2015-05-12

Improving configuration management, quality management and development methods in the computer game industry

Authors:
Richard Hultgren, Peter Kullgard

Advisor:
Conny Johansson

Blekinge Institute of Technology

Pt96rhu@student.hk-r.se
Pt96pku@student.hk-r.se

Abstract: There is a lack of maturity in the software game development that results in development companies using their own invented, but not fully completed, procedures. Through a case study this master thesis gives an overview of the game developers maturity in configuration management, quality assurance and development methods in general. It also presents a suggestion on how to address the issues found in the case study.

Keywords: Configuration management, quality, quality management, quality assurance, quality system, testing, development method, game development

This thesis is submitted to the Department of Software Engineering and Computer Science at the Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering.

Contact information:

Authors:
Richard Hultgren
Linnégatan 65 B
SE-21615 MALMÖ
Sweden
E-Mail: richie@pc.nu

Peter Kullgard
Trädgårdsvägen 73
SE-19146 SOLLENTUNA
Sweden
E-Mail: peterk@paradoxplaza.com

University advisor:
Conny Johansson
Department of Software Engineering and Computer Science
Blekinge Institute of Technology
SE-37225 RONNEBY
Sweden
E-Mail: Conny.Johansson@bth.se

1.1 Preface

When we decided to choose the subject of game development for our thesis we knew it was going to be a tough case. We had suspicions that it was going to be very difficult to get in touch with developers that would agree to be interviewed during our research phase. So it was! It is hard enough to get developers to answer questions if they aren't already acquainted with you, but even if they are most developers are very busy. Being in the same industry we sure can't blame them. Instead we would really like to thank all developers and companies that have participated in this thesis. Without you guys this Master Thesis would never been possible to complete!

A special thank you goes to our supervisor Conny Johansson who has not only guided us through the master thesis but also our entire software engineering education as supervisor for the program.

Ronneby June 2001

Peter Kullgard, Richard Hultgren

1.2 About the authors

As we have referred to our experience in this document we thought it would be a good thing to give you, the reader, a very brief presentation of ourselves.

We, as many others in the profession, both started out in a very young age playing around with computers giving us an informal education that we believe in many cases are underestimated. Before going to the university we have both worked at computer companies dealing with software as well as hardware. During our 4 years of education we have in project organisations worked for companies as Symbian, Ericsson and Europolitan, which roughly have given us one year of practical experience from the industry.

During our time as students we have in parallel continued to work within the computer industry. In our fourth year as master students we started to work part-time for a game company called Massive Entertainment where we now, when writing this, work full-time. Both of us were hired to the QA department, Richard as tester and Peter as Lead tester. After a few months Peter became the project manager of the game Ground Control and became a full-time employee. At this moment Richard is working as software engineer and Peter recently worked as studio manager at Massive Entertainment. These practical years within the industry, combined with the theory learned in our studies, has given us a coarse-grained but highly interesting insight into the software industry. This is what we will refer to as our experience.

One final note though. The interest for computers and software in special was many times renewed or reinforced by excellent computer games. We have never stopped playing games and will most likely never stop and therefore consider ourselves as being true gamers.

1.3 Audience

This document's intended audience are primarily software engineers or other people in the computer industry with the basic understanding of software engineering. Even so this document is written in such a way that we do believe that it can be fully comprehended by most readers.

1.4 Table of Contents

Abbreviations

CM	Configuration management or configuration manager.
QM	Quality management or quality manager.
SQA	Software quality assurance.
QA	Quality assurance.
SCM	Software configuration management.
PM	Project management or project manager.
OO	Object oriented.

1.5 Definitions

Build	The merge and source code of all the latest available sources. It includes the compilation of all the available data needed for the final product.
Beta	A widely used state of development in the game industry. It is an important state when a product gets feature complete.
Method	A method is a planned procedure by which a specified goal is approached step by step. A method is often described in an abstract manner; it includes a number of steps that should be performed during a development. All steps describe how the work is to be carried out assuming a certain underlying basic architecture [ANDERSSON].
Process	A process is the natural scaling-up of a method. The difference between method and process descriptions is that a method often is described in a waterfall model (i.e. step-wise). In another way it is similar to saying that a method is a description of a system's first product version development. The process, on the other hand, will continue to exist as long as the developed system is in operation [ANDERSSON].
Development phase (phase)	[BENNATAN] states “Management of software development within an engineering discipline is based on a much more orderly set of development phases”. He explains that phases are a part of the development cycle. Each phase is associated with a certain stage of the development and contains activities and processes that need to be carried out at that stage.

2 Introduction

2.1 Origin of this thesis

Like many thousands other people around the world we started our computer careers in a very young age with just one item that could fully absorb our attention: a computer with computer game. Since then much has changed, we have changed our focuses several times, from games to programming, from programming to software management, from management to quality, and finally we have come to the point where all this is combined into a science to pursue, namely software engineering. But in our hearts the joy of playing computer games never faded. In our last year, the master of software engineering year, we both decided to work part time at a game development company.

Seeing problems in the industry this master thesis came naturally. Secondly, the game industry is a very exciting and creative trade [MCCUSKEY1] that in a certain way has only recently been recognized as a serious way of making a living. Therefore there is much to be done and we would like to make a small contribution to a line of art that has given us so much.

2.2 Hypothesis

- There is a lack of maturity in the game industry that results in companies using their own invented, but not fully working, game development procedures.

2.3 The purpose with this thesis

There are three main purposes with this thesis:

1. To verify our hypothesis.
2. To get an overview of the game development today.
3. To suggest improvements for game development.

The first two goals are met by doing a case study, an analysis of the case study and research. Using the result from these two goals we will create our own suggestion for how game development could be improved.

2.4 Problem statements

We want the following problem statements answered:

- How does the general game development configuration management look like?
- What are the most common problems and solutions in software configuration management?

- How does the general game development quality assurance look like?
- What are the most common problems and solutions in software quality assurance?
- What are the most commonly experienced problems during the different development phases and how have these problems been solved, how should they have been solved?
- What are the most common problems and solutions in their development methods?
- How can game development be improved?

2.5 Scope

As this thesis is limited by its size in number of hours (400h + 400h) we had to constrain the size of the thesis. We have chosen to focus on configuration management, quality assurance and development phases. It is for these areas the improvement recommendations have been made. We have not made any cost estimations for applying the recommendations or any estimations for the actual cost benefits when these recommendations are in use.

The thesis is mainly focused on large game products focusing only on entertainment. A large game product is according to our definition a product that takes approximately ten man-years to develop. A man-year is about 1800 hours of work in Sweden.

2.6 Overview of the thesis

The main work in this thesis is the case study and the analysis. To reach this goal we had to go through some initial steps that will be presented in chapter 1, 2 & 3. Using the knowledge obtained from the analysis we created a suggestion for how it could be possible to improve game development. We have logically divided the document into the following chapters:

1 Introduction

In chapter 1 we will, besides presenting the origin, hypothesis, purpose, problem statements, and scope of this master thesis, answer why we believe there is a need for a separate case study in the game industry. At the end of chapter 1 we will also present the important definitions made for use in this document.

2 Work procedures

How the work in the master thesis was done is at least as important as the result itself. Chapter 2 describes our work procedures into details. It presents the procedures background and how we did when we put them into practice.

3 Pre-study

Chapter 3 presents the information we found in the pre-study phase. Its focus is on what have been done in the general software-engineering field but it also presents specific game development information as well as our experience from the game industry.

4 Interviews

Undoubtedly the most important chapter in our master thesis, chapter 4 presents the result of the case study and the analysis that followed it, which is the core of our master thesis.

5 Putting the interviews into use

This chapter takes a closer look at the findings made in the interviews. It presents, where possible, a reason, a solution and, a comment for each finding in an attempt to improve the game development of today.

6 Future work

Chapter 6 discusses some improvements and expansions that can be made to this thesis.

7 References

The references section is divided into books, articles and Internet addresses. It contains a lot of interesting material that is only touched by this thesis so for the readers interested in game development this is a good resource.

8 Appendix I - Interview questions

In this appendix the interview questions used in this thesis can be found. No answers to the questions are presented because by reading these answers it could be possible to trace down the individuals or the companies. This is not acceptable since we have promised not to give out any names.

2.7 Why separate suggestions for the game industry?

We believe that before starting working on a thesis one of the most important questions to investigate is whatever there is a need for the thesis or not. We found the following reasons for doing separate suggestions of improvements for the game industry:

- Experience-fun. There is a slight difference between games and other mass-market products that must be understood. Games are all about entertainment [Section 2.5 Scope] while products in most other parts of the software industry fills a function and produces a result. It may be to provide service, present information, control items, or in other words, easy up a task. The core reason for creating a game is to give the user as much entertainment as possible. That is why a good game is at least fun [GDEVINT].

- Creativity. One very important factor in game development seems to be creativity. According to [CATA] “game development is creative work” and that “workers need their freedom”. By creating an environment with too many administrative tasks you can kill the vision [GAMVIS].

- Re-use is often limited. The rapid development in computer multimedia hardware forces regularly updates in the software [HOWLAND]. A larger part of the games are usually re-written or changed for each new product released [SCHAEFER].

- Communication. Successful communication is essential in every software project [NICHOLAS]. In game development there is often a combination of musicians, graphic artists, software engineers, and managers involved [GRUBER]. One should suspect that at least one combination of these groups could have problems communicating [ARTNPRO]. With such many different backgrounds it can easily become a communication nightmare [GORDON]. Therefore communication must be enforced in game development.

- Education. According to [MCCONNELL] a lot of the people at the game companies does not have the right or any formal education. In order to make sure the project does not end up in chaos this must be considered too.

2.8 Definitions

In our years as software engineering students we have clearly seen the need for definitions or at least good explanations. We had two goals in mind when we created our own definitions. First of all they must cover our entire understanding of the area being defined and secondly they should be as understandable and short as possible. Considering the conflict in these goals we now understand how hard it is to make good definitions.

In the areas configuration management [WWW-ISO2] and quality assurance [WWW-ISO1] there are existing standards defining these areas. However, these standards are made with such a high level of generalization that they do not entirely fit our purpose due to the fact they are too abstract. Therefore we would like to define these areas ourselves before we proceed. To be able to do so we have looked at standard definitions as well as the non-standard, which is often found in literatures.

2.8.1 Not fully working development method

Not fully working is by its nature hard to define since it is a *relative* word. The way we have addressed it is by saying that we believe that there are such major problems originating from the quality management, configuration management, and development methods that we must consider them not fully working.

2.8.2 Quality as a general expression in the software industry

Quality is a widely used expression in the software industry. According to [SANCURRAN] *"The word 'quality' means different things to different people"*. Therefore we feel it is important to us to give the reader our definition of the expression "quality".

The British Standards Institution (1986) has stated that *"Quality is in the eye of the beholder, a matter of the client's judgement"* [TAKUBB] and we partly agree. What we would like to do is to extend this view and include all of the three parties usually involved in national and international game products [STARTUP], which is the end customer who actually buy the product of the shelf, the publisher who market, often test, and deliver the product to the shelf owner, and the developer who develops, maintain, and ships the product to the publisher.

Using these three parties and saying that quality is in the eye of the beholder we modify the British Standards Institution statement and say:

Quality is in the eye of the beholder, a matter of the end customer, publisher and, developer's judgement.

2.8.3 Software Quality Management and Software Quality Assurance

We have made the decision to split quality management and quality assurance in our definition. The reason for this is that in the game industry quality management and quality assurance are generally defined as the practical procedures of game testing [GS01]. We will define quality assurance as a part of quality management but because of its great importance in game development [SCHAEFER] we will also make a separate definition of quality assurance. To define quality management we must first give our opinion and explanation of what we consider is quality.

2.8.4 Software Quality Management

According to [SEERS] Software Quality Management *"...is concerned with the concepts, methods, techniques, procedures and standards for producing high-quality software products in an efficient and cost-effective manner."*

[NICHOLAS] states that:

"The quality assurance supervisor establishes and administers inspection procedures to assure fulfilment of all quality related requirements. Overall, his responsibility is to raise awareness of quality and to institute means for improving work methods and producing zero defects."

