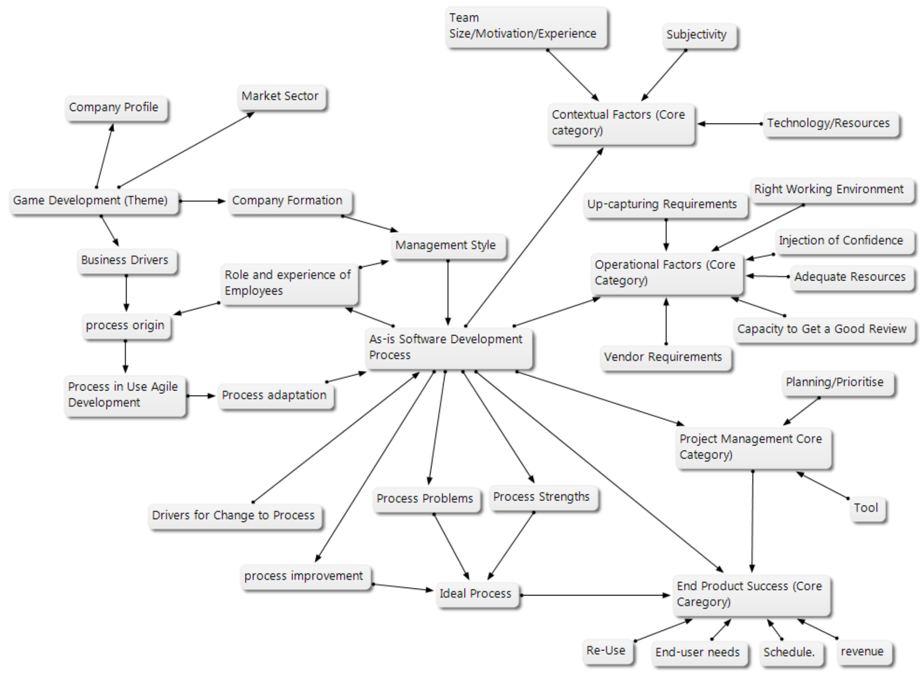
The *Game Software Development Process* can be described as follows. The *Drivers for Change* to the *As-is Software Development Process* can *lead to Process Improvement*. *Process Improvement* along with *Process Problems* and *Process Strengths* can contribute to creating an *Ideal Process*. The *As-is Software Development Process* is influenced by *Project Management*, *Contextual Factors*, *Operational Factors* and *End Product* requirements.

End Product is affected by an *Ideal Process* and *Project Management. End Product* can itself then impact the organisation's ability to *Reuse*, meet *End Users* needs, provide *Revenue* and the ability to deliver a product on *Schedule*.

Project Management impacts the organisations ability *for Planning/Prioritise* what gets done and *Tool* usage.

Contextual Factors affecting the as-is game software development process include *Team Size/Motivation/Experience*, the *Subjective* nature of games and is impacted by the *Technology/Resources* available.

Operational Factors affecting the as-is game software development process include '*Up Capturing*' the *Intention Correctly*, *Creating the Right Working Environment*, having an *Injection of Confidence*, having *Adequate Resources*, having the *Capacity to get a Review* and meeting *Vendor Requirements*.



Team Size/Motivation/Experience
Subjectivity
Company Profile
Market Sector
Contextual Factors (Core category)
Technology/Resources
Game Development (Theme)
Company Formation
Up-capturing Requirements
Right Working Environment
Injection of Confidence
Management Style
Business Drivers
Role and experience of Employees
Operational Factors (Core Category)
Adequate Resources
process origin
As-is Software Development Process
Capacity to Get a Good Review
Vendor Requirements
Process in Use Agile Development
Process adaptation
Planning/Prioritise
Project Management Core Category)
Drivers for Change to Process
Process Problems
Process Strengths
Tool
process improvement
Ideal Process
End Product Success (Core Caregory)
Re-Use
End-user needs
Schedule.
revenue

Fig 2. Theoretical Framework

In creating the theoretical framework, several of the Atlas TI features were utilised. The Code Family option allows codes, created from both the open and axial coding phases to be grouped together under a family heading, for example, *End Product*. This facility allowed the various interviews to be searched for passages where references to codes, which were classified as members of the *End Product* family, had been raised by the practitioners. Another feature of Atlas TI that was used in developing the framework was the Code Frequency Table. This option shows how often codes occurred within a particular interview, and across the entire suite of interviews, thus providing support for developing the more widespread categories. In addition to employing the code family and frequency table aids, Atlas's query tool also provided major assistance with data analysis. The query tool contains Boolean and proximity operators which test for the co-occurrence of codes in the data. For example, a Boolean query can search for occurrences of Code A and/or Code B, whilst proximity can test the distance between, or closeness of, code occurrences in the text. An example of a proximity query included examining the distance between developer references to end user and a subsequent reference to a code in the *End Product* category.

