Antti Kirjavainen

Development of Game-Based Learning Environments – Constructing a Quality Multi-Disciplinary Development Process

Dissertation
in Information Technology
October 6, 2009

University of Jyväskylä

Department of Mathematical Information Technology & Agora Center

Jyväskylä

Author: Antti Kirjavainen
Contact information: ajkirjav@iki.fi
Title: Development of Game-Based Learning Environments – Constructing a Quality Multi-Disciplinary Development Process
Työn nimi: Oppimispelituotanto – Laadukas monitieteinen kehitysprosessi
Project: Dissertation in Information Technology
Page count: 221
Abstract: This study examines the development process of digital game-based learning environments (GBLE). The goal of this study is on constructing a prescriptive process model for GBLE development. The research focuses on four main themes in GBLE development process: examining 1) teams and expertise, 2) tasks, activities and work products, 3) process control models and 4) constructing a prescriptive metamodel to guide future development projects.

The study is a part of the research project Human-Centered Design of Game-Based Learning Environments. The overall aim of the project is to construct a multidisciplinary and user-driven process for the development of digital GBLEs. The study was conducted according to the principles of development research. Process improvement and modeling methodology, reflective systems development, action research and NIM-SAD software process evaluation methodology was also adopted for use in this research. The research process was arranged around four GBLE development projects. The case projects utilized human-centered design methods to ensure user-driven approach to the development.

The main result of this study is the prescriptive metamodel for GBLE development. Te metamodel is based on the other results of the study: the experiences and insight on different team compositions, task and activity arrangements and development process approaches as well as different kinds of process control models used in the case projects.
Suomenkielinen tiivistelmä: Tämä on suomenkielinen tiivistelmä. Se ei ole vielä valmis.
Keywords: games, development, learning games, games for learning, education, game production, game development, game design, instructional design, learning sciences, instructional systems design, prototyping, development process
Avainsanat: pelit, kehitys, tuotanto, oppimispelit, opetuspelit, opetus, oppimisympäristöt, pelikehitys, pelisuunnittelu, kasvatustieteet, oppimisympäristöjen suunnittelu, tuotantoprosessi, prototyypit

Preface

This is the preface of the document. To be done.

Contents

1 Introduction

The aim of this dissertation is to offer solutions for arranging the development process of digital digital game-based learning environments in a way that maximises the possibility of success in the development. The focus is on the game development and software development disciplines, but the approach is by necessity multi-disciplinary. This goal is pursued through studying the existing literature on the subject and related disciplines as well as conducting and studying empirical development experiments of digital learning games.

This study is part of the research project Human-Centered Design of Game-Based Learning Environments. The overall aim of the project is to construct a multidisciplinary and user-driven process for the development of digital Game-Based Learning Environments.

The development of motivating and inspiring digital learning games is a complex and multifaceted task. At its best, the quality development of learning games is a multi-disciplinary and user-driven process, which thoroughly combines the expertise of fields such as educational sciences, software engineering, human-centered design, game design and development, and the content disciplines of a specific game [Kankaanranta et al. 2007]. Earlier research on game-based learning has located several elements that indicate the possibilities of digital Game-Based Learning Environments [Gee 2003] [Hämäläinen et al. 2008] [Kankaanranta 2007] [Lainema 2003a]. Earlier research has focused on defining the essential features of digital learning game as an end product [Chamberlin 2003] [Lainema 2003a] [Squire et al. 2004] [Stubbs & Pal 2003], the involvement of users in the design process [Nousiainen 2005], design of collaboration and authoring of collaborative scripts for game environments [Hämäläinen et al. 2008]. However, the process of digital learning game development has not yet been explored so thoroughly.

The related disciplines of software engineering and entertainment game development have however produced much research that is potentially applicable to the domain of game-based learning environment development as well. Disciplines applied to the research and development of other kinds of learning environments must also be taken into consideration. These include learning sciences and instructional design. These

related disciplines are discussed in chapter 1.4.1.

When it comes to related disciplines, there are not many studies that focus on the application of these to the development of game-based learning environments. Furthermore, the results derived from the diverse disciplines that could be applied to the game-based learning environment development offer different and often contradicting guidance in regards to how to arrange the development process.

This study started as a study on how to apply the knowledge from the software engineering discipline to the development of game-based learning environments. During the early stages of the study it became evident that the viewpoint of software engineering did not offer the optimal view to analyze and solve the problems faced in the development of GBLEs. Furthermore it was hypothesized that the other related disciplines, primarily those of entertainment game development, learning sciences and instructional design, could offer viewpoints and results that could be applied to GBLE development. Therefore the scope of the study was expanded to this multi-disciplinary treatment.

The main research question of this study is: *What are the key aspects of a quality learning game development process?*

The main research question breaks down to the following sub-questions:

1. What kind of team structure and composition is optimal for learning game development?

2. Into what tasks can the development process be divided? How are those tasks related? How can this be described with process models?

3. What kind of process control model would guide the digital learning game development toward high-quality products?

4. What kind of metamodel could be constructed to inform the development of game-based learning environments based on the results of the previous research questions?

The research questions and the research activities used to answer them are described in detail in chapter 2.1.

1.1 Game-Based Learning in Existing Research

To describe the subject of this research it is important to portray the current state of game-based learning. The field of game-based learning has exploded in the last ten years. Many research reports have been written about the use of games in the classroom (For example [Baltra 1990] [Becta 2001] [McFarlane et al. 2002]), the advantages and drawbacks of using game to support learning (for example [Amory et al. 1998] [Davidson 2004] [Egenfeldt-Nielsen 2004] [Fabricatore 2000], [Gee 2003], [Gee 2005], [Prensky 2001]) and the design of games for learning (for example [Chamberlin 2003], [Druin 1996], [Ravenscroft & Matheson 2002]).

Many teachers and researchers of learning have also discovered the possibilities of games to support learning. There are many types of learning games out there, and the field of game-based learning is too large describe in detail in this thesis. Therefore, we will discuss some examples of learning games to give an idea what types of learning games are most common and promising ones.

1.2 Examples of Game-Based Learning Environments

Here we will introduce some example games, Food Force (Unicef), RealGame (Timo Lainema) and Revolution (MIT) that do well to illustrate the varied field of game-based learning. We will use them throughout the introduction and Chapter 3 to illustrate practically how different aspects of game-based learning relate to actual game products and the different ways they can be taken advantage of. These three games have very little in common in addition to their common goal of supporting the learning of some skill or knowledge. They all, however, represent important advances in the research and practical use of game-based learning, each in its own field of use. However, the games have been selected by the criteria of offering a wealth suitable example material for this and the following chapter and not by their popularity or acknowledged excellence. In chapter 3 we will take a closer look about types and modes of learning that these games promote. The games are introduced below, but a more detailed description of each of them is in chapter 3.1.

The first of the example games, Food Force (Deepend 2006), is a Macromedia Director -based single player game that tackles the difficult subject of world hunger. It uses several mini-games and audio visually high class cut scenes to present the subject to the player. Food Force is published by Unicef. The mini-games include action and

strategic planning; in one mini-game the player has to utilize his or her reflexes to successfully drop a help packet from an airplane where in another the player is given the task of managing a farm to raise the self-sufficiency of the people in crisis.

Food Force uses the cut scenes as the most important way to allow the players to learn about the game's subject matter. In addition to that, the game's fictional world puts the player in the role of a humanitarian helper. This is used to motivate the player to get involved in the game and the subject matter. We will analyze Food Force more thoroughly in Chapter 3.1.1. Food Force is available as a free Internet download from its dedicated website www.food-force.com.

RealGame is a business simulation game that is developed to train and enhance the decision-making capabilities of experts and middle managers. It has been developed by Timo Lainema (2003) and is a multiplayer network game for Windows platforms made in Borland Delphi rapid application development environment. The RealGame is a business simulation game describing the main processes of a manufacturing company. The essential idea of the game is to position the participants in a role where they have to manage a manufacturing business. The game is similar to a wealth of business games of the same vein with one striking difference: the game time progresses in real time instead of just a few turns. The simulation system in the game enables the players to experiment with different decisions in different situations and presents them with a way to have quick feedback on their actions. The game is intended for both participants from middle management position to foremen of production and to business students.

While the game's audiovisual appearance is simple and "MS Office-like", the detailed simulation system that represents the reality of manufacturing companies allows for a high-quality game and learning experience. More detailed analysis of RealGame is included in chapter 3.1.2.

Revolution is the Education Arcade's multi-player, American Revolution-themed role-playing game based on historical events in the town of colonial Williamsburg (MIT). Revolution is a multiplayer 3D game-based learning environment developed by MIT in co-operation with colonial Williamsburg. Revolution has been developed as a game modification (aka. mod) on the basis of Neverwinter Nights engine, a popular computer roleplaying game engine.

Revolution is a multiplayer computer role-playing game. This means that the players take the roles of fictional characters of the fictional colonial Williamsburg and act through their characters in the fictional setting. The game promotes learning by allowing the players to experience and interact with a historical environment and events

related to larger-scale historical events of the period. A more detailed description and analysis of the game is presented in Chapter 3.1.3.

1.3 Games and their Properties

When discussing design processes it is important to explore the features of the artefact to be designed. The notion of games has been explored very much in the last few years. The emergence of game studies as a new discipline has contributed much to that discussion. The origins of game studies are, of course, in earlier research, namely that of Roger Caillois and Johan Huizinga. In this chapter we will introduce the notion of games and describe the ideas behind using games to support learning. A more thorough exploration of this subject will be presented in Chapter 3.

Game-playing is an ancient human activity - there is evidence of games from the stone age [Crawford 1982]. Today, the field of games is a varied one: There are many kinds of games, including board games , social games, digital games played on personal computers as well as dedicated gaming consoles, single and multiplayer games and a multitude of software toys that reside in the border of gameness. This versatility of games cultures and artefacts has presented a problem to the game scholars: How to construct a definition of games that describes the notion powerfully while still not excluding any facets of gaming.

Research on games has not really been widely spread until the recent years. Some academics like Huizinga [Huizinga 1955] and Caillois [Caillois 1961] have studied games from a cultural perspective during the 20th century, but most progress in this field has come in the turn of the millennium. During the 1990s academics from different disciplines started researching digital games (For example [Aarseth 1997], [Cassel & Jenkins 1998], [Funk & Buchman 1996]). In the same time, game designers have started gathering body of knowledge about their industry and work techniques in game design (for example [Costikyan 1994], [Church 1999], [Ryan 1999a], [Ryan 1999b]).

There has been numerous attempts of this kind of definition. We have described the most relevant ones, both for merit and historical significance, in Chapter 3. For this introduction, however, it suffices to present the general definition of game used in this study: *games are activity which is restricted by rules, goal-oriented and somewhat distanced from reality.* So, first of all, games are activity: playing is active ('doing') rather than passive experiencing. In Food Force, the player takes on missions to help the hunger problem, in RealGame, the players make decisions to control the fictional

manufacturing company and in Revolution players act as fictional citizens of colonial Williamsburg. Goal-orientedness is also clearly visible in the example games: The goal in RealGame is to make the company as profitable as possible, each of the Food Force's six missions have their own goals and different characters in Revolution have their own goals.

The distance from reality can mean many things in games. Even though RealGame represents the reality of a manufacturing company quite accurately, the potential losses the player may experience during the game are fictional, so the player can experiment freely and safe from going bankrupt. And, even if the hunger crisis in Food Force's back story is completely fictional, the game invites the players to the fictional crisis and to take it seriously during the course of the game. This is called *suspension of disbelief*.

The aforementioned goal-orientedness of games also results in two essential elements of games: challenges and conflicts. Challenges are obstacles that are in the players' way as they try to achieve the goals [Adams 2005]. Conflicts emerge when players' goals are inter-related so that not everyone can meet their goals at the same time. Challenges and goals make the pursuit of in-game goals uncertain and so more interesting.

1.4 The Development of Game-Based Learning Environments

The development of game-based learning environments, as said in the beginning of this thesis, is a complex and multi-faceted task. Digital games designed to facilitate and support learning have many features; they are part of learning interventions or instructional systems and they are games as well as software. There are distinct disciplines that thrive to study the design of all these things. In this study these disciplines are used as a basis for exploring this relatively new domain of development of game-based learning environments. The three main related disciplines studied are the design of learning interventions, entertainment game development and software engineering.

In addition to the three aforementioned disciplines, the discipline of human-centred design is used in this study. Human-Centred Design (HCD) is one of the principal approaches in user involvement. [Nousiainen 2008] In this study human-centred design has a special position: HCD methods and theory are used in every endeavour of the research group this study has carried out in takes part in. Regarding HCD, the focus of this study is how to arrange the rest of the development process so that the fundamentals and methods of the HCD disciplines can be carried out as suggested. Investigating

the HCD methods themselves and how to take advantage of them in GBLE development is not a part of this study. These questions are discussed in another Ph.D. thesis completed in the same research group (see [Nousiainen 2008]). The basis and process of HCD is described in overview chapter 1.4.2.

1.4.1 Related Disciplines: Design of Learning Interventions, Entertainment Game Development and Software Engineering

The design of learning interventions is a term chosen in this study to represent two related but distinct academic disciplines, namely learning sciences and instructional systems design (ISD). Learning sciences and design-based research study the essential features and design of learning interventions whereas instructional systems design studies the development of instructional systems. The entertainment game development discipline (also called game development for short) explores the development of games from concept to post-production. The related discipline of game design studies the design of games as well as the features a game must have in order to be enjoyable. Software engineering studies the development of high-quality software.

The knowledge of each of these disciplines is applied for this study. The knowledge of learning sciences and design-based research methodologies as well as instructional design and their application and relevance to learning game design is discussed in chapter 3.2. Although the fields of learning sciences and instructional design have traditionally taken quite different approaches to studying the design of learning environments, the study uses knowledge of both to get a picture of learning game design from the educational standpoint.

In chapter 4, we discuss entertainment game development and design from the point-of-view of learning games. Although the existing theory from game development and design is mostly from the domain of entertainment games, many principles and guidelines are also applicable to learning games. The literature of entertainment game development is studied in the domain of game-based learning in chapter 4. The applicability of entertainment game development methods in game-based learning development are investigated in the case projects of this research. Modifications of these methods to better suit GBLE development are also investigated.

The selection of software engineering knowledge and methods relevant to learning game software development is discussed in chapter 5. This includes the exploration of various types of software development process models and their application to learning game development. Also, methods for specifying and designing learning game software

are discussed.

Through all these three chapters we thrive to provide possible or partial answers to the three sub-questions presented at the beginning of this thesis. Therefore, all of these chapters concentrate on each of these disciplines' answers to successful team composition, high-quality development process as well as the process control model. In chapter 6 we will provide the results achieved in this study to the first sub-problem: the optimal team structure and composition for learning game development. Chapter 7 describes the results to the second sub-question of the research question, the development process and its tasks. Chapter 8 provides results on the selection and adaptation of a process ontrol model for the GBLE development process. Finally, a new metamodel of GBLE development process is presented in chapter 9.

1.4.2 Introduction to Human-Centred Design

The involvement of users in the design of technology is considered one of the prerequisites for high-quality design. Human-Centred Design (HCD) is one of the principal approaches in user involvement. [Nousiainen 2008]

As mentioned before, this study does not directly discuss using HCD methods in GBLE development. Regarding HCD, the focus of this study is how to arrange the rest of the development process so that the fundamentals and methods of the HCD disciplines can be carried out as suggested. Investigating the HCD methods themselves and how to take advantage of them in GBLE development is not a part of this study. These questions are discussed in another Ph.D. thesis completed in the same research group (see [Nousiainen 2008]).

HCD process is an iterative process [Nousiainen 2008]. HCD cycle consists of the following activities: 1) Planning the process, 2) specifying the context of use, 3) specifying the requirements, 4) producing design solutions, and 5) evaluating the design solutions [International Organization for Standardization 1999]. The first activity is normally performed by the HCD designer and the rest of the activities are performed collaboratively by designers and users (often called user designers in HCD).

There are no preset methods of working in HCD. The process and the execution of activities must be planned and customized separately for each individual project. [Nousiainen 2008]

In this study, the basic process model of HCD is used in conjunction with the process models of other related disciplines. The HCD cycle description is seen as a sequence of activities. The tasks used in these activities are not explicitly defined with

a literature review in this study. Instead, it is assumed that most of the HCD specific activities are carried out with the following general capability pattern (Fig. 1.1).

Figure 1.1: The capability pattern of HCD activities.

In the metamodel it is assumed that the collaborative work done by the development project group and the user designers is donw in workshops designed by the HCD expert in the project group. The first task in the activity is therefore planning the workshop, which results in the workshop plan containing an outline of the workshop along with its goals (the work product of the task). The next task is the workshop, which produces some kind of output as a work product (typically a design or assessment of a prototype). The final task is integrating the workshop output to the other outcomes of the project, which can mean refining the requirements, changing the idea or concept document, changing the design and/or modifying the end product of the project.

2 Research Methodology and Setting

The research questions and methods are described in this chapter.

First, we introduce the overall research question and it's composition to three sub-questions (chapter 2.1). Second, we describe the overall research methodology (chapter 2.2). To support the methodology description we introduce a general model of games for learning development process that we use to illuminate the intertwining nature of development and research and the approach used in this study to make the development research process a scientific one. Third, we introduce the several research methods used to collect and analyze data during the course of the study (chapter 2.3). Finally, we describe the case projects used in this study and their contributions to the research questions (chapter 2.4).

2.1 Research Questions

As outlined in the introductory chapter, this study focuses on the learning game development process. This process is recognized as a meeting point for developers from different fields and disciplines, such as designers of learning interventions, game designers, software engineers and user-centred design experts. The development process is therefore explored as a multi-disciplinary process. The main interest is to develop learning game development to allow for creation of quality learning game products. Therefore, the main research question is formulated as follows:

What are the key aspects of a quality learning game development process?

The main research question is further broken down into the following three sub-questions:

1. What kind of team structure and composition is optimal for learning game development?
    - What kind of expertise is needed in the development of digital learning games?
    - What kind of roles should people of different expertises take in learning game development?

- How do different kinds of learning game development projects affect this optimal team composition and structure?

2. What shape can the development procInto what tasks can the development process be divided? How are those tasks related? What are the work products of these tasks?

    - What kind of development processes can be elicitated in GBLE development?
    - How do these development processes compare with development process models in learning sciences, entertainment game development and software engineering?
    - What kind of prescriptive process models can be formulated for tasks and activities in GBLE involving more than one distinct discipline?
    - What are the essential tasks and activities in the learning game development process?

3. How to control the process of developing game-based learning environment?

    - What kind of process control model is most advantageous and feasible to the development of GBLE?
    - What kind of adaptation of that process control model would lead to a high-quality development process in this domain?

4. What kind of metamodel could be constructed to inform the development of game-based learning environments based on the results of the previous research questions?

2.2 Methodological Framework

The overall methodological framework of this study is adopted from development research, a research method for researching and developing pedagogical interventions. The methodology is supported with process improvement and modeling methodology adopted from software process study discipline. In this chapter we will outline the overall methodological framework of this study. The chapter begins with a description of development research and its application in this study. After that the process improvement and modeling methodology is described.

2.2.1 Development Research Process

Development research is a research method for researching and developing pedagogical interventions [van den Akker 1999]. Development research is an umbrella term chosen to represent all the research approaches and methods concentrating in problem-oriented and interdisciplinary approaches that study the design or development of educational materials, curricula, educational media & technology etc. The most important factor in research is the search of understanding. In development research, the object or content of the understanding is not unambiguous [Plomp 2002]. The focus of development research is to provide prescriptive knowledge and useful solutions for a variety of design and development problems in education [van den Akker 1999].

In development research the focus is on researching and developing the development process. The two main kinds of discoveries in development research are guidelines that enhance the development process and guidelines about the essential features of certain kinds of educational interventions. [van den Akker 1999] In this study, the research questions guide the research toward discoveries of the first kind: guidelines that enhance the development process.

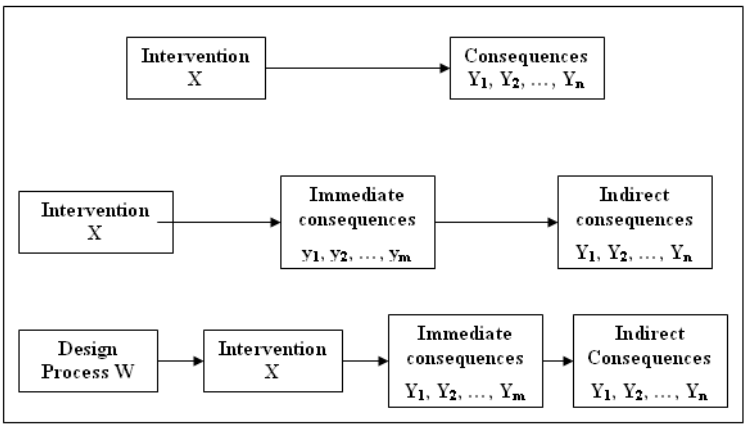

Figure 2.1: The Development Model of Development Research [van den Akker 1999].

The development model of development research is in illustrated in figure 2.1. This development model is based on the following assumptions:

- Intervention theory: Intervention X leads to assumed and desired immediate consequences (y1, y2, ..., ym)
- Influence theory: The immediate consequences lead to indirect consequences (Y1, Y2, ..., Yn)

- Design process theory: If we organize the design process of the intervention according to process W we expect that the resulting intervention will have the features X.

The goal of reducing uncertainty of decision making in designing and developing interventions can be divided in two more specific goals: 1) providing ideas for optimizing the quality of the intervention to be developed and 2) generating design principles [van den Akker 1999]. The first goal relates to the features of the intervention (X in the figure 2.1) and the second goal relates to the process (W in the figure 2.1).

In this study the focus is on the development process and design principles. The intervention in the context of this study is the game-based learning environment to be developed and the process is the process of GBLE development. The desired features of the intervention are analyzed in the literature review of existing game-based learning theory. The results are presented in chapter 3.3.

Development research is often used in connection to complex innovative tasks for which there are not many validated principles to structure and support the design and development activities [van den Akker 1999]. The research often focuses on realizing prototypes of complete interventions that serve as limited but promising examples. The prototypes are produced in succession with each successive prototype increasingly meeting the innovative aspirations and requirements. The process is cyclic: analysis, design, evaluation and revision activities are iterated until a satisfying balance between ideals and realization has been achieved. [van den Akker 1999]

The starting point in development research is the design of an intervention and its development as a solution for a need. This development is not done as research as the design or development itself is not considered research. The notion of intervention is defined broadly by development research: it can be a method of instruction or learning process, or a tool or product that supports learning or instruction [Plomp 2002]. This makes development research as a valid research method also in researching the development of game-based learning environments.

There are four major activities in development research methodology [van den Akker 1999]:

1. Preliminary investigation
2. Theoretical embedding
3. Empirical testing
4. Documentation, analysis and reflection on process and outcomes

Preliminary investigation is a more intensive and systematic examination of tasks, problems, and context. This is done to find information about good ways to go about the development work to be done as well as earlier experiences from similar domains. Typical activities include literature review, consultation of experts, analysis of available promising examples of related purposes, case studies of current practices.

In theoretical embedding more systematic efforts are made to apply state-of-the-art knowledge in articulating the theoretical rationale for design choices. A theoretical articulation of this kind can increase the transparency and plausibility of the rationale.

Clear empirical evidence about the practicality and effectiveness of the intervention is delivered through empirical testing. Testing is done with real user groups and in real settings. As the variation of possible interventions and contents is wide a broad range of indicators for success should be considered.

To contribute to the expansion and specification of the methodology of design and development much attention is paid to systematic documentation, analysis and reflection on the entire design, development, evaluation and implementation process and on its outcomes.

This study bases its research process in the research activities of development research. The research is arranged around four case projects, each of which feature a practical application of development of game-based learning environments.

The nature of the research process is cyclical: One cycle of research is arranged around each case project. Each cycle contains the three activities of development research. Before the start of the development project the preliminary investigation and theoretical embedding activities are carried out. Empirical testing is carried out as a part or after the case project. The documentation, analysis and reflection activity is carried out during the whole cycle of research, starting before the case project has started and ending after the project has been completed.

The preliminary investigation activity includes literature reviews, analysis of promising examples of other game-based learning environments and case studies of practices in GBLE development as well as the three related disciplines that are covered in the thesis' literature review chapters. In later cycles the results of previous research cycles is also analyzed in this activity to provide guidelines for the following case projects. The work product of this activity is knowledge and insight on arranging the development of the case project in question.

Theoretical embedding activity follows; in this activity the decisions of development methodology are made based on the results of the preliminary investigation. The

development methods used are connected to the existing knowledge in the field and the knowledge gained in the previous cycles of the research. The focus is to cover a range of state-of-the-art methodology.

The empirical testing activity is carried out during and after the project. The case projects had testing phases in the projects to ensure the quality of the products developed in the projects. In addition to that some of the game-based learning environments developed in the case projects were further tested in field trials related to other studies in the Agora Game Lab research group.

As mentioned before the features of the intervention i.e. the developed game-based learning environment is not the focus of this study. Therefore in addition to testing the quality of the finished product of the case project the quality of the development project is assessed as well. In this study this was carried out by using four distinct research methods in all four research cycles: historical mapping method and project diaries of reflective systems development methodology (see 2.3.2), process modeling (see 2.2.2) and the evaluation of project methodology with the NIMSAD framework (see 2.3.1).

Development research aims at making both practical and scientific contributions. The interrelation between theory and practice is complex. The research is not focused solely on examining if a theory is a good predictor of events but rather examining if it is possible to create a practical and effective intervention for an existing problem in the real world. Direct application of theory is not sufficient to solve these kind of complex problems. Researchers and practitioners co-operatively construct workable interventions and articulate principles that support the effects of these interventions.

Most of the knowledge gained by development research comes in form of 'design principles' to support designers in their task. These design principles are heuristic statements in the form of: "If you want to design intervention X for the purpose Y in context Z, then you are best advised to give that intervention the characteristics A, B and C and to do that via procedures K, L, M , because of arguments P, Q and R. These principles cannot guarantee success but are intended to apply the most appropriate knowledge for specific design tasks. [van den Akker 1999]

2.2.2 Process Improvement and Modeling

Each development project has a development process. When the process has not been planned, the process is implicit. The notion of process is prevalent in the software engineering discipline. The software process is the set of activities and associated

results which produce a software product [Sommerville 2001].

Making processes explicit is the first step towards process improvement [Germain & Robillard 2008 The whole process becomes visible by creating a model of the process. This defined process is the requirement for process understanding, analysis, execution guidance, learning and communication. It will also support the improvement of the process. [Terävä 2005] The ultimate benefit of explicit process models should be a better understanding of development processes, resulting in improved quality products and more realistic estimations of the budgeted effort [Germain & Robillard 2008].

A process model can be descriptive, prescriptive or proscriptive. Descriptive modeling tries to make the currently used processes explicit. Prescriptive modeling specifies the recommended way of executing the process. Proscriptive modeling describes non-allowed behavior and is usually used as an addition to prescriptive or descriptive modeling. [Koskinen 1999]

In this study, the processes of the case projects are elicitated and a multi-disciplinary prescriptive process model of GBLE development is formulated. The process modeling technique used in this study is Software & Systems Process Engineering Meta-Model version 2.0 (SPEM 2.0) [Object Management Group 2007]. The SPEM2.0 meta-model is introduced in the following paragraph.

Software & Systems Process Engineering Meta-Model 2.0

SPEM 2.0 is a metamodel used to define software and systems development processes and their components [Object Management Group 2007]. SPEM is based on the idea that a software development process is a collaboration between active abstract entities called roles which perform operations called activities on concrete and real entities called work products. The different roles act upon one another or collaborate by exchanging products and triggering the execution of certain activities. The objective of a process is to lead a set of products to a well defined state. [Combemale et al. 2006]

The models in SPEM 2.0 consist of method content, managed content, process structure and process behaviour descriptions. Method content consists of work definition (tasks and activities), work product and role elements. Managed content allows to associate guidance and metrics for different elements of the models. Process structure describes the decomposition of activities and their references to roles and work products. Process behaviour models allow to describe the behaviour in processes, including starting and finishing points of activities. Process models can also have the following logical objects: phases and iterations with which to divide the process into smaller segments. [Object Management Group 2007]

In this study the SPEM 2.0 meta-model is used together with the Eclipse Process Framework, which is an open source implementation of SPEM 2.0 [Object Management Group 2007]. The method content is modeled with work products, roles and tasks with guidance and the processes are modeled with phases, iterations, activities and the uses of the elements defined in the method content.

2.3 Data Gathering and Analysis Methods

In this chapter we describe the main methods used to gather and analyze data in this study. NIMSAD, a meta framework for evaluating methdologies, was used to evaluate the development methdologies used in the case projects. In addition to that, two methods from reflective systems development were used: mapping situations and diaries. Historical mapping of situations was used to gather knowledge about the case projects together with the developers and analyze the knowledge in collaboration with them. Diaries were used to gather data from all the people that worked with the development projects.

2.3.1 NIMSAD

NIMSAD [Jayaratna 1994] is a meta framework for evaluating methodologies. It was originally developed for evaluation of information systems development methodologies [Koskinen et al. 2004]. Jayaratna defines a methodology as a way to structure and rationalize activity and critical and creative thinking [Jayaratna 1994]. The definition is wide and applicable to various contexts [Koskinen et al. 2004]. NIMSAD has been developed for a long stretch of time amid practical work and consultation and has been developed as action research during that time. NIMSAD is the most known of the information systems development methodology evaluation frameworks.

The aims of NIMSAD are 1) to act as a means of understanding the realm of problem solving on a general level, 2) help to evaluate methodologies, their structure, phases, form, etc. and 3) to help in forming conclusions related to problem solving and methodologies.

NIMSAD uses four basic elements to examine problem solving and methodologies (Table 2.1). NIMSAD is based on these elements which are divided in the meta framework into 18 lower level elements described on the right-most column of the table.

The innovative aspect of this evaluation framework is that it tries to capture the

Table 2.1: The Basic Elements of NIMSAD

1. The problem situation (methodology context)	1. The use situation 2. The beginning of the use of the methodology 3. Stakeholders 4. Context description 5. Methodology's culture and politics 6. The risks of context description 7. The risks of methodology
2. The future problem solver (the methodology user)	8. Motives and values 9. Abstract reasoning needed 10. Skills needed
3. The problem solving process (methodology)	11. The problem situation and it's boundaries 12. Situation diagnosis 13. System prognosis 14. The problem specification 15. Derivation of conceptual systems 16. Design 17. Implementation
4. Evaluation.	18. Evaluation

methodology in action in an organization. Other advantages of NIMSAD are it's thoroughness and its wide area of usage. The drawback of NIMSAD is that it relies heavily on the evaluator's view of the methodology, target organization and the methodology user and their relationships. As such is does not offer formal measures of evaluation of any of its elements. NIMSAD is very well suited to evaluate overall development methodologies in an organization but the lack of formal measurements diminish its importance in evaluating specific methods for doing a specific task. Therefore NIMSAD will be used to form informed opinions on the methodologies of development projects as comprehensive wholes.

In this study NIMSAD was used to structure the researcher's evaluation of the case projects based on the project documentation and the comments of different stakeholders of the project. The evaluation focused on the elements of NIMSAD directly related to the research questions of the study. These were the future problems solver related to the first research question and the problem solving process related to the second research question.

2.3.2 Reflective Systems Development

Reflective systems development (RSD) is a methodology developed for researching and developing information systems. RSD has been developed in the Computer Science Department of Aalborg University by professor Lars Mathiassen and his research group [Mathiassen 1996]. RSD concentrates in the practice and research of information systems development. The basic principles of RSD from the practical and research standpoints are presented in table 2.2.

RSD aims to answer the following questions: how can we understand, support and develop systems development activity? How should we organize and do research in the area? How could we form a relationship between practice and research? [Mathiassen 1996]

RSD uses action research as its primary method of research. The problems, challenges and possibilities of information systems development are taken as start points in the research. The research work produces experimental knowledge that leads to new more developed practices. Most of the research material in RSD comes from the practical work done in information systems development. This material is deepened by reflection in action. [Mathiassen 1996] Action research sets practice as top priority and highlights the natural relationship between practice and research [Baskerville et al. 1996].

This kind of research approach is similar to development research. However, the development research can be considered a more rigorous overall methodology as it defines

Table 2.2: Basic Principles of Reflective Systems Development

	Research	**Practice**
Aim	To develop information with which one can understand, support and develop the practice as part of the ongoing professional development	To develop computer-based information systems as a part of ongoing organizational and institutional development
Framework	Dialectics	
Reference Sciences	Information Systems Development Theory	Information Systems Development Methodologies
Process	Action Research	Reflection in Action
Arena	Information Systems Development Practice	Information Systems Development Practice

a clear process for research in development projects and specific research tasks inside that process. The value of RSD comes in specific techniques of gathering information and reflecting on the development work. In this study the two most useful of those are used: mapping situations and development diaries.

Mapping situations introduces maps as descriptions and interpretations of project's situations [Lanzara & Mathiassen 1985]. They help gather and organize relevant and often overlooked information and observations. Maps in this sense are cognitive constructs that contain images made by actors about the situations they are in. The mapping activity can be done collectively to form a common view of a situation and to make implicit and personal knowledge about the situation explicit and shared.

In this research mapping situations is used during and at the end of development projects. The primary map type used is historical mapping which is sometimes deepened by diagnostic mapping. The main function of mapping is to document and reflect the development processes in the development of the learning game prototypes.

The aim of historical maps is to describe the whole development project from beginning to end. The main dimension of historical map is time and it is divided to as small units as necessary to provide a description as detailed as is needed. In the time line of the project the map depicts the events, issues,circumstances and the project team's actions in the project of the project as well as the relationships between those entities (Fig. 2.2).

The project's events and issues are marked on the up most axis of the map. Events are moments when planned situations are reached as planned. Issues are deviations

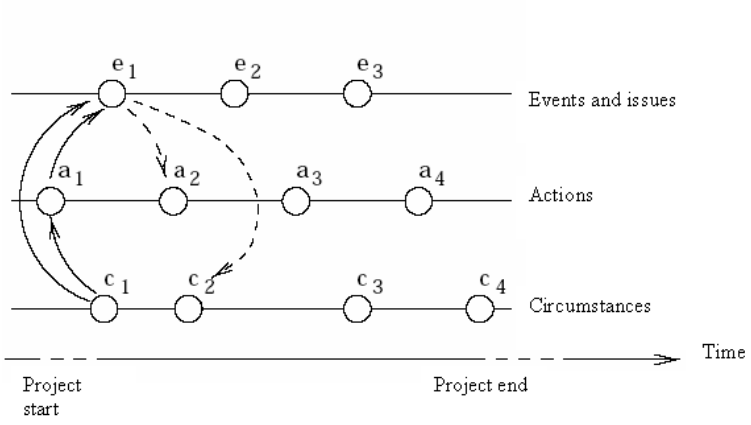

Figure 2.2: Example of a Historical Map [Lanzara & Mathiassen 1985].

from the planned proceeding of the project. What events and issues are described depends on what is interesting to the actors and the researchers.

The actions of the project team members are marked on the middle axis. Actions are the things the project team members actually did to get accustomed to the project circumstances and solve the issues in the project. The lowermost axis is for the circumstances of the project. These include time, staff and monetary resources, available technical solutions and organizational relationships and structures. Events and actions can change these circumstances during the project.

Historical mapping is used at the end of development projects to reflect the progress of the project and to give a consistent and shared view of the project. From that view it is possible to start to examine specific situations inside the project using for example diagnostic maps.

The aim of diagnostic maps is to describe an individual project situation from the point of view of problems and find causes of the problems from organizational and functional context. Diagnostic maps locate and describe problems in a project or organization as seen by actors.

A diagnostic map is formed in four phases (Fig. 2.3). The first phase is to describe what happened. At this stage the actors identify problems in the project relating to the topics they or the researchers are interested in. Of these identified problems the most suitable one is selected for treatment. In the second phase the actors try to come up with theories on what happened in the project situation in question. These theories are related to the cause for the problem and circumstances related to the problem.

Next the consequences of the problem are taken into consideration. The first thing

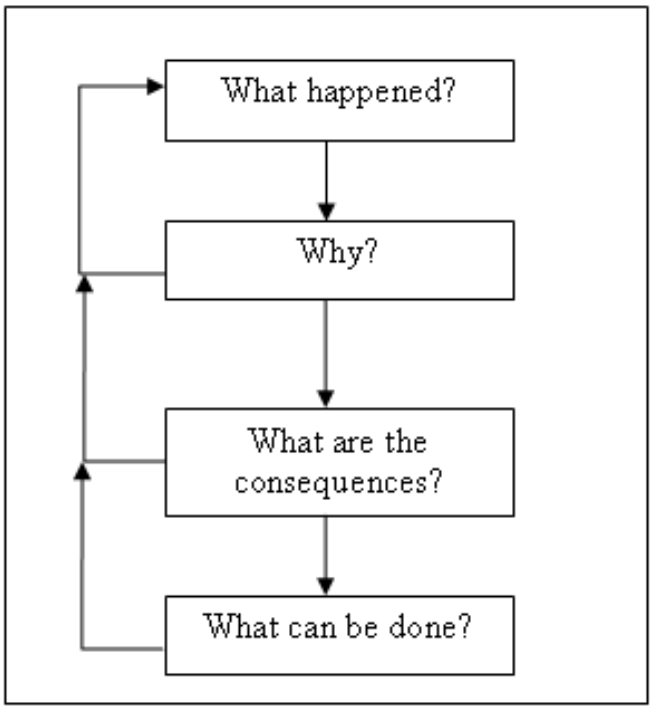

Figure 2.3: Diagnostic Map [Lanzara & Mathiassen 1985].

to sort out is if the problem is serious. If the problem is not considered serious there is no need to continue the examination and another problem can be examined. The second thing to consider is who the problem has an impact on. The final phase of the examination is to plan courses of action to solve the problem. The aim is to look for ways to solve the problem.

Diagnostic maps offer a way to look at specific problems, their cause, consequences and their possible solutions in a project. Their advantage is the structured form the examination of problems takes. This is also its disadvantage as this kind of sequential conception of causality can be artificial and thus not adequately correspond to real-life problems.

The development diaries are used in RSD to support reflection in work [Jepsen et al. 1989]. Using diaries as a method for reflection in work or learning is a wide-spread convention. In RSD the use of diaries to examine and develop information systems development methods is based on the notion that the methods alone do not guide the developers in their work but rather their intuition and experience are important aspects.

The use of development diaries has been found to have the following advantages [Jepsen et al. 1989]:

- It encourages the development of new project practices.
- It enhances the understanding of methodologies used in development.
- It is a quality tool for conceptualizing problem solving situations.
- Diaries can support the evaluation of projects and project situations.
- Writing diaries supports the interaction between design and evaluation tasks.

Most of these can be thought of being mutual for any method of self-reflection in the development process. Writing diaries can be thought by the development team as an extra activity and they do not necessarily see the value in it. Therefore it is important that the subject matters to write in the diary are defined clearly and the overall aim of the diary-writing activity is communicated to the development team. This, as well as systematic writing of diaries ensures that the diary writing is beneficial for the project and the research.

2.4 Experiment Projects

The practical part of the research is made in four learning game development projects. In each of these projects the focus on the research questions of this study varied depending on the following two aspects: 1) the practical goals of the development project in question and 2) the research done in previous projects. The foci of these six projects is summarized in table 2.3. The projects are on the left-most column and the research questions (described in detail in chapter 2.1) examined, along with their experimented solutions are described in the right column for each project. The research questions are also repeated here for reference:

- Q1: What kind of team structure and composition is optimal for learning game development?
- Q2: Into what tasks can the development process be divided? How are those tasks related? How can this be described with process models?
- Q3: What kind of process control model would guide the digital learning game development toward high-quality products?

Table 2.3: The focus on problems of learning game development in the experiment projects

Project	**Experimented solutions for research problems**
Gameli V.1	Q1: Software engineering functional teams in GBLE development Q2: Game development methods adopted to GBLE development, use of game design documents Q3: Use of defined process control model
Gameli V.2	Q1: Subject matter, learning and process experts as producers Q2: GBLE Concept creation with user designers Q3: Use of defined process control model
Social Responsibility Game	Q2: Game idea generation, game development methods, learning analysis in game design, iterative game design Q3: Use of defined process control model
Peatland Adventure	Q1: Flexible project team with strong core Q2: Agile game design with learning and subject matter experts, flexible process Q3: Empirical process control model

- Q4: What kind of metamodel could be constructed to inform the development of game-based learning environments based on the results of the previous research questions?

The methods used in different projects correspond to those described in chapters 3, 4 and 5. In the next four chapters each project is described in more detail along with the development methods used.

2.4.1 Gameli Version One

The aim of the Gameli V.1 project was to develop an engaging learning game on the foundation of the GameWorld modeling and simulation software developed by Centre of Information Technology for Education of the University of Hong Kong. The project took place in Agora Game Lab in the spring of 2005 in collaboration with the software project studies of the Computer Science Department of the University of Jyväskylä.

The Gameli V.1 project team had five members. Each member allocated at least 300 work hours toward the project. Other stakeholders of the project were a class of

elementary school pupils who were the HCD user designers in the project, the project manager from Agora Game Lab as well as three experts whose responsibility was to support the project team. These experts were a subject matter expert, a learning theory expert and a human-centred design expert. The duration of the project was five and a half months.

The project used an iterative modification of the waterfall process model where the design, implementation and testing were done in two iterations. The project also used game concept and design documents to structure the game design task. These documents were used as a basis for requirements specification which in itself consisted of a specification document with prioritized requirements.

A part of this project was completed before this research begun so this project was studied primarily as a case study. The project process was analyzed using historical mapping and the project members were interviewed individually and in group. The interviews' themes were the methods used and other experiences from the project.

2.4.2 Gameli Version Two

The goal of Gameli V.2 project was to expand on the earlier Gameli V.1 prototype and to produce a simulation game environment/simulation game design environment that could be used in classroom in natural sciences education. The project was a collaboration between Agora Game Lab and the software project studies of the Computer Science Department of the University of Jyväskylä. The project started in November 2005 and finished in March 2006.

The Gameli V.2 project team had five members. Each member allocated at least 300 work hours toward the project. Other stakeholders of the project were a group of science teachers and class of elementary school pupils who were the HCD user designers in the project, the subject matter and scientific learning expert in a producer role, the software engineering and game development expert in a producer role and two experts whose responsibility was to support the project team. These experts were a learning theory expert and a human-centred design expert. The duration of the project was five months.

In this project the experience from earlier projects was taken into account when deciding the development process and the methods used. This project used the same kind of process model as Gameli V.1 and Talarius projects: an iterative modification of the waterfall process model where the design, implementation and testing were done in two iterations. The focus however was more in the iterative and prototyping

nature of the model: the requirements phase used rapid prototyping to learn about the product that was being developed and the first implementation iteration was light enough to allow for significant changes after testing. This was done also to allow for more significant role for users in the development process.

The role of researcher in this project was active as a participator. The role could be best described as producer/client. This allowed personal views and reflection from inside the project to be added to the project examination. The project was also examined with a collaborative historical mapping workshop at the end of the project. The project documentation was also analyzed to gather additional material about the experiences of the process and methodologies.

2.4.3 The Social Responsibility Game

The goal of the Social Responsibility Game was to produce a game design for a game that would communicate the social responsibility aspects of a Agora Game Lab's partner organization. The project was a collaboration between Agora Game Lab and the software project studies of the Computer Science Department of the University of Jyväskylä. The project started in November 2005 and finished in March 2006.

The Social Responsibility Game project team had five members. Each member allocated at least 300 work hours toward the project. Other stakeholders of the project were the project manager from Agora Game Lab as well as three experts whose responsibility was to support the project team. These experts were a game development expert, a learning theory expert and a human-centred design expert. The duration of the project was five months.

In this project the focus was solely in game design. This allowed game concept generation and design methods to be taken into close inspection. The overall process model used was a sequential waterfall model, although this time consisting only of the concept generation (or requirements) and design stages. Game design documents were used to document the game concepts generated and the game design work done throughout the project. The methods used in game concept generation included brainstorming about the game's subject matter, harvesting ideas from existing games, using game design patterns as inspiration and creating competing game concepts. The game concepts were evaluated by testers from the target user group of the game as well as expert-evaluated. The game design was based on using the game design document but included also gameplay prototypes and expert evaluation.

The role of researcher in this project was that of an active participator. The role

could be best described as producer/client. As a producer the task of the researcher was to introduce practical knowledge about learning game design methods to the designers and as a client the task was to set the guidelines for the product and evaluate the product in inspections. This allowed personal views and reflection from inside the project to be added to the project examination. The project documentation was also analyzed to gather additional material about the experiences of the process and methodologies.

2.4.4 The Peatland Adventure

The Peatland Adventure is a web-based adventure game designed to support in learning natural science related to the peatland nature. It is designed to be used both in the school context and as a stand-alone game outside school for various user groups. It is a part of the Virtuaalisuo (Virtual Peatland) virtual environment (http://www.virtuaalisuo.fi/). The focus in the design of the game was to make a motivating and interesting game that would provide the players with positive attitudes towards the peatland nature and environment in addition to providing the content-specific knowledge.

This project started at the same time as Gameli V2 project. It used a different approach to the project process: It experimented on a more flexible team structure that started as a small two-people concept design team and expanded when the project proceeded to further phases. The project benefited from a lengthy exploration phase that preceded the project. The exploration phase included brainstorming possible game concepts to develop and testing and re-developing them with user participants.

The development process had features from both sequential and cyclical development model. One thing that made the project unique was the game genre: being an adventure game, the basic game mechanics did not involve a lot of design work. Instead, the level and game story design took most of the game design time. This allowed the team to adopt a kind of cyclical process where the game was developed one level at a time after the overall user interface and game look had been established.

The role of the researcher in this project was that of a consultant. The project had management from elsewhere and the researcher was not part of much of the actual design or development work. However, the researcher did take part in the inspections of the game in various points in development as well as inspecting the project documentation, developer diaries and project process and having informal conversations about the project with the developers.

3 Learning with Games

This first part of the literature review will concentrate on describing games for learning and games in general. The field of games for learning has exploded in the last ten years. Many research reports have been written about the use of games in the classroom (For example [Baltra 1990], [Becta 2001], [McFarlane et al. 2002]), the advantages and drawbacks of using game to support learning (for example [Amory et al. 1998], [Davidson 2004], [Egenfeldt-Nielsen 2004], [Fabricatore 2000], [Gee 2003], [Gee 2005], [Prensky 2001]) and the design of games for learning (for example [Chamberlin 2003], [Druin 1996], [Ravenscroft & Matheson 2002]).

The serious games initiative (www.seriousgames.org) is one driving force that promotes the use of games for learning and training. Many teachers and researchers of learning have also discovered the possibilities of games to support learning. There are many types of learning games out there, and the field of games for learning is too large describe in detail in this thesis. Therefore, we will discuss some examples of learning games in chapter 3.1 to give an idea what types of learning games are most common and promising ones.

Then we will take a look at games in general in chapter 3.2. We will examine different definitions of a game and topology of games and discuss the experience of playing games. In chapter 3.3 we will discuss games and learning in more detail using the games described in this chapter as examples.

3.1 Games for Learning

In this chapter we will introduce some example games that do well to illustrate the varied field of games for learning. These three games have very little in common in addition to their common goal of supporting the learning of some skill or knowledge. In chapter 3.3 we will explore what modes of learning that these games promote.

