Digital Educational Games: Methodologies for Development and Software Quality

by

Serdar Aslan

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

September 30th, 2016
Blacksburg, Virginia

Key words: Digital game development, educational game development, game development life cycle, game development methodology, game development processes, game development workflows, game software quality evaluation, and game-based learning.

Digital Educational Games: Methodologies for Development and Software Quality

by

Serdar Aslan

ABSTRACT

Development of a game in the form of software for game-based learning poses significant technical challenges for educators, researchers, game designers, and software engineers. The game development consists of a set of complex processes requiring multi-faceted knowledge in multiple disciplines such as digital graphic design, education, gaming, instructional design, modeling and simulation, psychology, software engineering, visual arts, and the learning subject area. Planning and managing such a complex multidisciplinary development project require unifying methodologies for development and software quality evaluation and should not be performed in an *ad hoc* manner. This dissertation presents such methodologies named: GAMED (diGital educAtional gaMe dEvelopment methoDology) and IDEALLY (dIgital eDucational gamE softwAre quaLity evaLuation methodologY).

GAMED consists of a body of methods, rules, and postulates and is embedded within a digital educational game life cycle. The life cycle describes a framework for organization of the phases, processes, work products, quality assurance activities, and project management activities required to develop, use, maintain, and evolve a digital educational game from birth to retirement. GAMED provides a modular structured approach for overcoming the development complexity and guides the developers throughout the entire life cycle.

IDEALLY provides a hierarchy of 111 indicators consisting of 21 branch and 90 leaf indicators in the form of an acyclic graph for the measurement and evaluation of digital educational game software quality.

We developed the GAMED and IDEALLY methodologies based on the experiences and knowledge we have gained in creating and publishing four digital educational games that run on the iOS (iPad, iPhone, and iPod touch) mobile devices: CandyFactory, CandySpan, CandyDepot, and CandyBot. The two methodologies provide a quality-centered structured approach for development of digital educational games and are essential for accomplishing demanding goals of game-based learning.

Moreover, classifications provided in the literature are inadequate for the game designers, engineers and practitioners. To that end, we present a taxonomy of games that focuses on the characterization of games.

Digital Educational Games: Methodologies for Development and Software Quality

by

Serdar Aslan

GENERAL AUDIENCE ABSTRACT

The game development consists of a set of complex processes requiring knowledge in many disciplines such as digital graphic design, education, gaming, instructional design, modeling and simulation, psychology, software engineering, visual arts, and the learning subject area. Planning and managing such projects require unifying methodologies for development and software quality evaluation. This dissertation presents such methodologies named: GAMED (diGital educAtional gaMe dEvelopment methoDology) and IDEALLY (dIgital eDucational gamE softwAre quaLity evaLuation methodologY).

GAMED consists of a body of methods, rules, and postulates and is embedded within a digital educational game life cycle. It provides a modular structured approach for overcoming the development complexity and guides the developers throughout the entire life cycle.

IDEALLY provides a hierarchy of 111 indicators consisting of 21 branches and 90 leaf indicators in the form of a graph, in which there is no cycle or closed path, for the measurement and evaluation of digital educational game software quality.

Moreover, classifications provided in the literature are poorly defined for the game designers, engineers, and practitioners. To that end, we present a taxonomy of games that focuses on the characterization of games.

DEDICATION

To my dear wife and parents

ACKNOWLEDGMENTS

I owe my deepest gratitude to my advisor Dr. Osman Balci for his support and guidance not only in this research, but also in many aspects of life since I started studying at Virginia Tech. I have learned a lot from him and I cannot thank him enough.

I would like to thank my committee members Dr. James D. Arthur, Dr. Reza Barkhi, Dr. Mat Grove, Dr. Anderson Norton, and Dr. Eli Tilevich. Moreover, I express my gratitude to Dr. Anderson Norton for helping us generate different game ideas and assisting us significantly throughout the development of the games. I also thank my fellow graduate students in the Mobile Software Engineering Lab.

Finally, my gratitude goes to my wife and my parents, for always being there when I need them and supporting me, no matter what.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: DEFINITION OF THE PROBLEM AND OBJECTIVES

1.1 Statement of the Problem

Science, Technology, Engineering and Mathematics (STEM) is one of the pivotal keys to the future of innovation, economic competitiveness and leadership in the United States. Studies have shown that STEM education has a direct relation to a nation's future in both positive and negative ways. It effects a nation's future not only in the areas of changing economy, leadership and competitiveness, but also global awareness, national security interests, unity and cohesion of the country [The National Commission on Mathematics and Science 2000; Langdon et al. 2011]. The US is losing their global economic competitiveness as developing countries are improving their STEM education and technology-driven economy [Tapping America's Potential (TAP) Coalition 2008].

A sustained growth in a technology-driven economy requires high quality STEM workers [Langdon et al. 2011]. To produce highly educated STEM workers, we need to provide students an education system that has high quality standards. However, US education system has not reached those quality standards yet. Despite the fact that there have been some improvements in the STEM education system, the proficiency levels for math and science are still low nation-wide [National Science Foundation 2013].

One of the biggest reasons of poor performances is that most of the classroom learning for students is superficial [The National Commission on Mathematics and Science 2000]. Instead of understanding the underlying reasoning for topics, students tend to memorize formulas [The National Commission on Mathematics and Science 2000]. This lack of knowledge behind math and science later causes serious problems, such as not being able to understand more abstract topics in higher education. In turn, this leads to students dropping out of college or changing degrees from STEM to non-STEM. Moreover, students mostly complain about the lack of interest for their classes because of the boring teaching style. They think that the course topics are taught in a way that that does not motivate them to learn the necessary subjects [Bridgeland et al. 2006; Prensky 2007].

When one navigates through the literature related to learning, one of the pivotal options for fostering the learning process effectively are digital educational games. According to the Prensky [2007] and Gee [2003], students understand and execute better when they visualize concepts. Similarly, many digital games provide an environment where kids can easily capture the rich graph content, which enables them to easily think, understand and execute effectively [Gee 2003; Ke 2009].

Digital educational game development is a complex process requiring an expertise on many areas, such as digital graphic design, education, gaming, learning sciences, modeling and simulation, psychology, software engineering, and visual arts. Although the current literature presents some guidance, they fail to provide a comprehensive life cycle. To address this problem, we propose a solution that addresses many facets of the problem in a comprehensive manner.

1.2 Statement of Objectives

The overall goal of this dissertation consists of the following three main objectives:

1. Create a methodology for the development of digital educational games:
   a. Identify a life cycle representing a framework for organization of the processes, work products, quality assurance activities, and project management activities required to develop, use, maintain, and evolve a game software application from birth to retirement.
   b. Modularize and structure game software application development.
   c. Provide guidance to game designers, software engineers, and project managers.
   d. Enable to view game development from four perspectives (or Ps): Process, Product, People, and Project.
   e. Specify the work products to be created under the designated processes together with the integrated quality assurance, verification, and validation activities.
   f. Advocate a quality assurance, verification, validation, and testing strategy from four perspectives (or Ps): Process, Product, People, and Project.
   g. Use the proposed framework in digital educational game development
2. Develop a methodology for software quality evaluation of digital educational games:
   a. Enable measurement and evaluation of qualitative and quantitative characteristics of a game.
   b. Enable the use of a hierarchy of quality indicators forming an acyclic graph.
   c. Provide relative criticality weighting of the child indicators influencing the same parent using the Analytic Hierarchy Process (AHP).
   d. Enable the use of Subject Matter Experts (SMEs) in evaluating the leaf indicators.
   e. Provide relative criticality weighting of the SMEs assigned to evaluate the same leaf indicator using AHP.
   f. Develop a repository of reusable indicators for quality assessment of digital educational games.
3. Create a taxonomy of games:
   a. Identify characteristics of games.
   b. Classify games based on their characteristics.
   c. Propose terminology for game-based learning.
   d. Provide guidance to digital educational game designers and developers.

1.3 Significance of the Research

This research addresses a vital subject matter that has been a center of discussion and research in the STEM education. Recently the STEM education has been a focal goal of many agencies, institutions, and governments. Software engineering research has always been one of the most effective research areas to provide solutions in all aspects of life. Hence, successful research in this area will bring a needed solution in STEM education to foster a well-educated society.

This research focuses on one of the alternative ways of fostering STEM education by way of exploring digital games domain. Digital Game-Based Learning (DGBL) can provide engaging and motivating environments for students. Moreover, DGBL can help students in many ways. It can provide an environment for students to practice and gain necessary skills needed. The content of the game can overlap with the school courses, which can foster students' learning in innovative ways. More specifically, games can prepare students for future jobs by teaching collaboration, problem solving, and communication skills.

One of the focuses of our research is to develop digital educational games in order to teach the desired concepts/topics to players. We present a systematic look for the problem domain considering the software from its' birth to retirement. To that end, we investigated the best ways to develop, document, and evaluate a digital educational game development life cycle. This life cycle represents a framework for the organization of processes, work products, quality assurance activities, and project management activities required to develop, use, maintain, and evolve a game software application from birth to retirement. The goal of the proposed life cycle is to be adapted by any digital educational game regardless of its size.

This work undertakes a significant research in evaluation of digital educational games software. Throughout the whole life cycle, the quality assessment is the main pillar of any type of software, which affects the level of success, and also poses significant challenges for managers, engineers, and practitioners. To that end, we intend to develop a game software quality assessment methodology to enable measurement and evaluation of qualitative and quantitative characteristics of digital educational games. We enable the use of hierarchy of quality indicators and Subject Matter Experts (SMEs). The goal of the indicator creation is to develop a repository of reusable indicators for the game software quality assessment.

Another aspect of significance of this research is in the area of taxonomy creation. Classifications provided in the literature are mostly done based on the old movie-genre approach. However, they are inadequate for the game designers, engineers and practitioners. The game industry constantly introduces new game types by combining multiple games. Moreover, the current terminologies in game domain involves many inappropriate terms for the game community. To that end, we present a taxonomy of games that focuses on the characterization of games and provides an appropriate classification based on identified characteristics.

In summary, our research leverages the best practices of software engineering to address a nation-wide problem and many of its related issues. The solution we present has a significant impact on a wide range of institutions from primary education to higher education. Our contribution addresses a class of problems related to digital educational games development, evaluation and taxonomy.

CHAPTER 2: BACKGROUND

In this section, we present all related areas that influence our research. Since the problem, STEM education, we are addressing is multi-faceted, our literature review spans different fields. We identify multiple areas that need to be studied in order to address this problem. These areas include game-based learning, software and system architecture, architecting and development process, quality assessment and quality evaluation of software, and classification in digital games.

This chapter is organized as follows. Section 2.1 introduces the digital games. Section 2.2 describes the importance and benefits of digital game based learning. Section 2.3 describes the current challenges in game development. Section 2.4 describes the quality assessment of digital educational games and section 2.5 talks about classification of games.

2.1 Digital Games

Video games have been around more than 40 years. The term "video game" refers to a digital game that requires an interaction between a human and some type of hardware, which provide visual feedback on a video device. According to Salen and Zimmerman [2004], a video game is "A system which players engage in artificial conflict, defined by rules, that results in a quantifiable outcome." Although the word "video" refers to Cathode Ray Tube (CRT) monitors, it is still improperly used to refer to digital games that have nothing to do with CRT.

Digital devices and hardware that players interact with are called gaming platforms. Platforms can range from big mainframe computers to small hand-held computers or smartphones. A gaming platform consists of a display to present output, a controller to take user input and a main gaming unit to run a game software. For instance, Xbox, Play Station, Nintendo are some of the most popular gaming platforms [Grand and Yarusso 2004].

Digital gaming started with the Pong [The International Arcade Museum 1995] in 1970 and it has been growing exponentially. The market worth is billions of dollars, and it is forecasted to reach $78 billion by 2017 [DFC Intelligence 2013]. Digital games present varieties and they can consist of simple or more complicated features. According to the Deubel [2006], games can be roughly categorized as "action, adventure, fighting, puzzle, role-playing, sports, and strategy". Game developers combine features from different game categories to come up with more interesting and feature rich games, which results in extension of the categories given above.

2.1.1 Digital Game Genres

Digital game genres are used to classify games based on their gameplay and interaction. The number of game genres has increased in recent years, since the popularity of more complex and diverse games. Following is a listing of commonly used digital game genres. [Sellers 2005; Pinelle et al. 2008]

- *Action*
- *Adventure*
- *Role-Playing*
- *Puzzle*
- *Sports*
- *Strategy*

2.2 Digital Game-Based Learning

DGBL is a learning approach, which incorporates the use of digital games for exploration or practice of an educational content that integrates learning principles while engaging students into the game environment [Coffey 2009].

Since digital games enable students to gain skills needed in an information-based culture and to learn innovatively, the interest in learning games has considerably grown within the last decade. We hope that today's children have the same interest in school as they do in video games. They are competitive, inquisitive, motivated, persistent and seek out new information when it comes to playing video games. [Aslan 2011]

2.2.1 The Importance of DGBL

Our world is getting more global than ever, which is also known as becoming a "flat world". According to the Reimers [2008] this issue raises a flag on how to prepare our students in the 21st century for their future. To compete globally, we should be able to have the necessary knowledge and be able to apply them. U.S. Department of Education reports that U.S. needs to have an education plan to be followed [U.S. Department of Education 2010]. In this report, they point out that there is a need for conducting researches in following areas such as simulations, collaboration environments, virtual worlds, and games. More importantly, they indicate that DGBL could provide more engaging and motivating environments for students.

Similarly, the Horizon Report [Johnson et al. 2011], which is published by a non-profit consortium, chose the DGBL as one of the only six topics that has significant value as a technology and research topics. Moreover, they point out that the DGBL will gain more popularity in upcoming years and it will help students in many ways. It can provide an environment for students to practice and gain necessary skills needed. The content of the game can overlap with the school courses, which can foster students learning in an innovative way. More specifically, they state that games will prepare our students for the future jobs by way of teaching collaboration, problem solving, and communication skills.

In addition, recent studies have shown that there is a significant interest in DGBL. Hwang and Wu [2010] state that during the past ten years (2001-2010) the number of published papers has significantly increased. More specifically, after 2006, the published research papers in selected journals are four times more than previous years.

Today's children have more tendencies to play digital games than any other activities. When it comes to playing games they are competitive, persistent, determined, and always look for new elements/features in the game environment to accomplish their goal. Almost any child plays digital games and spends time on that activity. According to the Lenhart [2008] 97% of US teens between the ages of 12-17 play some kind of digital games. Moreover, by the age of 21, an average kid spends about 10,000 hours for digital game playing [Prensky 2003]. Hence, recent studies have shown that researchers look for possible ways to foster the same engagement towards learning progress.

Bridgeland et al. [2006] report that some of the reasons that students drop out high schools are (1) almost half of the students think the classes were not interesting, (2) they are no motivated to work, (3) challenges in academia, (4) having poor academic background. These key findings show that the system that we use currently either needs to be changed or updated.

Prensky [2007] states that students have been affected by the rapid technological development. Many students grow up with smartphones, laptops, and other digital devices. Digital devices have potential to capture kids' attention very easily because it provides rich, animated, interactive contents including, games, music, and apps.

Gee [2003] points out that students understand and execute better when they can visualize a concept. Digital games provide an environment where rich graphical context is provided to enable kids to easily think, understand, and execute. Hence, digital games can be an alternative teaching way.

According to the Gee [2007] some of the learning principles are directly related with the digital games, these are:

- *Interaction*
- *Production*
- *Risk Taking*

2.2.2 Benefits of DGBL

According to the Gee [2009], digital games provide continuous learning through engagement and entertainment. For example, Neulight et al. [2007] experimented with two sixth-grade classes. The users played an infectious disease game, which was designed and developed, specifically for students. The students took pre-test and post-test and the results showed that twice as many students were twice as many students were able to provide reasonable answers for infectious diseases, which obviously proved the effectiveness of the DGBL.

2.2.2.1 General Skills

Joyce et al. [2009], who studied DGBL and reviewed 500 teachers across Europe, reports that an increased motivation in a classroom improves several key areas such as personnel skills (initiative persistence), spatial and motor skills (coordination, speed of reflex), social skills (teamwork coordination), and intellectual skills (problem solving). Similarly, McFarlane et al. [2002] state the similar key areas that DGBL can improve such as strategic thinking, planning, communication, application of numbers, negotiating skills, group decision-making, and data-handling.

2.2.2.2 Freedom to Fail

There are several reasons that cause students to fail in school. Students' lack of interest for their class and the boring traditional teaching style lead students to either dropout or change their degrees [Bridgeland et al. 2006]. Moreover, Prensky [2007] states that students are not convinced that the courses motivate them to learn the necessary subjects [Prensky 2007].

Klopfer et al. [2009] states that play can effectively incorporate with learning process. They emphasize that play gives kids the freedom to fail and freedom to experiment. Digital games enable kids to make mistakes and learn from them. They enable students to try continuously and experiment in different ways. Allowing students to make mistakes removes the fear of failure. For example, in traditional classroom students might be discouraged from giving wrong answers. However, play and games remove this issue by way of allowing students to make mistakes [Groff et al. 2010].

2.2.2.3 Clear Goals and Immediate Feedback

Clear goals enable players to focus on a specific target and decrease the level of distraction of trying to accomplish many tasks at once. All the rules, obstacles, game elements that are provided by the games are oriented towards a specific target or targets [Dickey 2005].

Properly presenting an information increases the engagement of the user. Rigby and Ryan [2007] show the importance and the benefit of immediate feedback with this quote “As a best practice we consistently find that providing feedback that is (1) clear and unobtrusive, and (2) immediately responsive to the players’ actions contributes to an experience of mastery and satisfaction particularly when engaging game challenges”.

2.2.2.4 Active learning

In traditional education system, most of the students are inactive learners because the interaction between students and the teacher is really low. While the teacher is explaining a concept, there are no motivations for students to follow the teacher, which degrades the students’ learning curve. Similarly, according to the Dalton [2000] 56% of students who took online classes went through the same experience where they did not feel like an active learner. However, in digital games the user has active role that makes the decision for any action that take place in a game. Hence, games provide high interaction, which can be used for students to take the active role during the learning process [Klopfer et al. 2009].

2.2.2.5 Challenge

Challenge, a task that tests someone’s ability, motivates players when it is completed successfully. It also enables players to think differently. According to Gee [2003], a good game should have many challenges with an appropriate level of difficulty, which can be achievable by user with a little extra effort. In a typical game, challenges increase with respect to the game levels, where in the low levels challenges are easy, and in the high levels, they get more difficult.

According to the Dickey [2005], challenge pushes players to think and come up with new strategies. When players fail to successfully complete a challenge or pass a level, challenges push them to think and analyze their process to see where they can make some improvements. Hence, the thinking process to reach the desired goal enables players to get more motivated and engaged. Challenge needs to be designed carefully so that they can match with the player’s skill [Kiili 2005a]. A difficult challenge could frustrate the user, which causes a decrease in level of engagement, and motivation.

2.2.2.6 Engagement

Several studies have shown that the traditional educational system is boring due to lack of course interest, passive learning, and students’ excessive interest on to the digital world [Bridgeland et al. 2006; Prensky 2007]. According to the MIT professor Dr. Semour, students find classes boring by the way they are taught currently [Prensky 2007].

Games provide more engaging environments and almost 97% of the teenagers in the US play digital games and spend more than 10,000 hours by the age of 21. Games with a robust narrative story motivates students to forget how the time passes due to high level of engagement [Dickey 2005; Rupp et al. 2010].

Several studies have shown that digital games increase the engagement. Dickey [2005] and Gee [2009] state that games that have clear goals engage and motivate players. Joyce et al. [2009] reviewed 500 teachers in Europe and reported that DGBL increases the motivation and engagement in their classes. Similarly, Groff et al. [2010] used console games in their classes and their findings show that DGBL has made the same positive impact on motivation and engagement.

2.2.2.7 Formative Assessment

The purpose of assessment in a game environment is to provide an evaluation of players' knowledge and abilities. Games by their nature provide an assessment. According to Ash [2011], any challenge that is given to the player is a process that evaluates the players' knowledge or personal skills. In addition, recent studies have shown that there is an increasing interest across the world to explore the use of digital games in learning and assessment [Shaffer 2006; Shute et al. 2009; Rupp 2010].

Digital games enable researchers to collect rich data about the players' activity. Recently game developers have started to use advanced game analytics method to capture data. According to IBM [2012], many of the digital games provide metrics such as daily active users, monthly active users, engagement, average revenue per user, and lifetime value. Similarly, Koenig et al. [2010] develop a framework to evaluate students' performance on their knowledge and skills. The developed framework represents players' real-time activities, which later used for evaluation.

Shute [2009] demonstrates a game called *Oblivion* to present how assessment can be done in a digital game. In their case study, a competency model was used and it was divided into efficiency and novelty indicators. Each of these elements is connected with an action in the game and each action has an effect on the chosen model. When players execute any of the defined actions, the game records the data in a log file and update the model, which can be directly measured.

2.3 Game Development

A digital educational game development requires a unifying process where all the learning elements such as instructional materials, activities, related resources, and required evaluation system needs to be analyzed, planned and implemented in a very careful manner [Smith and Ragan 2005].

2.3.1 Background of Digital Game Development

Game development is the process of creating a digital game. The game development team can consist of one developer to dozens developers depending on the project. Although in the past digital games were only available for mainstream computers, today there is almost a digital game for any platform, ranging from TVs to smart-phones. Most of the games are produced to make profits and the production is mainly supported by publishers. Yet, this has been changed recently due to the rise of smart phones and small-scale game development. Recent developments in the smartphone platforms enable many indie game developers to produce digital games in a short development time [Bethke 2003; Moore and Novak 2010].

The first commercial digital game development started in the 1970s with the digital game consoles and personal computers [The International Arcade Museum 1995]. Digital games became more available and popular with the recent advancements in the hardware technologies. Having smart phones with dual core processors, or a game console that has tremendous computing power have a huge impact on the development of digital game industry [DFC Intelligence 2013].

The game industry has been rising; however being a successful game developer is not getting easier [Bethke 2003]. Constantly, the game publishers need to revise and update their game or even release a new one to keep their customers happy and satisfied. Almost every quarter independent developers start companies to produce a game to bring them the title of the "killer game". However, most of them shut down before they even publish their first game [Moore and Novak 2010].

2.3.2 Challenges of Developing Digital Games

According to Bethke [2003], “game development is the software development process by which a video game is produced”. A video game at the core is a software, which integrates art, audio, and the gameplay in a common ground.

According to Charette [2005], most of the software projects have common problems that can be grouped as scheduling, budget, quality, management and business related problems. These problems overlap with the game projects. Flood [2003] states that game projects have issues such as: project completion was behind the scheduled time, the final product had many flaws, the provided functionalities were different from the ones expected, or there was tremendous pressure on game developers to complete the game. According to Bethke [2003], the game industry lacks adopted methodologies, and generally, they adopt poor methodologies, which leads their project to be delivered behind schedule, have inefficient use of resources, and published games with many defects.

Kanode and Haddad [2009] present the challenges of digital game development and possible solutions from a software engineering perspective. They point out that the digital game development differs from the regular software development with the preproduction stage where all the requirements from the customer is gathered and included in the game design document. More importantly, they indicate that a clear and carefully processed preproduction helps designer and developers to find the “fun” element during the development process. They emphasize that a project manager should carefully review the organization and the game requirements so that the best software engineering methods can be chosen. For example, the spiral process might work efficiently and increase the productivity for large-scale projects. For preproduction stage, agile methods can be used to increase the speed of prototyping.

- *Project Management:* Project management is a crucial stage in any development process. The project manager is the person who is responsible for planning, execution, managing the people, resources, and scope of the project. An effective manager and management can increase the productivity of development team. In many cases, the manager provides the interaction between different teams; s/he assigns the right person or people to the right tasks and makes the decisions for the project timeline. Any problems in these processes effect the project development in a negative way [Callele et al. 2005]. According to Callele et al. [2005], most of the internal problems in a development team or the game companies are related with project management issues.

- *Team Management:* Lanubile et al. [2003] discusses the challenges of developing software between different teams that are geographically dispersed. In such situations, the main challenge comes from the informal communication, different time zones, and different team cultures. To overcome such situations, they suggest iterative development methods, which can increase the trust between the teams and provide better synchronization.

- *Requirements Engineering:* Digital game development differs from the traditional software development by preproduction stage where the requirements are gathered from potential users and the design document is created. Callele et al. [2005] state that transitioning from the preproduction to production stage is the most crucial part in digital game development since the user requirements need to be converted to the formal requirements. They also point out that finding the common ground for the multidisciplinary teams, such as creation of the common language for effective communication and interaction, is a challenge.

- *Scope:* In games, scope creep is caused by failure to define projects’ scope clearly. Most of the developer teams welcome many features without having the required and available resources, which delays the original release date [Flynt and Salen 2004]. For instance, Bioshock game, took three years to develop which was originally promised to be delivered in two years [Kanode and Haddad 2009].

- *Crunch Time:* Crunch time is one of the inevitable facts of the gaming industry. Many games, require "crunch time" at the final week of the deadlines caused by extra added features or unexpected issues/bugs. During crunch time, game developers work 12 hours 6-7 days a week [Kanode and Haddad 2009; Petrillo et al. 2008].

- *Scheduling Problems:* Scheduling is a common problem in game development. Many teams can't publish their game on the scheduled day and some of the known issues are as follows.

    a. Lack of careful analysis of the scope, game story, gameplay, and game features [Flynt and Salen 2004].

    b. Waiting for other developers to finish their work [Flynt and Salen 2004]. In many cases, game development requires multiple assets to be combined on top of each other, such as graphics or sound integration. In many cases, teams are required to deliver their work to other team members so that other developers can implement the rest of the game features. However, this often fails because of scheduling problems.

    c. Third, having many meetings and other activities that interrupts developers' flow [Flynt and Salen 2004; Kanode and Haddad 2009].

- *Technological Problems:* Each gaming platform comes with different hardware specification and limitations. For example, while developers can keep track of user's location in a smartphone through GPS, this can be very challenging in a game console. Similarly, the computing power shows variety between different platforms. In some cases, the developers are forced to create their games for the least powerful platforms so that it can be downloaded by more people.

2.3.3 Game Development Methodologies

Game development is a process that starts from an idea or concept. Game designer usually produces initial game proposal document which contains the concept, gameplay, feature list, story, and target audiences. Companies have different formal procedures and philosophies for game design and development processes. There is no standardized development method; however, commonalities exist. Some of the game development methodologies are discussed in the following section.

2.3.3.1 Digital Educational Game Development Methodologies

There are many shortcomings in the development methodology for DGBL and we can group them as follows:

- Lack of clear, easy to follow, step by step instruction.
- Linear processes, no iterative development.
- Missing design and development stages.

A methodology should include instructional design and game development processes properly without any missing guidelines to increase the effectiveness of digital educational games.

Literature presents only a few digital game development methodologies for educational games. Kirkley et al. [2005] present the Simulation-game Instructional Systems Design (SG-ISD) model, which integrates the instructional systems development process with the game development processes. Their model is being used to create a prototype authoring system. It enables developers to analyze/investigate all game elements and design issues while enabling supported game engines in the game development. The model consists of analysis, design, development, implementation, and summative evaluation. Their research more focuses on the design of the educational games. Although the presented model provides sub-steps and corresponding descriptions, one cannot find any explanation for the implementation and development processes.

Hays [2005] discusses the effectiveness of instructional games and he proposes a systematic approach to instructional game design and use. The proposed methodology consists of understanding the instructional environment (develop problem statement, develop instructional objectives, select game strategy), developing the game (develop game model, develop students' role in game, develop rules), and implementing and evaluating the game (develop supporting game resources, evaluate game compared to alternate instruction, modify game based on evaluation and results). The proposed methodology follows the traditional waterfall model without any improvement.

Ho et al. [2006] present a case study of game design for e-learning. Their study focuses on the development of Role-Playing Game (RPG). The game integrates the problem based learning with the RPG, which is expected to increase the student's problem solving skills and develop a learning strategy. During the game development and the implementation process, they follow some of the traditional game development stages such as analysis, deciding on story and the demonstration of problem models; however, they do not provide any information about the development, implementation, and the quality assurance steps. Hence, the used development methodology is vague and it is not well documented for us to analyze and compare with other methodologies.

Watson [2007] presents a methodology called Games for Activating Thematic Engagement (GATE), which is developed for designing and implementing instructional digital games. Although the proposed methodology provides well-defined steps for game design, it doesn't provide sufficient detail for game development and implementation stages.

Flanagan and Nissenbaum [2007] present a methodology called Values in Play (VAP), which is developed to analyze the human themes in the game design and improve the integration of game elements in the design process. Their methodology is proposed as supplementary method to the existing methodologies. It includes three activities, discovery, translation, and verification. The proposed methodology can be used for digital entertainment games; however, it is not sufficient for educational games. It lacks instructional elements such as instructional objectives or learning outcomes. Thus, it is incomplete to provide clear guidelines for digital educational games.

Han and Zhang [2008] propose a methodology called Quasi Game Based Learning (Quasi-GBL). The proposed model provides instructional methods that integrates game elements with role-play in collaborative learning. Game elements such as goal, rule, competition, challenge, fantasy, safety, and entertainment are embedded into the roleplaying. They are presented like real problems, scores, puzzle, awards and replays in the game environment. Although the proposed methodology provides some of the common educational game development processes, it lacks of detailed information for each.

Zin et al. [2009] present Digital Game-Based Learning-Instructional Design Model (DGBL-ID). The proposed model consists of five different phases, which are analysis, design, development, quality assurance, implementation and evaluation. Although the proposed model seems promising, after the design phase the methodology lacks explanation for the rest of the phases. Only some steps and guidelines are provided. For example, the methodology does not provide any information about which methodologies to be used in the development or how to evaluate a game. Therefore, the methodology does not provide information and clear explanation beyond the design of a game.

Amory [2001] presents a Game Object Model (GOM), which makes a connection between pedagogical learning and game elements. The GOM is based on Object-Oriented Programming paradigm to facilitate complicated game design and development. In this model, games are represented by number of different components, which are described through abstract and concrete interfaces. The GOM consists of problem Space (motor, memory, literacy, and communication), elements Space (Actors Space), visualization space (elements space, problem space), game space (visualization space). Although the GOM has been successful, Amory [2006] updated the model to the GOM II. The new version of GOM II improves the problem space by integrating social space and it enables GOM to evaluate games.

Tang and Hanneghan [2010] studied the Model-Driven Serious Games Development Framework, which enables automatic generation of software artifacts through modeling to help non-technical domain experts. In addition, this model serves as a reference model for developers to create their own model-driven framework. The framework consists of user interface, models, model-driven engineering (MDE) tools, components library, code templates, artifacts, technology platform, operating platform, and software.

2.4 Quality Assessment / Quality Evaluation of Digital Games for Educational Purposes

To put it concisely, evaluation is the process by which the qualitative or quantitative value of an entity is determined. In the context of learning technology, these evaluations usually concern the educational value of innovation, or the pragmatics of introducing novel teaching techniques and resources [Oliver 2000].

Over the last decade, the use of digital educational games has been increasing. Although most of the teachers in schools still use traditional teaching methods, digital games have been adopted widely as supplementary classroom activities [Freitas and Oliver 2006; Prensky 2007].

Employment of digital games in classrooms highlights the importance of design and integration of pedagogical elements in order to develop effective educational digital games [Federoff 2002; Grammenos 2008; Mazzone 2007; Prensky 2001; Wall and Ahmet 2008]. This is important to make sure that both study and play can be integrated successfully to achieve predefined goals. More importantly, implementing formal evaluation procedures can save time and money [Federoff 2002].

As a generic approach, Kirkpatrick [1994] presents an evaluation model, which is used for evaluation of learning in organizations. The proposed model has been used widely by many researchers and it includes four levels as follows:

- *Level 1: reaction,* learners' feelings about the training
- Level 2: learning, results in learning
- *Level 3: behavior,* improvement in behaviors or capabilities
- *Level 4: results,* the effect on the business or environment

Although there has been an increasing interest in DGBL, only few attempts have been made to improve the current evaluation frameworks [Kirriemuir and McFarlane 2004]. The lack of useful frameworks has produced an obstacle for uptake of games [De Freitas 2004].

Squires and McDougall [1994] present a model, which considers the interactions between teachers, students and educational games. Similarly, CIAO!, a framework that is developed by Jones et al. [1996] considers the context, interaction between learners and technology, attitudes and outcomes. In addition, it uses interviews, observations, document analyses and surveys for evaluation. Ehrmann [1999] presents a model called Flashlight, which examines the relationship between a technology, the activity used for educational outcomes, and surveys. Moreover, Oliver [2000] discusses the TILT, CIAO! and Flashlight frameworks all together to evaluate technology in teaching.

However, all of these frameworks were designed to lay emphasis on technology, but not on game development in a pedagogical context [Woods 2004].

2.4.1 Evaluation of Educational Games

Effective evaluation of digital educational games is necessary to ensure that they have desired qualities. Different types of evaluation methodologies have been used in the evaluation of digital educational games and they mainly focus on the summative evaluation, which evaluates the predefined goals regarding the implementation of the developed games [Bernhaupt, et al. 2008; Marks and Wunsche 2007; Mueller and Gibbs 2007; Wang 2008].

In many studies, the evaluation only takes place after the game development [Ebner and Holzinger 2007; Owston et al. 2009; Robertson and Howells 2008; Robertson and Miller 2009]. For example, Mazzone et al. [2007] present work based on an evaluation study of a tangible game prototyping for children. In their study, the students were divided into two groups: inquiry, which focused on users' information and opinions by questions after their interaction with the game; and observational methods, carried out during the user performance. Two researchers conducted the study, one interacting with learners, and the other focusing on the observation like taking notes and video recording. During the study, learners are asked to think about how they would like to update the game.

In spite of the variety of evaluation frameworks, most of the proposed frameworks are dependent on the effectiveness of the generated questionnaire. For instance, Mueller and Gibbs [2007] use a mixed methodological approach for the evaluation. They use existing questionnaire surveys to collect quantitative and qualitative data, and observations.

Wall and Ahmet [2008] conducted a study on a simulation game called MERIT to explore the issues found on integration processes. As part of their evaluation, they conducted a survey with the participants and each participant was asked to evaluate the integration of the game. Some of the important questions were as follows: "Did the game assist in developing a greater understanding of the problems and decisions that are involved in running a modern construction company? Did the simulation game improve team working, communication and IT skills?" [Wall and Ahmet 2008].

Similarly, Grammenos [2008] presents a framework, which focuses on the design and an evaluation of games. The proposed framework, Game Over, is developed to provide and teach game accessibility guidelines. During the development process, evaluation took place under experts' leads. The first evaluation took place after the initial mockups of the game. The evaluation was based on the gameplay, game presentation, and levels. For each section, experts provided their feedback. During the game development, each time a new level was created, experts evaluated the game to detect flaws/bugs. After the game was finalized, an online questionnaire was prepared and provided to the players via email. Some of the prepared questions are as follows: "How familiar were you with the accessibility guidelines presented in "Game Over!" before playing the game? How much do you think that playing "Game Over!" helped you familiarize with game accessibility guidelines and their application? How much fun was playing "Game Over!"?" [Grammenos 2008].

As discussed above, most of the evaluation of digital educational games uses questionnaires [Annetta et al. 2009; Lacassa et al. 2008; Papastergiou 2009], interviews, log files and observations [Kiili 2005]. Other than those listed above, there were few attempts in developing frameworks to evaluate educational games [Freitas and Oliver 2006; Mohamed and Jaafar 2010].

Freitas and Oliver [2006] present a model, which addresses the evaluation framework shortcomings by introducing a four-dimensional framework as shown in Figure 1. The framework could be used for evaluation of games as well as for the selection of games. The first dimension focuses on the context where learning takes place. The second dimension focuses on attributes of the particular learner, which may include the age, and the level of the group. The third level focuses on the mode of the presentation, the interactivity, the levels of immersion, and the fidelity used in the game or simulation. The fourth dimension focuses on the process of learning both during the course of formal curricula based learning time and during informal learning.

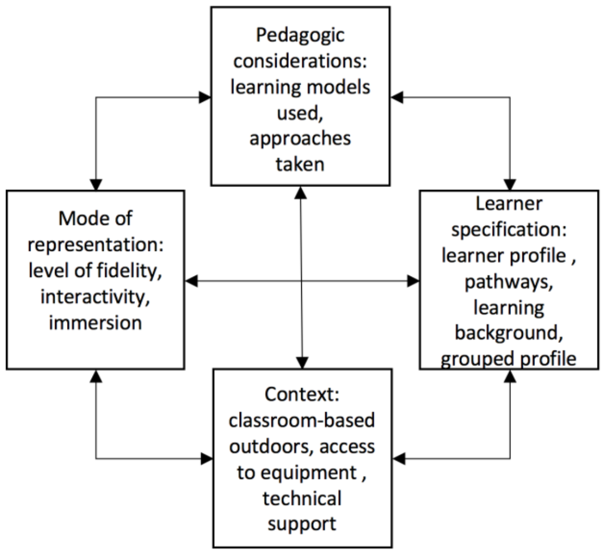

Pedagogic considerations: learning models used, approaches taken
Mode of representation: level of fidelity, interactivity, immersion
Learner specification: learner profile , pathways, learning background, grouped profile
Context: classroom-based outdoors, access to equipment , technical support

Figure 1. A framework for evaluating games and simulation-based education [Freitas and Oliver 2006]

Gunter et al. [2008] present a design and evaluation method called RETAIN (Relevance, Embedding, Transfer, Adaptation, Immersion, Naturalization). This model is based on instructional design principles. The model provides guidelines for educators and game designers. The model can evaluate the effectiveness of the current developed games and also can enable educators to choose games for their classes. The model consists of six different parts, Relevance, Embedding, Transfer, Adaptation, Immersion, and Naturalization. In each part, the model provides a leveling system (level1), (level2), (level3), which is used for evaluation.