Apparently, Software Quality Management is not just about Quality Assurance. Quality Assurance is just one part of it. To be able to deliver a product within budget on time that satisfies the customers, publisher and the own company, more actions need to be considered then just testing. If the developer want to be able to reuse parts of the source code for the next project actions has to be taken during development to assure the usability. These actions could be reviewing, implementing code standards, or the use of other engineering principles [BENNATAN]. If the customers are not satisfied with the features of stability of the game the developer needs to be able to make fast and low-budget changes into the system. If the developer is to deliver the product to the publisher in time the work has to be structured. There has to be a development plan that will meet the deadlines.

Considering these aspects we now define Software Quality Management as:

Software Quality Management is the process of ensuring that the product is delivered successfully considering all Quality aspects. The Quality aspects include procedures, methods, standards, people and all related actions that make a contribution of increasing the Quality of the project.

2.8.5 Software Quality Assurance

Our next definition is quality assurance, which we have defined to be a part of quality management.

[BEIZER] says that the aim of software quality assurance is bug prevention and that the primary tool is testing. [SANCURRAN] states that SQA is about ensuring that project standards and procedures are adequate to provide the required degree of quality, and that they adhered to throughout the project. [PRICE] supports the idea of error prevention by looking at the procedures by saying:

"Quality-thinking is to control the product by controlling the process and thereby making the delivery inspection completely unnecessary. Quality becomes an invisible effort "

In CMM [PAULK95] we find SQA as a key process area for Level 2: Repeatable. CMM explains the purpose of SQA as:

"The purpose of Software Quality Assurance is to provide management with appropriate visibility into the process being used by the software project and of the products being built."

If we collect the strong points from these statements we find that SQA at least involves:

- Bug prevention by testing
- Adequate and adhered standards and procedures
- Providing management with process visibility

These are the points by which we build our definition:

Software quality assurance is the managed on-going procedure where document and software errors are prevented, detected and removed according to given procedures, which constantly are improved.

2.8.6 Software Configuration Management

In [BENNATAN] an explanation are made rather than a definition. He explains CM as follows:

"Configuration management is the technical term and administrative application of configuration control. This also includes the maintenance of a configuration control organization, change and version control standards, and configuration control facilities."

Which calls for his explanation of configuration control:

"Configuration control is the process of evaluating, approving or disapproving, and managing changes to configuration items. Configuration control also often includes version control functions."

He continues in his way of giving explanations rather than definitions. From the foreword of the IEEE standard 828 (1978b) he presents their explanation of SCM:

"Software configuration management (SCM) if a formal engineering discipline which provides software developers and users with the methods and tools to identify the software developed, establish baselines, control changes to these baselines, record and track status, and audit the product.

SCM is the means though which the integrity and continuity of the software product are recorded, communicated and controlled."

[NICHOLAS] states:

"The project engineer also oversees configuration management – the purpose is to have uniform communication media for all areas and activities for identifying, documenting, and controlling information. Configuration management is used in projects where it is necessary to review frequent changes during system design and development, and to control and document theses changes in ways that account for their impacts on related component and the overall system."

[GILB] gives a rather limited explanation of SCM:

"Software configuration management is a design and maintenance process, modelled strongly on corresponding hardware disciplines. These are primarily from the aerospace and military fields. The basic idea is to keep track of the set of design and implementation components, which make up any one complex system. The purpose is to be able to analyse faults and the effects of changes on particular unique configurations, which might be in the hands of customers, by knowing exactly what their configuration is composed of."

[TAKUBB] is kind enough to give us a definition of SCM:

"Configuration management: Management of the development and evolution of a system by identifying its exact state and composition, its configuration, at discrete points in time, in order to control change and ensure trace ability of system evolution.

Software configuration management: Configuration management related to specifically to software systems."

In The Capability Maturity Model for Software framework we find SCM in key process area for level 2: repeatable [PAULK95]. The four goals are: 1) SCM activities are planned, 2) selected work products are identified, controlled, and available, 3) changes to identified software work procedures

are controlled, and 4) affected groups and individuals are informed of the status and content of software baselines.

When we looked for strong points in these six definitions/explanations we identified the following items of importance for the game industry:

1. Evaluating, approving/disapproving changes.
2. Version control.
3. Maintenance of standards.
4. Methods.
5. Tools.
6. Baselines.
7. Record and track status.
8. Audit.
9. Uniform Communication media.
10. Documentation of changes.
11. Impact estimation.
12. Analyse faults.
13. Tracking configurations.

We are aware that the items in the list are not fully defined but nevertheless an overview of what is considered to be software configuration management can be obtained. It therefore helps us in our definition. Our definition of software configuration management is:

Software configuration management is the on-going process of QM where all information and tools needed to perform a project within an organization is identified, documented, uniformly communicated, analysed, and audited according to a given standard. The information is tracked for version control, fault analyses, and change impact estimation.

2.8.7 Development method

When referring to a development method one usually think of a method within a paradigm. According to [CHAMP] such a method would consists of:

- A notation with associated semantics.
- A procedure/pseudo-algorithm/recipe for applying the notation.
- A criterion for measuring progress and deciding to terminate.

We partly agree to this definition. Our belief is that a development method at least consists of a criterion for measuring progress and deciding to terminate. Without this there is no development method. A notation and a procedure for applying it are recommended but not necessary. Nevertheless

as our thesis focuses on making a suggestion on how to improve game development without being dependant on a certain paradigm the first two bullets are not really of further interest to us. Notations and their procedures will not be covered by this thesis.

On the other hand we have included QM and CM in our thesis since we strongly believe that a development method without these methods is inadequate for the game industry. It is important to understand that we still accept a method where these methods are excluded.

Our definition of a development method is:

A development method is the complete collection of methods that is used to carry out the project. A complete collection is any collection that at least contains a main procedure with well-defined phases, a criterion for measuring progress and deciding to terminate exists.

Again, some of the processes included in this definition could be the processes for quality and configuration management, as defined earlier, and some other general processes that we believe belong to project management in the game industry. These are the ones that will be presented in the thesis. In addition to these there are most likely (or should be) other processes that make up a complete development process collection.

2.8.8 Summary

We have made the following definitions for use in this document:

Software quality management

Software quality management is the managing function that is responsible for the SQA activities, which are executed according to the SQA plan.

Software Quality Management is the process of ensuring that the product is delivered successfully considering all Quality aspects. The Quality aspects include procedures, methods, standards, people and all related actions that make a contribution to increasing the Quality of the project.

Software quality management

Software quality assurance is the on-going procedure where document and software errors are prevented, detected and removed according to given procedures, which constantly are improved.

Software configuration management

Software configuration management is the managed on-going process where all information and tools needed to perform a project within an organization is identified, evaluated, documented, communicated, analysed, and audited according to a given standard. The information is tracked for version control, fault analyses, and change impact estimation.

Development method

A development method is the complete collection of methods that is used to carry out the project. A complete collection is any collection that at least contains a main procedure with well-defined phases, a criterion for measuring progress and deciding to terminate exists.

3 Work procedures

3.1 Overall work procedure

There is a lot to say about two or more people working together with a master thesis. On the upside there is the possibility to get ones work continuously reviewed and discussed and thereby hopefully improving the quality of the work. On the downside there is the risk of getting ones work continuously reviewed and discussed and thereby getting nothing accomplished. There was one thing we both soon agreed on; the best way to get a result that we both would be satisfied with was to work separately with the same task as often as possible and then comparing, discussing, and compiling the two results. We made this our overall working procedure.

3.2 Pre-study

We had 2 sequential goals in mind when we started our pre-study. The first one was to expand our knowledge of how the software industry in general solve the problems we have addressed (see problem statements) in our theses. This was done by studying books and articles written in the subject as well as re-collecting information from some of the companies we have come in contact with in various ways during our years as software engineering students.

The second goal repeated this assignment but focused exclusively on the game industry. In this case we found a small amount of relevant information when searching in written books and articles. Most of the information we have found was located at web sites or collected from other game companies which we have come in contact with while working at one ourselves. We did not expect to find much information about game development in the industry but at least we found one book dedicated to game development without it focusing too much on the technical issues.

3.3 Main study

Our goal in the main study was to perform a case study in the game industry itself. Interviews with 10 former or current employees at game companies located in Scandinavian and USA where conducted in order to reach this goal. In this chapter you'll find more detailed information about how we worked towards this goal.

3.4 Developing the interview questions

3.4.1 Qualitative study

We made the choice to make a qualitative study rather than a quantitative study. There are two main reasons for this. First of all, we primarily wanted to perform oral interviews so we could make sure that the questions was correctly understood and if something interesting revealed itself we could simply ask supplementary questions. The idea was to obtain a deeper and different knowledge than the often-fragmented knowledge obtained when conducting quantitative methods [RUPABODA]. We also didn't believe there was any way to interview a large amount of companies within our time-budget.

Secondly, we suspected that most of the companies we planned to contact were all very busy and did not have much interest in filling out a written mass-survey. Our thought was that if we personally contacted them showing special interest in their company we would get a larger amount of replies. We reasoned a qualitative study with fewer but interested companies would be for the better.

Another reason for selecting the qualitative study was that it's practical to make analyses along with its progress while quantitative methods normally wait until all material has been collected [RUPABODA]. This suited our purpose well.

3.4.2 Who has been interviewed?

It is important for the reader to be aware of that we interviewed one employee at each company. In one case we interviewed a person that was not in the game industry anymore but had a lot of experience after several years in the game industry (although he has been employed at a new game company after the interview was preformed). However, we decided to let him represent a separate company and be a part of the research.

3.4.3 Engineering the questions

The questions was engineered using the following sources:

- Books written about the topics.
- Searching, finding and, tailoring questions used in other theses.
- By asking other people in the game domain what they believe is important to be answered within our thesis.
- Using our experience from the game domain and as software engineers.

Using our problem statements as guidelines for what should be asked we tried to keep the questions and answers within the focus of our thesis. We also kept in mind that in some situations people need warm-up questions to get going while others don't. In the first situation mentioned we simply asked more supplementary questions and accepted that in some cases they ended up outside our focus.

3.4.4 Selecting the interview objects

It was not a surprise that the game companies we contacted were all very busy and as a result they had little time to spare for our interview. This basically meant that we had to deviate from our original highly optimistic plan, which was to interview one programmer, one graphic artist and one manager at each company. This would have given us a good overview of the company and we could then have crosschecked the information for interesting observations. Sadly, considering our time budget, it was not economically viable to wait for the opportunity to get an interview from three roles at each company.

Our next thought was that we should to try to get hold of as many managers as possible. There were, however, two problems with this. Our intention with this thesis was to get an overview of the entire game development and its problem at all levels. If we had only interviewed managers there would have been a considerable risk that we would have got no more than a top-down view of the problems in the game industry. Secondly, the managers seemed to be the persons most occupied and reluctant to accept the interview.

At the end there was not much else to do than to ask for the person most suitable, and available, to answer our questions. Of course this was a problem since we then faced the risk to leave interviews

with unanswered questions, as the person interviewed perhaps did not have the knowledge to answer the question. This is the explanation to why we in some cases do not have full coverage of our questions.

3.5 Performing the interviews

All oral interviews were conducted with a voice-recorder activated. The information taped was then transferred to text. After that the text was re-read a couple of times while we searched for patterns by using the other interviews as a base. This was repeated for each interview along with the progress and in the end we made a larger final analysis using our notes from the previous smaller analyses. See "Compiling and analysing the interview results" for more information about what we considered a pattern and how the analysis was made.

After we had contacted the companies on our list, E-mails containing the questions were sent out to the ones that were interested in participating in our study. All the answers from the questions therefore reflect the employee's situation at his current company. The questions asked during the oral interview were exactly the same questions asked using email. During the design of the questions we made sure that the questions were very clear and short just to make sure the questions would fit email interviews too.

3.5.1 Reliability of data

The risk of getting the data twisted or forgotten decreased with the use of a tape-recorder. We did consider letting the persons interviewed confirm our documented interviews but decided not to as they already had been kind enough to give us the time for the interview. Another risk was simply the possibility of translation errors (i.e. languages tend to have different levels of expressions tied to their words and this could cause interpretation problems). Of course, keeping this in mind, we have made our best to avoid translation errors by first translating the same information individually and then by comparing the two results.