3.1.1 Example Game: Food Force

Food Force is a educational video game presented by the United Nations World Food Programme (WFP). The project has been developed to support the learning of children

in the matter of world hunger and fighting against it. The game is a Macromedia Director -based one-player computer game for PC and Mac platforms. The game consists of six game modes called as missions. Between the missions the players are shown cut scenes that tell the story of the game and offer the players facts about world hunger. The game's official promotion text describes the game as follows.

"[Food Force] is the first humanitarian educational video game on the subject of world hunger and the work that goes into feeding people. The game is designed for children between 8 and 13 years of age.

The game itself consists of six missions. Each mission begins with a briefing by one of the Food Force characters, who explains the challenge ahead. The player then has to complete the task - in which points are awarded for fast and accurate play and good decision making. Each mission uses a different style of gameplay to appeal to children of all abilities. Each mission represents a key step of the food delivery process - from emergency response through to building long-term food security for a community.

Following each mission a Food Force character returns to present an educational video showing the reality of WFP's work in the field. This allows children to learn and understand how WFP responds to actual food emergencies: Where food originates, the nutritional importance of meals, how food is delivered and how food is used to encourage development." [Food Force 2006]

The game's back story is set in the fictional country of Sheylan, where a crisis has struck. The game is divided into six missions that represent the actions that the World Food Programme does in the real world, despite its fictional setting. The six missions are:

1. Air surveillance
2. Food formulation
3. Food acquisition -– buying and selling food
4. Air drops from helicopters
5. Ground missions involving driving trucks through sometimes hostile territories
6. Future farming -– establishing self-sufficiency

Along with the game, the Food Force website offers guides for teachers to use the game as part of a larger programme in the school class, information about current

situations related to hunger crises in the world and tips on different ways to act locally to support the fight against world hunger. The players can also compare their scores online.

Food Force was chosen as an example game for three reasons. First, some of the minigames feature action gameplay that demands hand-eye coordination and reflexes from the player. Second, the game uses cut-scenes extensively to present educational and background information to the player in between the game playing sequences. And third, the game has a wealth of information on how to use it in a school context including lesson plans and advice on how to integrate the game to other classroom activities.

3.1.2 Example Game: RealGame

RealGame is a business simulation game to support learning of decision making in the environment of managing a manufacturing company. Realgame was developed at Turku School of Economics and Business Administration as a Doctoral Thesis by Timo Lainema (2003). Realgame has mainly been used by large and middle-sized Finnish companies as well as several universities as a part of their management training programs. The following is the RealGame company's own description of the game and its features:

"Realgame is a computer-based interactive business know-how and management training system - a business simulation game - that is being used by leading companies to develop the decision-making capabilities of their middle managers and experts. Realgame business training develops strategic, managerial and operational level capabilities by illustrating interdependence, cause-and-effect and providing an holistic understanding of business processes. Interdependence shows management how different functions, tasks and people work together to create a company result. It shows operational level employees how upper level decisions are connected to operations and the importance of appropriate operational and strategic responses to market situations. Cause-and-effect is vital because in new organization structures, decisions are made by those with local operational knowledge. They need to become comfortable with making informed decisions without perfect information and know the outcome of their decisions. Holistic understanding is the result of knowing the causal interdependencies of company processes." (www.realgame.fi)

The RealGame is a business simulation game describing the main processes of a manufacturing company. The essential idea of the game is to position the participants

in a role where they have to manage a manufacturing business. This is should be done in a profitable manner to keep the company alive. The companies are in continuous information exchange with their customers, suppliers, and indirectly with their competitors. The game company represents a total enterprise model of a manufacturing company in which decisions from one functional area interact with those made in other areas of the company. [Lainema 2003a]

The intended audience of the game varies. The game is suitable for participants from middle management position to foremen of production, and to business students. As the game clock speed and game complexity are variable, these parameters can be adjusted according to the audience. The participants adopt roles of decision-makers. They have to make decisions on production, supplies, sales, marketing, investments, transport, and so on.

As the game is complex, running a game training requires at least eight hours. Preferably the game session should last one and a half or two days to obtain all the benefits of the game. The required time depends of course on the selected complexity of the game model and the speed of the game clock. The participants, after having learned the rules of the game, are mostly free to manage their companies. Thus, the facilitator intervenes only if the participants wish to have additional guidance. [Lainema 2003a]

This game is presented as an example of a game for learning as an example of many simulation games for learning. The multi-player team competitive gameplay is also an exemplary element of the game.

3.1.3 Example Game: Revolution

Revolution is a multiplayer 3D game for learning developed by MIT in co-operation with colonial Williamsburg. Revolution has been developed as a game modification (aka. mod) on the basis of Neverwinter Nights engine, a popular computer roleplaying game engine. Here is the developers' own description of the game:

In the late 18th century, tensions between the British Empire and its colonies on the eastern coast of North America were reaching a critical point. From Georgia to Massachusetts talk of revolt hung heavy in the air, and the threat of war lingered on the horizon. The colonists were about to suffer the bloody birth of a nation, a nation that would eventually shape the course of human history. But this couldn't be known by the people of the time, who went about their everyday lives much as anyone had through the ages – one day at a time, their only goal to make their way through the world into which they were born.

Revolution is the Education Arcade's multi-player, American Revolution-themed role-playing game based on historical events in the town of colonial Williamsburg. Set in 1775, on the eve of violent revolt in the colony of Virginia, the game gives students an opportunity to experience the daily social, economic, and political lives of the town's inhabitants. By allowing role-play from one of seven social perspectives – from an upper class lawyer, to a patriotic blacksmith, to an African American house slave – Revolution places students in a situated learning context. (www.educationarcade.org)

The game offers a fundamentally new kind of experience compared to books or film. The player is at once an actor within the virtual drama capable of influencing events and a spectator watching the drama unfold around their character. This creates a level of immersion that can produce high levels of engagement as students explore the virtual world led by their own inclinations and curiosities within the multimodal interactive game text. Students start learning about multiple aspects of life in colonial Williamsburg from the moment they enter the game world. (www.educationarcade.org)

Revolution asks students to make hard decisions about the future of their town and even the whole colony. The game also teaches students an "ordinary" experience of history that includes passionate rhetoric and heroic battle, but also economic frustration, political indifference, and the mundane of everyday life. (www.educationarcade.org)

This game is a quality example of games for learning as the role-playing aspect of the game is essential in it. The pleyers' roles as the people of colonial Williamsburg and the environment of the town itself offer the possibility to explore cultural history in first person perspective. Other important factors include the evolving game story, which is rigid in the setting and set-up still letting the conclusion of the story in the players' hands and the multiplayer element that does not lead to any rigid competitive or collaborative game modes as such.

3.2 The Properties of Digital Games

In this chapter we will turn the attention toward the notion of digital games itself. As noted in the introduction (Chapter 1.3) Game-playing is an ancient human activity and nowadays game-playing has taken many forms in the society. Research on games, on the other hand, has not really been widely spread until the recent years. Some academics like Huizinga [Huizinga 1955] and Callois [Caillois 1961] have studied games from a cultural perspective during the 20th century, but most progress in this field has come in the turn of the millennium. At the same time, game designers have started

gathering body of knowledge about their industry and work techniques in game design (for example [Costikyan 1994], [Church 1999], [Ryan 1999a], [Ryan 1999b]).

Many articles and books have been written in attempt to define games. Nowadays the consensus seems to be that all-encompassing definition of a game that does not leave out any examples of games may be impossible to attain. Different definitions have their strong and weak points and examining the different definitions can help to understand the domain of games. Salen & Zimmermann (2003) have compiled a collection of definitions of what a game is in their game design handbook Rules of Play. In this thesis we will approach the essence of games by examining different definitions of games and identify essential properties of games (table 3.1) and essential features of game-playing activity (table 3.2). In addition to those definitions mentioned in [Salen & Zimmerman 2003], more recent definitions by Ernest Adams (2005) and Jesper Juul (2005) have been examined.

What most of those definitions have in common is that games are formal meaning that they are restricted by unambiguous rules, contain goals that players try to achieve and are interactive. Game-playing is an active endeavor involving decision-making, uncertainty of outcome as well as conflict and/or competition. The gameplay activity is somewhat distanced from reality. The distance from reality can take many forms, such as the safety for players, make-believe used in the game or suspension of disbelief. This general description is used in this study as the definition of a game.

Another way to answer the question 'what are games' is trying to to find other fitting definitions of broader concepts. The concept of system is one that fits well: all games can be understood as systems ([Järvinen 2007], [Salen & Zimmerman 2003]). System is defined as a group of interacting, interrelated, or interdependent elements forming a complex whole [Salen & Zimmerman 2003]. We will analyze the elements comprising games later in this chapter. At this point it is sufficient to say that games can be, at the same time, thought of as multiple different systems.

Let us concentrate, then, on the latter question: what are games and the game-play experience made of. This is actually two questions: 1) What are games as artefacts like and 2) what is the game-play experience like. In this we combine the game-play schema definitions introduced by Salen & Zimmerman (2003) as well as three levels of gameness introduced by Hunicke et al. (2003). This gives us a three levels of gameplay activity: mechanics, play and culture. Mechanics form the description of games as artefacts: the formal intrinsic structure of games. It includes all the game elements and their behavior as well as the rules that the players must follow in playing the game.

Table 3.1: Properties of Games

Property	**Characteristics**	**References**
Formality	Agreed-upon rules	[Parlett 1999] [Suits 1990] [Avedon & Sutton-Smith 1971]
Limiting context	Boundaries, social group	[Abt 1970] [Huizinga 1955]
Make-believe	Game as fiction	[Caillois 1961] [Adams 2005]
Uncertainty	Outcome not known beforehand	[Caillois 1961]
Representation	Game is a representation of real or fictional activity	[Crawford 1982]
Interaction	Actions and feedback	[Crawford 1982] [Adams 2005]
Art	Game as work of art	[Costikyan 1994]
Game tokens	Game objects	[Costikyan 1994]
Goal-oriented	Value assined to outcomes, set objectives	[Parlett 1999] [Abt 1970] [Avedon & Sutton-Smith 1971] [Costikyan 1994] [Juul 2005]
Entertainment	The purpose of the game is to entertain players	[Adams 2005]
Competition	Competing against other players	[Parlett 1999] [Avedon & Sutton-Smith 1971]

Table 3.2: Features of Game Playing Activity

Property	**Characteristics**	**References**
Activity	Active effort	[Abt 1970] [Huizinga 1955] [Caillois 1961] [Juul 2005]
Decision-making	Meaningful choices	[Abt 1970] [Costikyan 1994]
Outside ordinary life	Not part of everyday life	[Huizinga 1955]
Immersive	Absorbing and motivating	[Huizinga 1955] [Adams 2005]
Freedom	Freedom within the boundaries	[Huizinga 1955] [Caillois 1961] [Juul 2005]
Unproductive	Does not have external consequences	[Caillois 1961] [Suits 1990] [Juul 2005]
Conflict	Challenges for the players to overcome	[Crawford 1982] [Avedon & Sutton-Smith 1971]
Safety	Safe environment to experiment	[Crawford 1982] [Adams 2005]
Resource management	Managing limited resources to overcome challenges	[Costikyan 1994]
Creative Play	Creativity through gameplay	[Adams 2005]

The play level discusses the game-play experience: the interaction between players and the game mechanics. Culture is a contextual schema which highlights the cultural contexts into which games are embedded [Salen & Zimmerman 2003]. This level goes beyond the question asked in the start of this paragraph but is still a related matter. This division into three levels is in line of the systemic view of games mentioned earlier [Salen & Zimmerman 2003]. In the next chapter (3.2.1) we will concentrate on the mechanics level: the intrinsic structure of games. Chapter 3.2.2 will discuss the level of play, or the game playing experience. The culture schema is considered to be mainly outside of the scope of this thesis.

3.2.1 The Game as an Artefact

Mechanics, including game rules, define the form of game and its inner structure [Salen & Zimmerman 2003]. Games as an activity include also rules about strategy, etiquette and so on. These are not part of the game mechanics that are an intrinsic part of the game [Salen & Zimmerman 2003]. Rules are repeatable, binding, fixed, explicit and umambiguous and they limit player actions [Salen & Zimmerman 2003]. Salen & Zimmerman (2003) divide the formal rules into three categories: operational rules, constituative rules and implicit rules. Operational rules are the guidelines players require in order to play. They are usually more or less synonymous with the written game rules (in the case of board and card games). Constituative rules are the underlying formal structures that exist below the surface of the rules presented to the player. These structures are logical and mathematical. Implicit rules are the "unwritten rules" of a game. They may discuss etiquette, sportmanship or common functionality of a computer user interface.

Aki Järvinen (2003) offers a typology of game rules that is based on describing and categorizing different elements the game consist of. Most of the elements Järvinen describes in his article relate to the mechanics level of a game. Rules need to be assigned to actions the players are supposed to take, tools used in the process, and the means that the game-system treats player behaviour with [Järvinen 2003]. In a game the ruleset defines *procedures* in relation to *game components* within the *game environment* with which the actions are produced. These three elements belong to the formal schema of games. Järvinen also includes the *theme* of the game and the *game interface* to *game elements*.

Game components are usually represented by objects, or a single object, that the player can manipulate in the course of the game. In digital games objects usually take

the following forms: a character or a group of characters, a vehicle, a game piece, a tool or a resource. [Järvinen 2003]

The object(s) that the player manipulates are called the player-objects. This might be the player's avatar or a collection of game pieces belonging to the player. When the player-object is represented by a character or simulates the behaviour of one it is relevant to call it a player-character. Other components relate directly or indirectly to players' actions. These include antagonists, co-operators, systems, resources and props. [Järvinen 2003]

All components have rules governing their behaviour. Usually in games the components can be ordered according to their significance in game play. That allows for categorization of game components to core and marginal components. Individual components may change from one category to another as the game progresses.

The function of components, according to Järvinen (2003), is to provide a source of identification for the player as well as to provide the player with challenges in the form of adversaries, obstacles, resources to be had etc. Game-objects are the reference point of the player's needs and desires. Players are encouraged, or enforced, via components and environment constraints to play the game in a specified, rule-bound way.

Procedures are operations that the game-system makes possible for 1) empowering players with means to play the game, 2) assigning value to the different game states and outcomes by handing out penalties and rewards and 3) governing the interrelations of components. Any action by the player or the game system constitutes a procedure. Procedures combined with other gameplay phenomena form game *mechanics*.

Game environments provide the space for components and procedures. Game components reside within the game environment and procedures and mechanics are enacted in relation to it. While a board game's game environment is the game board a digital game's environment can be a virtual space or a virtual world.

Björk & Holopainen have presented their own typology of games: an activity-based framework for describing games. [Björk & Holopainen 2005] The framework presents a categorization of physical and logical game components (Fig. 3.1). As Järvinen's framework, some of the categories are outside the schema of rules. Most components of the 'holistic' category belong to the Play schema.

Attach:gamecompframework.png

The framework divides components into four categories: holistic, boundary, temporal and structural. These categories reflect four different ways of viewing the activity of playing a game. The categories can be roughly classified by their individual level of

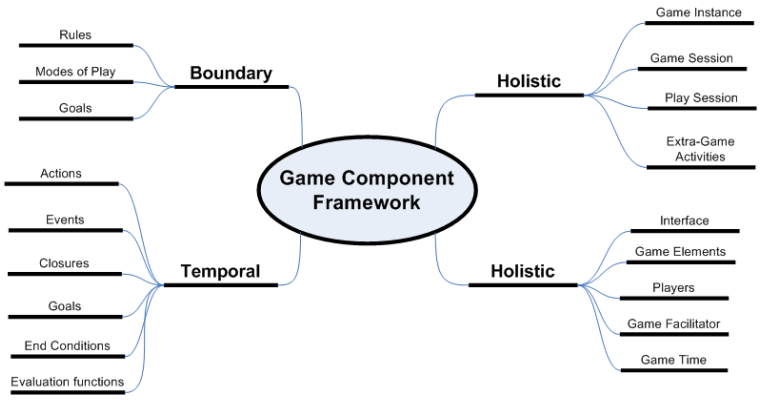

Figure 3.1: An activity-based framework for describing games [Björk & Holopainen 2005].

abstraction, with the more abstract categories building upon the more concrete ones.

Most concrete of these categories is the structural category. It includes all the physical and logical elements necessary for containing and manipulating the game state. This includes game elements which is comparable to the game components category of Järvinen's topology, allthough it includes the game environment as well. The next category in terms of concreteness is the temporal category. It contains all the events and actions that can happen during the course of gameplay. The boundary category deals with the boundaries of gameplay and the holistic category uses all the underlying categories to form a whole picture of the gameplay instances, sessions and so on.

3.2.2 The Game Playing Experience

In the previous chapter we discussed games from a formal standpoint. Now it is time to concentrate on the game playing experience: What is the activity of playing a game like. When looking at the definitions, we decided on some basic consepts: game-playing is an activity, which means it is active (doing), it is rule-bound, it is goal-oriented and it is distanced from reality.

Hunicke et al. present the MDA (Models, Dynamics, Aesthetics) framework to help analyzing and designing games. It illustrates the connections between the levels of Play and Mechanics during the game design and game playing processes. The consumption of games is formalized in the framework by dividing it to the three mentioned components. Mechanics describe the actual components of the game on the level of data representation and algorithms. This corresponds to the mechanics level

of games described in the earlier chapter. Dynamics describe the run-time behaviour of the mechanics or how the mechanics react to the players actions and how those reactions follow each other. Aesthetics describe the emotional reactions that are desired to rise when the player interacts with the game system. So, together the dynamics and aesthetics levels correspond to play. The dynamics component lies between the levels of mechanics and aesthetics and corresponds to the interplay of these two levels. The framework portrays the game more as an artefact than a medium. In the MDA framework, designer and player approach game from different viewpoints, the player from the viewpoint of aesthetics and the designer from the viewpoint of mechanics (Fig. 3.2). Player approaches the game from the aesthetics component of the model whereas mechanics are the starting point of the designer. Designer's task is to enable some kinds of player experience on the player of the game that he is designing by designing the mechanics of the game so that the desired player experience emerges from the mechanics and dynamics designed. [Hunicke et al. 2003]

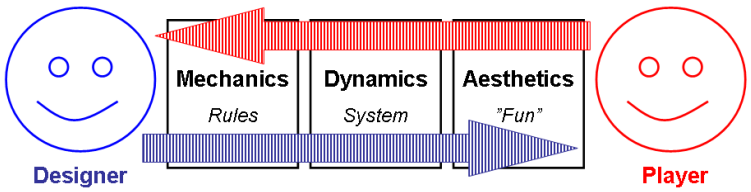

Figure 3.2: The player's and designer's viewpoints in the MDA framework [Hunicke et al. 2003].

Hunicke et al. also constructed their taxonomy of game aesthetics. This is a collection of different kinds of emotional rewards the players seek in playing different kinds of games. Now-defunct game developer Ion Storm added to this list to make up their own terminology on different kinds of fun in game playing. [Hunicke et al. 2003] The Ion Storm list of types of fun in game playing consists of seventeen identified types of rewarding game experience [Juul 2006]. Perusing these two sources, we will construct our own taxonomy of game aethtetics (Fig. 3.3). The collection shows also possible the relations between different kinds of aesthetic rewards related to game-play.

First of the aesthetics, *goal-completion*, can be derived straight from the definition of games. Most games have clear goals that can be accomplished by the means awarded to the player. The accomplishment is easily recognize by the player [Juul 2006]. An example of this is completing a mission in Food Force (for instance, performing the

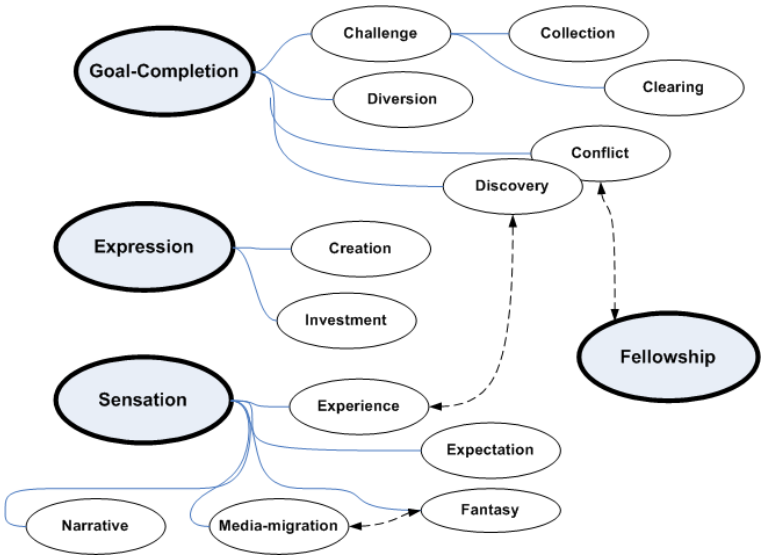

Figure 3.3: A Taxonomy of Game Aesthetics [Hunicke et al. 2003] [Juul 2006].

air drops successfully) or successfully selling making a profitable deal of manufactured goods in RealGame.

There are many kinds of *goal-completion* activities present in games. Many games have *challenges* that make accomplishing goals uncertain. *Challenges* and *obstacles* that the game itself provides offer a possibility of failure and thus make success more valuable. Timing the air drop just right to make it land correctly in Food Force's air drop mission is an example of this kind of *challenge*. There are two specific kinds of goal-completion/challenge combinations that are typical of digital games: *collection* and *clearing* (Ion StormX). *Collection* is the act of accumulating a things, sometimes tied to the desire of collecting a set. Card collecting games, such as Magic: The Gathering, are great examples of this. *Clearing* means cleaning up the game space (or part of it) from a scattering of interactive elements [Juul 2006]. Spotting and marking all the refugee camps from the helicopter in the first mission of Food´Force is a good example of this.

The third "sub-aesthetic" related to challenge is *discovery*, which means space to explore and gain mastery over [Hunicke et al. 2003]. Usually this is actual space, for instance an enemy territory to be charted (and conquered), but sometimes conceptual space, like the rules to a new game. [Juul 2006]

Another kind of game aesthetic that is related to goal-completion is *conflict*. Conflict arises naturally from the interaction in a game. The player has goals he is trying

to accomplish and obstacles and potentially other players' intentions prevent him from achieving them. [Crawford 1982] In games the conflict is staged: when players agree to the game's rules and victory conditions they agree to compete in achieving the victory. Conflict can be individual or team-based, cooperative or non-cooperative, direct or indirect [Adams 2005]. Many games mix and match different forms of conflict. Competition and conflict are essential elements of a game. Without the sense of competition, players could not measure the extent of their progress or success in the game. [Salen & Zimmerman 2003]

Diversion is related to goal-completion: it is the pleasure derived from performing routine game system activities; the mechanical act of manipulating the game [Juul 2006], usually related to overcoming the challenges in the way of accomplishing the goals of the game. A good example of a game that relies on the diversion aesthetic is solitaire.

Another major vein besides goal-completion in the taxonomy of game aesthetics is *expression*, games as self-discovery [Hunicke et al. 2003]. This is in effect when player expresses him/herself through his/her actions and choices in the game. An example of this is choosing a character nick name in a game like Quake or EverQuest or choosing a character's appearance and abilities in a computer role-playing game.

Creation is closely related to *expression*. It is the act of building something that feels like it belongs to you in a game [Juul 2006]. For example, building yoour own city in SimCity or an apartment for your Sim in Sims. *Investment* is spending time with a game element and thus coming to value it. For instance, investing time on your character on a role-playing game so that the character grows in in-game power.

The third "sub-category" of game aesthetics relate to experiencing something as a player. *Sensation* is game as sense-pleasure [Hunicke et al. 2003], aurally or visually pleasing aesthetics [Juul 2006]. For example, when the player first experiences the high visual and aural quality of the cut scenes of Food Force. Fantasy is games as make-believe [Hunicke et al. 2003], vehicle for imaginative or impossible activity [Juul 2006]. An example of this is the player of a role-playing game casting spells and fighting dragons.

But the players experiences do not have to be make-believe or fantasy in order for them to be enjoyable. *Experiences* that are merely beyond the boundary of the player's everyday life are sufficient to create enjoyment in a game. For example, piloting a helicopter in a hunger-stricken third world country in Food Force or managing a manufacturing company in RealGame are something that the player might want to

experience but probably will not be able to because of the constraints in real life.

Media-migration can be an important part of the make-believe or outside the real life experiences that the game offers. *Media-migration* is the players' desire to interact with familiar (and often well-liked) fictional elements from other media in ehich the important elements are familiarity with the established fiction and interaction [Juul 2006].

In addition to those, players tend to enjoy experiencing and interacting with *narratives*. By narratives in this sense we mean drama that unfolds over time, creating tension and engaging the players' attention. A good example of this is the events of colonial Williamsburg in the game Revolution as the fictional days events progress to their eventual solution (which side will the different characters take, what will be the resolution of the conflict).

Finally, but certainly not of least significance, there is the aesthetic of *fellowship*, or games as a scoial framework. This can mean a lot of things, from the game itself providing a diversion-like activity to make the social event more fluent and providing players an activity to follow as they learn to know each other, to co-operative games where the plyers work together to active common goals to competitive games where the players play against each other.

To illuminate the game-playing experience we will take another look at the notion of games as systems introduced in chapter 3.2.1. We will use four of systemic schemas describing gameplay that Salen & Zimmerman (2004) introduced: games as emergent systems, games as systems of uncertainty, games as systems of information and games as systems of conflict.

There are many examples of complex systems, including anthills, human economies and computer systems. They do not seem to have very much in common in regards to their topology but all complex systems are held to have behavioral and structural features in common, which at least to some degree unites them as phenomena. Not all systems are complex, there is a "complexity barrier" that only some systems reach ([Salen & Zimmerman 2003], from Campbell). In regards of this discussion the conceptual division to simple and complex systems suffices. Complex systems are a result of emergence. Emergence appears when a simple set of rules applied to a limited set of objects in a system leads to an unpredictable result. This unpredictability, however, is not the same as chaos. In emergent systems unpredictable results form coherent shapes that are meaningful. Chaotic systems are totally unpredictable and do not form meaningful shapes other than by accident. When system is not emergent, it is predictable. Predictability makes uncertainty impossible. Predictable games are not

aesthetic according to Hunicke et al. (2004) and not fun according to Ion Storm's list of types of fun in games [Juul 2006]. Games that are emergent systems enable players to explore the possibility space of the game to find different kinds of emergent behavior in the game system [Salen & Zimmerman 2003].

This leads to games as systems of uncertainty. Uncertainty is a part of every game. Games are uncertain in two levels: in the macro level of achieving goals and determining the outcome of the game and in the microlevel relating to the outcomes of singular actions within the game. All games have uncertainty in the macro level, but not all of games have that in the microlevel. [Salen & Zimmerman 2003] There are many sources of uncertainty: randomness, other players' actions and other actors in the game environment such as artificial intelligence-controlled game characters or rule-bound game objects. Salen & Zimmerman group outcomes in three categories according to their certainty: certain, uncertain and risk, as follows: "A certain outcome is completely predetermined. A risk is an outcome with a known probability of happening. An uncertain outcome is completely unknown to the player." [Salen & Zimmerman 2003] Most games combine some degree of these all. Different game types use different amount of uncertainty and that leads to different kinds of gameplay experience.

Games use game components or game elements to inform the player about the game state [Järvinen 2007]. In the case of digital games, the game user interface is the player's source of information. Some games offer all the information there is to the players. These are called games of perfect information. In the games of imperfect information some of the information is hidden from some or all the players. Games of perfect information tend to be analytically competitive, whereas games of imperfect information tend to have more uncertainty. [Salen & Zimmerman 2003]

According to Salen & Zimmerman (2003) there are four kinds of information in the game: Information known to all players, information only one player knows, information only the game knows and randomly generated information. In addition to that there is the division of information to objective information and perceived information. Objective information exists only in game system's internal information structure and perceived information is the information the player observes and interprets during play. [Salen & Zimmerman 2003]Information acquisition can be an important part of gameplay: for example in poker getting hints about other players' hands is often attempted by interpreting their physical gestures. Digital games in particular can make the discovery of hidden game rules and other information important part of the gameplay experience [Salen & Zimmerman 2003].

Table 3.3: Different kinds of challenges in a game. [Rollins & Adams 2003]

Challenge	**Description**
Challenge	**Related Player Activity**
Logic & inference	Assimilating information, deciding the course of action
Lateral thinking	Drawing on previous experiences, combining them in new ways
Memory	Recalling previous game events
Intelligence-based	Purely logical operations & deductions, similar to those in IQ test
Knowledge-based	Tests the knowledge of the player inside or outside the game fiction
Pattern-recognition	Comparing new experiences to the archetypes and categories formed by the brain
Moral	Making moral choices
Spatial-awareness	Navigation and perception in different spaces
Coordination	Performing many simultaneous actions
Reflex / reaction time	Timing actions
Physical	Physical exertion, e.g. strength or stamina

Conflict arises naturally from the interaction in a game. The player has goals he is trying to accomplish and obstacles and potentially other players' intentions prevent him from achieving them. [Crawford 1982] In games the conflict is staged: when players agree to the game's rules and victory conditions they agree to compete in achieving the victory. Conflict can be individual or team-based, cooperative or non-cooperative, direct or indirect [Adams 2005]. Many games mix and match different forms of conflict. Competition and conflict are essential elements of a game. Without the sense of competition, players could not measure the extent of their progress or success in the game. [Salen & Zimmerman 2003]

Challenges are instrumental in gameplay and are related to conflicts. A game's challenges are what players must overcome in order to be successful in a game. There are many kinds of challenges. Rollins & Adams categorize different kinds of challenges found in a game according to what kind of skill is required to overcome them (table 3.3). The type of challenge defines also the main activity of the player [Adams 2005].

Logic and inference challenges are typical in management-type of games e.g. Re-

alGame (see 3.1.2). Player tries to collect information about the situation and use that information to decide upon the best course of action [Rollins & Adams 2003]. Lateral-thinking challenges are an extension of inference challenges where the player must apply the information and previously learned courses of action in a new way.

Memory challenges require the player to recall a piece of information related to previous stage of game. Traditional memory games using pairs of identical cards is one example of a game relying on memory challenges. Intelligence-based challenges rely purely on the intelligence quotient of the player. This is extremely difficult to quantify and define, and, it is argued intelligence-based challenges do not exist in their pure form in games [Rollins & Adams 2003]).

Knowledge-based challenges can either be intrinsic, meaning that they require knowledge inside the game fiction, or extrinsic, meaning that they require knowledge from the outside world ([Rollins & Adams 2003]. Intrinsic knowledge challenges can thus also be categorized as memory challenges as playing the game is the only way to gain knowledge about the game fiction.

Pattern-recognition challenges utilize the human brain's natural ability to function as a pattern-recognition machine. The human brain implicitly forms archetypes of objects and events and compares new experiences with these archetypes to recognize which category they fall under [Rollins & Adams 2003]. These kinds of challenges can take many forms from the simple visual recognition of similar shapes to form groups or something more complex such as forming patterns in artificial intelligence controlled game entities' behavior and using that knowledge to overcome them.

A moral challenge is a high-level challenge that can operate at several levels: universal, cultural, tribal, and personal. In many games, the player is asked to make moral choices. In order for the choice to be moral, there has to be some conflict between the options on some levels of moral. [Rollins & Adams 2003]

Spatial-awareness challenegs involve navigating two- or three-dimensional spaces and/or assessing the dimensions of those spaces. Spatial-awareness challenges are a specialized hybrid of a memory challenge and an inference challenge [Rollins & Adams 2003]. Coordination challenges are present in most games. Coordination challenges basically test the ability of the player to perform many simultaneous actions [Rollins & Adams 2003]. For example in Food Force (see 3.1.1), in the first mission, the player has to control the flight of the helicopter whilst at the same time keeping an eye out for refugee camps and examining the map window so that she can scan the whole area within the time limit.

Coordination challenges are almost always found in combination with reflex/reaction time challenges and are usually tightly coupled with them. Reflex/reaction time challenges test the timing abilities of the player. [Rollins & Adams 2003] An example of reflex challenges is the third mission of Food Force (see 3.1.1) where the player has to time the food packet drops from an aeroplane so that they hit the targeted runway.

Physical activities are rare in games as the input methods available for computer games do not lend themselves to physical activity - at least, not without the purchase of specialized hardware [Rollins & Adams 2003]. In arcades there have been some games testing the players' strength such as arm wrestling games.

Different types of challenges can be combined in one game in a single gameplay mode or by providing different gameplay modes within a single game. A game's conflict types, challenges and allowed player actions are all interrelated and together with game's presentation provide a gameplay experience unique to the type of these features. These also affect what kind of learning the game facilitates. The next chapter discusses how games can support learning.

3.3 Learning Through Games

Now that we have discussed games in general its time to take a closer look on how games could support learning. As the focus of this thesis is on developing methods and techniques for designing games for learning and not on theories on learning applied to digital games, the discussion will be limited to examples and some general statements. First we will take a look at which features of digital games make them good artefacts to support learning and then we will take a look at how to design a game for learning.

Games are commonly though to be only for fun ([Smith 2003], [Gee 2005]). But, as described in chapter 3.2.2 the concept of fun can be more complex than most people normally think. Gee (2005) portrays playing a game as an activity that takes a long time and requires commitment as well as concentration. He sees that as the main reason to examine games to support learning. Motivation is also seen by other authors as one of the key advantages of games for learning (Lähteet Elinan selvityksestä). Gee has made an extensive list of other features of games that are good for learning (table 3.4). We will discuss them in context of our example games.

All of the example games take advantage of the *identity* a player assumes when he or she plays the game. In Food Force the player is cast into the role of humanitarian aid worker, in RealGame the player becomes the head of the company and in Revolution

Table 3.4: Features of games that are good for learning. [Gee 2005]

Feature	**Description**
Identity	No deep learning takes place unless learners make an extended commitment of self for the long haul. Learning a new domain, whether it be physics or furniture making, requires the learner to take on a new identity. Good video games capture players through identity. Players either inherit a strongly formed and appealing character or they get to build a character from the ground up.
Interaction	Plato in the Phaedrus famously complained that books were passive in the sense that you cannot get them to talk back to you in a real dialog the way a person can in a face-to-face encounter. Games do talk back. In fact, nothing happens until a player acts and makes decisions. Then the game reacts back, giving the player feedback and new problems.
Production	Players are producers, not just consumers; they are "writers" not just "readers". Even at the simplest level, players co-design games by the actions they take and the decisions they make. An open-ended game like Elder Scrolls III: Morrowind is, by the end, a different game for each player.
Risk Taking	Good video games lower the consequences of failure; players can start from the last saved game when they fail. Players are thereby encouraged to take risks, explore, and try new things.
Customization	Players can usually, in one way or another, customize a game to fit with their learning and playing styles. Games often have different difficulty levels and many good games allow players to solve problems in different ways.
Agency	Thanks to all the preceding principles, players feel a real sense of agency and control. They have a real sense of ownership over what they are doing.
Well-Order Problems	In good video games, the problems players face are ordered so that the earlier ones are well built to lead players to form hypotheses that work well for later, harder problems. It matters how the problem space is organized - that's why games have "levels"[47]
Challenge and Consolidation	Good games offer players a set of challenging problems and then let them solve these problems until they have virtually routinized or automatized their solutions. Then the game

the player is cast as one of the residents of colonial Williamsburg. These identities help player relate to the context of the games and the matters discussed in the games.

Interaction is also an integral part of all these games, yet in a slightly different way. Food Force's action game modes provide feedback for the player's actions in real time. In RealGame, the impact of player's actions can be revealed instantly or later on in the simulation. In Revolution the player is interacting not only with the game, but with other players through her avatar. The same is true on some level in RealGame: the actions of players have impact on other players. This constant feedback that interaction provides helps players to distinguish profitable actions from non-profitable ones and learn to succeed in the tasks of the game.

Production is in effect in Revolution and to a lesser effect in RealGame. As the player's are responsible for all the important Decisions in Revolution, the plot of the game as it happens in each game session is truly a product of the players' co-operation. In RealGame, the fortunes of the virtual companies are in the hands of the players.

All of the games reduce the effect of risks taken in the game. In Revolution the players can, for example, experience being a slave without having to stay oppressed for the rest of their lives. The lessened risks afford the players to explore different roles in a society. In RealGame the players can experiment with running a company without actually losing any money. Food Force, however, is the only example game in which risk taking is lowered also in the game. The player can fail as many times as he or she needs to in order to learn the level. She may try the level again after a failure.

The example games do not have much in the way of customization. The option to customize RealGame for different kinds of business areas is the best example of this. Food Force and RealGame take advantage of challenge: Food Force's game modes start from the easiest and get a little more complex in every new level. In RealGame, the challenge of other players gets tougher as the other players get better at playing the game.

Food Force makes good use of "just in time" information. Game uses other Food Force workers voices to present hints and other information during the game modes and the cut-scenes provide information about the next level to be played while at the same time providing information about the topic of hunger and humanitarian aid. The information about the game is provided "just in time" and the information about the important topics are layered between the in-game information.

All of the games take advantage of situated meanings. Food Force and especially Revolution have strong and immersive game environments that provide context for all

the important information about the topics at hand, namely the historical situation and cultural history in Revolution and world hunger in Food Force. RealGame takes the terminology of business in to action as the player has to cope with it to be able to run his business.

RealGame is basically based on the notion of system thinking. The backend of the game is a business simulation system, which works according to rules and relationships and not on isolated events or facts. The player must master this system in order to succeed in the game.

Revolution is a good example of smart tools and distributed knowledge. The player's avatar in fact holds much of the cultural information the player needs to act correctly in the colonial Williamsburg. The avatar's clothing and equipment, as well as her skills and social status restrict the actions the player is able to make. This ensures that the player stays within the context the games of cultural and historical environment. The actions available to the player via her character can tell a lot about the time to the player.

All of these games use the notion of performance before competence. The player is not required to know anything about humanitarian aid or hunger to begin playing Food Force; she receives all the information needed while playing. The same goes for the other games. In RealGame, real world business knowledge can help, but player can try to cope before acquiring it.

As different games meant for learning of different kinds of skills and subjects, these games make use of different kinds of features of games. But how would one start a games for learning design project about a particular subject and how would one choose a fitting set of features to support learning? In this thesis, we will not be able to make a conclusive answer that is based on contemporary learning theories, so a practical answer is given instead. As Gee (2005) noted, games are activity and their main advantage in support of learning is that they provide a meaningful motivation to enable commitment and concentration on an activity. It is then natural that the activity of the game should be designed to support the learning of the skills and subject matters that are the learning goals intended for the game.

So, the first task in designing a game for learning is to determine the skills and subject matters that are the learning goals of the game. The second task is to come up with the kinds of game activity that enforces the learning of those skills and/or subject matters. The types of challenges found in games listed in chapter 3.2.2 can be of help in this task, as the main activity of the game is related to the challenges. After these

two steps the game design process begins.

3.4 Learning Game Production in Learning Sciences and Instructional Design

Learning sciences (LS) and instructional systems design (ISD) are two related fields that have shared interests in the application of technology for advancing human learning [Kirby et al. 2005]. These two fields have different values, boundaries, and in some cases methods, they also share significant overlap of content and purpose. Both study learning and technology: ISD is primarily concerned with the design of materials for learning while LS is more closely related to cognitive sciences and promotes the scientific understanding of learning aided by technology where applicable. As learning games are technology to facilitate learning we will discuss the application of both of these fields in learning game production. Although these fields are traditionally seen as far apart from each other, it is seen as practical to try to apply both of their knowledge as there has been increasing interest within both of these fields in the application of the knowledge base of the other field (for example [Kirby et al. 2005], [Cronjé 2005]). One of the reasons for this is the increasingly related nature of scientific understanding or modeling of learning and design-oriented work that aims to change learning in some way (Design-Based Research Collective, 2003; Hoadley, 2002).

The practitioners of Learning Sciences are interested in collecting "theories of active, constructivist, and participatory learning to design software and learning environments and ways of educating that promote deep and lasting learning" [Kolodner 2004]. When this pursuit involves blending empirical educational research with the theory-driven design of learning environments, a set of methods known as design-based research are in use. Design-based research is conducted to create and extend knowledge about developing, enacting and sustaining innovative learning environments. Design-based research goes beyond merely implementing and testing learning environments. It strives to understand the relationships between learning theory, designed artefacts and practice. [Design-Based Research Collective 2003] In this study developmental research is used as the primary design-based research methodology and additional sources are cited when and where they add to the knowledge base of developmental research.

Instruction is the intended activity to promote learning, that is, the acquisition of knowledge, skills and attitudes [Dijkstra 1997]. Therefore, learning games can be considered a method of instruction. instructional design comprises the ideas, plans

and rules of what has to be done in order to develop the actual instruction, that is the artefacts and activities to promote learning and reach a learning outcome that is described in advance of the instruction [Dijkstra 1997]. From this standpoint we can discuss the learning game production process in the viewpoint of Instructional Design. Within the field of instructional design there has also been high interest in the systems approach, i.e. with viewing discussing the phenomena researched as systems. In the instructional design field this has lead to the emergence of instructional systems design (ISD). [Dick & Carey 1991] ISD considers the instructional process and learning environments as systems consisting of parts that depend on each other for input and output. ISD has been developed as a set of methodologies by which learning environments can be effectively designed and produced [Tennyson 1997]. It strives to improve the support those environments provide to learning through the application of contemporary theories of learning, measurement, technology and management.

Instructional design was chosen, because due to its systems engineering origins it has a structured approach to solving the problem of creating an artefact for instruction. The instructional design discipline has been criticized for two reasons: firstly because it is seen as based solely on behavioristic learning theories and secondly because it seemingly simplistic view of the design of instruction as a linear process ([Dijkstra 1997], [Silber 2007], [Tennyson 1997]). Although the instructional design disciplines' origins are in the theoretical trend mainly in the psychology of learning known as behaviorism [Dijkstra 1997], during the 60s and 70s an emerging trend in the psychology of learning, cognition, became predominant and also influenced the instructional design field [Tennyson & Schott 1997]. In the 90s the field and the discipline became even more diversified and nowadays its behavioristic origins are mostly in the past [Tennyson & Schott 1997]. ID nowadays encompasses a multifaceted approach on learning using multiple learning psychology theories to explain and research the learning process. The Instructional Design disciplines views on the instructional design process have also been discussed by many authors in the recent years. Tennyson (1997) has described Instructional Design as a systemic process where linearity is replaced by a component-driven model where interactions between different components drive the cause-and-effect flow of the instructional design process, whereas Silber (2007) describes Instructional design as "not a process, [it is] only a moderately structured problem-solving instead", thus proclaiming that the knowledge base of Instructional Design itself is valid but often ill-used. The ID design process is discussed in more depth in chapter 3.4.2.

Instructional design has also been viewed in the light of many contemporary learning theories, for instance situated learning [Mönkkönen & Enkenberg 1996]. Situated learning approach emphasizes learning in context that reflects the way the knowledge will be useful in real life. Principles of situated learning can be summed up as follows: (1) abstract knowledge should be embedded into the authentic activity, (2) freedom to perceive and negotiation of meanings should be stressed and (3) engagement to the activity of community of practice should be allowed. In this study the considerations of situated learning approach that refine and alter the traditional instructional design approach have been taken into account where appropriate.

3.4.1 The role of teams in the design of instructional technology

Development research approach focuses on the research-based design of an intervention or a product, such as a game-based learning environment, as a solution for a need to improve learning [van den Akker 1999]. As a digital learning game is a software product the goal is also to make this software product fulfill the software requirements. At its best the design process of quality and engaging game-based learning environments combines expertise and know-how of various disciplines and perspectives.

Hedberg and Sims (2001) argue that during last decades there have been different ideas about the expertise needed in instructional design. These ideas have been based, at least partly, on the technological developments. In the 1990s the focus shifted to the establishments of teams as integral components of development activities. According to Hedberd and Sims (2001) this was a result of emergent set of skill in terms of understanding technology. The development team should consider various elements: the design of interface, the structure of content, the creation of instructional interactions and the learner. McCandliss et al. (2003) emphasize the need for continued cross-disciplinary discussions in order to construct shared meaning across different contexts and disciplines. From a social view, design is a collaborative activity in order to achieve consensus about perspectives and actions that might be taken to solve the design problem. According to Hedberg and Sims (2001) the involvement of all members of the development team in all phases of the production decreases the likelihood of a technology-pedagogy mismatch.

In addition to the collaboration between diverse disciplines, a crucial assumption in the development research is that interaction with practitioners (i.e. teachers, policy makers, developers etc.) is essential for collaborative construction of workable intervention or prototypes. The involvement of practitioners is necessary both for social reasons

(e.g commitment and ownership of users) and for technical benefits, which means improvement of prototypes in real life contexts. The educational and learning expertise needed in the design of digital game-based learning environments can be presumed to include sound knowledge on learning theories and pedagogical design of quality learning environment and also content area expertise in the case of content-specific games.

3.4.2 The Design-Based Research and Instructional Design Models

In this chapter we will discuss how the development of a learning intervention or instruction in the case of instructional design should be carried out according to the findings in the research on design-based design and instructional (systems) design. These findings are discussed in the context of production of learning games per the focus of this study.

Instructional design processes main tasks at a high conceptual level are 1) task analysis, 2) the creation or selection of instructional method and 3) evaluation [Mönkkönen & Enkenberg 1
Also the research-based design models, including developmental research, split the design into three main phases: the analysis, the production and the evaluation. This makes it possible to combine the knowledge in these two fields in the following three paragraphs, each going discussing one phase of design.

Developmental research is often used in connection to complex innovative tasks for which there are not many validated principles to structure and support the design and development activities [van den Akker 1999]. The research often focuses on realizing prototypes of complete interventions that serve as limited but promising examples. The prototypes are produced in succession with each successive prototype increasingly meeting the innovative aspirations and requirements. The process is cyclic: analysis, design, evaluation and revision activities are iterated until a satisfying balance between ideals and realization has been achieved. [van den Akker 1999]

The instructional design model that is used in this discussion is the system dynamics model of ISD. The system dynamics model of ISD is a non-linear system that dynamically adapts to the problem conditions of a given situation [Tennyson 1997]. This model is based on the earlier linear models of ISD. It is preferred because the advancements in learning theory offered from the cognitive psychology have been difficult to implement in the linear models because of the failure to adapt to changes in such things as why and how of objectives [Hannafin 1995]. The strength and flexibility of the system dynamics model is that it is learning theory independent [Tennyson 1997]. Other current knowledge on ISD methods are cited where appropriate and where they

can be applied on top of the system dynamics model.

Analysis

Analysis phase precedes the actual development of the instruction and/or the artefacts that support the learning process. In developmental research, the main activities that fall under the vague 'analysis' umbrella are preliminary investigation and theoretical embedding. Preliminary investigation is a more intensive and systematic examination of tasks, problems, and context. This is done to find information about good ways to go about the development work to be done as well as earlier experiences from similar domains. Typical activities include literature review, consultation of experts, analysis of available promising examples of related purposes, case studies of current practices. In theoretical embedding more systematic efforts are made to apply state-of-the-art knowledge in articulating the theoretical rationale for design choices. A theoretical articulation of this kind can increase the transparency and plausibility of the rationale. [van den Akker 1999]

The analysis phase of the system dynamics model of ISD is called situational evaluation. The situational evaluation establishes a preliminary diagnosis of the learning problem followed by a prescription designed specifically for that situation. The diagnosis of the learning problem, also called problem assessment, contains the following tasks:

1. Identification of discrepancies between desired and actual learning,
2. Identification of the scope of the problem,
3. Definition of constraints regarding solution,
4. Determining the learner characteristics and
5. Determining the ID competence of author team.