- *Relevance:* The materials that are being used for teaching any concepts/contexts to the learners need to be relevant to the learners. They should match with learners’ needs. The context of the game should be familiar to the learners, which can increase their motivation, interest and overall emotional feelings.

- *Embedding:* The design of an educational game requires a careful integration process. One cannot simply embed any content to the game world. The content and the story of the game should be carefully embedded to the game world and the pedagogical aspect of the game should be designed in a way that the complexity increases iteratively starting from the simple to the complex levels. More importantly, one should make sure that the integration of those items is effective enough to provide satisfaction to the learners.

- *Transfer:* The game should enable learners to transfer the targeted learning outcomes from the game environment to the real world.

- *Adaptation:* Games should provide an environment where the learners can continuously learn and adapt. Games should focus on balancing learning and adaptation.

- *Immersion:* Games should provide an environment where objectives are clearly defined, and challenges show appropriate difficulties.

- *Naturalization:* Games should be designed in a way that they can provide motivating, engaging, and immersing environment for learners to play the game.

2.4.2 Heuristic Evaluation

Research by Mohammed and Jaafar [2010] indicates that Heuristic Evaluation (HE) has huge potential to be used in evaluation of digital games. HE has been used by Malone since 1982. Later, Clanton [1998] developed a set of game design principles based on HE. A more formal usability evaluation process was laid out by Federoff [2002].

Desurvire et al. [2004] present a Heuristic for evaluating playability. Similarly, Hannu and Elina [2006] develop playability heuristic for mobile games. Song and Lee [2007] compile key factors of heuristic evaluation for game design and categorize them into four areas: interface, play, narrative and mechanic.

Pinelle and Wong [2008] present a heuristic evaluation for digital game design, which adapts usability inspection for games. Some of the heuristics are as follows: “Provide consistent responses to the users’ actions. Allow users to customize video and audio settings, difficulty, and game speed. Provide predictable and reasonable behavior from computer-controlled units. Provide unobstructed views that are appropriate for the users’ current actions”.

2.4.3 Challenges in Evaluation

There are many issues related to the evaluation of games. According to the Mohamed and Jaafar [2010], there are three different challenges in the evaluation of digital games such as the evaluation criteria, evaluators, and the evaluation process. Evaluation criteria focuses on the required elements, which will be evaluated. Evaluators focus on the people (experts) that are required for evaluation. The evaluation process focuses on the process that is being used during the evaluation. The evaluation process can be summative or formative.

Yet, Clanton [1998] suggests that games should be examined by their interface, mechanics and game play.

- *Game interface:* Game interface is the hardware and software that users interact with.

- *Game mechanics:* Game mechanics are the physics and the rules of the game.

- *Game play:* Game play is the main activity pattern that is defined through the game rules, connection between player and the game, and the challenges.

Collings [1997] presents that developers can evaluate games with play testing that is mostly done in-house, and based on formal observation of temporary consumer testers. The proposed model suggests that during the play testing, the ratio of monitors to testers should not be less than one-to-one, and monitors should be observing where the players get stuck, answering the questions and comments, and responding to the emotional reactions they have. Although the author informs us on how to do in-house play testing, it lacks providing a solution on how to measure the feedback received from testers.

Marks et al. [2007] state that games should be evaluated in a more detailed manner. They point out that designers and developers should carefully choose game engines because game engines come with limitations. More importantly, the chosen game engine may not be appropriate for the targeted educational game. During their study, they examined a simulated surgical training game and focused on the relevance of the game engines for developing applications for medical education. They formally evaluated "the availability, the stability, the possibility of the content creation, and the interaction of multiple users via network". Results show that some of the game engines are not capable of creating fully interactive applications.

According to Pagulayan et al. [2003], it is important to evaluate users' subjective experiences and attitudes about games. They provide a list of items as evaluation criteria and some of them are listed below:

- Ease of use
- Overall quality (Is this game fun?)
- Tutorials or instructional missions
- In-game interfaces
- Challenge

Similarly, Kasvi [2000] lists the following requirements suggested by Norman [1993] for an effective learning environment; "provide a high intensity of interaction and feedback, have specific goals and established procedures, be motivational, provide a sense of direct engagement on the task involved and provide the appropriate tools that fit the task".

Pappa and Pannese [2010] present an evaluation framework that focuses on three criteria in digital games: (1) easy to use IT products, (2) fun and engaging, and (3) effective learning instruments. They examine each of these criteria from the perspective of user experience and have composed their overall evaluation framework as follows:

- Technical verification
- User experience evaluation
- Pedagogical aspects (evaluation of learning outcomes)

2.5 Taxonomy of Games

Taxonomy is the study of identification process on a specific domain, which organizes entities into a system of classification. The creation of taxonomy provides meaningful names. It enables us to see relation between entities in the taxonomy that can be used to identify patterns.

2.5.1 Overall Games Classification

Games always have been a part of our life. They appear in different shapes but mostly show similar characteristics [Huizanga 1955]. Salen and Zimmerman [2003, 2004] define "game" as "a voluntary interactive activity, in which one or more players follow rules that constrain their behavior, enacting an artificial conflict that ends in a quantifiable outcome."

Ellington et al. [1982] present that games should include a competition between individuals or teams and they categorize games in two sections as follows: psychomotor skills games and (computer-based) manual games as shown in Figure 2.

Psychomotor skill games	**(Computer-based) manual games**
Field games	Soccer, baseball, golf, tennis
Table games	Snooker, pool
Simple manual games	Charades, crossword, puzzles
Card games	Bridge
Board games	Chess, go, monopoly
Device-base games	Rubik's Cube

Figure 2. Game classification [Ellington et al. 1982]

According to the Caillois [1958] games can be classified based on two dimensions. The first dimension consists of four categories, which are AGON, ALEA, MIMICRY, and ILINX. The second dimension focuses on PAIDA and LUDUS as shown in Figure 3. While AGON and ALEA mainly focus on gamesmanship, the MIMICRY and ILINX represents the playfulness of activities.

PAIDA **(Freedom, free improvisation)**	**LUDUS** **(Rules & conventions)**
AGON- races, wrestling (competition: equal probability of success	soccer, chess
ALEA – counting rhymes Luck: players cannot exert any control over outcomes	lottery
MIMICRY – childish imitations (mask: players pretending to be someone else)	theatre
ILINX – merry go round (vertigo: attempts to disrupt regular perception patterns)	acrobatics

Figure 3. Game classification [Caillois 1958]

Familiar game classification systems are based on either the game's physical properties or its primary rules [[Ellis 1983; Thorpe et al. 1986; Werner and Almond 1996]. Hopper [1998] suggests a progressive principle of play based on the primary rules of games within each game category. The presented primary rules are as follows:

- In striking/fielding type games
- In target type games
- In net/wall type games
- In territory invasion type games

Hopper and Bell [1999] developed a theoretical framework (T.A.C.T.I.C.) for analyzing tactics in games based on five different components in play. Later Hopper and Bell [2000] presented a game classification system that categorizes games with fundamental consideration given to the tactical demands of the games within each category.

Klabbers [2003] combined the social systems theory with the semiotic theory of gaming and presented an integrated framework to understand the basic elements of gaming. Figure 4 shows their framework for the taxonomy of gaming.

Design specifications	1. Client 2. Purpose 3. Subject matter 4. Intended audience 5. Context of use		
Social system	**Syntax**	**Semantics**	**Pragmatics**
Actors	Players Number of game places	Roles	Allopoietic vs. Autopoietic Steering Knowledge as acquisition as interaction.
Rules	Game manipulation set: Preparatory rules; Normative rules; Behavioral rules Set of game positions Final game positions Evaluation functions	Relationships between roles Cultural, socio-economic situations Evaluation of places for resource allocation, and position within team of players	Team of game facilitators Format: rigid-rule vs. free form Assessment functions
Resources	Set of pieces Game space	Resources Set of places	Paraphernalia Equipment Facilities

Figure 4. Taxonomy of games [Klabbers 2003]

2.5.2 Games

We can separate games into two categories from a gameplay and interaction perspective.

- *Non-Digital Games:* In non-digital games the interaction directly takes place between the game players. Depending on the game type, a game can require special fields and equipment. For example, table tennis can be played with two or more players. The players hit a lightweight ball back and forth across a table using a small paddle.

- *Digital Games:* The interaction takes place between the player and the gaming hardware. Digital games are known as video games. Espesito [2005] describes the "video games" as: "a *game* which we *play* thanks to an *audiovisual apparatus* and which can be based on a *story"*. Although the definition is accurate, the author uses the term "video game" to define digital games, which is not appropriate to use in todays' world anymore. Currently, digital games are being played in so many different platforms such as smartphones and handheld computers.

2.5.2.1 Game Genres

Researchers have proposed different ways to analyze and categorize digital games [Gunn et al. 2009; Wolf and Baer 2002]. Konzack [2002] proposed a method that is based on seven different layers of a computer game. These layers are hardware, program code, functionality, game play, meaning, preferentiality, and socio-culture. Each of these layers is either analyzed individually or the entire analysis of computer the game is analyzed from every angle.

Alvarez et al. [2006] developed a tool called V.E.Ga.S., which is used to classify video games. The tool is used to create game bricks, which defines external game rules for classification of games. They define twelve different game bricks that are answer, manage, have luck, shoot, construction/creation, block, destroy, move, avoid, position, time, and score. The proposed tool enables researchers to see if the games offer similar objectives.

The complexity of games has been increased from many different perspectives such as graphics, interaction, and narrative. Nevertheless, researchers and the gaming industry have not come to consensus. A popular taxonomy system, Hertz system, divides games into eight categories: action, adventure, fighting, puzzle, role playing, simulations, sports, and strategy games [Kirriemuir and McFarlane 2004] Similarly the Kasvi system proposes the same categorization except the puzzle, adventure, and fighting [Kasvi 2000].

However, Crawford proposes a totally different classification which consists of only two categories: 1) *Skill and action games* that rely on hand-eye coordination and reaction; 2) *Strategy games* that rely on human reflection and involve strategy, adventure, puzzle, simulation and role-playing games [Kasvi 2000].

Gunn et al. [2009] define a taxonomy based on the interactivity in games. They present list of game types as follows: accessible vs. inaccessible, environmentally discrete vs. environmentally continuous, statistic vs. dynamic, deterministic vs. non-deterministic, turn-based vs. real-time.

Literature shows that researchers have proposed many different categorizations with respect to different perspectives. While some of the games cannot be categorized by any of the game classifications, some of them fit into many classifications at the same time. Game industry and developers are constantly trying to come up with new and interesting storylines and themes in which they combine many other game elements in one place. Hence, the classification of the games become very difficult.

CHAPTER 3: DIGITAL EDUCATIONAL GAME DEVELOPMENT METHODOLOGY (GAMED)

A *Digital Educational Game* (DEG) is a game created for the purpose of teaching a subject in the form of software that runs on a computer such as desktop, laptop, handheld, or game console.

Johnson et al. [2011] identify game-based learning as a technology expected to gain widespread usage. Gee [2007] indicates that the theory of learning in good DEGs fits better with the modern, high-tech, global world today's children and teenagers live in. Prensky [2007] points out that DEG-based learning (a) meets the needs and learning styles of today's and the future's generations of learners, (b) is motivating, because it is fun, (c) is enormously versatile, adaptable to almost any subject, information, or skill to be learned, and (d) when used correctly, is extremely effective.

Felicia [2011] identifies one of the DEG-based learning challenges as, "More developers need to be informed of best practices pertaining to the design of successful educational games." Development of a DEG in the form of software for DEG-based learning poses significant technical challenges for educators, researchers, game designers, and software engineers. The DEG development consists of a set of complex processes requiring multi-faceted knowledge in multiple disciplines such as digital graphic design, education, gaming, instructional design, modeling and simulation, psychology, software engineering, visual arts, and the learning subject area. Planning and managing such a complex multidisciplinary development project require a unifying methodology and should not be performed in an *ad hoc* manner.

In this chapter, we present such a methodology named GAMED (diGital educAtional gaMe dEvelopment methoDology). GAMED consists of a body of methods, rules, and postulates and is embedded within a DEG life cycle consisting of four phases and a dozen processes.

This chapter is organized as follows. After an introduction, Section 3.1 advocates the motto "Quality is Job 1!" and presents a DEG quality assurance strategy. Section 3.2 describes the DEG life cycle, which provides the foundation of GAMED.

3.1 Digital Educational Game Quality Assurance

As the saying goes "Quality is Job 1!" The ultimate goal of a DEG development project is to produce a game in the form of a software application with desirable quality characteristics such as customizability, dependability, engagability, functionality, performance, playability, supportability, and usability

DEG *Quality Assurance* (QA) refers to the planned and systematic activities that are established throughout the DEG life cycle to substantiate adequate confidence that the DEG software application possesses certain characteristics required for a set of intended uses of the DEG.

The more comprehensive and detailed the overall game quality evaluation is, the more confidently a decision can be reached for the acceptability of the game. Four major perspectives or four Ps influence the game quality as depicted in Figure 5.

DEG quality evaluation can be approached from any one of the four Ps, but a combination of all four will provide the best balance and result in a much higher level of confidence in the game quality. Therefore, we advocate the following strategy.

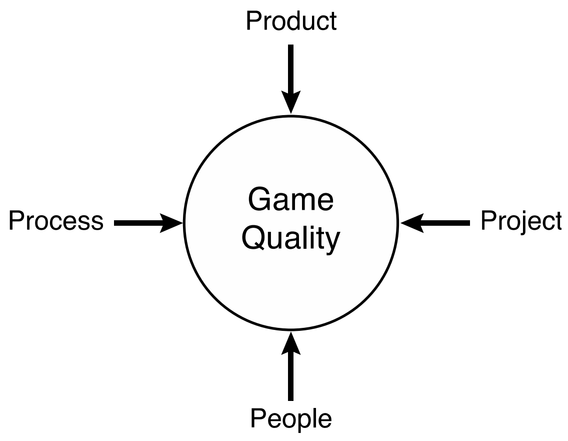


Figure 5. Four Ps Influencing the Game Quality

The DEG QA strategy should involve the measurement and integrated evaluation of a particular life cycle stage's (e.g., game design stage's)

- Output work *Product* (or artifact), (e.g., game design specification),
- *Process* used in creating the output work product, (e.g., game design),
- Quality of the *People* employed in executing the process to create the work product, and
- *Project* characteristics (e.g., configuration management, risk management, project planning, project monitoring and control).

DEG quality is the degree to which the DEG possesses a desired set of characteristics. An example set of quality indicators is provided below for evaluating the quality of a DEG:

1. *Acceptability*: the degree to which the DEG fulfills its requirements and learning objectives.
2. *Challengeability*: the degree to which the user finds the DEG to be exciting, stimulating, and inspiring to play.
3. *Clarity*: the degree to which the DEG is unambiguous and understandable.
4. *Effectiveness*: the degree to which the DEG improves the effectiveness of learning the subject in a significantly better way in comparison to other pedagogies.
5. *Engageability*: the degree to which the user is captivated and addicted by DEG playing.
6. *Enjoyability*: the degree to which the user finds the DEG playing to be fun.
7. *Interactivity*: the degree to which the user actively interacts with the DEG during playing.
8. *Localizability*: the degree to which the DEG can easily be adopted, preferably via preferences or options, (a) to satisfy the needs of languages other than English, and (b) to local standards such as decimal separator, currency symbol, time zone, calendar, etc.
9. *Rewardability*: the degree to which the DEG gives rewards (e.g., points, money, trophies, certificates) to the user so that the user feels a sense of accomplishment.
10. *Simplicity*: the degree to which the DEG can be understood without difficulty.
11. *Transformativeness*: the degree to which the DEG transforms the subject learning in a significantly better way in comparison to other pedagogies.
12. *Usability*: the degree to which the DEG can easily be employed for its intended use.

3.2 Digital Educational Game Life Cycle

GAMED requires the employment of the DEG life cycle presented in Figure 6 and is embedded within that life cycle. The DEG life cycle provides the foundation of GAMED. It represents a *framework* for organization of the phases, processes, work products, quality assurance activities, and project management activities required to develop, use, maintain, and evolve a DEG from birth to retirement. The DEG life cycle modularizes and structures the development of a DEG and provides guidance to game designers, software engineers, and project managers.

The DEG life cycle consists of four phases: game design phase, game software design phase, game implementation and publishing phase, and game-based learning and feedback phase. Each phase consists of a number of stages. Stages are named after the processes, e.g., game idea generation stage, architecting stage, programming stage. Each stage consists of a process and a work product, e.g., requirements development stage consists of the requirements development process producing the requirements specification document as a work product.

The DEG life cycle enables to view DEG development from four perspectives (or Ps): Process, Product, People, and Project. The DEG life cycle (a) specifies which work *Product* to produce by executing which *Process* together with the integrated quality assurance activities, (b) provides valuable guidance for *Project* management, and (c) identifies areas of expertise in which to employ qualified *People*. [Balci 2012]

The DEG life cycle represents the processes in a logical order starting with "Education Problem Domain" using double-line arrows as depicted in Figure 6. Although the arrows show a sequential progress, the life cycle should *not* be interpreted as strictly sequential. The sequential representation of the double-line arrows is intended to show the direction of workflow throughout the life cycle.

The DEG life cycle is *iterative* in nature and reverse transitions are expected. The backward solid line arrows between work products represent the iterative nature of the development. For example, a problem identified in game software design may require changes in game design and hence, we bounce back to an earlier process and redo the work. We typically bounce back and forth between the processes until we achieve sufficient confidence in the quality of the work products throughout the life cycle. [Balci 2012]

A DEG life cycle process, represented by a double-line arrow, is executed to produce a work product. For example, the *game design* process is executed to produce the work product *game design specification*, the *architecting* process is executed to produce the work product *game architecture specification*, and the *programming* process is executed to produce the work product *game software components*.

Quality Assurance (QA), shown in rounded rectangle symbols in Figure 6, cannot be designated as a stage in the DEG life cycle. QA consists of continuous activities conducted hand in hand as integrated with every DEG life cycle task [Pressman and Maxim 2015]. We represent this principle by placing the QA symbol below each work product as depicted in Figure 6.

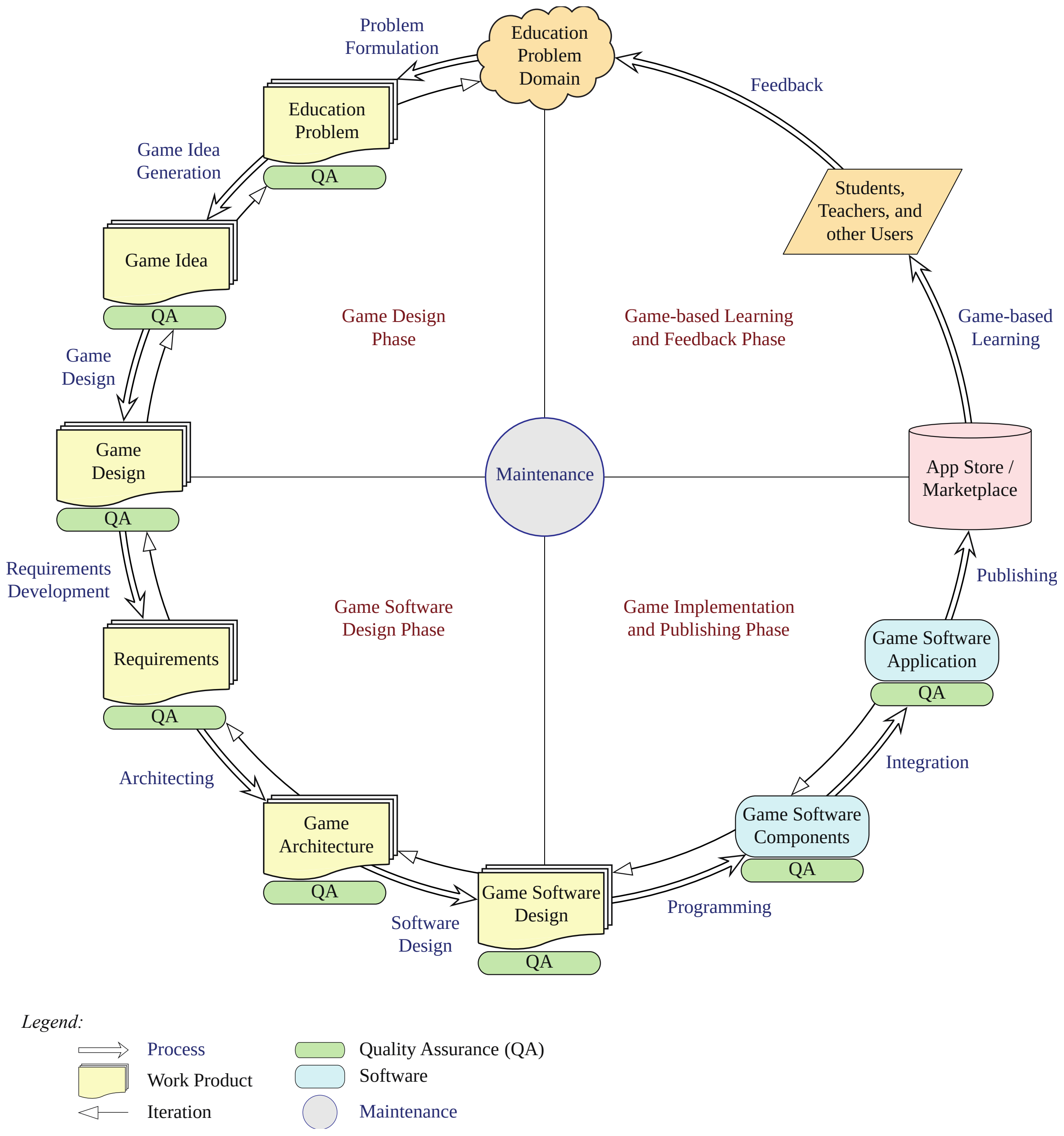


Figure 6. Digital Educational Game Life Cycle

3.2.1 Problem Formulation

Education Problem Domain is the starting point for the DEG life cycle as depicted in Figure 6. A DEG is developed to address a specific problem formulated in this domain. The process of *Problem Formulation* can be executed by following the steps described below.

Step 1. *Identification of the education problem*. Identify a subject that poses serious challenges for learning, e.g., learning fractions by middle school students poses serious challenges.

Step 2. *Significance of the education problem*. Explain why the identified education problem is critically important to solve.

Step 3. *Identification of the state of the art in solving the education problem*. Identify the state-of-the-art pedagogies under which the education problem is currently being solved. Explain why the existing pedagogies are not good enough.

Step 4. *Effectiveness of game-based learning for solving the education problem*. Describe why game-based learning would provide a more effective approach to solving the education problem in comparison with the existing pedagogies. *Continue with the DEG life cycle only if game-based learning is found to be an effective approach to solving the education problem.*

Step 5. *Definition of the education problem*.

a. Establish the problem domain boundary. A *problem domain boundary* identifies which elements will be included and which elements will be excluded. What to include and what not to include is considered as the art of balancing the opposites. The boundary must be large enough to encapsulate all essential elements of the problem domain.
b. Gather data and information about the problem domain within the established boundary.
c. Identify the stakeholders and decision makers who would be interested in the solution of the education problem.
d. Specify the needs and objectives of the stakeholders and decision makers identified.
e. Identify and specify the constraints.
f. Specify all assumptions made clearly and explicitly and validate each assumption individually.
g. Produce a well-structured definition of the education problem.

Step 6. *Specification of the game-based learning objectives*. Acceptability of the DEG to be developed will be judged with respect to the learning objectives, which must be clearly and explicitly stated.

As the saying goes "A problem correctly formulated is half solved." Sufficient time and effort must be expended at this life cycle stage to make sure that Type III Error is not committed. (Type III Error is the error of solving the wrong problem. [Balci and Nance 1985])

Execution of the *Problem Formulation* process produces the *Education Problem Specification* document as a work product containing the results of performing the steps described above.

The QA strategy for the problem formulation stage involves the measurement and integrated evaluation of the (a) education problem specification document as a work *Product*, (b) problem formulation *Process*, (c) quality of the *People* employed in executing the problem formulation process, and (d) *Project* characteristics that are specific to this life cycle stage.

3.2.2 Game Idea Generation

One of the pivotal options for fostering learning process effectively is the DEGs. DEGs provide an environment where kids can easily capture the context that enables them to easily think, understand and execute effectively [Gee 2003; Ke 2009].

Although DEGs provide benefits and being used to teach different concepts, development of DEGs is not a straightforward process [Bridgeland et al. 2006; Gee 2009; Joyce et al. 2009]. The development life cycle requires a structured guidance and more importantly to achieve the projected success the development needs to start with a quality game idea.

Idea generation has a crucial role in game industry. Generating the next expected game idea for game players can bring the success to the companies [Gabler et al. 2005]. Although the game industry holds most of the highly talented people, this does not simplify the idea generation process. Ideas can come from anywhere but they do not introduce themselves randomly. It requires creativity and also your mind actively needs to be searching for them [Tschang and Szczypula 2006].

Processes are important to streamlining the work required to develop a solution from inception through operation and to retirement. Solution providers typically focus on three dimensions of their development approaches: People, methods, and tools [SEI 2010]. To link these dimensions together, organizations follow processes to ensure the production of systems with the desired functionality and quality, on time, and within budget.

Similarly, coming up with a high quality game idea is positively correlated with the process used to develop such an idea. Just like any other product development, the development of a game idea also requires a structured guidance by identifying the process and work products. Therefore, we strongly advocate that the game idea generation should have its own process.

3.2.2.1 Game Idea Generation Process

We describe the game idea generation process following the CMMI model. In the following sections we define the purpose, activities, specific goals and specific practices.

3.2.2.1.1 The Purpose of Game Idea Generation Process

The purpose is to develop a game idea, which is unique and unusual, yet practical and useful that possesses sufficient quality such as acceptability, transformativeness and challengability as defined in section 3.1. The process of *Game Idea Generation* takes the *Education Problem Specification* document as input and produces *Game Idea Specification* document as an output work product.

3.2.2.1.2 Introductory Notes

The game idea generation process encompasses all activities performed to produce the *game idea specification document* as a work product. A *game idea specification* document defines an overview for the following items: the game idea, the goal of the game, game elements (e.g. characters, world, story and controls), the gameplay, the genre, and the game levels.

The game idea generation process is a structured process to generate a game idea and the corresponding game idea specification document. A game idea generation process consists of the following actions:

1. Performing quality assurance continuously
2. Assembling game idea generation team
3. Stimulating creativity
4. Holding idea generation meetings
5. Processing generated game ideas
6. Producing game idea specification document

3.2.2.1.3 Specific Practices and Goals of the Game Idea Generation Process

We describe the game idea generation process based on a structure advocated by SEI [2010] in terms of Specific Goals (SGs) and their associated Specific Practices (SPs) as shown in Table 1.

Table 1. Game Idea Generation Process

SG 1	Perform Quality Assurance Continuously
	SP 1.1 Employ For Ps for Quality Assurance
	SP 1.2 Employ Quality Characteristics
SG 2	Assemble Game Idea Generation Team
	SP 2.1 Identify a Moderator
	SP 2.2 Identify a Scribe
	SP 2.3 Identify Teachers
	SP 2.4 Identify Game Players
	SP 2.5 Identify Game Developers
	SP 2.6 Identify a Quality Bearer
SG 3	Stimulate creativity
	SP3.1: Select an Accepted Genre or a Game Type
	SP3.2: Combine Different Games Genres or Game Types
	SP3.3: Generate Ideas From Other Media
	SP3.4: Generate Ideas From Real Life
	SP3.5: Generate Ideas From Imagination
SG 4	Hold Idea Generation Meetings
	SP4.1: Provide Information to Team Members
	SP4.2: Task Team Members to Prepare
	SP4.3: Execute Game Idea Generation Meeting
	SP4.4: Document Each Game Idea Generated
	SP4.5: Produce and Distribute Meeting Notes
SG 5	Process Generated Game Ideas
	SP5.1: Group Generated Game Ideas
	SP5.2: Perform Comparative Evaluation
	SP5.3: Select the Best Game Idea
SG 6	Produce Game Idea Specification Document
	SP6.1: Define Learning Objectives and Target Audience
	SP6.2: Describe Game Story
	SP6.3: Identify Game Characters
	SP6.4: Describe Game Play and Genre
	SP6.5: Describe Game Levels
	SP6.6: Describe Quality Assurance Results

SG 1 Perform Quality Assurance Continuously

Goal Statement: *Game ideas are evaluated based on Four Ps and Game Quality Characteristics.*

QA is not a stage but a continuous set of activities conducted hand in hand together with the development activities. The QA strategy described in section 3.1 should be followed. To achieve this goal, the following specific practices are employed.

SP 1.1 Employ the Four Ps for Game Quality Assurance

Practice Statement: Game idea is evaluated based on four Ps.

The QA strategy for the game idea generation stage of the life cycle involves the measurement and integrated evaluation of the (a) game idea specification document as a work *Product*, (b) game idea generation *Process*, (c) quality of the *People* employed in executing the game idea generation process, and (d) *Project* characteristics that are specific to this life cycle stage. [Balci and Ormsby 2008]

SP 1.2 Employ Game Quality Indicators

Practice Statement: *Game idea is evaluated based on Game Quality Characteristics.*

Game idea is evaluated by employing game quality indicators such as the ones presented in section 3.1.

SG 2 Assemble Game Idea Generation Team

Goal Statement: *Game idea generation team is assembled.*

The game idea generation process should be executed in a team environment. The team holds as many game idea generation meetings as needed. The team may include subject matter experts (e.g., teachers), game-based learning experts, game designers, game players (e.g., students), and software engineers. To achieve this goal, the following specific practices are employed.

SP 2.1 Identify a Moderator

Practice Statement: *Moderator is identified.*

The moderator presides over the game idea generation meetings to facilitate the discussions and encourage team members to be creative.

SP 2.2 Identify a Scribe

Practice Statement: *Scribe is identified.*

The scribe records the participants' comments, game ideas proposed, decisions reached, action items, and produces meeting notes after the meeting.

SP 2.3 Identify Teachers

Practice Statement: *Teachers are identified.*

Teachers provide expertise about the state of the art in pedagogies for teaching the subject, define the learning objectives, and help the team understand the game-based learning subject.

SP 2.4 Identify Game Players

Practice Statement: *Game players are identified.*

Experienced entertainment or educational game players can generate game ideas based on their playing experience.

SP 2.5 Identify Game Developers

Practice Statement: *Game developers are identified.*

Game developers provide expertise about game design, software engineering, software graphics and visualization, game development tools and techniques, and game deployment platforms such as mobile devices.

SP 2.6 Identify a Quality Bearer

Practice Statement: *Quality Bearer is identified.*

The quality bearer provides feedback to the team about whether a proposed game idea can lead to the development of a game with acceptable quality.

SG 3 Stimulate Creativity

Goal Statement: *Creativity of the team members is stimulated.*

Creativity refers to the ability to produce a new game idea through imaginative skill, richness of ideas, and originality of thinking. To achieve this goal, the following specific practices are employed.

SP 3.1 Select an Acceptable Game Genre

Practice Statement: An accepted game genre or game type is selected to generate game ideas.

Creativity can be stimulated by narrowing the game idea generation under a specific game genre such as adventure game, multiplayer online game, puzzle game, or role-playing game. For example, the Real Time Strategy (RTS) games mainly consists of a battle, there are different cities, buildings, terrains, two or more different sides, various units and resources. Removing or adding a rule or a game element can result in a different gameplay experience. [Stuart 2012]

SP 3.2 Combine Different Game Genres

Practice Statement: *Different game genres or game types combined to generate game ideas.*

Combining the features of different types of games may lead to the creation of a new game type with innovative composite characteristics. For example, action and adventure games are combined to create action-adventure game types such as popular Tomb Raider series [Gal et al. 2002; Rollings and Adams 2003; Stuart 2012].

Also we can combine a theme of one game with a gameplay of the other game such as a puzzle game with a car theme or first person shooter (FPS) game with a prison theme. These combinations can be good a starting point to come up with ideas that we have not previously thought. Moreover, recent games have shown that game industry takes the best of all worlds and combine them into a new story and gameplay, which enables them to create better games with many rich features. Grand Theft Auto series is an example for this approach, which combines different genres such as crime, and fantasy and other gameplay features [Rockstar Games 2013].

SP 3.3 Generate Ideas From Other Media

Practice Statement: *Game ideas are generated based on other media (books, movies).*

Identifying stories in media such as books, magazines, movies, newspapers, and television can boost creativity for coming up with a game idea. For example, the game “Enter the Matrix” was inspired by “The Matrix” movie series, the game Interstate '76 was inspired by 1970s cop shows.

SP 3.4 Generate Ideas From Real Life

Practice Statement: *Game ideas are generated based on real life activities.*

Real life is a good source for game ideas. Think over the real life and ask yourself whether any of the activities in them could serve as the basis for a game [Rollings and Adams 2003; Stuart 2012]. Fishing, hunting, managing a store, cleaning, and many other daily activities can become a game idea. For example, a recent game Farmville, which has been converted to a game from a real-life farm idea, had such a big impact on the players. Farmville enables players to milk their cows, collect eggs, harvest crops and many other real life activities. It also enables players to share their scores, badges and other game related trophies in online platforms [Zynga 2013].

SP 3.5 Generate Ideas From Imagination

Practice Statement: *Game ideas are generated based on imagination.*

Imagination is considered to be the foundation for a creative mind. Albert Einstein said, "Imagination is more important than knowledge. For knowledge is limited to all we now know and understand, while imagination embraces the entire world, and all there ever will be to know and understand."

SG 4 Hold Game Idea Generation Meetings

Goal Statement: *Game ideas are generated and a bubble diagram is created to display game ideas.*

Creative team brainstorming should take the center stage for a game idea generation meeting. It encourages collaboration, enables team members to feel more open to bouncing ideas off one another, facilitates different perspectives, and improves the team's ability to think outside the box. To achieve this goal, the following specific practices are employed.

SP 4.1 Provide Information to Team Members

Practice Statement: *Team members received all necessary information before the game idea generation meetings.*

All team members receive the *Education Problem Specification* document and any other related information before the meeting.

SP 4.2 Task Team Members to Prepare

Practice Statement: *Team members are prepared for the game idea generation meetings.*

Adequate time is given to the team members to prepare themselves for the meeting by studying the information provided and conducting a literature review to familiarize themselves with the education problem domain.

SP 4.3 Execute Game Idea Generation Meeting

Goal Statement: *Game idea generation meeting is executed and game ideas are displayed in a bubble diagram.*

The execution of this activity requires the following steps [Institute for HealthCare Improvement 2004; Hill 2011]:

1. The moderator starts the meeting and states the purpose of the brainstorming session and goes over the agenda items.

2. The moderator presents the learning objectives and goals, also clarifies any related questions from the team members.
3. After successful completion of the presentation, the moderator starts the discussion for new game ideas.
4. The team members express one game idea at a time, either going around the round in turn, or at random.
5. The moderator manages the team discussion carefully to prevent team members from criticizing, and complimenting on game ideas, which can slowdown the process. S/he pushes the team to focus on generating as many game ideas as possible in a short period of time.
6. The scribe records all game ideas on a digital or non-digital platform where all team members can see. For example, large chart pads can be used to display the bubble diagram for game ideas.
7. The moderator encourages team members to build game ideas based on others' game idea.
8. If a team member cannot come up with an idea the moderator moves on to the next team member.
9. The moderator keeps repeating the process till the game ideas slow down.
10. After all ideas are listed on large chart pads as a bubble diagram, the moderator leads the team to clarify each of the game idea and eliminate any duplicates.

Figure 7 shows list of game ideas for a candy theme as a bubble diagram. Each bubble refers to a potential game idea. Although the bubbles have only couple of words in it, they refer to an overview of the game idea, which answers the following questions:

- What is the goal or purpose of the game?
- What is the gameplay style?

For example, the bubble "Create and ship candies" refers to an overview of the *Candy Factory Educational Game for iPad:* You are a factory worker and you are getting promoted to become a manager. Your task is to manufacture and ship candy bars ordered by customers. You need to ship customers' orders as fast as possible to keep customers happy, otherwise you will get fired [Aslan 2011].