3.5.2 Considering experience

It is also important to know that the data will most likely provide different results depending on how experienced the companies are that has been a part of the interviews. Therefore we decided to provide the reader with information about the companies' previous experience. In the chart below we are providing the reader with information about how many years the company has been in the industry and how many products the company has released. The number of products produced and released can of course differ because a lot of times the publisher cuts projects before they have been released (even if the products are completed). It can be hard to find out the number of products produced; therefore we just included the number of products released. As company IX is the person we let represent a company we will not give out any information about his experience since we have not asked for his approval of this.

3.5.3 Representation

As mentioned before it was very hard to find companies that wanted and could take part of the interviews. After asking about 100 companies it they wanted to participate we ended up with ten interviewed companies. It is important for the reader to know that we can not guarantee that these ten companies represent all the companies asked, due to the fact we could not choose among all the interviewing objects because they were too few. As found in the table we have interviewed companies with 0-40 released products, 5-57 employees, and 2-15 years in the industry. Since our analysis has

shown no difference between these companies regarding experience our guess is that we do have obtained a fairly representative set of companies.

Table 2.1 Experiences

Company	Number of released products	First year in industry	Number of employees
Company I	1	1998	5
Company II	More than 10	1993	N/A
Company III	0	1998	About 10
Company IV	1	1997	About 40
Company V	13	1993	57
Company VI	0	1999	6
Company VII	40	1986	20
Company VIII	3	1996	30
Company IX	N/A	N/A	N/A
Company X	0	1998	N/A

3.6 Compiling and analysing the interview results

3.6.1 Compilation

Before the final analysis could be made we had to compile all the answers that, after the transfer from tape to paper, where in written form. We compiled the result into three different 'views' of the data. The first one was a complete table (yes, it became very large and cumbersome) of all the answers given by the companies containing reduced answers. This gave us a good overview of the overall answers. The second view was simply a stack of paper with the complete answer from each company without any reduction made. This view was mainly used to confirm any findings found in the overview table but the opposite was also true, if there was any assumed finding in this stack it could be confirmed by the overview table. The third and last view contained complete company information so that the findings could be examined further.

3.6.2 Analysis of data

The main purpose with the analysis was to get an overview of how the game industry in general deals with the areas that we have included in our thesis. Using this overview we intended to answer our problem statements and whatever our hypothesis has been proven correct or not. It should be pointed out that in a qualitative study there is no definite method of how to conduct the work with the information obtained [RUPABODA] and we will therefore in this section describe our method.

3.6.2.1 Patterns, findings and conclusions

Before the analysis started we decided to use three terms namely pattern, finding and conclusion. A pattern is any comparable answer that repeats itself in two or more answers from the different

companies answering the same question. A special case of pattern is when all answers obtained from the same question are different. Defining comparable is of course hard. In some cases it is apparent that there is a pattern while in some cases it requires an interpretation of the answers. Luckily these occasions were rare and in any case both the authors had to agree to the existence of a pattern before it was accepted. Any found pattern was noted as a finding.

A finding is a candidate to become the data from which one draws a conclusion. The information used to draw any conclusion is based only on the information gathered from the interviews. Note that a finding does not have to be a pattern although it often is. We decided that a finding is any interesting observation found in the data obtained from the interviews. An example of an interesting observation could be that one company has a specialized unique solution to a certain common problem, or it could be that the most experienced company still don't use a plan for their development while other less experienced do. As with the patterns the findings are not always apparent and in many cases it lead to long discussions between the authors before it was accepted or rejected.

3.6.2.2 Analysis method

We started off the analysis by separately reading through the interviews several times thereby getting familiar with the data collected. Already at this stage we made some notes or markings directly on the paper when we found an interesting answer. As we made interviews using mail (5 companies) our next job was to make sure that all the answers came from the same interpretation of the question. For each question we filtered out those answers that had a different interpretation than the majority. However, we still kept the filtered answers since they might contain interesting information. We then sorted the answers to make sure the all answers were tied to the right question. This was needed since we had obtained answers to other questions in some of the questions.

Having done this our next step was to compare the answers from all companies to see whatever there was a pattern or not. This was primarily done using the overview table. When a pattern was found in the overview table it was categorized thus giving it a logical name and description. Those special cases where no pattern at all was found were of course also noted.

After going thru the overview table all findings were checked using the stack view in which the complete information from each company is stored. This was done to make sure that we have not misinterpreted the reduced overview table when looking for patterns.

The next step was to check the current findings against the companies' experiences [Table 2.1 Experiences] to see if any other findings related to experience could be obtained. We made this by sorting the companies in falling experience order then checking whatever any special pattern could be seen.

Having done that we then discussed the two results with each other to see whatever we both could agree on the findings or not. In those cases we could not agree whatever there actually was a finding or not the person in favour of the finding had to explain each step in the analysis trying to convince the other of the findings existence. If, after this stage, not both of us would agree on whatever there actually was a finding or not, then the finding was marked as a possible finding but it was not included in the final result from the analysis.

At this point we had a bunch of sorted and approved findings in each category (CM, QA and development phases). We now decided not to draw any conclusions from findings based on patterns with less than a total of 4 answers.

Now that the analysis had been made we started to look for ways of presenting the by us approved findings and conclusions with a minimum risk of getting it misinterpreted. The result of the analysis is presented in Presentation of results, which obviously contains the presentation method we selected.

3.7 Improvement design and requirements

3.7.1 Requirements

When deciding how to present the information we prioritised readability. We set up the following rules for readability:

1. All information should be presented with a short structured text.
    a. The number of text lines for each finding label should at a maximum be 8 lines.
    b. Structure should be upheld by maintaining the same structure as the presentation of the findings.
2. It should be possible to clearly see where the information came from.

Rule number 2 was made to make a distinction between knowledge obtained from the interviews, literature or our own experience. As we have promised to not reveal any companies or persons names we also made sure that the information presented couldn't be used to figure out any names.

3.7.2 Design

The design of the improvements section was made when we had completed both the analysis and the presentation. The thought, however, of having an improvements section in our master thesis had been with us from the beginning of the thesis but since we did not know how the case study would turn out there was at that stage no way to make a design for it.

We started out the design by asking ourselves whatever this section would add any value to the thesis or not since the major goal was to perform a case study. One of the reasons found for including this section was that a lot of the problems found in the game development industry are they same as the ones found in the regular software development, which we have encountered in our studies. Hence there are a lot of solutions and recommendations published about these problems that we felt should be presented. Seen from the game developers point a view it would certainly be interesting to get to know some of the published material written about their problems. Another reason for an improvement section is that in some of the cases the solutions found in the game industry itself really deserves to be presented. Finally there are findings that we ourselves would like to comment.

A simple table was constructed to present the information in the improvements section. We made a table with 2 columns and four rows. As a standard praxis the left column contains the labels and the right one contains the information. The following labels where added: the name of the finding, a reason, a solution and a comment. The name label is a single number that is the same as the number given in the findings presentation. A short description about the finding is also given in order to make each improvement suggestion standalone from the findings presentation. The reason is collected from the interviews alone. In most cases a reason is only found when the finding is a problem. The solution is composed both from the interviews and other sources except ourselves. By solutions found in the interviews we mean working solutions used by the participants. When other sources than the interview are used as a solution a reference is always given making it possible to distinct the solution's origin. We ourselves assembled the comments, which really is any relevant comment that we believe should be mentioned. The comment is free to include both reasons and solutions for the finding.

The final design is visualized in the table below.

Finding 1	*Description of the finding.*

Reason	*Reason found in the interviews*
Solution	*Solutions found in the interviews or other sources.*
Comment	*Own free comments, could include reasons and solutions.*

3.8 Summary

We had two goals with the pre-study. The first one was to expand our knowledge of how other parts of the software industry solve the problems we have addressed in this master thesis. The second goal was to gain the same knowledge from the game industry. The goal of the main study was to perform a case study of the game industry. During the case study 10 game companies were interviewed. One employee at each game company answered the questions. The employee was not required to hold a special position at the game company as long as the employee could make a contribution to our work.

The questions used in the case study were engineered using books, other thesis, information from people in the game industry and our own experience. The game companies answered the questions orally or via email.

To be able to perform the analysis of the data we had to compile it and make the data comparable. We achieved this by creating three different views of the data. Using the different views we tried to answer our problem statements and tried to find out if our hypothesis was correct or not. During the analysis we used the terms pattern, finding and conclusion. To be able to use the terms in a correct way we have stated our definitions of the term pattern and finding.

A pattern is any comparable answer that repeats itself in two or more answers given by the game companies for the same question. A special case of pattern is when all answers obtained from the same question are different. A finding is a candidate to become data from which we draw a conclusion. All data is based on the interviews only.

The analysis method used contained three different steps. During the first step the interviews and the answers were checked and sorted. The second step was to make a three-stage compilation of the interviews. This was done using the overview table, stack table and experience table. The third step was to compare the results of obtained by both of us. We had to make sure we agreed an all patterns, findings and conclusions.

To present the improvements suggested we designed a simple table. Our focus was the readability. The table contains the Finding, Reason, Solution and Comment.

4 Pre-study

In this section we will present some of the result from the pre-study. As stated in the work procedure the goals were to get an understanding of how the software industry and game industry in general handles configuration management, quality management, and development methods. In addition to this we will also briefly describe how we have learned to perform in this areas in our studies as well as working outside of the game industry.

4.1 Software quality management

When we choose the subject of our master thesis one of the cornerstones was QA. According to our experience we thought QA in the game industry were different to QA in other areas of the software industry. To be able to have a serious discussion about this we wanted to gain as much information as possible about QA so we read about nine books and a few articles on the subject.

4.2 Definition of QA

After examining the material found during the pre-study we found out that none of our sources had the exactly same definition of QA. This is a very interesting fact and concluded by [BENNATAN] that states “As there is no single widely accepted definition for quality, and because different people perceive quality in different ways, both the developer and the customer must reach agreement on metrics for quality. The method of measuring quality may differ for different projects”.

4.2.1 Process

One very important part of QA is using your experience and knowledge to take the QA process to a new level at each new project. If you improve the QA process itself you will most likely improve the products passing through it. [SANCURRAN] states: “The only way to improve quality reliably is to improve the software process.” [PRICE] agrees with this but add the fact that each activity concerning the product should be improved.

4.2.2 Management

About every book we have read has discussed the management part as an important cornerstone of QA. Even though the authors of the books have presented management differently, one book summarised the management principles in a very good way that almost covered all our sources.

[SANCURRAN] states that “Effective management... is based on the following principles.” The following four points are the principles he mentions:

- Clearly define the structure, roles, responsibilities and lines of communication between groups and individuals.

- Carefully plan all activities needed to complete the overall task within time, budget, resources and quality constraints.

- Continually monitor progress against plans, and revise plans appropriately when tolerances have been exceeded.

- Refine the detail of plans as knowledge of task increases.

In addition to these four principles we would like to add one that we believe is of great importance, namely education. [PRICE] supports this and mentions in his book that it is important to carry out training and education. He also mentions that if you want it to be successful you must also train and educate the management. We both believe this is an important part of QA management so we looked for further support for it and found it in both [SANCURRAN] and [GILB].

4.2.3 Activities

According to our own experience and education it seemed like the domain of QA in the game industry is far to narrow. Therefore we put some extra effort on finding out what activities are included in QA in other parts of the software industry. Of course this issue is very close related to the definition of QA. However, even if a definition covers a domain it does not specify the activities needed to fulfil the requirements of the defined domain.

[SANCURRAN] has included the following activities as parts of QA to fulfil his definition of the QA domain.

- Management

- Documentation

- Standards and practices

- Reviews and audits

- Testing

- Problem reporting and corrective action

- Tools, techniques, and methods

- Code and media control

- Supplier control

One interesting activity here is testing. According to both [MYERS] and [BENNATAN] this is an activity that should not be handled by the people making the software. [MYERS] states: “A programming organization should not test its own programs.” If testing is made by the same organization that produced the software the objectivity will be lost which will result in a less reliable product. We agree to their statements but we still believe in tests performed by the organization, as a way of increasing the product quality when considering that these tests does not have to be the final ones before shipment.

4.2.4 QA by experience

After been working mainly with QA of the game Ground Control, both of us have gained a lot of experience with QA in the game industry. What concerned us when we started at Massive working with Sierra Studios on Ground Control was that the QA domain was very limited. We think that QA in the game industry is defined only as testing (including parts as balancing etc.) and delivery process. However, the interesting part here is that the process and all definitions of testing and delivery are well defined. It seems to us that there is definitely a pattern how developers and publishers interact with each other on this issue.