The prescription phase also called the ISD solution plan proposal, consists of the following tasks:´

1. Determining what kind of material will be used and what new material must be developed,
2. ISD process & methodology selection,
3. estimating costs and resource requirements and

4. preparing a program evaluation plan.

Its is noted that the knowledge produced by the ISD problem assessment does not produce objective information about different learners. Because of this, anticipative design should not be too restrictive both on the design process and the end product. Additionally it is argued that according to the findings in situational learning theory, the results achieved in instructional methods are not universally valid and reliable. Therefore confidence in methods should be made questionable. Instead of methods and tricks, interest should be paid to those instructional solutions which do not restrict the learning situation, but give opportunities for various applications. [Mönkkönen & Enkenberg 1996]

Development

Developmental research does not form any general kind of prescription for the implementation of the development stage of intervention design. Instead it is assumed that suitable process and methods for each type of interventions are formed and/or selected during the preliminary investigation and theoretical embedding tasks [van den Akker 1999].

The system dynamics model of ISD does not explicitly divide its tasks to development and evaluation but instead presents a collection of components that contain a set of responsibilities with cause and effect relationships. The model does not present a general process model for design but instead suggests that a suitable set of components and responsibilities and a suitable process is formed during the analysis phase. [Tennyson 1997] In this study the majority of the components and their responsibilities are considered to belong to the development stage with the exception of formal evaluation which clearly supersedes the design of instructional system [Dick & Carey 1991].

The components and associated responsibilities related to the development stage are described in Figure 3.4. Each of the components (foundation, maintenance´, design, production and implementation) are written on large and bold fonts. They are also numbered and their area of responsibilities is visualized with boxes.

In the foundation component provides the flexibility to the design process. It allows the process to dynamically adapt to the problem conditions of a given situation. The selected educational philosophy and theory, as well as instructional theory will also affect all the other components of the design process. They are to be selected with regards to the problem in hand and the most recent research in learning.

Maintenance domain's goal is to keep the learning environment at the same level of effectiveness as originally developed. This can include making maintenance evaluations,

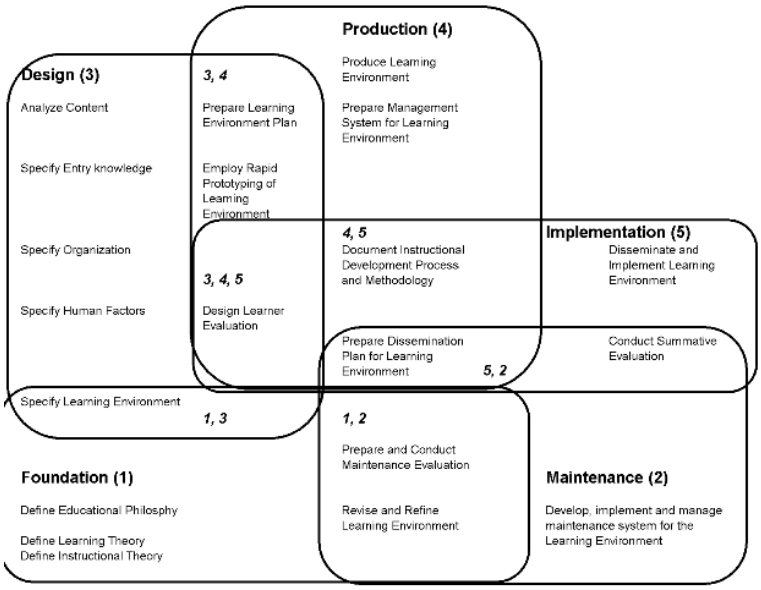

Figure 3.4: Components and responsibilities of the system dynamics model of ISD 3.4.

revising and refining the learning environment as well as developing, implementing and/or operating a maintenance system for the learning environment.

The design domain sets up the specifications and the plans for the learning environment. The important activity here is the definition of the information to be learned. How this analysis is conducted depends heavily on the learning theory used.

The production domain is involved in the actual development of the learning environment [Tennyson 1997]. In this sense the foundation and design components are semantically closer to the analysis phase of the design process and implementation are closer to the evaluation phase. However, as the process is non-linear such a direct temporal distinction cannot be made. This domain is usually associated with the most effort in time and cost especially if a computer-based learning environment is to be developed. Consequently most of the risks involved in the design are located in this component. The design and production components can involve rapid prototyping of the learning environment. The design of the learning environment can also be more plan and specification-based if this is considered to be more effective.

The implementation domain provides the means to put the learning environment into operation [Tennyson 1997]. The activities related to the implementation domain are often seen as not belonging to the actual development to the learning environment.

Evaluation

In developmental research, the activities that form the evaluation stage are a) em-

pirical testing and b) documentation, analysis & reflection. Clear empirical evidence about the practicality and effectiveness of the intervention is delivered through empirical testing. Testing is done with real user groups and in real settings. As the variation of possible interventions and contents is wide a broad range of indicators for success should be considered. To contribute to the expansion and specification of the methodology of design and development much attention is paid to systematic documentation, analysis and reflection on the entire design, development, evaluation and implementation process and on its outcomes. [van den Akker 1999]

ISD models, including the system dynamics model of ISD, include two kinds of evaluation, formative and summative. Formative evaluation is oriented to improve instructional programs by feeding back information into the system as the different program components are being used [Dick & Carey 1991]. Summative evaluation, in contrast, is used to measure the degree to which major outcomes are attained by the end of the course [Scriven 1972]. The system dynamics model of ISD includes conducting formative evaluation activities throughout the design process. In addition to this, the preparation of the summative evaluation is included in the responsibilities of the implementation and maintenance components. The summative evaluation is intended to be carried out by independent parties when the designed learning environment is in use.

Neither the system dynamics model of ISD nor developmental research make statements of how the evaluations are to be conducted. Instead they declare that the selection of evaluation methods to be used are depended on the learning theory selected as well as the features of the learning environment or intervention to be designed.

4 Entertainment Game Development

Producing a digital game on any platform is a task with many stages of work and a task that requires various kinds of expertise. In this chapter we review the accumulated knowlege and literature of entertainment game development to adopt it to learning game development. We have chosen the term emph(entertainment) game development to cover the whole process of developing a game to avoid the dual meaning of game development both as the overall process and one phase of the process wherein the game designs are implemented. By game development we mean the whole of the game development process, starting with coming up with a game idea, polishing it into a game concept, designing the game's functionality and developing the final game product based on the designs.

4.1 Game Production Team

In entertainment game development the producer is the one responsible for the whole project. He/she is the one responsible that the game is released on schedule and on budget and that everyone involved is doing what they should be [Novak 2005]. The rest of the development people are arranged into teams according to their responsibility area [Rollins & Morris 2004]. There are teams for design, art, programming, audio and testing and quality assurance (Figure 4.1).

The design team consists of developers of game design expertise. Sometimes also a story writer can be added. The art and audio teams consist of developers of graphics and audio production expertise. The programming team consists of developers of the game software. The testing and QA team, however, consists mostly of testers with game design and testing expertise. The teams can have team leaders of similar responsibility to software team leaders. The most notable difference between the team structures in entertainment game development and software engineering is that instead of a client, the game development team usually has a client-type relationship with a publisher. The publisher provides funding for the project, markets the game and sells it to suppliers after it has been completed. [Novak 2005]

Approaches that involve players (users) early the process have not been largely

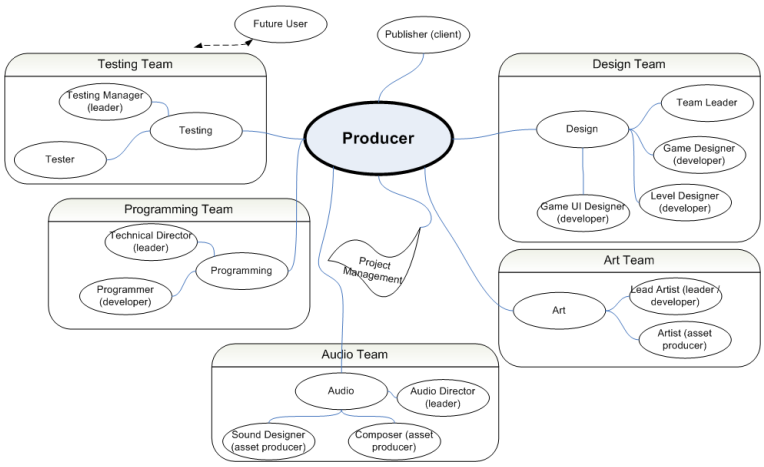

Figure 4.1: The Team Structure of an Entertainment Game Development Project .

adopted in game design although in the development of productivity software user-centred approaches are widely used and their benefits commonly recognized. Players are generally included in the game design process as testers, and the field of player-centred game design is only just emerging. ([Ermi & Mäyrä 2005]; [Sotamaa et al. 2005], [Sykes & Federoff 2006])

The expertise in game development can be described as all the know-how needed to develop a quality game. It involves game design, game asset production (such as game graphics and story production) and game development. Game design is the process of defining the rules according to which the game works. Game design consists of imagining the game idea, defining the action in the game, describing the game elements and their behavior and communicating the aforementioned to the game developers [Rollins & Morris 2004].

In game design, the expertise can be further divided to gameplay (or game mechanics) design expertise, level design expertise and game user interface design expertise [Novak 2005]. In addition to that, expertise and ways of working are needed to communicate the designs. Gameplay design involves the design of the main ruleset of the game and producing a coherent whole out of it resulting to a pleasurable game experience [Rollins & Morris 2004]. Level design deals with building the game environment or world [Novak 2005]. The nature of the game mechanics and environment depend on the type of game that is developed, game designers can have expertise on one or more game types (for example simulation games, adventure games and strategy games). The user interface designer determines the layout, content, navigation and usability features

of the game interface [Novak 2005]. This expertise ties in with the software usability design expertise.

Game asset production can be divided to graphics production, audio production and game story production [Novak 2005]. Graphics production involves several different domains including concept art, modeling and animation. Audio production consists of sound effect and game music production. There are not many reports on how important audio production is for games of learning. It can be assumed that the visual experience is more important to the game than the aural. Sound effects and music can however provide feedback and create moods in the game. Game story is reported to be important part of learning games [Chamberlin 2003]. However, not all games have elaborate stories, so the importance of game story design expertise depends on the type of game being developed [Rollins & Morris 2004].

Game software development is the expertise to produce the game software, including the game engine and editors needed to produce the game assets, if necessary. This includes making the requirements specification, designing the software, testing it and delivering it so it is ready to use. Game software development has a lot in common with traditional software development but it also has its own unique qualities, such as the need to produce high performance software which often leads to prioritizing code optimization above maintainability and readability. So, the learning game development project requires software engineering expertise and familiarity with the specific problems of game software development.

4.2 Game Production Process

Most games go through three development stages, from concept to design to production [Freeman 2002]. In the first stage, the concept proposal sets out the goals for the game. The stage of design involves a lot of discussions with artists, animators, musicians and engineers. The end product of this discussion is the design document. The final stage involves implementing the design plans. [Freeman 2002] The concept and design stages are collectively called pre-production [Novak 2005].

The design and production stages often overlap, because the game can be thoroughly evaluated only after it can be played. An extreme example of this is iterative game design, which is covered in chapter 2.2.5. Game concept generation is covered in chapter 2.2.2. Chapter 2.2.3. covers the game design phase. Finally, chapter 2.2.4. covers the documentation used in game concept generation and game design.

4.3 Pre-Production

The task of the pre-production phase is defining the rules according to which the game works. The work made in the pre-production phase is commonly called *game design*. As discussed in chapter 2.1.3. the player does not experience these rules but rather the playing experience that emerges from them. That means that game design is a second order design problem: the goal of design is to provide player with a particular kind of play experience but that experience cannot be designed, because it is an emergent consequence of the game rules the designer designs [Zimmerman 2003]. Another characteristic of game design is that in contrary to other design fields, game design is not design to fulfill needs. The goal of game design is not to fulfill any need but to please [Laurel 2003] [Zimmerman 2003].

The game design work includes imagining the game idea, defining the action in the game, describing the game elements and their behavior and communicating the aforementioned to the game developers [Adams 2005]. Game design is creation of something completely new [Rollins & Morris 2004]. Game design is not an organized discipline. It is described as an emerging field that is not yet fully understood and which has no established methods of working. [Costikyan 1994] [Church 1999] [Salen & Zimmerman 2003]

4.3.1 Game Concept Generation

As mentioned before, the designer's job is to create something new. This is a truly challenging task [Rollins & Morris 2004]. The first task in game design is to imagine the new game. This stage in game design is called concept or concept design ([Rollins & Morris 2004], [Ryan 1999a], [Ryan 1999b]). After the designers' have come up with a game concept, it must be analyzed. The game concept's potential is considered in the analysis: How good the game concept is from the playability perspective and how well it suits for its intended target audience [Rollins & Morris 2004]. The decision of producing this game is based on this analysis. In this chapter we examine the concept design stage and different techniques introduced for idea generation and evaluation.

According to Rollins & Morris [Rollins & Morris 2004], there are four phases in a creative process:

1. Inspiration - Where to get ideas,
2. Synthesis - Combining ideas,

3. Resonance - Creating synergy from ideas and

4. Convergence - Finishing the concept.

Ideas are first cultivated at the inspiration phase, combined for synergy and then rounded up and polished to form the game concept.

According to Rollins & Adams [Rollins & Adams 2003] and Adams [Adams 2005], game ideas begin as dreams or daydreams. Games as interactive entertainment are seen as a means to make these dreams reality. According to them the design of a computer game begins with the question, "What dream am I going to fulfill?". Rollins & Adams [Rollins & Adams 2003] suggest using different media for inspiration for these dreams. Ideas for games are currently gathered from a limited set of sources [Rollins & Morris 2004]. Other sources than sci-fi and fantasy fiction and action films should be used [Rollins & Adams 2003]. Game designer should be interested in a variety of topics and should have a broad education. This allows him to use ideas from a wide variety of sources. ([Crawford 1982], [Crawford 2003])

Rollins & Adams [Rollins & Adams 2003] also suggest using existing games as a source of new game ideas. This can be through developing a mastery in games:

-> *"When you play a lot of games, you develop a sense of how they work and what their good and bad points are. Playing games is a valuable experience for a game designer. It gives insight and lets you compare and contrast the features of different games."* [Rollins & Morris 2004]

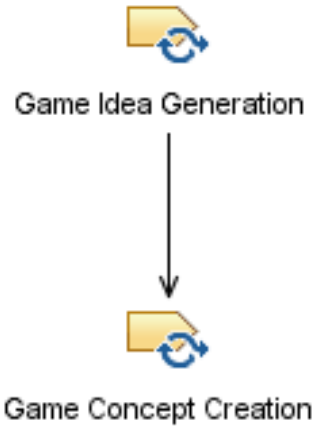

Figure 4.2: The Activity Diagram of Game Concept Creation.

They also suggest that frustration that is caused by a sub-par game can be a source of game ideas, too. This can conjure up an idea of ideal game that would have none of those problems that led to frustration. To conclude, Rollins & Morris also warn about the potential risks in mining existing games for game ideas. This technique can result in games that look or work alike and may result in mediocrity. Greatest games break new ground and don't necessarily follow the footsteps of earlier games. [Rollins & Morris 2004]

Adams [Adams 2005] suggests using the player's role as the starting point for formulating a game concept. Thinking about what the player is trying to be and what he will do in the game puts the player in the centre of the creative process and supports the notion that the concept is based on the player interacting with the game [Crawford 2003]. Another way to think about this is to think about the player experience that is desired for the players of the game. The game aesthetics and different kinds of fun categories described in chapter 2.1.3. can help in this consideration. These experience categories also put the player in the centre of game concept design.

Techniques that emphasize putting the player in the centre of the idea generation process include market or audience-driven methods. As Morris & Adams [Rollins & Adams 2003] remind, the games are ment to be played by players so understanding the audience is key. Erik Bethke [Bethke 2003] discusses the relationship between market or business context and design in game concept design. Tim Huntsman [Huntsman 2000] recommends asking questions from marketing experts in order to acquire data on customer expectations and different target markets.

The most common technique of evaluating game concepts is to pitch them to a group of game designers or testers. The more experienced the members of this audience are the more valuable insights can be gathered this way. [Rollins & Adams 2003] The game concepts can be also compared to successful games of the same genre. This treatment includes the comparison of key features in the game that are valuable to the player. [Ryan 1999a]

4.3.2 Game Design Phase

Game design is the task of defining the functionality of the game [Ryan 1999b]. As already mentioned this functionality should focus on the player and his actions. In the game design phase the gameplay modes of the game is defined and fleshed out in an iterative manner from general factors to the more detailed specifications [Adams 2005]. The main areas of game design are game mechanics design, game interface design,

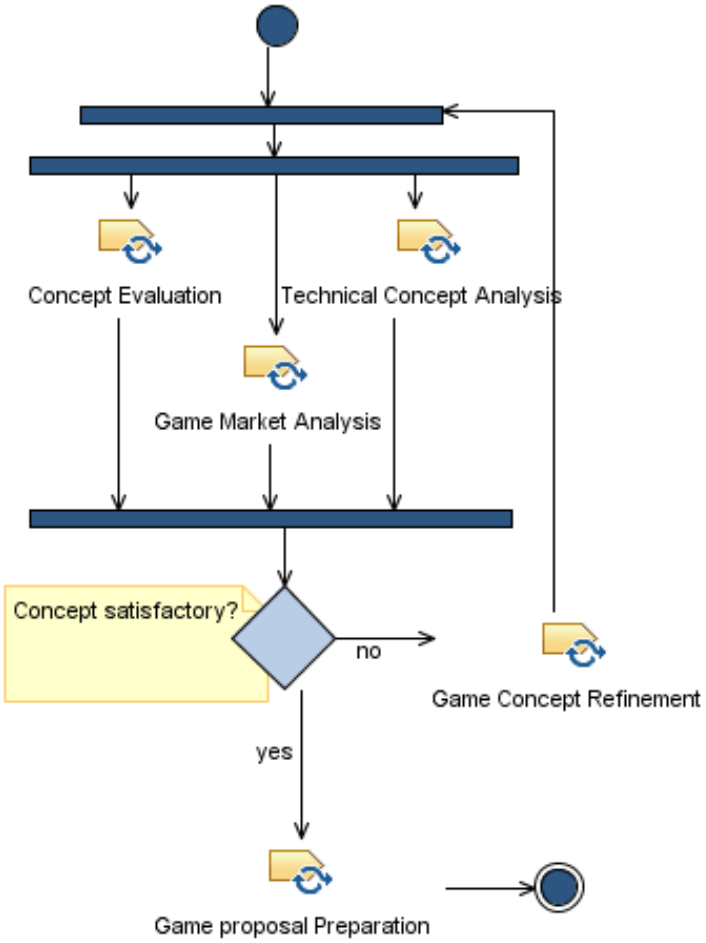

Figure 4.3: The Activity Diagram of Game Proposal Creation.

game graphics design, game audio design, game level design and game story design ([Novak 2005], [Rollins & Adams 2003], [Ryan 1999a], [Ryan 1999b]).

Game consists of one or more gameplay modes. A gameplay mode is defined by the view, interaction model and gameplay it has. If any of these changes that means that the game changes gameplay modes. [Adams 2005] Interaction model describes the way that the player is interacting with the game world. Two traditional interaction models are the avatar model and omnipresence. In the avatar model the player controls one character which in turn interacts with the game world. In omnipresence the player affects the game world directly. The game view can be two or three dimensional and viewed from different angles. Perhaps the most common views are three dimensional

views from the first or third person, where the game world is viewed either through the avatar's eyes (first person view) or behind his or her back (third person view). An isometric view is a two dimensional view viewed from a slight angle from above. Other types of game views are also used, for example context sensitive and from above. [Adams 2005]

The gameplay is defined in terms of what the player can and must do in the game mode. Gameplay can be defined as the challenge given to the player and the possible actions that the player can use to overcome that challenge [Adams 2005]. Gameplay is tied to the game mechanics in use and the mode of competition. As noted in the chapter 3.2.2 the player can compete against other players or against the game. The player can also team up with other players in some games. In some games a combination of different competition modes is used [Adams 2005]

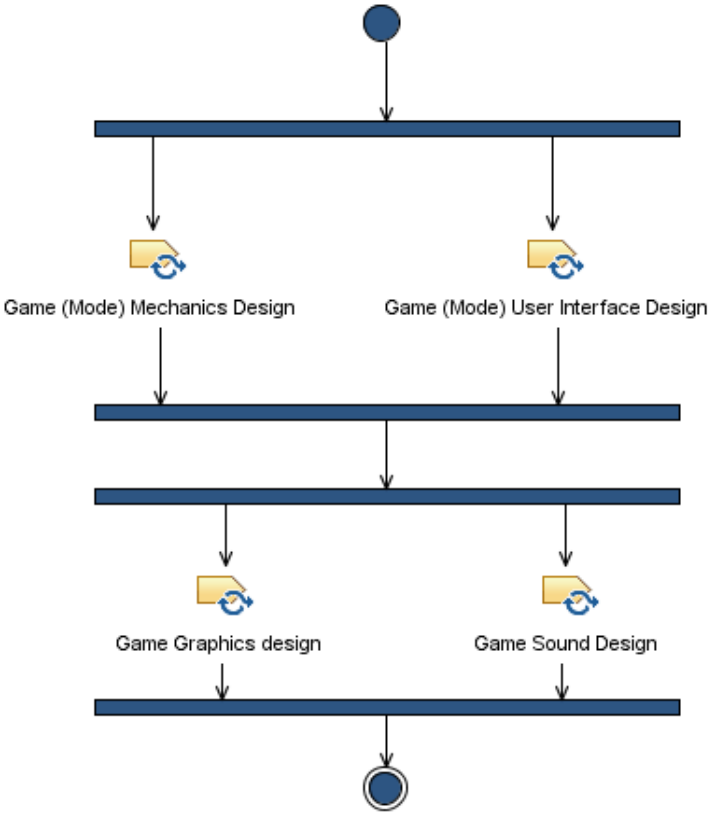

Figure 4.4: The Activity Diagram of Game Mode Design.

The design of game mechanics involves the design of challenges, internal economy and the task of balancing the game mechanics. There can be many kinds of challenges

in a game, as listed in chapter 3.2.2. The type of challenge also defines the main activity in the game mode. Because one of the main things the designer must do is communicate the design decisions to other team members, documenting the designs is an important part of the design work.

Different kinds of design documents are kept in different phases of pre-production. The first document used is the game concept document that describes the game's basic idea in short. It is then expanded to game proposal, which contains analysis of the game's potential and resources needed in design and development. The final document made in game pre-production is the game design document. It contains all the features of the game-to-be. Design documents are described in chapter 4.4. It is also good to keep a design diary during the game design process [Rollins & Adams 2003]. The designers should write down design decsions made and their justifications to the diary. Also the ideas not implemented should be written down. The diary will be useful as a resource when making decisions later in the process. It also saves the team from contemplating same decisions again later in the process [Rollins & Adams 2003].

4.4 Communicating a Design (Game Design Documents)

> *[Ryan 1999a]] The purpose of game design documentation is to express the vision for the game, describe the contents and to present a plan for implementation.*
>
> — *[*

The producer communicates the goals of the project through documentation, whereas designers demonstrate their ideas and programmers get their instruction and express their expertise [Ryan 1999a]. Design documents help a team of people to create a game [Rollins & Adams 2003]. A great design document is one of the key elements for a successful game development project [Freeman 2002]. Design documents also play a sales role. This does not mean selling the game to the public; it means selling the idea of the game to a publisher [Rollins & Adams 2003].

The authors of the documentation should be given guidelines on how to build the documents to reduce the *hype language* and enforce the definition of solutions in concrete form, to promote clarity and certainty and ease the drafting of schedules and test plans [Ryan 1999a]. Design documents are written to be used so they should be complete, well-ordered and easy to read [Freeman 2002].

As described in the previous chapter, each stage of pre-production produces its

own kinds of design documentation ([Ryan 1999a], [Freeman 2002]). There are many approaches to identifying and naming documents of different game design phases and tasks. Freeman [Freeman 2002] divides the documents used in a game development process into three categories according to the stage of development they are related to: concept paper, design document and production documents. Ryan ([Ryan 1999a], [Ryan 1999b]) identifies four documents, two of which are produced in the concept phase (game concept, game proposal) and two in the design phase (functional specification, technical specification). Rollins & Morris [Rollins & Morris 2004] introduce another document categorization, which consists of two concept treatments, one design document and one informal notes compilation. Rollins & Adams [Rollins & Adams 2003] have same kind of categorization of documents, albeit the informal document is missing from that list.

In this study, we use the following three-tier design documentation identification: the concept stage uses game concept and game proposal documents and the design stage produces the game design document (table 4.1). We also acknowledge that the technical documentation is an important part of the game development process, but that is a matter of the production phase of game development process. Technical documentation is therefore discussed in chapter 4.

Informal documentation may naturally be used on top of the afore-mentioned documents. The lead designer is the principle author of all formal design documentation, except the technical documents [Ryan 1999a]. In the next three chapters we will discuss the concept document (chapter 4.4.1), game proposal (chapter 4.4.2) and game design document (chapter 4.4.3) in more detail.

4.4.1 Game Concept Document

The purpose of game concept document is to communicate early game ideas and concepts to acquire resources for implementing the game. The audience for these kinds of documents are usually clients, publishers or management of the game development company. In addition to this, the same documents also serve the purpose of selling the game idea to the development team. Concept documents (called also high concept documents or concept papers) only cover the main features of the game concept whereas proposals (or treatments) offer analysis of marketing, development, resource and gameplay factors. This chapter discusses game concept documents. The game proposal is described in the next chapter.

In the game concept document the game's basic idea and activity is expressed in

Table 4.1: The three stages of game design documentation

Stage	**Document**	**Purpose**	**Contents**
Concept Generation	Game Concept Document	Expresses the core idea of the game, advancing the idea to proposal [Ryan 1999a]	Introduction, Background, Description, Key features, Genre, Platforms, Concept Art [Ryan 1999a]
Concept Generation	Game Proposal	Formal project proposal used to secure funding for a game development project [Ryan 1999a]. Document for presenting the concept to publisher and development team [Rollins & Morris 2004].	Expansion of the game concept; market analysis, technical analysis, cost and revenue projections, art [Ryan 1999a]
Game Design	Game Design Document	Outlines the features and functions of the product for the team doing the work on the product [Ryan 1999b]. It ensures that what is produced is what you want to produce [Freeman 2002].	Game mechanics, user interface, art and video, sound and music, story, level requirements [Ryan 1999b]

a general level [Ryan 1999a]. In addition to that the playability and game experience goals are set. Important factors related to the game's basic idea are game genre, the main game mode and the intended target audience [Ryan 1999a]. The game mode may vary for example according to the duration of the game and the number of players. A description of gameplay from the player's point of view is usually included to give insight on the gameplay and playability aspects of the game concept. The main contents of game concept document are:

- Introduction, short and catching description of the game,
- Background, information about the game's background and connections to other products (this part is optional),
- Description, the description of gameplay of the main gameplay mode from the player's perspective,
- Most important features, a list of features that make the game unique (from three to five),
- Genre, description of the game's genre and it's position compared to previous games,
- Platform, to which hardware platforms the game is intended and
- Concept Art, extracts from the game art. [Ryan 1999a]

The game concept document should be concise, only a few pages [Ryan 1999a]. It should not go into details but rather cover the general lines of the game concept. On the other hand it should not be too vague. The reader must be able to form a clear conception of the game [Freeman 2002]. The document should describe what the player is doing in the game and why that is interesting to the player [Ryan 1999a]. This reduces the risk of misunderstanding.

4.4.2 Game Proposal

The game proposal document is a formal project proposal with which the funding and resources for the game are applied [Ryan 1999a]. Game proposal is an updated version of the game concept document to which market and technical analysis is included. In addition to that the person responsible for software design makes the analysis the

resources needed to implement the game and risks involved in the development. These risks can relate to experimental functionality. The analysis includes weighing different options of design and implementation. The main contents of the game proposal are:

- Introduction, short and catching description of the game,
- Background, information about the game's background and connections to other products (this part is optional),
- Description, the description of gameplay of the main gameplay mode from the player's perspective,
- Most important features, a list of features that make the game unique (from three to five),
- Genre, description of the game's genre and it's position compared to previous games,
- Platform, to which hardware platforms the game is intended,
- Market analysis, target market of the game, comparisons to similar games in the market and appraisal of the features and their attractiveness,
- Technical analysis, list of experimental features, risks of development, alternative paths of development and appraisal of resources needed and
- Concept Art, extracts from the game art. [Ryan 1999a]

In the market analysis the games target market and the reasons that the game will appeal to it must be defined [Ahearn 2002]. The market analysis cannot be based on opinions and predictions but instead on realistic numbers. The most important part of market analysis is the comparison of the game concept to other products already on the market. An essential part of both the comparison and target market definition is a comparison of game features [Ryan 1999a]. Because the aim of the market analysis is to point out the need for producing the game, should these analyses point that need and the demand for the game concept [Ahearn 2002].

In the technical analysis the experimental features, the major development tasks, biggest risks of development, alternative paths of development and the needed resources and time of development of the game should be covered [Ryan 1999a]. The experimental features are features of the software that have not been implemented before. This

includes only such features of which no development reports are available or which have not been implemented even elsewhere. In other words, the list is not comprised according to the previous experience of the development team [Ryan 1999a]. The description of experimental features should include an estimate of how long it will take to prototype them.

In the major development tasks the main areas of which the game development will consist of are described. The descriptions should include the time estimates of implementation [Ryan 1999a]. After that the risks threating the development of the game are described. The risks may originate from experimental technology, new development tools or software libraries, areas new to the development team and so on. This section is important because a detailed description of risks raises the client's and other stakeholders' confidence in the development team's ability to cope with problems and to foresee them [Ryan 1999a].

In the alternatives section of the game proposal alternatives to the risky development tasks and experimental features may be presented. These descriptions should include time estimates and estimation of impact to the gameplay. Finally, in the resources and time estimates section of the document the time estimates and needed resources are gathered and an overall estimation of budget and timetable is made. [Ryan 1999a]

4.4.3 Game Design Document

The purpose of the game design document is to describe the contents of the game thoroughly ([Rollins & Adams 2003], [Ryan 1999b]). A good design document is one of the prerequisites of a quality game development project [Freeman 2002]. The target audience for the game design document is the development team of the game [Ryan 1999b]. A good design document ensures that the whole development team is developing the same kind of game. The information of design decisions made are written into the design document and what is more important, the process of writing the document transforms vague ideas into precise definitions [Rollins & Adams 2003].

The length of the design document can range from dozens of pages up to over a hundred pages. More important than the length is that every design decision is included in the document. This does not mean that the design cannot change in the course of the development even after the document is written. Changes can occur in the development and even in the testing phase but they should be written down in the design document too. The game design document consists of the following main

chapters [Ryan 1999a]:

- The game mechanics,
- user interface,
- graphics,
- sound and music,
- story (if the game has one) and
- level design plan.

The game mechanics chapter consists of description of the gameplay modes, the game elements and the internal economy of the game. The main content of the gameplay mode description is the description of game activity of the gameplay mode. The game elements described include player's avatar or character (if he or she has one in the game), other characters, objects and locations. In the game economy sub-chapter the interaction between game elements is described. If applicable the chapter includes also the description of the game's artificial intelligence, multi-player modes and the flow of the gameplay from beginning to the end from the player's perspective. [Ryan 1999b]

The user interface is described with user interface drawings, flowcharts and functional requirements in the user interface chapter. Functional requirements are discussed in chapter 5.4. The graphics and sound chapters are very similar to each other. Both include the description of aims in the area in question and description of the game art in the area in question accompanied with examples. [Ryan 1999b]

The story chapter starts with the synopsis of the story: a short description of the whole story and its theme. After that the story is described completely. This description can be accompanied with flowcharts which are useful especially if the story branches at some point. Level design plan chapter describes how many and what kind of levels the game has and how they follow each other. This is accompanied with a plan of when the more advanced features of the gameplay are revealed to the player as the game progresses. It is common to design the in such a way that he cannot use all of the game's features right away but they will be uncovered when the player advances in the game. This helps to engage the player's attention. [Ryan 1999b]

The preparation of the game document is a challenging task ([Ryan 1999b], [Freeman 2002]). Even though the game design document is supposed to be as detailed as possible, it

should be able to communicate the soul of the game or the inner essence of the game as well [Freeman 2002]. Game development involves a lot of creative people that like to advance the game design to their own desired direction [Freeman 2002]. Thus it is important that the lead designer directs their creativity to the right direction so that the game design stays coherent. Another important feature of the game design document is readability. The game design document is made to be used; it is probable read all through the implementation phase. Thus it is important that the text is not hard to read, that the contents are easy to find and ordered consistently and the layout is clear [Freeman 2002]. The use of figures and tables is also important and in some cases compulsory.

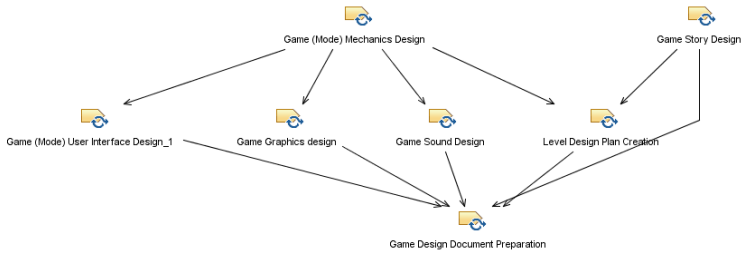

Figure 4.5: The Activity Diagram of Game Design Document Preparation.

4.4.4 Iterative Game Design

As noted in the beginning of chapter 4.3, game design is a second order design problem. The game designer designs the rules of the game such as competition mode, attributes and behavior of the different game elements, the internal economy and all the other rules of the game. The game's possible states and outcomes are an emergent outcome of all of those and the actions of the player. This makes game design more difficult because the causal relationships between the game's rules and the gameplay experience that is the result of the rules is not straightforward ([Laurel 2003], [Zimmerman 2003]). The methodology of iterative game design attempts to solve this problem [Zimmerman 2003].

Iterative game design is based on a cyclic process in which the production of gameplay prototypes, analysis and refining the prototype take turns (Fig. 4.6). In iterative design the interaction with the system that is being designed is used to inform and develop the future versions of the product and the future iterations of the design. In the case of game design iterative design means building and testing gameplay proto-

types. In iterative design the roles of designer and user, creator and player, are mixed. [Zimmerman 2003]

Iterative game design pursues information about the biggest uncertainties related to the game design in question. These uncertainties are related to the game's main activity, that is, what the player mainly does in the game. This means the main gameplay mode and the main game mechanic. In a later stage the game's basic mechanics and the player's actions are clear. At this point the design concentrates on balancing the game. The list of uncertainties should be made at the start of the project and it should be updated after every iteration. Because the player experience cannot be fully predicted, the design decisions are made on the basis of the experiences gathered from the prototype tests. The project evolves as a continuing dialogue between the designers, the game and the testers. [Zimmerman 2003]

The group of game's testers grows as the project goes on. Typically in the beginning of the project the designers themselves test the game, but outside testers are used more and more as the project proceeds. As much data as possible is gathered from the game test situations by observing the game sessions and interviewing the testers during and after the testing. [Zimmerman 2003]

The design of the first gameplay prototype demands strategic planning in how to most quickly build a prototype with which information about the projects major uncertainties can be gathered in a quality way. Often the first prototype is a paper prototype which is built like a board game. It is important to figure out how to prototype different kinds of games on paper ([Juul 2006], [Zimmerman 2003]). In iterative design one must also find balance between long term plans and short term needs, which usually come up during testing. Because a lot of testing is made, too ambitious long term plans cannot be made or they should not prevent short term findings from taken into account. [Zimmerman 2003]

Before every testing the design team should plan what are the aspects of the game that are to be tested and how to test those aspects. Implementing the variables of the internal economy of the game with soft architecture like easy-to-modify text files or databases lowers the resources needed in the balancing phase of game design as the game economy can be refined on fly as testings progress [Rollins & Morris 2004]. After every testing a meeting must be held to go through all the information learned in the test. [Zimmerman 2003]

The main advantage of iterative game design is that the resources are used in the design process in a streamlined way. Because the testing is constantly informing the

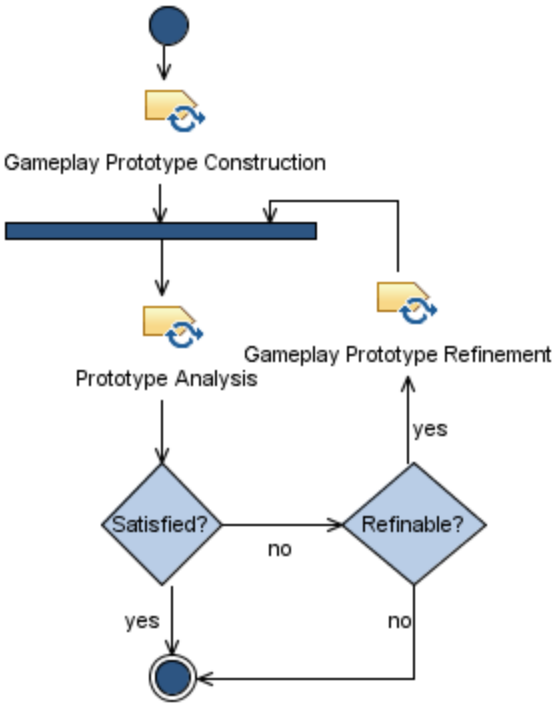

Figure 4.6: The Activity Diagram of Iterative Game Design.

design during every iteration there is no danger of using most of the resources in the design of useless features of the game [Zimmerman 2003]. Constant playtesting also helps in balancing the game. If the balancing can only be started after the implementation of the game, the time used for it will automatically delay publishing of the game. [Cook 2002] The third advantage is that with prototypes the design can concentrate around the main gameplay mode and basic gameplay, but as the project proceeds the focus can be shifted towards designing new features around that main gameplay mode [Cook 2002].

5 Methods from Software Engineering

This chapter deals with adopting software engineering methods to a quality learning game development process. Software engineering, as the name suggests, deals with the development of software products and solutions through a quality development process. The design and production of the learning game software is naturally one of the most important tasks in the development of a learning game. One has to note, though, that it is not the be-all and end-all of learning game development. Other tasks, such as the concept and gameplay design of the game and design of the educational aspects of the game, precede and support it and are necessary for the quality of the multi-disciplinary process. [Kankaanranta et al. 2007]

The development of learning game software is based on the learning game concept and design documents described in chapter 4.4. First phase in software development is to form a requirements specification based on these documents. The second phase is to form a software design that provides a solution to meet the requirements specified. The first part of software design is architecture design, which ties together the requirements and design phases by assigning requirements to specific software components. After the design is complete the software is implemented and then tested. As learning game software is usually exploratory in nature and no standard working solutions exist, the software development process is often done in an iterative nature.

In this chapter we review some basic methods and techniques from the software engineering literature. First we take a look at people and teams in software engineering followed by the exploration of the software development process. Then we discuss requirements engineering, the methods of collecting and specifying users needs and hopes for the software being built. After that we discuss software architectures and finally we describe the software development process and models to structure that process to enhance productivity and the quality of work.

5.1 Teams in Software Engineering

Traditionally in software engineering the people working in software projects can be organized into teams in three different ways. In the first option, people are assigned

their own functional tasks and relatively little combined work occurs. In the second option, informal teams are formed around single functional tasks so that one person can be a part of more than one functional team. In the final option individuals are assigned into permanent teams to which individual functional tasks are then assigned. The last option is the most productive according to the growing software engineering body of evidence [Pressman 2000]. All of these options assume that the project is administered by a project manager. In addition to that individual teams can have team leaders (Figure 5.1).

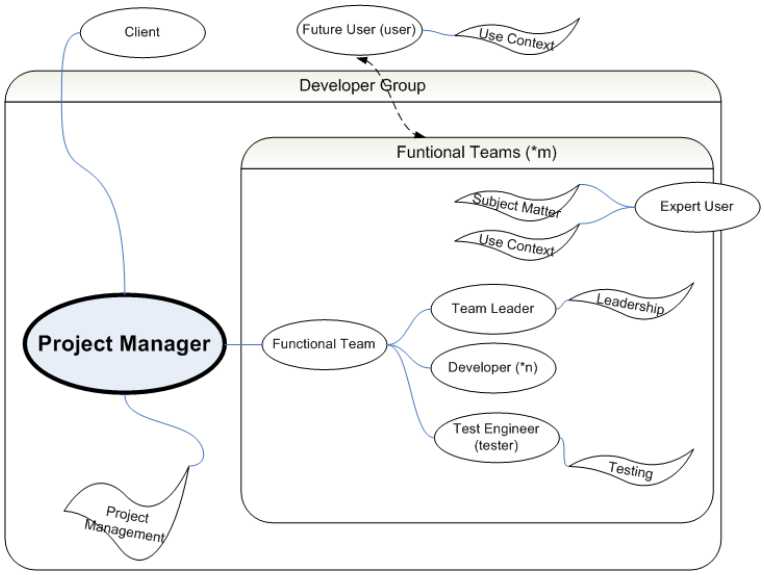

Figure 5.1: The Team Structure of a Software Development Project

In a software engineering project, every person is thought as having expertise on some area of software development. The idea of functional teams is that each team has expertise on the different tasks relating to producing the functionality needed for the end product. Team leaders also need expertise on team management and leadership. The project manager, obviously, needs expertise on project management and planning.

As for responsibility, most of the software project people fall under the role 'developer'. Developers are responsible for the functional portion of the project outcome that has been assigned to their team. Some teams may include 'testers' which are often called test engineers. They are responsible for reviewing and validating the software product in different points of development. The role of the team leader falls between a developer and project manager. He has more limited responsibility on project conditions than the project manager, but a more direct responsibility of the project outcome

(specifically the functional portion of the project outcome his team is responsible of).

Software development also has other stakeholders than the development group. They are not thought of as part of the development team as they are not thought of as having direct responsibility of the project, but they are part of the project environment and provide the developers important expertise and insight on the project's context and outcome. These can include the client, future users, possible subject matter experts and expert users.

Client is the person who pays for the development of the software product. His main responsibility is to set the goals of the project with the project manager. This responsibility is often stated as being more on the project manager's side with the client's main responsibility in the process being available for the discussion and ready to impart all the facts needed for the goal negotiation [Pressman 2000]. The future users of the software product are used by the developers to set the requirements for the product and get insight on the context the software product is to be used in [Pressman 2000]. In this case also, the responsibility for the requirements process is considered to be in the developers' side and it is considered to be up to their expertise and skills to be able to gather the right requirements [Pressman 2000]. The users can also be used to review the outcomes (that is, test) of the project [Budde et al. 1991]. In some cases the development group can recruit a special user to be a part of the development group, to act as an expert on the subject matter the software deals in and to contribute on requirements and testing. This kind of user is called expert user [Budde et al. 1991].

5.2 Software Development Process

Each development project has a development process. When the process has not been planned, the process is implicit. The notion of process is prevalent in the software engineering discipline. The software process is the set of activities and associated results which produce a software product [Sommerville 2001]. There are four fundamental process activities which are common to all software processes:

- Software specification, which is also known as requirements specification, and is discussed in chapter 5.4,
- Software development, wherein the software that meets the requirements is developed,

- Software validation, wherein the software's suitability to customer's needs is validated and
- Software evolution, where the software evolves to meet the users needs.

[Sommerville 2001]

In order to solve actual problems in software engineering the software development team must deploy a strategy to use all the different techniques, tools and methods of developing software in an organized manner. This strategy is referred to as a software process model. [Pressman 2000] In a process model, the different activities of software development are decomposed in different ways [Sommerville 2001]. The choice of process model depends of the software product to be developed and the organization that develops it ([Pressman 2000], [Sommerville 2001]).

Software process models are also known as software life cycle models [Pressman 2000]. A traditional software process model is the linear sequential model, which is commonly known as the waterfall model [Pressman 2000]. Other kinds of models are cyclical or incremental, for example the spiral model and evolutionary development. These are discussed in chapter 5.2.3. Newer software development methods known collectively as agile methods also offer a model for software process. They are discussed in chapter 5.2.4.

5.2.1 Process Improvement and Modeling

Making processes explicit is the first step towards process improvement [Germain & Robillard 2008]. The whole process becomes visible by creating a model of the process. This defined process is the requirement for process understanding, analysis, execution guidance, learning and communication. It will also support the improvement of the process. [Terävä 2005] The ultimate benefit of explicit process models should be a better understanding of development processes, resulting in improved quality products and more realistic estimations of the budgeted effort [Germain & Robillard 2008].

A process model can be descriptive, prescriptive or proscriptive. Descriptive modeling tries to make the currently used processes explicit. Prescriptive modeling specifies the recommended way of executing the process. Proscriptive modeling describes non-allowed behavior and is usually used as an addition to prescriptive or descriptive modeling. [Koskinen 1999]

5.2.2 Waterfall Process Model

The first explicit model of software development was derived from other engineering processes. Because of the cascade from one phase to another, the model is known as the waterfall model. There are numerous variations of the waterfall model. The principal stages of the model are (Fig. 5.2):

1. Requirements analysis and definition

The systems services, goals and constraints are gathered from the system's users and they are defined by a manner which is understandable by both the users and developers.

2. Software design

emphThe software design process establishes an overall software architecture.

3. Implementation and unit testing

emphThe software code is written in packages and the finished packages are tested for bugs and defects.

4. Integration and system testing

emphThe individual packages are integrated to form the software product and the product is tested for functionality compliance of the requirements.

5. Operation and maintenance

emphThe software is installed and taken into use. Maintenance phase begins.

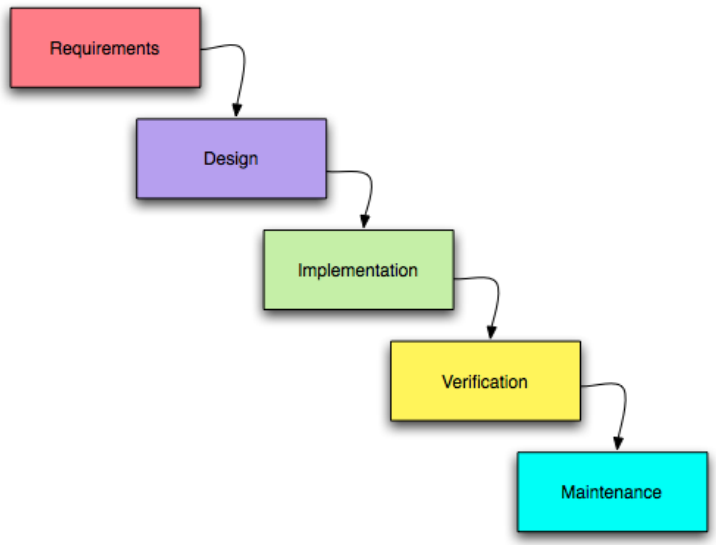

Figure 5.2: The Stages of the Waterfall model

The problem with the waterfall model is its inflexible partitioning of the project into these distinct stages. Requirements analysis may not be possible to do completely before even the design phase begins. Delivered systems are sometimes unusable as they do not meet customer's real requirements. [Sommerville 2001] Real projects rarely follow the sequential flow that the model proposes. It is often difficult for the client to state all the requirements excplicitly. [Pressman 2000] Despite all its shortcomings, the waterfall model has its place in software engineering: It provides a template into which methods for analysis, design, coding, testing and support can be placed. [Pressman 2000]

5.2.3 Incremental Models

In an incremental software process model the software development project goes over multiple mini-projects that follow the waterfall model. The requirements that are implemented in that increment are defined in the beginning of every increment. Many process models follow the incremental metamodel. The spiral model and evolutionary prototyping will be discussed in this chapter. In addition to them, for instance Rational Unified Process (RUP) uses one variation of the incremental model. [Haikala & Marijärvi 2001] In addition to evolutionary prototyping there is a prototyping method called rapid prototyping, which involves rapidly producing prototypes to test different aspects of the software solution to be built. We will discuss rapid prototyping as one way of specifying and validating software requirements in chapter 5.4.2.