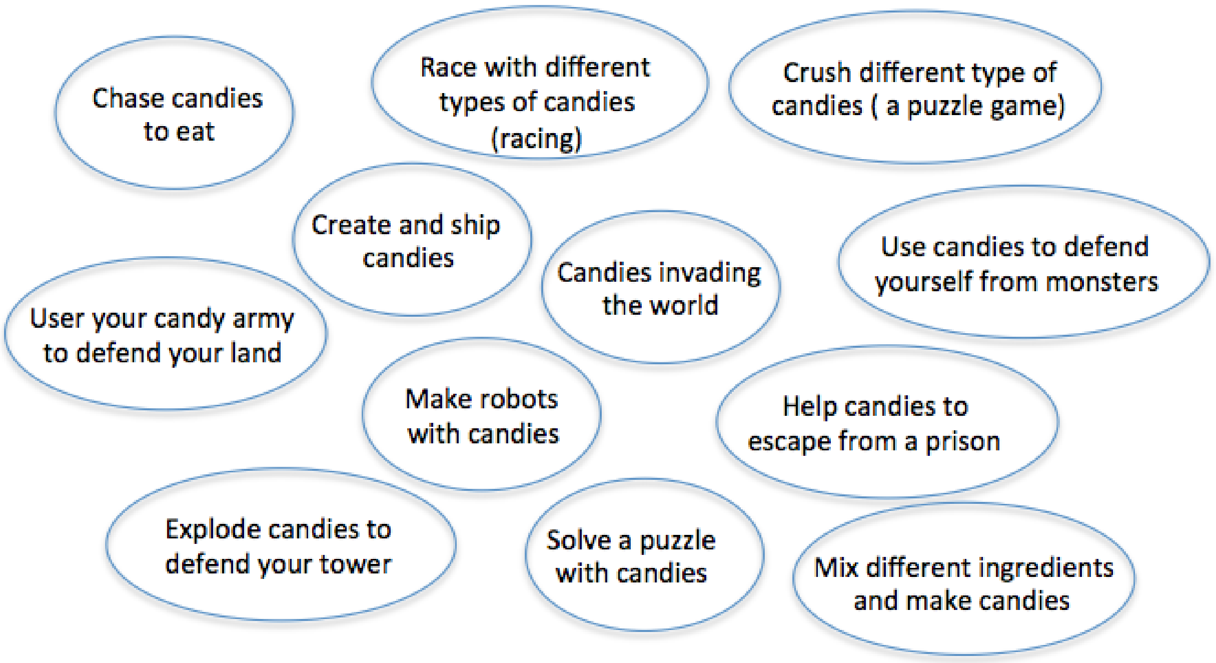

Chase candies to eat
Race with different types of candies (racing)
Crush different type of candies (a puzzle game)
Create and ship candies
Candies invading the world
Use candies to defend yourself from monsters
User your candy army to defend your land
Make robots with candies
Help candies to escape from a prison
Explode candies to defend your tower
Solve a puzzle with candies
Mix different ingredients and make candies

Figure 7. Bubble diagram

SP 4.4 Document Each Game Idea Generated

Practice Statement: *Game ideas are documented.*

In this practice each game idea should be documented with more detail. Mostly couple of paragraphs, which broadens the information, will be adequate. For example, we can expand the overview of the "create and ship candies" bubble, to explain what is the story behind the creating and shipping, what are the challenges during the creation and shipping processes, and how the game is unique.

Answering some of the questions given below can help us to describe the game idea: [Mitchell 2012; Rogers 2010]

- Who is the main character (a hero, a monster)?
- What does the main character want (save the princess, manage a town, kill monsters)?
- What are the challenges (killing all monsters, destroying all buildings, finishing the race in a limited time)?
- What is the game world (a factory, a real world, a town, a space)?
- What is the narrative?
- What is the game play?
- How the game is different or fascinating than other games?

The outcome of this specific practice should be a description of an idea. Below is an example of a description for the "create and ship candies" bubble which is developed as *Candy Factory Educational Game for iPad* game.

"The player is a factory worker who has been working in the candy factory for several years. The president (Carmelo) thinks that you deserve a promotion to become a manager; despite this the Boss Cog thinks that you are not ready for that kind of promotion. As a player you need to prove yourself by way of manufacturing and shipping the right amount of candy with in a given limited time and keep your customers happy.

Producing a right amount of candy in limited time will be challenging, you need to make sure that each piece manufactured correctly and shipped in the shortest time to keep the customer satisfaction as high as possible. If the customers are not happy with their orders, you will be fired, which will be the end of the game.

The game is intended to help students gain a deep understanding of fraction concepts that are critical as they begin to learn algebra. Especially letting students comprehend and understand the features of denominator and numerator is the main focus of the game. The game will be different from its competitors in the sense of not providing instruction to the players but letting them to understand the new conception of fractions and splitting. While other educational games, focusing on fractions, mostly provide drills and practice, this game is unique in terms of helping students understand the idea of splitting and conception of fractions" [LTRG 2014].

SP 4.5 Produce and Distribute Meeting Notes

Goal Statement: *Meeting notes are produced and distributed in the team.*

The scribe produces the meeting notes and distributes them to the team members after the meeting.

SG 5 Process Generated Game Ideas

Goal Statement: *Game ideas are grouped, a comparative evaluation is executed and the best game idea is selected.*

After a game idea generation meeting, generated game ideas are analyzed, grouped, and compared with each other. If a game idea with sufficient quality cannot be selected, the SPs of SG 4 above should be performed again. The SPs of SG 4 and SG 5 are performed in an iterative manner, bouncing back and forth, until a game idea with sufficient quality is selected. To achieve this goal, the following specific practices are employed.

SP 5.1 Group Generated Game Ideas

Practice Statement: *Game ideas (bubbles) are organized and grouped based on the similarity of game features such as gameplay, story.*

Some game ideas may be related to each other and grouping them facilitates the analysis of their characteristics. Combining the features of related game ideas can produce a better composite game idea.

The practice is executed as follows: [Institute for HealthCare Improvement 2004; MSH and UNICEF 1998]

1. The moderator starts the grouping activity and states the purpose of this activity.
2. The moderator leads participants to define a relationship between two ideas at a time, either going around the round in turn or at random.
3. Each team member looks for two bubbles (ideas) that seem to be related and places these together without any discussion, off to one side on a new large chart pads.
4. Other team members can add additional bubbles, to a group as it forms or reform existing groups.
5. While grouping the bubbles, different colors can be used for different groups to help team members to visually analyze the difference between each group.
6. Team members continue until all items have been grouped.

7. The moderator starts a team discussion for each grouping and team generates an overview for each idea. If there are conflicts bubbles can be removed from one group and add to another one or a bubble can be duplicated and keep it in two different groups.
8. If necessary, the moderator can consider additional brainstorming to capture new ideas.

For example, Figure 8 displays grouped ideas based on the bubble diagram we developed earlier. Group 4, 5 and 6 are the ideas that are set aside.

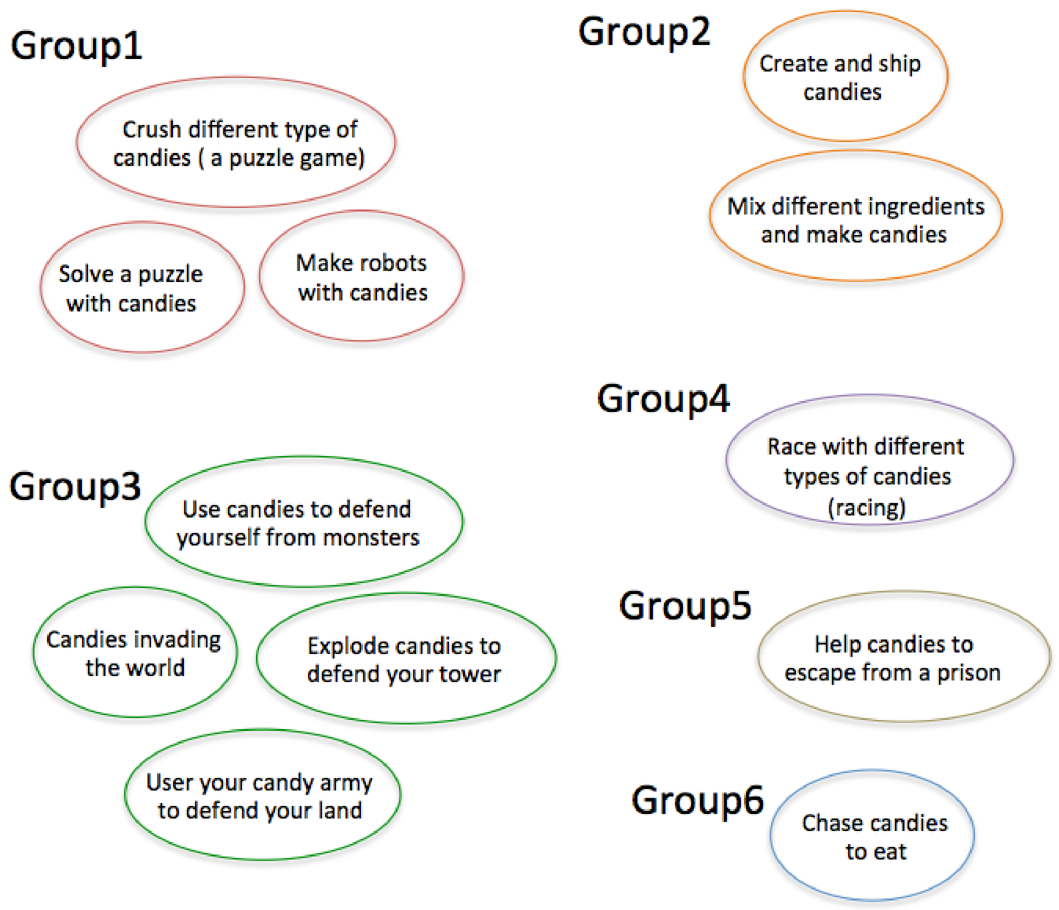

Group1
Crush different type of candies (a puzzle game)
Solve a puzzle with candies
Make robots with candies
Group2
Create and ship candies
Mix different ingredients and make candies
Group3
Use candies to defend yourself from monsters
Candies invading the world
Explode candies to defend your tower
User your candy army to defend your land
Group4
Race with different types of candies (racing)
Group5
Help candies to escape from a prison
Group6
Chase candies to eat

Figure 8. Grouped game ideas

SP 5.2 *Perform Comparative Evaluation*

Goal statement: *Game ideas are compared and evaluated based on the game quality indicators*.

Identified game ideas in different groups are comparatively evaluated using quality indicators such as the ones listed in section 3.1.

SP 5.3 *Select the Best Game Idea*

Practice Statement: *The best game idea is selected.*

The game idea with the best quality characteristics is selected. The SPs of SG 4 are executed again if no selection can be made.

Selecting the best game idea consists of series of votes by a team, in order to reduce list of ideas [Institute for HealthCare Improvement 2004].

1. The moderator starts the voting activity and states the purpose of this activity.
2. The moderator gives a number to each grouped idea in an area where each team members can see.
3. The moderator leads participants to vote for ideas at a time, either going around the round in turn or at random.
4. Each team member chooses one-third of the items.
5. The moderator tallies votes.
6. The moderator eliminates items that have few votes.
7. If necessary, the team repeats the selecting activity with remaining items.
8. The moderator announces the selected best idea.

SG 6 Produce Game Idea Specification Document

Goal Statement: *The Game idea specification document is generated.*

Execution of the *Game Idea Generation* process produces the *Game Idea Specification* document as a work product, which describes at least the following: learning objectives, target audience, game story, game characters, gameplay, game levels, and quality assurance results. To achieve this goal, the following specific practices are employed.

SP 6.1 Define Learning Objectives and Target Audience

Practice Statement: *Learning objectives and target audience are defined.*

The game-based learning objectives for a particular subject and the intended target audience (e.g., middle school students) are clearly and explicitly defined.

Below is an example from the game design document of the *Candy Factory Educational Game for iPad.* [LTRG 2014]

Target audience: “The primary users of the game are targeted to be ten-twelve years old sixth grade students.”

The instructional Objectives: “The educational goal of the game is not to teach fractions as a topic but to support students' development of two important constructs: splitting and a new conception of fractions. Splitting is a mental action that composes existing mental actions of partitioning and iterating. Partitioning is breaking a continuous whole into n equal parts. For example, given a line segment, students can project five equal parts into it. Iterating is making connected copies of a given part. For example, given a 1/5 part, students can imagine what 3/5 would look like by making two more connected copies of the given part. Splitting composes partitioning and iterating as inverse operations; iterating and partitioning undo each other. For example, students who can split understand how they can move back and forth between 1/5 and 3/5 by iterating and partitioning; the two mental actions are coordinated.

The game will allow students to connect and transfer the content emphasized in the game related with curriculum on fractions. The game can be used to revisit fractions topics for reinforcement during the academic year when the related topic is studied”.

SP 6.2 Describe Game Story

Practice Statement: *Game story is defined.*

A high-level description of the game story is provided. The ultimate goal of creating a game story should be always finding the balance between the gameplay and the story. We should be aware that too much of a story can decrease our game quality.

Below is an example from the game design document of the *Candy Factory Educational Game for iPad* [LTRG 2014].

The story: "You are a factory worker who has been working in the candy factory for several years. The president of the factory Carmelo thinks that you deserve a promotion to become a manager; despite this the Boss Cog thinks that you are not ready for that kind of promotion. As a player you need to prove yourself by way of manufacturing and shipping the right amount of candy with in a given limited time and keep your customers happy."

The Game World: "The game world is formed in a factory environment; it has the theme of candy."

The Theme: "The theme of the game has been chosen to reflect the content of the game. Candy Factory theme describing a worker producing exact same sizes of chocolate bars is chosen to connect the players to the real world setting rather than driving them to the imaginary world. The Candy Factory Game is expected to have a strong sense of aesthetics, high quality visuals, and animations".

SP 6.3 Identify Game Characters

Practice Statement: *Game characters are defined.*

Characters significantly influence game quality. A good character is someone a player can relate to. Game characters are explicitly identified.

Below is an example from the game design document of the *Candy Factory Educational Game for iPad* [LTRG 2014].

The Characters:

1. *Carmelo*: "She is between ages of 35 and 40. She is the president of the Candy Factory Inc. She is an African-American woman. She has a strong personality, and positive attitude overall. Also, she is good at her job. She likes to motivate other employees in the company to do their job better".
2. *Boss Cog*: "He is between the ages of 45 and 50. He is an old grumpy an American guy and he doesn't trust people easily. Also, he is a hard man to satisfy. He demands a lot from company employees and makes sure that everybody is doing his or her job very well".

SP 6.4 Describe Gameplay and Game Genre

Practice Statement: *Gameplay and game genre are described.*

Gameplay defines how players interact with a game under a set of rules and mechanics, as distinct from the game's graphics and sound effects. A high-level description of the gameplay is provided.

Game genres are used to categorize digital games based on their narrative content. Defining game genre help us to see the common game features of that specific game genre. Some of the popular game genres are, drama, crime, fantasy, horror, mystery, science fiction, war and espionage [Grace 2005].

Below is an example from the game design document of the *Candy Factory Educational Game for iPad* [LTRG 2014].

The Gameplay: "The player's task is to manufacture the candy bar ordered by a customer by using whole candy that is obtained from factory's warehouse. The Candy Factory game starts with a randomly created customer demand of a candy bar, which is a part of whole candy bar. The player is given a whole candy bar, and is expected to reproduce the customer order by partitioning this candy bar and then iterating through appropriate number of times. In order to progress through this level, the players have to make rough estimates of the relative sizes of the bars, using partitioning and splitting operations.

You need to make sure that each piece manufactured correctly and shipped in the shortest time to keep the customer satisfaction as high as possible. If the customers are not happy with their orders, you will be fired, which will be the end of the game".

The Genre: "The genre of the Candy Factory game is educational, "educational games emphasize learning" (Pedersen, 2002, p.23)".

SP 6.5 Describe Game Levels

Practice Statement: *Game levels are described.*

Games may be decomposed into modules (levels) to overcome complexity. The game levels are defined in increasing difficulty requiring the player to complete a lower level to advance to the next level.

Game level refers to an environment, a location where game play occurs or a different physical game space. While some game levels are pretty complex, some can be pretty simple based on the game idea and the game play. Carefully designed levels should provide new and interesting challenges as players' progress through the game. Also, depending on the game idea and the game play, levels can either begin with a start point or a key event such as solving a puzzle or battle to fight. Designing successful levels requires a careful thinking process to generate the expected levels for game players. Some of the following questions can assist creating game levels [Byrne 2005; Mitchell 2012; Rogers 2010; Schell 2008].

- What does the game environment look like? (Jungle, ocean, desert, etc.)
- Is the environment friendly, scary, dark?
- Where does hero character want to go?
- Where will hero start to game?
- Where will hero end up?
- What will hero do along the way?
- What should be the difficulty of a level?
- How long should a level take for game players to complete?
- How should be the game environment if the game has multiplayer mode?

Below is an example from the game design document of the *Candy Factory Educational Game for iPad* [LTRG 2014].

Game Levels: "The level 1 of the game sticks to the familiar part- whole concepts for students to gain understanding of the task. Once the foothold is firm, the players can progress to level 2 and 3. In level 1, the candy bar has existing part markings on the candy bar, which makes it easier for the user. But in level 2 and 3 the candy bar is continuous, without any markings".

SP 6.6 Describe Quality Assurance Results

Practice Statement: *Quality assurance results are described.*

The results of applying the QA strategy described in Section 2 and under SP 1.1 are fully documented.

Below is an example of *Quality Assurance Result* for a Candy Span game [LTRG 2014].

Table 2. Informal Assessment of CandySpan Game

Quality Indicator	Description	Score	Technical Rationale
Acceptability	is the degree to which the game fulfills its requirements and learning objectives.	A	It is an effective game to improve one's memory span (skills)
Challengeability	is the degree to which the user finds the game to be exciting, stimulating, and inspiring to play.	A	Immediate feedback after each trial challenges the player
Clarity	is the degree to which the game is unambiguous and understandable.	A	We will make sure that this is A
Effectiveness	is the degree to which the game improves the effectiveness of learning the subject matter in a significantly better way in comparison to other pedagogies.	A	It is not learning but improving memory skills of someone / or learning to remember
Enjoyability	is the degree to which the user finds the game playing to be fun.	A	I love it!
Engageability	is the degree to which the user is captivated and addicted by game playing.	A	It is like Angry Birds; immediate feedback keeps the player addicted.
Interactivity	is the degree to which the user actively interacts with the game during playing.	A	Full interactivity with touching buttons
Localizability	is the degree to which the game can easily be adopted, preferably via preferences or options, (a) to satisfy the needs of languages other than English, and (b) to local standards such as decimal separator, currency symbol, time zone, calendar, etc.	B	We could provide a Preference under which the user sets the language. Since main interaction is based on gestures, it needs few language changes.
Rewardability	is the degree to which the game gives rewards (e.g., points, money, trophies, certificates) to the user so that the user feels a sense of accomplishment.	A	The performance measures listed at the end provide a sense of accomplishment.
Simplicity	is the degree to which the game can be understood without difficulty.	A	We will provide introductory info. It is a very simple game to play.
Transformativeness	is the degree to which the game transforms the subject matter learning in a significantly better way in comparison to other pedagogies.	A	Excellent game in terms of improving memory skills in comparison to other techniques.

Usability	is the degree to which the game can easily be employed for its intended use.	A	The iOS app is well human engineered.

3.2.3 Game Design

The game design process takes the *Game Idea Specification* document as input and produces the *Game Design Specification* document as a work product. We advocate the use of the *Spiral Game Design* strategy as depicted in Figure 9 [Boehm 1986].

Although there are a number of different definitions for digital games, they have one thing in common: they consider games as a kind of system which includes different sets of parts, which are in relationship to each other to create a whole [Juul 2003; Salen and Zimmerman 2004].

Simply the game design defines that system, and it is a process of designing any kind of game components such as content, gameplay, environment, storyline, and characters. The game design determines what choices players will be able to make in the game world and what affects those choices will have on rest of the game. Also, it determines what kind of win or lose criteria/rules the game will include, how the player will interact with the game, what kind of guidelines will be presented to the player. In short, the game design determines every detail of the game [Rouse 2010; Salen and Zimmerman 2004].

After decades of evolution of the games, the design-centered techniques are still limited and none of the design or development techniques can guarantee to provide complete guidelines to create a game design and development [Almeida and Silva 2013; Bethke 2003]. However, a well-designed game undoubtedly has a better chance of being successful than a poorly designed one [Adams 2013].

Also, the process of designing games is strictly related with the creativity of the game designers. During the game design process, a game designer will be responsible for creating a world, an environment, a society, characters, chains of different events and many other game elements which requires the designer to evaluate different possibilities, make decisions and combine different items to provide the best experience to the players [Schell 2008].

“Game design is a complex, multilayered design activity [Salen 2007]” in which different contents, actions and rules are designed and combined to create a dynamic system that provides a meaningful experience for players. This dynamic system generates and presents different contents with respect to the decisions made by players. Defining rules that guide players on how, when and why to interact with these dynamic systems forms the game design practice [Brathwaite and Schreiber 2009; Fullerton 2008; Salen 2007].

Game design is a design of systems, which combines different systems in a way that they work smoothly and flawlessly with each other to provide the best look and feel [Hunicke et al. 2004]. For example, rules are one of the fundamental parts of any game system and designing rules that fit to each other logically and meaningfully is crucial in game design [Dondlinger 2007]. Similarly, the creation of game characters and their corresponding behaviors, the game world, and the interaction between the characters are also important because these behaviors and interactions establish some of the major rules in a game [Salen 2004].

A meaningful gameplay is another important part in game design. Meaningful gameplay is the part, which makes the game successful [Ermi 2007]. A designer might not need elegant graphics; however, you definitely need to have a good gameplay to satisfy the players [Kiili 2005]. A gameplay is “the process by which a player takes action within the designed system of a game and the system responds to the action” [Salen 2007].

Games need to be designed in a way that they should make players feel like they are in charge of the game. The decisions and the corresponding rules that players make should relate with the previous rules and the future ones. Otherwise, it might cause an ambiguity in the game. Players might get confused about what to do and how to react in a given situation. Therefore, chain of activities and rules should be following each other [Adams 2010].

Games also need to have a balanced design. It should not be too easy or too difficult. The game designers should come up with a way to make the game challenging and engaging to provide the right balance in a right way [Kiili 2005; Chen 2007]. For example, the resources in strategy games should be placed strategically by game designers to encourage players to race for it, at the same time it should not be too challenging for players to find and use as needed [Chan et al. 2007]. Similarly, the reward mechanism needs to be designed in a way that it should provide useful feedback and also make sense for players to compete for it. The reward mechanism should foster the racing between players. However, should not be impossible for players to get them [Jakobsson and Sotamaa 2011].

Game design is an interdisciplinary process and it involves collaboration between experts in graphic design (visual design, interface design), programming, animation, interactive design, writing, and audio design. Game designers should know how to speak the “language” in each of these fields, in order to see the possibilities and constraints of their design [Salen 2007].

3.2.3.1 Game Design Team

Generally games are designed and developed by a design team and based on the games’ complexity and the size, the design team can include a number of members. More importantly, depending on the project size some team members can have more than one role. Although there are not standardized team roles in industry, some of the roles can be described based on the common responsibilities as shown below:

1. *Lead Designer:* A project can have only one lead designer and s/he is responsible for keeping the project on track and making sure that it is complete and fully developed. Lead designers do not necessarily need to be creative; however, they have to make sure that the design team is getting the work done.
2. *Game Designer:* The game designer is responsible for defining and documenting how the game actually works. Game designer defines the gameplay and the core game mechanics. Based on the project size there can be more than one game designer.
3. *Level Designer/World Builder:* Level designers get different parts of game components such as core mechanics, gameplay, and the user interface and use them to create levels/worlds. A project can have more than one level/world designer.
4. *User Interface Designer:* User interface designer is required to design the layout of the screen. Depending on the team structure and the game complexity, the team can have a team member who is fully responsible for the user interface.
5. Writer: Writers are responsible for creating the instructional or fictional content of the game such as backstory, instructions, cut scenes.
6. Art Director: The art director is responsible for managing the production of all the visual assets such as sprites, animations, and textures. In addition, they also make decision on creating the visual style of the game.
7. Audio Director: The audio director is responsible for the production of all the audible assets such as background music, effect sound, and dialog between game characters.

3.2.3.2 Spiral Game Design

We advocate the use of the *Spiral Game Design* strategy as depicted in Figure 9 [Boehm 1986]. The Spiral Game Design strategy employs the following principles:

Evolutionary Development: Develop the game design gradually in increasing level of detail as more experience and insight are obtained through prototyping and playtesting.

Incremental Development: Develop the game design in increments by identifying each increment with a version number and with a set of design features under a configuration baseline.

Iterative Refinement: Refine the level of game design granularity in an iterative manner based on experience and insight obtained through prototyping and playtesting.

Progressive Elaboration: Add details to the game design progressively by elaborating (i.e., expanding and embellishing) the game design characteristics based on experience and insight that become available through prototyping and playtesting.

The game design process consists of four major activities:

- Prototyping
- Playtesting
- Evaluation
- Risk analysis

The spiraling depicted in Figure 9 starts with the Game Idea Specification document, based on which the *Prototyping* activity is conducted to produce Game Design Prototype 1. In the next activity *Playtesting*, potential players of the game experiment with the prototype under simulated conditions for the purpose of discovering design flaws, providing insight, and giving feedback. The *Evaluation* activity is conducted to judge the game quality using quality indicators such as the ones presented in Section 2. The *Risk Analysis* activity is conducted to identify potential risks and determine how to mitigate them. Based on the experience and insight gained from playtesting, evaluation, and risk analysis, an improved Game Design Prototype 2 is created. The spiraling, consisting of prototyping-playtesting-evaluation-risk analysis, is repeated to develop Game Design Prototype 3. The spiraling continues until the game design is judged to possess acceptable quality in the Nth spiraling. The number N changes from one game design to another depending on the size and complexity of the game design. The Game Design Prototype N is described in the *Game Design Specification* document as the work product produced by the game design process.

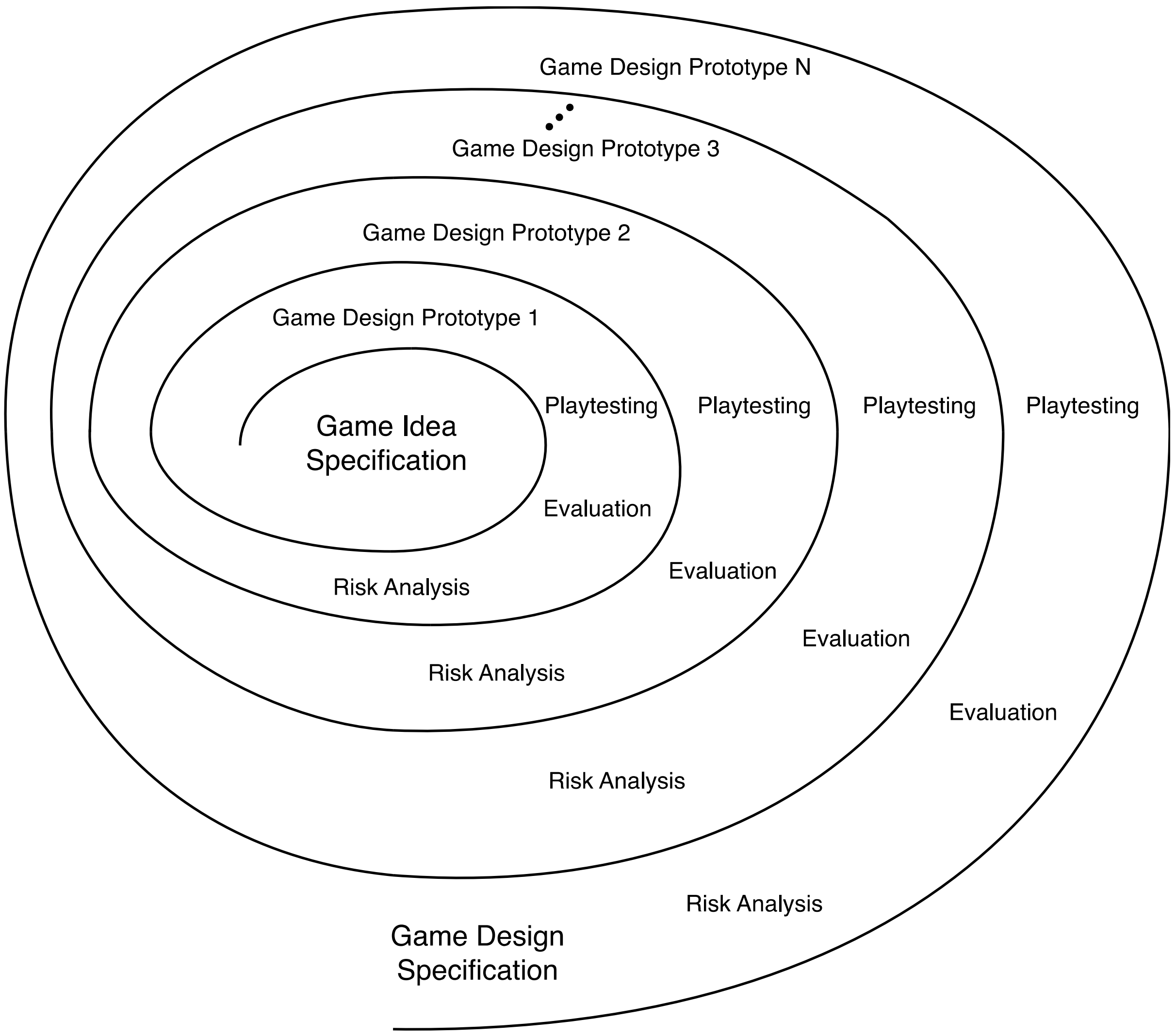


Figure 9. Spiral game design

3.2.3.2.1 Preparation for Game Design

Upon execution of the processes that are listed above (i.e. Prototyping, Playtesting, Evaluation, Risk Analysis), the designers need to elaborate the current game idea specification document to start the game design process. In game idea generation, designers have made fundamentals decisions about the game. At this point, the game design moves into from general to more specific.

In this stage, designers define gameplay modes, characters, game world, core game mechanics, levels, the story, and the flowboard. Designers gradually get into the details and create a working prototype of the game to check how the game ideas/features actually look and feel in a real game environment.

1. *Gameplay Modes:* After a successful execution of the *game idea generation* process, the designers' first duty is defining the primary *gameplay mode*. The *gameplay mode* refers to a mode in which the player spends most of his/her time [Adams 2009]. The *gameplay mode* consists of three elements [Adams 2004]:

a. Perspective: The game world such as 2D, or 3D space
b. Interaction model: The way character interacts in the game world, such as avatar-based model
c. Gameplay itself: The different types of challenges and players

These three elements form a *gameplay mode*, and a significant change in any of these elements changes the *gameplay mode* [Adams 2004]. Although most games, such as Tetris, have only one primary gameplay mode, current games, which are more complicated, have more modes. For example, in a simple racing game, driving the car is the primary gameplay mode; however, tuning up the car is secondary gameplay mode or the popular MineCraft has many gameplay modes such as survival, creative, adventure, and hardcore gameplay modes [Mojang 2013].

2. *Game Characters:* In order to create an immersive game story, a game must have well-designed, believable, and realistic characters to drive its players [Barnhardt 2011]. If a designer can create deep and lovely characters, the players might connect with them, which can affect the success of the game. Therefore, the goal of character design should be creating characters that players can truly engage and love [Adams 2009]. In addition, a designer must think about how the character looks and behaves: what actions she is capable of, what emotions her face and body language can register, and what kind of language and vocabulary s/he uses. Figure 10 represents the look of President Carmelo character in the Candy Factory Educational Game for iPad [LTRG 2014] Moreover, depending on the audience, the game should allow character customization and provide male and female characters.

   Characters can be designed in two ways: [Adams 2009; Mitchell 2012; Rollings and Adams 2003]

   a. The *art–visual driven character design* focuses into the visual look of the character. Characters' appearance is designed first and then the story is implemented to the character.
   b. The *story-driven character design* focuses on defining the personality and the behaviors of the character rather than the look. Every aspect of the character is defined first, and then the character is created visually.

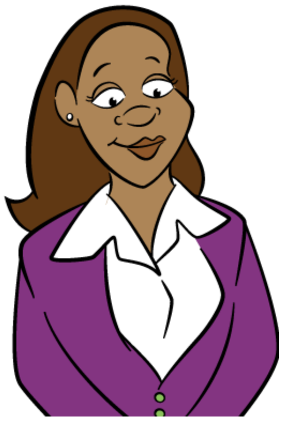


Figure 10. President Carmelo in CandyFactory Educational Game for iPad [LTRG 2014a]

3. *Game World:* A game world is an artificial environment that hosts the gameplay. In many cases, a game world is created based on imagination. Depending on the game, a game world can have an artificial or inartificial environment. For example, a football game requires a realistic game field; on the other hand, a space game might require different artificial game elements to form an environment. Regardless of the environment, the main purpose of a game world should be entertaining the target audience [Adams 2009].

During the design process of the game world, designers' main focus should be keeping the game world simple and elegant, which fits to the game story and the other game elements such as characters. In addition, the design decision that designers make should be consistent. For example, the game world should not break any of the game logic such as physics of the game. If the game world is visually and logically consistent, this will allow players to have smooth transitions between different play areas without any distraction.

Game worlds can be created in different physical shapes, which are 2D, 2.5D and 3D. *2D game world design* refers to a flat design as it can ben seen in Figure 11. *2.5D game world design* refers to a design that works like a two-dimensional game, but it allows a limited control of Z-axis. In order to create 2.5D game worlds, the designers can either use 2D graphics to give the feeling of 3D look or use real 3D objects and limit the game world in two-dimensions. *3D game world design* refers to a design that is created in three-dimensional space. This approach uses three-dimensional representation of geometric data and it is commonly used by many popular games [Watt and Policarpo 2003].

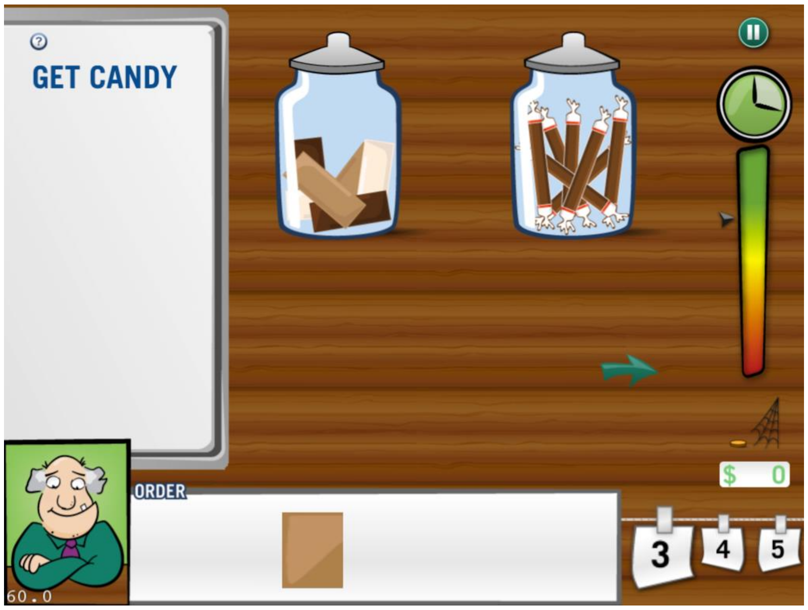


Figure 11. 2D game world of CandyFactory Educational Game for iPad [LTRG 2014a]

4. *Core Game Mechanics:* Core game mechanics is an interaction that occurs most frequently in a game. For example, the core game mechanic in a shooter game is firing the weapon [Kim 2012]. After determining the gameplay mode, designers can start thinking about how challenges and actions occur in gameplay mode and how core game mechanics forms them. For example, if the game uses a sports theme to teach mathematical concepts, designers should consider specific topics related to that sport, such as acceleration, strength, and accuracy as well as the intersection between that specific sport and the mathematical concept.

5. *Game Levels:* Designing levels requires game designers to use all different game elements such as characters, game world, core mechanics, and challenges to create an experience that a game offers to its players. During this process, although any of the game elements does not need to be completed, there has to be enough elements that a level designer can work with. In the early stages of the game development process, the game designers and the level designers should focus on constructing the first playable level, rather than spending time on the least valuable game parts such as the opening

screen [Adams 2009; Mitchell 2012]. In addition, game designers should always keep in mind that the constructed levels should challenge, entertain, and immerse game players [Ryan 1999].

6. *Additional Modes:* After successful creation of the levels with all the necessary game elements, the game designers should focus on other additional modes, which are not primary. The creation of additional modes can happen naturally while the designers are constructing new game features. For example, if a racing game enables players to upgrade their cars, then the game should provide an additional gameplay mode in which players can upgrade the different car parts.

   Designers should keep in mind that additional modes should be carefully planned. Any extra modes will require extra graphics, programming efforts, and more testing. Although it might bring new features and capabilities to the game, the designers should consider the feasibility of the additional modes before its' execution [Adams 2004; Adams 2009].

7. *Writing the Story:* Stories are one of the most essential parts of the game. Story keeps the player interested and immersed in to the gameplay; it motivates players to keep moving to the next level. Designers can implement stories in different ways. For example, transitional scenes can be presented between different levels to show a successful completion of a level or during the game play with the core game mechanics. In addition, a story can be linear or non-linear based on players' choices [McDevitt 2010].

8. *Flowboard:* Once designers have all necessary pieces such as gameplay mode, game world, characters, levels, and core game mechanics, they should structure the game. Although storyboards and flowcharts are used in game design, we advocate the use of *Flowboard* approach.

   A flowboard is a combination of a storyboard and a flowchart, which documents the structure of game progression. Unlike the storyboards, the flowboard is non-linear which enables designers to transition in any direction. Flowboard can consist of many scenes, and each scene should have the related sketch or mockup, the brief description of that specific scene, the mode name, the challenges, the actions, and the interaction model. Scenes are connected to each other with an arrow, and arrows include text that indicates when and why to make transitions between scenes [Adams 2009].

   Flowboard can be created in different ways. For example, Microsoft Visio software [Microsoft 2014] or regular papers on a large wall can be used. Designers start with title of the scene that indicates the mode of the game. After that, designers draw the sketches in the middle of the paper/screen that shows the perspective of the camera. Then, designers add menu and user interface items and document them. Next, the designers list challenges and the players' actions for that specific part. Lastly, the designers add arrows and the circumstances that cause transitions between the scenes [Adams 2004].