We must say though that the conclusion of our own thoughts is that the entire QA domain of the game industry is very immature. The reason is that the QA domain is too narrow and most of the issues regarding activities and managements are missing.

4.2.5 QA in the game industry

We have read a lot of articles and web pages but information about QA in the game industry is hard to find. However, we found some very interesting statements.

When looking for information about testing in the game industry the word beta testing often comes up. For example both [GDC01] and [HOWLAND] talks in their articles about beta testing. According to [GDC01] the game moves to the final revision when the beta testing is done. Since it obviously is of great importance another interesting question is; who is supposed to test the game? [MCCUSKEY1] states that "try to avoid letting the developers play-test. The people making the game are not the best ones play-testing it." But according to [HOWLAND] both internal and external test teams are used in game industry. He states the purpose of the external test team is to "try to find bugs that the internal team overlooked or didn't consider problems." This is interesting since it matches our opinion of how testing should be done.

We have almost not found any information about what is influencing the quality of a game (except for the testing procedure). Only one article has discussed this issue. [MCCUSKEY1] states that "two things influence the quality of your project – the quality of your design document, coupled with how solid and consistent your team's work habits are." An important statement and hopefully our interviews will reveal more information of this kind.

The last interesting point we found was stated by [WIERZBICKI], as "Production quality doesn't necessarily mean the best in the industry; more often it means done" Such a statement inevitably makes us wonder which quality goals are set in the game industry.

4.3 Configuration management

Configuration management has always been present in the projects we have participated in but for some reason the projects have always failed to identify exactly which tasks that belongs to CM. Still we have not yet found anyone who can argue against the importance of CM in a software project and therefore we wanted to take a closer look at it in our thesis. This section describes what we found in our pre-study regarding CM.

4.3.1 Cornerstones of CM

We have selected three descriptions of CM that we believe capture the essence of what we have found about CM in our pre-study. The first description is by [BENNATAN] and goes as follows:

"Any configuration control method must be based on the following four concepts:

- A clearly defined configuration management authority must be established.
- Configuration control standards, procedures and guidelines must be produced and distributed to the developers.
- Configuration control cannot be effective without the necessary tools and facilities.
- A configuration management plan must be developed at the beginning of the project."

To us, being software-engineering students, this seems like a natural list but it still takes discipline to make sure that everybody in the organization understands the importance and carries out these fundamentals found in the list.

4.3.2 Activities in CM

In [PAULK95] activities are used instead of concepts. These activities are:

- A SCM plan is prepared for each software project according to a documented procedure.
- A documented and approved SCM plan is used as the basis for performing the SCM activities.
- A configuration management library system is established as a repository for the software baselines.
- The software work products to be placed under configuration management are identified.
- Change request and problem reports for all configuration items/units are initiated, recorded, reviewed, approved, and tracked according to a documented procedure.
- Changes to baselines are controlled according to a documented procedure.
- Products from the software baseline library are created and their release is controlled according to a documented procedure.
- The status of configuration items/units is recorded according to a documented procedure.
- Standard reports documenting the SCM activities and the content of the software baselines are developed and made available to affected groups and individuals.
- Software baseline audits are conducted according to a documented procedure.

This is a pretty long listing that probably will be even longer when broken down and adapted to an organization. Nevertheless, we believe using activities listed for different tasks is the best way to avoid important steps being forgotten or neglected.

4.3.3 Reproducibility

Our last description of CM comes from [TAKUBB]. He starts of by saying: "A major aim in configuration management and change control is reproducibility. We may wish to reproduce an older version of the system if a new one has serious shortcomings or we may wish to reproduce specific functionality in a particular environment. In fact, the need may arise to reconstitute both the product and the process at any stage in its evolution.

In pursuit of this aim, three objectives of configuration management are control, consistency and minimizing cost."

Looking at [TAKUBB]'s statement we strongly suspect that reproducibility is an important part of product maintenance for any mass-market product that ships to users with unknown configurations, which is often the case with games. When we worked at the QA department at Massive we both found it to be vital to be able to reproduce an error in order to find a way to fix it.

4.3.4 Selection

Rounding up our presentation of the main results found about CM in the industry, from [TAKUBB] we quote a couple of words that we believe is very important not to forget:

"Configuration management is more than just a set of methodologies and tools. An organization will select configuration management methodologists appropriate to its specific needs but considerable skill is needed in both selection of the right techniques and methodologies and the effective imposition of them upon the software development and maintenance process."

4.3.5 CM by experience

In our experience CM is still mainly focused on tools and file management. Generally there is no documented CM plan that is distributed and followed by the developers. A CM is usually appointed officially or unofficially (usually the one most familiar with both hardware and software used in the project) and in both cases he or she often becomes the "handy-man" in the project that not only has to fulfil the full-time role as software engineer but also as a CM.

Foremost the tasks include dealing with hardware, administrative software, software configuration and backup, source control and merging, network configuration, and last but no least trying to figure out the developer's problems found outside of the source code or resources.

4.3.6 CM in the game industry

We have found very little information about CM in the game industry. The information we did find was mainly about different file control tools and scripting tools and in no case we saw any definition or description of CM. At that stage we thought there could be several reasons for this. One reason could be, which would strengthen our thesis, that there is almost none CM used in the game industry at all, hence the lack of information. But for all we knew at that moment, it could simply be that there is no interest in this area or that there are no problems at all worth attention. A case study then seemed the only way to obtain the actual situation.

4.4 Development methods

Below we have made a short presentation of some of the development methods we have encountered in our studies [DEVREP99]. They also represent the few mentioned development methods found in our game development pre-study phase and we therefore decided to include them in this section. When we refer to widely known development methods these are the ones we mean.

4.4.1 Rational Unified Process

Rational is described as an iterative, architecture-centric, use-case driven, and risk driven process. The process is organized around four phases; inception, elaboration, construction and transition, and further organized around five workflows: requirements capture, analysis, design, implementation and

test. It is a process described as an industry model, which in turn is structured in terms of workers, activities and artefacts. Well-defined milestones, which are used both for progress checking and as a checkpoint for evaluating and decision-making, conclude all phases.

4.4.2 Extreme Programming

Extreme programming was made with the goal to combine advantages from other development methods into one functional development method. The method uses evolutionary deliveries, testing by programmers as well as customers, continuous integrating and testing, pair programming, a search for simplicity in design and code, daily design, and refining and defining the architecture.

4.4.3 Daily Build

The most fundamental aspect of Daily Build is the fact that the product being developed is compiled, linked and automatically tested every day. This puts a high demand on the use of a powerful configuration management tool and the writing of test code and test scripts parallel to the regular product development. With the development of Windows 2000 and previous products, Microsoft have proven that the method is applicable on a product with over 30 million lines of code.

4.4.4 Cleanroom

Cleanroom is a method that focuses on defect prevention with the goal to reach zero defects. The system specification is written with a formal technique and the designers have to use a structured programming theory to ease the verification process. All possible execution sequences have to be defined and the probability for each has to be estimated. This way the errors with the highest frequency can be found easier. The developers do not perform unit test, instead the software is tested all at once by an independent test organization. Management is responsible for that all employees know the benefits from implementing Cleanroom.

4.5 Summary

The conclusion of the examination of the material found on the subject QA is that none of our sources outside the game industry uses the exactly same definition. One of our sources states “As there is no single widely accepted definition for quality, and because different people perceive quality in different ways, both the developer and the customer must reach agreement on metrics for quality. The method of measuring quality may differ for different projects.” One very important part of QA is to improve the process itself. We have found out that the only way to improve quality reliably is to improve the software process.

During our pre-study we also found out that management is a cornerstone of QA. Effective management is based on a few principles”. The principles we found almost summarized all our sources but we added on to make the list of principles more complete. We did also put some extra effort on finding important activities for QA. One very interesting activity is testing. Two of our sources pointed out that a software organization should not test their own programs.

According to our own experience QA in the game industry is defined only as testing, but there is definitely a pattern how developers and publishers interact with each other on this issue.We hardly found anything about QA in the game industry. However, we found some interesting statements like “Production quality doesn’t necessarily mean the best in the industry; more often it means done.”

We have selected three descriptions of CM that we believe capture the essence of what we have found about CM in our pre-study, namely concepts, activities and reproducibility. According to our

experience CM is still mainly focused on tools and file management. A CM is usually appointed officially or unofficially and often becomes the “handy-man” in the project.

During our studies we have encountered information about the following development methods: Rational Unified Process, Extreme programming, Daily build and Cleanroom.

5 Interviews

5.1 Background and purpose

Working in a profession one often gets a natural, and useful, interest in other companies problems and solutions that are in the same type of industry. As this work is public by default we though it would be great to collect this kind of information and share it with all the game companies out there and hopefully thereby contribute to the overall game development.

The second reason is simply that we needed the information from the interviews to be able to do our master thesis. Without the connection to the industry the thesis would be seriously limited to our knowledge alone. Don't get it wrong, this could be interesting too (oh well…) but there would be no way to make a good industry-attached improvement suggestion section out of it.

5.2 Limitations

There are some limitations to the interviews that we have found. They are more or less all based on time problems. If we, and the companies that participated in the study, had had more time the limitations would probably not exist. Anyway, we have found the following limitations:

- Limited amount of persons interviewed.

  We only got to interview one person from each company. This is by far not a complete view of the company but due to lack of availability within our time budget there was not much else to do. A full coverage would probably reveal new information not found in our interviews.

- No crosschecking.

  This limitation is tied to the above. Since only one role was interviewed we might have gotten the one and only person within a certain role that felt the opposite compared to the majority.

- Limited interview time.

  The interview time was limited. In some cases we had to move on despite the fact that we suspected that we could stumble into something interesting if we just had taken some extra time to complete the discussion. But having promised a certain amount of interview time it would have been very impolite not to move on.

- Lack of a complete follow-up.

  This is primarily a limitation to the mail interviews. In three cases, where questions were not answered or misunderstood, we did not receive an answer for our request for new answers.

- No verifications

If there had been enough time we would have requested two verifications from the companies interviewed. The first one after the data was transferred from tape to paper and the second one confirming our findings.

5.3 Presentation of results

In this section we will present the results from our analysis. The presentation is divided into three main categories, configuration management, quality assurance, and development methods. Each main category is divided into smaller ones that are based on one or more findings.

It is important to understand that a finding is an interesting observation found during the analysis and one cannot always draw a conclusion from it. For example there are some occasions where the amount of answers were too low (3 or less) and in those cases we have not and cannot draw any conclusions. The findings are always presented in a table while the conclusions drawn from the findings can be found separately.

5.3.1 Configuration management

Our overall impression from the analysis is that Configuration management is not a widely used term, regardless of what definition has been made, in the computer games domain. In many cases we had to explain the meaning of the term configuration management before conducting this part of the interview. That does, however, not say that the companies are not aware of the problems nor does it say that they don't have any form of CM incorporated into their industry.

In our presentation of the CM part of the analysis we made the decision to divide the subject into the following three subsections: Tools, Responsibility & Planning, and Builds & Problems.

5.3.2 Tools

As much as 80% of the interviewed companies said that they are using some sort of tool to avoid the problem with two or more people working in the same file at the same time thus overwriting each others work. This is not surprisingly since this is a problem that is often encountered early in the life cycle of a project, or in other words, when the first files are being created and worked on. Many of today's modern software also warn if more than one person opens a file and the problem thereby becomes even more apparent early in the project.

Usually such tools devoted to software engineering also support rollback which is when one returns to a previous version of the file, or files for that matter. 4 of the 6 answers we received said that their tool did support rollbacks. One of the two remaining answers said that they did not know how to do this even though they use a tool that does support rollbacks. Another company said that they only had one programmer and consequently they do not need it, which to us indicates that they believe only source code files can be roll-backed. When we continued this line of questions asking them how earlier versions are re-created 5 of 8 either did not know how to or simply didn't do it. The rest answered that they used the tool, which was the expected answer.

When a bug is reported it is important, since bug tracking becomes easier, to know which version/build of the product he or she has installed. Only one third of the answers said that they keep this information tracked. According to the companies that do keep track of this information it is done manually by a person or automatically by software.