The spiral model is an evolutionary model where iterative and free prototyping interconnects eith the systematic nature of a linear model. This model is also based on the rapid production of incremental versions of the software. [Pressman 2000] The spiral model has six main tasks: communication with the users, design, risk analysis, engineering, implementation and delivery and client estimation (Fig. 5.3).

The spiral model can be adapted throughout the life of a software product, so it is not restricted to development as are many other models [Pressman 2000]. The spiral process can continue to shape a new product as it becomes unusable during its life. The spiral model is a realistic approach to the development of large scale software systems. Because software evolves as the process progresses, the developer and customer better understand and react to the risks at each evolutionary level. [Pressman 2000]

The spiral model has also drawbacks. It may be difficult to convince customers that the evolutionary approach is controllable [Pressman 2000]. Also, the spiral paradigm can thought of as unintuitive for project managers and producers. Time does not go in

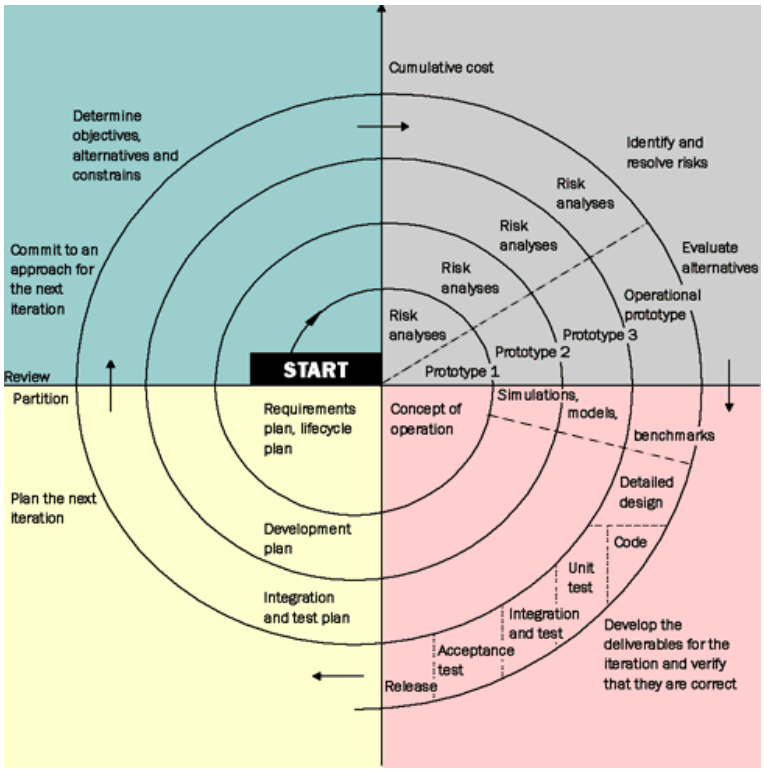

Figure 5.3: The Spiral model [Pressman 2000].

spirals, it goes straight onwards (at least in gantt charts). [Cockburn 1997] The model also demands a great deal of risk assessment expertise. If a major risk goes unnoticed and unmanaged, problems will occur [Pressman 2000].

Evolutionary prototyping differs from rapid prototyping in two ways. First, earlier prototypes are used as a basis for later development of the software. Nothing is thrown away and the final prototype to be built will be the delivered final software product. Second, the prototypes produced are of high fidelity [Crinnion 1991]. The aim of evolutionary prototyping is to produce a stable prototype in an organized manner and enhance it all through the development project [Crinnion 1991]. Using this method the developers can add features to the software at a later stage or make such changes that oculd not be foreseen at the requirements engineering stage of the project. The prototypes must be easily modifiable [Budde et al. 1991]. This results in an iterative development process in which new versions of the product is built on top of the earlier versions [Pressman 2000].

The biggest benefit of an evolutionary process is that each increment offers the development team insight about the product they are developing as well as the devel-

opment process itself [Cockburn 1997]. When using an evolutionary process model the development team can also concentrate on developing those parts of the software they have good understanding of. At the same time they can experiment with the more risky development tasks. Prototyping techniques are most valuable when full requirements specification is not, for some reason, available at the beginning of the project ([Budde et al. 1991], [Krief 1996]). This is true especially when the software's target context is not fully explored. Prototyping has also been found effective with software that are highly interactive [Crinnion 1991].

5.2.4 Agile Methods

emph"Core to agile software development is the use of light-but-sufficient rules of project behaviour and the use of human and communication-oriented rules." [Cockburn 2001]

Agile methods are less document-oriented than traditional methods. They concentrate on program code. Agile methods are not as resistant to change as traditional software methods. [Fowler 2001] Agile methods are more people-oriented than process-oriented [Fowler 2001].

Agile methods are an attempt of compromise between too complex methods and no methods at all. They try to provide just enough process to gain a reasonable payoff. [Fowler 2001] The reasons for this are that traditional software process models are not always feasible in the rapidly changing business environment.

There are many software methods and process models that call themselves agile. Some of the most prominent of them include Extreme Programming (XP) [Extreme Programming 200 Scrum [Schwaber & Beedle 2001] and Feature-Driven development [Palmer & Felsing 2002]. The term agile methods and the Agile Manifesto, the collection of princeiples and values of agile software development, were made popular by the Agile Alliance. The Agile Alliance was formed when seventeen representatives of different agile method saw met to discuss alternatives to documentation driven software development [Kalermo & Rissanen 2002]. The goal of Agile Alliance was not to specify detailed project tactics but agree on values that support agile software development at a high level [Agile Alliance 2002]. The Agile Manifesto has twelve principles (table 5.1). The principles support the values of agile alliance:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation

- Customer collaboration over contract negotiation
- Responding to change over following a plan

Discussing all the varied methods of agile software development would be too big a task in the scope of this work. Therefore, Extreme Programming (XP) is chosen to exemplify agile methods. Kent Beck, Ron Jeffries and Ward Cunningham developed XP in the mid 1990's [Paulk 2001]. XP was developed before Agile Manifesto was writen, and it has widely influenced the principles of Agile Manifesto [Kalermo & Rissanen 2002]. The principles that XP was based on were improving communication, seeking simplicity, getting feedback and proceeding with courage [Cockburn 2001].

The rules and practices of extreme programming are divided into four areas: planning, designing, coding and testing. The rules and practices are illustrated in table 5.1 [Extreme Programming 2006].

The project process starts with collecting and writing the user stories. This is the XP equivalent of requirements specification. The idea behind the user stories is that they are understandable from the user standpoint, they can be scheduled as programming assignments and they can be tested with acceptance tests. If a proposed user story takes over three weeks to implement, it must be broken into smaller ones. After writing the user stories a release planning schedule is made. Over the course of the software project a new version of the product is released on a one to three-week basis. In release planning each release is assigned the user stories to be implemented on that release. Each release is planned on the beginning of the iteration at an iteration planning meeting. It is tested during the implementation with unit tests and after the release with acceptance tests. The project velocity is measured by comparing the implemented user stories to the release planning schedule and adjustments are made when necessary. This is all the process, scheduling and planning there is in XP.

The rest of the rules and practices relate to the work done inside the iterations. All code is produced using pair programming - that literally means two people working on the same computer. Pair programming is used to enhance the quality of code without impacting the time to deliver [Extreme Programming 2006]. This is combined with the practice of moving people around. This means mixing the pairs as well as people working on different areas of the software code in different pahses of the project to obtain more holistic understanding of the software. This leads to collective code owneership - no single part of the code is no one person's responsibility, each project

Table 5.1: Rules and practices of eXtreme Programming

Planning	**Coding**
User stories are written.	The customer is always available.
Release planning creates the schedule.	Code must be written to *agreed standards*.
Make *frequent small releases*.	Code the *unit test first*.
The Project Velocity is measured.	All production code is *pair programmed*.
The project is divided into *iterations*.	Only one pair integrates code at a time.
Iteration planning starts each iteration.	*Integrate* often.
Move people around.	Use *collective code ownership*.
A stand-up meeting starts each day.	Leave optimization till last.
Fix XP when it breaks.	No *overtime*.
Designing	**Testing**
Simplicity.	All code must have unit tests.
Choose a system metaphor.	All code must pass all unit tests before it can be released.
Use CRC cards for design sessions.	When a bug is found tests are created.
Create spike solutions to reduce risk.	Acceptance tests are run often and the score is published.
Refactor whenever and wherever possible.	

member has the right and the responsibility for the software code as a whole.

Design tasks and testing are also integrated to the coding. Design is done when needed and as little design work is done before coding as possible. All implementation of user stories must begin with the designing of unit tests for the solution. A unit test is a procedure used to validate that a particular module of source code is working properly [Pressman 2000]. When implemented, the user stories' unit tests are run. Acceptance tests are performed by the customer. They are created from the user stories. After a user story is implemented the customer tests it with corresponding acceptance tests. The customer is responsible for verifying the results. The results are published to the whole team. No user story is considered completed until it has passed the acceptance tests.

It can be argued that none of the practices of XP are not so extreme and in fact are combined from previous methodologies [Paulk 2001]. The developers of XP maintain that XP is not a finished idea but rather an evolving concept. [Beck 1999] The process of iterations producing releases are similar to evolutionary prototyping. What is extreme is the way things are done. XP combines light of process with control on all levels [Beck 2000]. Extreme programming is recommended to be used on projects whose functionality is expected to change every few months. XP was also set up to address the problems of project risk. [Extreme Programming 2006] XP is set up for small groups of programmers. Between 2 and 12, though larger projects of 30 have reported success. [Extreme Programming 2006]

XP has been experimented with very much during the past few years. Most of the experiments have concentrated on the impacts of pair programming [Kalermo & Rissanen 2002] but other practices of XP have been explored as well. Pair programming has been found to improve software quality and reduce time-to-market and increase programmer happiness and self-confidence [Williams et al. 2000]. Starting pair programming practise with programmers with no previous experience of it has been however found problematic. Programmers predujices about the effectiveness of the practice can be significant [Williams et al. 2000]. Also, pairing inexperienced programmers with experts may slow the experts down considerably [Cockburn 2001]. Other experiences of using XP include the notion of being able to accurately estimate the work done in each iteration, consistency of the process and the improved communication [Taber & Fowler 2000].

XP also has limitations. Projects with complete requirements do not seem to profit much from XP. The method is not feasible with large teams as lack of documentation combined with the fact that communication with large groups of people to maintain

common perception of the software is very difficult lead to knowledge of the software "falling between the cracks". Making critical software quality and relating software architecture desicions leading to optimal software architecture could also prove to be difficult in a software project that uses XP. XP does not accommodate for much modeling during the start of the project and even though refactoring of the code is encouraged, the changes that have to be made on the software architecture during later stages of the project may result to a lot more work than the ones that were planned in the beginning of the project during architecture design and analysis.

5.3 Process Control

Project management has three main responsibilities: making a plan to deliver the desired product, monitoring the work and reacting to emerging problems ([Novak 2005], [Sommerville 2001]). These three responsibilities are combined in process control ([Ogunnaike 1994], [Schwaber & Beedle 2001]).

Traditionally, software development has implemented a defined process control model meaning that every task has been defined as a process and the inter-relation of these processes has been defined in detail [Schwaber & Beedle 2001]. The control comes from predictability, meaning that with the same inputs the defined processes should produce the same results in different implementations. The manager can plan the whole project before it starts and knows what resources it takes to complete [Sommerville 2001]. The drawback of this is that if the processes are not explicitly defined, the plans and estimations are not reliable [Sommerville 2001].

The other approach is empirical process control which is represented in this study by the Scrum process control methodology. An empirical process control models have been developed for processes that are highly unpredictable. They provide control through frequent inspection and adaptation for processes that are imperfectly defined [Schwaber & Beedle 2001].

5.3.1 Defined Process Control

Defined process control means that every task has been defined as a process and the inter-relation of these processes has been defined in detail [Schwaber & Beedle 2001]. The control comes from predictability, meaning that with the same inputs the defined processes should produce the same results in different implementations. The manager

can plan the whole project before it starts and knows what resources it takes to complete [Sommerville 2001]. The advantage of this is the predictability: the decision of undertaking a development project can be made with the help of estimations of resources needed and the benefits that the end rpoduct provides. The drawback of this is that if the processes are not explicitly defined, the plans and estimations are not reliable [Sommerville 2001].

The software project is first initiated and its scope is defined via requirements elicitation tasks. The initiation includes the feasibility analysis task in which the possibility of completing the project with available resources is estimated. Then the project itself is planned in the form of hierarchical decomposition of tasks, the associated deliverables of each task are specified and characterized in terms of quality and other attributes in line with stated requirements, and detailed effort, schedule, and cost estimation is undertaken [Tripp et al. 2004].

In the ongoing software project change is managed with the following processes: risk management, quality management and plan management. In risk management risks threatening the success of the software project are identified, estimated and plans for avoiding and minimizing them are made. In quality management, the quality is defined by terms of attributes of the project and its end products. The desired quality attributes of the end products are specified in requirements. The procedures for ongoing quality assurance processes are defined in quality management.

Plan management involves directing, monitoring, reviewing, reporting and revising adherence to the project plans. Plan management is primarily responsible for responding to changes in the project environment. The project process is also monitored throughout the project. Adherence to plans, outputs and completion conditions for each task are analyzed. Deliverables are evaluated for their adherence to required characteristics. Resource use is also monitored. When the monitoring indicates that adherence to the plan is at risk, process control is responsible for making changes in the project plan. [Tripp et al. 2004]

As the end product quality evaluations are tied to completion of the said products the process monitoring and control cycle is related to the completion of the end products. In a linear process model this can mean that failure to fulfill the desired quality attributes of the end product is only found out after the development phase. This and the lack of instruments to control major change in the midst of the project means that defined process control model does not yield good results in projects with a lot of uncertainty and change.

5.3.2 Empirical Process Control

Empirical process control models have been developed for processes that are highly unpredictable. They provide control through frequent inspection and adaptation for processes that are imperfectly defined [Schwaber & Beedle 2001].

In this study Scrum methodology is examined as an example of empirical process control. Scrum is a software process model that focuses on project management ([Pikkarainen et al. 2008], [Racheva et al. 2008]). Scrum has been noted to be effective in software development projects were there was much uncertainty and change expected [Rising & Janoff 2000].

One of the principal tenets in Scrum is that everyone on the team is involved in the process [McGuire 2006]. A scrum team is typically made up of between five and nine people [Schwaber & Beedle 2001].

The product owner is the project's key stakeholder and represents users, customers and others in the process. The produt owner is the person who is responsible for adding items to the product backlog and re-prioritizing the items in the backlog. The product backlog is a prioritized features list containing every desired feature or change to the product. All additions and changes to the backlog must go through the product owner. This helps the scrum team deal with changes in the project. [Schwaber & Beedle 2001]

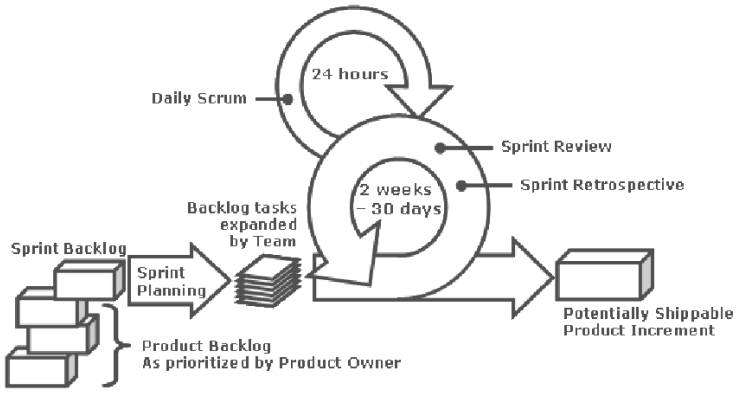

Figure 5.4: Diagram of the Scrum process [Schwaber & Beedle 2001].

The Scrum Master is responsible for making sure the team is as productive as possible. The Scrum Master does this by helping the team use the Scrum process, by removing impediments to progress and by protecting the team from outside pressure during Sprints [Schwaber & Beedle 2001].

Scrum breaks down production into short work cycles called Sprints [Schwaber & Beedle 2001]

(Fig. 5.4). Every Sprint produces a visible, usable, deliverable product [Rising & Janoff 2000]. Before every Sprint the development team sets its own goal for the Sprint and commits to completing it [Schwaber & Beedle 2001]. A sprint is timeboxed development, meaning that the end date for a sprint does not change. The team can reduce delivered functionality to achieve the sprint goal in schedule. [Rising & Janoff 2000]

At the start of each sprint, a sprint planning meeting is held during which the product owner prioritizes the product backlog, and the scrum team sets their Sprint goal, including selecting the work they can complete during the coming sprint. That work is then moved from the product backlog to the sprint backlog, which is the list of tasks needed to complete the product backlog items the team has committed to complete in the sprint. [Schwaber & Beedle 2001]

Each day during the sprint, a brief meeting called the daily scrum is conducted [Rising & Janoff 2000]. All team members are required to attend the daily scrum. In daily scrum every team member presents work done after last meeting, what she will do before next meeting and possible obstacles that hinder her work. [Schwaber & Beedle 2001]

At the end of each sprint, the team demonstrates the completed functionality at a sprint review meeting, during which, the team shows what they accomplished during the sprint. [Schwaber & Beedle 2001]

The main advantages of Scrum are 1) the increased capability to deal with change and 2) the shortened feedback cycles in development. The Scrum process combines flexibility and freedom to pragmatic assessment of real work done as well as the process itself [McGuire 2006].

Stakeholders can redirect project's goals between Sprints. Because the Sprints are short work cycles, this rarely results in a lot of wasted works. [McGuire 2006]

Iterative development is stressed in Scrum. The focus on producing shippable versions of the product means that the project's stakeholders can base their assessments and decisions on concrete products, not just documents, specifications, plans or estimations. [McGuire 2006]

With Daily Scrum meetings the progress and possible deviations from normal progress become visible to every team member including the project manager (scrum master) [Rising & Janoff 2000]. In game projects Scrum has the benefit of enabling game designers see complete pieces of finished functionality in regular intervals [McGuire 2006].

The potential drawback of experimental process control is that there are no projections at the start of the project on how much resources, including time, is needed to complete a finished solution for the problem in question with the desired quality

attributes. This drawback is lessened by the facts that as the project goes on this estimation is made possible by constant monitoring of the project's progress and that if the project contains much uncertainty and is subject to change this kind of estimaion could not be done accurately with any other methods.

5.4 Requirements Engineering

The objectives of the requirements specification process, which is also called requirements engineering or requirements process, are estimating the necessity and feasibility of a software project, setting the goals and objectives of the project and forming a model for the solution [Haikala & Marijärvi 2001]. The requirements engineering process answers to the question: "Have we specified a system that will satisfy the customer's needs and the customer can be content with?" [Pressman 2000]

Software requirement is a quality that software must have to solve a real-world problem. The requirements can come from the users, the target organization or the hardware [Pressman 2000]. The actions of users, the target organizations and hardware are complex by nature. The hardware is a pretty significant factor for example in mobile game development, where the constraints of the hardware set a pretty tight limits on what can be implemented. As a result the software requirements are usually a complex combination of requirements set by people in different levels of the target organization and their working environment. [Tripp et al. 2004]

Requirements have to be verifiable in each of the stages of the software development project. Many methods have been devised to facilitate this. These include architecture design & analysis (chapter 5.5), incremental prototyping (chapter5.2.3), rapid prototyping (chapter 5.4.2) inspections and testing. Typically, software requirements are uniquely identified so that they can be subjected to software configuration control and managed over the entire software life cycle. [Tripp et al. 2004]

Requirements can be divided to functional and non-functional requirements ([Haikala & Marijärvi [Pressman 2000], [Tripp et al. 2004]). Functional requirements describe the functions that the software is to execute; that is, the actions that can be performed with or by the application or system. Non-functional requirements are the ones that act to constrain the solution. They are also known as constraints or quality requirements. They can be further classified according to whether they are performance requirements, maintainability requirements, safety requirements, reliability requirements or dome other kinds of quality requirements. [Tripp et al. 2004]

Software requirements should be stated as clearly and as unambiguously as possible. Vague requirements are hard to verify and they don't inform the design solutions. This is particularly important to non-functional requirements [Tripp et al. 2004].

The requirements process is a process initiated at the beginning of a project and continuing to be refined throughout the software life cycle. It identifies software requirements and manages them using the same software configuration management practices as other products of software life cycle process. Requirements process has four distinct tasks: requirements elicitation, requirements analysis, requirements specification and requirements validation (Fig. 5.5). [Tripp et al. 2004]

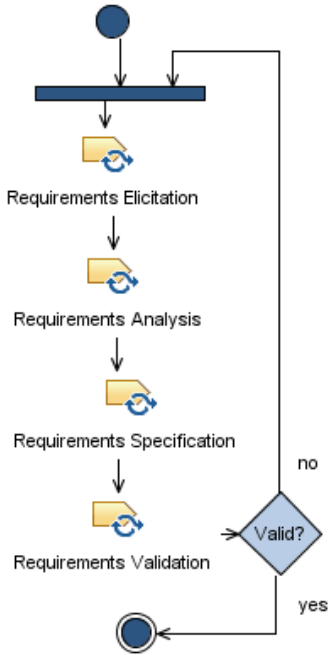

Figure 5.5: The Activity Diagram of Requirements Engineering.

Requirements elicitation is concerned with how and where from the software requirements can be collected. It is the first stage in building an understanding of the problem the software is required to solve. Requirements elicitation is fundamentally a human activity and good communication between software users and the software developers is essential. [Tripp et al. 2004] Requirements can be collected by interviews, inspecting the actions of the people for whom the software is to be built and by analyzing the functions and activities that the users will be performing with the software [Pressman 2000]. The work product of the requirements elicitation task is a list of

possible requirements for the system.

In the requirements analysis task the requirements are analyzed to detect and resolve conflicts between requirements, to discover the bounds of the software and how it must interact with its environment. There are two main subtasks in this task: the classification of requirements and conceptual modeling. [Tripp et al. 2004]

The classification of requirements includes informing the trade-offs between requirements and negotiating those trade-offs. Conceptual modeling deals with developing models of the real-world problem the software is meant to solve. Requirements can be classified with many variables:

- the type of requirement, either functional or non-functional,
- Whether the requirement concerns the product or process,
- Whether the requirement is derived from high level organizational goals,
- the priority of the requirement,
- the scope of the requirement, i.e. what software components the requirement concerns and
- the stability of the requirement or an estimation of how much the requirement is expected to change during the life cycle of the software.

The classifications that are appropriate for the project in case are used.

Conceptual modeling comprises models of entities from the problem domain configured to reflect their real-world relationships and dependencies. Their purpose is to aid in understanding of the problem, rather than to initiate design in the solution. The models that are developed can include data and control flows, state models, event traces, user interactions, object models, data models, and so forth. Which type of model is chosen depends from the nature of the problem, process requirements and the availability of methods and tools. At this moment the UML (Unified Modeling Language) method is the de facto standard in the software development industry for modeling because it is used so widely in the industry. [Tripp et al. 2004] One of the most common and useful modeling methods of UML, the use cases, are used as part of many software engineering process models as a way to model the functional requirements of a software. More information on use case modeling is in chapter 5.4.1.

The requirements analysis task takes the list of requirements produced in the elicitation task as its input. The work products of the requirements analysis task are the

classification of requirements and the conceptual model of requirements, such as the use case and conceptual system models.

Requirements negotiating task is also known as the conflict resolution. In requirements negotiation the conflicts between the requirements are solved. These include conflicting requirements from two different sources, a conflict between resources and requirements and conflicts between functional and non-functional requirements. The software developer shouldn't make these decisions alone. Usually the decisions are made in negotiation with different stakeholders of the software development project. Requirements negotiating is both related to requirements analysis and requirements validation. [Tripp et al. 2004]

The requirements negotiation task produces new information in the form of comments and asssessments of the requirements and the conceptual model from the project's stakeholders.

Requirements specification task is the task of producing a document describing the software requirements which can be systematically reviewed, evaluated and approved [Pressman 2000]. In the case of complex systems, up to three different requirements specification documents may be produced: system specification, system requirements and software requirements. For simpler programs, the last one is enough. The software requirements document forms the basis of the deal between the client and the vendor of the software. In the document it is specified what tasks the software should perform and what it shouldn't do. The software requirements document gives a thorough way to assess the software requirements before design and production. It should offer a realistic basis for cost, risk and schedule analysis. The requirements are often specified using natural language but in this document a formal description should also be offered. The choice of right technique is important. [Tripp et al. 2004]

Requirements validation task is concerned with checking that the requirements have been elicitated and understood correctly. In addition to that the requirements' intelligibility, consistency and completeness is checked. Requirements are validated once or multiple times during the software development process. Different stakeholders, including the representatives of the customer and developer, should review the documents. [Tripp et al. 2004]

5.4.1 Use Cases

Use cases model the system from the end-users point of view [Pressman 2000]. The development of use cases has become a popular method for elicitating and describing

functional requirements. The use case method has two parts: the use case model and the procedure for modeling use cases. [Mäkinen 2003] The use case model consists of actors, use cases, relationships and the system boundary [Arlow & Neustadt 2002]. A use case is a scenario that identifies the use event queue of the software. Together these scenarios form a description of how the software will be used. [Pressman 2000] The aim is to define the behaviour of the software without revealing its inner structure [Mäkinen 2003].

The first task in defining use cases is to identify the people and/or other objects that will use the software. These will be the actors of the use cases. [Pressman 2000] In some cases it is feasible to group similar types of users of software under the same actor. This is possible when these actors will perform the same actions with the software. [Arlow & Neustadt 2002] Relationships between use cases and actors are also depicted. The system boundary identifies what is included in the system and which parts of the activity related to the system is performed outside of the system [Arlow & Neustadt 2002].

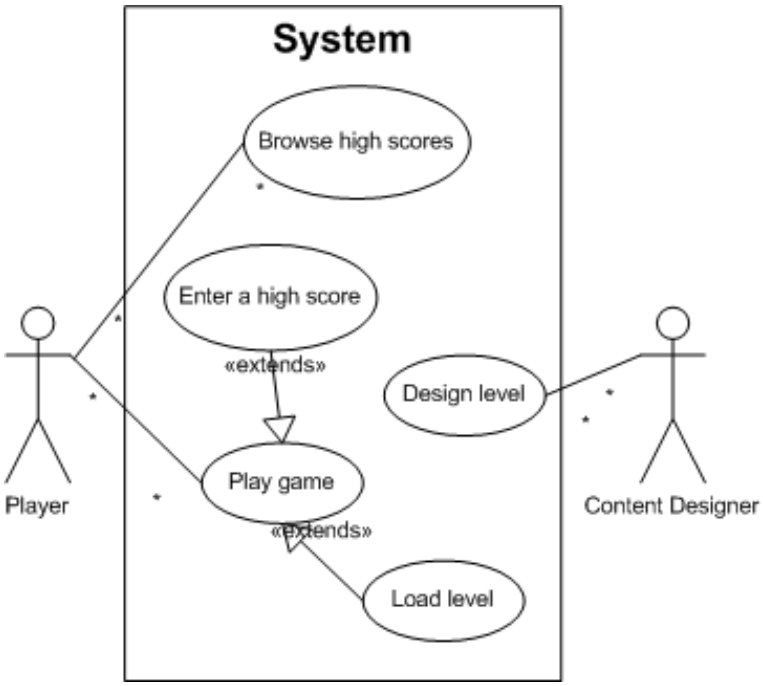

Figure 5.6: The Use Case Diagram [Arlow & Neustadt 2002].

Use case models are usually presented with use case diagrams (Fig. 5.6) and use case specifications (Fig. 5.7). Diagrams represent the system boundary by a box labeled by the name of the system. They show the actors outside the system boundary box

and use cases which constitute the system behaviour inside the box. The relationship between an actor and use case is shown using a solid line, the UML association symbol. The use case specification consists of [Arlow & Neustadt 2002]:

- Preconditions - these are the things that must be true before the use case can execute - they are constraints on the state of the system;
- Flow of events - the steps in the use case;
- Postconditions - things that must be true at the end of the use case.

High Score List

Use Case Number	P1.4
Name	Enter a high score
Actors	Player
Description	The Game has ended and the player has made a high score. The player enters her name and it is recorded into the high score list.
Priority	2 – medium
Preconditions:	1. The game has ended. 2. The player has acquired a score higher than the last score in the high score list.
Postconditions:	1. Player's name and score are at in the high score list at the correct position.
Flow of events:	1. Player is prompted to enter her name. 2. The player enters her name and presses enter. 3. The player is shown the high score list with his name and score on it.
Alternative flow of events:	None.
Exceptions	None.
Extensions	1. Show the high score list.

Figure 5.7: The Use Case Specification [Arlow & Neustadt 2002].

The use cases of a software can be identified by starting with the list of actors and considering how each actor is going to use the system. Each candidate use case should be named descriptively so that the name includes a verb - the use cases describe action. Use case modeling is iterative and proceeds via stepwise refinement. First the use cases are identified and on the later iterations they are described in more detail. [Arlow & Neustadt 2002]

5.4.2 Rapid Prototyping

Prototyping can be a good way to discuss about the software requirements with the end-users [Tripp et al. 2004]. A software prorotype can be designed for an expert user representing all the end-users or for wider testing by the users of the software. The users produce most of the feedback in the prototype tests. The risk of having to readjust the software to better accomodate the needs of the users after it has been developed can be lowered by being in constant interaction with the end-users throughout the development [Krief 1996]. The use of prototypes has been standard procedure in many engineering and manufacturing industries [Charette 1989]. In software engineering the prototyping process is a relative fresh concept. There's no risk in mass production in software engineering as software can be duplicated effortlessly. The development risk, however, is comparable to other industries. Without prototyping the risk of developing software that does not correspond to end-user requirements rises. [Davis 1992]

There are two kinds of prototypes ([Budde et al. 1991], [Crinnion 1991], [Krief 1996]). Prorotypes can either be used as a part of the requirements specification of the software to be built or they can itself be the early versions of the software or parts of it. In the first case the prototypes are thrown away; their code will not be used as the basis for the software [Krief 1996]. This prototypes are called throw-away prototypes and the method of using them rapid prototyping. Prototypes that form the basis of the software to be built and which are developed incrementally are called evolutionary prototypes and the process of using them in software development is called evolutionary prototyping [Budde et al. 1991].

Working models of systems can support the communication between users and developers in ways that are not possible with mere word descriptions [Budde et al. 1991]. Prototyping offers a possibility to validate and adjust the requirements [Tripp et al. 2004]. The production of throw-away prototypes is an informal process in which the most important thing is that the prototype can be produced fast and cheap [Andriole 1990]. Prototypes can be divided according to their fidelity. The fidelity of a prototype defines how literally the prototype imitates the functionality, interaction and appearance of the software being developed. One form of low-fidelity prototypes are paper prototypes [Snyder 2003]. The production of higher fidelity prototypes like functional user interface prototypes demands more resources.

The main advantage of rapid prototyping is that feedback about the software being developed can be gathered at a very early stage of the development project without consuming large amounts of resources towards this end ([Budde et al. 1991], [Krief 1996]).

The validation of requirements is also much esier with throw-away prototypes: in a traditional waterfall development model one cannot be absolutely sure about the validity of requirements until the end-user can test the complete software product [Budde et al. 1991].

Rapid prototyping has a few potential problems. First, prototyping may lead to insufficient requirements specification [Pressman 2000]. The task of the designer is to make design decisions when building the prototype. If these decisions are not made consciously and made transparent in the prototype tests the user may not be able to validate them or give alternatives to them. This is why another means of requirements description and validation, for example use cases, are a good compliment to rapid prototyping. Second, the developers can get too attached to their prototypes [Andriole 1990]. If the prototype is meant to be thrown away, it should under no circumstances be used as a basis for the code of the development of the final product [Rollins & Morris 2004]. The temptation for that might be high but if the development is done in an organized manner and started from the scratch the programming work will comply with agreed work practices and the code will be easier to manage. In rapid prototyping manageability and good work practices are not priorities as the goal is to build these prototypes fast and cheap. Third, the clients might underestimate the resources needed for the actual development of the software product based on the time and resources used to build the throw-away prototype [Pressman 2000]. The difference between a working software product and low-fidelity prototype might be difficult to explain to a client.

5.5 Architecture Design

Software architecture has emerged as a crucial part of the software design process. It ties the requirement engineering phase to the first and most crucial design decisions together allowing software projects to plan their products and process on the basis of customer and end-user needs. Software architecture describes the components of the system and their external features as well as relationships between the components. [Bass & Kazman 2003] It can be used for communicating with the different stakeholders of a software development project (i.e. end-users, customers, developers, managers etc.), to divide the workload of the project and most importantly, to evaluate the software's quality and functional properties before implementing it.

5.5.1 Describing Architectures

Architecture must be described in a detailed and understandable view in order to be used as a basis of communication between the stakeholders of the software [Brodlie et al. 1992]. Dai et al. [Dai et al. 2006] suggest that using graphical notation is the most productive way to provide a description of a software architecture, as graphical notations provide the effective presentation of significant system features, enhance human understanding of concepts and processes, and serve as a medium of communication and collaboration.

There are three categories of architectural descriptions that can be specified by the notation used. These are informal, semi-formal and formal notations. Informal notations use natural languages accompanied by block diagrams to describe architectures. These kinds of descriptions are easy to understand and convenient to use. Their main disadvantage is ambiguoity which is caused by the lack of formality of the decription method [Dai et al. 2006]. Many sources including the more authoritative architecture design handbooks use this kind of architecture description (see for example [Bass & Kazman 2003])

A well-known semi-formal notation used to describe architectures is the Unified Modeling Language (UML, [Object Management Group 2003]). UML is a graphical language with semi-formal syntax and semantics. UML has emerged as the de-facto standard in software modeling and its graphical representation is relatively easy to understand and to learn. [Dai et al. 2006] UML, because its semantics are defined in a natural language, lacks the ability of formally representing a system's specification.

Formal notations, such as Petri Nets, Z and architectural descrription languages (ADL) etc., have both formally defined syntax and semantics. This allows rigorous tool supports for automated analysis. They have the advantages of conciseness and completeness, but lack in readability and are considered difficult to produce specifications with.

Architectural descriptions describe the architecture structure using architectural views. An architectural view is a coherent set of architectural elements. It is made by system stakeholders, usually designers, and the intended audience is all members of the system's stakeholders. An architectural view consists of a set of architectural elements and their relations. [Bass & Kazman 2003] No single view of the architecture is the architecture itself. The architectural structure is the set of the elements described by views and implemented in software and hardware.

The structure of an architecture can be discussed in many ways depending on the viewpoint. Bass et al. have defined a ten item list of different architectural structures,

of which the following five are of use in describing the architecture for the uses of architecture design:

- Module structure, units of which are work assignments and have products of the work associated with them.
- Conceptual or logical structure, the units of which are abstractions of the system's functional requirements and the relations between units show with which unit every unit shares data with,
- Process structure, which deals with the dynamic aspects of a running system. It's units are threads. Relations are represented by links that include synchronizes-with, cannot run without, cannot run with, pre-empts or any other relations dealing with process synchronization or concurrency,
- Physical structure, the mapping of software into hardware,
- Uses structure, which shows which procedures or modules use which modules.
- Calls structure, which shows that which procedures in the software call other procedures. The call structure is used to trace flow of execution in a program.
- Data flow, which describes which modules may send data to other modules.
- Control flow, units of which are modules or system states and it shows the relationships between them as the change of control or which module becomes active after which one. May be identical to the calls structure.
- Class structure. Shows the decomposition of the static system into objects and the inheritance and instancing relationships between them.

The views that are chosen to describe the software architecture are chosen by the needs of the project and the type of the software.

UML has two main types of diagrams: structure and behavior diagrams [Pressman 2000]. Structure diagrams depict the static structure of the system and behavior diagram depict the dynamic run-time behavior of the system. The UML notation has five major views which are used to describe the software from different perspectives (see chapter 5.4.1): user model view, structural view, behavioral view, implementation model view and environmental model view. These can be used to describe software architecture structure. The UML analysis modeling focuses on the user model and the structural

model and the behavioral model must be included to describe the dynamic aspects of the architecture [Pressman 2000]. The use case model (see chapter 5.4.1) is used to describe the functional requirements and the relation of individual requirements to each other. Analysis class model can be used to describe the static structure of the architecture whereas activity model can be used to describe the dynamic structure [Arlow & Neustadt 2002]. The UML models meant to be applied to design of object-oriented software [Arlow & Neustadt 2002]. Their application in different kinds of software design projects is less feasible.

5.5.2 SAAM

Software Architecture Analysis Model, SAAM, was developed in 1993 at the Software Engineering Institute (SEI). Its goal is to verify basic architectural assumptions and principles against the documents describing the desired properties of an application. Additionally, SAAM highlights the risks related to a specific software architecture by highlighting potential trouble spots. SAAM can also be used to compare alternative architectural solutions. [Dobrica & Niemelä 2002]

The SAAM evaluation consists of six steps (Fig. 5.8). However, not all evaluations consist of all those steps. Here we will describe how SAAM was applied in this study. First step is to develop scenarios that illustrate the kinds of activities that the system must support and the kinds of changes that it is anticipated in the system. To develop these scenarios all possible data of possible uses of the system much be collected from the stakeholders of the system. At this evaluation we concentrated on collecting the scenarios that involved the activities that are not yet implemented on the software but are likely to be required on later stages.

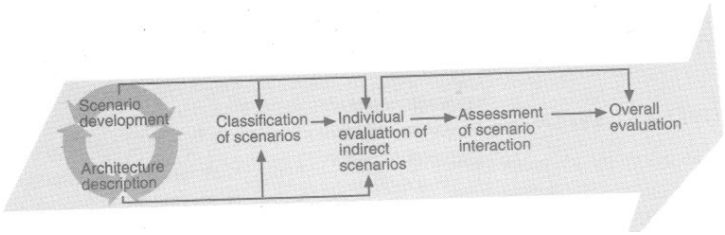

Figure 5.8: Steps of the SAAM evaluation [Dobrica & Niemelä 2002].

Second step of SAAM is to describe the architecture in architectural notation. The description must indicate the system's computation and data components and all

relevant connections. The architecture description and notational figures can be found in the Results chapter. After that we classify scenarios to direct and indirect scenarios. Direct scenarios are scenarios that can be implemented with the described architecture and indirect scenarios are those that require some modifications to the architecture. In this case we will only have indirect scenarios. The fourth step of SAAM is performing the scenario evaluations. In scenario evaluation the indirect scenarios are evaluated on what modifications they command in the system architecture. In addition to the list of components that must be modified in each scenario each scenario should be assigned a variable that illustrates the weight of these changes so that larger values indicate more drastic modifications. This weighting is usually coarse grained, depending of the understanding of the architecture.

The fifth step is to reveal scenario interaction. When two or more indirect scenarios require changes to a single component of a system, they are said to interact in that component. Scenario interaction is important to highlight because it exposes the allocation of functionality to the product's design. If semantically unrelated scenarios interact in a component it is a sign that the component performs two semantically separate functions. This can lead to problems in modifiability. If interactions are perceived, it can lead to three kinds of hypotheses: 1) the scenarios involved are similar and are in fact variations of the same basic scenario, 2) the scenarios are dissimilar but the component can be further subdivided so that each scenario is mapped to its own subcomponent or 3) the scenarios are dissimilar and the component cannot be further subdivided. In the first two cases there are no sign of problems but the last case indicates a potential problem in architecture. The final step is the overall evaluation. This is used to compare two or more scenario alternatives on their compliance with the scenarios.

5.5.3 ATAM

The creators of SAAM at the Software Engineering Institute (SEI) have later developed another software architecture analysis method, the Architecture Tradeoff Analysis Method (ATAM). Kazman et al. [Barbacci et al. 1998] define ATAM as follows:

"Architecture Tradeoff Analysis Method (ATAM) is a structured technique for understanding the tradeoffs inherent in the architectures of software-intensive systems"

The method is developed to provide a structured way to evaluate a software architecture's suitability in comparison to multiple competing quality attributes: modifiability, security, performance, availability and so forth. These quality attributes often inter-

act in way that improving one of them may come at a price of worsening the others. [Barbacci et al. 1998] The method identifies tradeoff points between quality attributes, facilitates communication between stakeholders from the perspective of each attribute, clarifies and refines requirements and provides a framework for an ongoing process for system design and analysis. [Barbacci et al. 1998]

Like SAAM, ATAM has both social and technical aspects. Technical aspects deal with what kind of data is to be collected and how it is analyzed. Social aspects deal with the interactions among the system's stakeholders and area-specific experts and create a social framework to enable their communication. ATAM is a spiral model of design. It consists of four phases and six tasks (fig. 5.9). ATAM is similar to spiral process model for software development in that each iteration takes designers to more complete understanding of the project.

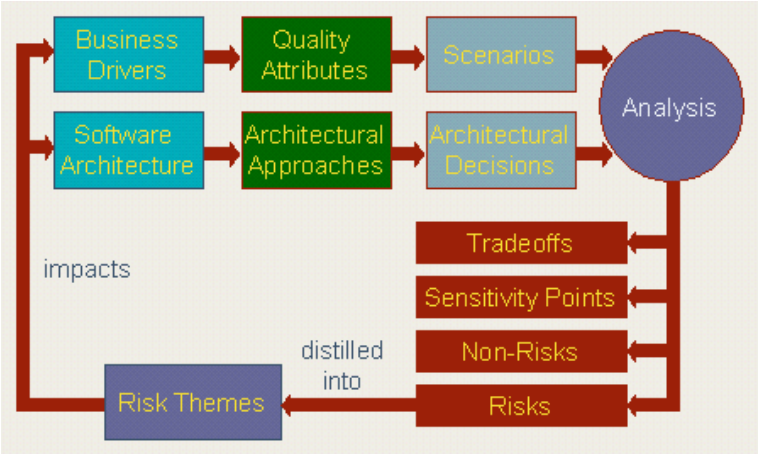

Figure 5.9: Phases and tasks of ATAM [Barbacci et al. 1998].

First scenarios are collected in a similar fashion as in SAAM (see chapter 5.5). The second task is to identify attribute-based requirements, constraints and environmental details of the software. The kinds of requirements are identified are based on what are the most important quality attributes of the software to be designed. For example, if the software will be performance critical the requirements relate to performance. Constraints and environment are characterized because subsequent analyses and constraints on the design space affect attribute analyses. The first two tasks can be carried out in any order or concurrently. [Barbacci et al. 1998]

The third task is to describe architectural views of the architecture candidates for the software. Architecture candidates are generated from the scenarios and re-

quirements identified in previous tasks. Additionally, legacy systems, needs for interoperability and experiences from previous projects can offer a basis for a architecture candidate and restrict the design space. The architecture candidates must be described in terms of their architectural elements and properties that are relevant to each of their important quality attributes. Common techniques for describing different architectural views and their uses are in chapter 5.5.1.

After the architecture candidates are described, each quality attribute must be analyzed in isolation with respect to each of the architecture candidates. The analysis is done by experts and the different quality attributes are identified as precisely as possible. For attributes like performance and fault-tolerance this can mean that a mathematical equation of certain scenarios response time is formed as a basis of analysis and this equation is modified as per different architectural solutions. For less quantifiable attributes such as modifiability, different analysis methods must be used. The articles describing ATAM ([Barbacci et al. 1998]) do not provide specific instructions on performing those analyses as earlier work is done on analyzing individual quality attributes of a system (for example [Klein et al. 1993], [Smith & Williams 1993]).

The fifth task is to identify sensitivities in the attribute analyses. A sensitivity is identified when one or more features of the architecture are varied and that variation results in a change in the results of attribute analysis. For example if using two concurrent servers does not affect performance compared to one server but enhances availability, the option of concurrent servers is sensitive to availability but not to performance.

After the sensitivities in architectural design decisions are located the architectural tradeoff points can be identified. Architectural tradeoff points are the architectural elements to which multiple quality attributes are sensitive. To continue the earlier example about two concurrent servers, if the concurrent server solution would increase availability but reduce maintainability, that element would be a tradeoff point between availability and maintainability.

When all these tasks have been carried out, it's time to compare the results of analyses to the requirements of the software. If the prevailing architectural candidate meets the quality attribute requirements the design can go into more detail. If the requirements are not met, the following iteration will stay in the same level of detail, more candidate architectures will be introduced and the scenarios and requirements can be re-examined, too. In both cases the new iteration will consist of the same tasks. This iterative process is intended to be followed through the course of the design task

of the software project.

6 Developers and Teams in Development of Games for Learning

In this chapter, the results of the study concerning the human resources needed in the development of digital game-based learning environments are discussed. This includes the description knowledge needed to complete a development project as well as the exploration and assessment of different possible ways to construct a development team out of the stakeholders of the project. The subject of human resources is important on two levels: First, it is necessary to be sure that one has the necessary resources to complete a development project and second, it is necessary to use those resources in an efficient way; that means maximizing the quality of the end product with the resources available.

6.1 Research Questions concerning Developers and Teams

This chapter tackles the first part of the research question: what kind of team is needed for a games-for-learning development project and how should this development team be best structured. This question can be broken down to the following subquestions:

- what kind of knowledge, expertise, skills and experience is needed in the development of games-for-learning?
- how to form the needed people in to functional teams so that the development process would be of high quality?
  - how to balance the need for continuous collaboration with people of different knowledge and expertise with the performance advantage of having separate teams for separate functional tasks?

The literature review presented in chapters 3, 4 and 5 provides a general answer to the first question: a multi-disciplinary approach is necessary in order to complete a games-for-learning development project. The multi-disciplinary team needs to have knowledge and expertise on pedagogy, learning theory, instructional technology design, software development, game design and development as well as the content area of the

game. As interaction with practitioners (i.e. teachers, policy makers, developers etc.) is also essential for collaborative construction of workable intervention, knowledge and expertise in user-centred design is also needed. This is the high-level statement of the expertise needed in the development of games-for-learning and the study described in this chapter gathers experiences from the case projects to form more detailed statements of the knowledge needed in the development.

The second question discusses the team structure and the arrangement of the development team. As mentioned in chapter 3.4.1, in design of any kind of education technology there is a need for cross-disciplinary collaboration in every phase of the development in order to avoid technology-pedagogy mismatches [Hedberg & Sims 2001]. On the other hand the participation of every member of the team in every task of the development is hardly optimal use of the development resources. There are approaches to solve this in the entertainment games industry (see chapter 4.1) and software engineering discipline (chapter 5.1).

These questions are tied to the process and tasks of development of game-based learning environments, which is discussed in chapter 7. In this part of the research, a very rigid division of the separate tasks in the development of games-for-learning is used: the development of a game for learning is considered to constitute a pre-production and a production phase. Pre-production is the act of forming the basic concept of the developed product and production is the task of building that product. The in-depth discussion of the impact of different combinations of collaborative and isolated tasks and the experiences what would be a best combination of them will be presented in chapter 7.

6.2 Developers and Teams in the Case Projects

Developers and teams was one of the focuses in all four case projects in this study: Gameli 1.0, Gameli 2.0, Social Responsibility Game and Peatland Adventure. In each of these projects a different approach was taken into the examination of knowledge and team structure of the games-for-learning development (Table 6.1). The team structures in the case projects are described in the sub-chapters (chapters 6.2.2, 6.2.3, 6.2.4 and 6.2.5) followed by a description of experiences and remarks made about the team structures used and their impacts on the project.