9. *Game Design Specification Document:* After generating the flowboard and all the necessary details of a game, a designer should create the game design specification document (GDSD). Once a designer creates the GDSD, s/he should constantly update and maintain it throughout the whole game development process.

   GDSD is a blueprint that describes every aspect of the game [Mitchell 2012]. The game elements that have been listed above (i.e. game world, game characters, levels, and game play modes) all goes into the GDSD. GDSD can have different meanings for different team members. For designers, it can provide a detailed description of the game functionalities, and for graphic artists and programmers, it can provide guidelines [Ryan 1999].

   Designers should keep in mind that GDSD does not remove the need for design meetings, and discussions in electronic environments. For example, if the design team is geographically dispersed, chatting rooms or video conferencing can be used to hold meetings [Ryan 1999].

Although GDSD provides many advantages to game development process, there isn’t a consensus on what exactly forms a valid GDSD. Each game company creates their own style of GDSD and it can hold many different subsections, which is mainly caused by the significant differences between games [Baldwin 2005; Bartle 2004; Rouse 2004; Rogers 2010; Salazar et al. 2012].

Nevertheless, there is still a set of common elements that can be used to define a structure of GDSD. These elements include overview, mechanics, dynamics, aesthetics, and limitations [Salazar et al. 2012].

a. *Overview*: The *overview* section summarizes the key features of the game. It guides and keeps the designers and the developers on track. The overview section should be concise and direct. Readers should be able to capture the main idea and fundamentals of the game without getting into specific details. Moreover, it can also include the goals and the objectives of a game [Adams 2013; Rogers 2010; Salazar et al. 2012].
b. *Mechanics*: The *mechanics* in game literature describe the game elements and the interaction between them. For example, character or challenge lists, interfaces, levels, and artificial intelligence can be provided in this section. Depending on the game’s complexity and the designer’s need each of these game elements can also be defined in a combined section [Bates 2004; Rogers 2010; Salazar et al. 2012; Schell 2008].
c. *Aesthetics*: The *aesthetics* refer to the graphics and the audio elements that are perceived by the game players. This section of the document should provide all the required graphics assets and sound elements [Baldwin 2005; Salazar et al. 2012; Salen and Zimmerman 2004].
d. *Limitations*: The *limitations* refer to any constrains that effect the game design and the development decisions. For example, the hardware of a game console, the screen size, and the control mechanisms (e.g. touch screen) are some common technical limitations that can affect the game design and development decisions [Baldwin 2005; Bates 2004; Rogers 2010; Salazar et al. 2012].

3.2.3.2.2 Prototyping

Prototyping is conducted to develop a prototype of a game design to enable potential players of the game experiment with the expected game functionality under simulated conditions for the purpose of discovering design flaws, providing insight, and giving feedback [Fullerton 2008; Salen 2004; Schell 2008; Snyder 2003].

The overall purpose of prototyping is to understand the game features and enhance the game play experience. It provides an invaluable opportunity to the game designers to get a better understanding of the implemented idea and later enable them to improve it [Brathwaite and Schreiber 2009; Iacucci et al. 2000; Manker 2011]. During the prototyping process, specifically in early stages, the designer should only focus on the fundamentals of the game such as gameplay and the game mechanics, not the graphics or other game components.

The nature of prototyping requires failing. The design team should be ready to fail and throw a prototype away at any time. Since the focus of a prototyping process is exploring new ideas and finding solutions to specific design problems, prototyping should embrace the failing in design process [Gray et al. 2005].

Prototyping enables game designers to validate their ideas and avoid drastic changes by way of early detection in the game design process [Salen and Zimmerman 2004]. Prototypes in game design reflect an implementation of an idea in a game environment to facilitate a design problem and its solution [Fällman

2003]. In the early process of game design, prototypes can act like a sketching tool [Agustin et al. 2007; Manker and Arvola 2011].

A game design prototype (a) represents the game's user interface to enable a player to interact with the game in a simulated manner, (b) is built rapidly and cheaply, (c) must be easily modifiable, and (d) facilitates communication between the game designers and potential players. Two major types of game design prototyping exist: [Fullerton et al. 2006; Schell 2008]

- Physical Prototypes
- Digital Prototypes

3.2.3.2.2.1 Physical Prototypes

Physical prototypes are the easiest type of prototypes to construct by game designers. They are generally created using very common materials such as paper, cardboard and household objects [Agustin et al. 2007; Fullerton 2008].

The simplicity of this approach enables game designers to focus on the gameplay and the game mechanics rather than the aesthetics and the technology. Also, it enables game designers to make immediate changes based on the players' feedback, which allows more iteration. Moreover, they also enable non-technical team members to contribute in the game design process very easily because the approach does not require any specialized knowledge or skills [Snyder 2003; Salen 2004; Fullerton 2008].

At the beginning of the prototyping process the focus should be on the game mechanics. The designer or the design team should avoid spending too much time on the artwork. The prototype should be as simple as possible such as, simple drawings or objects, which focus to satisfy only a specific feature of the game. Spending time on the artwork might cause game designers to get attach to their work, which can degrade the iterative prototyping process and the creativity [Adams 2012; Fullerton 2008]. Figure 12 shows the paper prototype for Candy Factory Educational Game for iPad.

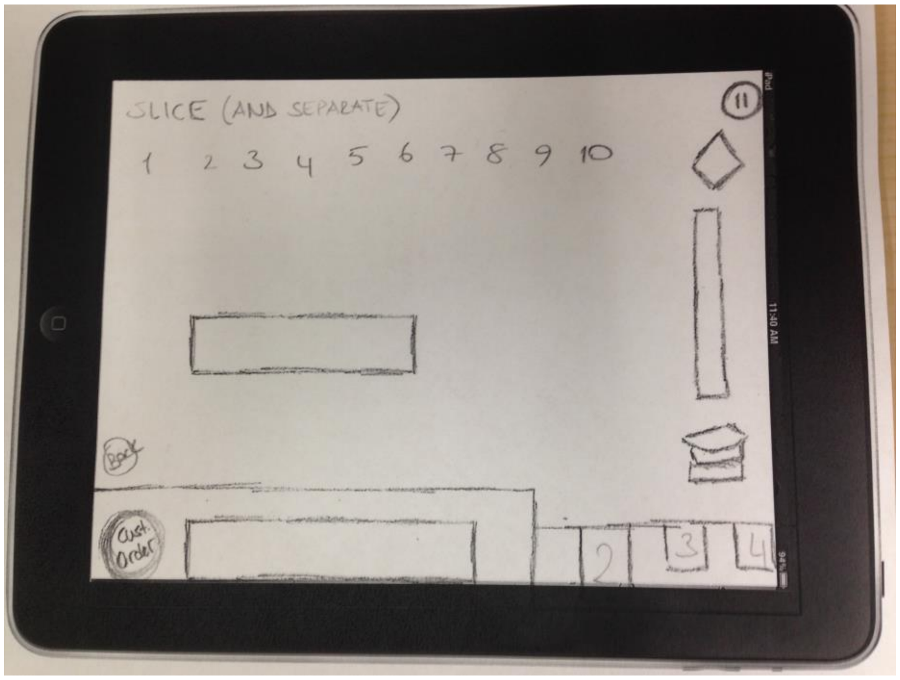


Figure 12. CandyFactory slicing operation

There are four steps that can help designers to create physical prototypes, which are foundation, structures, formal details, and refinement.

1. *Foundation:* Build a representation of the core game play with papers, plastic materials, glue, pen and so on. Use a pen to draw the game world, cut some papers to create the game characters, and glue any of the necessary game elements to form the game world. In this step, designers' only goal is trying to setup the *foundation* of the game in which a designer can play the core game mechanics. After that play with the game and deepen the understanding on the core game mechanics. Check to see if the core gameplay is working in an expected way. During the execution of this process if any ideas come to your mind put them in a list and avoid adding a new feature unless it is required by the core gameplay [Fullerton 2008].
2. *Structure:* This step requires game designers to identify the required rules and the features. Designers should be aware of the difference between the rules and the features. A *rule* is a game mechanic, which can change the functionality of the game, on the other hand a *feature* is an attribute, which makes the game experience richer. Designers can have a rule without having a feature; however, they cannot have a feature without a rule. Rules can be applied to the existing features; on the other hand, features need to be associated with rules to logically make sense in the game [Fullerton 2008]. For example, if a designer wants to incorporate with a scoring feature in the game, s/he must have a rule for that scoring mechanism such as in which condition the score will go up, and also how many points will be added to the current score.
3. *Formal Details:* The goal of *formal details* is to add all necessary rules and features to make the game fully functional. In this step, the designers should focus on identifying the game elements, which are appropriate for the game.

   One of the ways of identifying right features and rules is the isolating each rule and testing it individually. For example, the designers can add a feature and test it, then remove it and repeat the process. This will enable designers to clearly see which rules are necessary and which are not [Fullerton 2008].

4. *Refinement:* This step requires designers to focus on small details rather than the fundamentals of the game. The designer can add new features to make the game as rich as possible; however, the new game elements should be added and tested gradually. Adding new features and rules, then testing them will enable designers to see which game elements are more valuable than others [Fullerton 2008].

3.2.3.2.2.2 Digital Prototypes

Although physical prototypes can help us in so many ways to improve the game design, they have limitations. A game designer cannot validate all of his/her ideas with a physical prototype. To reach the ultimate game design, a game needs to be prototyped in a digital environment where a game designer can test features like special game physics and different interactions styles [Adams 2010; Salen 2004].

Similar to the physical prototypes, digital prototypes are not completed final designs either. The main focus is testing some features, such as game logic, special physics, environments, levels, and control systems. Generally, digital prototypes are developed with the least necessary game objects such as art and sound, which are not finalized. Furthermore, digital prototypes should try to answer design questions or clarify some of the unexplored technological features [Adams 2010; Fullerton 2008; Rouse 2010]. Figure 13 shows a digital prototype for Candy Factory Educational Game.

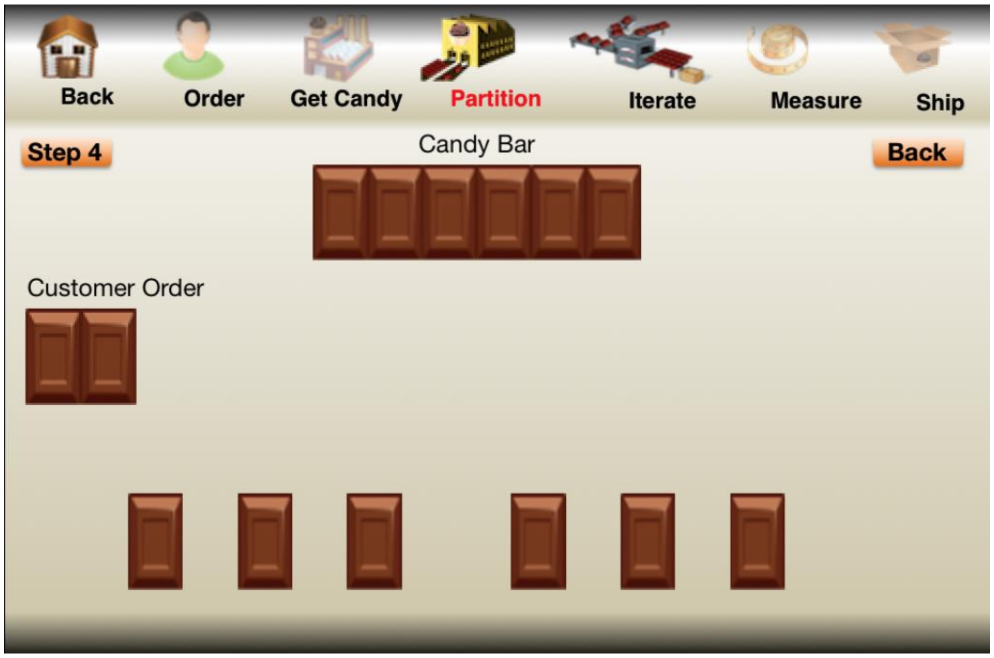


Figure 13. CandyFactory Educational Game [Aslan 2011]

Digital prototypes can focus on four different aspects of a game as follows:

1. *Prototyping Game Mechanics:* Game designers should start prototyping with core game mechanics and assure that they work as they are expected. Unless the core game mechanics are engaging and satisfying, designers should not proceed to the next step [Fullerton 2008].
2. *Prototyping Technology:* Before making a final decision about the technology, the developers should test, debug and validate the tools that will be used during the design and development. This

enables designers and developers to see any possible obstacles in the process and prove the functionality of ideas and concepts before beginning the final project [Fullerton 2008].

3. *Prototyping Aesthetics:* Working on the art and sound can articulate the game mechanics and can give designers an overall idea of how some of the key features will look like when the game is finalized. Designers need to assure that all of the game elements fit together and work as expected. Storyboards, concept arts, an animatic, an interface prototype, and audio sketches are some of the tools that can be used for prototyping on aesthetic game elements [Fullerton 2008].
4. *Prototyping Kinesthetic:* One of the main challenges in game design is satisfying the player throughout the whole gameplay. Designers need to make sure that the interaction between the game and the game player works as smoothly and flawlessly as possible. This can be achieved by prototyping the kinesthetic aspects of the game that concern the control mechanisms, interfaces, and the responsiveness of the game. For example, a game that is being developed for a console platform, designer can have a separate controller. On the other hand, on a touch screen device they have to define the interactions based on the device's touch capability.

   For example, although the designers might have a successful engaging game that uses different button combinations for the game play on a console platform, they might not have the same success on the touch screen device because of the controller limitations. Therefore, kinesthetic aspects should always be kept in mind and the game should be digitally prototype to test all different ways of interactions [Fullerton 2008].

3.2.3.2.3 Playtesting

Playtesting refers to the activity in which potential users experiment with a prototype of a game design under development for the purpose of discovering design flaws, providing insight, and giving feedback. The prototype can be a paper or software prototype.

Confirming the games' usability, accessibility, and engageability are some of the key elements that a game designer needs to make sure before the game is released to the public [Isbister and Schaffer 2008; Yuan et al. 2010]. As a designer it is really challenging to come up with a complete and successful game design [Adams 2010; Sánchez 2009]. Therefore, receiving players' feedback and improving the game design is one of the most important ways of reaching the projected success for the game.

Playtesting is an iterative process and requires designers to keep evaluating and revising the design of the game continuously [Fullerton 2004; Fullerton 2008; Keith 2010; Law 2012; Salen and Zimmerman 2006]. The playtesting process should start early in the design process and continue to the release of the game. Also, in some cases it can even continue after the public release [Ambinder 2009]. In the early stages of the game design, the playtesting process can take more time to process, which mainly focuses on the fundamental aspects of the game design such as core gameplay. Toward to the end, it should get smaller and focus on the minor issues [Fullerton 2008].

Playtesting is conducted under the following three steps:

1. *Preparation for playtesting:* This step consists of the following tasks:
   a. *Define Goals for Playtesting:* Playtesting is conducted in a goal-directed manner. All goals should be clearly and explicitly specified. The goals are the core of the playtesting sessions. Ambiguous goals can waste the time and effort of the design team [Niedenthal 2007]. For example, "identifying the collision errors between the game elements in fighting arena" can be a goal for playtesting [Wolters 2012].

b. *Select a Playtesting Method:* A playtesting method such as group testing, individual testing, interviews, feedback forums, guided, or unguided is selected. Questions should be created in a way that they cannot be answered by only “Yes “or “No” answers. One commonly used approach is using "who-what-where-when-how" questions, which would enable designers to get more valuable answers then a short "Yes" or "No" response [Wolters 2012].

   Unless the designers ask the right question in the right way, they will not receive the most valuable feedback from the testers. Therefore, preparing well structured, carefully planned, simple, clear, and right to-the-point questions are important [Fullerton 2008; John 2013]. For example, if the designers have testers who have never played a digital game before, they should avoid using any specific gaming terms in questions that the testers might not know. It might cause them to give irrelevant answers.

   Moreover, designers should avoid creating unnecessary questions. Asking too many questions can make playtesters feel tired and decrease the quality of their answers. Therefore, designers should focus on generating high quality questions rather than creating many of them.

c. *Develop a Procedure for Playtesting:* A playtesting procedure is developed to describe how the playtesting will be performed systematically and what test data and simulated conditions will be employed. There are different ways of setting up a playtesting session and depending on the game, the objectives, and the structure of the playtesters designers can choose either one or combine different methods. Some of the playtesting methods are one-to-one testing, group testing, feedback forums, interviews, open discussions, data hooks, physiological measurements and automated testing.

   Moreover, depending on the playtesting objectives, designers might use guided or unguided playtesting sessions. Guided playtesting requires game designers to explain as many game elements as possible; unguided playtesting requires the game designers to explain as little information as possible upon execution of the playtesting session [Rouse 2010; Wolters 2012].

d. *Recruit People for Playtesting:* Qualified people with multifaceted knowledge are recruited to perform playtesting. Depending on the game, they might need different test groups who can serve different needs [Fullerton 2008; John 2013; Isbister and Schaffer 2008]. For example, during early prototyping, designers can test the game by themselves to verify if the game is missing anything fundamental features.

   Although having close friends, strangers as testers are good options; the ultimate solution is having testers from the target audience [Fullerton 2008; Rouse 2010; Sykes 2006]. In reality, the target audience can provide the most valuable feedback because they are the group of people who would be spending most of their time with the game. Their feelings, the level of engagement, any negative and positive feedback can point designers to the right direction to make the right design decisions.

   It is also important to consider the prior gaming experience that the playtesters have. A player who has a gaming experience can provide different feedback than a player who lacks of such experience [Gerling and Masuch 2011].

2. *Execution of playtesting:* Playtesting is executed according to the steps of the *Playtesting Procedure* in a controlled environment where players are observed while playing the game and observations and notes are recorded concurrently.

   The designer should have *test script* in which detailed testing guidelines are described. A *test script* enables designers to avoid unnecessary interactions and dialogues with the testers and forces them to act as an objective observer instead of being someone who defends his/her works’ missing elements or the flaws in the game design [Fullerton 2008; Trefry 2010]. Some of the required items that need to be addressed in the *test script* are *introduction, wrap-up discussions, play session, discussion of game experience, and wrap-up* [Fullerton 2008]:

3. *Documenting playtesting results:* Recorded videos of playtesting, observations collected, notes taken, and any other recordings are all documented.

3.2.3.2.4 Evaluation

The Evaluation activity is conducted by analyzing the documented playtesting results and judging the game design quality. Each of the game quality indicators such as the ones given in Section 3.1 is evaluated based on the documented playtesting results. The evaluation aims to identify game design flaws. Based on the evaluation results, the game design is improved and a new game design prototype is created to reflect the improvements.

3.2.3.2.5 Risk Analysis

Risk analysis is executed as part of the *risk management* process and it is a step in which the team considers each previously identified risks and estimates the probability of that risk [CMMI 2010; Pressman 2010; Sommerville 2007]. "A risk is a probability that some adverse circumstance will occur" [Pressman 2010] and the purpose of *risk management* is identifying potential problems and transforming them into distinct, concrete risks that are clearly documented, analyzed and identified possible solutions [Alberts and Dorofee 2010; CMMI 2010; Pressman 2010; Sommerville 2007].

Developing digital game is a complex process that includes people, technology and different forms of interactivity [Salen and Zimmerman 2004]. The complexity of the game development requires careful planning and risk management because of the high possibility of failure such as, a small mistake in the game design might cause developers to recreate many of the game components or the developed game might not serve to the objectives. Hence, the proper application of risk management can help you to reduce the development risks [Carpenter 2003].

A game development project management is expected to include proactive risk management, which consists of risk identification, risk analysis, risk planning, and risk monitoring. Risks, including the ones described below, should be (a) identified, (b) prioritized in terms of their estimated probability of occurrence and consequences, (c) monitored during the project, and (d) mitigated under a plan [Pressman and Maxim 2015; Sommerville 2011].

1. *Risk Identification:* Effective risk management includes proactive *risk identification* through collaboration. Identifying potential issues that could negatively affect the game design and the development process, form the basis for a successful risk management. Therefore, risks should be identified and described as clearly as possible before risk analysis stage is executed. All identified risks should be reviewed periodically to reexamine possible sources of risk and changing conditions [CMMI 2010]. More importantly, the design team should be aware that, "*risk identification* focuses on the identification of risks, not the placement of blame" [CMMI 2010].

   Some of the possible and common game design and development risks can be grouped as follows:

   a. *Product Risk:* the probability that the game's characteristics will be unsatisfactory.
   b. *Acceptance Risk:* the probability that the game will not possess sufficient *Acceptability*, the degree to which the game fulfills its requirements and learning objectives.
   c. *Integration Risk:* the probability that the game modules will not be successfully integrated.
   d. *Performance Risk:* the probability that the game will be too slow to play.

   e. *Usability Risk:* the probability that the game will not be easily employed for its intended use.
   f. *Supportability Risk:* the probability that the game cannot be properly maintained after its delivery.
   g. *Utility Risk:* the probability that the game will be less useful than required by its intended users.
   h. *Resource Risk:* the probability that the game development will exceed the allocated resources such as budget and time.
   i. *Cost Risk:* the probability that the game development will exceed the budgeted amount.
   j. *Schedule Risk:* the probability that the game will not be delivered by the required deadline.
   k. *Technology Risk:* the probability that the software and hardware technologies used in developing the game will be unsatisfactory.
   l. *People Risk:* the probability that some adverse circumstances will occur due to the game development team members.

2. *Risk Analysis: Risk analysis* requires team members to use their own judgments to make estimation on the possibility and the seriousness of the possible risks. Since it is not possible for the designers to make a precise assessment based on the judgments, the expected estimation should provide a range. For example, a risk can be defined as very low (<10%), low (10-25%), moderate (25-50%), high (50-75%), or very high (>75%) [Sommerville 2007]. During the risk analysis, the design team can follow steps that are shown below to execute the risk analysis stage [Pressman 2010].
   a. Create a scaling system that represents the possibility of the risks.
   b. Identify the possible consequences of the risks.
   c. Evaluate the impact of the risk on the game.
   d. Assess the overall accuracy of the risk for clarification in the team.

   During early stages of the game design process, some of the risks can be defined as general. After couple of iterations the design team should break up into smaller risks that can be easily manageable [Pressman 2010].

3. *Risk Planning: Risk planning* step requires the team to examine previously identified and analyzed risks. This step results with plans and strategies, which targets minimizing risks when they occur. The team should be aware that there is no silver bullet to come up with the best plan to minimize risks; instead *risk planning* relies on the team members' experience and judgments [Pressman 2010; Sommerville 2007]. Risk plans are developed and implemented for selected risks to proactively reduce the potential impact of risk occurrence [CMMI 2010].

   Some of the possible risk management strategies are [Sommerville 2007]:
   a. *Avoidance strategies,* which reduce the probability of the risks.
   b. *Minimization strategies,* which reduce the impact of the possible risks.
   c. *Contingency plans,* which prepare designers for the worst case and provide a strategy in place to deal with risks.

   As a rule of thumb, the team should target to avoid risks as much as possible. If it's not possible to avoid, they should have plans to minimize the effects of those risks. Finally, they should have strategies and plans to cope with those risks if they occur.

4. *Risk Monitoring:* Risk monitoring is a step in which the team checks the assumptions about the possible risks identified. For example, designers examine if the probability of the risk is getting higher or not, or the effect of the risk is being reduced, or increased. Depending on the projects and

also in different stages of the development and design process, the possible factors that can affect those risks might change. Therefore, the design team should regularly monitor risks and effective plans should be developed based on the current state of the risk. For example, if the current game design is inadequate to satisfy learning objectives and not increase the engagement on a specific topic, the team should decide whether to change the current design or start a new one from scratch [CMMI 2010; Pressman 2010; Sommerville 2007].

3.2.4 Requirements Development

Requirements development is the process of elicitation of requirements based on the game design specification document in an authoritative manner. This process takes the game design specification document as input and produces a requirements specification document as the output work product. This stage is also known as transitioning between pre-production (game design) to production phase (software development) in digital games [Callele et al. 2005].

The requirements for a game are the descriptions of how the game should work - the game features, the gameplay, the rules, the interactions and the limitations that are provided by the game. Moreover, these requirements aim to satisfy the target audiences' needs such as fun and engagement [Hammersly 2009A; Hammersly 2009B; Pressman 2009; Sommerville 2011].

Requirements are classified into two major categories:

- Functional Requirements
- Non-Functional Requirements

Functional Requirements (a) are statements of game features that the game should provide, (b) describe how the game should react to particular inputs, (c) dictate how the game should behave in particular situations, (d) describe the game's functionality, and (e) define the behavior and input-output transformations of the game [Pressman 2009; Sommerville 2011].

Non-functional Requirements are the ones that are unrelated to game functionality. For example, they describe qualities or properties that the game must have, such as: [Pressman 2009; Sommerville 2011]

- *Usability* is the ease of use and learnability of a game object
- *Interoperability* is the ability of a system to work with other systems
- *Privacy* is the ability of a system to protect the privacy of its users
- *Portability* is the usability of the same system in different environments

Depending on the project needs, requirements can be defined as either high-level or detailed. This difference can serve to the different readers. For example, a manager can be only interested on the high-level requirements; on the other hand, developers might be interested on detailed requirements [Pressman 2009; Sommerville 2011].

During requirements development, the software engineers collaborate with the game designers and carefully examine each scene in the flowboard to generate use-cases. Later, each of these use-cases is used to create associated functional and non-functional requirements.

Identifying the functional requirements is a challenging process for games. For example, the GDSD might contain many ambiguous game features, which can lead to incorrect requirements [Callele et al. 2005]. Also, it is a common issue that game projects suffer from unrealistic scope and feature creep. In many games, new requirements emerge in later development cycles. For example, managers could ask for extra features during beta testing, which can delay or postpone your release date [Callele et al. 2005; Morgan et al. 2009].

Requirements errors are also known to be one of the costliest errors to fix during development of a software project. Fixing an error after a release of a software product is 100 times costlier than fixing it at the start of the project [Boehm and Basili 2001].

Also, it can be challenging to create some key aspects in digital games. For example, the emotional aspects of the game, such as fun and engagement, cannot be easily satisfied by defining couple of requirements. These features generally require many trials and errors to identify the necessary requirements [Callele et al. 2006].

Therefore, an effective requirements development process can save many resources such as time, money, and effort by focusing on the right requirements, which will satisfy the target audience. This can be done with use case-based requirements development, which is known as the best practice to generate requirements.

3.2.4.1 Use Case-based Requirements Development

Use case-based requirements development is considered the best practice to generate functional requirements because of the following benefits:

- A use case represents a small amount of work the software system is required to perform. Thus, decomposing complex game software system functionality into use cases enables the modularization needed to overcome the complexity.
- Identifying the "real" functional requirements is always challenging for complex software systems. A use case describes an interaction, and based on that description, "real" functional requirements can be more successfully identified and associated with that use case.
- Listing requirements one after the other, even in different categories, does not provide any help for transitioning from requirements to software system design. On the other hand, Use Cases turn themselves into classes in an object-oriented design and significantly facilitate the transition.
- Associating functional requirements with a Use Case enables better traceability:
    - Requirement ⇔ use case ⇔ class in design ⇔ class in code

A Use-case "is a description of a set of sequences of actions, including variants, which a system performs to yield an observable result of value to an actor" [Booch et al. 1999]. It specifies the behavior of a system or a part of a system.

Use cases are the fundamental features of the unified modeling language (UML) [Rosenberg 1999]. In their simplest form, they tell us a story about how an actor interacts with the system under a specific set of circumstances. They provide information describing the interaction with the system and this information can be supplemented in a text or graphical format [Sommerville 2010].

The first step in writing a use case is to define the set of "actors" that will be involved in the story. An Actor represents a coherent set of roles that users play when interacting with use case. It is important to realize that an actor and player are not necessarily the same thing. A player may play a number of different roles when interacting with a game, whereas an actor represents a class of external entities that play just one role in the context of the use case [Pressman 2011].

An actor can be:

- A player (human)
- A game component
- A subsystem of a game
- A subsystem of another game
- Another game as a whole

Once the actors and the actions have been identified, use case can be developed. Figure 14 shows a use-case diagram for a CandyFactory Educational Game for iPad. Stick figures represent the actors and each class of interaction is named in an ellipse. Lines link the actors with the interaction. Optionally, arrowheads can be added to lines to show how the interaction is initiated.

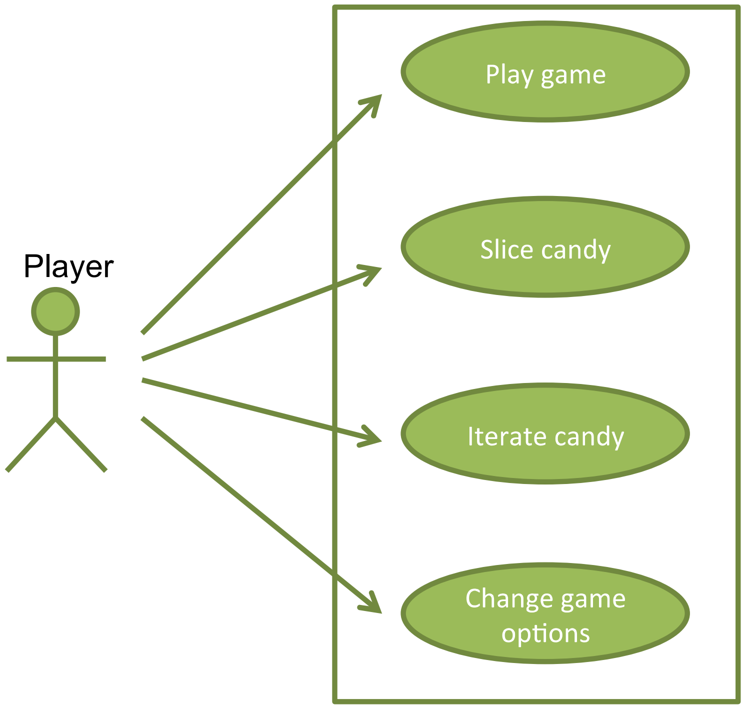


Figure 14. A use case diagram for CandyFactory Educational Game for iPad

A use case includes scenarios that are specific sequence of actions that illustrate behaviors. Scenarios are to use-case as instances are to classes, meaning that a scenario is basically one instance of a use-case. Software engineers can use the information gained from scenarios to form the actual game requirements [Pressman 2009].

For Example: The use-case “Slice a Candy”

Scenario 1: slice a candy into three equal parts
Scenario 2: slice a candy into three unequal parts
Scenario 3: slice a candy with a number pad
Scenario 4: slice a candy with swipe gestures.

Each of the above variants can be expressed in a different sequence of actions, called Scenario.

After a successful creation of a use-case diagram, each use case should be documented with a textual description, which later can be linked to other models in the UML. To document a use-case diagram we advocate the use of *use-case documentation template* as shown in Table 3.

Table 3. A use-case documentation template

Use Case Documentation Template	
Use Case ID:	<Each Use Case will be assigned a unique identifier>
Use Case Name:	<Use case name and a brief description; usually a paragraph or less.>
Actors:	
<A list of the actors who communicate with this use case.>	
Preconditions:	
<A list of conditions that must be true before the use case starts. Precondition describes the state the system must be in at the start of the entire use case.>	
Flow of Events of the Primary Scenario:	
<Most probable flow of events defining the main or primary scenario. The flow of events can be represented using a communicative form such as an activity diagram, sequence diagram, or structured English.>	
Flow of Events of the Alternative Scenarios:	
<An Alternative Scenario is one that allows a different sequence of events than what was described for the primary scenario. The flow of events can be represented using a communicative form such as an activity diagram, sequence diagram, or structured English.>	
Flow of Events of the Exception Scenarios:	
<An Exception Scenario is one where an error, an interrupt, or an exception is handled.>	
Extension Points:	
<If the use case has extension points, list them here. Extends are used when you have an optional sequence of events you want to include in a use case.>	
“Used” Use Cases:	
<If the use case uses/includes other use cases, list them here.>	
Postconditions:	
< A list of conditions that must be true when the use case ends, regardless of whether the primary scenario or an alternative scenario is executed. Execution of an exception scenario indicates errors that must be corrected so that either the primary scenario or an alternative scenario execution can be completed. Postcondition describes the state the system must be in at the end of the use case.>	

Table 4 shows an example of use-case documentation template for “slicing candy operation in CandyFactory Educational Game for iPad in Level 1”.

Table 4. An example of use-case documentation for CandyFactory Educational Game for iPad

Use Case Documentation Template	
Use Case ID:	UC001
Use Case Name:	Slicing candy operation in level 1
Actors:	
The player	
Preconditions:	
The player shall be playing the Level 1. The player shall be in the “Slice” operation.	
Flow of Events of the Primary Scenario:	
1. The use case starts when the player chooses a number from the number pad. 2. The player chooses any number from the number pad. 3. The player examines if the number of equal parts are matching the expected candy size that s/he desires. 4. While the player does not tap the “Home” button or “Shift log” from the “Pause Menu” loop If the player taps the “Forward” button, the game goes to the “Copy” screen Else if the player taps the “Back” button, the game goes to the “Select a candy” screen Else if the player taps another number from the number pad, the game repeats the steps starting from number 3. 5. End Loop 6. The use case ends.	
Flow of Events of the Alternative Scenarios:	
None	
Flow of Events of the Exception Scenarios:	
None	
Extension Points:	
None	
“Used” Use Cases:	
Choose a number from the number pad Go to the “Copy” screen Go to the “Choose candy” screen	
Postconditions:	
A function or Exit is selected on the “Slice” screen.	

3.2.4.2 Requirements Identification

The first step in identification of requirements stage is modularizing the complex game design specification into a number of use cases. The GDSD includes many game components such as rules, interactions, stories, and flowboards. In this stage, the software engineers and the game designers should carefully examine the related game components and modularize them into small different parts. The modularization should focus on separating and extracting functionality of a game into independent parts such that each module focuses on only one or a small number of interactions. This will enable software engineers to create more substantial use cases that will lead them to generate better requirements.

Second, the software engineers should be aware that each created use case should be developed in a way that, it describes an interaction between the actor and the game component or more than one game component. Use cases should tell a stylized story about how an actor interacts with the game under a specific set of circumstances. Also, an actor does not necessarily need to be only a player; it can also be a game component. Moreover, a stylized story should be provided within a use-case template that is shown at Table 3.

Next, the software engineers should examine the interaction described by a use case, based on the sequences of events represented in the primary and secondary scenarios. Scenarios help software engineers to understand how interactions are being used by different game players. Each scenario should usually cover one, or a small number of possible interactions. Also, different type of scenarios could provide different types of information at different levels of details about the game. Thus, software engineers should generate alternative scenarios, which may lead them to better requirements.

Finally, for each sequence of events, software engineers should identify the associated functional requirements. Although, functional requirements should be identified for a given use case and they should describe how the game should react to particular inputs, the non-functional requirements should be described with respect to the overall game and specific qualities or properties that the game must have.

3.2.4.3 Requirements Specification

In the process of *Requirements Specification*, we technically establish each requirement, identified in the previous process, in a written form. By executing this process, we produce the requirements specification document, which is often part of the legal contract signed between the game designer and software developer.

A requirement is specified using "shall" in a sentence or a paragraph. For example, "The player *shall* slice the candy bar into 2 to 10 equal pieces by using a swipe gesture from top to bottom over the candy bar." However, the requirement specification should not be viewed as a sentence or a paragraph.

Example functional requirements for *Slicing candy in Level 1* use-case:

1. The game shall display a number pad that includes integer numbers from 2 to 10.
2. The game shall display an indicator as to which number is chosen, when the player taps a number on the number pad.
3. Tapping a number "n" from the number pad shall display "n-1" dashed lines over the candy.
4. Dashed lines shall be removed by the game when the player taps the "Back" or "Forward" button.

Example non-functional requirements for CandyFactory Educational Game for iPad [LTRG 2014a]:

1. The game shall be free to all users.
2. The game shall be developed as an iPhone operating system (iOS) application that runs on all iPad devices, namely, iPad (1, 2, 3, 4), iPad Mini (1,2), and iPad Air.

3. The game shall be playable only in landscape device orientation.
4. The game shall support all screen resolutions.
5. The game shall be playable without requiring Internet connection.
6. All assessment data to be collected during the game play shall be stored on the iOS mobile device individually for each player (student).
7. A tutorial shall be provided to teach how to play the game.

3.2.4.4 Requirements Quality Assessment

Requirements Development Quality Assurance (QA) integrates the assessments of quality of the Requirements Specification Document (RSD) work *product*, requirements engineering *process* quality, quality of the *people* employed in requirements engineering, and *project* characteristics related to the life cycle stage for requirements engineering.

A requirement is developed as a product with required quality characteristics as shown in Figure 15.