4 Discussion

The focus of this research was on the software development processes in game development. Based on the proposed gap in the literature identified in this paper, the aims of this research was to explore the gap that exists about software development processes in game development in the research literature, and address the gap in the research literature by investigating and reporting information about the software development processes used in game development; and to Investigate the role of the software development process in relation to the game development process, and to better understand the processes and practices used in game software development. A set of four research questions were formulated. These research questions are revisited below and an analysis of the findings is reported.

Research Question 1 relates to identifying the software processes used by the game development company and finding out how the software processes was established. The software process in use is agile development using the Scrum methodology. The process in use has been established from the previous work experience of the CEO, CTO and the Developer. The previous experience of the CEO in managing previous companies led to a management style (umbrella) that in conjunction with the previous software development experience of the CTO and the Developer in Agile Development using the Scrum methodology contributed to process establishment. In this instance the CEO had little technology experience and was relying on the software development team for expertise in process for game software development. This could indicate that it is not a pre requisite for the CEO of a game development company to have technical expertise.

Research Question 2 relates to identifying the phases/steps that are involved in the software processes used in game development. It is interesting to note that the CTO describes all phases/steps of the game development process. He does not see a distinction between the game development process and the software processes used in game software development. The CTO has more experience of game development and is more aware of all that is involved in a full game development process. The developer is only aware of the current and preceding phase of game development. By contrast to the CTO perspective the game software development process as described by the developer is a subset of the game development process and is described as a design/development phase. The developer is describing the as-is software development process that is the design/development phase. There is a difference in the process described by both the CTO and Developer and the reason why is because the company is a start up game development company and the process is not fully enacted or established. It could be that the ideal game software development process involves a hybrid of process described by both the CTO and Developer and could consist of a design, develop and test steps within a development phase.

Research Question 3 relates to how the software process that the game development company is using change and to identify what causes the software process to change. There were variations between the CTO and the Developer as to how the software process changed. The CTO was aware of changes to do with tools such as the software repository tool: The software repository system was left at times because it was unreliable and was done in an alternative fashion. The Developer was aware of changes to some of the steps in the software development process such as: the Sprint cycle time was reduced from 2 weeks to 1 week. The Developer is best positioned to describe the actual software development process because he is the one doing the development work. The CTO carries overall responsibility for security in terms of version control, backup and security codes. Some of the above changes to the process caused an improvement to the process. An example of this was introducing the tool Illustrator to the process. This helped speed up the process. Any tool that helps speed up the process in terms of creating graphics is very worthwhile in game software development. The process in game software development is inextricably linked to satisfying the needs of end-users.

Research Question 4 relates to how the Contextual and Operational factors, present in game development companies influence the content of software processes. Contextual Factors cited by both the CTO and the Developer related to team attributes and resources. While these are common to both traditional and game software development there are variations in emphasis. The subjective nature of the game software development process alluded to by the CTO is critical in game software development. The following contextual factors can influence the game software development process: The team size affects the volume of work than can get done. A small, co-located team allows for a fluid process where creativity can flow and eliminates the need for a change management process. The small team size means that the workers need to be flexible and may need to share roles and tasks. The game is a moving target at all times and can require that the workers are highly enthusiastic and well motivated, also there needs to be a clear vision about the goal being undertaken. Some of the roles within the team are part time which means that workers must have the discipline and motivation to work on their own without too much overseeing. Often there is a lack of experience which means there is a very high learning curve. There can be a lot of experimentation needed and ideally the majority of this will have been worked out prior to the development phase. Games are played for pleasure, emotional challenge and not for functional reasons. The appeal of a game is an emotional contract formulated between the designer, developer and the end-user. The best way to counterbalance the deeply subjective nature of games is to engage with end-users as much as possible during the software development process. It is possible here to see what appeals to the end-user and cut out the functions that are not used or not appealing to the end-users.

4.1 Future Work

This research potentially forms the foundation for further research and as a follow on to the research the researcher would like to outline three areas with potential for future research: Firstly a multiple case study to investigate game software development processes would be of benefit. An advantage of a multiple case study would be its increased scope for replication and generalisation. This research made certain propositions and a multiple case study could build on these propositions and makes the results more generalisable.

Second, this research showed various gaps in research relating specifically to the game software development process. There is no ‘best practice model for game software development’. A best practice model for game software development could be beneficial for the games industry and as such could reduce development time which could reduce time to market; it could also help improve the quality of game software. This is a gap here for this research to be done. Such a best practice model could be based on existing standards, such as ISO/IEC 29110 [22] if the were accepted [23] by organizations.

Finally there is no easy method to capture the likes and dislikes of computer game end-users. A tool that could easily capture these requirements of these end-users could greatly improve the game software development process. It was shown during this study that the interaction with the end-user is of paramount importance, but a tool to do this could effectively save time and money in terms of creating art assets.