Table 6.1: The Focus of the Case Projects in Developer and Team Area

Project	**Focus**
Gameli 1	explore the traditional software engineering project team structure
Gameli 2	explore the impact of adding the responsibility of learning and game development experts compared to that of a subject matter expert's in traditional software engineering projects
Social Responsibility Game	further investigate game development and design expertise and game development process and methods expertise on a learning game development project
Peatland Adventure	to expand on the findings of previous case projects and experiment with a more flexible team structure

6.2.1 About Roles and Teams

In this chapter the role and functional team definitions used in the team structure descriptions of the case projects are described. The role definitions describe a single developers responsibilities and knowledge in the project whereas functional team definitions are used to define responsibilities for a group of developers. The role and functional team descriptions are adopted primarily from the software engineering and game development disciplines (see chapters 4.1 and 5.1).

Speaking in a high level, a person in the development team may either have a responsibility related to the project process and its outcomes or not. Those with direct responsibility to the project are members of the project team and those who do not are part of the support group. The responsibility to the project is categorized to responsibilities of 1) the project outcome, 2) the project conditions and 3) the goal of the project. In addition to responsibility to the project, a person related to the project may have responsibility to review the project outcome and provide evaluation or feedback on it, to offer his or her knowledge to the use of the project and to offer insight on the subject matter and/or on the context in which the product is to be used in.

According those categorizations we can find the following roles from software engineering and game development teams: project manager, client, developer, asset producer, tester, expert user, consultant and (future) user (Table 6.2). The person whose responsibility is to define the project's goals and manage the project's conditions so

Table 6.2: The Roles and responsibilities in a Learning Game Development Project

Team Role	Responsibilities					
	Project Goals	Project Conditions	Project Outcome	Reviewing Outcome	Provide Expertise	Provide Insight
Project Manager	x	x	x			
Client	x	(x)	X			
Developer		(x)	x			
Asset Producer			(x)			
Tester				X		
Expert User				X	(x)	x
Consultant				(x)	x	
User	(x)			(x)		x

that the developers can work, and who is also responsible of the project meeting its goals (outcome), is called the project manager. Client is responsible for setting the project's goals with the manager, and is responsible for reviewing the end product for its acceptance. He/she is also indirectly responsible for project's conditions through making enough resources available to the project.

Developer is the main worker in the project and mainly responsible for producing the project's outcome or some part thereof. Developer may also be responsible for arranging some project conditions in a productive way. The asset producer role is similar to the developer role, the difference being that the asset producer is responsible for designing and developing assets requested by a manager and/or developers that will be integrated to the product outcome (e.g. graphics or music).

A person (usually working with the development team) that is responsible for reviewing the project outcome compliance to project's goals is called a tester. Expert user role is similar to the tester, the difference being that the expert user is a representative of the end product's user group and is responsible also for offering knowledge on the use context and/or subject matter of the product. A member of the users of the end product who review the product or provide input on its essential development goals is called the user. A person working with the product team that has responsibility on providing insight or knowledge on one or more subjects is called a consultant.

Consultant may also be responsible for reviewing the project outcome from his/her knowledge perspective.

The knowledge of the developers is be described individually from his or her role. The description method used in this study is a list of knowledge and experience of the various aspects relevant for games-for-learning. This includes knowledge on the design of instructional technology, the knowledge of learning theories and pedagogy, the knowledge on the content area of the game, user-centred design, the knowledge on software engineering and game development or any methods within those disciplines. The method of acquiring a description of a developer's knowledge and experience was pretty straightforward as it could be deduced from the records of his or her earlier experiences on related disciplines (e.g. previous development projects and studies), his or her own assessment from the interview and the assessments of other developers.

The functional teams used to describe the structure of the case projects are the management team, the pre-production team and the production team. These are used to further specify a the role of a specific person in the development team. A membership in the project management team implies responsibility and say in the project goal specification and the management of project conditions. The division of the development to pre-production and production is used to allow for a description of a time or causal relationship or shifting of responsibilities between the phases of the project. Pre-production team is responsible for specifying the developed product whereas the production team is responsible for producing that product. This is not necessarily a straightforward time relationship where pre-production always precedes production. Instead, the tasks of these two teams can overlap and/or follow each other in a cyclical fashion. This temporal relationship between responsibilities and tasks is discussed in chapter 7.

6.2.2 Gameli 1.0

The Gameli 1.0 project had a team structure similar to the software engineering team structure described in section 2.1 (Fig. 6.1). The main production team consisted of five developers of which each one served as a team leader for a portion of the project. The project management duty was handled by the management team which consisted of the client and the current team leader. The production team had the possibility of consulting three experts that the client provided. The project had a class of sixth-graders, aged 12 to 13, as participants in three different sessions (group interviews and two prototype evaluation sessions) during the course of the project. The developers

had expertise on software engineering, project management (Table 6.3). One developer had also expertise on game design and graphic design. The development team did not have any learning or subject matter expertise, but experts on learning and the subject matter of the learning game were available for consulting.

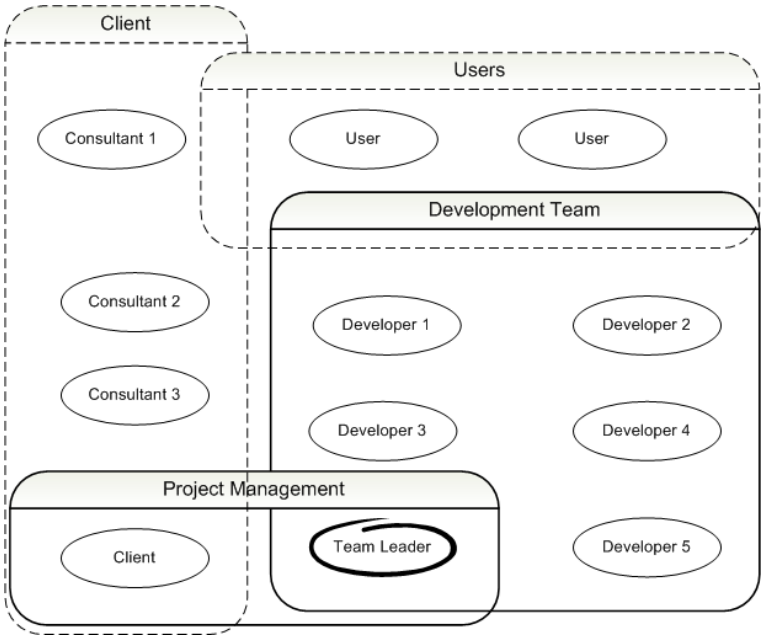

Figure 6.1: The Team Composition in the Gameli 1 Project

The production team had their doubts about developing a game for learning as only one of them had experience of game development and even that was for entertainment games. This was natural but unavoidable as experience on learning game development and literature on the subject was not available at the time.

Game development and graphics design expertise were noted to be key assets for the production team. After the project it was noted that the lack of simulation game mechanics balancing / level design expertise combined with the fact that only one member of the development team had previous game design expertise proved to be a hindrance to the team schedule-wise as level design took more time than was planned.

The main issue with this project was the availability and usage of the learning and subject matter expertise. The development team felt that they could not peruse the learning and subject matter experts enough and in the project review report the aforementioned experts were not happy on their role either.

As regards the involvement of users in the process, the role division within the development team was rather clear-cut: specific team members had the principal re-

Table 6.3: Expertise in the Gameli 1.0 Project

Person	Expertise
Developer 1	Software engineering (design, development), project management
Developer 2	Project management, software engineering (testing, usability design)
Developer 3	design (user interface, game level design), idea generation
Developer 4	Project Management, project communication, test design
Developer 5	Software engineering (design, development), game design, game asset production (graphics)
Consultant 1	user-centred design, user-centred methods
Consultant 2	Learning, subject matter (ecology)
Consultant 3	Software engineering (process, design, methods)
Consultant 4	Learning
Users	Insight on context, use experiences

sponsibility of planning, carrying out and analyzing the outcomes of the activities conducted with the users. The team considered it a problem, however, that they did not have much prior experience of working with children, and felt that they would have benefited of more training about these issues earlier in the project. Moreover, the involvement of users in the process was seen by the development team as a challenge requiring them to pay especially careful attention to scheduling and time resources, and consequently forcing the project process to become more dynamic.

6.2.3 Gameli 2.0

The Gameli 2.0 project's team structure (Fig. 6.2) differed from the structure of Gameli 1.0 team in the roles that the client side of the project management team were different. This time it consisted of two producers who had an active role in the pre-production and production. These two producers were actually two of the consultants from the Gameli 1.0 project. The change was made to allow for more active participation of these individuals into the project. Otherwise the development team structure was similar; the only other difference was the availability of teachers as another group of

design participants.

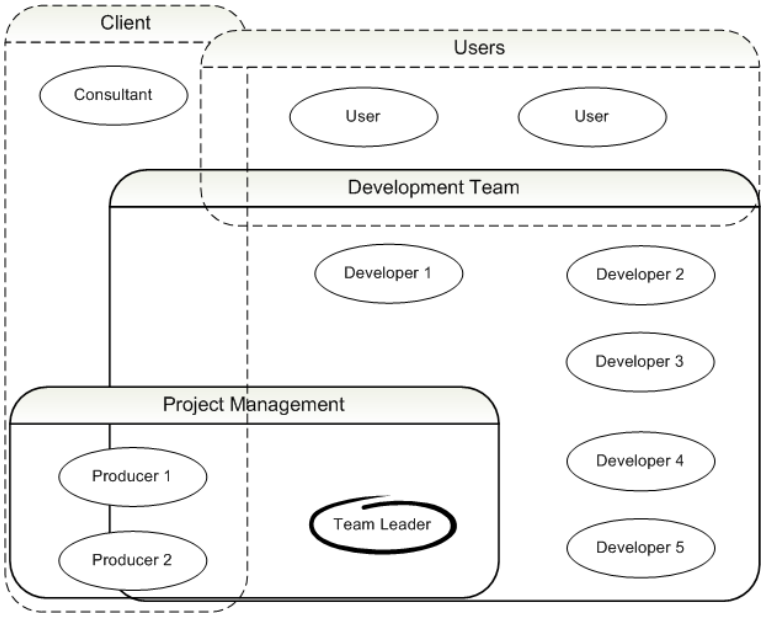

Figure 6.2: The Team Composition in the Gameli 2 Project

The Gameli 2.0 production team had expertise on software engineering. One member of the group also had expertise on graphic design (Table 6.4). The development team's expertise was broadened by the expertise of the producers, who had expertise on learning, game design & development and software engineering. It is also notable that these producers acted as consultants in the earlier Gameli project so they had more experience on learning game software development projects than in the previous project.

The developers noted that the expertise on learning and insights about the use context of the application from the future end users were crucial for the development, especially at the pre-production phase. Graphics design expertise was also deemed important for the project.

The Gameli 2 production team did not have as much game design expertise as Gameli 1 project. However, the project was more focused on developing the game playing / creation environment further and improving its usability in the classroom, and therefore – as commented by the developers in the post-mortem interview – the lack of this kind of expertise was not a problem.

Table 6.4: Expertise in the Gameli 2 Project

Person	Expertise
Developer 1	Software engineering: user interface design, game asset production (graphics)
Developer 2	Software engineering: project management, requirements specification, test engineering
Developer 3	Software engineering: sw design, development
Developer 4	Software engineering: sw design, development, project management
Developer 5	Software engineering: sw design, development, user interface design
Producer 1	Learning (theories, instruction), subject matter expert (ecology)
Producer 2	Software engineering, game design
Consultant	User-centred design
Users (teachers)	Instruction, insight on use context
Users (pupils)	Insight on use context

6.2.4 Social Responsibility Game

The team structure of the Social Responsibility game (Fig. 6.3) was similar to that of Gameli 1.0 project where the stakeholders and their roles are concerned. The focus of the project being in the production of a game design, the tasks of pre-production and production teams were different than in other projects, namely game concept generation and game design generation, respectively. The main difference in the composition of the development team is in the future users involved in the development. This game was not targeted to schoolchildren and so they were not involved in the project. Instead, the development team gathered a group of users from the target audience of the game to evaluate and provide feedback on their game concepts.

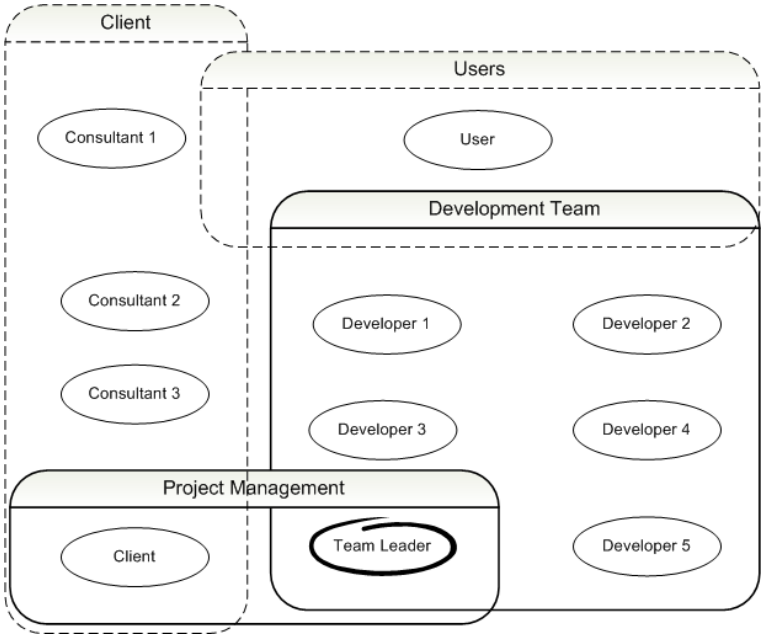

Figure 6.3: The Team Composition in the Social Responsibility Game Project

The pre-production team which outlined the game concept consisted of the production team, the client from AGL's partner organization and consultants from AGL. The production team had only introductory experience on game development & design, but they had a lot of experience on digital games (Table 6.5). The beginning of the project was spent on getting a better understanding of game design with the supervision of the game development consultant. Two developers had graphic design expertise. None of the developers had experience on the subject matter, but the client was available to provide their expertise on those matters during pre-production.

Table 6.5: Expertise in the Social Responsibility Game Project

Person	Expertise
Developer 1	graphics design, game design*, games
Developer 2	game design*, games
Developer 3	game expertise, project management
Developer 4	game design*, game, usability design, test engineering, project management
Developer 5	game, game design*, usability design, test engineering, graphics design
Consultant 1	user-centred design, user-centred methods
Consultant 2	Learning(Theories, Constructing learning environments)
Consultant 3	Software engineering (process, design, methods), game design
Consultant 4	Usability design, game usability studies
Client	Game design, subject matter
Users	Insight on context, use experiences
	** The developers had only introductory expertise from game design prior to the project.*

The potential problem in the Social Responsibility game expertise-wise was the lack of game design expertise. The view of the team members was that the tutoring on game development methods was crucial to the success of the project, but more experience inside the production team on the methods and process of game design & development would have resulted in a better use of project resources and therefore a more polished game design.

6.2.5 Peatland Adventure

The team of The Peatland Adventure game was structured in a different way to explore a possibility of a more flexible team composition (Fig. 6.4). The pre-production team consisted of only two developers and a manager with a role that could be considered to be between a manager and a client in terms of responsibility. This means that the managerial responsibilities of the project were divided between the development team and the manager. At the production stage a third developer was added into the mix. However, the role of the third developer was more on the technological side. At still a later stage a graphic designer was added to the project. The project team also had

access to two consultants and a user participant group consisting of pupils of a primary school.

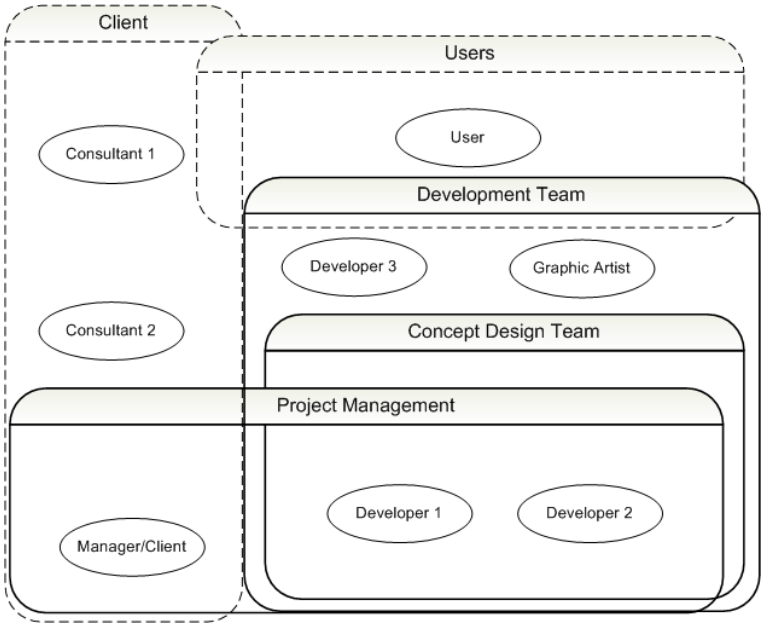

Figure 6.4: The Team Composition in the Peatland Adventure Game Project

The two original developers had expertise on learning, subject matter, game design and software engineering (Table 6.6). One of them also had experience on past learning game development projects (Gameli 1 & 2). The third developer had past experience on learning game projects as well as software engineering and game design expertise. As the core development team had most of the expertise they needed the consultants were used mostly to provide feedback on the development process and its products.

Before the project, ideas for the game had been gathered with upper secondary and elementary school students, and these ideas were drawn upon in the project. Students of the same school levels also evaluated the prototype of the game.

The project had the expertise it needed and the project management was able to add more experts into the project when needed. Against this background it is not unexpected that neither the developers nor the review report had much to criticize the project in terms of expertise and roles. Although in the beginning of the project the views of the learning / subject matter expert and the game design expert differed a lot, the small development team structure and the early stage of development enabled them to exchange views and form a common concept to work on according to the interviews conducted later. Even as the learning / subject matter expert's role in the

Table 6.6: Expertise in the Peatland Adventure Game Project

Person	Expertise
Developer 1	Learning (theories, instruction), subject matter (ecology)
Developer 2	Software engineering, game design, multimedia design
Developer 3	Software engineering, game design
Graphic Designer	Game asset design (graphics)
Consultant 1	user-centred design, user-centred methods
Consultant 2	Software engineering (process, design, methods), game design
Manager/Client	Not applicable
User	Insight on context, use experiences

project became smaller in the course of the development, this was reported not to be a problem as the other original developer became a champion of the original shared concept and was able to communicate it to the rest of the team.

6.3 Guidelines for Developers and Teams

This chapter discusses the experiences and discoveries made in this part of the study in more detail. The knowledge needed in the development of games-for-learning is discussed in chapter 6.3.1. After that, each type of team structures explored in the case studies is assessed according to the questions of enabling the effective work and collaboration of experts of different disciplines and the optimal use of human resources in the project.

6.3.1 Knowledge Needed in the Development

The second subquestion of the research question discussed in this section was 'what knowledge, expertise and experience is needed to develop games-for-learning?'. In this chapter, the answer is provided and formulated from the literature presented in chapters 3, 4 and 5 and the experiences of the case projects.

The purpose of a learning game is to provide an experience that facilitates learning of some subject area and to be interesting and rewarding from its gameplay aspects. So, to put it simply, the expertise needed to develop a game for learning is the expertise to design instruction technology and the expertise to develop a game. As the discussion here concerns digital games, software engineering expertise can be considered an

important part of the game development expertise. In addition to those, expertise on the content area that is treated in the game is needed ([Prensky 2001], [Gander 1998], [Stubbs & Pal 2003], [Squire et al. 2004]). The expertise needed in learning game design is described in figure 6.5.

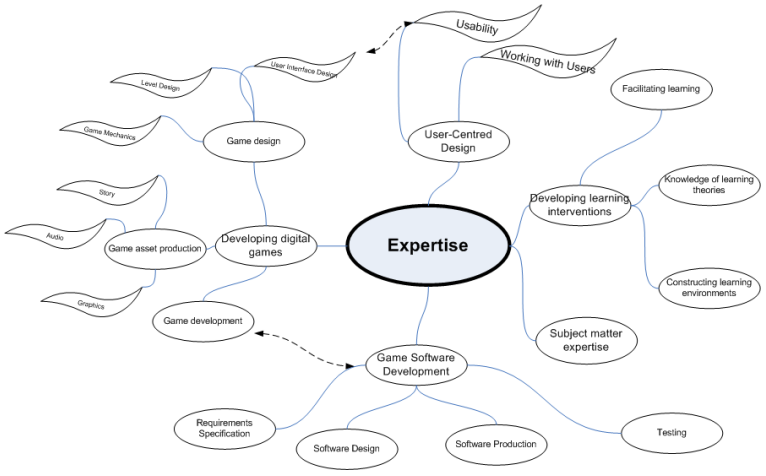

Figure 6.5: The Expertise Needed in a Games-for-Learning Development Project

The expertise to develop learning interventions can be further categorized in knowledge on learning theories, the expertise to facilitate learning and the expertise to construct quality learning environments. The importance of learning theories is not emphasized on all the studies about learning games: some reports suggest that the use of games itself affects the nature of learning and no additional ruminations on the learning theories are needed (e.g. [Prensky 2001]). On the other hand, there are plenty of reports that indicate the importance of the expertise on learning facilitation ([Lainema 2003a], [Squire et al. 2004], [Egenfeldt-Nielsen 2004]). It is important to be able to design structures and processes to accommodate the playing of games to other activities such as reflection [Squire et al. 2004] as well as taking account the needs of teachers and the nature of context the game is to be used in [Squire et al. 2004]. Especially in exploring the context, user involvement plays a crucial part. User-centred design attempts to take the context into account in a broader sense than traditional approaches, putting emphasis on the real environment into which the application must fit instead of relying on simplified ideas of the users and their tasks [Karat 1997].

The experiences from the case projects tell a similar story: The main deficiency in the Gameli 1.0 project was reported to be the lack of learning knowledge in the development team (in addition to the lack of content area knowledge) whereas the Gameli

2.0 project was reported to be a success in part because of the learning theory knowledge provided by one of the producers and the practical pedagogical insights coming from the teachers involved in the project. The collaboration between the developers of learning theory (and content area knowledge) and game development expertise were also reported to be crucial success factors in the Peatland Adventure development.

The expertise in game development is discussed in detail in section 3.2. In development projects of games for learning the following notions have already been made. The importance of gameplay design expertise is well-documented in the literature: games for learning should be challenging, engage users with activity and offer a good gameplay experience ([Chamberlin 2003], [Stubbs & Pal 2003], [Squire et al. 2004]). The interface of the game is important: it should adhere to usability standards and provide feedback to the player ([Chamberlin 2003], [Stubbs & Pal 2003]). Interface issues may lead to the game being hard to learn and adapt to the use context ([Gander 1998], [Elliott et al. 2002]). It is noted that although high end graphics are not essential and games for learning need not rival entertainment games in production value, the graphic design and good graphics are important for the gameplay experience of a learning game ([Chamberlin 2003], [Elliott et al. 2002], [Gander 1998], [Squire et al. 2004]). Game story is reported to be important part of learning games [Chamberlin 2003]. However, not all games have elaborate stories, so the importance of game story design expertise depends on the type of game being developed [Rollins & Morris 2004].

The presence of game development and design knowledge seems to be important according to the experiences from the case projects, especially when the focus of the product is on the game-like qualities. In the Gameli 2.0 project the game design and development expertise was not considered to be such a big issue as the project was more focused on developing an environment for game design and development than creating a game. In the other projects game development and design knowledge was noted as one of the key assets. In Gameli 1.0 project some problems were faced because only one member of the production and pre-production team had prior experience from game development. That member became overtaxed during the project and some tasks were delayed because none of the other members of the team could share the responsibility on tasks related to that knowledge. In Social Responsibility Game project the lack of prior experience of game design and the game development process was considered a major challenge by the developers. Although this challenge was partly overcome by the extensive coaching by the consultants and the research of the subject by the developers they felt that they could have performed better with more prior experience. It was also

noted that the project itself offered enough experience on the subject to the developers that would have been better equipped to handle a similar project in the future.

Reports from earlier studies do not have many experiences on what kind of software engineering expertise is important on a learning game development project. It is noted that the configurability of the learning game is important, as it allows customization for different use contexts and facilitates maintenance ([Lainema 2003a], [Stubbs & Pal 2003]]. Evaluation with users is also considered a key point ([Chamberlin 2003], [Elliott et al. 2002], [Gander 1998]]. The lack of experiences from the software engineering aspect of learning game development is not a surprise; game development literature suggests that the development of game software is not an especially problematic sub-area of software development and other types of software may well be more state-of-the art than computer games [Rollins & Morris 2004]. This notion, combined with the experiences that learning games need not rival commercial educational games in production value [Squire et al. 2004], explains why this facet of learning game development has not been considered problematic.

The case projects were were quite strong in terms of software engineering knowledge. Therefore, especially taking into account the earlier reports regarding games-for-learning development, it was not a surprise that software engineering knowledge did not raise many problems or issues within the projects. There were, however, a number of smaller problems relating to applying the software engineering methods into the development of games-for-learning.

The knowledge of the content area of the game and learning with the game is structured around is important and the developers should acquire this expertise in some way. In some earlier projects the content area expert has been a developer in the learning game project ([Lainema 2003a], [Stubbs & Pal 2003]); in others, the developers have consulted content area experts ([Gander 1998], [Squire et al. 2004]). This was also noted in case projects. In the two Gameli projects, the story was the same as in the learning knowledge: in Gameli 1.0 the developers noted the lack of availability of content area (and learning theory) knowledge as the main deficiency in the project whereas in the Gameli 2.0 project the availability of content area knowledge was described as being crucial to the success of the project. In Social Responsibility Game the content area knowledge did not seem to produce any issues during the project or comments from the developers in the interviews or in the project reports. This can be a result of multiple factors: 1) the client/consultant of the project provided ample knowledge of the content area of the game in the pre-production phase of the product,

2) the content area covered by this game was not such a complex matter to grasp compared for example to the natural science content areas of Gameli 1.0 and Peatland Adventure projects and 3) the content area was much closer to every-day life than in other projects. In the Peatland Adventure Game the content area knowledge did not provoke many comments or problems either; this is most likely the result of there being enough knowledge on the content area in the course of the project from one of the main developers and the fact that the developer could use his/her knowledge in a beneficial way as he/she had experience from two previous game-for-learning projects and could use and transform her knowledge of the content area to be easily adaptable to the game development.

In addition to these issues, skills and experience of working with users and sound understanding of user involvement methods manifest as important issues in learning game design. As indicated e.g. by the comments of the development teams and findings from other projects [21], especially the lack of experience of working with children can present challenges for the development process.

Particularly in the Gameli 2.0 project the involvement of users in the development process was noted as beneficial. In the Gameli 1.0 project, working with users was noted to be problematic as the developers responsible with that task did not have experience on working with children and the tasks related to this (planning and carrying out the activities with users) were deemed time-consuming. The knowledge needed to introduce user-centred methods is two-faceted: one both experience and knowledge in working with the user group and the knowledge to plan the sessions so that they best benefit the development.

6.3.2 Roles and Team Structures

In this chapter the different types of team structures suggested by the literature and experimented in the case projects are discussed. The different team structure solutions are compared by the factors they bring into the quality development process of game-for-learning in terms of enabling the development team to collaborate in the way that produces the best possible end result and also the way that they try to conserve the team resources by making the development as efficient as possible. The three basic structures we will discuss is the software engineering team structure adopted for games-for-learning used in Gameli 1.0 and Social Responsibility Game projects, the software engineering team structure augmented with producer roles used in Gameli 2.0 project and the flexible game development structure used in Peatland Adventure Game project.

Traditional Software Engineering Team Structure

Two case projects, Gameli 1.0 and The Social Responsibility Game, explored some aspects of the traditional software engineering team structure. As the previous knowledge of these teams was mainly in the software engineering discipline, the traditional team structure used in software projects (see chapter 5.1) was considered to be a good starting point for exploring the optimal team structure in games-for-learning development. The team structure would of course have to be adapted to fit the games-for-learning development. In software engineering, the subject matter experts usually take the roles of expert users or consultants that are available for consultation by the developers. This arrangement was used in the case of learning theory and content area experts. As for game development knowledge it was considered that in light of the literature on game development the experts of game development and design should be more involved in the project. This was arranged by including game development, design and graphics production knowledge in the pre-production and production teams.

The more involved role of game development experts was experienced as beneficial for the projects. In the Gameli 1.0 project the developer with game development and graphic design knowledge was noted as a crucial part of the project team and there was even need for more game development experts as this one developer was noted to be too busy to be included in all the tasks that he/she would have been useful in. In the Social Responsibility Game project the lack of genuine game development and design experience was noted to be a challenge for both the development and project management. The project participants noted that if more experience on game development would have been available, it would have resulted in a more suitable plan for the execution of the whole project and the individual tasks in the project would have been easier to carry out. It was also noted that although the coaching the developers had on game development and design was valuable, it probably would not have been enough if the developers did not have earlier experience on game cultures and digital entertainment games.

Of the game asset production expertise the graphics design expertise was deemed the most important. Experiences from the Peatland Adventure project would suggest that the graphics design expert does not have to be in a responsible or active role in the beginning of the project, but can instead be added to the production team later in the project.

It was discovered that the way of using consultants for the learning theory, pedagogy and the content area knowledge was not optimal in the projects. It was noted that a

more responsible and direct approach was needed. In Gameli 1.0 project it was noted that a consultant's role for these experts was not sufficient.

In addition to these issues, skills and experience of working with users and sound understanding of user involvement methods manifest as important issues in learning game design. As indicated e.g. by the comments of the development teams and findings from other projects [21], especially the lack of experience of working with children can present challenges for the development process.

As for the optimal use of human resources in the project several drawbacks were faced using this team structure. In the Gameli 1.0 project the shortage of game development knowledge lead to delays in the project and to uneven division of work between project members. The lack of knowledge and planning how to apply and use the input of the users lead to frustration in the development team and into statements that the time spent with users was in part wasted even though the feedback gathered was noted as very important. The problems with game development expertise would not have been occurred if the needed expertise would have been available in the project. Problems on collaboration with the users could have been lessened, if the user-centred design consultant could have had a more influential role in the project process design. Having that consultant in project management team is one potential solution to this problem. Also the problem with non-optimal use of the learning theory and content area consultant described earlier in this chapter can be considered a problem of the use of human resources. The potential solutions are presented in the following paragraphs in this chapter: software engineering team structure augmented with producer roles and flexible team structure. In the Social Responsibility Game there were not that many comments on the non-optimal use of human resources. Some comments were made on the overlapping tasks of the game designers in the game concept generation phase and the need to specify collaboration between developers better, but those issues were not a result of the team structure but rather the development process.

Software Engineering Team Structure Augmented with Producer Roles

A step was taken in Gameli 2.0 project to increase in the responsibilities of learning and content area consultants to that of producers. The producers took a more active role in the project which can be seen in their inclusion in the project management team and increased presence in the pre-production and production teams. This was done solve the availability problem of learning and content area consultants in the Gameli 1.0 project. The promotion of the more active role in the project worked well as other members of the development team felt they could approach the producers easily. This

resulted in much more communication between the producer with learning and content area expertise and the rest of the production team than in the Gameli 1.0 project. Both parties welcomed this as the production team felt that they had adequate expertise in hand and the learning/content area expert was glad to see that the end product was up to her expectations.

There were not many comments of non-optimal use of human resources in the project. One concern heading into the project was the workload of the two producers. It was estimated that the producers would have to invest a lot of time into the project especially in the beginning but it was not sure if their involvement could be lessened slightly during the production phase. From the resources standpoint this was an important examination to make as the strength of the traditional software engineering and game development project models is that the project personnel can be divided to designers and developers who can do their work separately. This allows the projects to free up human resources not involved in the current phase of the project into other tasks in the organization. In this project, the workload of the two producers was noted to decrease somewhat as the project carried on to production. This was evidenced by more time between meetings between the producers and rest of the development team and less requests for comments and feedback from the other developers and suggestions for change in the project from the producers.

Flexible Game Development Team Structure

In the Peatland Adventure project a different approach to team structure was explored. This was due to the opportunity to form a team inside the research group instead of using students as the main body of the development team. This opportunity was used to explore development process that adopted more from entertainment game development than the software engineering discipline. The team structure used reflected this focus. On the previous projects of this study the pre-production and production teams have had roughly the same members. In this project a different approach was used. The pre-production team consisted only of two developers who had broad knowledge of all the necessary areas of games-for-learning development. They created the original game concept and kept on polishing the game design during the project. The production team was formed after the original concept had been finalized. The production team was formed according to the needs of the production task. The team structure can also be thought as a modification of the augmented team structure presented in previous section of this chapter, the roles of the producers being close of those of the two developers in the pre-production team in the Peatland Adventure.

This team structure was explored as an another way to overcome the drawback of the too inactive role for the learning theory and content area consultant experienced in the traditional software engineering team structure. This experiment was also a success as it was noted that there was enough expertise on those areas to produce a quality game for learning. From the positive experiences in this project it can be said that the active and responsible role of these experts is important especially in the pre-production team.

It was also noted that a relatively small pre-production team could be more efficient and cost-effective in human resource terms than a large one used in the other projects. The potential drawback is that it may be more difficult to communicate the product concept to those developers that are not part of the pre-production team and get them engaged to the project. This drawback did not manifest itself in the Peatland Adventure project as the main developers were able to communicate their concept clearly.

From user-centred design standpoint this project was not the most efficient as the users were not part of much of the pre-production process or the production process, although their input was one of the factors informing the concept-generation. This can also be considered as a waste of human resources available to the project as collaboration with users could have presented design alternatives and therefore affected the product's quality in a beneficial way. However, this is considered to be a issue mainly connected to the development process and it will be discussed in detail in chapter 7.

6.3.3 New Metamodel

In this chapter a metamodel for team structure in games-for-learning development is presented. The metamodel is based on the results of the study presented in the previous chapters: the knowledge needed in development of games-for-learning (chapter 6.3.1) and the experiences from different team structures (chapter 6.3.2). The results indicate that especially the knowledge of learning theories, pedagogy, game development and design, user-centred design and the content area of the game are needed in the development along with software engineering expertise. It was also noted that pedagogy, learning, game development and content area expertise should be actively involved particularly in the pre-production team. The project management team should include members with game development process and user-centred design knowledge. The production team should include game development and software engineering knowledge. Both the pre-production and production team should include user participants who

could provide insight and feedback for the development.

Based on this, the following is presented as a metamodel for team structure of games-for-learning development (Fig. 6.6). The development team consists of management, pre-production, production and design partner teams. Management team has a project manager and representation from the other teams. Pre-production team consists of developers that have expertise on learning, game's content area, game design and software engineering. The production team consists of the core group of pre-production team and additional asset producers and developers as needed. At least the experts of game design and software engineering from the pre-production team should be included in the production team also. The design partner team consists of a producer that has expertise on working with users and is responsible for arranging the user participation in the project and the future users of the project.

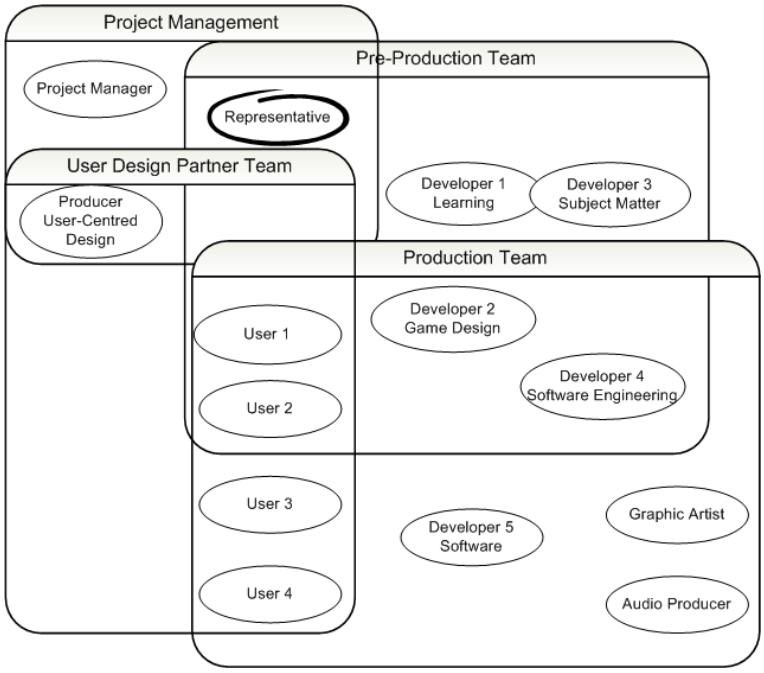

Figure 6.6: The Metamodel for Team Structure in the Development of Games-for-Learning

The inclusion of the design partner team is important especially at the pre-production stage of the project as evidenced in the experiences from both the Gameli 2.0 and Peatland Adventure projects. The team should also stay active during the later stages of the development as feedback on the product is important for the quality of the end product.

7 Modeling the Process of Multi-disciplinary Development of Games for Learning

This chapter deals with the tasks, activities and the work products in the development process of game-based learning environments. This the second part of exploring the process models in GBLE development. The first part of the process model exploration, focusing on team member roles and expertise, was described in chapter 6. The third and final part, focusing on process control, is described in chapter 8. The literature review chapters discuss the tasks, activities and work products in the related disciplines of learning intervention development (chapter 3.4), game development (chapter 4.2), human-centred design (chapter 1.4.2) and software engineering (chapter 5.2).

Process modeling is especially important in GBLE development as the work involves many distinct disciplines that all have their own process prescriptions. In practice each GBLE development project must plan how to fit the activities and tasks from different disciplines into their own project process. This planning could benefit from research on how to model a quality development process where activities and tasks from different disciplines are combined.

The research questions of this part of the study are:

1. What kind of development processes can be elicited in GBLE development?
2. How do these development processes compare with development process models in learning sciences, game development and software engineering?
3. What kind of prescriptive process models can be formulated for tasks and activities in GBLE involving more than one distinct discipline?
4. What kind of prescriptive general process model or metamodel could be constructed for GBLE development?

The first question is approached by eliciting the development processes of the case projects and describing their processes with the aid of a process modeling language.

The second part of this study compares the elicited development processes with development process models of the three disciplines that are combined in GBLE development: Learning Sciences, game development and Software Engineering.

The final research question considers the general guidelines for GBLE development process modeling. General guidelines are derived from the results of the previous research questions.

7.1 Process Models in the Case Projects

The results of process inspections of the four case projects are presented in this chapter. The presentation consists of a textual description with accompanying SPEM 2.0 process (see chapter 2.2.2) and activity diagrams.

7.1.1 Gameli V.1

The aim of the Gameli V.1 project was to develop an engaging learning game on top of the GameWorld modeling and simulation software developed by Centre of Information Technology for Education of the University of Hong Kong.

The project used an iterative modification of the waterfall process model where the design, implementation and testing were done in two iterations. The project also used game concept and design documents to structure the game design task. These documents were used as a basis for requirements specification document with prioritized requirements.

The work in Gameli project was divided into five phases: startup, research, game design, prototype 1 and prototype 2 (Fig. 7.1).

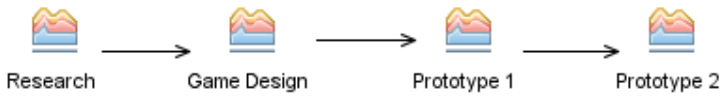

Figure 7.1: The phases of Gameli project.

The aims of the research phase were to collect requirements for the learning game and to generate game ideas to fulfill those requirements. The activities to fulfill those aims were analysis and concept creation, respectively (Fig. 7.2).

In game design phase one of the game concepts created in the research phase was elaborated in the Concept Refinement activity. After that the concept was evaluated against the requirements. The evaluation activity was followed by game design and

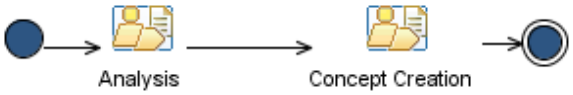

Figure 7.2: The activity diagram of the research phase of Gameli project.

software design activities where the plans of making the concept into a game were made (Fig. 7.3).

Figure 7.3: The activity diagram of the game design phase of Gameli project.

In the prototype phases the game and software designs were refined and the game and game software were implemented (Fig. 7.4). The resulting game version was formally inspected at the end of each prototype phase.

The analysis activity was planned according to the knowledge from learning intervention development (see chapter 3.4.2). The activity consisted of literature review of learning theory, examination of similar learning products and studies of the learning environment as well as future users of the game. The main end product of this activity was the requirements specification document of the Gameli learning game. The project group also produced a document on the literature review and the examination of similar learning products.

The concept creation activity (Fig. 7.5) was informed by human-centred design and game concept generation process theory. First, the project group studied the cababilities of the simulation environment the game was to be developed with and examined the results of an earlier HCD design workshop where the users had proposed game concepts to ccreate with the environment as well as commented on an earlier game created with the environment. The project groupintegrated the results of the HCD workshop to their concept creation process and created multiple game concepts to be further developed in the game design phase. The main work product of this

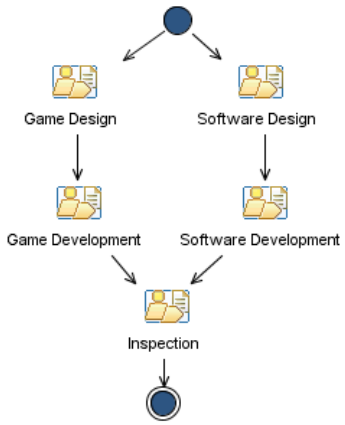

Figure 7.4: The activity diagram of the prototype phases of Gameli project.

activity was the game concept documents produced by the project team.

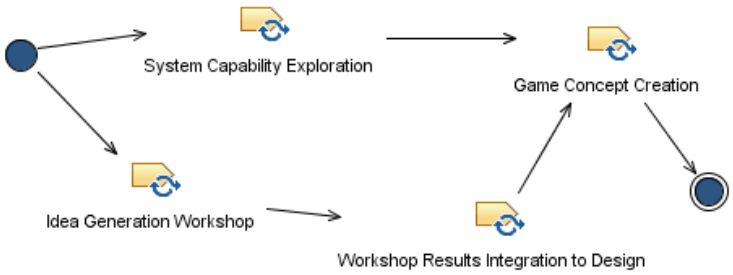

Figure 7.5: The activity diagram of the game concept activity of Gameli project.

Concept refinement activity was modeled after the game proposal creation activity in the game development discipline (chapter 4.3.1). The activity took one of the game concepts created in the previous phase as input. The end work product of this activity was a refined version of the game concept.

The evaluation activity of the game design phase consisted of formal evaluations of the game concept conducted by stakeholders of the project. The project group had planned to go back to the game concept refinement activity should the result of the evaluation be that the game concept would have been declined. This was not necessary as the game concept was approved with minor modification requests.

The inputs of the software design activity were the requirements document and the

refined game concept. The software design phase was modeled after software design activity in the sequential software engineering process model (see chapter 5.2.2). In the software design activity the project team identified the changes to the simulation software that would have to be made in order to satisfy the requirements and the the new features of the game concept. The software design activity was also informed by the concurrent game design activity. The work product of this activity was the software design document that included the class level plans to carry out the software changes. In prototype phases this design activity was re-iterated and the design document was refined based on changes in the game design.

The game design activity followed the task structure of the game design activity in the game development discipline (chapter 4.3.2). In addition to that the Gameli project constructed a prototype of their game design and analysed and refined it during the design activity.

The game design refinement activity in the prototype phases had the same format. The activity differed in its inputs: it was informed by analysis of the design and tests with prototypes and in the latter prototype phase, inspection of the first version of the game.

The inspection activity consisted of HCD evaluation of the game version and formal evaluation by experts and stakeholders of the project.

7.1.2 Gameli Version Two

The goal of Gameli V.2 project was to expand on the earlier Gameli V.1 prototype and to produce a simulation game play/design environment that could be used in classroom in natural sciences education.

This project used the same kind of process model as Gameli V.1 project: an iterative modification of the waterfall process model where the design, implementation and testing were done in two iterations. The focus however was more in the iterative and prototyping nature of the model: the requirements phase used rapid prototyping to learn about the product that was being developed.

The development project was divided to five phases: analysis, design, implementation 1, implementation 2 and delivery (Fig. 7.6).

The aim of the analysis phase was to analyse the requirements for the next version of the game application. The analysis phase consisted of the following activities: software adaptation, knowledge gathering, requirements elicitation and requirements analysis (Fig. 7.7).

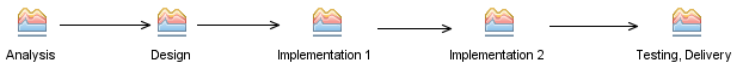

Figure 7.6: The phases of Gameli V2 project.

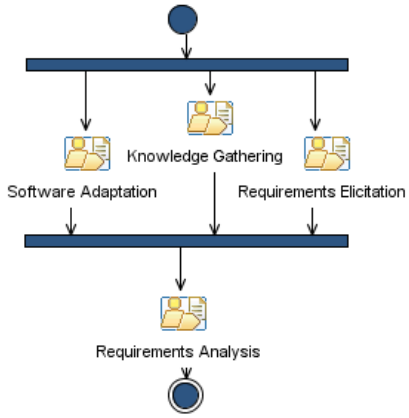

Figure 7.7: The activity diagram of the analysis phase of Gameli V2 project.

The goal of the software adaptation activity was to get the project group acquinted with the latest version of the Gameli software. The end product of this activity was the architecture description of the Gameli software.

The knowledge gathering activity consisted of literature reviews and workshops of human-centred design, learning theory concerning GBLEs and game design. The end product of the activity was knowledge of related disciplines available to the project team.

The requirements elicitation activity consisted of a HCD design solution creation workshop, definition of constraints regarding solution and workshop results integration to the requirements (Fig. 7.8). The requirements elicitation activity produced a set of requirements as a work product.

The requirements analysis activity took the set of requirements produced in the requirements elicitation activity as input. The main work product of this activity was the use case definitions of the Gameli V2 software, which were made based on the activity input, definition of constraints regarding the solution and definition of the learner characteristics (fig. 7.9).

The design phase consisted of UI and functionality design, software design and test

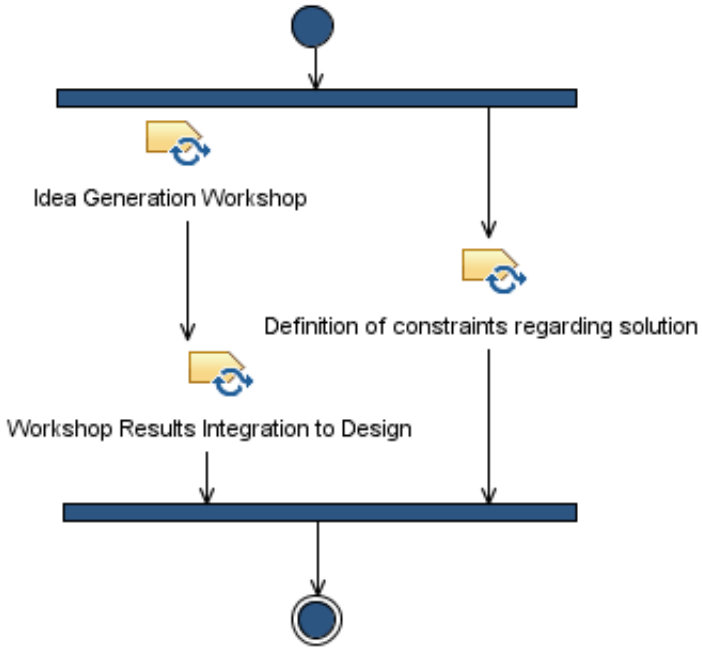

Figure 7.8: The activity diagram of the requirements elicitation activity of Gameli V2 project.

plan generation acivities (Fig. 7.10). The UI design activity consisted of making a draft of the user interface design, building it as a low fidelity prototype, testing the prototype with user designers and then refining the user interface and functionality (Fig. 7.11).