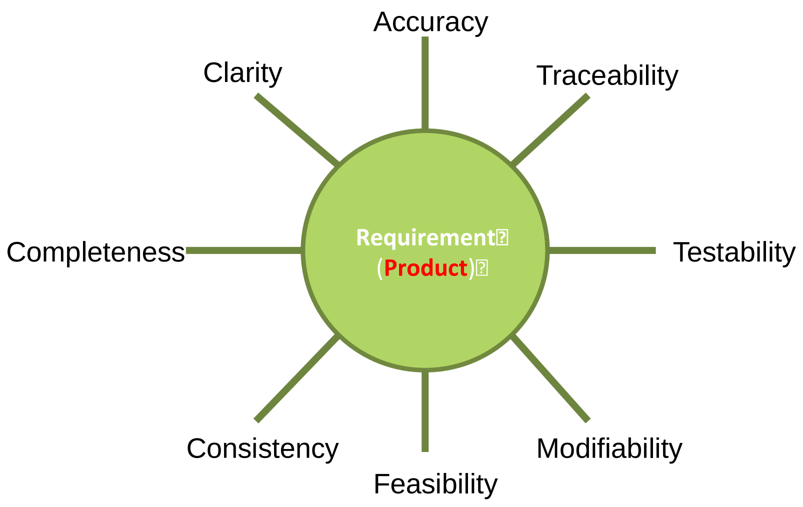


Figure 15. Requirement Quality Characteristics

1. *Requirements Accuracy:* is the degree to which the requirements possess sufficient transformational and representational correctness.
   a. *Requirements Verity:* is evaluated by conducting requirements verification. *Requirements verification* is substantiating that the requirements are transformed from higher levels of abstraction into their current form with sufficient accuracy. Requirements verification addresses the question of “Are we creating the requirements right?”
   b. *Requirements Validity:* is evaluated by conducting requirements validation. *Requirements validation* is substantiating that the requirements represent the *real* needs of the player (student) and the education problem domain with sufficient accuracy. Requirements validation addresses the question of “Are we creating the right requirements?”
2. *Requirements Clarity:* is the degree to which the requirements are unambiguous and understandable.

  a. *Requirements Unambiguity:* is the degree to which each statement of the requirements can only be interpreted one way.
  b. *Requirements Understandability:* is the degree to which the meaning of each statement of the requirements is easily comprehended by all of its readers.

3. *Requirements Completeness:* is the degree to which all parts of a requirement are specified with no missing information, i.e., each requirement is self-contained.
4. *Requirements Consistency:* is the degree to which the requirements are specified using uniform notation, terminology, and symbology, and any one requirement does not conflict with any other.
5. *Requirements Feasibility:* is the degree of difficulty of implementing a single requirement, and simultaneously meeting competing requirements. Sometimes requirements conflict with each other. It may be possible to achieve a requirement by itself, but it may not be possible to achieve a number of them simultaneously.
6. *Requirements Modifiability:* is the degree to which the requirements can easily be changed.
7. *Requirements Stability:* is the degree to which the requirements are changing while the game software application is under development, and the possible effects of the changing requirements on the project schedule, cost, risk, quality, functionality, design, integration, and testing of the application.
8. *Requirements Testability:* is the degree to which the requirements can easily be tested. A testable requirement is the one that is specified in such a way that pass/fail or evaluation criteria can be derived from its specification.
9. *Requirements Traceability:* is the degree to which the requirements related to a particular requirement can easily be found.

3.2.5 Architecture

Software architecture can be defined as "the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time" [DoD Integrated Architecture Panel, 1995, based on IEEE STD 610.12] or "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution " [IEEE STD 1471-2000].

Game software/systems can be developed in two different ways, (a) standalone, and (b) network-centric. A standalone game software/system does not require an interaction over a network and operates on a single digital device. On the other hand, a network-centric game system is an interconnection of software, hardware, and humans that operate together over a network to accomplish a set of goals. The main idea behind the "network-centric" characterization is to *connect everything with everything else* such as communities of interest, computers, databases, mobile devices, organizational entities, processes, satellites, sensors, and software.

Architecting is the process of creating and specifying an architecture for a network-centric DEG software application. This process is required only for those DEGs requiring network connection. It can be skipped for stand-alone DEGs.

The process of architecting takes the requirements specification document as input and produces a game architecture specification document as the output work product. It can be conducted under the guidelines provided by Chigani and Balci [2012].

The *game architecture specification* refers to the fundamental organization of DEG software application components that interoperate over a network (e.g., Internet), relationships among the DEG components, and the principles and guidelines governing the design and evolution of those components. DEG architecture is specified by typically using DoD Architecture Framework (DoDAF), which provides 52 diagrams for representing an architecture [DoDAF 2009a, b, c].

Commonly used architectures include client-server architecture, service-oriented architecture, and peer-to-peer architecture. The architecting process can be conducted by (a) selecting a known architecture, (b) composing an architecture from a set of known architectures, or (c) creating a new architecture.

3.2.5.1 Known Architectures for Network-Centric Software-based Systems

1. *Client-Server Architecture:* The client-server architecture consists of the following five conceptual layers [Chigani and Balci 2011].
   - *a.* *Layer 1. Data Source Layer* manages large set of structured game data, and run operations on the game data requested by numerous players. For example, a game database management system.
   - *b.* *Layer 2. Data Mapping Layer* focuses on improving the performance of the game system by way of creating mapping layer between different game data objects. For example, Entity Enterprise Java Beans, ActiveX Data Objects.
   - *c.* *Layer 3. Business / Application Logic Layer* manages the game logic. For example, all interactions between game objects are executed in this layer.
   - *d.* *Layer 4. Web Container Layer* consists of controller/mediator and game server-side presentation preparation components.
   - e. *Layer 5. Client Presentation Layer* manages the presentation of the game objects/components to the player. It can consist of Asynchronous JavaScript and XML (AJAX), Cascading Style Sheets (CSS), Document Object Model (DOM), Extensible HTML (XHTML), Extensible Markup Language (XML), Extensible Stylesheet Language Transformation (XSLT), HyperText Markup Language (HTML))
2. *Peer-to-Peer Architecture:* The peer-to-peer (P2P) architecture consists of a set of networked digital game devices, where each game device makes some of its resources (e.g., computational power, storage, music files, and technical documents) directly available to other game devices on the network. A P2P game device (node) acts as a supplier or consumer of resources [Chigani and Balci 2011].
3. *Service Oriented Architecture:* Service oriented architecture provides back-end services to the game systems. It is the underlying structure supporting communications between services. This approach defines how two computing game entities, such as enemy and hero, interact in such a way as to enable one game entity to perform a unit of work on behalf of another game entity. Typically this approach employs the client-server architecture. The client connects to the server to get different services [Chigani and Balci 2011].

3.2.5.2 Example Digital Game Architectures

Massively Multiplayer Online Game is a game played over the Internet by large numbers of players concurrently such as World of Warcraft [Blizzard 2014] and EVE Online [EVE 2014]. This type of game generally mixes a variety of gameplay types with multiplayer network-centric games in which large numbers of players cooperate and compete with each other on a large scale. The large scale poses significant technical challenges such as concurrency, fault tolerance, performance, persistence, and scalability. Employing an effective architecture can alleviate these challenges.

Commonly used software architectures for MMOGs are client-server architecture, multi-server architecture, peer-to-peer (P2P) architecture or combination of many of these architectures. For example, popular multiplayer online games Counter Strike, Doom, Medal of Honor, Quake, and Unreal Tournament are created based on the client-server architecture [Feng et al. 2002]. In client-server architecture, the server is typically responsible for both maintaining and disseminating the game state, as well as account management and player authentication [Berson 1996]. In a typical scenario, the client connects to the server (centralized-server) and sends all required game inputs. The server executes the input commands within a game logic, and then sends back the list of objects and related game data to the client. The clients can be anything from text terminals to advanced 3D rendering systems that allow the player to see the world in which s/he is playing. Moreover, the game data that is received by the client can show differences based on the client type [Kanter and Scott 1998].

There are two types of clients:

1. *Thin-Client Model:* In this model, all of the application processing such as game logic, and data management is carried out on the server. In some of the cases even rendering is executed by the servers [OnLive 2014]. In Figure 16, the client such as desktop, laptop, tablet and smartphone is simply responsible for transmitting user inputs, such as keyboard and mouse events. It also runs the presentation of the game software that is received from the server. The Http Server receives the requests and sends to the game servers. Game servers execute the game logic, and game states. After that send data back to the to the client. For example, most of the multiplayer online games that run on web-browsers can be considered thin-client. Companies like OnLive [OnLive 2014], and Gaikai [Gaikai 2014], have offered thin-client solutions for online gaming. While their design and implementations may differ from browser-based games, the concept is the same: game software/system runs on the server, and players just need to install the provided thin-clients to play the games. Although this approach causes heavy processing load on both the network and server, it removes many hardware requirements from the client [Kanter and Scott 1998; Lewandowski 1998].

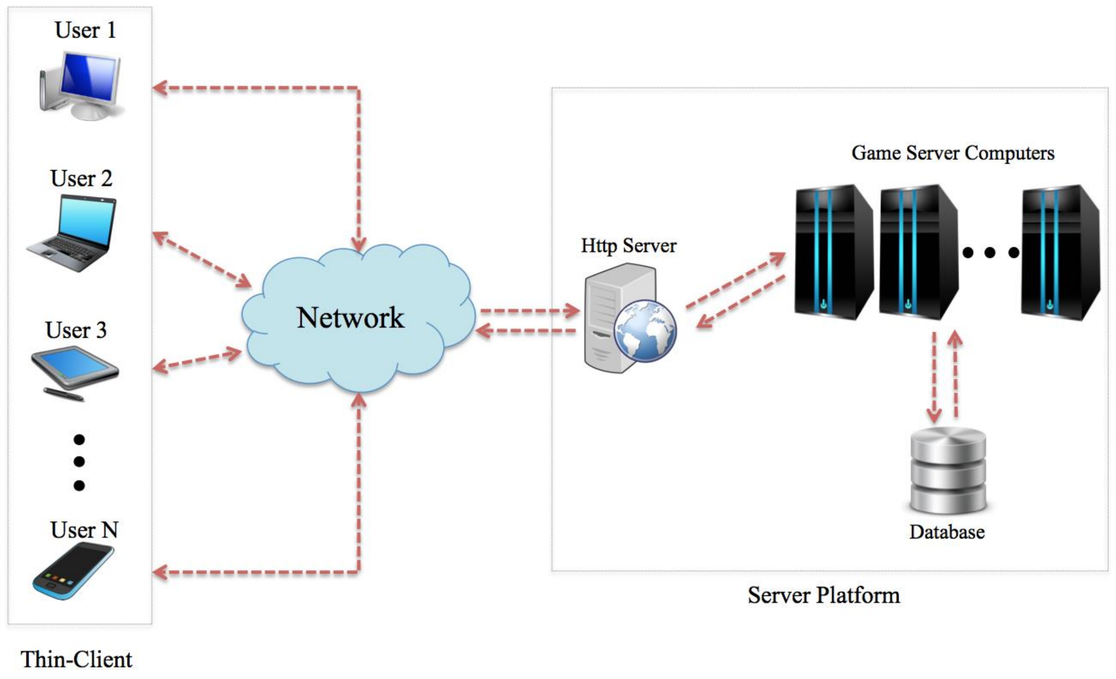


Figure 16. Thin-Client Server Architecture

2. *Thick-Client Model:* In this model, the server is only responsible for data management, the rendering and most of the game logic is executed on the thick-clients. As shown in Figure 17 the software on the powerful client implements the application logic and the interaction with the game user. First, the users interact with the Load Balancer (Controller/Mediator), which distributes the workload across multiple servers. Then, the game servers make necessary changes between the game objects, update the game state and sends back the data to the thick-client [Lewandowski 1998]. For example, popular game consoles such as Xbox and PlayStation can be considered a thick-client in which most of the game logic and the rendering gets executed by the game console (client) itself. Although this approach uses the network connection very efficiently, it requires clients to have powerful computing resources.

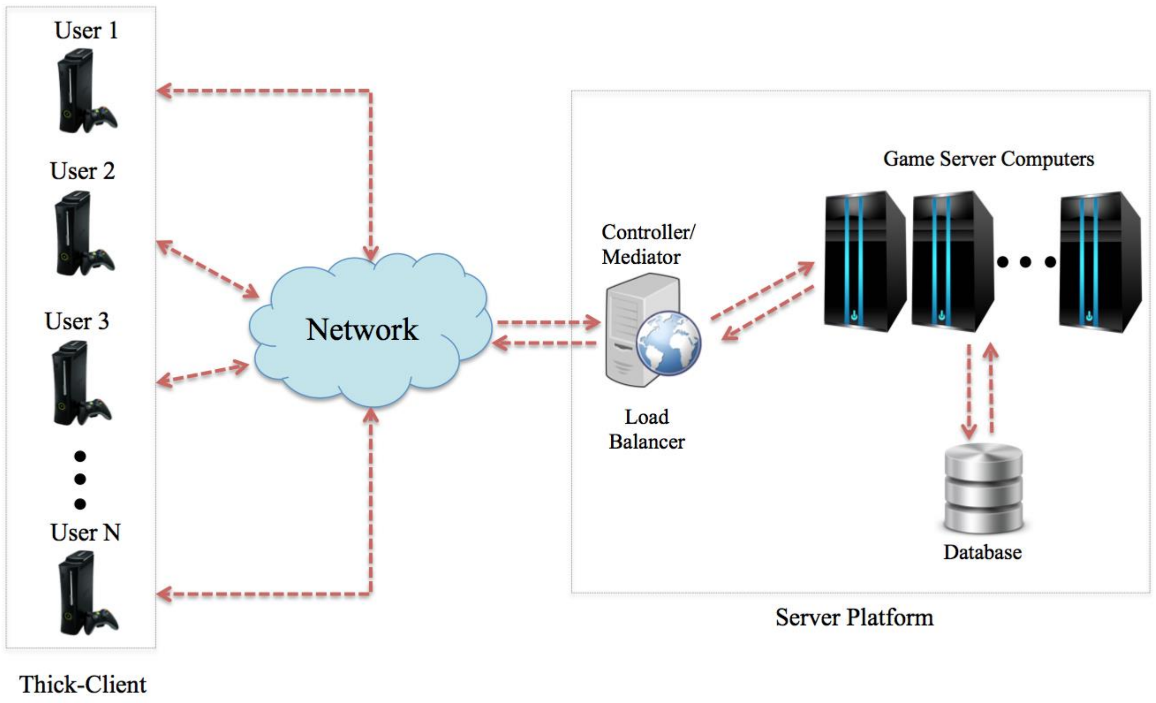


Figure 17. Thick-Client Server Architecture

Although the client-server architecture is commonly used in multiplayer online games, one of the main drawbacks is that, even the best dedicated-server can only support limited number of players, which is also known as scalability issue. The multi-server (client-server) architecture can be a solution to this issue [Kulkarni et al. 2010].

We divide the multi-server architecture into two categories:

- In the first approach, a server maintains each game world. Every server is responsible for a different set of clients and has a complete copy of the game world. That is, each set of clients and their server follow the traditional client-server architecture. Also, there is usually no need for communication between these servers. Players are prohibited to interact with a player who is in another game world. Furthermore, typically players are assigned to the servers based on their geographical location (e.g. North America, Europe). For example, in World of Warcraft, each game world is called *realm* and each realm is hosted in a different server. Each game world is duplicated across multiple realms to prevent any server from becoming too crowded. Consequently, the game provides thousands of servers to keep its 7 million users satisfied [Esbensen 2005; Karmali 2013].

- In the second category, a single game world is divided into multiple regions and each region is maintained by a different server. Similar to the previous approach, players from different regions are not allowed to interact with each other, but they are allowed to move between regions. For example, a popular game, *Second Life,* contains 18,000 regions where each of the regions is managed by a dedicated server [Varvello et al. 2009].

While it may be costly to set up dedicated servers for hundreds or thousands of players, they have advantages over some of the architectures such as peer-to-peer (P2P). For example, dedicated servers enable more players to play the game simultaneously in a more reliable manner. It removes the host migration interruptions, home network constraints, and cheating issues. Furthermore, it improves the game experience and game availability. For example, the Xbox Live Compute promises to provide dedicated servers for its Xbox One console players to remove these issues [Microsoft 2013].

Although the client-server architecture helps you to solve the scalability issue, it has its own drawbacks. For example, players from different regions cannot interact with each other, which can be a disadvantage. On the other hand, if the game allows players to move between regions, a complicated hand-off mechanism needs to be implemented to maintain the game state consistency [Knutsson et al. 2004]. Moreover, this architecture lacks flexibility and the server has to be over-provisioned to handle peak loads. It limits the deployment of user-designed games, which is an important trend in game design. Furthermore, the cost of setting up such a system can also be expensive [Mulligan et al. 2003].

A possible alternative to the client-server architecture is peer-to-peer (P2P) architecture in which players communicate with each other directly or via a dynamically chosen peer [Yahyavi and Kemme 2013]. In Ghost Recon, Halo series, and other Xbox 360 FPS games, generally a server assists players in discovering other peers to host and play with. For example, The Halo series, which has sold over 50 million copies worldwide, one console in each game session is selected to be the game host or server and all game communication between other players is relayed through this host console. On the other hand, a dedicated Xbox Live server provides accounting and matchmaking services for the players [Helgeson 2012; Lee et al. 2008].

Peer to peer (P2P) systems are decentralized systems where computations may be carried out by any node in the network. The overall system is designed to take advantage of the computational power and storage of a large number of networked computers. From a game development perspective, a particular advantage of this approach is that, the resources are added without any cost to the game provider. Each added node in the network shares the system load, which can remove the need for a dedicated server, if not completely eliminate it. Moreover, since each node only executes a small part of the whole process, a failure of a node does not affect the whole game system, if the game system is implemented properly. Also, an efficient implementation of this approach can provide a low latency because it removes the communication need between the server and the clients. Instead, required updates are directly exchanged between peers [Rieche et al. 2007; Yahyavi and Kemme 2013]. For example, the popular game Second Life increased the avatar interactivity five times faster by way of implementing P2P architecture [Varvello et al. 2009].

Although P2P architectures have been used widely, they also have several drawbacks. One of the commonly known issues is the system security [Rieche 2007]. Researchers showed that cheating in P2P architectures could be easier if there are no dedicated server to manage user accounts and game states [Baughman et al. 2007; Goodman and Verbrugge 2008; Kabus et al. 2005;]. Moreover, the implementation of P2P architecture is a challenging process. Distribution of the game state and other game related data between nodes makes it difficult to create efficient, and well-coordinated game system.

Depending on the game size and the type, different architectures can be combined. The client-server and P2P architectures are commonly used in a combined style [Ahmed et al. 2009; Najaran and Krasic 2010]. For example, popular BattleField and Call of Duty games use the combination of client-server and P2P architectures. These types of hybrid-architectures can be divided in different categories as follows:

- *Cooperative Message Dissemination:* In this approach, the game state is hold in one or more servers. The peers send their interaction to the servers and the server executes the data and sends back to the peers with P2P multicasting approach [Yahyavi and Kemme 2013].
- *State Distribution:* In this approach, the game state is distributed among peers and peers are responsible for the execution of players' actions. On the other hand, communication between peers

can be managed by servers. Moreover, the servers are responsible for player authentication and their status [Yahyavi and Kemme 2013].

- *Basic Server Control:* In this approach, the servers are responsible for managing user logins, payment information and players' progress. On the other hand, message dissemination and state distribution are done through P2P [Yahyavi and Kemme 2013].

While client-server and P2P architectures are commonly used by game developers, recent advances in cloud computing technology have turned the idea of executing the game software in the servers over the network instead of a user's devices. A great deal of interest has been generated for this approach recently, and several companies offer services, such as OnLive, Gaikai, and OTOY [Shea et al. 2013; Gaikai 2014; OnLive 2014; OTOY Inc. 2014;].

Cloud Computing is a style of computing in which dynamically scalable and often virtualized resources (e.g., application, desktop, operating system, server computer, storage) are provided as a service over a network, e.g., Internet, virtual private network (VPN), wireless network, Secret Internet Protocol Router Network (SIPRNET) [Rimal 2009]. The term "cloud" is used as a metaphor for "network". Networks have been traditionally represented using a cloud image. Hence, this style of network-centric computing technology has been called "Cloud Computing".

Games software/system that are hosted on the servers allow its users to play graphic intensive games without having the hardware that meets the minimum requirements of these games. This is an advantage for less powerful devices that are otherwise incapable of running high quality games. This technique stores and renders all game related data in its servers, and the user is only required to have a device that can play video, and receive input and access to the high speed internet [Shea et al 2013]. Moreover, this technique uses the latest streaming, server, and networking technologies to enable all different types of games to run on any devices. It enables users to take their game anywhere, anytime, on any device such as smartphones, tablets or PCs [Huan et al. 2013]. It also removes common problems that players receive on a daily basis such as the need for installation, booting, and error screens.

3.2.6 Software Design

The process of *software design* deals with the instantiation (creation) of a DEG software design from the game architecture specification. The design process takes the game architecture specification document as input and produces a game software design specification document as the output work product.

Game software are complex systems in that they comprise a large number of game components with many interactions between them [Blow 2004]. Understanding the underlying complexity of the system and performing the required design process is crucially important to develop high-quality game software.

Software designers should be aware of the distinct difference between *architecture* and *design.* Design in software engineering describes a life cycle phase that is concerned with *how* the system components will be built. On the other hand, architecture or architecting is a separate phase that is concerned mainly with the *structure* of the system in terms of its components, the relationships among these components, and the constraints that are placed on these relationships.

Moreover, the design is an instantiation from architecture similar to how an object is an instantiation of a class and we can derive two different designs from architecture. For example, consider the client-server architecture; we can design a software system using either the Java platform (JAVA EE) or Microsoft platform (.NET framework) [Pressman 2010]. Therefore, architecture is an abstraction of the components of the system and the way these components relate to one another.

DEG software is typically designed by using (a) object-oriented paradigm (OOP), (b) procedural paradigm (PP), or (c) combination of OOP and PP. Under the OOP, game software is designed by using objects instantiated from classes. Objects communicate with each other via message passing. Under the PP, game software is designed by using a main program calling subroutines at different logical points and subroutines calling other subroutines in a procedural manner.

The game software can be designed in one or more stages depending on the design complexity. Typically, we design complex game software in two stages, high-level design and detailed design or in three stages, preliminary design, interim design, and critical design. The initial design process of identifying sub-systems and establishing a framework for sub-system control and communication is called high-level design. The high level design can deal with structural design of a computer-based system or structural design of a software system. On the other hand, the detailed design exposes the details of each of these sub-systems [Pressman 2011].

3.2.6.1 Decomposition/Modularization

Decomposition/Modularization is a concept that is introduced in 1972 by Parnas with information hiding. It helps us to characterize game software/systems. The objective of the modularity is to decompose the game software/system into modules in such a way as to reduces the cost of maintenance, increase reusability and dramatically improve the adaptability of software. Moreover, this approach has contributed to the development of abstract data type programming languages, object-oriented design and programming.

Decomposition or modularization is the best solution for reducing and managing game software design complexity [Tarr et al. 1999]. A game software design is decomposed into modules (e.g., classes, subroutines) at level 1. Level 1 modules are further decomposed into modules at level 2. Level 2 modules are further decomposed into modules at level 3, and so on as depicted in Figure 18. The decomposition is carried out horizontally or vertically and continues until the leaf modules are manageable in complexity. Horizontal decomposition defines separate branches of modular hierarchy for each major program function. Under this technique the system is generally divided into various layers such as Level 1, Level 2 and Level 3 as shown in Figure 18. Vertical decomposition suggests that control and work should be partitioned in top down manner. In Figure 18 The bold lines show the vertical decomposition of Software Design to Module 2 then Module 2.2 and Module 2.2.2 and so on.

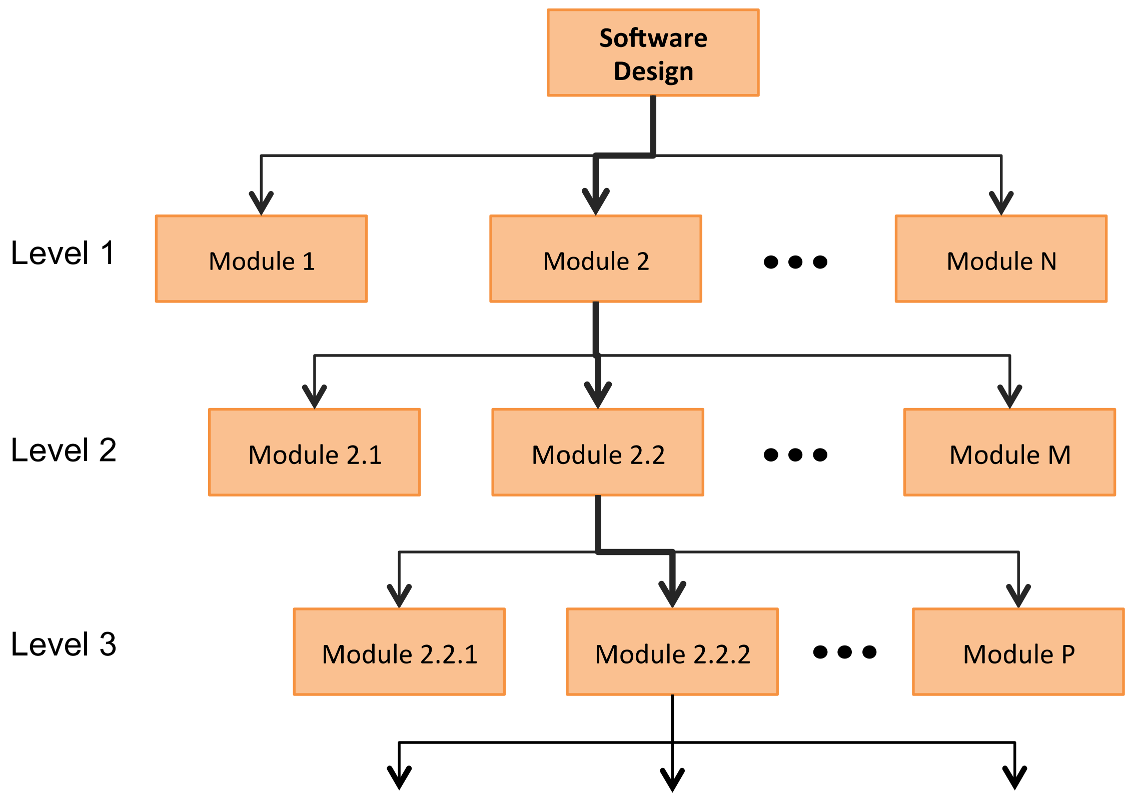


Figure 18. Modular decomposition of a software design

The Decomposition/Modularization approach brings the notion of interdependence within modules and independence between modules, which also can be referred to as *cohesion* and *coupling*. The module cohesion refers to degree of interaction within a module. The module coupling refers to the degree of dependency between modules. When there is a maximal interaction within each module (cohesion) and minimal dependencies (coupling) between modules, the maintenance effort is reduced and reusability is increased.

Furthermore, modularization yields several benefits in a design process such as increasing the manageability of the software by decomposing the functions of a complex system into parts that can be developed independently. It also allows designers to work in parallel even if they are geographically dispersed [Baldwin and Clark 2000].

We advocate the following design principle: Create a game software module in such a way that it has the highest possible cohesion and the lowest possible coupling. Moreover, the game software maintenance is facilitated and reusability is improved when the modules have high cohesion and low coupling.

There are 7 levels of cohesion as described below: [Schach 2002]

- *Level 7: Informational cohesion (good):* A module has informational cohesion if it performs a number of functions, each with its own entry point, with an independent code for each function, all performed on the same data structure.
- *Level 6: Functional cohesion (good):* A module has functional cohesion if it performs exactly one function or achieves a single goal.

- *Level 5: Communicational cohesion:* A module has communicational cohesion if it performs a series of functions related by the procedure to be followed by the product, but in addition all the functions operate on the same data.
- *Level 4: Procedural cohesion:* A module has procedural cohesion if it performs a series of functions related by the procedure to be followed by the product.
- *Level 3: Temporal cohesion:* A module has temporal cohesion if it performs a series of functions related in time.
- *Level 2: Logical cohesion:* A module has logical cohesion if it performs a series of related functions, one of which is selected by the calling module.
- *Level 1: Coincidental cohesion (bad):* A module has coincidental cohesion if it performs multiple, completely unrelated functions, or it is described (named) in terms of its logic.

There are 5 levels of coupling as follows [Schach 2002]:

- *Level 5: Data Coupling (good):* Two modules are data-coupled if all passed parameters are homogeneous data items. That is, every parameter is either a simple parameter or a data structure, all of whose elements are used by the called module.
- *Level 4: Stamp Coupling:* Two modules are stamp-coupled if a data structure is passed as a parameter, but the called module operates on some but not all of the individual components of that data structure.
- *Level 3: Control Coupling:* Two modules are control-coupled if one passes an element of control to the other module, that is, one module explicitly controls the logic of the other.
- *Level 2: Common Coupling:* Two modules are common-coupled if they have access to global data.
- *Level 1: Content Coupling (bad):* Two modules are content-coupled if one directly references the contents of the other.

3.2.6.2 Design Patterns

Game software is designed by using software design patterns. A *software design pattern* is a reusable and community accepted solution to commonly encountered software design problems. Many design patterns exist including creational patterns, structural patterns, and behavioral patterns [Gamma et al. 1995].

Some of the design patterns we used in the development of the *CandyFactory Educational Game for iPad* are described below.

- *Singleton Pattern:* The Singleton pattern is a creational software design pattern that restricts the instantiation of a class to one object [Gamma et al. 1995]. This pattern is commonly used when there is an object that needs to be coordinated across the system. The singleton pattern makes instantiated objects global, which can be accessed from any class. Although it is a useful pattern, the overuse of this pattern can introduce unnecessary restrictions and it can even be an anti-pattern. This pattern is used in the CandyFactory Educational Game for iPad by adapting a *GameManager* class that manages to load resources (e.g. sounds) and also objects that are needed to keep track of global game information.

- *Adapter Pattern:* Sometimes software designers may want to use some of their well-designed/tested classes in their system. On the other hand, it's very well known that your classes may not fit into the current system that you are working on. In such cases, the adapter pattern is used. The adapter is a structural software design pattern to fit two different types of objects together and make them work with each other without any problem [Gamma et al. 1995]. The adapter pattern enables classes to work together that couldn't otherwise because of incompatible interfaces. In CandyFactory

Educational Game for iPad the adapter pattern is used by way of adapting delegates. For example, the representation of different candies in the *GameScene* is achieved by adapting *TableView* and its corresponding delegates.

- *Delegation Pattern:* Delegation is a pattern in which one object in an app acts on behalf of, or in coordination with, another object [Apple 2014]. It is one of the fundamental abstraction patterns that underlie other design patterns such as composition and decorator. It has many advantages such as reducing the coupling between class methods or acting as an alternative to the inheritance. In CandyFactory Educational Game for iPad the delegation patterns is commonly used by the adapted game engine (Cocos2d) and also in the game software such as for handling touches.

- *Observer Pattern:* The Observer is a behavioral design pattern that enables objects to be notified when other objects' state changes [Gamma et al. 1995]. The implementation of observer pattern enables objects to broadcast notifications about its status changes to all of its observers. We adapted this pattern in CandyFactory Educational Game for iPad to handle touch events. The game engine (Cocos2d) provides *CCTouchDispatcher* class that has a list of observers (called delegates), and every time the *CCTouchDispatcher* class receives a touch it broadcasts it to all of its delegates.

- *State Pattern:* The State pattern is a behavioral software design pattern that describes how an object can alter its behavior when its internal state changes [Gamma et al. 1995]. This pattern enables software designers to divide the code into smaller self-contained units. Furthermore, it gives better control over the application's flow. This pattern is used in the CandyFactory Educational Game for iPad to manage overall application flow. Different game screens such as *IntroductionScene*, *MainMenuScene*, *TutorialScene*, and *GameScene*, are implemented as separate states.

3.2.7 Programming

The *programming* process takes the game software design specification document as input and produces executable game software components as the output work product. The programming transforms a game software design into executable code by using a high-level programming language (e.g., C, C++, C#, Objective C, Java) under an Integrated Development Environment (IDE), or a game engine (e.g., Cocos2d, Unity, Unreal Engine). Components of a large complex game design can be programmed by different teams, resulting in the production of multiple executable game software components.

During the programming process software designers can employ game engines. A game engine is a system that enables game developers to create games. Game engines make the development process easier for developers by way of abstracting details of common game-related tasks such as rendering, physics, networking, and dealing with inputs and outputs. This abstraction enables developers to focus on their game more than the underlying technology. Furthermore, most of the game engines offer reusable components that can shorten the development process such as loading, displaying, animating models and collision detection between objects and dealing with artificial intelligence [Gregory 2009]. There are plenty of game engines currently available and we review some of them based on their popularirty.

1. *Unreal Game Engine:* The Unreal Engine (UE) is a game engine technology that is developed by Epic Games. Although the game engine was first introduced in 1998 in a first-person-shooter (FPS) game Unreal, it has been widely used by developers to create a variety of Massively Multiplayer Online Role Playing Games (MMORPG) and other RPGs. The latest version of the UE 4, which is offered under a subscription plan per month, supports many features such as fully dynamic lighting features, updating directly in game without pausing the gameplay, viewing your game in full-screen within the editing environment, interactively visualizing the flow of gameplay code while testing the game and blueprint debugging. Although the UE supports 2D and 3D rendering, it is popular for

3D games. The latest version of UE enables developers to deploy games to Windows, Mac, Linux, PlayStation, Xbox, iOS, Android, Oculus VR, and HTML5. Furthermore, the game engine gives full access to its C++ source code [Epic Games 2014, ModDB 2014]. Unreal Tournament, Mass Effect series, Infiniti Blade series, and Gears of War series are some popular games that are developed with UE.

2. *CryEngine:* CryEngine is a game engine technology that is developed by Crytek. It was originally developed as part of the FPS game Far Cry. The latest version of CryEngine supports many features such as natural lighting & dynamic soft shadows, real time dynamic global illumination, AI system automated navigation mesh generation, motion blur & depth of field, and stereoscopic 3D support for all platforms. CryEngine is currently used to create games for a wide range of platforms such as Windows, Linux, OS X, Xbox, Play Station, Wii, iOS and Android. Furthermore, CryEngine is popular for developing 3D games. Aion, Crysis series, Far Cry series, Ryse: Son of Rome and Warface are some of the popular games developed with CryEngine [Crytek 2014].
3. *Unity:* Unity is a cross-platform game engine technology that is developed by Unity Technologies. The latest version of Unity game engine supports many features such as Mecanim, an animation system to animate any character or object, real-time shadows for all platforms, particle system update with world collision functionality, dynamic off-mesh links and obstacle carving. Although Unity game engine is not as powerful as CryEngine and UE, it stands out due to the efficiency of multiplatform publishing ability. The engine enables developers to build anything from complex 3D games to 2D mobile games. It also supports many different platforms such as iOS, Android, Windows, BlackBerry 10, OS X, Linux, web browsers, Flash, PlayStation 3, PlayStation Vita, Xbox 360, Windows Phone 8, and Wii U. Need for Speed World, Tiger Woods PGA Tour, Wolf Quest and Global Conflicts are some of the popular games that are developed with Unity game engine [Unity Technologies 2014].
4. *Cocos2d:* Cocos2d is a game engine for building 2D games and other graphical interactive applications. The original Cocos2d game engine is developed with Python language but later it's ported to the different platforms such as iOS platform (cocos2d-swift) or cross platform (cocos2d-x). It is open source, royalty free, and easy to extend. It is commonly used for mobile games. Although it supports only 2D rendering, it provides many features such as particle system, integrated physics engine, parallax scrolling support, streak motion support, and touch/accelerometer support. The cocos2d-swift uses objective-C and supports both iOS and Android platforms. On the other hand, cocos2d-x uses C++ and supports platforms such as iOS, Android, Windows Phone, BlackBerry, Tizen, and HTML5. Badland and Tiny Tower are some of the popular mobile games that are created with cocos2d game engine [cocos2d-swift 2014; cocos2d-x 2013].

3.2.8 Integration

Integration is the process of combining individually tested executable game software components developed by multiple teams into an integrated whole. This process takes the executable game software components as input and produces the integrated game software application as a finished product.

The process of integration should be well planned and executed. It is considered best practice to (a) establish an integrated product team consisting of one member from each component development team to oversee the planning and execution of the integration process, and (b) perform integration in a continuous and iterative manner, as opposed to all at once, to reduce the integration risk.

3.2.9 Publishing

Game software application can be published either as *Software as a Product* (SaaP) or *Software as a Service* (SaaS). Under SaaP, the game software is (a) made available either for download from a server computer (e.g., Apple App Store) or for delivery in a shrink-wrap box, (b) received by a user as a product, and (c) installed on a computer (e.g., desktop, laptop, handheld, or game console) for use. Under SaaS, the game software is (a) provided on a server computer, (b) connected to by a user over the Internet, and (c) used remotely either free of charge or for a subscription fee (e.g., World of Warcraft [Blizzard 2014], EVE Online [EVE 2014]).

3.2.10 Game-Based Learning

The *game-based learning* process refers to the process in which the completed game software application is used by students for learning a subject. The ultimate goal of a DEG is to provide a learning environment with the highest possible quality. During this process, that quality is assessed.

Assessment of game-based learning poses significant technical challenges for researchers and educators [Ifenthaler et al. 2012]. Available assessment techniques should be employed in this process for determining how well the students learn with a DEG and how effectively a DEG transforms learning in a way that game-based learning provides a better learning experience in comparison to other pedagogies. Ifenthaler et al. [2012] indicate that "Aligning learning and assessment is at the core of creating a favorable and effective learning environment, one that is learner-centered, knowledge-centered, and assessment-centered."

3.2.11 Feedback

This process focuses on documenting the game-based learning assessment results and communicating them to the game developers as feedback. The game development restarts based on this feedback and goes through another life cycle iteration to produce a revised improved version of the DEG. This iteration is repeated to continuously improve the quality of the DEG during its life cycle.