By studying the interviews we soon found one interesting question; Is CM handled by one tool alone in the game companies? And the answer is in most cases, yes. 6 of 9 answered that they use only one

CM tool. This is clearly a limitation since the tool they said they use is not capable of fulfilling our definition of CM. In table CM1 we have summarised the tool findings.

Table CM1

Findings 1-5	Percentage	Answers (10 max)
Uses a file-sharing tool.	80%	10
Uses a tool that supports rollback.	67%	6
Knows how to re-create earlier versions.	38%	8
Keep track of which version is installed where.	33%	6
Uses only one tool for CM.	67%	9

Conclusion 1

In eight cases of ten the game companies use only one CM tool which primary purpose is file version and sharing control.

5.3.3 Responsibility & Planning

In every company there was someone responsible for that CM is handled properly within either the company or the project. One attention-grabbing observation was that in 5 of these cases the responsible person either has the role as a QM, lead programmer, or lead designer. These are all roles with high responsibilities.

Even though every company has a person responsible for CM, planning for CM is poor. Of the 9 answers we received there was only one case in where we could interpret it as they had a complete and written official plan. Their CM routines usually involve basic file tasks as handling shared files, merge, and builds.

In the planning phase tools should be selected that suits the project. Only 3/7 interviewed game companies said that they had some form of tool evaluation and in two of those cases only one of them evaluated between more than two tools.

Table CM2

Findings 6-10	Percentage	Answers (10 max)
Has a person responsible for CM.	100%	10
The person responsible for CM has other management functions.	50%	10
Has a complete written official CM plan.	11%	9
Evaluates between tools.	43%	7

Evaluated between more than two tools.	14%	7

Conclusion 2

Game companies assign CM responsibility roles but there is almost none documented planning made and the tasks performed are few.

Conclusion 3

The majority of the game companies interviewed do not perform tool evaluation.

5.3.4 Builds & Problems

Builds is one of the few issues nearly all the game companies take use of in a similar way. Basically there are two ways of doing a build. The first one is to make a daily build and the other one is to make builds, from the start of the project, in decreasing time interval until it too becomes a daily build.

Merge problems exist in none less than 3 of the 8 given answers. Other examples of problems are localisation, tool crash, and software version conflicts. It seems to be a list of rather trivial problems but remember that they can slow down a project at a serious cost when the daily efficiency drops due to these problems. A rather surprisingly observation is that the problems are only temporary solved without the root cause being solved. Merge problems are often caused by interface and data design slips but the companies still look for the problem in the code when the problem could have been avoided in the design phase [WALTON].

A to us unexpected finding was that none of the three companies that work with projects that are divided into two or more physical locations claimed that they never had a problem with CM.

Table CM3

Findings 11-13	**Percentage**	**Answers (10 max)**
Makes builds.	88%	8
Has merge problems.	38%	8
Have problems with CM due to different working locations.	0%	3

Conclusion 4

Builds are normally used in the game development industry.

Conclusion 5

Most problems found in CM are due to design and planning errors.

5.3.5 Final CM conclusion

Using these five conclusions obtained from the analysis we can now answer our problem statements regarding CM. The first question “How does the general game development configuration management look like?” is answered by conclusion 1-4. The answer is that there is responsibility assigned but planning is poor except for when it comes to making builds. The tasks are few and focuses on file management alone thereby making it possible to assign the responsibility of CM to people that already have other time consuming roles. As stated in the beginning of this section CM is not something that is widely used in the art of game creation.

Our second problem statement “What are the most common problems and solutions in software configuration management?” does consequently not have many answers beyond file problems. Our conclusion is that the problems found are caused by lack of CM planning, evaluations and a proper design. The solutions used in the industry are of quick fix (short term) characteristics.

5.3.6 Quality Assurance

In our presentation of the Quality Assurance part of the analysis we have made the decision to divide the subject into the following subsections: Problems in Quality Assurance and, Quality Assurance in general.

5.3.7 Problems in Quality Assurance

Developing games is a very time consuming and complex task [SCHAEFER]. According to our own experience most people in the software industry still miss judge the complexity and high percentage of human resources needed for QA. According to [BEIZER] "Testing and debugging costs ranger from 50% to 80% of the cost of producing a first working version of a software package." Therefore we decided to examine what QA in the game industry looked like and if this part of the development cycle would confirm our thesis.

Of the companies asked at least three stated that time was a problem. One company stated, “There is a compromise between time and quality”. Another one said, “There was too little time.”

One of the questions we asked was “who decides when the game is balanced, fun and ready for delivery?” It would be very interesting to find out who has the power of judging the product when it is ready for delivery. Quite astonishing we did not find any kind of general pattern during the analysis. It seemed to be “one company – one method of judgement”.

Other more interesting “hands on problems” experienced with QA that was found during the analysis was things like; lack of prioritising during testing, compability with software/hardware and communication.

It is also an interesting fact that the only three solutions have been presented for the problems encountered namely communication, education and making sure all testers has the right hardware.

Table QA1

Findings 14-15	**Percentage**	**Answers (10 max)**
Feels that they have too little time for QA.	30%	10
Companies with similar methods of judging when the product is ready for delivery.	N/A	10

N/A means that there were two pairs were each pair used the same method but the pairs did not use any method similar to the others.

Conclusion 1

Three companies stated that time is a problem to perform QA.

Conclusion 2

There is no general pattern of the power to judge when a game is ready for delivery.

5.3.8 Quality Assurance in general

During the analysis we discovered that the QA-domain is very limited in the industry. Only three out of eight companies stated that the QA-domain includes more that just testing of the games. Three other companies have very limited QA due to the fact that they make statements like "We have nothing said here. Nothing on paper." Our own definition of QA is stated at [Page 12] "Software quality assurance is the managed on-going procedure where document and software errors are prevented, detected and removed according to given procedures, which constantly are improved." However, even if the QA-definitions of the most game companies falls into this definition we can hardly agree to that the most of them actually have QA, more likely just pure software testing. According to [SANCURRAN] the QA-domain includes far more than just actually testing of a product. [SANCURRAN] states activities such as "documentation" and "Reviews and Audits". That is activities that almost no company (at the most three) have stated they are using and considering during QA.

We also asked each company to describe their testing procedure. Sadly enough, we could hardly draw any conclusions of the processes themselves. However, one very interesting fact was found. Three of all the companies did only have internal QA and three of the companies only had external QA (according to our definition), even though QA is usually a services publisher's offer. Two of Eight companies who answered the question had both internal and external testing. Of the eight companies who mentioned that they had some kind of only five of the companies had anyone responsible for QA.

An interesting fact found during the analysis is that of the 5 companies that have internal testing 4 has well defined testing procedure.

Table QA2

Findings 16-20	**Percentage**	**Answers (10 max)**
More activities than testing is part of the QA-domain	38%	8
Has someone responsible for QA	63%	8
Has internal QA	63%	8
Has external QA	63%	8
Companies with internal QA that have clearly defined testing procedures	80%	5

Conclusion 3

Three out of eight companies has stated that the QA-domain includes more parts that just testing.

Conclusion 4

Five out of eight companies have someone responsible for QA.

Conclusion 5

Both internal and external QA is used in the game industry.

Conclusion 6

Four out of five companies that are using internal QA has testing procedures that are clearly defined.

5.3.9 Final Quality Assurance conclusion

Using these 7 conclusions obtained from the analysis we can now answer our problem stated regarding QA. The first question "What does the general game development quality assurance look like?" is answered by conclusion 3-7. The answer is that QA is generally clearly defined, but the definition of QA in the game industry is different compared to other parts of the software industry [Page 12]. It is also a very interesting fact that a big part of the game companies has QA but no one responsible for it. Assigning responsibilities is a task of management and if the responsibility of QA has not been assign to anyone this is definitely a sign of poor management.

The answer to our second problem statement "What are the common problems and solutions in software quality assurance?" can be obtained from conclusion 1-3. The second conclusion is maybe the most interesting. There is no general pattern for power of judging when a product is ready for release. However, we cannot say if this is good or bad, just that differs from company to company. As stated above testing demands a very high percentage of the total development budget. This is a problem that has been defined in the game industry due to the fact that time is often a problem in game QA [QA-Conclusion 1]. According to the problems experienced in QA we must once again say that most of this problems occur due to the fact of poor management.

5.3.10 Development Method

In our presentation of the Development Method part of the analysis we have made the decision to divide the subject into the following subsections: Planning & Structure, Methods & Phases, Phases in detail and Problems & Solutions.

It is important for the reader to know that we believe QA and CM are natural parts of the Development Method (see our definition). Therefore the answers in the last subsection – Problems & Solutions might refer to QA and CM.

5.3.10.1 Planning & Structure

According to our own experience and our education the Development Method is one of the cornerstones in software development. Basically it stands for structure and planning (according to our definition). This has also been experienced by [GORDON2].

After analysing [Question 25] and [Question 26] (please see the appendix) we found out that seven out of ten companies uses some kind of plan. However, this doesn't necessarily mean that they have a defined Development Method. We also wanted to know if the companies are working in a structured way. According to seven of the companies we interviewed they are working in a structured way, only one said straight out that they don't. We found an interesting observation during our analysis of the two questions mentioned above. Six of the companies that are working in a structured way are using

some kind of plan. It is important to the reader to know that the definition of "a plan" reaches from a master schedule to a to do-list.

Table DM1

Findings 21-23	Percentage	Answers (10 max)
Feel that they work in a structured way.	70%	10
Uses a plan for their development.	70%	10
Uses a plan and feel that they work structured.	86%	7

Conclusion 1

The majority of the companies feel that they work in a structured way.

Conclusion 2

Having a plan makes people feel that they work more to structured.

5.3.10.2 Methods & Phases

It would also be interesting for our work to know what kind of Development Methods the game companies are using. During our research we asked the companies "Are you using any kind of official (widely known) Development Method?" [Question 26] No company answered that they do. However, this doesn't necessarily mean that the companies aren't using any kind of method. During our analysis we found out that five of seven companies did not have well-defined phases of the their development method. Only one company said that their phases were well defined and documented. One interesting observation was that the three companies that made the most detailed description of their phases had in general the same steps and phases of development.

Table DM2

Findings 24-26	Percentage	Answers (10 max)
Are using a widely known development method.	0%	10
Have well defined phases in their development method.	71%	7
Have well defined and documented phases.	14%	7

Conclusion 3

Game companies do not use widely known [Page 27] Development Methods.

Conclusion 4

Game development seems to be ad-hoc due to the fact that the development phases aren't well defined.

Conclusion 5

Even though game development seems to be ad-hoc a pattern of development phases have been found.

5.3.10.3 Phases in detail

We also wanted to examine more detailed facts about the phases in the development phases. Depending on the results from [Question 29] and [Question 30] it would be possible to partly confirm our thesis. We did found patterns both on resource allocation and costly problems. Three out of four companies stated that changes of resource allocation take place when development reaches the state of Beta. We have also found out that four out of five companies most costly problems can be derived from the design phase.

Table DM3

Findings 27-28	Percentage	Answers (10 max)
Changes resource allocations at Beta state.	75%	4
Most costly problems can be derived from the design phase.	80%	5

Conclusion 6

Changes of resource allocation take place when development has reach the state of beta.

Conclusion 7

Most costly problems can be derived from the design phase.

5.3.10.4 Problems & Solutions

As one can understand by now there are a bunch of problems the game companies suffers from. The last four questions asked to the companies were committed to solutions and recommendations. Four out of eight companies stated that communication is a problem or needs to be improved. Five out of nine companies stated that planning is a problem or a solution to a lot of problems. It also seems that game companies suffer from bad management. According to [Question 33] and [Question 34] three out of eight companies said that poor management is a problem or needs to be improved. Another problem presented is the working hours. Two companies stated that if the employees aren't working the same hours or if they are working too much the development would start to suffer.

Table DM4

Findings 29-32	Percentage	Answers (10 max)
Have problems with communication or feel that it needs improvement.	50%	8
Lack of or planning or planning is a problem.	56%	9

Management is a problem or need to be improved.	38%	8
Irregular working hours causes development to suffer.	25%	8

Conclusion 8

Generally the source of a lot of problems in the game industry is bad management.