The test plan generation activity was planned as per the same activity in the sequential software engineering process model. The activity produced test plan which included plans for unit, system and integration tests for the software being developed.

The aim of the implementation phases was to implement an increment of the new software. The phase was carried out with two activities: software implementation and evaluation (Fig. 7.12).

The delivery phase consisted of HCD user evaluation of the software as well as a formal inspection (Fig. 7.13). Passing the both evaluations was a requirement to complete the project.

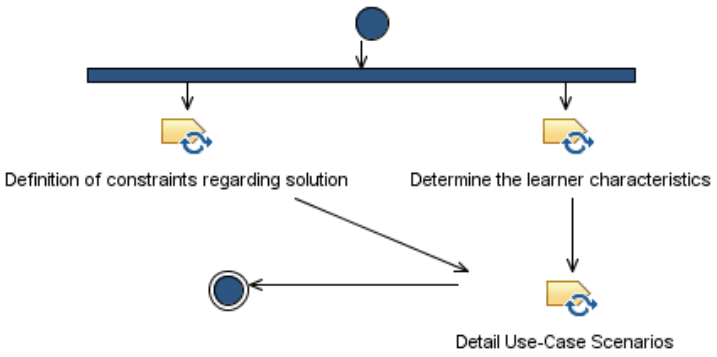

Figure 7.9: The activity diagram of the requirements analysis activity of Gameli V2 project.

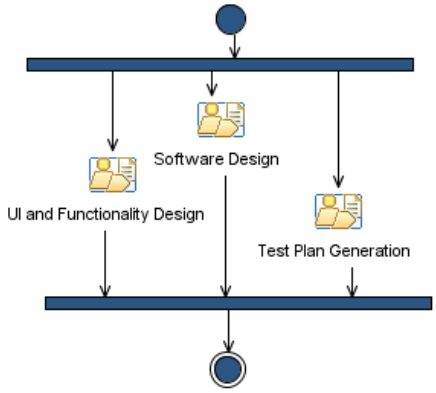

Figure 7.10: The activity diagram of the design phase of Gameli V2 project.

7.1.3 The Social Responsibility Game

The goal of the Social Responsibility Game (SR Game) was to produce a game design for a game that would communicate the social responsibility aspects of a Agora Game Lab's partner organization.

In this project the focus was solely in game design. This allowed game concept generation and design methods to be taken into close inspection. The overall process model used was a sequential model, although this time consisting only of the concept generation (or requirements) and design stages. Game design documents were used to document the game concepts generated and the game design work done throughout the project.

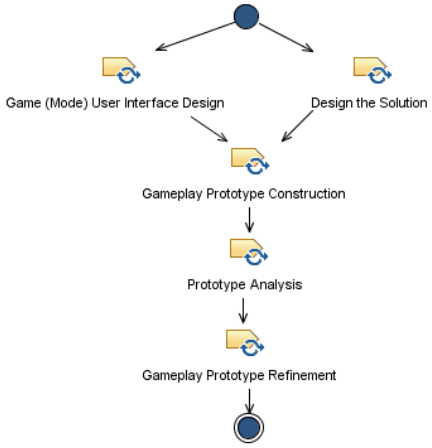

Figure 7.11: The activity diagram of the UI and functionality design phase of Gameli V2 project.

The SR Game project had three phases: concept, design and delivery (Fig. 7.14). The process of concept phase was modeled after process knowledge from learning sciences, ISD and game development and also experiences of previous projects. The activities of the concept phase were analysis and concept creation.

The analysis activity was modeled with knowledge of analysis process in the learning intervention development discipline (see chapter 3.4.2). The process for the activity was similar to the analysis activity in the research phase of the Gameli project (see chapter 7.1.1).

The concept creation activity was planned according to the game concept creation and game concept refinement activities of game development discipline (Fig. 4.3.1). The activity was enriched by adding a HCD concept assessment workshop where user designers assessed the game concepts the project group had prepared (Fig. 7.15). The game concept refinement task used these HCD assessment results as inputs in addition to the normal evaluation and analysis results. The work product of this activity was a set of refined game concept documents.

The aim of the design phase was to develop the set of game concepts produced in concept phase into one extended game proposal. This was carried out with concept refinement, game proposal preparation and game design activities.

In the concept refinement activity, The concepts were refined with the help of evaluation feedback and comments from stakeholders and experts. In this activity one

Figure 7.12: The activity diagram of the implementation phases of Gameli V2 project.

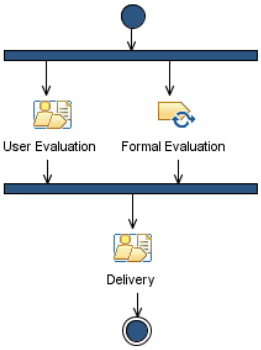

Figure 7.13: The activity diagram of the delivery phase of Gameli V2 project.

game concept was chosen as a basis of future design and selected features from other game concepts were added where applicable.

The aim of the game proposal creation activity was to produce a game proposal document. The game proposal document format was adopted from game development discipline (see chapter 4.4.2) and adapted for GBLE development with the addition of learning intervention design analysis chapter. The activity planning was based on game proposal creation activity of the game development discipline (see chapter 4.3.1). Tasks were added to the activity plan to implement the learning intervention design analysis (Fig. 7.16): The entry knwledge and learning environment specification tasks from ISD process were added for learning requirements specification work and an intervention design analysis task was planned to implement the actual analysis based on the learning

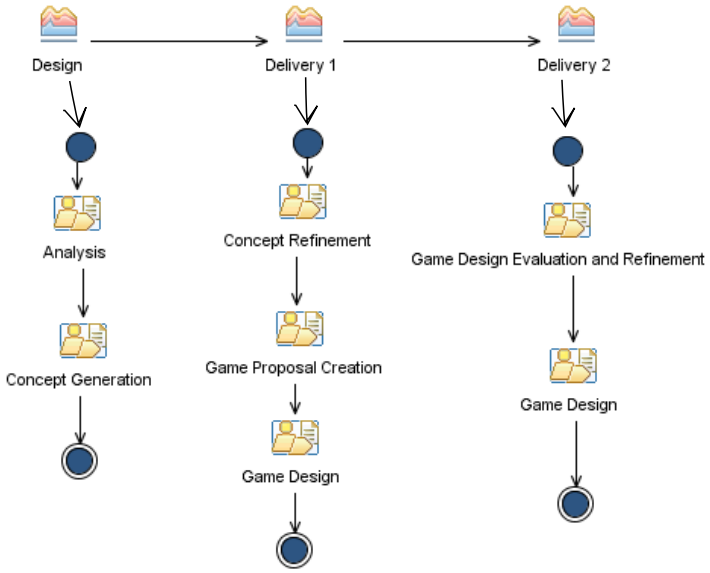

Figure 7.14: The phases and activities of the Social Responsibility Game project.

requirements.

The game design activity was modeled after the game development discipline activity, enhanced by aspects of iterative game design. The work product of this activity was a refined game design document. The game design document creation was informed by the prototype evaluations performed by the project team. The prototypes built demonstrated some of the key design problems the project group was solving.

The aim of the delivery phase was to refine the game design to better fulfill the goals of the game and to produce the complete version of the game design document. These aims were met in their own activities. The game design refinement activity had the same kind of structure as the game design activity in the design phase.

7.1.4 The Peatland Adventure

The Peatland Adventure is a web-based adventure game designed to support in learning natural science related to the peatland nature. It is designed to be used both in the school context and as a stand-alone game outside school for various user groups. It is a part of the Virtuaalisuo (Virtual Peatland) virtual environment (http://www.virtuaalisuo.fi/).

The development of this game started as a part of the development of the Virtuaalisuo virtual environment development. The original pre-production phase was very

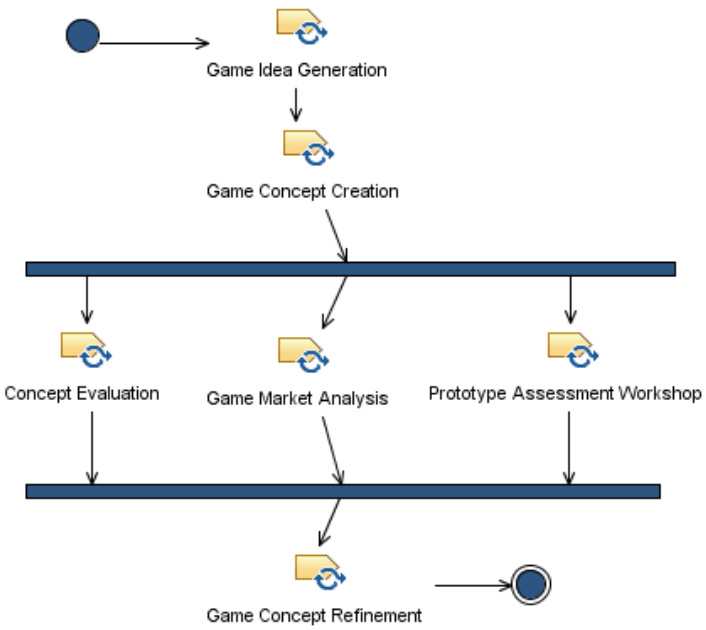

Figure 7.15: The activity diagram of the concept creation activity of the Social Responsibility game project.

long as the whole environment was specified at the same time. After the first analysis phase the game project was separated as a project of its own. The latter part of the project used a different approach to the project process: It experimented on a more flexible team structure that started as a small two-people concept design team and expanded when the project proceeded to further phases. The project benefited from a lengthy concept generation phase that preceded the project (Fig. 7.17). The concept generation phase included brainstorming possible game concepts to develop and testing and re-developing them with user participants.

The development process had features from both sequential and cyclical development model. One thing that made the project unique was the game genre: being an adventure game, the basic game mechanics did not involve a lot of design work. Instead, the level and game story design took most of the game design time. This allowed the team to adopt a kind of cyclical process where the game was developed one level at a time after the overall user interface and game look had been established.

The development process was divided into three phases: the initial concept creation phase and the second part of the development consisting of an analysis phase and a

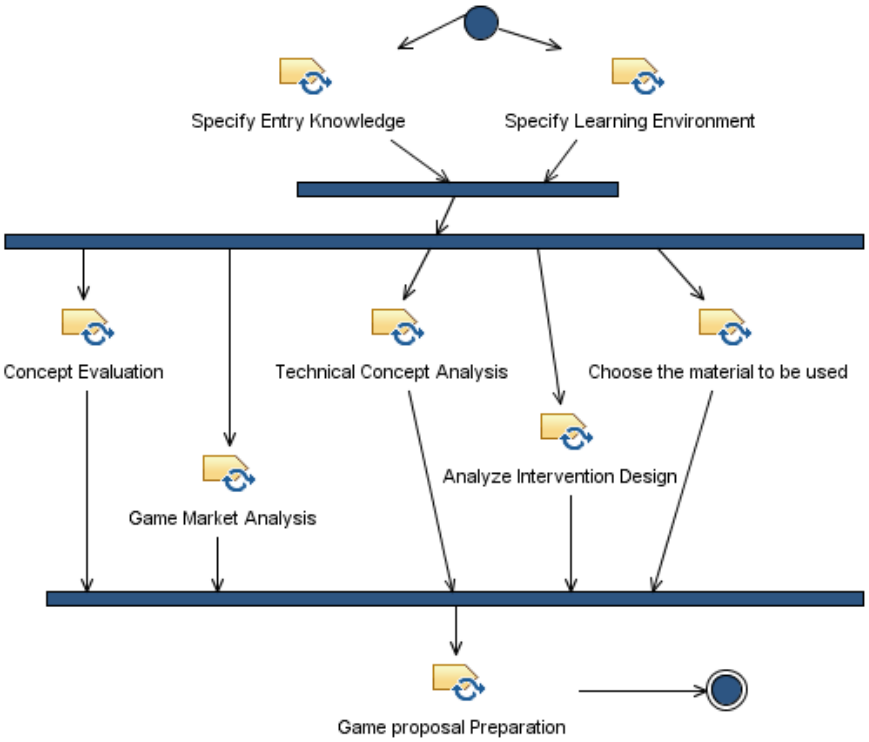

Figure 7.16: The activity diagram of the game proposal creation activity of the Social Responsibility game project.

design and development phase. The goal of the original concept creation phase was to specify the requirements for the game-based learning environment and create a game concept to fulfill those requirements. The goal of the analysis and design phase was, again, to set requirements of the game, develop a working theory and to create a game concept to build on. The analysis and design phase's activities were analysis, intervention design and level design.

The concept generation phase consisted of requirements specification, process design, concept generation and concept evaluation activities. The requirements specification activity was planned according to the requirements specification activity of software engineering (see chapter 5.4). The work products of the requirements specification phase were the user profiles document and the requirements specification document with functional requirements.

The process design activity consisted of planning the process of carrying out the

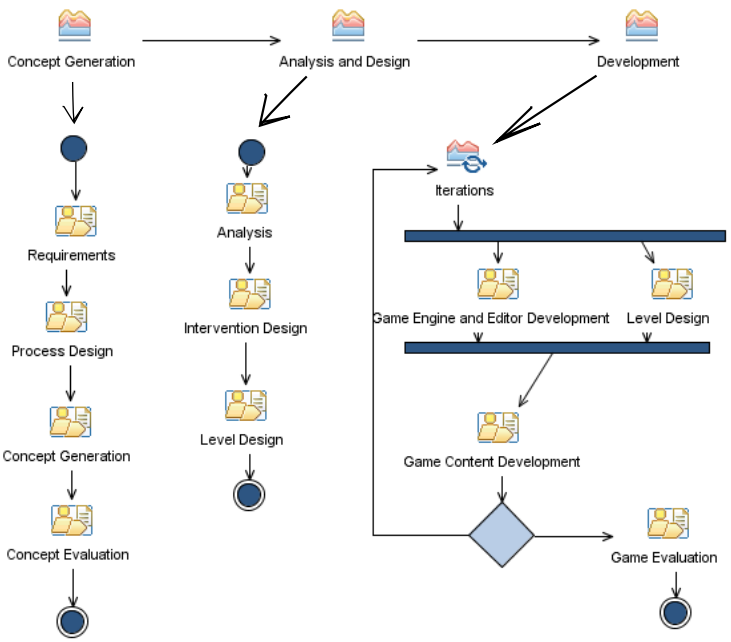

Figure 7.17: The phases and activities of the Peatland Adventure project.

development project. The work product of the process design activity was project process plan.

The concept generation activity was planned according to the game concept generation activity of the game development dicsipline (see chapter 5.4). The work product of this activity was a set of game concept documents. The game concept evaluation activity was planned according to the iterative game design process (see chapter 4.4.4) and HCD process from HCD discipline (see chapter 1.4.2). The input for the activity were the game concept documents. Prototypes of the game concepts were constructed and they were analyzed together with a group of user designers. The main work product of this activity were the prototype evaluation reports that were produced based on formal evaluation and HCD evaluation of game concepts and prototypes. In this project the conclusion of the reports was that none of the game concepts fulfilled the requirements set for the game.

The analysis and intervention design activities were carried out according to the process of learning sciences (see chapter 3.4.2). The intervention design activity also included tasks that produced the game concept: game idea generation and game concept generation.

The level design activity consisted of choosing the learning material to be used in

the level, designing the game story and creating the level design plan.

The development phase was divided into two iterations. A working version of the game was be produced in each iteration. Activities to support that goal were game engine and editor development, level design and game content development. These all were undertaken in an iterative manner and the results were integrated and tested throughout the phase. This approach was similar to iterative game design.

At the end of each iteration the game version underwent a formal inspection by experts and stakeholders. The feedback from the inspection was carried out to the objectives of the following iteration.

7.2 Evaluating the Processes of Case Projects

The experiences of the processes and their impact on the products of the process are presented in this chapter.

7.2.1 Experiences from the Gameli project

The overall process model followed incremental software life cycle model. The phases were enriched by activities and tasks from the learning sciences, HCD and particularly the game development discipline. The activities from game development discipline contained its own whole in the process. In contrast the learning sciences and HCD tasks were in a supporting role. The game design activities and documents were seen beneficial to GBLE design where it came to designing a engaging game.

HCD methods were not taken into account when first planning the project process. Because of that the analysis and integration of HCD workshop results into designs took more time than planned and caused the proect's schedule to slip. Some project workers and experts also noted that if an HCD workshop would heave been arranged at the first part of the project, the user designers could have had more impact on the projects outcomes.

The project group also felt that the project's goals were not defined at the start of the project clearly enough. On the other hand the learning sciences expert noted that the game's learning goals and the project's working theory was not defined clearly and that could have affected the qualities of the game regarding supporting learning. It is possible that this kind of problem could have been dealt with with process planning where the project's goals would have been described and prioritized multi-disciplinarily.

7.2.2 Experiences from the Gameli V2 project

For this project too, the overall process model followed the incremental software life cycle model. The process was planned according to experiences from the Gameli 1 project. This meant that the HCD and learning sciences activities and tasks were more closely knit to the software engineering and game development activities.

To specify the application's learning goals a HCD workshop with teachers was organized in the analysis phase of the project. The workshop was regarded as crucial to the goal definition of the project. The fact that the co-operation with this teacher HCD group did not continue throughout the project was seen as a weakness in the project process, although the project continued to work with student user-designers throughout.

The validation of designs through HCD workshops and stakeholder evaluations were seen important for the success of the project. Both could be used to provide information for actions in the project's later iterations.

As the projects goals were to build a toolkit for learning game development and modeling, the game development side of the process was not prevalent. The software engineering side of the process dominated and the process model was seen as functional. As the project was making a new version of existing software it needed to adapt the processes of software development to this task.

7.2.3 Experiences from the Social Responsibility Game project

This project used a sequential model of game pre-production as the overall process model. The process was enriched by by activities and tasks from the learning sciences and HCD disciplines. The experiences of Gameli project were taken into consideration and those activities were integrated as parts of the game pre-production process.

The main result in the project relating the process model was that the overall pre-production process model used in this project worked very well to produce a game design document for game-based learning environment that satisfied both game and learning requirements. It was noted that the iterative prototyping of the game designs could have been started earlier in the process to have more time to polish the prototype and maximize its quality.

The project also demonstrated the importance of clear goals. The projects goals were clearly set at the start and could be then used as basis for design decisions later in the project. This was noted as a clear advantage in the project.

7.2.4 Experiences from the Peatland Adventure project

The original pre-production phase of the project failed to produce a quality game proposal. This can be seen as a result of many factors: the project team did not have experience with the methods and processes used in GBLE pre-production, the pre-production phase did not have learning goals of enough precision to guide their work and the HCD methods used did not support the pre-production work.

These experiences were taken into account when setting up the analysis phase. The learning requirements and goals specifically for the game and not just the learning environment as a whole were defined and on the basis of those a game concept was created. This led to a concept that could be validated in regards to learning requirements.

The iterative and experimental model of work used in the project lead to a good product and efficient work by the project team. There are two factors to this: First, the game design work was modeled after iterative game design process model of the game development discipline and the results from this case project support the notion of game development literature that this process model allows for continuous refinement of the game throughout the process. Second, the core project group had experience in GBLE development and related disciplines as described in chapter 6.2.5.

7.3 Guidelines for GBLE Development Processes

The experiences in the case projects resulted in the following observations related to the GBLE development processes. The three main findings are described in detail in the following three chapters along with a prescription on how to take advantage of these findings in process planning of future GBLE development projects.

7.3.1 Integrating HCD Activities as Part of the Process

First, as was seen in the Gameli V1 project, it took planning work to integrate the activities of supporting disciplines like HCD to the development work. It benefited the later projects to include this in the process planning work. Methods used to integrate the activities tighter were basing and evaluating the game concepts on the learning goals and/or requirements documents (in SR Game and Peatland Adventure), adding learning plan and evaluation chapters in the game proposal documents (Gameli V2 and SR Game) and planning the project phases around HCD workshops (Gameli V2 and SR Game).

Incremental development, especially when planned so that it was integrated with HCD workshops between iterations, was seen as beneficial to tackle the experimental nature of GBLE development. To make this work the amount of work needed to analyze the evaluation feedback needed to be taken into account in the process planning.

What this means for GBLE development process planning for projects that see value in using HCD methodology is that the process needs to be modeled to take advantage of the feedback-cycle structure of HCD process. For each activity of development that is decided to take advantage of HCD methods, the following structure must be formed: The analysis tasks must include a HCD session where the requirements for the design are decided with the user-designers. The requirements and other work products of the analysis tasks must be validated with the user-designers. The design decision-making must be informed with user-designers, preferably with a UCD session or workshop. The design decisions must also be validated by the user-designers. During development the increments of the developed products must be validated byt the user-designers.

In practice all of this means that each of the analysis and design tasks are accompanied with associated HCD participatory analysis or design tasks as well as UCD evaluation tasks for each work product. It must also be taken into account that the HCD evaluation tasks can produce new findings about requirements or pressure to change designed or even developed functionality.

This suggests that incremental or iterative development where change control is more cost-effective or even agile development process could accommodate UCD methods easier than traditional process models that do not have many tools to deal with chance would be preferable. This is discussed in more detail in chapter 8.

7.3.2 Iterative Game Design Beneficial

Adapting iterative game design model was seen as beneficial in Peatland Adventure and SR Game projects. In case of SR Game project the opinion was that there should have been iterative game design activities from the earlier on in the project.

This finding coincides with the projects with a focus towards ambitious game design. In projects where there is uncertainty and possible risk in regards with the game design adapting aspects of the iterative game design process model can therefore be advised.

As described in chapter 4.4.4, the iterative game design process consists of iterations where some subset of the game is designed and a representative prototype is constructed and tested. The test results of the previous iteration then inform the planning and carrying out of the next iteration. Before each iteration the design problems to be

worked on during that iteration are decided and the tests to validate the solutions regarding the problems are planned. The areas considered to containt the most risks are worked on first. The process is iterated until the design is acceptable as a whole. The other end condition for iterative game design process is that the project stakeholders are not satisfied with the results and do not wish to assign more resources to solve problems in the design.

It must be noted that iterative game design does not guarantee success in terms of producing a game design of desired qualities as was seen in the original analysis phase of the peatland adventure project. It does, however, guarantee that the quality of the design can be assessed in more accurate way than the evaluation of game design documents provides and before resources are committed to the full-scale development of the game design.

7.3.3 Definition of Goals as a Potential Risk

The unambiguous definition of goals for the GBLE to be developed was seen as a potential risk in all the case projects. This was tackled with experimenting with different kind of game concepts, HCD workshops, elicitating requirements, literature reviews and reviews of other promising GBLEs.

Two of the most promising solutions for this was demonstrated in the Gameli V2 and the Peatland Adventure projects. Gameli V2 project, in short, defined their working theory, held a HCD workshop to elicitate requirements for the intervention that would support the earning of the subjects in question and went on to analyze the set of requirement for the GBLE. In Peatland Adventure the learning requirements were set after an experimental concept phase and they informed the game concept creation.

To overcome this potential risk in GBLE development projects the project must evaluate if the goals of the project are defined unambiguously or can be further specified to be unambiguous with reasonable effort. If that cannot be achieved tasks must be planning to work towards setting unambiguous goals at the beginning of the project. These can include HCD workshops, requirements elicitation methods as well as literature reviews and reviews of promising similar products, as mentioned before.

8 Process Control in GBLE Development

This part of the study focuses on the process control in the game-based learning environment development process. Project management has three main responsibilities: making a plan to deliver the desired product, monitoring the work and reacting to emerging problems ([Novak 2005], [Sommerville 2001]). These three responsibilities are combined in process control ([Ogunnaike 1994], [Schwaber & Beedle 2001]).

The research questions of this part of the study are: 1) What kind of process control model is most advantageous and feasible to the development of GBLE (both in general and especially in those projects concentrating on game design and human-centred design) and 2) what kind of adaptation of that process control model would lead to a high-quality development process in this domain.

The two main approaches of process control in software engineering discipline are defined process control and empirical process control. Defined process control is presented in the traditional software project management literature [Schwaber & Beedle 2001]. Empirical process control is implemented in the more iterative software process models such as the spiral model and the Scrum process model [Rising & Janoff 2000]. These two process control models are discussed in chapter 5.3.

The results of this part of the study are presented as follows: first the case projects are described along with their characteristics realted to process control and experiences noted related to the chosen approach in process control in each of the projects. After that the key activities in GBLE development, identified by previous study, are discussed in the light of the two process control models. Based on these results conclusions are drawn into the benefits and drawbacks of the two process control models compared in the study for GBLE development. Finally, guidelines are drawn for combining empirical process control, exemplified by the Scrum process, with the key activities of GBLE development.

8.1 Process Control in Case Projects

In this chapter the four case projects are described in terms of goals and overall process. Their process control model is then discussed and the experiences of the chosen process

control model and its impact on the project are described.

8.1.1 Gameli V.1

The aim of the Gameli V.1 project was to develop an engaging learning game on top of the GameWorld modeling and simulation software developed by Centre of Information Technology for Education of the University of Hong Kong.

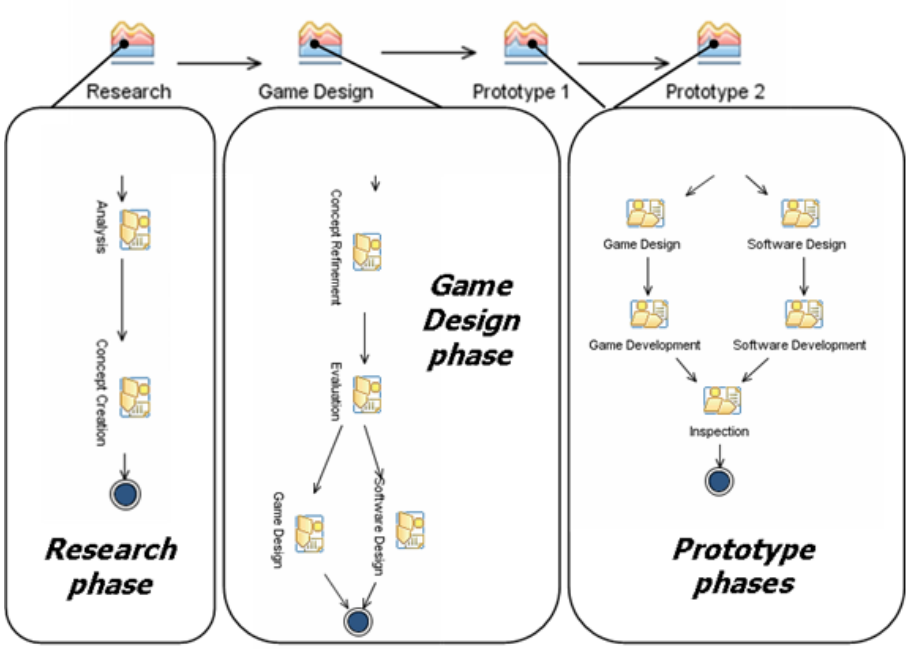

Figure 8.1: Diagram of the phases and activities in the Gameli project.

The project used an iterative modification of the waterfall process model where the design, implementation and testing were done in two iterations. The project also used game concept and design documents to structure the game design task. These documents were used as a basis for requirements specification which in itself consisted of a specification document with prioritized requirements.

The process control model in the Gameli project was defined process control. Each phase of the project had planned activities which were expected to have the assumed results and a set schedule. At the end of each phase the work of the project was inspected by the steering group for compliance to the plan. The plans for later phases were revised where applicable informed by the experience on the activities gained from project work but this was limited by the strict planning before the project and limited resources available in the middle of the project.

The project team members noted that many of the original estimations of the workloads of the planned activities were found as inaccurate: *"Most notable risk in this [game design] phase was the tight schedule. Calculated with planned resources the risk did occur."* [1]

In addition to that the early concept creation and subsequent HCD evaluation activities called for change in the project plans as well as requirements which had to be discarded as the project had not planned to deal with that much change and did not have the resources to implement the change demands when they were encountered. Small changes were made to the project plans where possible but the project reviews indicate that neither the project team members or other stakeholders were satisfied with the project's ability to change its course in the midst of the project.

This inability to deal with change could probably have been lessened by some degree with risk management but these experiences indicate that the inherent drawback of defined process control of not coping with uncertainty or change is also present in GBLE development projects.

8.1.2 Gameli V.2

The goal of Gameli V.2 project was to expand on the earlier Gameli V.1 prototype and to produce a simulation game environment/simulation game design environment that could be used in classroom in natural sciences education.

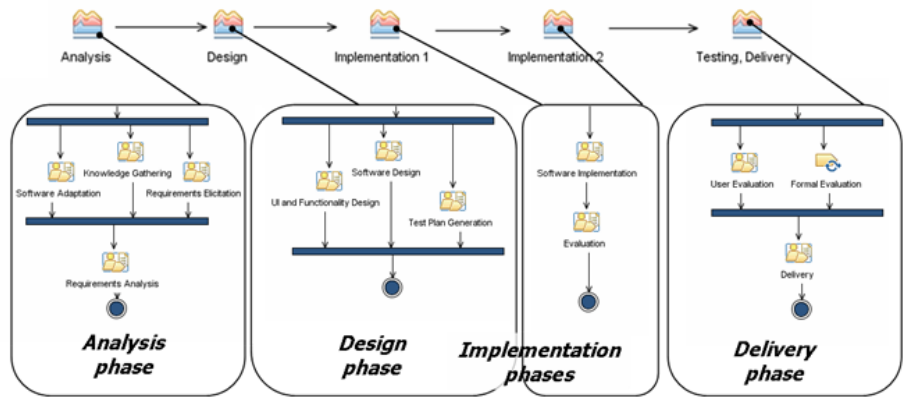

Figure 8.2: Diagram of the phases and activities in the Gameli V.2 project.

In this project the experience from earlier projects was taken into account when deciding the development process and the methods used. This project used the same

[1] From Gameli V.1 Game Design Phase Report, translated by author

kind of process model as Gameli V.1 project: an iterative modification of the waterfall process model where the design, implementation and testing were done in two iterations. The focus however was more in the iterative and prototyping nature of the model: the requirements phase used rapid prototyping to learn about the product that was being developed and the first implementation iteration was light enough to allow for significant changes after testing. This was done also to allow for more significant role for users in the development process.

The second Gameli project also used the defined process control model. The process design was altered slightly to allow for more flexible adaptation to changing requirements and findings during the HCD activities by defining the goals for each phase in a more general manner. This allowed the team to expand on the goals after they had the results from HCD activities: *"The information gathered at the [HCD] teacher workshop vas extremely valuable in specifying the requirements."* [2]

This however lead to more uncertainty as the team was not sure if they had resources to complete the goals with the activities planned because the activities had been planned before the goals were specified in high fidelity: *"The change control was certainly an issue at that moment. There were a lot of possible routes to take and it was evident that everything could not be included."* [3]

This uncertainty did not manifest itself in actualized risks as the project group was able to prioritize the goals they set for each phase and complete the at least the goals that were crucial for the completion of the project but still the risk of not having enough resources to complete crucial goals was present.

8.1.3 The Social Responsibility Game

The goal of the Social Responsibility Game (SR Game) was to produce a game design for a game that would communicate the social responsibility aspects of a Agora Game Lab's partner organization.

In this project the focus was solely in game design. This allowed game concept generation and design methods to be taken into close inspection. The overall process model used was a sequential model, although this time consisting only of the concept generation (or requirements) and design stages. Game design documents were used to document the game concepts generated and the game design work done throughout the

[2] An excerpt from the Analysis Phase Report, translated by author

[3] A comment from a developer in the post-mortem interview, transcribed and translated by the author.

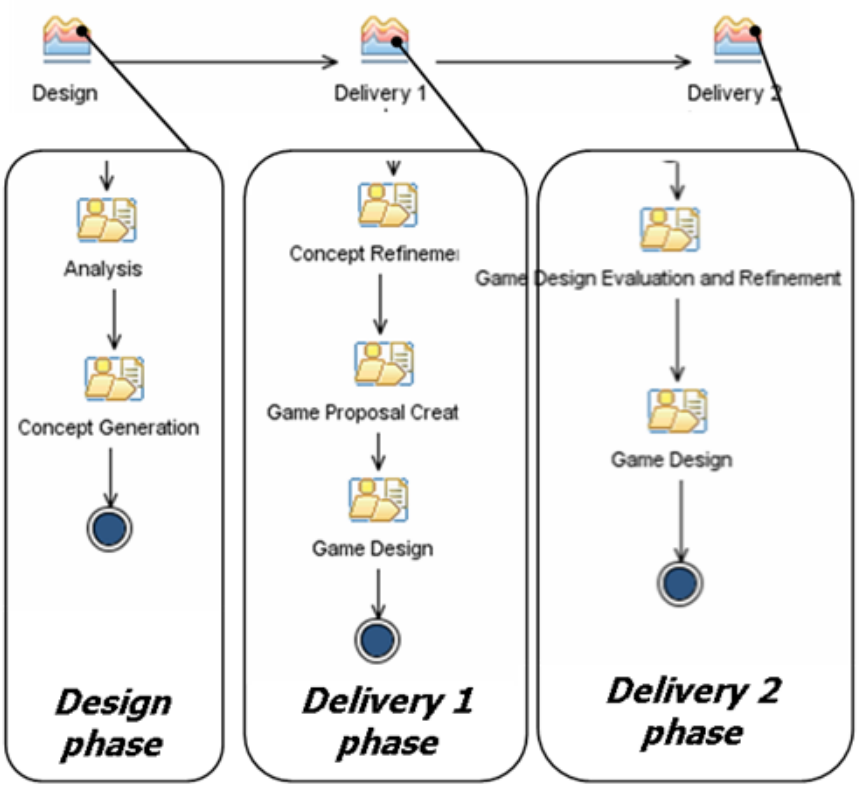

Figure 8.3: Diagram of the phases and activities in the Social Responsibility Game project.

project. The methods used in game concept generation included brainstorming about the game's subject matter, harvesting ideas from existing games, using game design patterns as inspiration and creating competing game concepts. The game concepts were evaluated by testers from the target user group of the game as well as expert-evaluated. The game design was based on using the game design document but included also gameplay prototypes and expert evaluation.

The process control model of the project was a mix between defined and empirical process control. The project had a fixed set of phases with distinct responsibilities, but the activities were not planned in detail beforehand. Instead the project group used the phases as iterations to expand the game design work and explore the different possibilities and used the inspections between phases for feedback and advice on the planning and goal-setting of the next phase. In the end a game design document was produced and in fact the project had time to revise the created document for one increment before ending the project.

The project reviews indicate that all stakeholders considered the project successful. The project team members commented that they could have validated the design decisions made in the process better if they had made and evaluated more gameplay prototypes during the project: *"The prototyping of the game should be started as early as possible."* [4] This argument was supported by some of the experts that reviewed the project.

8.1.4 The Peatland Adventure

The Peatland Adventure is a web-based adventure game designed to support in learning natural science related to the peatland nature. It is designed to be used both in the school context and as a stand-alone game outside school for various user groups. It is a part of the Virtuaalisuo (Virtual Peatland) virtual environment (http://www.virtuaalisuo.fi/). The focus in the design of the game was to make a motivating and interesting game that would provide the players with positive attitudes towards the peatland nature and environment in addition to providing the content-specific knowledge.

The development of this game started as a part of the development of the Virtuaalisuo virtual environment development. The original pre-production phase was very long as the whole environment was specified at the same time. After the first analysis phase the game project was separated as a project of its own. The latter part of the project used a different approach to the project process: It experimented on a more flexible team structure that started as a small two-people concept design team and expanded when the project proceeded to further phases. The project benefited from a lengthy concept generation phase that preceded the project. The concept generation phase included brainstorming possible game concepts to develop and testing and redeveloping them with user participants. The development process had features from both sequential and cyclical development model. One thing that made the project unique was the game genre: being an adventure game, the basic game mechanics did not involve a lot of design work. Instead, the level and game story design took most of the game design time. This allowed the team to adopt a kind of cyclical process where the game was developed one level at a time after the overall user interface and game look had been established.

The Peatland Adventure project used different manifestations of the empirical process control model. The first, used in the original requirements phase, was a model

[4] Comment from an interview with one developer, translated by author.

driven by the HCD process. The idea was to iteratively elicitate requirements for the game an create game concepts that were assessed by the HCD design collaborators. This phase was terminated when the concepts created in the final iteration were assessed as not fulfilling the requirements gathered.

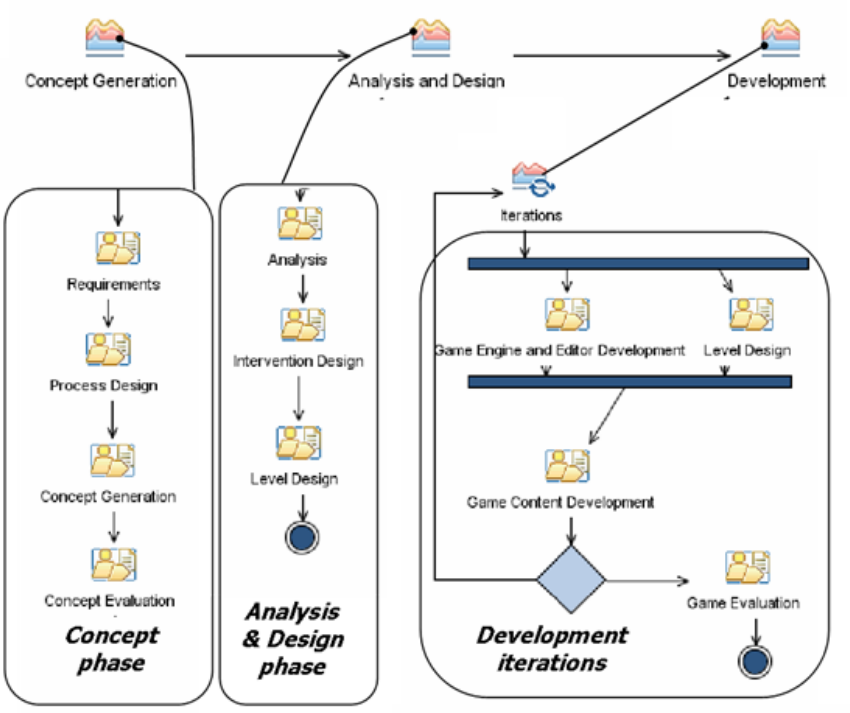

Figure 8.4: Diagram of the phases and activities in the Peatland Adventure project.

The second part of the project started with a game concept and learning goal definition on top of which the project was founded. The project group worked autonomously to refine the game concept and design to confirm to the requirements set in the first part of the project. The project work was iterative and empirical and the project group was in control of deciding what to work on in each iteration. This approach lead to the experience that development iterations were also a learning experience: *"The first level design and learning goal and content design iteration served as a learning experience for us designers."* [5]

The process control was close to Scrum but did not adhere to the specific activities of the Scrum project model. Also in contrast with other projects in this study, HCD

[5]Excerpt from the historical mapping session with the project group, transcribed and translated by author.

methods were not used during the development work of this project.

The self-directed and iterative manner of working was noted as effective and rewarding by the project team members. The team members also commented that they were learning very efficiently about different aspects GBLE development while working together as a tightly collaborating and self-directed project team. In the project report other stakeholders also commented on the high quality of the finished game and the ability of the project team to produce the game in a relatively short time.

8.2 Examination of Key Activities

Processes for GBLE development were constructed in chapter 7. This part of the result description of the study discusses the key activities in the GBLE development process and the impact of those to the process control of the project. The three key activity groups identified in chapter 7 discussed here are 1) GBLE concept generation combined with learning goal setting, 2) HCD activities and 3) game design activities.

8.2.1 Concept Generation and Learning Goal Setting

In chapter 7 it was noted that the learning goal setting and game concept generation should go hand in hand in GBLE development. It was also been discovered that GBLE projects benefit from iterative approach to game concept generation. The outcome of these activities has also been experienced to be unpredictable as evidenced for example by the unsuccessful termination of the first requirements phase of the Peatland Game project.

The notion that these activities are closely knit indicates that the defined process control model, where goals are first set and then a process is planned to meet the goals would be unfeasible for these activities. This is because the both activities are in a way a prerequisite for each other; learning goals cannot be set first and then a game concept created to meet these goals, as the choice of feasible game concepts restricts the attainable learning goals and vice versa.

This is also a matter of planning the resources for these activities as the uncertainty involved would seem to limit the possibility of estimating the resources needed to carry out these activities in a way that enables the desired quality for the work products.

These findings indicate that empirical process control would be a better match for these activities. This would allow the project team to take control of the work being

done. The sprint structure of work would allow the activities to be iterated until the work products have been validated to have sufficient quality or the whole project could be terminated after any sprint if the activity would be deemed as unfeasible.

8.2.2 HCD Activities

In the Gameli V.1 project there was a perceived difficulty into fitting HCD activities within a project that used defined process control model. This was perceived to be more because the project planning did not take into account these HCD methods than the inherent incapability to control these processes using defined process control.

In Gameli V.2 and the SR Game projects the incorporation of HCD activities to defined process control was considered more successful, although in Gameli V.2 project the team perceived potential risks related to the combination.

These risks, namely the change in scope and requirements resulting from HCD workshops can be considered as a real one in any HCD project. This comes from the inherent iterative manner of HCD process where user designers are used throughout the project but only in specific points to provide information on requirements and to evaluate work done up to that point.

This kind of work structure would be supported inherently by empirical process control which in itself has the same kind of feedback pattern. For example Scrum every sprint could have one major HCD process iteration where sprint planning and review would be coupled with HCD requirements elicitation and HCD evaluation, respectively.

8.2.3 Game Design Activities

Game design is described as a second-level design activity where the designer designs the rules of the system in order to make the interaction of the player and the game to occur in a specific way [Zimmerman 2003]. The result of this is that game design can be evaluated only with an end product a player can interact with — a prototype of the game.

The results of this study indicate that this is true to the design of GBLEs as well. Particularly the Peatland Adventure project was reported to benefit from iterative game design, namely producing concurrent versions of the game along the design process. The study of social responsibility game project also reported similar findings.

The iterative game design process demands empirical process control as the iterative game design activity cannot be estimated accurately beforehand. On the other hand,

if the project is using traditional document-based game design process the quality of game design cannot be accurately evaluated before the first version of the game is ready.

Although experienced game designers could possible lessen this risk somewhat, it must be concluded that any GBLE development project with a considerable amount of uncertainty related to the game design would benefit from empirical process control related to the game design activities.

The uncertainty can be a result of not having enough results of a certain game type supporting the type of learning intended or not having enough information about the quality of gameplay experience being designed. Projects working with game genres with known advantages of supporting learning in the desired field and a tried and tested game design could therefore be designed with the traditional document-driven approach.

8.3 Process Control Selection in GBLE Projects

This chapter summarizes the findings done in the previous two result chapters regarding experiences on the process control model selection in GBLE projects.

In the case projects risks and problems were encountered with the defined process control model:

- Results of GBLE concept creation can force changes in the project planning,
- risk of feature creep caused by HCD workshop results,
- planning with incomplete goals and requirements leads to problems and
- GBLE tasks that are not explicitly defined are a risk to schedule and resource planning.

These problems were highlighted in projects with much uncertainty in terms of goals and game design solutions.

Many of the key activities of GBLE development were found to be inexplicitly defined or subject to much uncertainty relating to the quality of their end products and the resources needed to complete them.

These finds indicate that, particularly when working with GBLE projects with uncertain elements, an empirical process control model would be preferable to a defined

one. In the following chapter we discuss how to adapt the Scrum process model, used as an example of empirical process control to different facets of GBLE development.

8.4 Adapting the Scrum Process to GBLE

Guidelines for adapting the Scrum process, used in this study as an example for empirical process control, to aspects of GBLE development are presented in this chapter. The guidelines are drawn from the literary review of the Scrum process and the experiences in the game projects on carying out the key activities of GBLE development.

8.4.1 Adapting Scrum to GBLE Concept Creation

This compound activity consists of two sub-activities each producing their own work product. The sub-activities are learning goal definition and GBLE's game concept creation and the corresponding work products are learning goal definition (or requirements) document and game concept document.

As the in Scrum every sprint should produce a shippable delivery of the product and design documentation cannot be considered to be same as the product, adapting this activity to Scrum is not a straight-forward process.

One possible solution would be to add a third sub-activity, producing a prototype of the game concept, to this activity. The prototype would act as the shippable version of the product in Sprints and could be evaluated in Sprint reviews. This would also work towards more valuable evaluation of the other work products of the activity as prototype would offer more evidence to the validity of the learning goals and the ability of the game concept to fulfill them.

This would indicate that in order to keep the process agile the prototype building activity would have to be light, meaning a prototype of low fidelity and therefore also of technology.

8.4.2 Adapting Scrum to HCD

In human-centered design process the three main tasks of the user designer participants are setting the requirements, generating design solutions and evaluating designs and end products [Nousiainen 2008].

In a project using Scrum these tasks would fit the overall process in the following way: generating requirements would mean submitting items in the project backlog as

well as informing the product owner of priority changes of existing items in the project backlog. The design solutions would be created in collaboration with the development team during sprints and the evaluation of end products would be done to inform the sprint review meeting or as part of the sprint review itself.

This indicates that the product owner role should be assigned to the person also responsible for the HCD process and the workshops. Workshops would be arranged at the end of each sprint to review work done and to inform the product owner of backlog changes needed. Additional workshops would be called during sprints to support the design of solutions as needed.

8.4.3 Adapting Scrum to Game Design

Adapting Scrum to game design would mean changing the game design process into a more empirical one and one focusing more on making versions or prototypes of the game along the design process.

This kind of process for game design is described by Eric Zimmerman in the article Iterative Game Design [Zimmerman 2003]. Iterative game design is based on a cyclic process in which the production of gameplay prototypes, analysis and refining the prototype take turns [Zimmerman 2003].

Iterative game design process could be used also in GBLE projects using Scrum. The analysis of prototypes would effectively inform/or take place in the Sprint review meetings and the Sprints itself would be centered around producing and refining the game prototypes.

8.5 Conclusions

The aim of this part of the study was to explore the alternatives to process control in GBLE development and provide guidelines of planning and carrying out process control in GBLE projects. The research questions were: 1) What kind of process control model is most advantageous and feasible to the development of GBLE (both in general and especially in those projects concentrating on game design and human-centered design) and 2) what kind of adaptation of that process control model would lead to a high-quality development process in this domain. The research was conducted according to the principles of action research and developmental research. Four case projects were analyzed and their experiences were assessed.

In examining the experiences of the case projects regarding process control models and the key activities of GBLE development there were found multiple factors supporting the advantages of using empirical process control model in GBLE development. Problems were noted with activity planning and scheduling in the case projects using a defined process control model. It was found that there were a large number of important activities that were not explicitly defined and thus could not be accurately estimated or planned. The key activities were also found to contain a lot of uncertainty related to their completion. These findings support the notion that utilizing an empirical process control model such as Scrum would be preferable in the development of GBLE.