3.2.12 Maintenance

The *maintenance* process, designated by a circle in Figure 6, starts after the DEG software application is published. It deals with the modification of a DEG software application after being published to (a) cope with changes in the DEG's external environment, (b) diagnose and correct faults, (c) improve quality by satisfying new or changed user requirements, or (d) prevent problems in the future. Maintenance process touches every stage of the lifecycle.

Four types of maintenance exist:

1. *Adaptive maintenance*: adaptations required as changes occur in the DEG's external environment (e.g., new operating system, new game engine, new screen sizes)
2. *Corrective maintenance*: fixing errors reported by the players and others.
3. *Perfective maintenance*: adding new capabilities, improving existing ones, or making functional enhancements based on feedback from the players and others.
4. *Preventive maintenance*: making modifications or reengineering so as to prevent potential future problems.

CHAPTER 4: DIGITAL EDUCATIONAL GAME SOFTWARE QUALITY EVALUATION METHODOLOGY (IDEALLY)

The evaluation of a *Digital Educational Game* (DEG) possesses significant technical challenges for designers, engineers, target users, and teachers. Evaluation is a complex process that involves measuring and evaluating hundreds of quantitative (e.g., throughput, utilization) and qualitative elements (e.g., design quality, maintainability), mandating subject matter expert (SME) evaluation, and requiring the integration of disparate measurements. Planning and managing such evaluations requires a unifying methodology and should not be performed in an *ad hoc* manner.

A *DEG* quality consists of *Digital Educational Game Software Quality* and *Digital Game-Based Learning Quality* as shown in Figure 19.

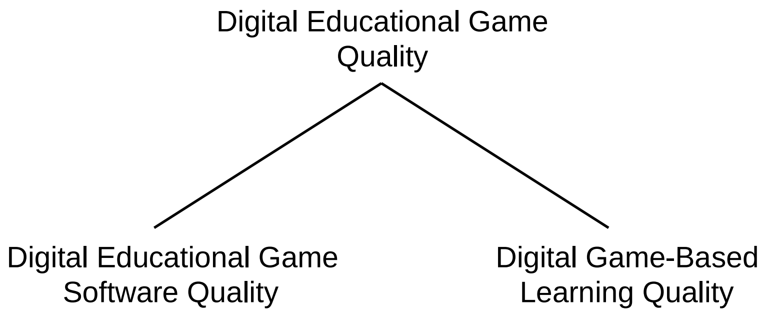

Digital Educational Game Quality
Digital Educational Game Software Quality
Digital Game-Based Learning Quality

Figure 19. Digital Educational Game Quality Decomposition

- *Digital Educational Game Software Quality* is the degree to which the DEG software possesses a desired set of characteristics.
- *Digital Game-Based Learning Quality* assessment deals with the determination of how well the DEG teaches the subject it is intended to teach.

This chapter focuses on the *Digital Educational Game Software Quality* evaluation.

4.1 Background

We advocate the use of the Evaluation Environment (EE) methodology [Orca Computer 1999a, b]. The EE methodology is chosen based on the reasons that are listed in Table 5.

Table 5. Evaluation using the EE Methodology and Common Practices

Evaluation using the EE Methodology	**Evaluation using Common Practice**
Provides a disciplined and structured approach	Conducted in an ad hoc manner
Enables integrative evaluation	Checklist item evaluations are not integrated
Evaluation is provided on a score scale of 0 to 100	Evaluation is based on 0-1 decision variable
Provides criticality weighting of different evaluations using AHP	Does not use criticality weighting and does not use AHP
Enables what-if analysis	No what-if analysis capability
Allows aggregated assessment	Does not allow aggregated assessment
Enables collaborative evaluations	Does not allow collaborative evaluations

The preliminary research for the EE methodology is done by Talbert [1995] for the measurement and evaluation of complex system designs. Later a software tool, EE, was designed and developed at Orca Computer, Inc. with significant improvements to the methodology. The EE software tool was developed to facilitate the application of the methodology. The methodology and the tool have been used in many Department of Defense evaluation projects either throughout the software / system development life cycle or after the development is completed. Some of the example applications of the EE methodology and software are listed below:

- National Missile Defense (NMD) System Performance Evaluation
- NMD System Interoperability Assessment
- Software Acceptability Assessment for Certification
- BEST Master Test Plan Evaluation

The EE methodology consists of the following body of methods, rules, and postulates:

1. Employment of subject matter experts (SMEs),
2. Construction of a hierarchy of indicators,
3. Relative criticality weighting of indicators using the analytic hierarchy process,
4. Using a rule-based expert knowledge based with an object-oriented specification language,
5. Assignment of crisp, fuzzy, and nominal scores for the indicators,
6. Aggregation of indicator scores,
7. Graphical representation of the indicator scores and weights,
8. Hypertext evaluation report, and
9. Interpretation of the results.

Evaluation is an important activity in many disciplines and it focuses on accurate assessment of a given concept. Although the goal is the same for all disciplines, different disciplines use different terminologies such as index, measure and metric. The EE methodology uses the term *indicator* to refer to any of those terms. An *indicator* is defined as an indirect measure of a qualitative concept or a direct measure of a quantitative concept.

The construction of indicators is the main challenge in EE methodology. It requires SMEs to hold numerous meetings for carefully examining the problem domain and to create a hierarchy of indicators for the given concept. Later, these indicators are processed by SMEs and the scores are calculated and graphically represented in different meaningful ways as a result.

4.2 A Hierarchy of Indicators

Digital educational game software quality is a qualitative concept and it cannot be directly measured or assessed. Therefore, a set of indicators is created to measure it indirectly at the top level as shown in Figure 20.

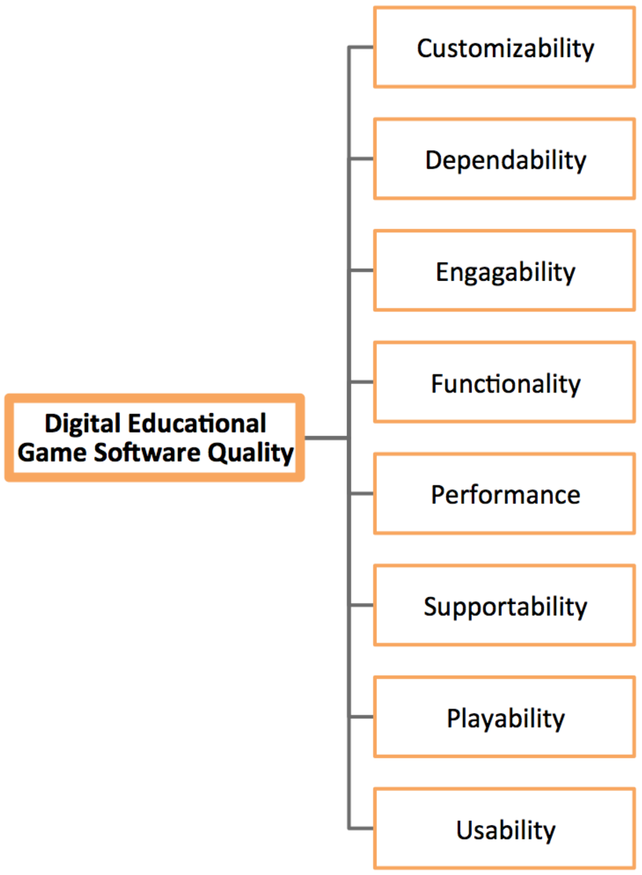


Figure 20. A set of hierarchy of indicators at top level

The following indicators are proposed to evaluate digital educational game software quality. It should be noted that the following list is not a complete list of indicators; it is an example list of indicators that can be modified.

The indicator name specified between < and > indicates that the indicator is defined somewhere else in the hierarchy.

1. **Customizability** is the degree to which the player can customize the game by using available game options, preferences or settings.

    a. **Game Difficulty Adjustability** is the degree to which the player can adjust the game difficulty level according to his or her skill, e.g., how well the game levels (easy, medium, and hard) are provided for intended players.

    b. **Internationalization** is the degree to which the game can be used in many different countries.

        i. **Diversity** is the degree to which the game accommodates different cultures and races by using culture-specific graphics, visualizations, and sounds, e.g., how well the game character reflects the characteristics of the African-American or Asian people.

ii. **Language Selectability** is the degree to which the game can be played using different languages, e.g., how well the different language options such as German, Swedish are provided.

iii. **Localizability** is the degree to which the game can be installed or setup to use region-dependent options such as measurement units, currency symbol, decimal separator, time zone, or calendar, e.g., how well the metrics (e.g. miles for United States or meters for European countries) are provided for different regions.

c. **Personalizability** is the degree to which the player can customize the available game characters and other game objects, e.g., how well the game object (e.g. delivery truck) is customizable by the player.

d. **Resetability** is the degree to which the player can reset his or her own game scores, trophies or collected data, e.g., how well the trophies or the scores are reset when the player wants to start from scratch.

2. **Dependability** is the degree to which the game (a) delivers services when requested, (b) delivers services as specified, and (c) protects itself against accidental or deliberate intrusion from malware.

a. **Availability** is the degree to which the game delivers services when requested, e.g., how well the game is paused and resumed at any given time.

b. **Reliability** is the degree to which the game performs its required functions correctly and without failure under prescribed conditions, and enables the player to recover from errors.

i. **Accuracy** is the degree to which the game possesses sufficient transformational and representational or behavioral correctness.

1. **Validity**: Game validity is assessed by conducting game validation. Game validation is substantiating that the game possesses sufficient representational and behavioral accuracy. Game validation addresses the question of "Are we building the <u>right</u> game?"

2. **Verity**: Game verity is assessed by conducting game verification. Game verification is substantiating that the game is transformed from one form of abstraction into another with sufficient accuracy. Game verification addresses the question of "Are we building the game <u>right</u>?"

ii. **Fault-Tolerance** is the degree to which a multi-player online game continues to function properly in the event of unexpected failure of some of its components, e.g., how well the game functions when there is no network connection.

iii. **Recoverability** is the degree to which the game enables players to recover from errors, e.g., how well the game resumes from the previous game state after a crash or an error.

c. **<Security>**

3. **Engagability** is the degree to which the player is captivated and addicted by playing the game.

a. **Challengability** is the degree to which the game provides increasingly challenging modes, e.g., how well the different game levels (easy, normal and difficult) are provided.

b. **Creativity** is the degree to which the game uses nonconventional ways to teach the learning objectives. (Good games are not merely clones of other games, but add something original), e.g., how well the different techniques or metaphors (e.g., splitting and iterating) are adopted to teach fractions.

c. **Curiousness** is the degree to which the game arouses or excites the player's speculation, interest, or attention through being inexplicable or highly unusual.

d. **Engrossness** is the degree to which the game absorbs all the attention or interest of the player.

e. **Enjoyability** is the degree to which the user finds the game playing to be fun, e.g. how well the game entertains the intended users.

f. **Excitability** is the degree to which the game is exciting to play.

g. **Game Story Quality** is the degree to which the game story possesses a desired set of characteristics.

    i. **Characters Quality** is the degree to which the game characters such as people, animals, or creatures in a story possess a desired set of characteristics.

        1. **Graphical Representativeness** is the degree to which the game character properly looks like the real or fictitious character it is intended to represent, e.g. how well the game object (e.g., the delivery truck) is represented.

        2. **Behavioral Representativeness** is the degree to which the game character properly mimics the behavior (e.g., emotions) it is intended to represent, e.g., how well the game character (e.g., the manager) becomes happy when the player completes the level.

        3. **Capability Representativeness** is the degree to which the game character demonstrates the capability (e.g., magical powers) it is intended to possess within the game world, e.g., how well the game characters adopt powers such as changing shapes, flying, being invisible and self-healing with respect to the game story.

        4. **Motional Representativeness** is the degree to which the game character possesses proper movements (e.g., walking, running, jumping), postures, and gestures with respect to the game story.

    ii. **Game World Quality** is the degree to which the place, time, and surroundings in which the story takes place possesses a desired set of characteristics.

        1. **Environmental Quality** is the degree to which the game meaningfully presents the game world's appearance, terrain, climate, habitat, atmosphere, and historical perspective that are expressed, assumed, or implied in the

game world, e.g., how well the buildings are represented for a game that takes a place in the Dark age.

2. **Universe of Discourse Quality** is the degree to which the game meaningfully presents the game world's universe of discourse including objects, events, activities, relations, concepts, beliefs, attitudes, that are expressed, assumed, or implied in the game world, e.g., how well the cars in the traffic or the people in the crowd are animated and affected by each other.

3. **Dimensional Quality** is the degree to which the game world is properly and meaningfully represented with a physical dimension such as 2D (two-dimensional) or 3D (three-dimensional).

4. **Proportional Quality** is the degree to which the game world objects are created in proportion to each other in term of size, judged with respect to the game objectives, e.g., how well the people and the buildings are proportionally scaled down (Based on the game story the proportional quality may show variety).

5. **Ethical Quality** is the degree to which the game world possesses a desired set of ethical characteristics in reality games, e.g. how well the education is promoted.

6. **Temporal Quality** is the degree to which the game world properly manages time throughout the game play, e.g., how well the daytime or night is managed.

7. **Plot Quality** is the degree to which the game story properly structures and sequences the (a) introduction of game characters, (b) identification of challenges faced by the characters, (c) overcoming a challenge, and (d) ending of the game.

iii. **Stimulability** is the degree to which the game story arouses, activates, and triggers the player's interest.

h. **<Informative Feedback>**

i. **Inspirability** is the degree to which the game encourages players to do something creative, e.g., in an educational mathematical game, how well the equivalent fractions are accepted as the correct answer (e.g. ½ = 2/4).

j. **Interactivity** is the degree to which the user actively interacts with the game during playing, e.g., how well the player is actively engaged in the game by way of requiring player's input and and let him/her decide to perform different tasks.

k. **Internal Locus of Control** is the degree to which the player feels that s/he is in charge of the game and the game responds to her/his actions, e.g., how well it is allowed to skip a tutorial or an introduction.

l. **Liveliness** is the degree to which the game possesses a desired set of characteristics such as animated, eventful, exciting, spirited, thrilling, and vivacious, e.g., how well the game objects are animated.

m. **Randomness** is the degree to which the game uses randomly determined values or conditions, e.g., how well the different (non repeated) questions, tasks and challenges are represented.

n. **Satisfaction** is the degree to which the game fulfills the desires, expectations, needs or demands of a player. Games that captures player's full attention or encourage them to play more are the games that is satisfying its players, e.g., how well the game is desired to be played.

o. **Sociability** is the degree to which the game enables the players to have real-time conversations with other players, publish their game information, and play the same game with other players in real time.

    i. **Communication** is the degree to which the game enables the players to have a real-time conversation with each other via text, audio or video, e.g., how well the communication between players is provided.

    ii. **Multiplayerness** is the degree to which the game allows more than one player to play in the same game environment at the same time by way of partnership, competition or rivalry manner, e.g., how well the same gaming environment is shared by two or more players.

    iii. **Publishability** is the degree to which the game enables the player to publish his or her game information on social networks (e.g., YouTube, Facebook, Twitter), social gaming networks (e.g., Game Center, Xbox Live), or public and personal websites.

p. **Versatility** is the degree to which the game provides a rich set of features such as actions, animations, characters, game levels, and scenes, e.g., how well the different game levels easy, medium and hard are provided.

4. **Functionality** is the degree to which the game provides all of the desired features and capabilities.

    a. **Capability** is the degree to which the game performs its set of functionalities (features), e.g., how well the player's input is captured.

    b. **Comprehensiveness** is the degree to which the game includes all of the desired functionalities (features), e.g., how well the sound is provided.

    c. **Correctness** is the degree to which the game performs its required function in accurate and right way without any error or faults, e.g., the how well the game score is calculated.

    d. **Security** is the degree to which the game provides protection and authentication of information in transit or stationary, as well as the confidentiality of sensitive information, e.g., how well the players data is securely transferred in multiplayer gaming environment.

5. **Performance** is the degree to which the game executes its work in a speedy, efficient, and productive manner.

a. **Efficiency** is the degree to which the game fulfills its purpose without waste of resources.

    i. **Algorithmic Efficiency** is the degree to which the algorithms used in the game fulfill their purposes without waste of resources, e.g., how well the algorithm that is being used to propagate different game objects is provided.

    ii. **Architectural Efficiency** is the degree to which the game's components are structured to interact with each other without waste of resources.

    iii. **Execution Efficiency** is the degree to which the game utilizes central processing unit (CPU) and graphics processing unit (GPU) without waste of resources.

        1. **CPU Efficiency** is the degree to which the game uses CPU without waste of resources, e.g., how well the CPU is being used by the game.

        2. **GPU Efficiency** is the degree to which the game uses GPU without waste of resources, e.g., how well the GPU is used by the game.

    iv. **Storage Efficiency** is the degree to which the game utilizes random access memory, local disk storage, and cloud storage without waste of resources.

        1. **Cloud Storage Efficiency** is the degree to which the game utilizes the cloud storage efficiency, e.g., how well the cloud storage is used by the game.

        2. **Local Disk Storage Efficiency** is the degree to which the game utilizes the local disk storage, e.g., how well the local disk storage is used by the game.

        3. **Random Access Memory Efficiency** is the degree to which the game utilizes random access memory, e.g., how well the random access memory is used by the game.

    v. **Cloud Data Access Efficiency** is the degree to which the game accesses cloud data without waste of resources, e.g., how well the game downloads/uploads player's related data to the cloud servers.

b. **Hardware Speed** is the degree to which the hardware (e.g., game console, mobile devices, laptop, desktop, server computers) the game runs on provides adequate performance, e.g., how well the game objects are animated (are there any lag or slowness during the animation?).

c. **Network Speed** is the degree to which the network (e.g., cellular, wireless, wireline) the game uses provides adequate performance, e.g., how well the game objects are updated in a multiplayer gaming environment.

d. **Responsiveness** is the degree to which the game quickly responds to player actions, e.g. how well the player movements are updated when there is a player input to move a character.

e. **Symmetry** is the degree to which the game provides equal resources to the players across the game, e.g., how well the resources are shared between different players in a multiplayer gaming environment.

6. **Playability** is the degree to which the game provides acceptable player experience, which includes the player's accomplishments, emotions, perceptions, and physical and psychological responses that occur before, during and after the game play.

    a. **<Clarity>**

    b. **Consistency of gameplay** is the degree to which the game provides persistent gameplay and game mechanics to the players, e.g., how well the gameplay and the game mechanics are persistent between different game levels.

    c. **<Challengability>**

    d. **<Customizability>**

    e. **<Direct Manipulation>**

    f. **<Functionality>**

    g. **Game Controllability** is the degree to which the game provides easy to use game controls, e.g., how well the game controllers are provided.

    h. **Game Rating** is the degree to which the game provides an appropriate classification according to the rating categories (e.g. early childhood, teen, adult), content descriptors (e.g. blood, nudity, use of alcohol) and interactive elements (e.g. shares info, shares location, users interact).

    i. **<Help Quality>**

    j. **Learnability** is the degree to which the player can easily understand the game play such as objectives, rules, and game mechanics, e.g., how well the game characters are animated and controlled by the player's input.

    k. **Pauseability** is the degree to which the player can temporarily stop the game and resume playing at a later time.

    l. **Reloadability** is the degree to which the player can save the state of the game so that the saved game can be reloaded at a later time for the player to resume playing, e.g., how well the last game state is reloaded (does the reloaded game state match the saved game state?).

    m. **Self-containedness** is the degree to which the game requires previous game experience to enable players to play the actual game, e.g., how well the player's previous game experience is needed to play the actual game.

    n. **<Tutorial Quality>**

7. **Supportability** is the degree to which the game can be supported.

    a. **Adaptability** is the degree to which the game can be modified to satisfy changing requirements.

b. **Agility** refers to the ability of a game to both be flexible and undergo change rapidly, e.g., how well the game software is implemented.

c. **Compatibility** is the degree to which the game is compatible with different platforms such as Android, iOS, Play Station, Xbox, and Wii, e.g., how well the game is compatible with Android or iOS platform.

d. **Conciseness** is the degree to which the game is compact.

e. **Correctability** is the degree to which the game facilitates changes for fixing bugs and making corrections.

f. **Extensibility** is the degree to which the game (a) is capable of growing by including more and a greater diversity game of components, and (b) facilitates the extension of its capabilities by modifying current game features or adding new game features.

g. **Installability** is the degree to which the game can be seamlessly installed and prepared to play, e.g., how well the game is installed in iOS platform.

h. **Interoperability** is the degree to which a game in a distributed environment can exchange data with one or more other game system and be able to use the data that has been exchanged (multiplayer games).

i. **Maintainability** is the degree to which the game facilitates changes for:

    i. Adaptations required as the game's external environment evolves (adaptive maintenance)

    ii. Fixing bugs and making corrections (corrective maintenance)

    iii. Enhancements brought about by changing customer requirements (perfective maintenance)

    iv. Preventing potential problems or for reengineering (preventive maintenance)

j. **Modifiability** is the degree to which the game can easily be changed.

k. **Modularity** is the degree to which the game is decomposed into game components in such a way that each game component has the highest possible cohesion and the lowest possible coupling.

l. **Portability** is the degree to which the game can be transferred from one hardware or software environment to another, such as from iOS to Android platform.

m. **Scalability** is the degree to which a game working well under light loads will continue to function correctly as its workload (e.g., number of users, size of network, and amount of processing) is increased within anticipated limits (e.g., capacity of game infrastructure to accommodate increasing number of players).

8. **Usability** is the degree to which the game can easily be employed for its intended use.

a. **Aesthetics** is the degree to which the game possesses sufficiently pleasing appearance or behavior. The sufficiency is judged with respect to the game's intended users, e.g., how well the game object (e.g., the delivery truck) is represented to it's intended users.

b. **Clarity** is the degree to which the game is unambiguous and understandable and the game objectives and instructions are conveyed to the player in a clear and comprehensible manner, e.g., how well the different game mechanics are defined in the tutorial.

c. **Closureness** is the degree to which the game provides closures to dialogues organized into stages such as beginning, middle, and end, e.g., how well the feedbacks are displayed for each task that is performed by the player.

d. **Consistency** is the degree to which the game employs uniform layout, notation, terminology, and symbology, e.g., how well the metrics are employed in the game.

e. **Direct Manipulation** is the degree to which the player manipulates a game object by directly interacting with it, e.g., how well the candy bar animated when the app receives the touch event.

f. **Help Quality** is the degree to which the game possesses a desired set of characteristics for providing assistance to the player.

    i. **Context-Sensitive Help Quality** is the degree to which the game provides assistance to the player for the particular feature that s/he is in the process of using, e.g., how well the help information is defined to complete the level one.

    ii. **Global Help Quality** is the degree to which the game provides assistance to the player for the whole game, e.g., how well the help information is described to define the overall game goal.

g. **<Internal Locus of Control>**

h. **Informative Feedback** is the degree to which the player receives informative feedback in the form of animation, graphics, audio, video, or text when performing certain actions in the game.

    i. **Accomplishment Feedback** is the degree to which the game provides rewards (e.g., points, money, trophies, certificates) to the player so that the player feels a sense of accomplishment, e.g., how well the reward is provided when the player completes a task.

    ii. **Action Feedback** is the degree to which the informative feedback is provided based on the particular action that the player is in the process of performing, e.g., how well the results are displayed when the player completes a task.

    iii. **Error Feedback** is the degree to which the game provides polite, concise, consistent, and constructive error messages, e.g., how well the error message is defined when the player makes a mistake.

i. **Representativeness** is the degree to which the game represents what it is supposed to represent.

i. **Detailedness** is the degree to which the game provides sufficient level of detail, e.g., how well the game robot is represented.

ii. **Fidelity** is the degree to which the game mimics the state and behavior of a real-world object, feature or condition, e.g., how well the delivery truck mimics the state and behavior of a real world truck.

iii. **Realisticness** is the degree to which the game represents familiar things in a way that is sufficiently accurate or true to life, e.g., how well the simulated candy bar is represented.

j. **Short Term Memory Load** is the degree to which the game requires the player to remember certain information to be able to perform a game action.

k. **Tutorial Quality** is the degree to which the game teaches the player how to play the game, e.g., how well the game mechanics are defined in the tutorial.

4.3 Case Study: CandyBot Software Quality Evaluation by Using IDEALLY

In this case study, we provide instructions for using IDEALLY to assess the quality of the CandyBot educational game.

4.3.1 Step 1: Project Identification and Setup

Identify a digital educational game to assess. Identify the IDEALLY Project Administrators and subject matter experts (SMEs). Have all Administrators and SMEs register as IDEALLY users.

4.3.2 Step 2: Creation of Hierarchy of Indicators

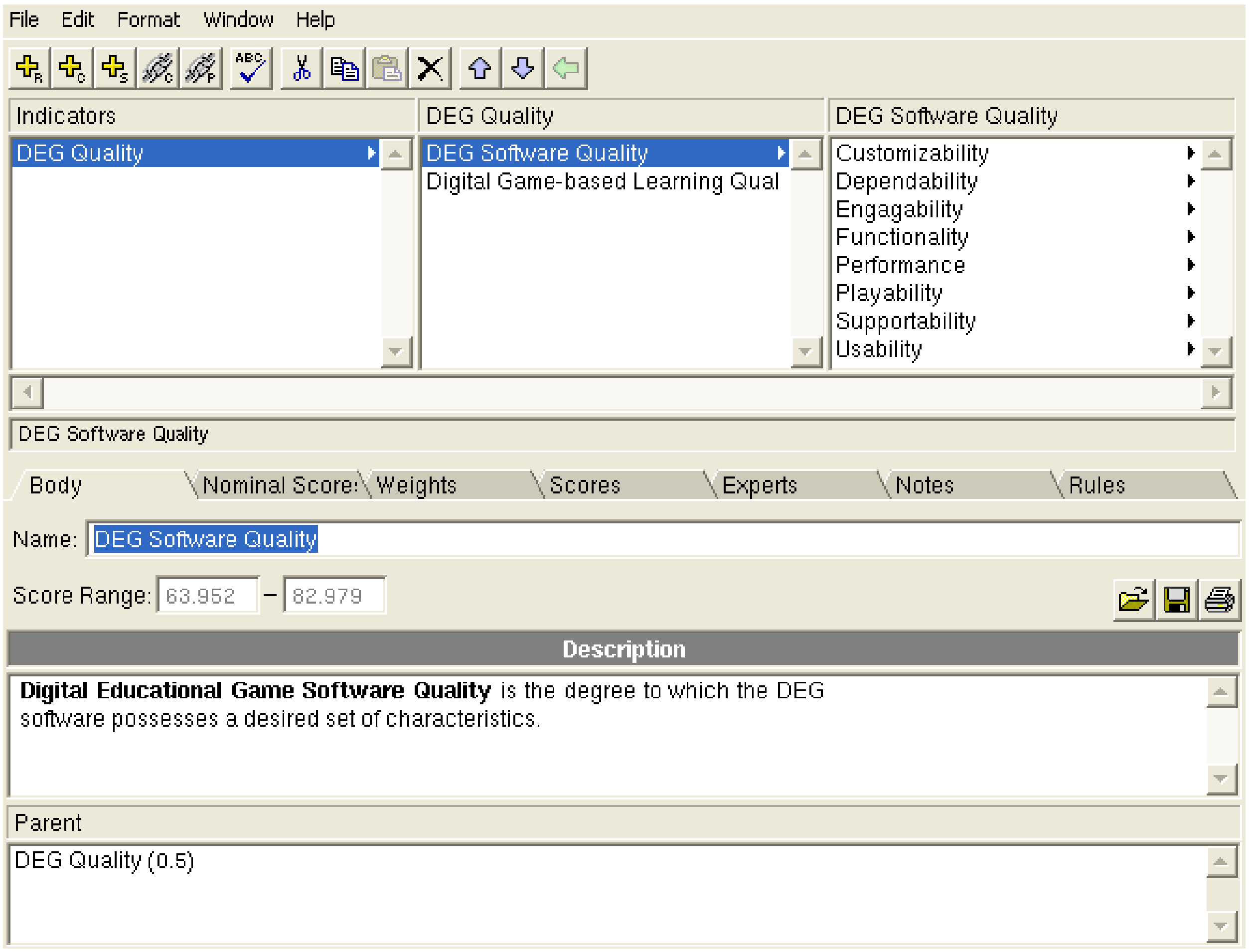
Create a hierarchy of indicators with all of the Administrators and SMEs. Figure 21 shows the top two levels of hierarchy of indicators for quality assessment of CandyBot game software. The evaluation of the CandyBot game software quality is a qualitative concept which cannot be assess directly. Therefore, to measure it indirectly, we decompose it at level 1 into quality assessments of Customizability, Dependability, Engagability, Functionality, Performance, Playability, Supportability, and Usability. Each of the eight quality indicators at level 1 is also a complex qualitative concept and cannot be directly measured. Hence we decompose each further at level 2 into quality indicators as show in Figure 21.

Each quality indicator at level 2 is also a complex qualitative concept and cannot be directly measured. Hence, we decompose each further at level 3. The decomposition continues until the leaf indicators are assessable [Balci 1997; Balci 2001; Balci et al. 2002].

The hierarchy consists of three types of indicators: root, branch, and leaf [Balci 2001].

- A *root indicator* is at the top of the hierarchy without any parent.
- A *branch indicator* is an indicator which has at least one parent indicator and at least one child indicator.
- A *leaf indicator* is an indicator that has at least one parent and no child indicator.

The hierarchy of indicators for IDEALLY is listed in section 4.2.



Indicators	DEG Quality	DEG Software Quality
DEG Quality	DEG Software Quality	Customizability
	Digital Game-based Learning Qual	Dependability
		Engagability
		Functionality
		Performance
		Playability
		Supportability
		Usability

DEG Software Quality

Name: DEG Software Quality

Score Range: 63.952 – 82.979

Digital Educational Game Software Quality is the degree to which the DEG software possesses a desired set of characteristics.

DEG Quality (0.5)

Figure 21. Top Two Levels of a Hierarchy of Indicators for CandyBot Assessment

4.3.3 Step 3: Review of the Hierarchy of Indicators

Create an independent review panel, and assess the appropriateness of the hierarchy of indicators. The review panel may include, SMEs, technical people, managers, and decision makers [Balci 2001]. The hierarchy should be judged to determine if [Balci et al. 2002]:

- it covers all essential elements of the problem/application domain
- it has sufficient detail in measurement
- the leaf indicators are directly measurable

4.3.4 Step 4: Assignment of Nominal Scores Sets

An SME can assign a crisp, fuzzy or nominal score, defined as follows [Balci 2001, Balci et al. 2002]:

- A *crisp score* is a single real value between 0 and 100 (e.g., 65.5)
- A *fuzzy score* is an interval of real values within the range of 0 and 100 (e.g., [65.5, 80.5]).
- A nominal score is a named score with a predefined crisp or fuzzy value.

Table 6 shows a set of nominal scores that is used in IDEALLY to judge the quality of the CandyBot game.

Table 6. Nominal score set for judging the quality of CandyBot educational game based on its quality attributes

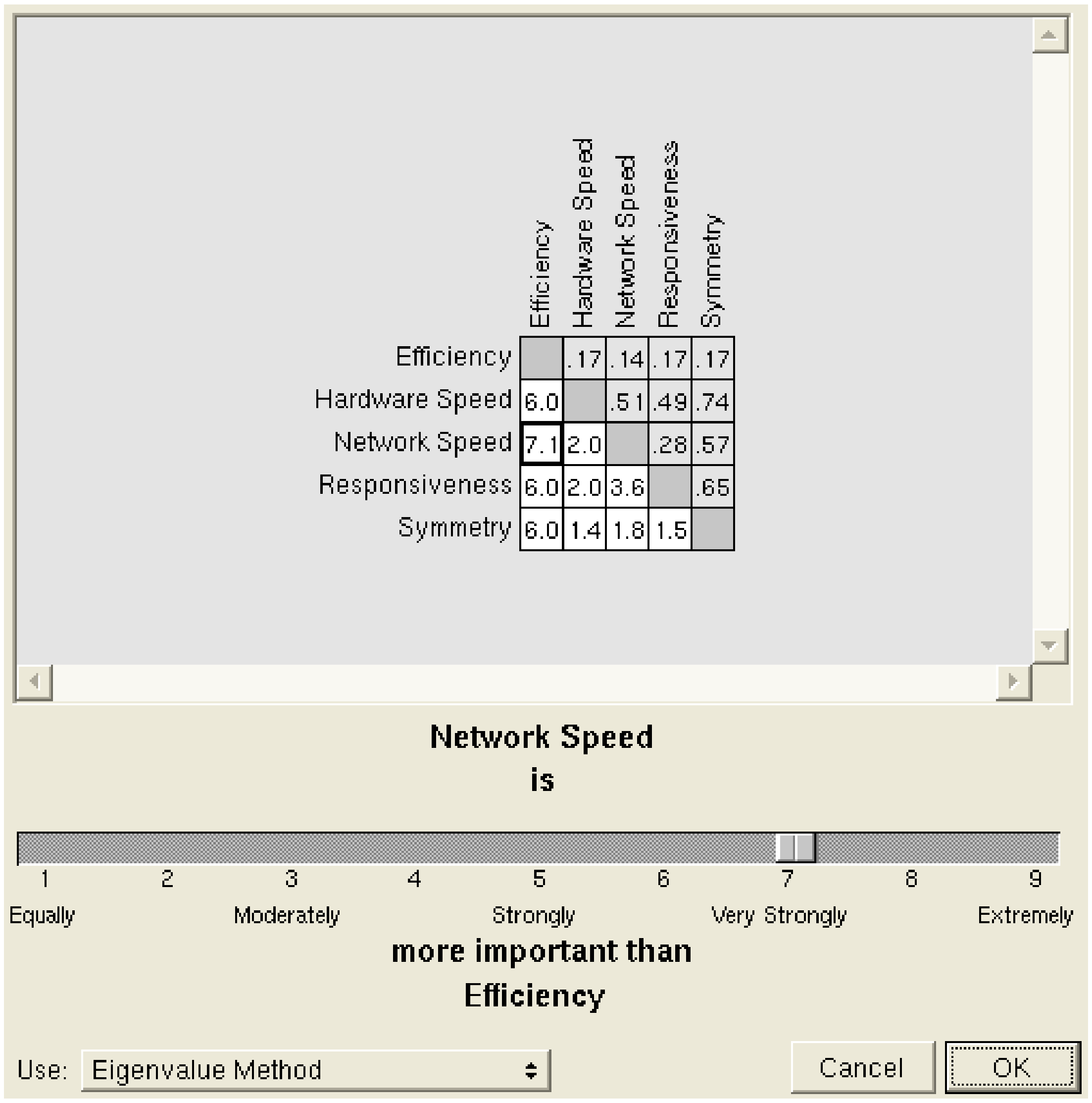
Nominal Score	Numerical Score	Description
Excellent	[81-100]	Game comprehensively supports achieving quality attribute and resolves all related tradeoffs among attributes
Good	[61-80]	Game explicitly supports achieving quality attribute and resolves major related tradeoffs
Average	[41-60]	Game supports achieving quality attribute but major tradeoffs remain to be resolved
Marginal	[21-40]	Game implicitly supports achieving quality attribute but directly conflicts with one or more quality attributes
Poor	[0-20]	Game is not suitable for achieving quality attribute

4.3.5 Step 5: Relative Criticality Weighting of Indicators

"Child indicators may influence the score of their parent indicator in different amounts" [Balci et al. 2002]. For example, Customizability may affect the DEG Software Quality more than Engagability. Moreover, a child indicator may influence the score of a parent indicator differently from the other sibling indicators. For instance, responsiveness may be much higher than efficiency in digital game performance. Therefore, child indicators must be weighted among themselves [Balci 2001].

Weights are used to express a child indicator's level of influence. "A *weight* is a fractional value between 0 and 1" [Balci et al. 2002]. The weights of the child indicators belonging to the same parent must sum to one. Since the weights are assigned relative to the sibling indicators to express criticality of influence on the parent indicator, the process is called *relative criticality weighting* [Balci 2001 and Balci et al. 2002].

We use the Analytic Hierarchy Process (AHP) to ease the relative criticality weighting process [Balci 2001]. Figure 22 shows the use of AHP for criticality weighting of the Performance indicator's child indicators.



	Efficiency	Hardware Speed	Network Speed	Responsiveness	Symmetry
Efficiency		.17	.14	.17	.17
Hardware Speed	6.0		.51	.49	.74
Network Speed	7.1	2.0		.28	.57
Responsiveness	6.0	2.0	3.6		.65
Symmetry	6.0	1.4	1.8	1.5	

Network Speed
is
1 2 3 4 5 6 7 8 9
Equally Moderately Strongly Very Strongly Extremely
more important than
Efficiency

Use: Eigenvalue Method

Figure 22. Relative Criticality Weighting of Indicators Using Analytic Hierarchy Process

Figure 23 shows a kiviat graph for relative criticality weighting of performance indicators. The kiviat graph shows that Responsiveness is the most important and Efficiency is the least important indicator in the assessment of CandyBot educational game.