References

1. McShaffery, M. 2005. Game Coding Complete, Paraglyph Press.
2. Ampatzoglou, A. & Stamelos, I. 2010. Software engineering research for computer games: A systematic review. Information and Software Technology, 52, 888-901.
3. Petrillo, B., Pimenta, M., Trindade, F. & Dietrich, C. 2008. Houston, we have a problem: a survey of actual problems in computer games development. Proceedings of the 2008 ACM symposium on Applied computing. Fortaleza, Ceara, Brazil: ACM.
4. Shull, F. 2011. Managing Montezuma: Handling All the Usual Challenges of Software Development, and Making It Fun: An Interview with Ed Beach. Software, IEEE, 28, 4-7.
5. Kanode, C. M. & Haddad, H. M. Software Engineering Challenges in Game Development. Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on, 27-29 April 2009 2009. 260-265.
6. Winget, M. A. & Sampson, W. W. 2011. Game development documentation and institutional collection development policy. Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries. Ottawa, Ontario, Canada: ACM.
7. Schell, J. 2008. The art of game design – A book of lenses, Burlington, Morgan Kaufman Publishers.
8. Srinivasan, J. & Lundqvist, K. 2009. Organizational Enablers for Agile Adoption: Learning from GameDevCo. In: Abrahamsson, P., Marchesi, M. & Maurer, F. (eds.) Agile Processes in Software Engineering and Extreme Programming. Springer Berlin Heidelberg.
9. Osborne O’Hagan, A., Coleman, G., O’Connor, R.V.: Software development processes for games: a systematic literature review. In: Barafort, B., O’Connor, R.V., Poth, A., Messnarz, R. (eds.) EuroSPI 2014. CCIS, vol. 425, pp. 182–193. Springer, Heidelberg (2014)
10. O'Connor, R., Using grounded theory coding mechanisms to analyze case study and focus group data in the context of software process research, in Mora, M., Gelman, O., Steenkamp, A. and Raisinghani M. (Eds), Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems, Chapter 13, IGI Global, pp. 1627-1645, 2012.
11. Strauss, A., & Corbin, J. 1990. Basics of qualitative research: Grounded theory procedures and techniques, Newbury Park, CA, Sage.

12. Coleman G. and O'Connor R., Using grounded theory to understand software process improvement: A study of Irish software product companies, Journal of Information And Software Technology, Volume 49, Issue 6, Pages 531-694, June 2007
13. Coleman, G.,O'Connor, R.: Investigating software process in practice: A grounded theory perspective. Journal of Systems and Software 81, 772–784 (2008)
14. Coleman G. and O'Connor R., An Investigation into Software Development Process Formation in Software Start-ups, Journal of Enterprise Information Management, Vol. 21, No. 6, 2008, pp.633-648
15. O'Connor, Rory V., and Gerry Coleman. "An investigation of barriers to the adoption of software process best practice models." ACIS 2007 Proceedings (2007): 35.
16. Clarke, Paul, and Rory V. O'Connor. "An empirical examination of the extent of software process improvement in software SMEs." Journal of Software: Evolution and Process 25.9 (2013): 981-998.
17. Clarke, P., O'Connor, R.V.: Business success in software SMEs: recommendations for future SPI studies. In: Winkler, D., O'Connor, R.V., Messnarz, R. (eds.) EuroSPI 2012. CCIS, vol. 301, pp. 1–12. Springer, Heidelberg (2012)
18. Clarke, P., O'Connor, R.V.: The situational factors that affect the software development process: Towards a comprehensive reference framework. Journal of Information and Software Technology 54, 433–447 (2012)
19. Jeners, S., Clarke, P., O'Connor, R.V., Buglione, L., Lepmets, M.: Harmonizing software development processes with software development settings – a systematic approach. In: McCaffery, F., O'Connor, R.V., Messnarz, R. (eds.) EuroSPI 2013. CCIS, vol. 364, pp. 167–178. Springer, Heidelberg (2013)
20. Glaser, B. G. 1978. Theoretical Sensitivity: Advances in the Methodology of Grounded Theory, Mill Valley, CA., Sociology Press.
21. Petrillo, F. & Pimenta, M. 2010. Is agility out there?: agile practices in game development. Proceedings of the 28th ACM International Conference on Design of Communication. Brazil: ACM.
22. Mora, M., Gelman, O., O'Connor, R., Alvarez, F., & Macias-Luevano, J. (2009). An overview of models and standards of processes in the SE, SwE, and IS Disciplines. In A. Cater-Steel (Ed.), Information technology governance and service management: Frameworks and adaptations (pp. 371–387). Hershey, PA: IGI Global
23. Sanchez-Gordon, M.L., O'Connor R.V. and Colomo-Palacios, R., Evaluating VSEs Viewpoint and Sentiment Towards the ISO/IEC 29110 Standard: A Two Country Grounded Theory Study, In Rout, O'Connor, R.V. and Dorling. (Eds), Software Process Improvement and Capability Determination, CCIS 526, Springer-Verlag, 2015.