Scrum was introduced for software development and it must be adapted for game development [Lahti 2008]. The study presents guidelines for adapting Scrum to the learning-goal focused game concept creation in GBLE development, human-centered design and game design activities. These include guidelines of using activities defined in earlier research for development, assessment and control functions within the Scrum framework.

The main results of this study are the evaluation of the two main process control models for the project management of game-based learning environment development as well as guidelines for adapting Scrum for the development of game-based learning environments. These results are based on experiences in the four case projects and a closer examination of key activities in GBLE development identified by earlier research. These esults provide a good foundation for future development projects in the field. It will also serve further studies into the process of developing game-based learning environments.

9 A General Metamodel for Development of Game-Based Learning Environments

The earlier result chapters explored the GBLE development from three different viewpoints: developer roles and expertise (chapter 6), tasks, activities and work products (chapter 7) as well as process control (chapter 8). The aim of this chapter is to build a foundation of guidelines to GBLE development in a form of a process model or a metamodel. The research question this part of the study addresses is: What kind of general prescriptive process model or metamodel could be constructed for GBLE development?

The research question is answered by creating SPEM 2.0 (see chapter 2.2.2) compatible process models based on the results of the study described in the earlier result chapter. The resulting prescriptive process model consists of role, task and work product descriptions as well as activity and process descriptions. The activity descriptions use the role, task and work product descriptions and the process descriptions use all the other descriptions as needed. The activity and process descriptions are accompanied with activity, activity detail and work product dependency diagrams where applicable. All the process elements are further described by guidance elements.

The results of this phase of study are discussed in the following chapters: description of the overall process, role descriptions, activity descriptions (including task descriptions), work product descriptions and finally conclusions. This phase of research has also produced the results in another form: The metamodel has been modeled with Eclipse Process Framework Composer (see chapter 2.2.2).

9.1 Process

The description of the process metamodel begins with the overview of the development metamodel. Individual activities and tasks as well as roles and work products involved are described in the following chapters. This chapter focuses on describing the high-level composition of the process and its performers. Are there phases the development process can be divided into, what activities the phases composed of and what is the project team composition in each of the phases.

The approach chosen here is to base the process metamodel on the selection of the process control model. This approach is chosen because the process control model can affect the makeup of the process in question in all levels of process modeling. Process control model defines the process of project management: the way project planning, project monitoring and reacting to problems and changes are carried out. Process control models in GBLE development is examined in chapter 8.

In chapter 8 it is concluded that an experimental process control model is preferable to defined process control model in GBLE development based on the findings in the case projects. Based on that find the metamodel is based on the Scrum process control model, an experimental process control model widely used in the software engineering discipline.

9.1.1 Scrum Process Control Model

Scrum process control model is discussed in detail in chapter 5.3.2. The main features are repeated here to make the presentation of the metamodel complete. The presentation in this chapter is based on chapter 5.3.2. In Scrum, the development is divided to short work cycles called sprints (Fig. 9.1). Each sprint produces a visible, usable and deliverable product. A sprint is timeboxed development, meaning that the end date for a sprint does not change. Each project can specify the exact length of a sprint according to its needs. Generally sprint lasts from 2 to 4 weeks.

Scrum has two special roles defined: the Scrum Master and the Product Owner. It also divides all the people involved in a project into two general roles of developers and stakeholders. The two special roles are also in developers. Essentially, everyone who is committed to completing the project is a developer and the rest are stakeholders.

The Scrum Master is a managerial role; the goal of the scrum master is to make sure the team is as productive as possible. The product owner is the project's key stakeholder and represents users, customers and others in the process. These roles are discussed in more detail in the context of GBLE development metamodel in chapter 9.2.

Scrum has a special mechanism for requirements process, change management, planning and monitoring called the product backlog. The product backlog is a prioritized features list containing every desired feature or change to the product. All additions and changes to the backlog must go through the product owner. This helps the scrum team deal with changes in the project.

Before every Sprint the development team sets its own goal for the Sprint and

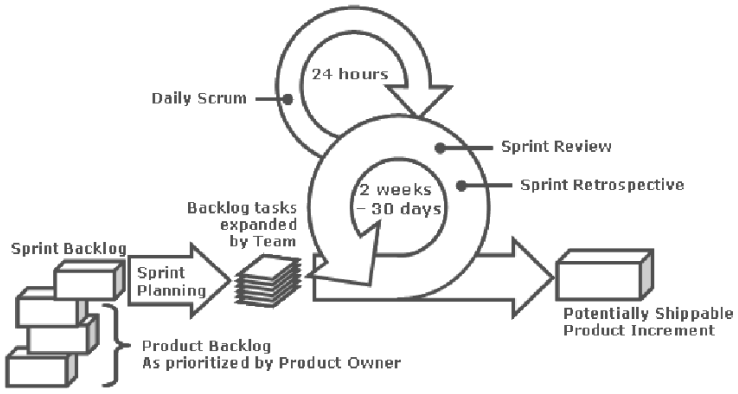

Figure 9.1: Diagram of the Scrum process.

commits to completing it. At the start of each sprint, a sprint planning meeting is held during which the product owner prioritizes the product backlog, and the scrum team sets their Sprint goal, including selecting the work they can complete during the coming sprint. That work is then moved from the product backlog to the sprint backlog, which is the list of tasks needed to complete the product backlog items the team has committed to complete in the sprint. The team can reduce delivered functionality to achieve the sprint goal in schedule.

Each day during the sprint, a brief meeting called the daily scrum is conducted. All team members are required to attend the daily scrum. In daily scrum every team member presents work done after last meeting, what she will do before next meeting and possible obstacles that hinder her work.

At the end of each sprint, the team demonstrates the completed functionality at a sprint review meeting, during which the team shows what they accomplished during the sprint. In the sprint review all the sprint backlog items that are marked as completed by the project team during the sprint are assessed and if accepted by the product owenr are confirmed to be completed. If any items remain in the sprint baklog at the end of the sprint, they are returned to the product backlog. The product owner updates the product backlog for item descriptions and priority.

9.1.2 Scrum in GBLE Metamodel

The roles, backlogs and the general sprint cycle organization of Scrum can be translated into GBLE development as well. However, there are a number of considerations about arranging the development process with Scrum. The sprints are relatively short cycles

for producing a deliverable end product such as a learning game from scratch. The start of first sprint assumes that some kind of starting product backlog exists by which to set the frist sprint goal and from which to select items into the sprint backlog. And finally, how to test the completion of backlog items in a GBLE development project.

The first consideration of producing a deliverable product in a short time applies to first sprints of a project. A similar problem exists in software development, where it is advised that the product of the first sprint should have minimum functionality but of a key feature [Schwaber & Beedle 2001]. This approach can be taken in GBLE development. It could be feasible, for example, to focus the first sprint around GBLE concept creation activity described in chapter 9.3.1. The end product of that sprint would be a prototype demonstrating one or more of the main features of the GBLE concept.

Similar approach was taken in the Gameli V2 project. The prototype was a paper prototype in that case. The experiences from the project suggest that in that case building and testing the prototype provided the project team with more knowledge about the desired features of the project and helped it on its way.

The aforementioned approach is also one solution for the second consideration of minimum starting product backlog. The product backlog for the first sprint could consist of just enough items for the development team to go about the concept creation effort. This starting backlog could be constructed in the same way as in a new software engineering project is suggested to be started in Scrum literature: Project team and key stakeholders collaborate for a few days to populate the backlog with key features and technical requirements [Schwaber & Beedle 2001]. In the case of GBLE development, requirements related to potential use context, target group, learning style and subject matter could also be included.

As for the final consideration, a number of parts in this study focus on potential methods to test the developed GBLE for fulfilling a requirement or a goal set to it. As each sprint produces a deliverable product, all sprint backlog items can be validated as complete by testing the product itself. HCD prototype tests described in chapter 9.3.4 are one way. The prototype analysis task of iterative game design activity is another one 9.3.5.

The nature of Scrum process control model that presents itself with short work cycles and focus on producing deliverable versions of the product make the efforts of modeling a stardard sprint activity model or standard Scrum process model redundant. The only tasks common to any sprint are sprint planning, daily scrum and sprint review

meetings. The other tasks in the sprint depend on the goals of the sprint and the sprint backlog.

Also the fact that every sprint produces a deliverable product means that each sprint contains some tasks from each phase of the traditional software project phases: requirements specification, design, development, testing and delivery. The same is true forthe other involved disciplines also: from the game development point of view every sprint has tasks from pre-production and production phases.

Rather than sketching out potential processes for different kinds of sprints the description of the metamodel will consist of different activity capability patterns for different activities that the project is likely to undertake on its course (chapter 9.3). These support the planning work of the project team in the actual sprints of the project.

9.2 Role Descriptions

First part of the metamodel description is the role description. Roles in a GBLE team are explored in detail in chapter 3.4.1. This chapter summarizes those findings and builds upon them to build a set of roles to base the GBLE development process model upon. The roles are described in terms of skills and expertise needed, responsibility and involvement in the process. As for responsibility this presentation uses the terms described in chapter 6.2.1: Project manager, developer, producer, consultant, asset producer and user.

9.2.1 Learning Expert

Learning expert is the member of the development team whose responsibility is to represent the educational andlearning expertise within the team. This is a developer role as well as a producer role.

As summarized in the literature review (chapter 3.4.1), the educational and learning expertise needed in the design of digital game-based learning environments can be presumed to include sound knowledge on learning theories and pedagogical design of quality learning environment and also content area expertise in the case of content-specific games.

On top of that, as noted in chapter 3.4.1, it is preferable that the learning expert is familiar with aspects of game design. This helps his or her communication with the game designers. When HCD methods are used, skills and experience in working with

user designers to collaborate in design is also beneficial.

The results of the study of teams and roles (see chapter 3.4.1) indicate that it is beneficial for the development project that learning expert is involved in the development process from beginning to the end. The learning expert is important in developing the game concept and seeing it come true during the course of the project.

9.2.2 Lead Designer

Lead designer is the development teams leading game designer responsible for the vision of the game and its game design aspects. Like learning expert, this is a developer role as well as a producer role.

The skill requirements for a lead designer are the game design skills described in chapter 4.1. The lead designer should also have good communication and presentation skills as well as leadership skills if he is to work with a team of game designer and asset producers.

In a GBLE development project the lead designer also benefits from knowledge of educational and learning expertise. This helps his or her communication with the learning experts. When HCD methods are used, skills and experience in working with user designers to collaborate in design is also beneficial.

Like the learning expert, lead designer is present in the development from start to finish. With the learning expert, he or she is the creator and champion of the game concept. Maintaining solid vision and communication regarding the am of the project to the rest of the development team is their main responsibility.

9.2.3 HCD Producer

HCD producer is responsible for planning, arranging and supporting the use of human-centred design methods in the GBLE development process. As stated in the title this is a producer role.

The HCD producer role requires knowledge of the theory of human-centred design and collaborative design methods as well as skills to plan and organize the HCD design workshops. The HCD producer also needs project planning skills and familiarity with the GBLE development process in order to be able to integrate HCD methods seamlessly into the development process.

In GBLE projects that use HCD methods the HCD producer should be present thorughout the project. He or she is needed at the start of the project to plan for HCD

method integration to the rest of the development process and later on to facilitate the HCD activities and tasks.

9.2.4 User Designer

User designer is the designer role adopted by the future users of the GBLE to be developed. User designers take part in HCD design activities and tasks and provide user perspective to the design process.

The user designers only need knowledge of their own needs and the context they are going to integrate the GBLE into. In fact that is the main value of their involvement.

User designers should be present in GBLE development projects utilizing HCD methods during the stages of the process where HCD methods are used for requirements engineering, design and assessment.

9.2.5 Scrum Master

The Scrum Master is a managerial role; the goal of the scrum master is to make sure the team is as productive as possible. The Scrum Master does this by helping the team use the Scrum process, by removing impediments to progress and by protecting the team from outside pressure during sprints. [Schwaber & Beedle 2001]

9.2.6 Product Owner

The product owner is the project's key stakeholder and represents users, customers and others in the process. The product owner is a developer and managerial role. The produt owner is the person who is responsible for adding items to the product backlog and re-prioritizing the items in the backlog. All additions and changes to the backlog must go through the product owner. [Schwaber & Beedle 2001]

In GBLE projects the product owner may benefit from knowledge of game design, learning, the subject matter of the game or the context it is intended to be used with.

9.2.7 Supporting Roles

There are a number of supporting roles that can be present in a GBLE development process, but which are not necessarily needed. The need for these roles depends on the content of the process of the project in question. The supporting roles are covered in this chapter.

If the project has more than one people responsible for designing the gameplay aspects of the GBLE, it is recommended that one of these people assumes the role of lead designer and the others are assigned the **game designer** role. The game designers work on the game design activities in the project and are lead by the lead designer. Additional game designers can enter the project after beginning when needed.

For game design and development a number of **asset producer** roles can be beneficial. These include game artists, audio producers and designers and story writers. These roles are discussed in detail in chapter 4.1. Typically these roles enter the project after the basic game concept has been designed. They are needed in producing the different kinds of assets used in the game.

Subject matter experts are consultants who have special knowledge and expertise on the subject matter of the GBLE being developed. They can be utilized to help in concept creation, design and evaluation of the GBLE.

If the GBLE being developed needs new software to be developed or changes to the existing software, a number of software engineering roles is needed as well. These are not covered in detail, as the software engineering aspect of GBLE development activities were seen as the least problematic activities of GBLE development. The metamodel therefore uses roles defined in Eclipse Process Framework Practice Library: software architect, software developer, stakeholder and tester [Eclipse Process Framework Project 2008]. These roles are describe in detail in appendix chapter A.

The **software architect** is responsible for defining the software architecture, including key decisions dealing with the structure of the system. Software architect should be present during the whole GBLE project if major software engineering activities are planned. [Eclipse Process Framework Project 2008]

The **software developer** is responsible for developing a part of the system, including designing it to fit into the architecture, possibly prototyping the user interface, and then implementing, unit-testing, and integrating the components that are part of the solution. [Eclipse Process Framework Project 2008]

The **project manager** leads the planning of the project, coordinates interactions with the Stakeholders, and keeps the project team focused on meeting the project objectives. [Eclipse Process Framework Project 2008]

The **stakeholder** role represents interest groups whose needs must be satisfied by the project. It is a role that may be played by anyone who is (or potentially will be) materially affected by the outcome of the project.´[Eclipse Process Framework Project 2008]

The **tester** is responsible for the core activities of the test effort. Those activities in-

clude identifying, defining, implementing, and conducting the necessary tests, as well as logging the outcomes of the testing and analyzing the results. [Eclipse Process Framework Project 200

9.3 Activity Capability Patterns

9.3.1 GBLE Concept Creation

The importance of the GBLE concept creation activity was highlighted both in chapter 7.3.3 and in chapter 8.4.1. Therefore building the prescriptive capability pattern for this activity is a high priority task in this part of the study.

The findings in chapter 7.3.3 indicate that the learning goal definition should precede or be a part of the GBLE game concept creation. In chapter 8.4.1 it is suggested that in order to create a validatable GBLE concept, learning goal definition should be done as part of game concept creation activity. Furthermore it is noted that in order to integrate experimental process control with GBLE concept development, a prototype should be constructed as a part of GBLE concept creation activity.

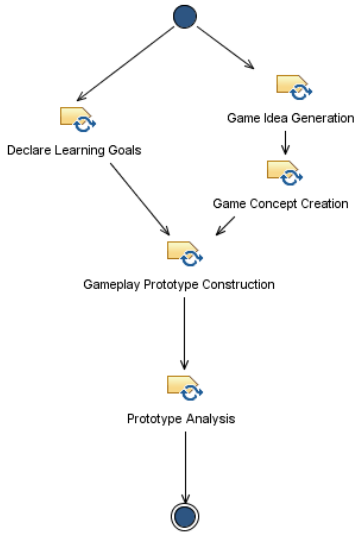

Figure 9.2: The activity diagram of GBLE concept creation capability pattern.

The requirements that these results give to the GBLE concept creation activity can be satisfied with the following capability pattern (Fig. 9.2): The activity begins with simultaneous learning goal definition task. At the same time game idea and after that the game concept are created. When these tasks are completed a game concept

prototype is constructed based on the learning goal definition and the game concept document. After that the prototype is analysed by playtesting. The prototype analysis is used when reviewing the result of the activity.

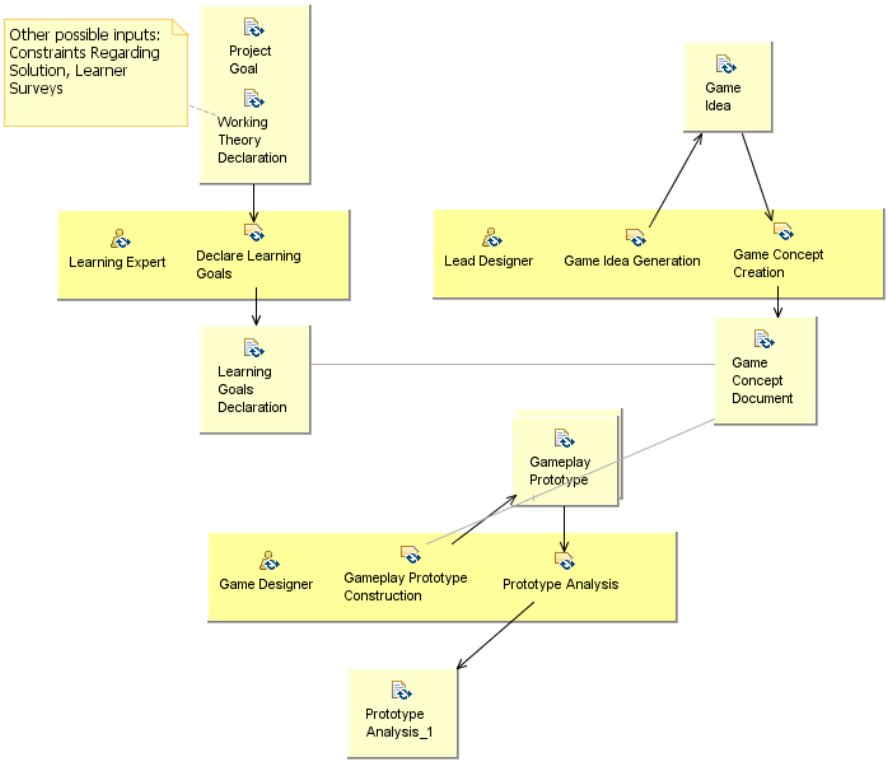

Figure 9.3: The activity detail diagram of GBLE concept creation capability pattern.

The capability pattern of this activity is further illustrated by the activity detail diagram (Fig. 9.3). The learning expert is responsible for the learning goal definition task. The lead designer is responsible for game idea generation, game concept creation and prototype construction. The both aforementioned roles perform the prototype analysis. All of these tasks also have optional additional performers: HCD designer and user designer in case HCD methods are used in the concept creation and stakeholders in the prototype analysis task.

This capability pattern demonstrates the separate tasks in GBLE concept development adequately. It does not, however, demonstrate the idea that learning goal definition, game idea generaton and game concept creation are interconnected tasks. Therefore an alternative capability pattern that demonstrates this has been developed (Fig. 9.4).

In this capability pattern learning goal definition, game idea generation and game

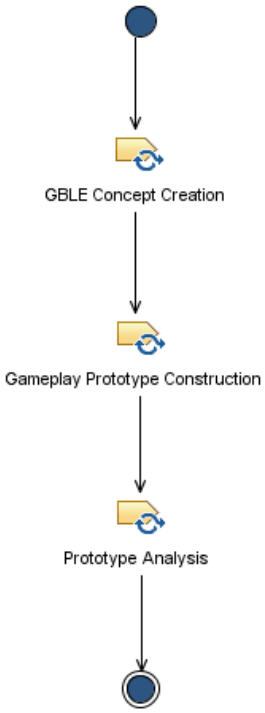

Figure 9.4: The activity diagram of the alternative GBLE concept creation capability pattern.

concept creation are combined into a single task, GBLE concept creation. This produces a single work product, GBLE concept document. The GBLE concept document includes the game idea description, game concept document and learning goal definition. The primary performers of GBLE concept creation task are the learning expert and the lead designer (Fig. 9.5).

Otherwise this the make-up of this pattern is identical to the first capability pattern. These two patterns are alternatives to each other and the person responsible for planning the project process should choose between them based on whether emphasizing the distinct tasks or the importance of collaboration is more important in the concept creation of the project in question.

9.3.2 HCD Requirements Specification

Integrating HCD methods to the rest of the design process was reported as problematics in findings of the study described in chapter 7. Consequently the adaptation of HCD

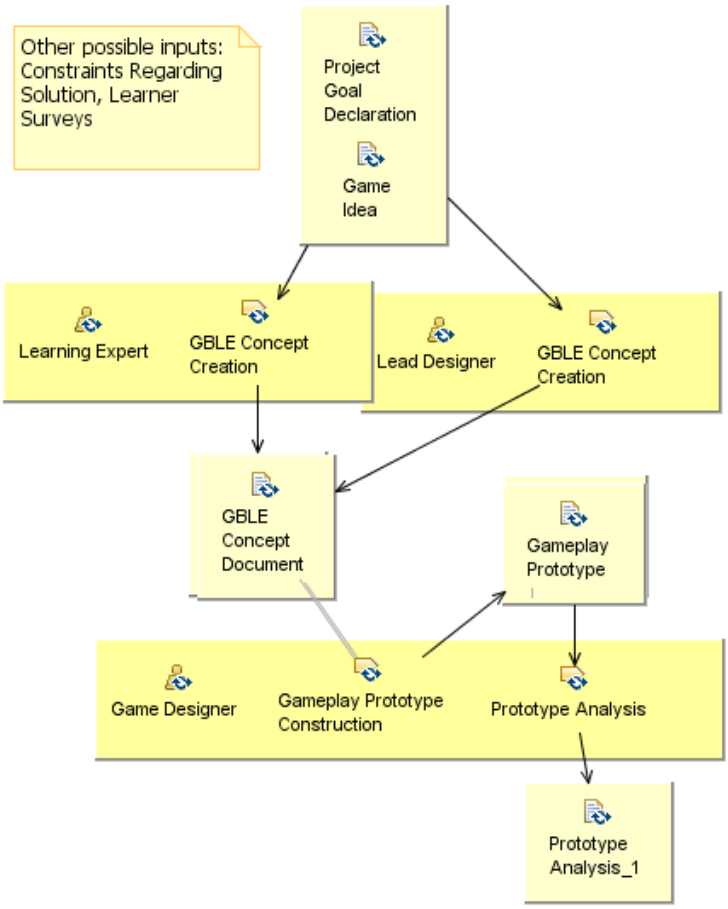

Figure 9.5: The activity detail diagram of alternative GBLE concept creation capability pattern.

methods to experimental process control were discussed in chapter 8.4.2.

In this part of the study, the reported problems and adaptation needs are discussed with three capability patterns: HCD Requirements Specification (this chapter), HCD Design (chapter 9.3.3) and HCD Prototype Evaluation (chapter 9.3.4).

The HCD Requirements Specification capability pattern describes the general activity model for using HCD methods in GBLE requirements engineering. It is designed to be used to plan a requirements specification process that allows the project to use HCD methods and plan for them in a way that minimizes the reported risks in including HCD methods in the process. The assumption in these capability patterns is that the process control model used in experimental process control model of Scrum.

The activity consists of three tasks: HCD requirements generation workshop plan-

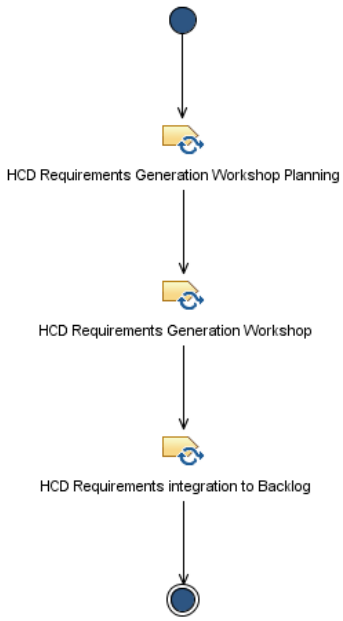

Figure 9.6: The activity diagram of HCD Requirements Specification capability pattern.

ning, HCD requirements generation workshop and HCD requirements integration to backlog (Fig. 9.6). The first task is performed by the HCD producer (Fig. 9.7). The user designers, along with the designers in the development team, perform the workshop. These designers can include the learning expert, lead designer, software developers and other developers as needed as per the project in question.

The HCD requirements generation workshop planning tasks does not have any required inputs (Fig. 9.7). It can, however, be informed by a number of work products if they are available: the GBLE concept, the product backlog (see 5.3.2), the various results of the tasks of learning sciences analysis phase including but not restricted to learner study, context and resources study and problem assessment report.

The HCD workshop plan includes the goal declaration of the requirements workshop and the plan to perform the workshop including a schedule and make-up of the workshop.

The workshop produces a set of new requirements and changes to existing requirements. These changes are integrated to the product backlog by the product owner, who is responsible for keeping the product backlog up to date. As the project uses product backlog to plan sprints and prioritize their work the results of the workshop

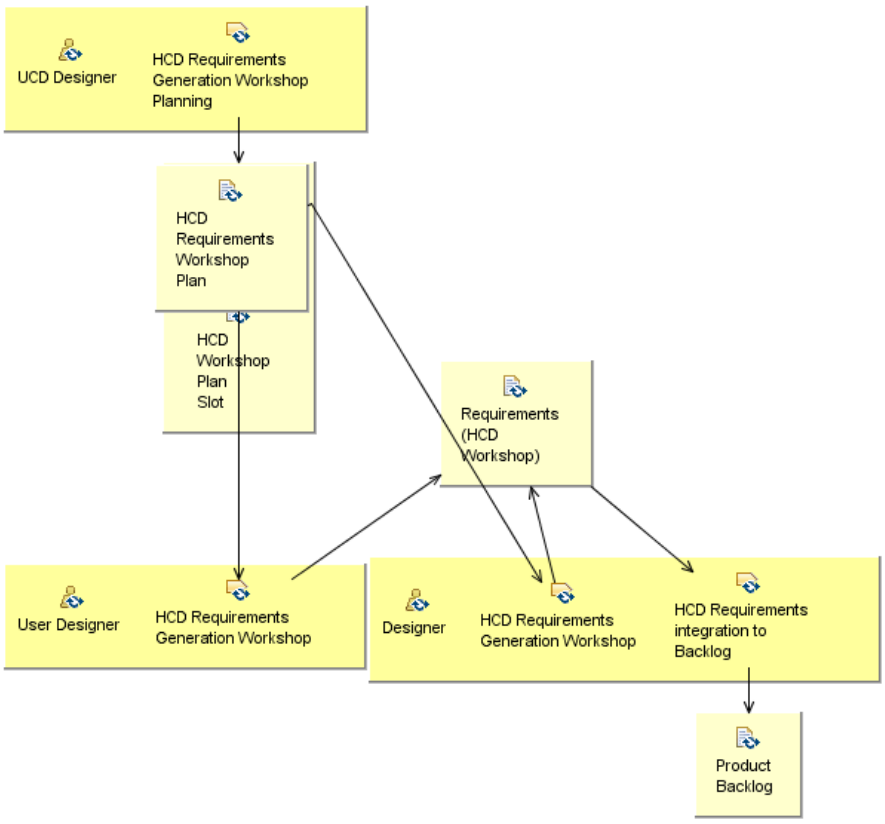

Figure 9.7: The activity detail diagram of HCD Requirements Specification capability pattern.

carried on into the project work during the next Sprint planning (see 5.3.2).

Similar capability pattern for this activity could be devised for projects not using Scrum as the process control model. In that case the results of the integration activity would be an updated requirements specification document. In that kind of process setup the risks of HCD requirements generation highlighted in chapter 7, namely the pressure to change and enlarge project scope become more probable. The actualization of these risks would result in change of project plans in the following phases of the project. This is the reason why it is considered advantageous in this process model to not include that kind of capability models and to recommend the use of Scrum process control model if HCD methods are chosen as part of the requirements specification process.

9.3.3 HCD Design

The human-centred design activity capability pattern models the general pattern for arranging the use of HCD design methods as part of the design work in the project.

The overall course of this pattern is very similar to the HCD requirements generation capability pattern. The first task is planning, which is followed by workshop (Fig. 9.8). The final task is integrating the workshop results to the overall design of the GBLE.

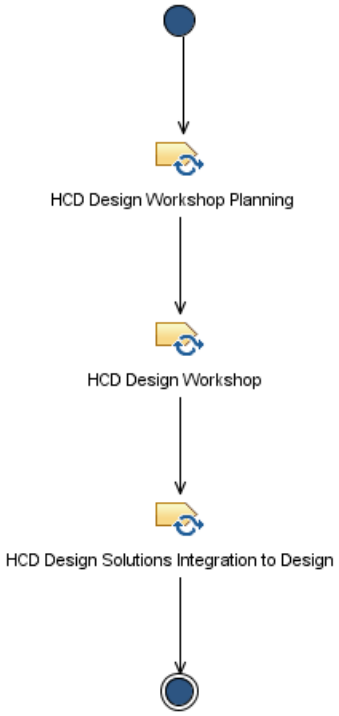

Figure 9.8: The activity diagram of HCD Design capability pattern.

The simplicity of this model is a requirement for its use in various phases in the GBLE development process. The aim of the activity that is formed informed by this pattern can be a solution for a specific and small-scale design problem or a more large scale problem such as the whole user interface of the environment.

The planning task is identical to the planning task in the HCD requirements generation capability pattern. The workshop task involves solving the design problem at hand by the user designers and designers in collaboration. The integration task involves integrating the results of the workshop to existing design, solving potential problems that the new design may produce in other parts of the system at the same time. The

work products and performers of this activity pattern are similar than in the previous pattern (Fig. 9.9).

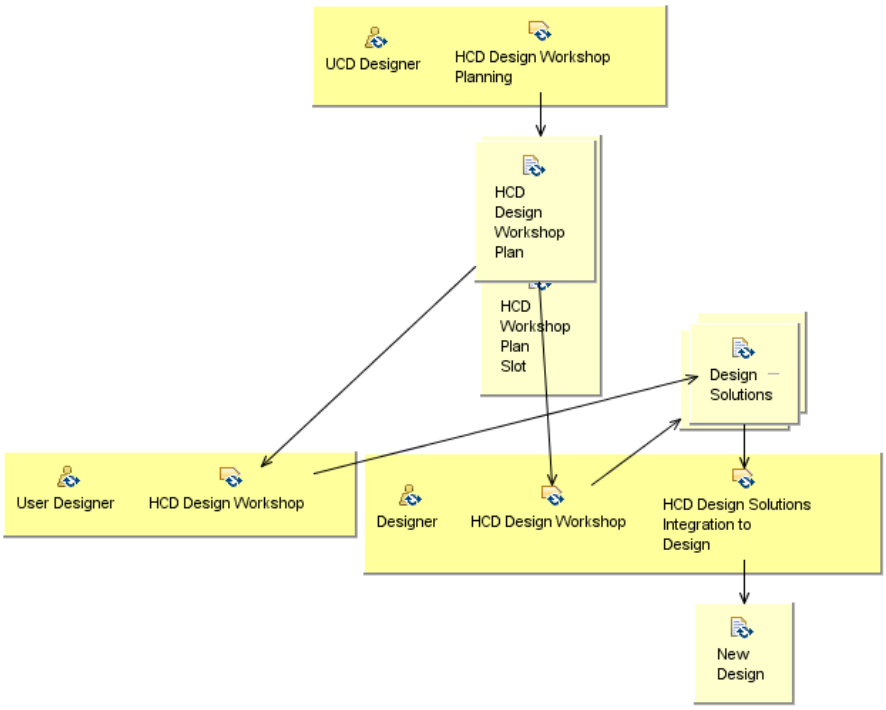

Figure 9.9: The activity detail diagram of HCD Design capability pattern.

Although the planning task has no mandatory input work products, usually the existing requirements, product backlog and the existing design documentation along with the most recent version of the shippable product (usually meaning a prototype at this stage) are the optional inputs if available. The most recent version of the shippable prototype, requirements and the existing design documentation can also be used as optional inputs for the HCD design workshop.

This activity does not dictate what is done with the new design that is the main end product of the activity pattern. However, if the project is using Scrum as a process control model as is advised, the design is most likely used to build the next version of the prototype of the GBLE. In that way it can be tested at the end of the Sprint.

9.3.4 HCD Prototype Evaluation

This third HCD capability pattern deals with evaluating a prototype together with the user designers. Once more, the capability pattern assumes that the process control

model is Scrum. The capability pattern consists of three tasks: prototype testing workshop planning, prototype testing workshop and test assessment (Fig. 9.10).

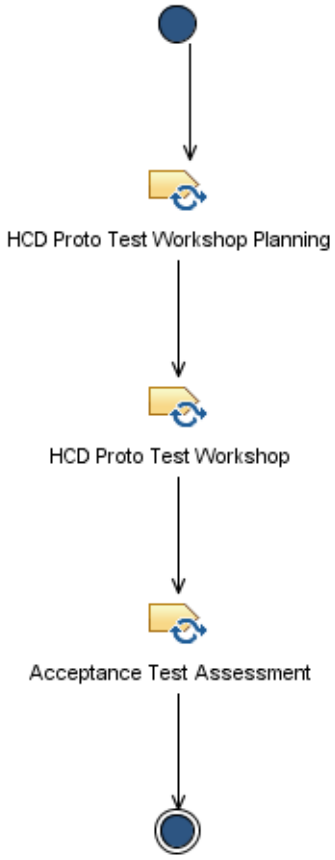

Figure 9.10: The activity diagram of HCD Prototype Evaluation capability pattern.

The workshop planning task is similar to the two other HCD capability patterns. The planning task takes the prototype as an input (Fig. 9.11). It can also take additional optional inputs such as the sprint goal of the current Scrum sprint (see chapter 5.3.2). In the workshop task the user designers test and evaluate the prototype. After that the designers and the product owner assess the test results. They determine based on the test results whether the Sprint goal of the current sprint was met (work product acceptance test results) and what changes to the product are needed to satisfy the user designers' evaluations (work product updated product backlog).

It is to be noted that the question of whether the sprint goal was reached and the sprint backlog items were completed in a satisfactory way do not affect the duration of the sprint directly. The sprint normally ends when the deadline date is reached and

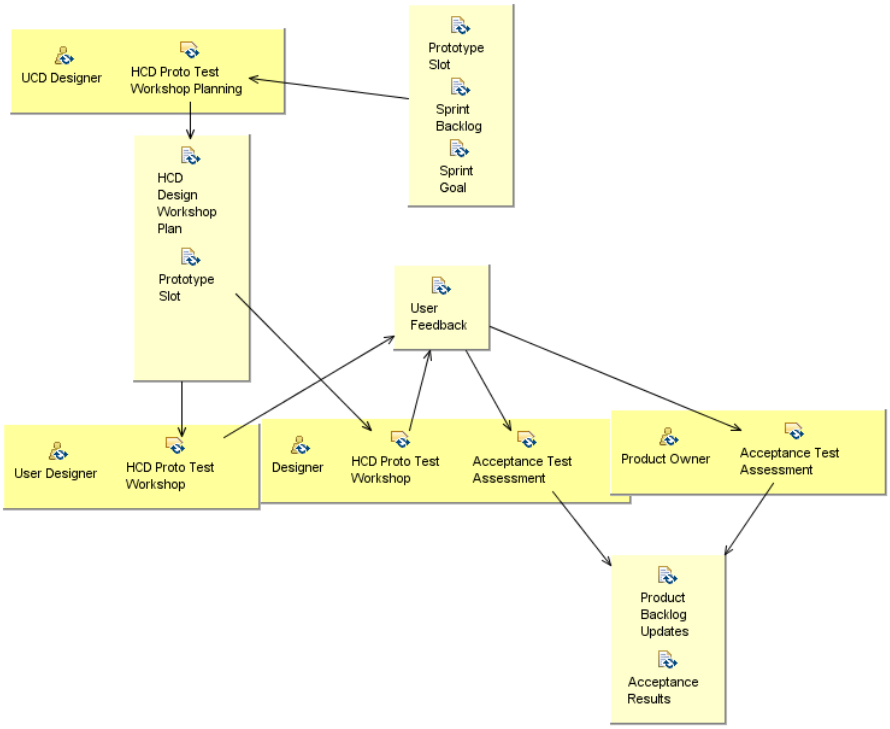

Figure 9.11: The activity detail diagram of HCD Prototype Evaluation capability pattern.

a sprint review is held. It is during this review that the completion of the sprint goals are assessed. If the sprint goals were not met, it must be assessed why that was. This assessment may lead to changes in the procedures of the project. The stakeholders of the project can also decide to end the project. This can be done whether or not the sprint goal was met during the last sprint.

If the project is not cancelled, the effect of the accepatnce test results are projected into the project by changes in the product backlog. The items that were assessed as not completed are returned to the backlog, with more description and changed priority if needed. If the sprint goal is deemed important or essential to be stated as a product backlog item, it can be added to the backlog as well. The next sprint goal and sprint backlog are selected by the basis of the new item prioritization in the product backlog.

9.3.5 Iterative Game Design

As discussed in chapter 7.3.2 the adaption of iterative game design model was seen beneficial in two of the four case projects. There were also opinions raised that more

iterative game design activities could have increased the quality of games in the two projects that did not use them at all or used them only at the later stages of the project. Therefore including the iterative game design model as a capability pattern in this process model is justifiable.

This presentation of the iterative game design activity is in essence the same as in chapter 4.4.4. The activity description is included here in the form of a capability pattern for completeness.

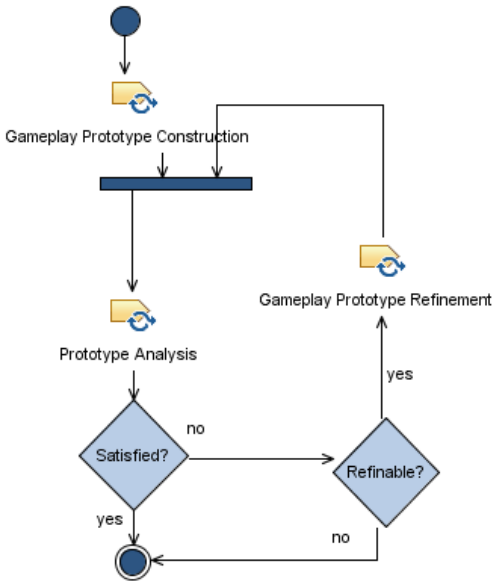

Figure 9.12: The activity diagram of Iterative Game Design capability pattern.

The activity in this capability pattern is cyclic in two levels. In the higher level cycles consist of goal-oriented design cycles. In the beginning of each iteration goals are defined and in the end the latest version of the prototype can be assessed as fulfilling or not fulfilling those design goals. The activity diagram (Fig. 9.12) describes the one iteration of this higher level cycle. The prototype is first constructed. The construction task is informed by the goals set for the current iteration.

The prototype is then analyzed in regards to goals set. If the goals are satisfied, the iteration ends. If the goals are not met, a decision is made whether the prototype can be further refined to meet the design goals. If not, the iteration ends. If it is determined that further refinement can be made, the prototype is refined and further analyzed.

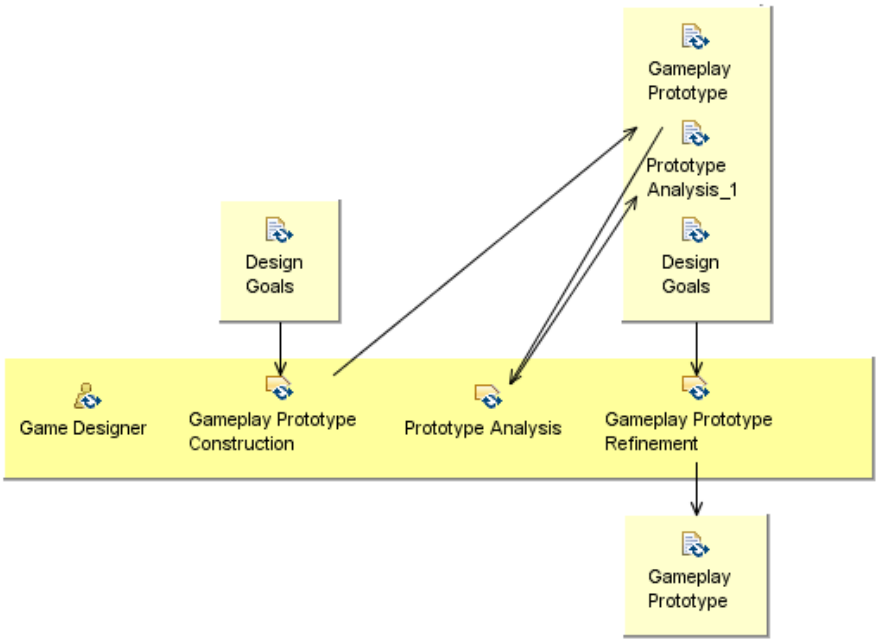

Figure 9.13: The activity detail diagram of Iterative Game Design capability pattern.

The inner cycle in the activity therefore consists of the prototype refinement and analysis tasks. The work product of the refinement task is the refined prototype and the analysis task more experiences on the performance of the prototype (Fig. 9.13).

The prototype construction, refinement and analysis tasks are not excplicitly defined in this capability pattern. The prototype construction and refinement tasks can contain a variety of game design tasks or even be modeled after another game design activity capability pattern depending on the needs of the project. It could be beneficial for a project, for example, to utilize human-centred design methods in the prototype construction task. This can be arranged by modeling the prototype construction task after the HCD design capability pattern.

The same goes for the analysis task. The analysis of a gameplay prototype should always include actually playing the prototype, but how to arrange the task is not explicitly defined in the capability pattern. Using HCD prototype evaluation capability pattern as a basis for prototype analysis task design could be a viable option for example when the design process has gone on for a while.

9.4 Work Products

In this chapter we'll discuss the work products in this game-based learning environment development process metamodel. Work products have not been the focus of any of the earlier parts of research. In this chapter the focus is in presenting the experiences of the case projects regarding work products of different tasks in GBLE development. Guidance regarding the work products is then presented based on the experiences recounted.

In the following chapters experiences in the case projects are recounted and guidance is derived from them work product by work product. First, concept documents, the work products of GBLE concept creation activity are discussed. That is followed discussion about GBLE design documentation discussion and finally discussion about software requirements specifications.

9.4.1 Concept Documents

The concept documentation have been discussed in the earlier result chapters related to the perceived challenges in game concept creation and learning goals definition tasks (see chapter 9.3.1. While not focusing on the work products, the capability pattern suggests that the tasks of game concept creation and learning goals definition should be combined into one. Therefore it makes sense to guide the process performers to combine the work product of these tasks as well.

GBLE concept document is the output of GBLE concept creation and GBLE concept refinement tasks. Another type of concept documentation, GBLE proposals, can also be seen as beneficil in the GBLE development. As described in chapter 9.3.1, GBLE concept document contains the contents of the game concept document and also learning goals definition. GBLE proposals are derivative of the game proposals of entertainment game development. A form of the GBLE proposal document was used as a work product in the Social Responsibility Game project. They contain the contents of the game proposal (described in chapter 4.4.2), learning goals definition and analysis of learning features.

The contents of the game concept document are described in chapter 4.4.1. The purpose of the game concept document is essentially the same as entertainment game design: to communicate the game idea to the stakeholders and funders of the project and to introduce the game concept to the development team.

In the game concept document the game's basic idea and activity is expressed

in a general level [Ryan 1999a]. The playability and game experience goals are set. The suggested contents of game concept document are same as in entertainment game development:

- Introduction, short and catching description of the game,
- Background, information about the game's background and connections to other products (this part is optional),
- Description, the description of gameplay of the main gameplay mode from the player's perspective,
- Most important features, a list of features that make the game unique (from three to five),
- Genre, description of the game's genre and it's position compared to previous games,
- Platform, to which hardware platforms the game is intended and
- Concept Art, extracts from the game art. [Ryan 1999a]

The rest of the advice given in entertainment game development literature also apply: the concept document should be concise, but describe the game in an unambiguous and clear manner [Ryan 1999a]. One should get a clear conception of the gameplay from reading the document [Freeman 2002].

The guidance for the learning goals definition part of the GBLE concept document is by necessity much more ambiguous. The learning sciences literature does not provide clearcut guidelines for the learnin goals definition. Learning goals were defined as part of the analysis phase in all of the case projects, but the form of the definition varied from project to project. Thus it can only be said that the definition of learning goals is an important part of the GBLE concept document and that the learning goals should be defined clearly and if possible in a way that the GBLE can later on be tested in grounds of fulfilling those goals.

The GBLE suggested content of the GBLE concept document presented is therefore as follows:

- Introduction, short and catching description of the GBLE and the gameplay,

- Background, information about the GBLE's background and connections to other products (this part is optional),
- Description, the description of gameplay of the main gameplay mode from the player's perspective,
- Most important features, a list of features that make the GBLE unique (from three to five),
- Learning goals definition, short, clear and testable description of learning goals,
- Genre, description of the GBLE's genre and it's position compared to previous games,
- Platform, to which hardware platforms the GBLE is intended and
- Concept Art, extracts from the GBLE art.

GBLE proposals can be used to further validate the feasibility of the GBLE concepts. They are produced from GBLE concept documents by means of market, technical and learning feature analysis. GBLE proposal is a refined version of the GBLE concept document with the reports of the aforementioned three analysis added. The suggested content of the GBLE proposal is:

- Introduction, short and catching description of the GBLE and the gameplay,
- Background, information about the GBLE's background and connections to other products (this part is optional),
- Description, the description of gameplay of the main gameplay mode from the player's perspective,
- Most important features, a list of features that make the GBLE unique (from three to five),
- Learning goals definition, short, clear and testable description of learning goals,
- Genre, description of the GBLE's genre and it's position compared to previous games,
- Platform, to which hardware platforms the GBLE is intended and

- Market analysis, target market of the game, comparisons to similar games in the market and appraisal of the features and their attractiveness,
- Technical analysis, list of experimental features, risks of development, alternative paths of development and appraisal of resources needed and
- Analysis of learning features, analysis of the gameplay and learning goals definition and the GBLE's potential to reach the desired goals with the current concept,
- Concept Art, extracts from the GBLE art.

9.4.2 Design Documents

Three of the case projects (Gameli V1, Social Responsibility Game and the Peatland Adventure) adapted the game design documentation format described in chapter 4.4.3. The adaptation process consisted of choosing the right content chapters for the document depending on the type and genre of the game in question.

The experiences of using the game design documentation format was positive; no negative experiences were recorded. In social responsibility game and the peatland adventure game projects the game design documentation was supplemented with learning design documentation; text documenting the features and content design of the game related to factors that were designed primarily to support learning.

The guidance related to game design documentation takes this into account. It is recommended that the project in question formulates the content of the game design document according to the type of the project and the game concept to be designed. Further guidance on using game design documents is availbale in chapter 4.4.3.

Software design was not seen as a potential problem in any of the case projects and therefore the question of what kinds of work products to use in software design of GBLE development projects did not surface during the study. The design documentation described in software engineering literature, including various UML diagrams such as class and state diagrams can be used.

9.4.3 Software Requirements Specification

Software requirements specification was noted as a significant activity in the two Gameli projects. The first project (Gameli V1) used prioritized feature and quality requirement

lists as the requirements specification format whereas the second project (Gameli V2) used use case diagrams and descriptions.