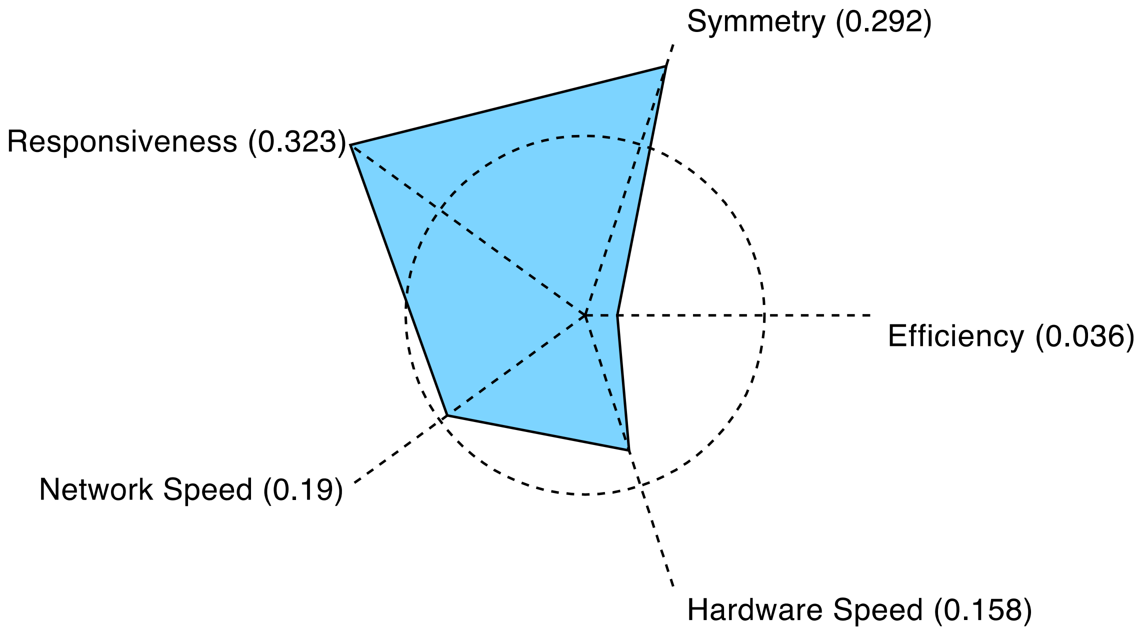


Figure 23. Kiviat Graph for Criticality Weighting of Performance

4.3.6 Step 6: Project Management Attributes

In this step administrators create attributes for the leaf indicators. These attributes can consist of status, priority, clarity and importance [Balci et al. 2002]. For example, "High" or "Low" can be used to specify the importance of evaluation.

4.3.7 Step 7: Identification of Evaluation Techniques

Determination of scores for the leaf indicators can be done in different ways as follows [Balci et al. 2002]:

- Testing
  - System Testing
  - Product Testing
- Direct Measurement
- Analysis
  - Data Analysis
  - Discrete Analysis
  - Performance Analysis
  - Risk Analysis
  - Statistical Analysis
  - Systems Analysis
- Examination
  - By domain SMEs
  - By technical SMEs

For example, SMEs can decide a score based on their domain knowledge and expertise under examination approach.

4.3.8 Step 8: Identification of Evaluators

Identify SMEs that are technical people who can examine the CandyBot game and assign a score for the leaf indicator [Balci et al. 2002].

4.3.9 Step 9: Assignment of Evaluators

Depending on the expertise of the SMEs, assign them to leaf indicators in the hierarchy [Balci et al. 2002].

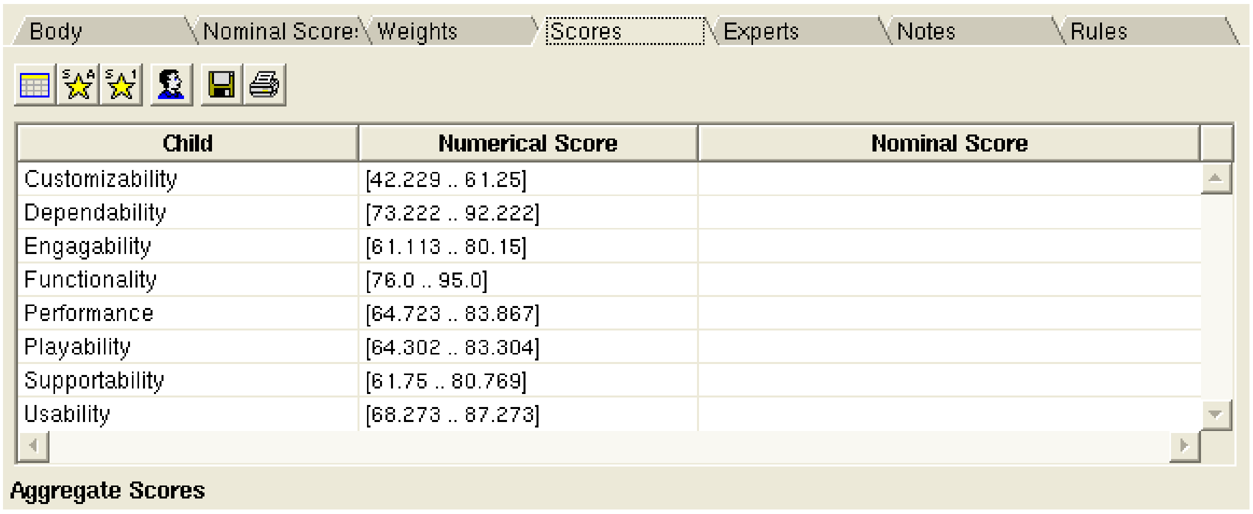
4.3.10 Step 10: Relative Criticality Weighting of Evaluators

A leaf indicator can be assessed by more than one SME and each SME's opinion may be taken into consideration with different weights. The SMEs assigned to the same leaf indicator are weighted among themselves and a weighted score is computed for the leaf indicator [Balci 2001; Balci et al. 2002]. Hence, we use the AHP approach similar to the section 4.3.5.

4.3.11 Step 11: Evaluations

An SME judges the results of testing, direct measurement or analysis and determines a score for a leaf indicator. An SME can leave notes or upload any kind of documentation including media files, slides and reports for his/her rationale for a leaf indicator.

Once all SMEs enter scores for all leaf indicators, the scores are accumulated in a bottom-up fashion from the leaf indicators to the root indicator by IDEALLY. Figure 24 shows Aggregated Scores for the top seven branch indicators.



Child	Numerical Score	Nominal Score
Customizability	[42.229 .. 61.25]	
Dependability	[73.222 .. 92.222]	
Engagability	[61.113 .. 80.15]	
Functionality	[76.0 .. 95.0]	
Performance	[64.723 .. 83.867]	
Playability	[64.302 .. 83.304]	
Supportability	[61.75 .. 80.769]	
Usability	[68.273 .. 87.273]	

Aggregate Scores

Figure 24. Aggregated Scores for Top Level Indicators

Successful execution of IDEALLY creates Kiviat graphs in Scalable Vector Graphics (SVG) to visualize the scores simultaneously.

A kiviat graph of the scores of eight indicators is depicted in Figure 25. The center of the circle shows a score of zero and the perimeter shows a perfect score of 100. Each axis from zero to the perimeter 100 displays the score range of the corresponding indicator as low score, average score, and high score. The score closest to the center are low scores. The score closest to the perimeter are high scores and the middle score is the average score.

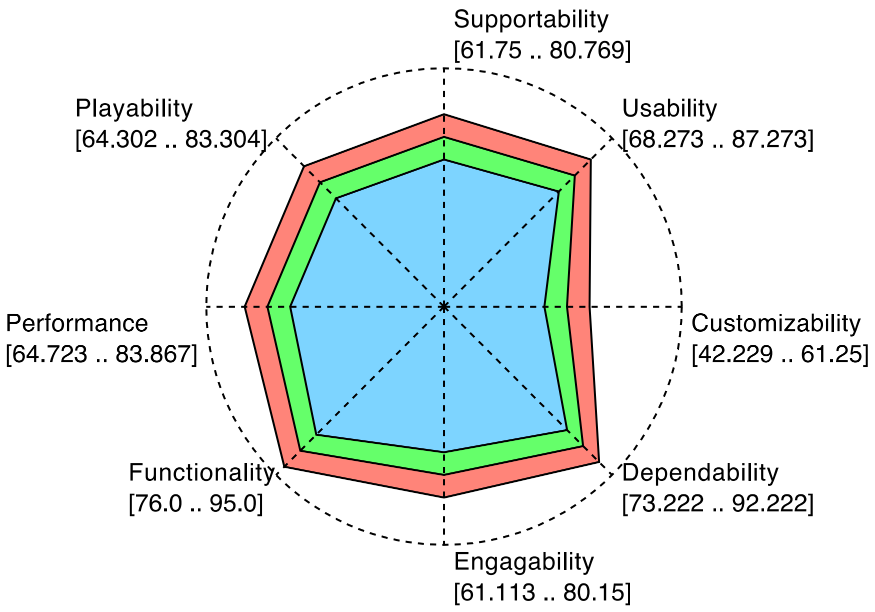


Figure 25. Kiviat Graph

CHAPTER 5: TAXONOMOY OF GAMES

This section introduces a taxonomy of games for the purpose of identifying their characteristics. This section will also assist the game developers in identifying the structure and rationale so that appropriate terminology can be used.

The following characteristics are used to classify games. It should be noted that the following list is a generalized list of characteristics.

5.1 Non-Digital Games

5.1.1 Board Games (Non-Digital)

Board games are played on specially designed boards in which players make moves based on a set of rules. Most board games require many small pieces to play and players move their pieces on the board. While most of the popular board games are based on luck; some of them are heavily based on strategy and require critical thinking [Bell 1980; Hopper and Bell 2000].

There are many different types of board games such as alignment, chess variants, paper and pencil, race, roll-and-move games. Different games come with different rules, and the difficulty of games changes based on the defined rules and the goals. Board games can be really simple such as Tic-tac-toe or can be complicated such as Dungeon & Dragons. Some of the popular board games are chess, backgammon, monopoly, tic-tac-toe, and Dungeons & Dragons.

Characteristics:

- Requires a board
- Requires equipment (e.g. rock or wooden pieces)
- Requires time limit
- Requires set of defined rules
    - Have restrictions such as the number of players in a game
- Requires more than one player to play
- Requires skills (logical skills to make strategic moves)

5.1.2 Card Games (Non-Digital)

Card games use playing cards as the main gaming instruments. The playing cards are mostly identical in size and shape and generally form a deck. Depending on the game, the number of cards and the number of players, the game rules can change. Culture and region highly effects the game rules. The same card game can be played in different ways in different regions. Only a small number of card games have standardized rules.

There are a variety of card games and some of them are as follows: trick-taking, matching, shedding, accumulating, fishing, comparing, solitaire (patience), drinking card, multi-genre, collectible card, casino or gambling card, poker, and fictional card games. Some of the popular card games are bridge, whist, blackjack, and poker [McLeod 2013].

Characteristics:

- Requires a deck of cards (e.g. plastic or paper)
- Requires more than one player to play
- No time limitation

- Requires set of defined rules
    - Culture and region effects the game rules
- Limited number of players in a game or the limited number of hands to complete a game
- Require shuffling the playing cards for each game
- Essential skills (logical skills to make strategic moves)
- Direction of play
    - Clockwise
    - Counter-clockwise

5.1.3 Educational Games (Non-Digital)

Educational games are designed to teach specific subjects or skills. Many of the educational games focus on teaching certain subjects, presenting topics that are difficult to understand, and assisting learners for different skills. Non-Digital educational games do not require any digital device or digital equipment to play. They can be played with cards, boards, sticks, and balls.

Characteristics:
- Require clear learning goals
- Require a systems of rewards
- Include narrative context
- Provide relevant learning content
- Provide interactive tutorial
- Provide immediate feedback
- Require non-digital equipment

5.1.4 Sports Games (Non-Digital)

Sport is a physical activity that is played according to rules for entertainment. Each sport uses a distinct set of rules. For example, soccer requires players to make goals to get a point and the baseball requires players to run all the bases and go back to home base that equals to one run. There are two different types of sports, and they can be categorized as table and field games.

5.1.4.1 Table Games (Non-Digital)

Table games are a type of sport that require a table to play. The table can be made of wood, marble or plastic. It generally requires more than one player to play. Many of the table games require equipment such as a racket, net and stick to play. Some of the popular table games are table tennis, pool, dominos, and foosball [Bell 1980; Hopper and Bell 2000].

Characteristics:
- Requires specific type of table (e.g. wood, plastic, and rock)
    - Game table can have different shapes (e.g. rectangle or circle)
- Requires equipment (e.g. racquet or stick)
- Mostly requires limited time but not necessarily
- Requires body management skill includes run, stop, and turn
- Requires physical skills including running, stopping, turning and equipment handling abilities such as sending-away (throwing or striking) and receiving (collecting and catching)

5.1.4.2 Field Games (Non-Digital)

Field games are a type of sport that require a field to play. Fields can be composed of wood, grass, metal, and ice. Field games require more than one player to play. Teams can vary in size from two to dozens. Field games require different equipment to play such as balls, sticks, goals and baskets. Some popular field games are baseball, basketball, football, soccer, and ice hockey [Hopper and Bell 2000].

Characteristics:

- Requires specific type of field (e.g. wood or grass).
  - Fields can have different shapes (e.g. rectangle or square)
- Requires specific equipment (e.g. ball, goal, basket, and stick)
- Requires more than one player
- Requires limited time
- Requires physical skills including running, stopping, turning and equipment handling abilities such as sending-away (throwing or striking) and receiving (collecting and catching)

5.1.5 Role-Based Games (Non-Digital)

A role-based game is an interactive and collaborative game in which players are required to physically act or make decisions. There are different types of role-based games. For example, the tabletop games are played through a discussion in a small group of people, which does not require any physical activity. However, live action games require players to physically perform their roles by wearing costumes and using different tools [Harrigan and Wardrip-Fruin 2007; Cover 2010].

Characteristics:

- Highly interactive and collaborative
- Mostly requires storyline
- Players act as a special character
- Requires more than one player
- Mostly requires costumes or tools
- Different themes and rules for different games
- Requires body management skills including performing, interacting between players and equipment handling skills using special tools

5.2 Digital Games

5.2.1 Role-Playing Games (Digital)

In Role-playing (RPG) games, players go through numerous adventures and explore new worlds, which are mostly fictional. While completing different missions and defeating enemies, the game character gains powers that increase the character’s skills and abilities. The game character can be one person or a group of people. The subgenres of RPGs are action, tactical, and massively multiplayer online RPGs. There are also combinations of games, which include features of RPG and other game types. Some of the popular RPGs are Dungeon & Dragons and Final Fantasy [Adams and Rollings 2003; Adams and Rollings 2010].

Characteristics:

- Rely on a story
  - The game world is set in a fantasy or science fiction world
  - Players enjoy the story of the game
- Exploration and quests

    - Exploring the game world is one of the main actions
    - Different game maps are provided
    - Players are required to complete different quests to complete the game
- Items and inventory
    - Found items throughout the game can be collected and stored in an inventory
- Character actions and abilities
    - Mostly the player moves the game character (In some cases AI can help players to move their game character)
    - The abilities of the game character have importance to succeed the mission
    - The improvement of character abilities/skills increases the player's chances to win/complete a mission
- Experience and levels
    - Experience is generally gained by defeating enemies
    - Gaining new skills, abilities make the player's game character more powerful
- Combat
    - Combat takes an important place
    - Mostly the game creates and presents enemies in the game world
    - An enemy is created when a player enters to a certain location/level
- Graphics and interface
    - Player navigates the game world from the first or third person perspective in 2D/3D environment.
    - Isometric and top-down perspective is commonly used.

5.2.2 Action Games (Digital)

"An action game is one in which the majority of challenges presented are tests of the player's physical skills and coordination. Puzzle-solving, tactical conflict, and exploration challenges are often present as well" [Adams and Rollings 2010]. Action games require quick reflexes and reactions from the player. Players are not required to make strategic plans for a mission. Levels can be completed by overcoming individual challenges. Most of the popular arcade games are also considered as action games. The action games' subgenres are beat'em up, fighting, maze, platform, rhythm, and shooter games. Street Fighter and Dead or Alive are well known action games [Adams and Rollings 2003; Adams and Rollings 2010]

Characteristics:

- Levels
    - Provides countless levels to complete.
    - The current level needs to be completed to move on to the next level
- Character abilities
    - The game characters have special skills (e.g. punching or kicking)
    - The player is required to collect items
    - The player can improve character's ability
- Obstacle and enemies
    - The player encounters many enemies, obstacles, and traps
    - The enemy strength increases once the player reaches the next level
    - At the end of each level, players encounter a strong enemy
- Health and lives
    - The player has a certain amount of lives that is decreased by hits
- Graphics and interface
    - 2D or 3D

- Scoring and victory
    - Provides simple goals
    - Requires beating the final enemy at the end of the game
    - The players' score is saved for future games
    - Points are earned when a task is completed
    - Offers bonus for skillful play

5.2.3 Adventure Games (Digital)

"An adventure game is an interactive story about a protagonist character who is played by the player. Storytelling and exploration are essential elements of the game. Puzzle solving and conceptual challenges make up the majority of the gameplay. Combat, economic management, and action challenges are reduced or nonexistent" [Adams and Rollings 2010]. In adventure games, the player highly interacts with the game. The characters become the gamer himself. Throughout the whole game the story becomes the player's life. The subgenres of adventure games are text, graphic, and puzzle games. Some of the most popular adventure games are Collosal Cave Adventure, Zork, Indiana Jones and the Emperor's Tomb [Adams and Rollings 2003; Adams and Rollings 2010].

Characteristics:

- Puzzle solving
    - Requires decoding messages, finding items, exploring certain locations
- Gathering and using items
    - Requires inventory management
    - The players are required to use items from their inventory to complete certain missions
- Story, settings, and themes
    - Generally single-player games
    - Highly story-driven
    - Uses immersive environment or fantasy worlds
- Dialogues
    - Includes many conversations between player and non-player characters
    - Dialogues and conversations can enable players to make decisions for their future moves
- Goals, success and failure
    - The goal is the completion of assigned missions
    - High scores are considered as achievements
- Graphics and interface
    - Player navigates the game world from the first or third person perspective in 2D/3D environment.
    - Isometric and top-down perspectives are common

5.2.4 Strategy Games (Digital)

"A strategy game is one in which the majority of challenges presented are strategic conflict challenges and the player may choose from a large variety of potential actions or moves at most points in the game. Victory is attained by superior planning and taking the optimum actions; the element of chance must not play a large role. Other challenges, such as tactical, logistical, economic, and exploration challenges, may also be present. Physical coordination challenges play little or no part," [Adam and Rollings 2010]. Strategy games require a heavy thinking process by the players. The player needs to carefully plan, make decisions and take actions. The subgenres of strategy games are 4X, artillery, real-time strategy, real-time tactics, turn-based strategy, turn-based tactics, and war-games. Some of the most popular strategy games are StarCraft, Age of Empires, Box Wars.

Characteristics:

- Units
  - Use units, which is used for many purposes such as defending a city or attacking to enemy, taking care of farms, lands, mines and many other purposes.
  - Theme units can be people, tanks, a group of buildings or some other type of tools
- Economy, resource management, upgrades
  - Include economic challenges
  - Players are required to collect resources and use them to create different types of units
  - Resources and the economy management effects unit's upgrade
- Map and exploration
  - Isometric perspective is used for camera angle
  - Game maps are required to be explored for new resources or enemies
- Themes
  - Themes show varieties such as world war, naval war, and management.
- Goal
  - Completing assigned duties
    - Requires defeating an enemy
    - Requires collection of resources and reaching the targeted economical growth
- Graphics and interface
  - 2D or 3D

5.2.5 Sports (Digital)

"A sports game simulates some aspect of a real or imaginary athletic sport, whether it is playing in matches, managing a team or career, or both. Match play uses physical and strategic challenges; the management challenges are chiefly economic," [Adams and Rollings 2010]. Sports games enable users to play on their favorite team with their favorite sports players. This makes sports games one of the most popular game categories. There are so many sports games in real life as well as in the digital game world. Some of the most popular sports games are baseball, basketball, boxing, football, golf, ice hockey, soccer, swimming, wrestling, and tennis. Some of the most popular digital sport games are FIFA Soccer, NBA basketball, Championship Manager [Adams and Rollings 2003; Adams and Rollings 2010].

Characteristics:

- Game Structure
  - Simulating the game as realistically as possible
- Game Play and Rules
  - Game rules follow the actual sports rules
- Competition Modes
  - Provide all possible modes of competition
    - Single player, multi-player, collaborative or in a league environment
- Victory and loss conditions
  - Follows the same rules as in the real sports games
- Graphics and interface
  - 2D or 3D

5.2.6 Artificial Life Games (Digital)

Artificial life games are a simulation of real life. The game characters can be a pet, person, or any kind of living opponent. Artificial life games enable players to control the lives of autonomous creatures or people.

Subgenres of artificial life games are artificial pets, second life games, and god games. Some of the most popular artificial life games are The Sims and Nintendogs from Nintendo [Adams and Rollings 2003; Adams and Rollings 2010].

Characteristics:

- Needs
  - The game character is required to do certain activities such as feeding, resting, working
- Skills
  - The game character is required to be taught skills such as cooking, repairing or doing some certain activities
  - Taught skills can affect the game character's life in both negative and positive ways
- Personalities
  - The game character is supposed to simulate a relationship with the player
- Victory, lose
  - The game continues as long as the game character is happy.
  - Unhappiness of the game character mostly results with failure
- Graphics and interface
  - 2D or 3D

5.2.7 Puzzle Games (Digital)

Puzzles can be found in so many game genres such as actions, and adventures. The main activity of puzzle games is solving a puzzle. Puzzles can be based on recognizing patterns, making deductions or understanding processes. Although there are different types, the goal of the puzzle games always remains the same. The subgenres of puzzle games are action puzzle, hidden object, reveal the picture, physics, tile-matching and traditional puzzle games. Some of the most popular puzzle games are Minesweeper, The Splatters, and Tetris [Adams and Rollings 2003; Adams and Rollings 2010].

Characteristics:

- The main activity is solving a puzzle
- They can be imbedded in a story
- Presents numerous levels/challenges to be completed
- Provides clues/tips to players so that they can complete the puzzle
- Graphics and interface
  - 2D or 3D

5.2.8 Educational Digital Games (Digital)

Educational games are designed to teach people certain subjects, improve their skills, and expand concepts. They require digital equipment and devices to play such as controllers, and digital screens. The educational games focus on problem-solving, strategizing, and learning outcomes more than entertaining players. There are three different types of educational digital games which are as follows: simulation, creative expression, and teaching games. [Dondlinger 2007; O'Brien 2010; Freitas 2006]

Characteristics:

- Requires clear learning goals
- Requires systems of rewards
- Requires narrative context
- Provides relevant learning content
- Provides interactive cues

- Provides immediate feedback
- Requires digital equipment and devices
- Graphics and interface
    - 2D or 3D

5.2.9 Training Games (Digital)

Training games are interactive games that focus on teaching and training. They are generally used for defense, education, scientific exploration, health care, emergency management, city planning, and engineering. Training games can be used to gain various types of skills. Mostly, they are simulations of real world events and they are designed to teach domain specific activities. Training games are not entertainment games; their focus is to educate and train players in an exciting way [Peixoto et al. 2011; Transue 2013].

Some of the training games are as follows: business, management, economics, flight simulation, and scientific experimental games.

Characteristics:

- Requires clear goals
- Requires problem solving activities
- Requires rules and strategies
- Requires practice and repeat
- Requires players to highly participate
- Provides immediate feedback
- Provides a rewarding mechanism
- Represents a real world system which is embedded in a virtual world
- Includes game features such as fantasy and mystery genres
- Graphics and interface
    - 2D or 3D

The characteristics listed above are generalized characteristics. We identified them based on careful analysis of numerous different games.

5.3 Game Classification

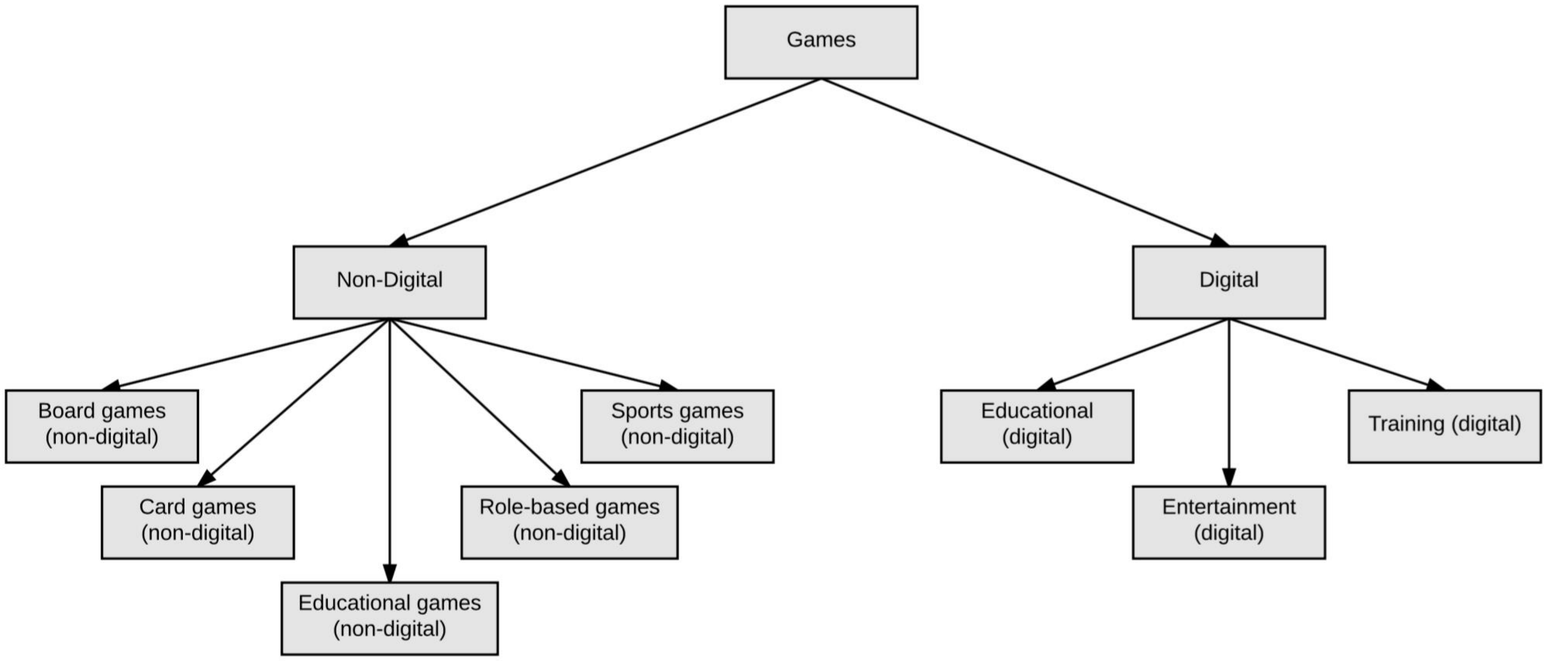

Games
Non-Digital
Digital
Board games (non-digital)
Card games (non-digital)
Educational games (non-digital)
Role-based games (non-digital)
Sports games (non-digital)
Educational (digital)
Entertainment (digital)
Training (digital)

Figure 26. Taxonomy of games

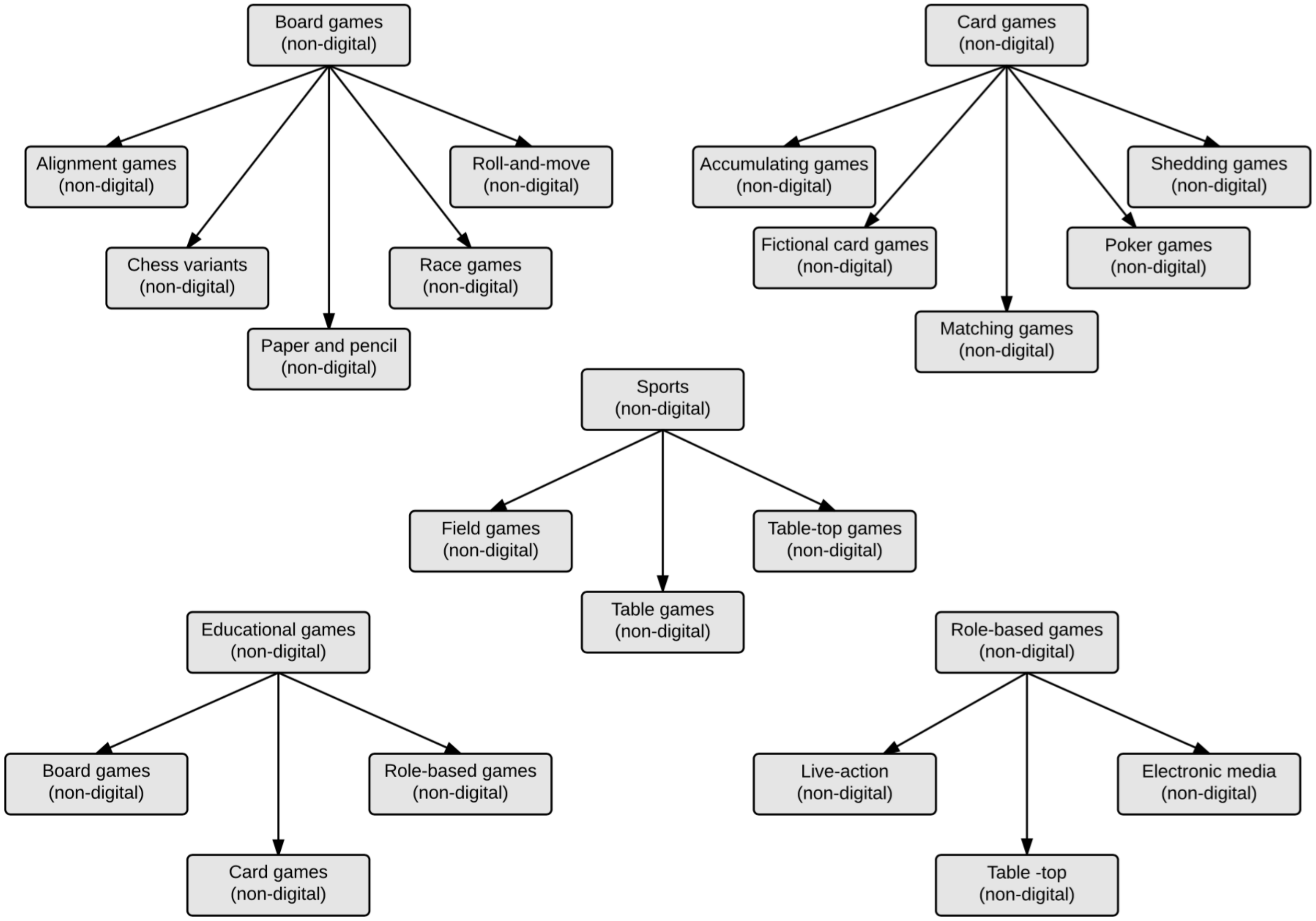

Board games (non-digital)
Alignment games (non-digital)
Chess variants (non-digital)
Paper and pencil (non-digital)
Race games (non-digital)
Roll-and-move (non-digital)
Card games (non-digital)
Accumulating games (non-digital)
Fictional card games (non-digital)
Matching games (non-digital)
Poker games (non-digital)
Shedding games (non-digital)
Sports (non-digital)
Field games (non-digital)
Table games (non-digital)
Table-top games (non-digital)
Educational games (non-digital)
Board games (non-digital)
Card games (non-digital)
Role-based games (non-digital)
Role-based games (non-digital)
Live-action (non-digital)
Table -top (non-digital)
Electronic media (non-digital)

Figure 27. Taxonomy of games (Continued)

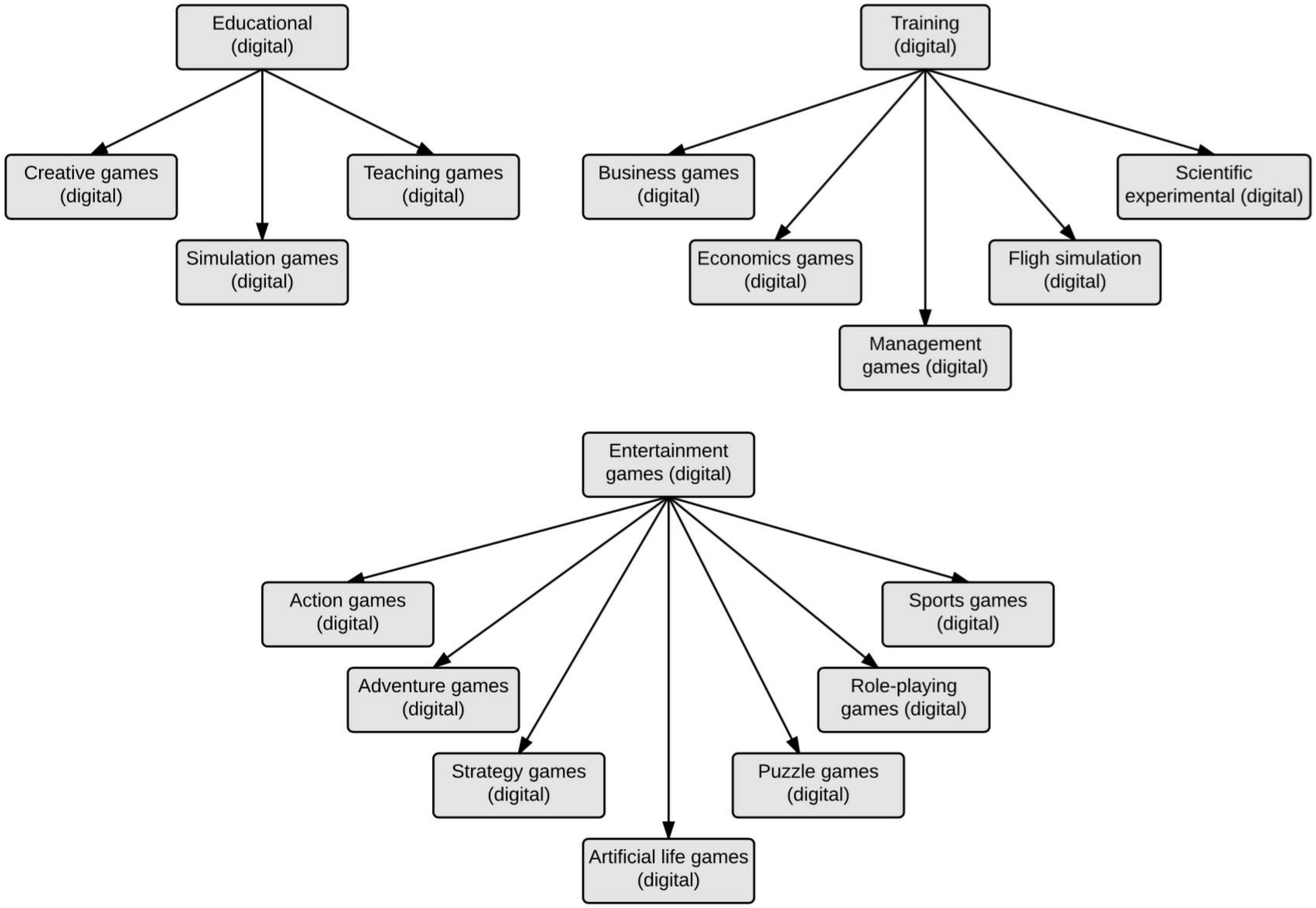

Educational (digital)
Creative games (digital)
Simulation games (digital)
Teaching games (digital)
Training (digital)
Business games (digital)
Economics games (digital)
Management games (digital)
Fligh simulation (digital)
Scientific experimental (digital)
Entertainment games (digital)
Action games (digital)
Adventure games (digital)
Strategy games (digital)
Artificial life games (digital)
Puzzle games (digital)
Role-playing games (digital)
Sports games (digital)

Figure 28. Taxonomy of games (Continued)