5.3.11 Final Development Method conclusion

Using these 8 conclusions obtained from the analysis we can now answer our problem stated regarding Development Method. The first question "What does the general Development Method look like?" is answered by conclusion 1-6. Even though no companies are using any widely known Development Method and the phases are usually not well defined or documented a structure exists. We have made an observation that most companies use the same steps and phases during development. A conclusion that confirms this is the fact that changes in resource allocation is the same for the most companies.

Our second problem statement "What are the most common problems and solutions in the Development Method?" relies on conclusion 7-8. Many problems can be derived from the design phase, which will affect the development in a bad way. The problems and solutions for game development have generally been detected as management issues. It seems that the major problem in game development is not the development method it self, but rather poor management. If management could be improved most of the problems presented in this domain would disappear.

5.4 Summary

There are some limitations of the interviews we preformed.

- Limited amount of persons interviewed.
- No cross checking
- Limited interview time
- Lack of complete follow-up
- No verification.

The presentations of the results are divided into three main categories – configuration management, quality assurance and development methods.

During our analysis we found out that configuration management is not a widely used term in the game industry
Conclusions of the case study concerning configuration management:

- In eight cases of ten the game companies use only one CM tool which primary purpose is file version and sharing control.
- Game companies assign CM responsibility roles but there is almost none documented planning made and the tasks performed are few.
- The majority of the game companies interviewed do not perform tool evaluation.
- Builds are normally used in the game development industry.
- Most problems found in CM are due to design and planning errors.

The summary of our conclusions found is that the problems are caused by lack of CM planning, evaluations and a proper design.

During our analysis we discovered that the QA-domain is very limited in the game industry,

Conclusions of the case study concerning QA:

- Three companies stated that time is a problem to perform QA.
- There is no general pattern of the power to judge when a game is ready for delivery.
- Three out of eight companies has stated that the QA-domain includes more parts that just testing.
- Five out of eight companies have someone responsible for QA.
- Both internal and external QA is used in the game industry.
- Four out of five companies that are using internal QA has testing procedures that are clearly defined.

The summary of our conclusions found is that QA in the game business is generally clearly defined, but the definition of QA in the game industry is different compared to other parts of the software industry. Most of the problems in the game industry regarding QA is caused by poor management.

During our analysis we discovered that most of the companies uses some kind of plan, However, this doesn't necessarily mean that they have a defined Development Method.

Conclusions of the case study concerning development method:

- The majority of the companies feel that they work in a structured way.
- Having a plan makes people feel that they work more to structured.
- Game companies do not use widely known Development Methods.
- Game development seems to be ad-hoc due to the fact that the development phases aren't well defined.
- Even though game development seems to be ad-hoc a pattern of development phases have been found.
- Changes of resource allocation take place when development has reach the state of beta.
- Most costly problems can be derived from the design phase.

The summary of our conclusions found regarding development methods is that even though no companies are using a widely known development method and phases are usually not well defined or documented a structure exists.

6 Suggested improvements

6.1 Configuration management

6.1.1 Tools

Finding 1	80% use a file-sharing tool.
Reason	Of the 20% not using a file-sharing tool either the companies are small and located on different geographical locations working on internet and don't see no use in a file-sharing tool or they just simply don't use any file-sharing tool.
Solution	[BENNETAN] states: "Configuration control cannot be effective without the necessary tools and facilities." The solution is to make sure that the company has a file-sharing tool (e.g. Source Safe).
Comment	We cannot imagine the situation of game development without a file-sharing tool. Sooner or later someone is going overwrite or delete file. This will most likely mean, "lost of A LOT of man-hours".

Finding 2	67% use a tool that supports rollback.
Reason	The remaining 33% states that they simply just don't do this.
Solution	N/A
Comment	Not being able to do rollbacks should be considered as very immature of a company in the software industry. According to our experience there are so many times when you want to make a rollback just to get things working. However, it is not clear to us if they just don't do it or if the tools (if they are using any) have the functionality to do a rollback.

Finding 3	38% know how to re-create earlier versions.
Reason	A lot of companies seem to have a lack of knowledge in this field.
Solution	[BENNETAN] states: "Configuration control cannot be effective without the necessary tools and facilities." The solution on how to keep track of earlier versions is simply to make sure that the company has a version-handling tool.

Comment	This seems to be very strange! Either the persons we asked don't have the knowledge of this issue or their game companies must suffer from a lot of problems. How many of the game developers have not experienced a situation where a build or a delivery might be delayed because of critical crashes that are very hard to fix? If you cant recreate an earlier version you may loose valuable time just to track down what went wrong and what was changed.

Finding 4	33% keep track of which version is installed where.
Reason	Most companies say that they don't need to care about this matter. It is automatically handled by some kind of software.
Solution	[BENNETAN] states: "Configuration control cannot be effective without the necessary tools and facilities." The solution on how to keep track of installed versions is simply to make sure that the company has a version-handling tool that make sure each product can report its version.
Comment	According to our own experience it is very important to know that version is installed where. During the last project we participated in we came across a lot of problems during testing due to the fact people had different version installed. Either the version number is manually changed or, to save some time, a tool is used.

Finding 5	67% use only one tool for CM.
Reason	The tool in use is the only tool the companies know of that satisfies the working situation or it is the only tool they have tried. One company stated that they have made their own tools just to be sure that there are no drawbacks.
Solution	[BENNENTAN] states: "Effective configuration control requires effective and well-defined organizations." He also says that it is based on four concepts. The third concept is "Configuration control cannot be effective without the necessary tools and facilities." If a game company wants to have an effective configuration control it is therefore necessary to identify all areas of CM that needs specific tools.
Comment	The days of ad-hoc game development are gone. It is still surprising that professional developer does not understand the importance of good CM. Tools minimize risks mostly by compensating for human errors but they also save time.

6.1.2 Responsibility & Planning

Finding 6	100% have a person responsible for CM.
Reason	N/A
Solution	N/A
Comment	This is a sign of a software organization being mature. It is great to see that the game developers have recognized this important issue.

Finding 7	There is a 50% chance that the person responsible for CM has other management functions.
Reason	One company states that the reason is that the lead programmer was the only one with the knowledge of the CM software.
Solution	N/A
Comment	As we have stated earlier in the work we believe that the game industry is a very immature industry. Not having a person dedicated to an office like Configuration manager can be a sign of the immature nature of the game industry. However, the amount of workload for CM is depending on the size of the project. Therefore it might not be a full-time job required.

Finding 8	11% have a complete written official CM plan.
Reason	N/A
Solution	[BENNENTAN] states: “Effective configuration control requires effective and well-defined organizations.” He also says that it is based on four concepts. The fourth concept is “A configuration management plan must be developed at the beginning of the project.” He also states it must be a document. The solution is to establish a configuration management plan in a document.
Comment	Once again this is a sign game developers are not mature organizations. We have experienced the problems without a written configuration management plan, and we cannot imagine work in another project without one. A written document is an excellent way of ensuring that everyone always has access to the plan and thereby deviations are easier to avoid.

Finding 9	43% evaluate between tools.
Reason	Some companies states: that they are using tools that they know and which works fine.

Solution	[ROLMOR] gives a suggestion to form a tools-group that is responsible for all tools at the project/company.
Comment	Contemplating the kind of environment the game developers usually works in, we think that standard products are most of the time the best solution (easy to purchase, everybody knows them etc) so in one way we can understand why so many companies don't even spend time evaluating tools. On the other hand, we don't think that there are so many standards products, so evaluation should be a more present activity. If a group is dedicated and responsible for the evaluation and insurance of a consistent usage of tools in the company the activity are more likely to be carried out than if this is a shared responsibility.

Finding 10	14% evaluate between more than two tools.
Reason	Some companies states: that they are using tools that they know and which works fine.
Solution	[ROLMOR] gives a suggestion to form a tools-group that is responsible for all tools at the project/company.
Comment	In the kind of environment the game developers works in we think that standard products are most of the time the best solution (easy to purchase, everybody knows them etc) so in one way we can understand why so many companies don't even spend time evaluating tools. On the other hand, we don't think that there are so many standards products, so evaluation should be a more present activity. It is the groups' responsibility to regularly find and evaluate new tools and taking the decision whatever to upgrade or not.

6.1.3 Builds & Problems

Finding 11	88% make builds.
Reason	N/A
Solution	[ROLMOR] states: that "the software needs to be buildable every day…irrespective of whether system testing is preformed daily or not." [BENNATAN] recommends that "procedures" should be a part of the configuration management plan. Therefore the build activity should be planned and written down in the configuration management plan.
Comment	We think it is a very different way to develop games without the activity of builds. It must be very hard to test, make demos and fast be able to go back to an earlier working version of the product. It is extremely easy to loose time building the wrong product without regularly builds that makes it possible to obtain very early feedback on whatever the game is fun or not.

Finding 12	38% have merge problems.
Reason	One company stated that too large changes in the system before merging sometimes create huge problems.
Solution	One company stated: "…keep a good overview of our code and use languages that simplify merges."
Comment	According to our own experience this can be one of the most costly problems during development. Merging must been taken care of in the beginning of a project and there definitely must be well-established procedures for this task. The solution or rather tip found in the interviews are our recommendation to. At all time it is extremely important that the design is up to date and that there is at least one person that has a complete overview of it.

Finding 13	0% has problems with CM due to different working locations.
Reason	Development on two different physical locations is not a problem when you have access to internet.
Solution	N/A
Comment	Very interesting. Our experience does not agree with the answers we have received from the interviewed companies. Even though internet makes things a lot easier we don't think you can ever compensate the face-to-face communication.

6.2 Quality Assurance

6.2.1 Problems in Quality Assurance

Finding 14	30% feels that they have too little time for QA.
Reason	One company stated that they pressure of releasing the product on time is the major reason.
Solution	[ROLMOR] explains that the best way to save time is to defining, establish and write down a test process that covers the ENTIRE development of a game.
Comment	Wonder if there is ever going to be enough time for QA? According to [SCHAEFER] Diablo II had about 100.000 beta testers, and still Blizzard released a patch the same day as the release of Diablo II. Still a test process can assure you that at least the tested parts can be marked as upholding a certain quality.

Finding 15	There is no overall pattern of judging when the product is ready for delivery.
Reason	Basically all developers and publisher seems to work very different.
Solution	N/A
Comment	Very surprising! Both of us thought there were very well established in official process regarding this issue.

6.2.2 Quality Assurance in general

Finding 16	38% have more activities than testing part of the QA-domain.
Reason	N/A
Solution	[ROLMOR] states explain that the best way to save time is to defining, establish and write down a test process that covers the ENTIRE development of a game.
Comment	Once again we have found a sign that game developers are not mature organizations. Just pure testing is not what we Software Engineers calls quality assurance. Every software engineer knows that the sooner a problem is found the less it cost to repair.

Finding 17	63% have someone responsible for QA.
Reason	One company stated: “We let our publisher handle this function.”
Solution	According to [ROLMOR] this is a role that should exist at a game developer. If you don’t have one, hire one.
Comment	It is understandable if first time developers do not realize the importance of this role. However, it seems that a large percentage does not have a QA lead and that is really surprising. Our recommendation is to take use of someone familiar with QA and/or educate the project member(s).

Finding 18	63% have internal QA.
Reason	N/A
Solution	N/A
Comment	During the development of Ground Control we felt the internal QA team made a great contribution. There was a few guys dedicated to testing so the programmers could concentrate on fixing bugs instead of spending time to try to re-create hard bugs. We suggest that every game developer should

	have a small QA team of his or her own.

Finding 19	63% have external QA.
Reason	N/A
Solution	N/A
Comment	According to our experience external QA works fine. Usually it is one of the services that publisher offers/demands.

Finding 20	80% with internal QA have clearly defined testing procedures.
Reason	One of the companies that does not have a defined testing procedure states "Nothing on paper"
Solution	N/A
Comment	Working as a QA lead without a defined test procedure must be very hard. How do they know what has been tested and what has not? We just wonder how such a team can do their job properly!

6.3 Development Method

6.3.1 Planning & Structure

Finding 21	70% feel that they work in a structured way.
Reason	Iterative phases, design, a code standard, to-do lists, a plan, and planning made at the beginning of the project are the reasons why they feel that they work in a structured way.
Solution	According to [WALTON] Kesmai Studios started to use a new development process based on "methodologies used by other industries including the SEI Capability Maturity Model (CMM), ISO 9000…" He states that the process made the "developers feel more in control of the project". He also states that the processes "have the potential to take much of the chaos and uncertainty out of development".
Comment	It surprises us that 30% of the companies' feels that they are not working in a structured way. How come that the owners of a company can permit that kind of management? How can employees accept it? Development processes is a must in every software-project.