The success of the projects indicate that both methods are adequate for GBLE projects. Use cases were seen as very easy to use by the Gameli V2 team. The team also noted that the completeness of the requirements specification was easily validated based on the use case diagrams as use case diagrams allow the requirements specifiers to see every action a particular type of user can perform in the system at once. This suggests that use cases could be beneficial in projects where the requirements specification is deemed to be non-trivial.

The use case diagrams and descriptions are described in detail in chapter 5.4.1.

9.5 Conclusions

The aim of this part of the study was to provide a solution for the research question: **What kind of general prescriptive process model or metamodel could be constructed for GBLE development?**. The solution was pursued by constructing a prescriptive metamodel for GBLE development process based on the results achieved in the previous parts of the study.

The constructed metamodel was described in the previous chapters: overall process (chapter 9.1), role descriptions (chapter 9.2), activities (chapter 9.3 and work products (chapter 9.4). The process metamodel is based on Scrum process control model. The main features of the process are short work cycles called sprint adapted from Scrum and activity capability patterns that respond to the problem of arranging key activities in GBLE development identified in case projects. The process metamodel also describes and given guidance on key roles and work products.

The research question is approached in the way of making a practical example of a model. The solution provided must therefore be considered as one possible solution and not an all-encompassing framework providing a definitive and exhausting answer. The process metamodel provided in this chapter is a baseline for future GBLE development process modeling. The state of the model is included in the process description: it addresses specific problems and provides guidance to specific situations, but allows for freedom and experimentation with those parts that this research has provided no guidelines on.

The solution provided here is based on the experiences observed in the case projects. This includes the problems, key activities in the development process, development

methods and tasks that were experimented with. This means that only the subset of GBLE development process activities, entities and events existing in the case projects was considered. The metamodel provides a process for the kind of development work that was carried out in the case projects and a solution for the problems that were faced in the case projects.

The development of the metamodel to cover more ground is an on-going process. The first part of that is applying the process metamodel to a future GBLE development project. More research is needed to make the process more comprehensive and beneficial to future projects.

10 Conclusions

This study has explored the development process of digital game-based learning environments. The study approached the development process from the perspectives of software engineering and game procuction. This viewpoint was widened to a multidisciplinary perspective in order to paint a holistic picture of the process. The subject was researched by four development projects each having a different kind of focus into the development of game-based learning environments.

The research, applying the methodology of action research and development research, consists of the researcher actively participating in the case development projects as well as observing the events in the case projects. The research was also informed with reviewing the other developers in the projects and also the other stakeholders of the projects. The documentation of the projects were also analyzed for further insight into the development process. Experts outside the project team also analyzed the products of the projects and the process itself.

The focus of the research was divided into four main topics: 1) the composition and expertise of the development team, 2) elicitation and analysis of the development processes, 3) exploring the process control models and 4) constructing a prescriptive process metamodel for the development of digital game-based learning environments. All of the case projects contributed to knowledge of all the main topics.

The rest of the chapter is organized as follows: First the results of this study are summed up (chapter 10.1), then the study is evaluated for its contributions and limitations (chapter 10.2) and finally suggestions for future research are detailed (chapter 10.3).

10.1 Summary of the Results

The results of this study are discussed in this chapter.

10.1.1 Teams, Developers and Expertise in Development of Game-Based Learning Environments

The first research question was about the personnel and expertise needed in GBLE development projects.

The presence of game development and design knowledge seems to be important according to the experiences from the case projects, especially when the focus of the product is on the game-like qualities. This need can be lessened by coaching the development team but prior expertise remains valuable.

The knowledge of the content area of the game and learning with the game is structured around is important and the developers should acquire this expertise in some way. This knowledge can be present in the development team itself or the project can consult experts.

Skills and experience of working with users and sound understanding of user involvement methods manifest as important issues in game design to support learning. This includes experience on working with children and knowledge of human-centred design methods.

Three kinds of team compositions were explored in the case projects. The assumption based on literature was that the traditional software engineering team structure, enhanced by some game development roles, would be a suitable startpoint for exploring team structure. This structure was later modified by extending the development team with experts in producer roles. The final case project used a more flexible team structure that was based on game development discipline.

Adding game designer roles to the development team was seen as a beneficial decision in all of the projects. Moreover, the presence of game designer was seen as necessary from the start of the project by all case project teams.

There were two problems identified with the software engineering team structure. 1) The inclusion of subject matter, learning and HCD experts in consultant roles was seen as unoptimal and 2) the division of work between team members was found inefficient.

The development team members called for more direct involvement from the experts. The experts had similar views on the matter:

"It would have been good to discuss the [natural science] content and the implementation of game levels with the group concurrently." (The natural science and learning theory expert, researcher's own translation)

The uneven division of work was mostly related to game development roles. There was more work for game developers than there were people with necessary expertise.

This problem could have been eradicated with better estimating, but the problem remains that the software engineering structure of functional teams needs accurate planning of resource needs. Other methods of building the optimal team may be more efficient.

Allocating experts to producer roles was experimented with to solve the problems with experts as consults. The expert producers were active during the course of the project. This was seen as beneficial by the project teams and stakeholders. Other members of the development team felt they could reach the producers easily. The development team felt that they had adequate expertise available and the learning and content area expert could monitor the project and steer it towards the right track.

In one project the team was structured in a way that used the knowledge from game development discipline. The pre-production team was small, comprising only two members, but those two members had all the expertise necessary in the GBLE development. More people were added to the team as needed as the project went on.

This team structure was explored as an another way to overcome the drawback of the too inactive role for the learning theory and content area consultant experienced in the traditional software engineering team structure. This experiment was also a success as it was noted that there was enough expertise on those areas to produce a quality game for learning. From the positive experiences in this project it can be said that the active and responsible role of these experts is important especially in the pre-production team. It was also noted that a relatively small pre-production team could be more efficient and cost-effective in human resource terms than a large one used in the other projects.

10.1.2 Examination of the Process in Case Projects

The development processes of the four case projects were elicitated. The elicitated processes were described with SPEM2.0 process modeling language using Eclipse Process Framework Composer. The majority of development work was found to follow the existing activity models of game development, software engineering, learning intervention design and human-centred design.

Specific guidelines were drawn from the analysis of the elicitated processes. These include 1) integrating the HCD process as an integral part of the design and development process, 2) using iterative game design methods and 3) the risk of unambiguous project goals and a proposed ssolution for the problem..

Incremental development, especially when planned so that it was integrated with

HCD workshops between iterations, was seen as beneficial to tackle the experimental nature of GBLE development. IF HCD methods are to be used in GBLE development, the process needs to be modeled to take advantage of the feedback-cycle structure of HCD process.

Adapting iterative game design model was seen as beneficial in two case projects. This finding coincides with the projects with a focus towards ambitious game design. The iterative game design process consists of iterations where some subset of the game is designed and a representative prototype is constructed and tested. The test results of the previous iteration then inform the planning and carrying out of the next iteration. It was noted that the iterative game design model does not guarantee the success of the game design in terms of fulfilling its quality requirements but the quality of the design can be assessed in more accurate way than the evaluation of game design documents provides.

The unambiguous definition of goals for the GBLE to be developed was seen as a potential risk in all of the case projects. The case projects solved this in two ways. One project defined its working theory, held a HCD workshop to elicitate requirements for the intervention that would support the earning of the subjects in question and went on to analyze the set of requirement for the GBLE. In the other project the learning requirements were set after an experimental concept phase.

It was noted that to overcome this potential risk in GBLE development projects the project must evaluate if the goals of the project are defined unambiguously or can be further specified to be unambiguous with reasonable effort. If that cannot be achieved tasks must be planning to work towards setting unambiguous goals at the beginning of the project. These can include HCD workshops, requirements elicitation methods as well as literature reviews and reviews of promising similar products.

10.1.3 Process Control in Development of Game-Based Learning Environments

The third part of the study focused on the process control in the game-based learning environment development process. Process control is defined as the activities for planning, monitoring the work and reacting to emerging problems.

The research questions of this part of the study are: 1) What kind of process control model is most advantageous and feasible to the development of GBLE (both in general and especially in those projects concentrating on game design and human-centred design) and 2) what kind of adaptation of that process control model would

lead to a high-quality development process in this domain.

Two prominent process control models, defined process control and experimental process control, were evaluated in the context of GBLE development and the case projects. Experimental process control was exemplified with Scrum methodology. Three of the case projects used defined process control and one used a version of experimental process control. Three key activities or activity groups identified previously in the study were highlighted in the analysis of process models: GBLE concept creation, HCD activities and game design.

GBLE concept creation was identified as an activity with a lot of interdepence between tasks and potentially highly uncertain. As such, defined process control was deemed unfeasible for the activity. Instead experimental process control was seen as more beneficial for the activity.

Regarding HCD activities it was identified it would be possible to plan these around a defined process control framework. The handling of change inherent in HCD methods was however identified a phenomenon not handled well in defined process control. It was noted that experimental process control has mechanics for handling change in the middle of the project and therefore could be a better match with HCD activities.

Game design activities were identified as experimental in themselves. According to literature and experiences in case projects the quality of game design could be accurately evaluated only with playable prototypes. It was noted that defined process control does not have control mechanisms to handle that kind of activities. The iterative game design process was seen as an experimental process control model itself. Therefore it was noted that experimental process control would be a better choice for game design activities too.

Based on those choices the study recommends an experimental process control model for all GBLE development projects save for those with very little uncertainty and change expected. This part of the study continues to identify how Scrum could be adapted to GBLE development. It was noted that GBLE concept creation activity could be coupled with prototype construction to allow for evaluation of the concept at the concept phase. Game design activities were advised to follow the iterative game design activity model (see 4.4.4). HCD activities were identified to have distinct uses in different mechanisms of Scrum: HCD requirements gathering as a way to populate the project backlog and HCD prototype tests as ways to inform the sprint review meeting.

10.1.4 Process Metamodel for Development of Game-Based Learning Environments

The aim of the fourth part of the study was to build a foundation of guidelines to GBLE development in a form of a process metamodel. The research question this part of the study addressed was: What kind of general prescriptive process model or metamodel could be constructed for GBLE development?

The research question was answered by creating SPEM 2.0 (see chapter 2.2.2) process models based on the earlier results of the study. The resulting prescriptive process model consists of role, task and work product descriptions as well as activity and process descriptions. The activity descriptions use the role, task and work product descriptions and the process descriptions use all the other descriptions as needed.

The process metamodel uses Scrum as process control model. The process model descirbes the main roles of GBLE development: Learning expert, lead designer, HCD producer, user designer, scrum master and product owner. A number of supporting roles are also described. Activity capability patterns were modeled for key activities in the GBLE development process: GBLE concept creation, HCD activities and iterative game design. Guidance for the following work products was given: concept documentation, game design documentation and requirements specification.

10.2 Evaluation of the Study

The leading research question of this study was formulated in the introduction as follows: **What are the key aspects of a quality learning game development process?**. The results described in the previous chapter answer each part of the research question. The focus of the research is such that has not been undertaken before in the research of game-based learning environments. Most of existing studies only cover the development process briefly and concentrate on the quality features of GBLEs or the effect they have on learning.

The research started as a software engineering study with additional information from the game development discipline. During the course of the study it became evident that a mor holistic viewpoint was required to fulfill the research goals. This resulted in adopting the development research methodology and including learning sciences and instructional design literature into the literature review.

The research methodology in this study is based on the principles of development

research. Development research is a research method for researching and developing pedagogical interventions. The focus of development research is to provide prescriptive knowledge and useful solutions for a variety of design and development problems in education. The nature of development research makes the study a form of action research.

The nature of case projects as part of active research activity and also activity to be observed and analyzed adds facets of case study methodology in the study. This, enriched by the literature review with the twin goals of informing the research and the case projects, allowed for triangulation of the research information from three distinct viewpoints. First, as a active participationeer in the development projects, second as a researcher observing and analyzing the process it self and third, a comparison between the artefacts of the projects and the related literature.

The results of the study do not form a definitive answer on how to solve the problem of arranging the development of digital game-based learning environments. Instead, it describes the development process through the case projects and literature, highlights key issues and concerns and explores solutions to those issues and concerns. The results propose solutions and guidelines which are proven in light of literature and the case projects, but which are not, and are not advertised as general and all-encompassing.

The results of this study are related to the context of the case projects. They provide the future developers of digital game-based learning environments with a variety of tools to use when planning their next project. The literature review, although carried out at the first phase of the research, is still up-to-date and provides insight on the possibilities of GBLEs. The process metamodel is designed to be customizable; it provides an overall framework and pinpoints key points of the development process as well as provides guidance on them. It can also be expanded or updated. Other results are presented in context so that their significance and scope can be assessed by the readers.

As a whole, this study recommends the developers of game-based learning environments to approach the development of GBLEs as a experiment. It strives to provide readers with methods to control the development process while giving it the necessary freedom to ensure that solutions can be reached.

10.3 Future Research

As mentioned in the previous chapter, the empiric results of this study are related to the context of the case projects. Therefore it is logical to conclude that more research work in studying the GBLE development process that takes a more general approach is needed. This future research work could profit from taking this qualitative study as a foundation.

More specifically future research on adapting the Scrum process control model to GBLE development is needed. The results of this study indicate that GBLE development benefits from using an experimental process control model, but specifically using Scrum and adapting it to GBLE development has not been studied.

The GBLE development process metamodel is a work in progress. In its current form it covers the situations and activities that were highlighted as important or problematic during the study. The metamodel could be expanded to include more situations and activities. If the development of game-based learning environments as a disciplines solidifies and ages, it is probable that more common methods of working emerge. The discipline would benefit from collecting these methods of working to an open process model such as the process metamodel presented in this study.

References

[Aarseth 1997] Aarseth, E. J. (1997). Cybertext – Perspectives on Ergodic Literature. London: Johns Hopkins.

[Abt 1970] Abt, C. (1970) Serious Games. New York: Viking Press.

[Adams 2005] Adams, E. (2005) Story Game 2 Workshop. Games & Storytelling Lecture Series. Helsinki 19.-23.4.2005.

[Agile Alliance 2002] Agile Alliance (2002) Agile Manifesto. <http://www.agilealliance.org/>.

[Ahearn 2002] Ahearn, L. (2002) The Game Proposal, Part One: The Basics. Verkkojulkaisussa Gamasutra (http://www.gamasutra.com/).

[Ahearn 2006] Ahearn, L. (2006) The Game Proposal, Part Two: The Contents. Verkkojulkaisussa Gamasutra (http://www.gamasutra.com/).

[Amory et al. 1998] Amory, A.& Naicker, K., Vincent, J. & Adams, C. (1998) Computer Games as Learning Resource. Saatavilla www-muodossa: <URL: http://www.und.ac.za/und/biology/staff/amory/edmedia98.html>. Luettu 15.2.2005.

[Andriole 1990] Andriole, S.J. (1990) Information System Design Principles for the 90s: Getting it Right. Fairfax(VG): AFCEA International Press.

[Arlow & Neustadt 2002] Arlow, J. & Neustadt, I. (2002) UML and the Unified Process. Practical Object-Oriented Analysis & Design. London: Pearson Education.

[Avedon & Sutton-Smith 1971] Avedon, E.M. & Sutton-Smith, B. (1971) The Study of Games. New York: John Wiley.

[Baltra 1990] Baltra, A. (1990) Language learning through computer adventure games. Simulation and gaming 21, 445–452.

[Barbacci et al. 1998] Barbacci, M., Carriere, S., Feiler, P., Kazman, R., Klein, M., Lipson, H., Longstaff, T. & Weinstock, C. (1998) Steps in an Architecture Tradeoff Analysis Method: QUality Attribute Models and Analysis. Technical Report. Pittsburgh(PA): Carnegie Mellon University.

[Barsalou 1999] Barsalou, L. W. (1999) Perceptual Symbol Systems. In Behavorial and Brain Sciences, 22.4 (1999): 577-660.

[Baskerville et al. 1996] Baskerville, R. & Wood-Harper, A. T. (1996) A Critical Perspective on Action Research as a Method for Information Systems Research. In Journal of Information Technology, Vol. 11.

[Bass & Kazman 2003] Bass, L. & Kazman, R. (2003) Software Architecture in Practice. Boston(MA): Addison-Wesley.

[Beck 1999] Beck, K. (1999) Embracing Change With Extreme Programming. In Computer, Vol 13., No. 10. Pp. 70-77.

[Beck 2000] Beck, K. (2000) Emergent Control in Extreme Programming. In Cutter IT Journal, Vol 13, No. 11. Pp. 22-25.

[Becta 2001] Becta (2001) Computer games in Education Project. Saatavilla www-muodossa <URL:http://www.becta.org.uk/research/research.cfm?section=1&id=2826>. Luettu 18.1.2005.

[Bethke 2003] Bethke, E. (2003) Structuring Key Design Elements. Teoksessa Game Development and Production. Wordware Publishing.

[Björk & Holopainen 2005] Björk, S. & Holopainen, P. (2005) Patterns in Game Design. Hingham (MA): Charles River Media.

[Brodlie et al. 1992] Brodlie, K. W., Carpenter, L. A., Earnshaw, L. A., Gallop, L. A., Hubbard, R. J., Mumford, A. M., Osland, C. D. & Quarendon, P. (1992) (eds.), Scientific Visualization, Techniques and Applications, Springer-Verlag.

[Budde et al. 1991] Budde, R., Kautz, K., Kuhlenkamp, K. & Züllighoven, H. (1991) Prototyping. An Approach to Evolutionary System Development. Berlin: Springer-Verlag.

[Caillois 1961] Caillois, R. (1961) Man, Play and Games. Free Press of Glencoe.

[Cassel & Jenkins 1998] Cassel, J. & Jenkins, H. (1998) (eds.) From Barbie to Mortal Kombat. Cambridge: The MIT Press.

[Cazden 1981] Cazden, C. (1981) Performance before Competence: Assistance to Child Discourse in the Zone of Proximal Development. In Quarterly Newsletter of the Laboratory of Comparative Human Cognition 3.1 (1981): 5-8.

[Chamberlin 2003] Chamberlin, B (2003) Creating Entertaining Games With Educational Content. Case Studies of User Experiences With The Children's Website, Food Detectives Fight BAC!. Virginia: University of Virginia.

[Charette 1989] Charette, R. (1989) Software Engineering Risk Analysis and Management. New York(NY): McGraw Hill.

[Church 1999] Church, D. (1999) Formal Abstract Design Tools. In the online Magazine Gamasutra (http://www.gamasutra.com/).

[Cockburn 1997] Cockburn, A. (1997) Using "V-W" Staging to Clarify Spiral Development. Julkaisussa OOPSLA'97 Practitioner's Report, Humans and Technology technical report. Atlanta (GA).

[Cockburn 2001] Cockburn, A. (2001) Agile Software Development. Boston(MA): Addison-Wesley.

[Combemale et al. 2006] B. Combemale, A. Caplain, X. Crégut, and B. Coulette (2006) Towards a rigorous process modeling with spem. Poster at the ICEIS'06 Conference, 2006.

[Cook 2002] Cook, D. (2002) Evolutionary Design. A practical process for creating great game designs. Verkkojulkaisussa GameDev (http://www.gamedev.net).

[Costikyan 1994] Costikyan, G. (1994) I Have No Words & I Must Design. Julkaisussa Interactive Fantasy, The Journal of Role-Playing and Story-Making Systems. Numero 2. London: Hogshead Publishing.

[Crawford 1982] Crawford, C. (1982) The Art of Computer Game Design. Available on th World Wide Web (http://www.vancouver.wsu.edu/fac/peabody/gamebook/Coverpage.html).

[Crawford 2003] Crawford, C. (2003) Chris Crawford on Game Design. Berkeley (CA): New Riders Publishing.

[Crinnion 1991] Crinnion, J. (1991) Evolutionary Systems Development. A practical guide to the use of prototyping within a structured systems methodology. New York: Plenum Press.

[Cronjé 2005] Cronjé, J. (2005). Paradigms Regained: Toward Integrating Objectivism and Constructivism in Instructional Design and the Learning Sciences, in Educational Technology Research & Development, Vol. 54, No. 4, pp. 387-416, Mahwah, NJ: Association for Educational Communications and Technology.

[Dai et al. 2006] Dai, L., Cooper, K. & Wong, E. (2006) Modeling and Analysis of Performance Aspects for Software Architecture: A UML-Based Approach. In International Journal of Software Engineering. Vol. 16, No. 3, p. 347-378. World Scientific Publishing Company.

[Davidson 2004] Davidson, D. (2004) (toim.) Second generation e-learning: serious games. England, Bradford: Emerald Group Publishing Limited.

[Davis 1992] Davis, Alan M. (1992) Operational Prototyping: A new Development Approach. Julkaisussa IEEE Software, September 1992. P. 71.

[Delta Method] The Delta Method Handbook. (http://www.deltamethod.net/)

[DeMaria 2005] DeMaria, R. (2005) Postcard From The Serious Games Summit: How the United Nations Fights Hunger with Food Force. Verkkojulkaisussa Gamasutra (http://www.gamasutra.com/).

[Design-Based Research Collective 2003] Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry, in Educational Researcher, Vol. 32, No. 1, pp. 5-8.

[Dick & Carey 1991] Dick, W. & Carey, L. (1991). The systematic design of instruction (2nd ed.). Glenview, IL: Scott, Foresman, Glenview.

[Dijkstra 1997] Dijkstra, S. (1997). Theoretical Foundations of Instructional Design: An Introduction and overview. In Tennyson, R.D., Scholt, F., Seel, N. & Dijkstra, S. (Eds.) Instructional Design: International Perspective Volume 1. Theory, Research and Models. Mahwah, NJ: Lawrence Erlbaum Associates.

[Dobrica & Niemelä 2002] Dobrica, L. & Niemelä, E. (2002) A Survey on Software Architecture Analysis Methods. In IEEE Transactions on Software Engineering, Vol. 28, No. 7.

[Druin 1996] Druin, A. (1996) Designing multimedia environments for children. New York: John Wiley & Sons cop.

[Eclipse Process Framework Project 2008] Eclipse Process Framework Project (2008) Eclipse Process Framework Practice Library. <http://epf.eclipse.org/wikis/epfpractices/>.

[Education Arcade 2005] Education Arcade website. <http://www.educationarcade.org/revolution/>

[Egenfeldt-Nielsen 2004] Egenfeld-Nielsen, S. (2004) Practical barriers in using educational computer games. Teoksessa D. Davidson (toim.) Second generation e-learning: serious games. England, Bradford: Emerald Group Publishing Limited, 12 (1), 18–21.

[Elliott et al. 2002] Elliott, J., Adams, L. & Bruckman, A. (2002) No Magic Bullet: 3D Video Games in Education. In Proceedings of ICLS 2002, Seattle, WA, October 2002.

[Ermi & Mäyrä 2005] Ermi, L. & Mäyrä, F. (2005) Fundamental Components of the Gameplay Experience: Analysing Immersion. In de Castell, S. & Jenson, J. (eds.) Changing Views: Worlds in Play - Selected Papers of the 2005 Digital Games Research Association's Second International Conference, pp. 15–27. Vancouver: Digital Games Research Association & Simon Fraser University.

[Extreme Programming 2006] The Extreme Programming Website. <http://www.extremeprogramming.org>.

[Fabricatore 2000] Fabricatore, C. (2000) Learning and videogames: an unexploited synergy. Saatavilla www-muodossa: <URL: http://learndev.org/dl/FabricatoreAECT2000.PDF>. Luettu 24.1.2005.

[Food Force 2006] Food Force website. <http://www.food-force.com/>.

[Fowler 2001] Fowler, M. (2001) The New Methodology. <http://www.martinfowler.com/articles/newMethodology.html>.

[Freeman 2002] Freeman, T. (2002) Creating a Great Design Document. In the web publication Gamasutra (http://www.gamasutra.com/).

[Funk & Buchman 1996] Funk, J. B. & Buchman, D. D. (1996). Children's Perceptions of Gender Differences in Social Approval for Playing Electronic Games. In: Sex Roles, no 35.

[Gander 1998] Gander, S.L. (1998) Case Study: Development of a Corporate Learning Game. Cerner Virtual University. Available online (http://www.usq.edu.au/electpub/e-jist/docs/old/vol3no2/article3/).

[Gee 2003] Gee, J. P. (2003) What video games have to teach us about learning and literacy. New York: Palgrave Macmillan.

[Gee 2005] Gee, J.P. (2005) Good Video Games and Good Learning. Available online <url:http://www.academiccolab.org/resources/documents/Good_Learning.pdf>.

[Germain & Robillard 2008] Germain, É. & Robillard, P.N. Towards software process patterns: An empirical analysis of the behavior of student teams. Information and Software Technology, 50:1088–1097, 2008.

[Glenberg 1997] Glenberg, A. M. (1997) What is Memory For. In Behavioral and Brain Sciences, 20.1: (1997): 1-55.

[Haikala & Marijärvi 2001] Haikala, I. & Marijärvi, J. (2001)Ohjelmistotuotanto. Pieksämäki: Talentum Media.

[Hannafin 1995] Hannafin, M.J. (1995). Open-ended learning environments: Foundations, assumptions, and implications for automated design. In R.D. Tennyson & A.E. Barron (Eds.), Automating instructional design: Computer-based development and delivery tools (pp. 101-130). Berlin: Springer.

[Hedberg & Sims 2001] Hedberg, J. & Sims, R. (2001) Speculations on Design Team Interactions. In Journal of Interactive Learning Research, Vol. 12, no. 2, pp. 193-208. Chesapeake(VA): Association for the Advancement of Computing in Education.

[Hoadley 2002] Hoadley, C. (2002). Creating context: Design-based research in creating and understanding CSCL. In G. Stahl (Ed.), Computer Support for Collaborative Learning 2002 (pp. 453 - 462). Mahwah, NJ: Lawrence Erlbaum Associates.

[Huizinga 1955] Huizinga, J. (1955) Homo Ludens: A Study of the Play Element in Culture. Beacon.

[Hunicke et al. 2003] Hunicke, R., LeBlanc, M. & Zubek, R. (2003) MDA: A Formal Approach to Game Design and Game Research. Saatavilla pdf-muodossa osoitteesta <url:http://www.cs.northwestern.edu/h̃unicke/pubs/MDA.pdf>.

[Huntsman 2000] Huntsman, T. (2000) A Primer for the Design Process, Part 1: What to Do. In the online publication Gamasutra (http://gamasutra.com/features/20000630/huntsman_01.htm).

[Hämäläinen et al. 2008] Hämäläinen, R., Manninen, T., Järvelä, S. & Häkkinen, P. (2008) Learning to Collaborate: Designing Collaboration in a 3-D Game Environment. In The Internet and Higher Education, Volume 9, Issue 1, pp. 47-61. Amsterdam, The Netherlands: Elsevier Science.

[International Organization for Standardization 1999] International Organization for Standardization. (1999) ISO 13407. Human-centred design processes for interactive systems. International Organization for Standardization. Geneve.

[Jacobson et al. 1992] Jacobson, I., Christerson, P. & Overgaard, G. (1992) Object-Oriented Software Engineering: a Use Case Driven Approach. Reading(MA): Addison Wesley.

[Jacobson et al. 1999] Jacobson, I., Rumbaugh, J. & Booch, G. (1999) The Unified Software Development Process The complete guide to the Unified Process from the original designers. Reading(MA): Addison Wesley.

[Jayaratna 1994] Jayaratna, N. (1994) Understanding and evaluating methodologies. NIMSAD, a systemic framework. London: McGraw-Hill.

[Jepsen et al. 1989] Jepsen, L.O., Mathiassen, L. & Nielsen, P.A. (1989) Back to Thinking Mode - Diaries as a Medium for Effective Management of Information Systems Development. Behaviour and Information Technology. Vol. 8.

[Juul 2005] Juul, J. (2005) Half-Real. Video Games between Real Rules and Fictional Worlds. Cambridge(MA): MIT Press.

[Juul 2006] Juul, J. (2006) Game Studies. Lecture series & Workshop. Jyväskylä, May 2006.

[Järvinen 2003] Järvinen, A. (2003)

[Järvinen 2007] Järvinen, A. (2007)

[Kalermo & Rissanen 2002] Kalermo, J. & Rissanen, J. (2002) Agile Software Development in Theory and Practice. Jyväskylä: University of Jyväskylä.

[Kankaanranta 2007] Kankaanranta, M. (2007) Pelien kentillä ja oppimisen maailmoissa. (In the fields of games and worlds of learning.) In H. Haapamäki-Niemi & S. Noponen (Eds.) Elämää bittien kanssa - opiskelu verkossa ja Internetin mahdollisuudet. Äidinkielen opettajain liiton vuosikirja. Pp. 73-88. Helsinki, Finland: Äidinkielen opettajain liitto.

[Kankaanranta et al. 2007] Kankaanranta, M., Kirjavainen, A. & Nousiainen, T. (2007) (Eds.) Oppimispelien suunitteluprosessin rakentaminen. (Constructing design process for learning games) Final report of CoEduGame project. Jyväskylä: University of Jyväskylä. Agora Center & Institute of Educational Research.

[Karat 1997] Karat, J. (1997) Evolving the scope of user-centered design. Communications of the ACM, 40, 7, 33-38.

[Kazman et al. 1998] Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H. & Carriere, J. (1998) The Architecture Tradeoff Analysis Method. In Proceedings of the International Conference on Engineering of Complex Computer Systems, Monterey, CA.

[Kirby et al. 2005] Kirby, J.A., Hoadley, C.M. & Carr-Chellman, A.A. (2005). Instructional Systems Design and the Learning Sciences: A Citation Analysis. In Educational Technology Research & Development, Vol. 53, No. 1, pp. 37-48.

[Klein et al. 1993] Klein, M., Ralya, T., Pollak, B., Obenza, R. & Gonzales Harbour, M. (1993) A Practitioner's Handbook for Real-Time Analysis. Boston(MA): Kluwer Academic.

[Kolodner 2004] Kolodner, J. L. (2004). The learning sciences: Past, present, future. Educational Technology, 44(3), 34 - 40, Englewood Cliffs, NJ: Educational Technology Publications.

[Koskinen et al. 2004] Koskinen, J., Ahonen, J., Tilus, T., Sivula, H. & Lintinen, H. (2004) Business Information Systems - BIS 2005: Using NIMSAD Meta Framework. Jyväskylä: Univeristy of Jyväskylä.

[Koskinen 1999] Koskinen, M. Metamodelling Approach to Process Concept Customisation and Enactability in MetaCASE. Jyväskylä University Printing House, Jyväskylä, 1999.

[Krief 1996] Krief, P. (1996) Prototyping with Objects. Prentice-Hall.

[Lahti 2008] Lahti, K. (2008) Adapting to and assessing scrum in game development. Master's thesis, Swedish School of Economics and Business Administration, Helsinki, Finland.

[Lainema 2003a] Lainema, T. (2003) A Continuously Processed Business Game Construction for Business Process Training. TUCS Technical Report 499, ISBN: 952-12-1101-6.

[Lainema 2003b] Lainema, T. (2003) Enhancing Organizational Business Process Perception – Experiences from Constructing and Applying a Dynamic Business Simulation Game. Turku, Finland: Turku School of Economics and Business Administration.

[Lanzara & Mathiassen 1985] Lanzara, G.F. & Mathiassen, L. (1985) Mapping Situations within a Systems Development Project. Information and Management. Vol. 8. No. 1.

[Laurel 2003] Laurel, B. (2003) Design Research: Methods and Perspectives. Cambridge(MA): MIT Press.

[Luerig 2003] Luerig, C. (2003) Efficient XML Reading for Game Development. Verkkojulkaisussa Gamasutra (http://www.gamasutra.com/).

[Mathiassen 1996] Mathiassen, L. (1996) Information Systems Development. Reflections on a Discipline. Accounting, Management & Information Technology. Vol. 6. No. 1/2.

[McCandliss et al. 2003] McCandliss, B.D., Kalchman, M. & Bryant, P. (2003) Design Experiments and Laboratory Approaches to Learning: Steps Toward Collaborative Exchange. In Educational Researcher, Vol. 32, No.1. Available on WWW: <URL:http://www.aera.net/pubs/er/pdf/vol32_01/AERA320106.pdf>. Washington(DC): American Educational Research Association.

[McFarlane et al. 2002] McFarlane, A., Sparrowhawk, A. & Heald, Y. (2002) Report of the educational use of games. Teachers Evaluating Educational Multimedia. Saatavilla www-muodossa: <URL: http://www.teem.org.uk/publications/teem_gamesined_full.pdf>.

[McGuire 2006] Mcguire, R. (2006) Paper burns: Game design with agile methodologies. Gamasutra The Art & Business of Making Games, June 2006.

[Mäkinen 2003] Mäkinen, V. (2003) Analysis Of Use Case Approaches To Requirements Engineering. Jyväskylä: Tietojärjestelmätieteen laitos, Jyväskylän yliopisto.

[Mönkkönen & Enkenberg 1996] Mönkkönen, H. & Enkenberg, J. (1996) Situated Learning and instructional Design. Implementation of Situated Learning in Computer-Based Environments. Joensuu, Finland: University of Joensuu.

[Nousiainen 2005] Nousiainen, T. (2005) Lapset suunnittelukumppaneina oppimisohjelmiston kehityksessä. Master's Thesis in Computer Science. Jyväskylä: Tietojenkäsittelytieteiden laitos.

[Nousiainen 2008] Nousiainen, T. (2008) Children's Involvement in the Design of Game-Based Learning Environments. Ph.D. thesis, University of Jyväskylä, 2008.

[Novak 2005] Novak, J. (2005) Game Development Essentials: An Introduction. Clifton Park(NY): Thomson Delmar Learning.

[Object Management Group 2003] Object Management Group (2003) UML Profile for Schedulability, Performance, and Time Specification, OMG Documents ptc/2003-03-02.

[Object Management Group 2007] Object Management Group (2007) Software & Systems Process Engineering Metamodel Specification, version 2.0. Available online (http://www.omg.org/docs/ptc/07-08-07.pdf).

[Ogunnaike 1994] Ogunnaike, B. A. & Ray, W. H. (1994) Process dynamics, modeling, and control. Oxford U.P.,New York.

[Palmer & Felsing 2002] Palmer, S.R. & Felsing, J.M. (2002) A Practical Guide to Feature-Driven Development. Upper Saddle River (NJ): Prentice-Hall.

[Parlett 1999] Parlettt, D. (1999) The Oxford History of Board Games. Oxford: Oxford University Press.

[Paulk 2001] Paulk, M. C. (2001) Extreme Programming from a CMM Perspective. Software, Vol. 18, No. 6. Pp. 19-26.

[Pikkarainen et al. 2008] Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P. & Still, J. (2008) The Impact of Agile Practices on Communication in Software Development. In Empirical Software Engineering, 13:303–337.

[Plomp 2002] Plomp, T. (2002) Some reflections on development research (DR). Speaking notes for a breakfast meeting at the Faculty of Education, University of Pretoria. August 2002.

[Plummer 2004] Plummer, J. (2004) A Flexible and Expendable Architecture for Computer Games. Phoeniz(AZ): Arizona State University.

[Porres 2001] Porres, I. (2001) Modeling and Analyzing Software Behaviour in UML. väitöskirja, Tietojenkäsittelytieteiden laitos, Turun yliopisto.

[Prensky 2001] Prensky, M. (2001) Digital Game-Based Learning. New York: McGraw-Hill.

[Pressman 2000] Pressman, R.S. (2000) Software Engineering A Practitioner's Approach. Maidenhead: McGraw-Hill.

[Racheva et al. 2008] Racheva, Z., Daneva, M. & Buglione, L. (2008) Complementing measurements and real options concepts to support interiteration decision-making in agile projects. In 34th Euromicro Conference Proceedings. IEEE Computer Society.

[Ravenscroft & Matheson 2002] Ravenscroft, A. & Matheson, M. P. (2002) Developing and evaluating dialogue games for collaborative e-learning. Journal of Computer Assisted Learning 18 (1), 93–101.

[RealGame 2005] RealGame website. <http://www.realgame.fi>.

[Rising & Janoff 2000] Rising, L. and Janoff, N.S. The scrum software development process for small teams. IEEE Software, July/August 2000.

[Rollins & Adams 2003] Rollins, A. & Adams, A. (2003) Andrew Rollins and Ernest Adams on Game Design. Berkeley (CA): New Riders Publishing.

[Rollins & Morris 2004] Rollins, A. & Morris, D. (2004) Game Architecture and Design. A New Edition. Berkeley (CA): New Riders Publishing.

[Ryan 1999a] Ryan, T. (1991a) The Anatomy of a Design Document, Part 1. Verkkojulkaisussa Gamasutra (http://www.gamasutra.com/).

[Ryan 1999b] Ryan, T. (1991b) The Anatomy of a Design Document, Part 2. Verkkojulkaisussa Gamasutra (http://www.gamasutra.com/).

[Ryan 1999c] Ryan, T. (1991c) Beginning Level Design, Part 1: Level Design Theory. Verkkojulkaisussa Gamasutra (http://www.gamasutra.com/).

[Salen & Zimmerman 2003] Salen, K. & Zimmerman, E. (2003) Rules of Play. Game Design Fundamentals. Cambridge(MA): MIT Press.

[Schwaber & Beedle 2001] Schwaber, K. & Beedle, M. (2001) Agile Software Development with Scrum. Upper Saddle River, NJ: Prentice-Hall.

[Scriven 1972] Scriven, M. (1972). Pros and Cons about goal-free evaluation, Evaluation Comment, 3(4), 1-8.

[Serious Games 2005] Serious Games Initiative Website. <http://www.seriousgames.org/>.

[Silber 2007] Silber, K.H. (2007). A Principle-Based Model of Instructional Design; A New Way of Thinking About and Teaching ID. In Educational Technology, Vol. 47, No. 5, Englewood Cliffs, NJ: Educational Technology Publications.

[Smith & Williams 1993] Smith, C. & Williams, L. (1993) Software Performance Engineering: A Case Study Including performance Comparison with Design Alternatives. In IEEE Transactions on Sfotware Engineering Vol 19, No. 7, p. 720-741.

[Smith 2003] Smith, H. (2003) Orthogonal Unit Differentation. Konferenssiesitys Game Developers' Conference 2003:ssa.

[Smith 1991] Smith, M.F. (1991) Software Prototyping: Adoption, practice and management. London: McGraw-Hill.

[Snyder 2003] Snyder, C. (2003) Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces. San Francisco, CA: Morgan Kaufmann.

[Sommerville 2001] Sommerville, I. (2001) Software engineering. 6th Edition. Harlow: Addison-Wesley.

[Sommerville & Sawyer 1997] Sommerville, I. & Sawyer, P. (1997) Requirements Engineering: A Good Practice Guide. Chichester: John Wiley & Sons.

[Sotamaa et al. 2005] Sotamaa, O., Ermi, L., Jäppinen, A., Laukkanen, T., Mäyrä, F. & Nummela, J. (2005) The Role of Players in Game Design: A Methodological Perspective. In Proceedings of the 6th DAC Conference. Copenhagen: IT University of Copenhagen. p. 34-42.

[Squire et al. 2004] Squire, K., Barnett, M., Grant, J. & Higginbotham, T. (2004) Electromagnetism supercharged!: learning physics with digital simulation games. In Proceedings of the 6th international conference on Learning sciences (Santa Monica CA, 2004), International Conference on Learning Sciences, pp. 513-520.

[Stubbs & Pal 2003] Stubbs, M. & Pal, J. (2003) The development, design and delivery of a retail simulation. In British Journal of Educational Technology, Vol. 34, No. 5, pp. 651–661. Oxford, UK: Blackwell Publishing.

[Suits 1990] Suits, B. (1990) Games, Life, and Utopia. Boston: David R. Godine.

[Sykes & Federoff 2006] Sykes, J. & Federoff, M. (2006) Player-centred game design. In Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montreal, Quebec, Canada, April 22-27, 2006.

[Taber & Fowler 2000] Taber, C. & Fowler, M. (2000) An Iteration in a Life of a XP Project. Cutter IT Journal, Vol. 13, No. 11. Pp. 13-21.

[Tennyson 1997] Tennyson, R. (1997). A System Dynamics Approach to Instructional Systems Development. In Tennyson, R.D., Scholt, F., Seel, N. & Dijkstra, S. (Eds.) Instructional Design: International Perspective Volume 1. Theory, Research and Models. Lawrence Erlbaum Associates. 1997. Mahwah, NJ.

[Tennyson & Schott 1997] Tennyson, R.D. & Schott, F. (1997) Instructional Design Theory, Research, and Models. In Tennyson, R.D., Scholt, F., Seel, N. & Dijkstra, S. (Eds.) Instructional Design: International Perspective Volume 1. Theory, Research and Models. Lawrence Erlbaum Associates. 1997. Mahwah, NJ.

[Terävä 2005] Terävä, H. Software process modeling with eclipse process framework and spem 2.0. Master's thesis, University of Turku, 2007.

[Tripp et al. 2004] Tripp, L., Alan, A., Moore, J., Bourque, P. & Dupuis, R. (2004) (toim.) SWEBOK - Guide to the Software Engineering Body of Knowledge. 2004 Version. Los Alamitos(CA): IEEE Computer Society.

[van den Akker 1999] van den Akker, J. (1999) Principles and Methods of Development Research. In J. van den Akker, R.M. Branch, K. Gustafson, N. Nieveen & T. Plomp (Eds.) Design approaches and tools in education and training (pp. 1-14). Boston: Kluwer Academic.

[Wiegers 2000] Wiegers, K. E. (2000) When Telepathy Won't Do: Requirements Engineering Key Practices. Saatavilla WWW-muodossa osoitteesta <url: http://www.processimpact.com/articles/telepathy.html>.()

[Williams et al. 2000] Williams, L., Kessler, R.R., Cunningham, W. & Jeffries, R. (2000) Strengthening the Case for Pair Programming. In Software, Vol. 17, No. 4. Pp. 19-25.

[Zimmerman 2003] Zimmerman, E. (2003) Play as Research: The Iterative Design Process. Teoksessa Laurel, B. (toim.) Design Research: Methods and Perspectives. Cambridge (MA): MIT Press.

A Role Definitions from Eclipse Process Framework

This appendix chapter describes the roles used in the GBLE development metamodel (see chapter 9) that are defined in the core role definition of the Eclipse Process Framework Practice Library [Eclipse Process Framework Project 2008]. The descriptions are cited from the EPF Practice Library [Eclipse Process Framework Project 2008].

A.1 Software Architect

The Architect is responsible for defining the software architecture, which includes making the key technical decisions that constrain the overall design and implementation of the system.

A.1.1 Role Description

The person in this role leads or coordinates the technical design of the system and has overall responsibility for facilitating the major technical decisions expressed as software architecture. This typically includes identifying and documenting the architecturally significant aspects of the system as views that describe requirements, design, implementation, and deployment.

This role is also responsible for providing the rationale for these decisions, balancing the concerns of the various stakeholders, reducing technical risks, and ensuring that decisions are effectively communicated, validated, and followed.

This role works closely with project managers in staffing and planning the project, because it is recommended that the team be organized around the architecture.

This role also works closely with analysts and developers to make sure that the architecturally significant requirements are assigned to the proper components of the system.

A.1.2 Skills

Architects must be well-rounded people with maturity, vision, and a depth of experience that allows for grasping issues quickly and making educated, critical judgments in the absence of complete information. Specifically, the person must possess this combination of qualifications:

- Experience in both problem and software engineering domains, with evidence of a thorough understanding of the requirements to solve the problem and active participation in software development. If there is a team, this experience can be represented by different team members, but at least one person must be able to describe the overall vision for the project.

- Leadership ability to motivate and maintain momentum for the technical effort across the various teams and to make critical decisions under pressure, plus make those decisions stick. To be effective, this role must have the authority to make technical decisions. This role cannot lead by decree, but only by the consent of the rest of the project team. To be effective, this person must earn the respect of the team members, project managers, the customer, and the user community, as well as the management team.

- Excellent communication skills to earn trust, persuade, motivate, and mentor. The person in this role must have good communication skills, both verbally and in writing.

- Critical review skills to make sure that the requirements to be built are clear and consistent and to make sure that the developed system adheres to the architecture.

- Goal-oriented and proactive orientation with a relentless focus on results. This person is the technical driving force behind the project, not a visionary or dreamer. The career of a successful architect is a long series of sub-optimal decisions made in uncertainty and under pressure. Only those who can focus on doing what needs to be done will be successful.

From an expertise standpoint, the Architect also needs to show both design and implementation abilities. However, from the design perspective, the effective Architect typically exhibits these traits:

- Tends to be a generalist, rather than a specialist, who knows many technologies at a high level rather than a few technologies at the detailed level
- Makes the broader technical decisions, thereby demonstrating broad knowledge and experience, as well as communication and leadership skills

A.1.3 Assignment Approaches

The person in this role should be engaged in the project from start to finish.

For smaller projects, a single person can act as both Architect and project manager. However, it is better to have these roles performed by different people to ensure that the pressures of one role does not cause neglect of the other role. The Architect and Project Manager must work together closely.

For systems of scale, it is a common best practice to have an architecture board that is populated by the architects of each system, plus one or two chief architects. In such cases, the members of the architecture board collectively play the role of the Architect.

A.2 Software Developer

A.2.1 Role Description

The Developer is responsible for developing a part of the system, including designing it to fit into the architecture, possibly prototyping the user interface, and then implementing, unit-testing, and integrating the components that are part of the solution.

A.2.2 Skills

The person in this role needs the following knowledge, skills, and abilities:

- Enough expertise and experience to define and create technical solutions in the project's technology
- Ability to understand and conform to the architecture
- Ability to identify and build developer tests that cover required behavior of the technical components
- Ability to communicate the design in a way that other team members understand

A.2.3 Assignment Approaches

A person performing this role can have specialized skills in a particular technical area but should also have a broad understanding of all of the technologies involved to be able to work with other technical team members.

Even in the smallest team, multiple individuals should be working together to create the technical solution. In small, agile teams, this role is often shared among several team members who also perform other roles.

A.3 Stakeholder

This role represents interest groups whose needs must be satisfied by the project. It is a role that may be played by anyone who is (or potentially will be) materially affected by the outcome of the project.

A.4 Tester

The Tester is responsible for the core activities of the test effort. Those activities include identifying, defining, implementing, and conducting the necessary tests, as well as logging the outcomes of the testing and analyzing the results.

A.4.1 Role Description

The person in this role is primarily responsible for the following tasks:

- Identifying the tests that need to be performed
- Identifying the most appropriate implementation approach for a given test
- Implementing individual tests
- Setting up and running the tests
- Logging outcomes and verifying that the tests have been run
- Analyzing and guiding the recovery from execution errors
- Communicating test results to the team

A.4.2 Skills

A person performing this role needs the following skills:

- Knowledge of testing approaches and techniques
- Diagnostic and problem-solving skills
- Knowledge of the system or application being tested (desirable)
- Knowledge of networking and system architecture (desirable)

Where automated testing is required, consider requiring these additional qualifications:

- Training in the appropriate use of test automation tools
- Experience using test automation tools
- Programming skills
- Debugging and diagnostic skills

Note: Specific skill requirements vary depending on the type of testing that you are conducting. For example, the skills needed to successfully use system load-testing automation tools are different from those needed for the automation of system functional testing.

A.4.3 Assignment Approaches

This role can be assigned in the following ways:

- Assign one or more testing staff members to perform this role. This is a fairly standard approach and is particularly suitable for small teams, as well as for teams of any size where the team is made up of an experienced group of testers of relatively equal skill levels.
- Assign one or more testing staff members to perform only this role. This works well in large teams. It is also useful to separate responsibilities when some of the testing staff has more test automation experience than other team members.
- Assign one (or more) team member who is already playing other role in the project to be responsible for the testing of some part of the system's capabilities. The team member must have the appropriate test skills