Finding 22	70% use a plan for their development.
Reason	There are few things more annoying that having to back up and redo weeks of works because of poor planning up front.
Solution	According to [ROLMOR] "The three things you need for trouble-free development are a plan, a plan, and a plan". They also states that "In a small team, planning helps to keep you on the critical path even when several tasks pile up and demand your attention. In a big team, planning is the oil that keeps that machine from jamming."
Comment	It is a mystery to us how you can plan to make the best game in the world without doing a plan for it! Again we consider this to be a sign of lack of maturity. Without a doubt a plan is needed just as the solution states.

Finding 23	86% of the ones that use a plan and feel that they work structured.
Reason	We all need to know the pieces of the project and what is supposed to be done. A plan provides this kind of information along with how it is structured.
Solution	N/A
Comment	This finding should be pretty obvious, however we find it interesting. If the most developers using a plan feel that they are working structured, how come that 30% of the asked companies does not have a plan [Finding 22]?

6.3.2 Methods & Phases

Finding 24	0% uses a widely known development method.
Reason	Their methods have evolved naturally over the years or they have already an idea of how to build the software.
Solution	N/A
Comment	As we see it the game industry is different to the most other software industries. The aim of the products is often different. Therefore it is not surprised that no game developer uses a "widely known" development method. However, for every new game developer out there we strongly suggest that investigate the existing ones before creating one from scratch.

Finding 25	71% have well defined phases in their development method.
Reason	As they can name all the phases included in their development they answer that the phases are well defined.
Solution	According to [ROLMOR] "In reality, things are more complex, with multiple interweaving strands of the main development phases occurring

	simultaneously." However, [BENNATAN] states, "It is… important to understand the different phases and how they relate to one another." To summarize [BENNATAN] a developer must have well defined phases to be able to be as productive as possible.
Comment	We interpret this as the game developers does not understand or care about the importance of planning and management. To be able to estimate the cost more accurate, phases are needed. Since some phases heavily depend on the completion of other it is a must that everyone know exactly in which phase the project is.

Finding 26	14% have well defined and documented phases.
Reason	The phases can be named but are usually not documented due to time problems.
Solution	According to Jan Benjaminson (CEO at Massive Entertainment) "If it isn't written down it does not exist!" Define and document it!
Comment	This is strictly a management problem. The project managers must take more responsibility for the project documentation and planning. Without documentation things will surely be forgotten and hence errors made.

6.3.3 Phases in detail

Finding 27	75% changes resource allocations at Beta state.
Reason	At this point most of the art team have finished their work at this time and can therefore be re-allocated to other projects.
Solution	N/A
Comment	It seems this is a way to make a game developer more productive. When you use the Beta stage e.g. all issues related to the art team should be done. Since there are often not many "art bugs" (according to our experience) parts of the art team can be reallocated to another project.

Finding 28	80% says the most costly problems can be derived from the design phase.
Reason	A slip in the design phase can result in months of worthless programming. If the design phase is to short it spills over into the development phase in a manner that is not properly controlled.
Solution	One of the interview companies suggest that the result of the design phase is tested – "if someone has a question which cannot be answered from the design document of prototype, then the game is not ready to go into full production". Take use of a development method with a proper design phase and specified activities.

Comment	This is probably a result of the bohemia game industry. From history point of view [ROLMOR] early game projects was done by just two or three guys hacking and slashing. Those days are gone but the mentality is still around. Hacking and slashing is no longer a proper way to do a game. Early design errors will most likely result in late and costly errors.

6.3.4 Problems & Solutions

Finding 29	50% have problems with communication or feel that it needs improvement.
Reason	People are not working standard working hours. Too few meetings being held when problems arise. There is no strict line of communication.
Solution	One of the interview companies suggests "Strict lines of communication. No shared responsibilities. Small groups that makes the decisions. And actually, regular working hours."
Comment	The game industry still has a tendency of being out of rules. One developer we interview stated that at their company everyone worked whenever they felt for it. According to him it was an obvious and serious problem, but the guys at the company just felt that that's the way you make games. By a strict line of communication one can make sure that everyone receives the important information. Small groups are used to increase effectiveness and regular working hours to make sure there is no time lost waiting for people with required information.

Finding 30	56% says lack of planning or planning is a problem.
Reason	It is hard to plan for new technology. People are being too overconfident in creating art or code. Too many members in the planning meetings make the planning unproductive.
Solution	One of the interview companies suggested, "Plan up font. No amount of time can make up for bad planning – especially when you need to make a milestone to get paid."
Comment	Is this a problem due to the fact of inexperience or lack of education, or maybe of the unwillingness to try to become a professional software developer. We don't know, but it is for sure a very serious problem that all companies should take care of in the beginning of a process.

Finding 31	38% says management is a problem or need to be improved.
Reason	There is a lack of proper education among the managers. The managers do not hold risks at a minimum and there is no safe solution at hand. There is too much shared responsibility and single individuals making decisions.

Solution	Hire an experienced person for this task.
Comment	Probably this is a result of that the average age in the game industry is low and the experience is missing. Also, a lot of times people are promoted by the wrong reason. Sure, you can be a great programmer, but it does not necessary mean that you are a good project manger. As the solution say it can be well worth to take use of experienced managers.

Finding 32	25% says irregular working hours causes development to suffer.
Reason	N/A
Solution	People simply miss each other due to the irregular working hours. One put it this way: "We would like to kick people out of the office at 5.30, get home, get a life and come back to the office tomorrow with a fresh mind."
Comment	Making games are fun; otherwise we would not be in this industry. However, it is actually a job and with a job comes responsibilities. Our advice it to use regular working hours.

7 Conclusion

After interviewing ten companies, reading about ten books and what seemed to be almost an infinite amount of articles we have come up with the following conclusion,

According to our analysis there is definitely a lack in the field of CM considering tools. As many authors has stated tools is a very important part that improves the activity of CM. Companies in the game industry must open their minds and integrate tools in their organisations.

Another important part of CM is planning and documentation. According to our case study this is also a field that can be summarized as immature. How can you plan to make a triple AAA game without a plan how to do it? Of course, a CM plan does not necessarily mean that you make a great game but it increases your chances of getting there.

There is definitely a lack of CM in the game industry. Without being too judgement full we can say that the game industry needs to improve a lot to reach the level of CM compared to other parts of the software industry. The advice we want to give the game industry is to appoint a fulltime Configuration Manager that is responsible for all planning and documentation together with all associated activities.

We found the QA results of the case study very interesting. During our pre-study we learned that QA is a very important and time-consuming task that is part of software projects from the day one. We have come to the conclusion that there is definitely a difference between QA in the game industry and other parts of the software industry.

Some of the game developers complained about that there was not enough time for testing. Would there had been enough time if they would have considered all QA aspects from the beginning? Maybe there will never be enough time, but if QA aspects are considered from day one a lot of the problems occurring late in the projects will never take place.

The QA tasks as testing have been concluded as clearly defined in the game industry. We have concluded that there is structure and well-defined procedures. However, our general conclusion of QA tasks is that the QA domain covers more than just testing in other parts of the software industry.

Even though the game industry is showing signs of maturity the conclusion is that the domain is far behind other parts of the software industry. The advice we want to give the game industry is to start consider a broader perspective of QA if they want to make products on a higher quality level. QA aspects of game development should start very early to be able to meet the quality level of other industries.

It is a very interesting observation that most companies we interviewed feels that they are working structured even though they don’t have well-defined development phases. Almost no company stated that they have documented their phases. Having this in mind, adding the fact that no company is using a widely known Development Method makes us really question the level of education and experience of the management of the game developers.

We have also come to the conclusion that game development seems to be very ad-hoc and that the most costly problems stated by the game companies can be derived from the design phase. It really surprises us that the game companies are aware of the importance of good design, but they does not seem to enforce it with structured development methods.

We must recommend the game industry to adopt, document and in force well defined development methods. This does not just include the development teams, but the entire organization. If the marketing departments understand the development methods they might not try to push unfinished products to the market.

The final word of all the information and observations we have gathered during this thesis must be that almost all problems we have found is related to either the management or Development Method. If parts of the project suffers bad planning, or the employees does not work during the same hours the management must to take its responsibility to improve the process of developing games. If the Development Method is so vague that the different phases are unclear improvements to the Development Method must be considered or a new Development Method must be adopted by the organisation.

The last remaining question to be answered is “is there a lack of maturity in the game industry that results in companies using their own invented, but not fully working, game development procedures”. We have concluded that the domains of CM and QA are far too narrow in the game industry, which results in a number of problems that can be avoided with a broader perspective of these domains. We believe this master thesis show that these domains still suffers from lack of maturity. Also, we have concluded that the development methods them self generally suffers from lack of planning, documentation and structure. Having all observations and information in mind, which has been gathered during our work, we must state that our thesis has been proved and confirmed.

We strongly believe that we have made a contribution to the game industry by publishing this master thesis. During our pre-study we tried to find as much information about the game industry as possible. We hardly found any papers or reports written on the subject in a scientifically way. By doing a qualitative study with contributions from ten of the worlds most respected game companies we think that we have shown what parts of the game industry looks like. By considering this work along with the problems and solutions presented, we believe that most game companies are able to improve their productivity and quality of their products.

We have not in any way tried to evaluate the estimated cost for applying our recommendations. This is a shortcoming since we believe most managers and the person in favour of applying the recommendations would ask themselves this question. However, due to the nature of the recommendations we have come to the conclusion that the best estimations are probably made at the company itself. Secondly, we do not feel qualified to do a general estimation for all the recommendations.

8 Future work

8.1 Improvements and additions

There are always improvements to be made. If we had more time our first job would have been to include more companies in our interviews. Increasing the coverage will most likely reveal new problems and solutions for use in the thesis. Some of the conclusions drawn will then doubtlessly slightly change and the ones not yet fully confirmed can perhaps finally be set. Using the analysis of the companies made so far new questions can gave been engineered to get an even wider understanding of today's game industry.

Other interesting areas such as requirement change management, concept engineering, and usability concerns in the game industry can be added. Perhaps concept engineering is the most interesting one since most of the gamers we have come in contact with argue that every good game's sales result depends heavily on it. It is therefore extremely appealing to investigate whatever a process for creating and measuring game-play can successfully be made or not.

8.2 Evaluation

It would be interesting to see our improvement suggestions applied in the computer game industry. This is also the only practical way to see whatever we have found relevant findings or not and without this direct form of feedback it is extremely tough to evaluate our work. Even then the evaluation is hard to perform since we have merely given directions rather than a full-scale solution. As in many other cases we would have needed as many companies as possible participating to see whatever their next project, after having applied our recommendations, is more successful than their previous or not. Still we need to consider the fact that one more finished project means more experience. Do the improvements then come from our suggestions or their previous experience? To make a long story short an evaluation would require a time-budget that we do not have.

8.3 Cost benefits

As with the cost estimations for applying the recommendations we have not put any effort in trying to estimate the cost benefits. We can, however, as software engineers in the game industry try to make some qualified guesses of where the major cost benefits can be made. Below the reader will find a short list with what we believe are the three best cost benefits that can be gained.

- Less changes

  If management adopts a documented development method that uses a well thought through design phase with clearly defined activities most of the costly changes can probably be avoided. By using common sense one can understand that a problem caught early in the development cycle cost less to correct than a problem caught later on. Keep in mind that 80% in the interviews said that the most costly problems can be derived from the design phase. Further, by adopting a complete

configuration management process it is always possible to immediately restore a project to its former state. This is not only a project security but also a well-needed function that we know will be used by the engineers.

- Less maintenance

  It is only human to make errors and perhaps even more human to make assumptions. That is exactly why quality assurance is needed and why a strict communication scheme between the different roles in the game development is of major importance. Sloppy planning or communication that have not gone through proper quality assurance can never ever justify re-shipment and product updates in a mass-market situation.

- Increased work satisfaction

  Having a management that understands what the project and project members need to carry out their everyday best will most likely increase work satisfaction. Bad planning, impossible deadlines, re-work, and a never-ending stressful work situation will probably decrease work satisfaction. It does not take much to realise that a company with such an environment will face problems, which already are named. The main reason for us mentioning this point is that we believe that the best games created out there has been created by employees having a dedication to game creation that have not yet been suffocated by these problems. Our suggestions take some actions to avoid this and that is a cost benefit not to be forgotten.

8.4 Possibilities

From these suggestions we would really like to see a complete game development method that enforces the findings we have made. We realize we have only touched a very small part needed for creating such a method but without some sort of confirmation from the industry that it can be done the actual value of our work still remains unknown. Nevertheless we will definitely make our best to incorporate, evaluate, and improve the result from our thesis in our everyday work.

9 References & bibliography

9.1 Literature and papers

[BENNATAN] E.M Bennatan; *Software Project Management: A practitioner's approach*; 2nd Ed; 1997; McGRAW-HILL Book company Europe.

[PRICE] Frank Price; *Kvalitet i alla led*; 1992; Svenska Dagbladets Förlag AB.

[MYERS] Glenford J. Myers, *The Art of Software Testing*; 1979; John Wiley & Sons, inc.

[GILB] Tom Gilb, Susannah Finzi; *Principles of software engineering management*; 1997; Addison Wesley Longman Limited.

[NICHOLAS] John M. Nicholas; *Managing industry & engineering projects*; 1994; Prentice Hall.

[TAKUBB] Armstrong A Takang, Penny A Grubb; *Software Maintenance Concepts and Practice*; 1996; International Thomson computer press.

[PAULK95] Paulk, M.C.; Weber, C.V.; Curtis, B.; Chrissis, M.B. *The Capability Maturity Model for Software - Guidelines for Improving the Software Process*, Reading, Addison-Wesley, 1995.

[BEIZER] Boris Beizer, *Software System Testing and Quality Assurance*; 1996; International Thomson Computer Press, inc.

[SANCURRAN] Joc Sanders, Eugene Curran; *Software Quality*; 1994; Addison-Wesley Publishing Company Inc.

[RUPABODA] Runa Patel, Bo Davidson; *Forskningsmetodikens grunder*; 1994; Studenlitteratur

[ROLMOR] Andrew Rollings, Dave Morris, *GAME ARCHITECTURE AND DESIGN*, 2000; The Corialis Group, LLC.

[DEVREP99] Papers written for the course PQM, 1999.

9.2 Web-pages

[PERSONALMD] Web page, *Stress linked to memory impairment,*
http://www.personalmd.com/news/a1999061412.shtml

[UDS00] Web page, *Welcome to UDS*,
http://www.uds.se/

[DI00] Web page, *Digital Illusions CE*,
http://www.dice.se/

[MSV00] Web page, *Massive Entertainment*,
http://www.massive.se

[SEERS] Web page, *Software Quality Management Knowledge Area*,

http://computing.db.erau.edu/SEERS/3-3-x.html

[CHAMP] Web page, *Object-oriented System Development*,
http://g.cs.oswego.edu/dl/oosdw3/ch1.html#SECTION00040000000000000000

[ARTNPRO] Web page, *The Psychology of Artists and Programmers*,
http://www.gamasutra.com/features/visual_arts/091997/synapse_psychology.htm

[GAMVIS] Web page, *The designers notebook*,
http://www.gamasutra.com/19990618/0618Ernest/0618Ernest.htm

[GDEVINT] Web pages, GameDev.Net Interviews,
http://www.gamedev.net/columns/interviews/dlong.asp
http://www.gamedev.net/columns/interviews/ghjelstrom.asp
http://www.gamedev.net/columns/interviews/ghowland.asp

[HOWLAND] Web page, *10 Things Gamers Need to Know About Game Development*,
http://www.lupinegames.com/articles/10gamerknow.html

[HOWLAND], Web page, *The 3 Completes of a Game*,
http://www.lupinegames.com/articles/3completes.html

[STARTUP], Web page, *How to start and grow your own game development company*,
http://www.gamespot.com/features/startup/sec1_6.html

[WIERZBICKI] Web page, *Production Quality Game Code*,
http://www.gamedev.net/reference/articles/article751.asp

[MCCUSKEY1] Web page, *Lone Wolf Killers, Part I: The Design Phase*,
http://www.gamedev.net/reference/industry/archive/article913.asp

[GDC01], Web page, *The Game Development Cycle*,
http://www.gdcentral.com/tgdc.htm

[MCCONNELL], Web page, *Software Engineering is not Computer Science*,
http://www.gamasutra.com/features/19991216/mcconnell_pfv.htm

[SCHAEFER], Web page *Postmortem: Blizzard Entertainment's Diablo II*,
http://www.gamasutra.com/features/20001025/schaefer_pfv.htm

[GORDON], Web page, *Artists and Game Design Documents: From Interpretation to Implementation*,
http://www.gamasutra.com/features/20000104/gordon_pfv.htm

[WALTON], Web page, *Problems, Benefits & Future Expectations*,
http://www.gamasutra.com/features/production/19981218/eng_disc_04.htm

[GS01], Web page, *Diablo II*,
http://www.gamespot.com/features/diablo2diary_dd/011200/p2.html

[GRUBER], Web page, *Do You Have What it Takes to Be A Game Developer?*,
http://makegames.com/chapt1.html

[WWW-ISO1], Web page, *Learning Center*,
http://www.iso9000.org/

[WWW-ISO2], Web page, *Configuration Management and ISO 9001*,
http://www.mks.com/products/scm/si/2134.htm

[CATA], Web page, *Company*,
http://catalystwars.com/Company.html

[GORDON2], Web page, *Problems, Benefits & Future Expectations*,
http://www.gamasutra.com/features/production/19981218/eng_disc_04.htm

[ANDERSSON], Web page, *2 Software Development*,
http://www.efd.lth.se/~d87man/EXJOBB/Software_Engineering.html

9.3 Bibliography

[MCCUSKEY2] Web page, *Lone Wolf Killers, Part II, The Development Phase*,
http://www.gamedev.net/reference/articles/article943.asp

[OLIVER], Web page, *Battling Bad Press: How to Survive When Your Game Falls Victim to Media Massacre*,
http://www.gamasutra.com/features/20000718/oliver_1.htm

10 Appendix 1 Interview questions explained

This part explains the background for the questions that are used in the survey.

The questions are divided into four categories: General, CM, QM and development phases.

10.1 General

During the interviews we want to extract as much information as possible from each developer. It is important to get as many questions answered as possible, but in case a developer cannot answer a certain question we have to live with it. In some cases we might interview a person that is an expert on a specified subject of our focus (i.e. quality assurance manager). In that case we will try to extract as much information about his field of expertise as we can.

The first set of questions asked is both used to warm-up the developer as well as forcing the developer to start thinking about how he/she really reflects the surrounding. The first questions focuses on the developer, as people tend to like to talk about themselves. Hopefully this will give the developer a good start and make him/her pay more interest in the interview.

Q1 What is you role (work assignment, responsibility) in the company?

Q2 What kind of previous experience do you have from the gaming industry?

Q3 What is your formal and informal education?

The first three questions are asked to get a picture of the developer. Which kind of insight into the company are we getting? Is it manager or worker level and what kind of background does he/she have? Ideally we would interview at least one manager and a worker in each company to see if their apprehensions of their work situation matches. It is also very interesting to see what kind of education the developer has, as this will most likely affect the answer's we're getting.

Q4 How would you describe your development environment? Please include tools, open-plan office, communication, and management.

This is the first step towards making the developer think about his current situation. This is really a warm up question but partial or whole answers for the later questions could be received here.

Q5 What is the average education level in your company?

Q6 What roles are included in your development team/process? Is there a need for more people in any role or is there a need for more roles?

Now that we have made the developer look at him/her-selves and the environment we want the developer to start thinking in terms of people and roles too. At this point we are all set for the real questions and perhaps we have already received some of the answers for them.

10.2 Configuration management

Q7 How do you prevent people from working in the same file at once?

Q8 How do you do to return to an earlier version of the software?

Q9 How is earlier versions re-created?

Q10 How do you know which versions that are installed where?

Q11 What tools are you using for CM? Do you see any special benefits from using this tool over other tools? Any drawbacks?

Q12 Have you used any other tools before? Which? Why did you decide to not use them?

Our impression is that configuration management maturity is often looked at as binary; either a tool exists or not. We do not share this binary vision but we still need to find out if the company uses a tool or not. If they do have a tool then did they evaluate in a product/short-term/long-term fashion or did they pick the first tool they stumbled onto? If they don't have one then have they solved the problem common file-problem or not?

The next step is to find out how well they know their tool. Games often involve multiple customers and if this is the case how do they keep track of the versions installed at the different customers? If one of them has made an evaluation then perhaps this experience could be used in our work.

Q13 In brief, how do your CM routine look?

Q14 Is there anyone responsible for configuration management?

Q15 How often do you create some sort of baseline (often called build)?

How do they handle backups, program configurations (localizations etc), program education and distribution of the compiled product-versions for testing? Are they limited in their view (i.e. binary) of configuration management?

Q16 What are the most common problems you have encountered within CM?

Q17 How did you solve them? How do you think they should have been solved?

Finally, we want to know the most common problems in CM. If we are to give suggestions on how to solve them, then it is very interesting to see if there are any good solutions already existing in the industry. The reason for having two questions, one asking how they were solved and another asking how the developer think they should have been solved, are based on our experience that many solutions are project-term fixes and perhaps not the best. In many situations there is simply no time for solution evaluation and in other situations experience is needed to find the most suitable solution.

10.3 Quality Assurance

Q18 How would you define a bug/defect/error?

Q19 Which areas are defined to fall within QA?

Q20 Please describe your testing procedure. Include start, input, output and when you regard yourselves as done with the testing.

Do they look at bugs at something that only exists in the product? How wide are their QA? Our assumption is that most companies believe that QA is all about testing. If this is not the case then what more do they include?

How does their testing procedure look? Is it ad hoc or do they follow any kind of structure? We suspect that most testing are ad hoc and do not follow any specific testing method besides this. Do they use people from outside the project/company to do the testing or is it people from inside? We believe in using both for several reasons but believe that this is not common. Is there any follow up, besides the fix, on the testing? Again, this is a matter of maturity and we want to know how matured the average gamed development company is. Does the developer have a test team of his or her own or does the publisher do all the testing? We believe that the most developers do not have a test team of their own.

Q21 Do you have any person responsible for QA?

Q22 Who decides when the game is balanced, fun and ready for deliverance? Criteria used?

How do they judge the fun of the product? Which criteria do they use? This is what games are all about. Using a test group is the expected answer but perhaps there is more to it that is not obvious. As with testing we believe in an internal as well as an external QA, has the company the same opinion?

Q23 What are the most common problems you have encountered in QA?

Q24 How did you solve them? How do you believe the problems should have been solved?

Once again we want to know the common problems and if there was a difference between the actual solution and the (now known) most suitable solution.

10.4 Development phases

Q25 Do you feel that you are working in a structured way? Why or why not?

Q26 Are you using any kind of official (widely known) development method? Which one?

Q27 How did you do when you decided which method to use?

Q28 What phases are included in your development method? Are the phases well defined?

The first question is really tied to the second one. Even though not expected there could be companies claiming to work structured but not working after any specified development method (they are then probably using one but do not know it). Secondly we want to know if any company uses an official development method, and if so why did they choose it? This is important since we assume that at least some modifications on the most known methods are needed to make them fit the game industry. The last question is used to get an idea of how the methods actually being used in the industry looks like. If we are going to build one ourselves then we to know how they look as there is often a reason for why they have been shaped that way.

Q29 How are the resource allocation divided among the phases?

Q30 In which phases do normally the most costly problems occur, of what kind are they?

Q31 What are your solutions to these problems? Do you evaluate between different solutions?

Q32 How do you think the problems should be solved?

As there is no time to create a complete development method we need to know where the most costly problems occur and of which kind they are. Again, by searching for solutions as well as problems we can get a basic set of common problems and solutions to evaluate and refine.

10.5 Other

Q33 Have you experienced any other problems that you think should be mentioned? How were they solved?

Q34 Do you have any recommendations for which you think is important in a game development process?

In the two last questions we let the developer talk freely in hope of getting hold of problems and solutions not mentioned earlier. We try to capture some of the developer's experience for use in our work.