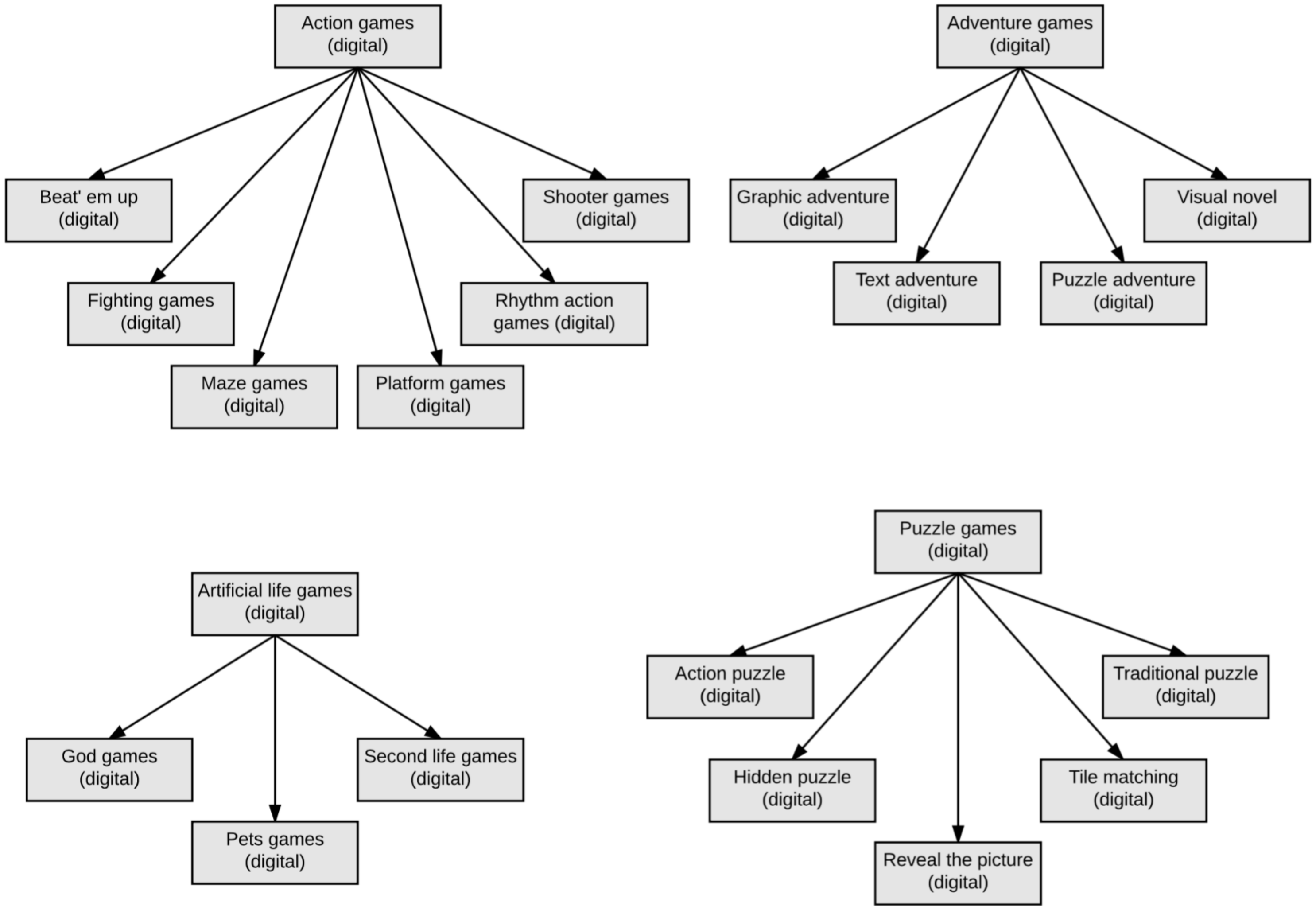


Figure 29. Taxonomy of games (Continued)

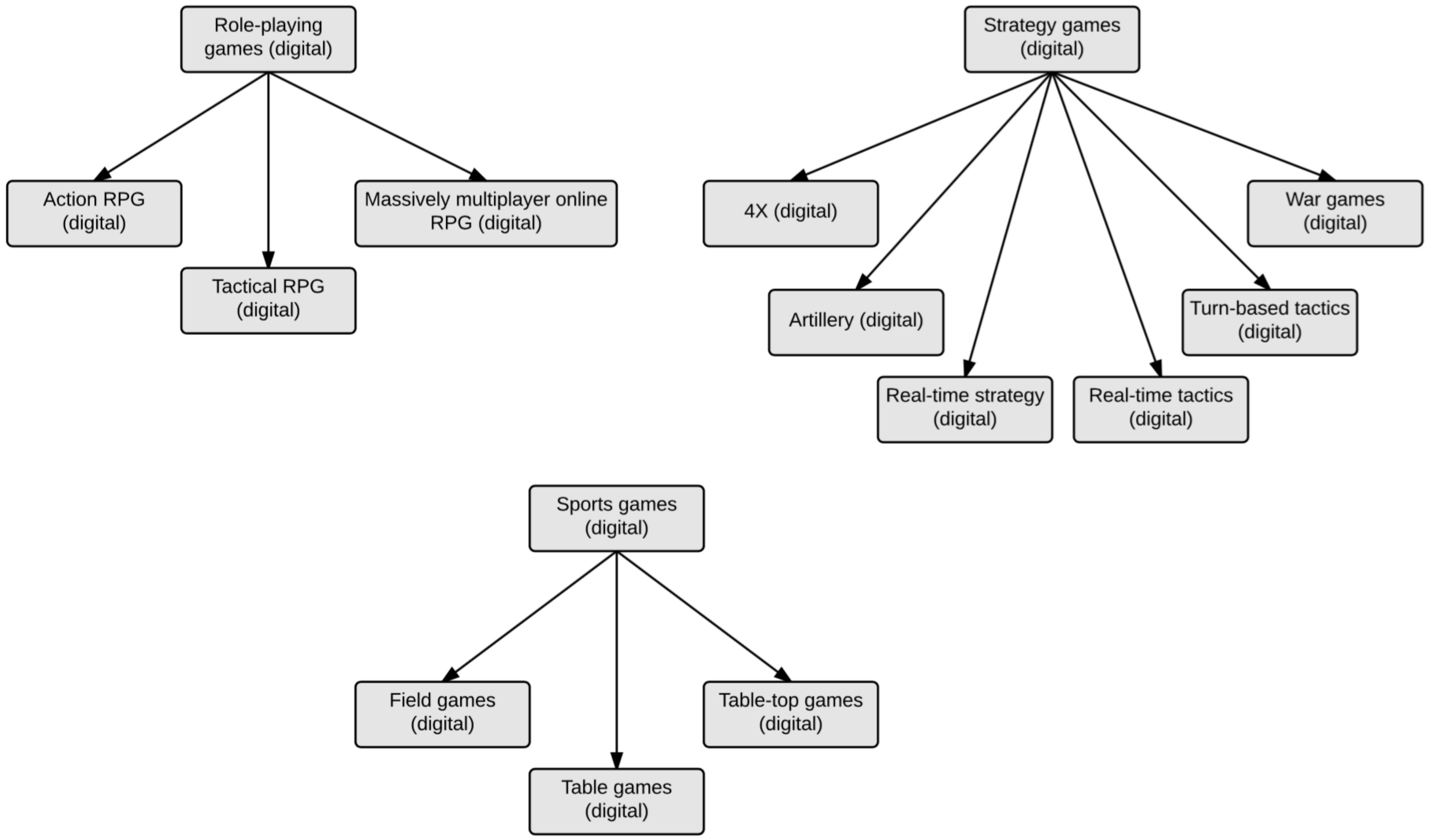


Figure 30. Taxonomy of games (Continued)

CHAPTER 6: DIGITAL EDUCATIONAL GAMES DEVELOPED FOR iOS MOBILE DEVICES

We developed five different games to teach different mathematical concepts for middle school students. These games support developing a model to research, and implement the impacts on student engagement as an innovative educational application in the mathematics curriculum based on advances in software and mobile computing devices. We designed and created these games by using principles from the learning sciences, developmental psychology, computer science, and mathematics pedagogy to achieve the most effective teaching and learning applications.

GAMED is used to develop CandyFactory Educational Game for iPad, CandySpan, CandyDepot, and CandyBot. Table 7 presents the list of digital educational games that we developed and published for mobile devices.

Table 7. The list of digital educational games developed for mobile devices

App Name	Publishing Date
Candy Factory Educational Game	July 14, 2011
Candy Factory Educational Game for iPad v1.0	June 13, 2012
CandySpan	January 20, 2013
CandyDepot	August 27, 2013
Candy Factory Educational Game for iPad v2.0	April 30, 2014
CandyBot	April 2, 2015

6.1 CandyFactory Educational Game

6.1.1 Description

CandyFactory is an educational game teaching the concept of fractions to middle school students. It supports more powerful conceptions of fractions. “By coordinating actions of partitioning (slicing) and iterating (copying) candy bars, students learn to perceive fractions as parts relative to the whole. The idea is that students begin to understand partitioning and iterating as inverse operations: If students partition the whole into *n* parts, they can reproduce the whole by iterating any one of those parts *n* times. Hence, they can establish unit fractions as 1-to-*n* size relations with the whole. Likewise, students begin to understand non-unit fractions, *m*/*n*, as *m* iterations of 1/*n*; i.e., a fractional part that is *m* times as big as 1/*n”* [Aslan et al. 2011; LTRG 2014a].

The CandyFactory Educational Game is developed as a Universal application, a single application that is optimized for iPhone, iPod touch, and iPad devices, and it can be downloaded from Apple App Store [Aslan et al. 2011].

The game levels:

- *Level 1:* “Teaches Proper Fractions as Part-Whole concept” [Aslan et al. 2011]. A *proper fraction* is a fraction in which the numerator is smaller than the denominator.

- *Level 2:* “Teaches Proper Fractions as Whole concept” [Aslan et al. 2011].

- *Level 3:* “Teaches Improper fractions, as well as Proper Fractions as Whole concept” [Aslan et al. 2011]. An *improper fraction* is a fraction in which the numerator is larger than the denominator [Aslan et al. 2011].

6.1.2 Functionality

In this game, the player’s task is manufacturing the candy bar ordered by a customer. The game starts with a customer demand for a certain amount of candy. The player is given a whole candy bar, and is expected to reproduce the order by partitioning and then iterating. The level 1 focuses on the familiar part-whole concepts for students to gain understanding of the task. Once the level 1 is completed, the players can move on to level 2 and 3. In level 1, the candy bar is displayed with existing part markings, which makes it easier for the user. However, in level 2 and 3 the candy bar is continuous, without any markings. Leve 2 and 3 require players to make rough estimates of the relative sizes of the bars using partitioning and splitting operations [Aslan et al. 2011].

Figure 31 shows four different components of the main screen: Level 1 button, Level 2 button, Level 3 button and Information button. Each button enables players to launch a game based on the chosen level. The Information Button displays the information screen that defines each level.

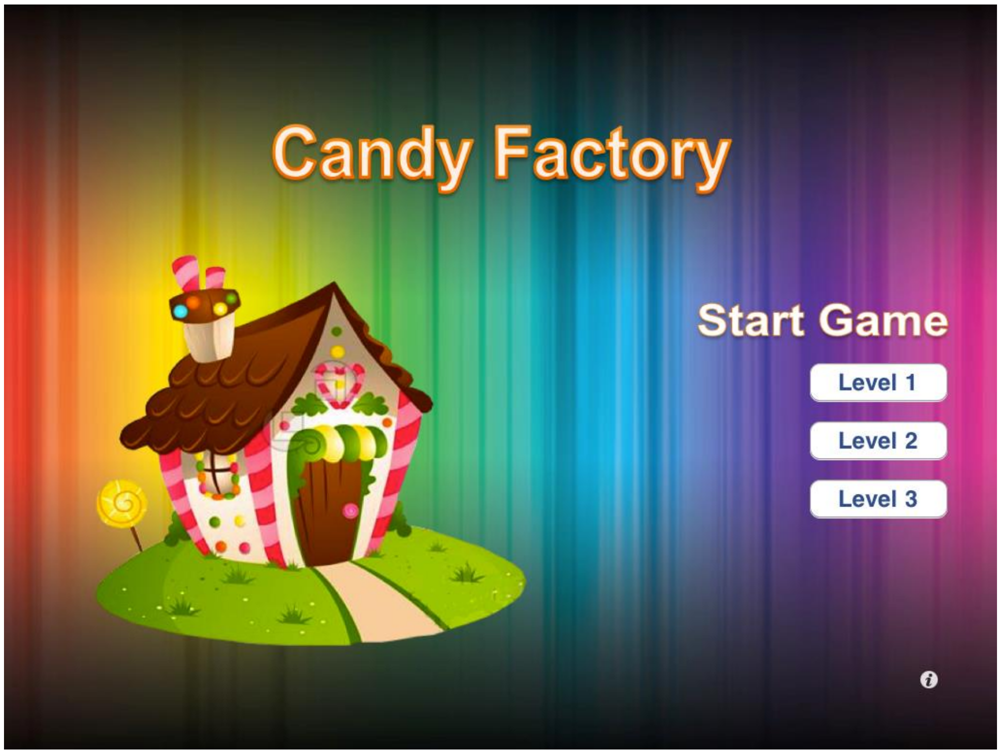


Figure 31. CandyFactory Educational Game main screen

Figure 32 shows the game screen where all players’ actions take place such as acquiring customer orders, getting the candy bar from the warehouse, partitioning, iterating, measuring, and shipping the candy to satisfy customer’s order.

Figure 33 shows the Warehouse display of the whole candy bars available. Depending on the game level selected, the warehouse candy bar list changes as follows:

1. If Level 1 is selected, the application displays the six different whole-part candy bars as shown in Figure 33.

2. If Level 2 or Level 3 is selected, the application displays five different continuous candy bars.

The player is expected to select a candy bar from the warehouse list so that it matches the one ordered by the customer. Figure 34 shows the portioning animations and Figure 35 shows the iterating animation. If players make a mistake, then they have ability to recreate the customer order.

When the player determines that the manufactured candy is an exact match with the customer order, s/he drags and drops the manufactured candy on top of the "Ship Candy" icon. Then, the application displays a message informing the player as either successful or unsuccessful upon the ending of the game, as shown in Figure 36.

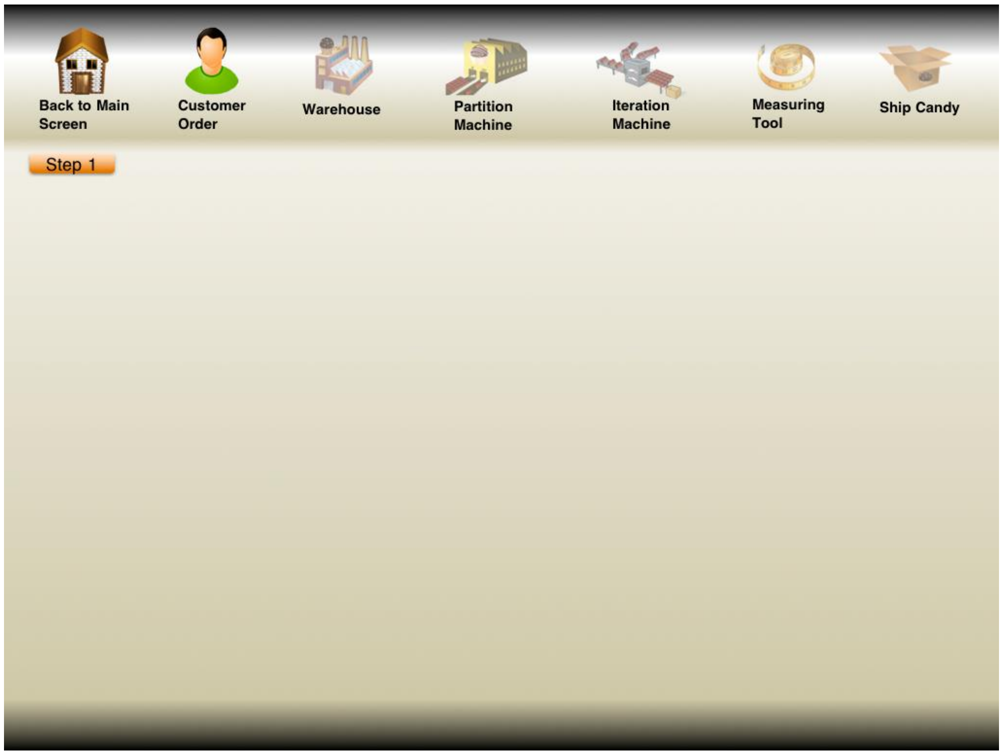


Figure 32. CandyFactory Educational Game game screen

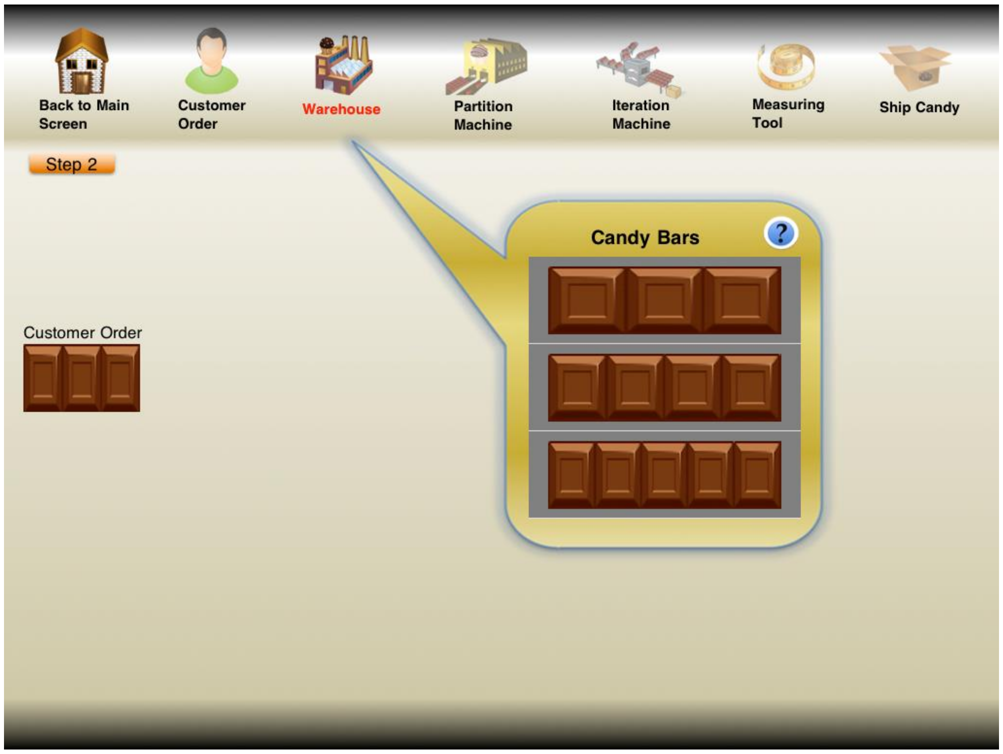


Figure 33. CandyFactory Educational Game candy bar selection screen

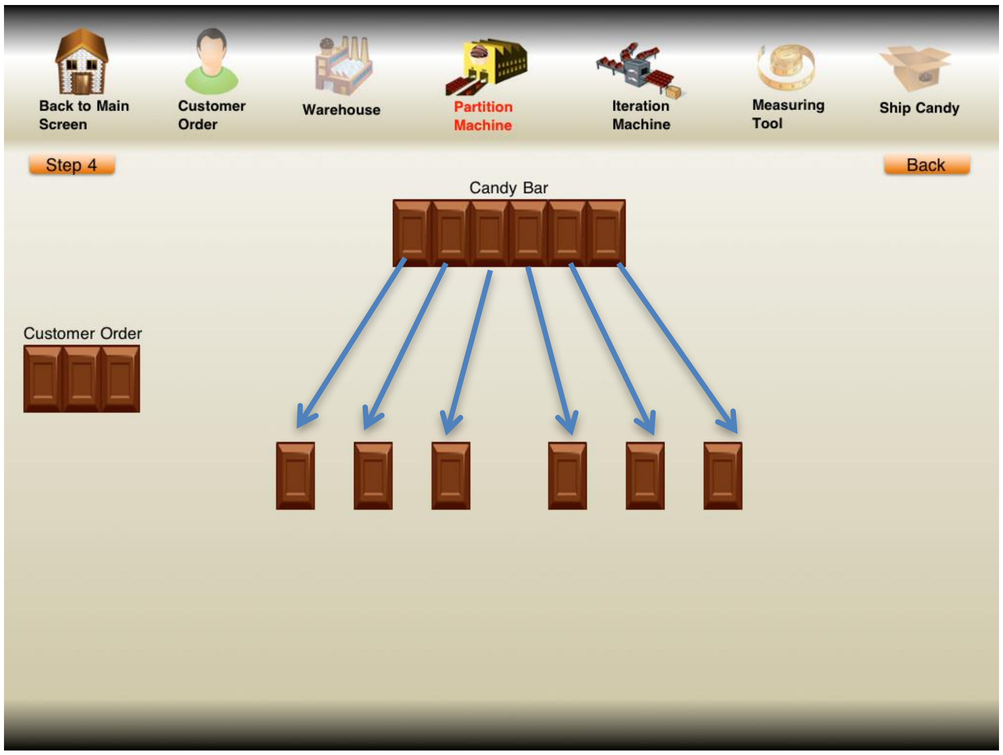


Figure 34. CandyFactory Educational Game candy bar partitioning screen

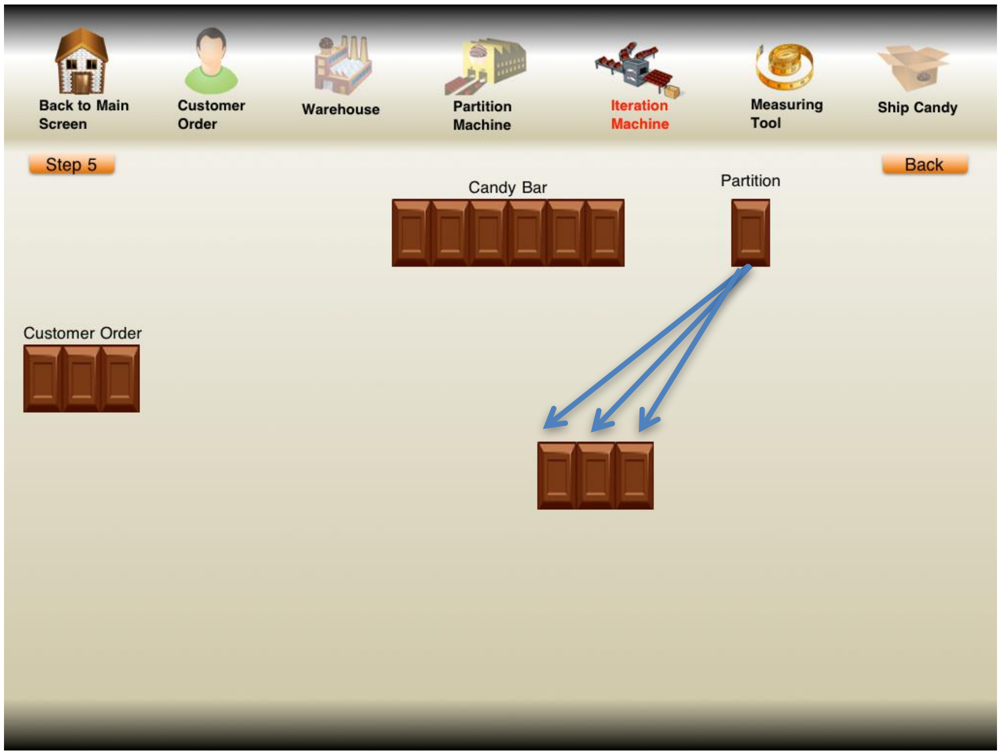
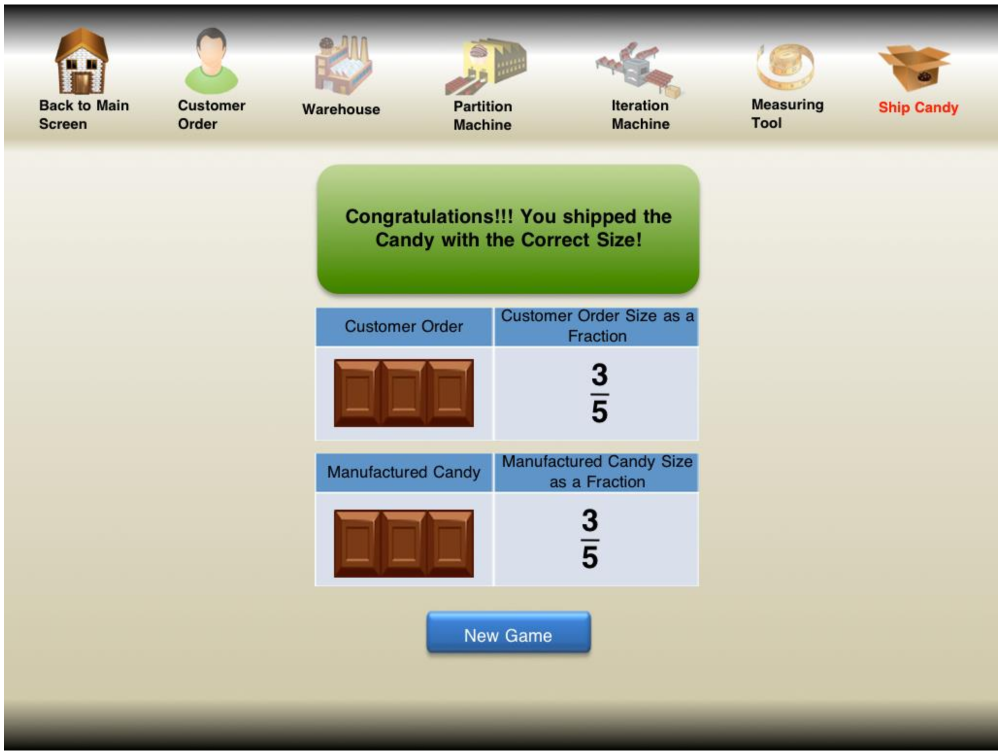


Figure 35. CandyFactory Educational Game candy bar iteration screen



Congratulations!!! You shipped the Candy with the Correct Size!

Customer Order	Customer Order Size as a Fraction
	$\frac{3}{5}$
Manufactured Candy	**Manufactured Candy Size as a Fraction**
	$\frac{3}{5}$

Figure 36. CandyFactory Educational Game successful order shipment screen

6.2 CandyFactory Educational Game for iPad

6.2.1 Description

CandyFactory Educational Game for iPad is an updated version of the Candy Factory Educational Game. Based on the knowledge and experience that we have gained with the first game, we updated the user interface, game play, and created new levels. Similar to the previous game, we use the same idea of partitioning and iterating over candy bars to teach fractions to the middle school students [Aslan et al. 2012; LTRG 2014a].

The CandyFactory Educational Game for iPad can be downloaded from the Apple App Store [Aslan et al. 2012].

The game consists of five levels:

- Level 1: "Partitions are visible in the whole and the customer order is always a proper fraction (m/n, where m is less than n). This way, students can get oriented to the game while relying on only part-whole concepts (m/n as m parts out of n equal parts in the whole)" [LTRG 2014a].

- Level 2: "Partitions are no longer visible in the whole, but the customer order is always a unit fraction (1/n). At this level, students can practice slicing the whole with finger swipes. Then, they begin to understand 1/n as the unit fraction that fits into the whole n times" [LTRG 2014a].

- Level 3: Partitions are no longer visible in the whole and the customer order can be any proper fraction (m/n, where m is less than n). Students should begin to understand m/n as m copies of 1/n and as a size relative to the whole [LTRG 2014a].

- Level 4: Partitions are no longer visible in the whole and the customer order can be any fractions, including improper fractions (m/n, where m can be greater than n) [LTRG 2014a].

- Level 5: "This level is the reverse of Level 4. Students are given a fraction (proper or improper) and asked to produce the whole from it. For example, given a piece that is m/n of the whole, students need to slice the given piece into m parts and make n copies of that piece" [LTRG 2014a].

6.2.2 Functionality

Figure 37 shows six different components of the main screen: How to Play, Play, Achievements, Options, and About buttons. Each button takes players to the corresponding game screen. For example, tapping on the Achievements button takes player to the Achievements screen, listing all the collected trophies and money as depicted in Figure 38. Tapping the sound button turns On and Off the game sound.

Candy Factory Educational game for iPad adds two more game levels to the previous game. Players need to successfully finish the previous level to move to the next one, which gradually becomes more challenging. Figure 39 displays the five different levels in the game.

Upon selection of one of the levels, the game shows an interactive tutorial for the player. The tutorial walks the player through each step and provides detailed description and an animation of each task. In Figure 40, the President Carmelo informs the player about the customer order. In Figure 41, she presents an animation of slicing and informs the player to tap on the green button when the action is successfully completed.

When the player's shift is over, the game displays the Shift log screen. The Shift log screen provides a performance report for the player. The report includes the overall customer satisfaction, earned bonus money, a verbal representation of the player's score, and each shipment result with the corresponding completion time. Moreover, the Shift log screen enables the player to share his/her performance report as depicted in Figure 42.

The player can control the game settings by using the Options screen, which can be seen in Figure 43. Figure 43 shows the Options screen that gives the player to control the game settings. For example, the player can turn the game background sound off and still keep the game action sounds on. Similarly, s/he can lock or unlock levels and can reset the game data.

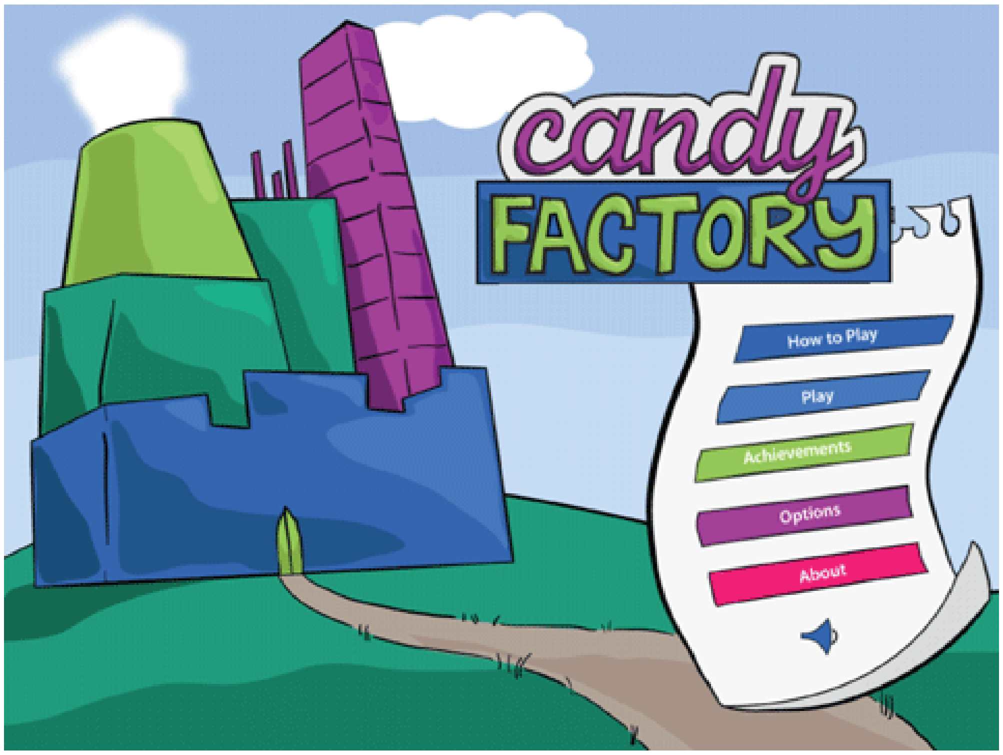


Figure 37. CandyFactory Educational Game for iPad main screen

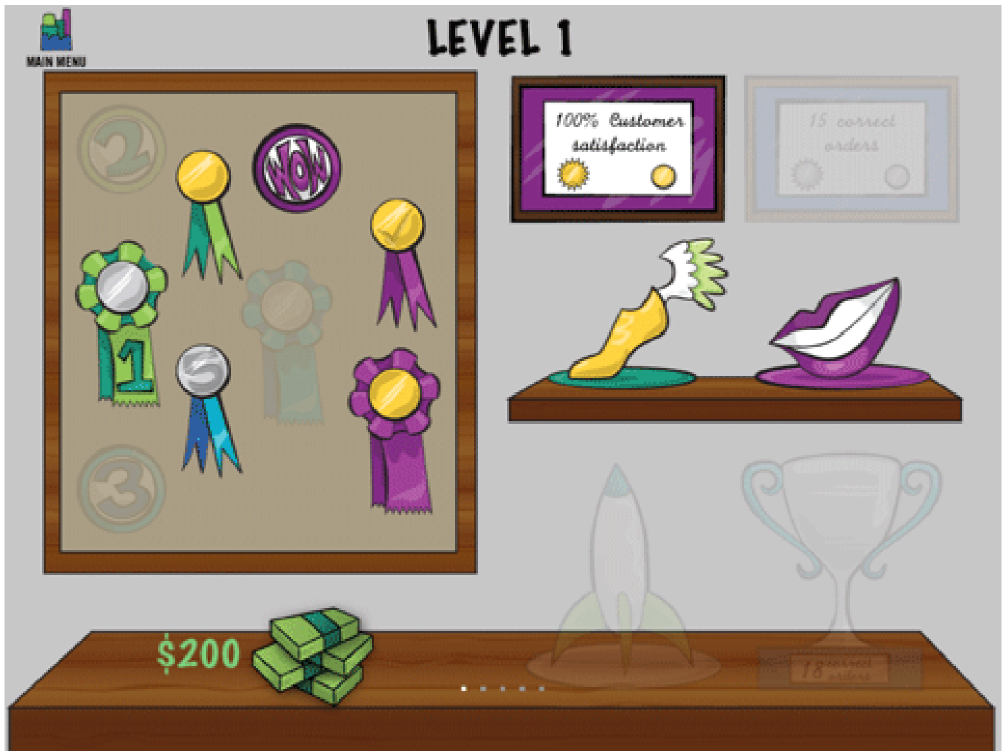


Figure 38. CandyFactory Educational Game for iPad achievements screen

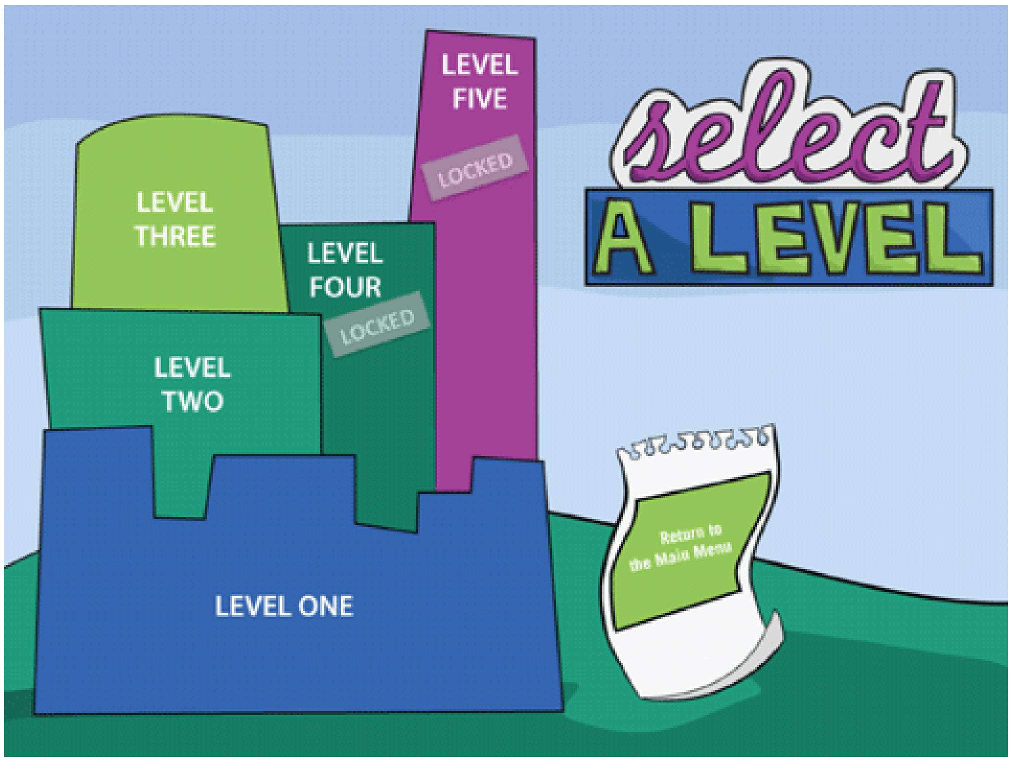
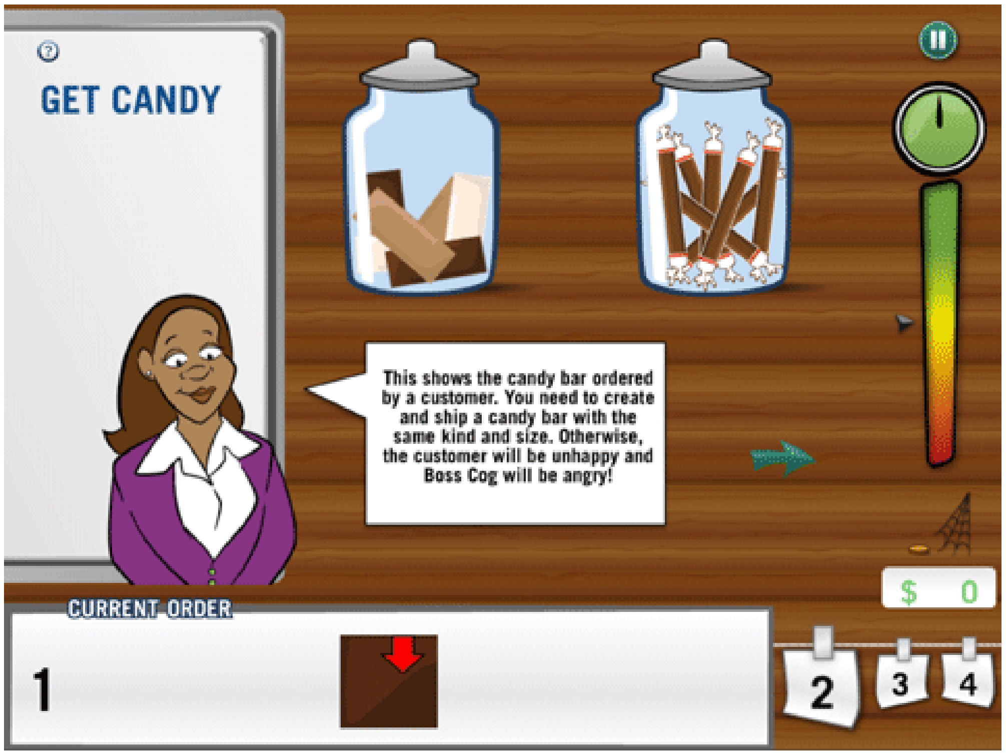
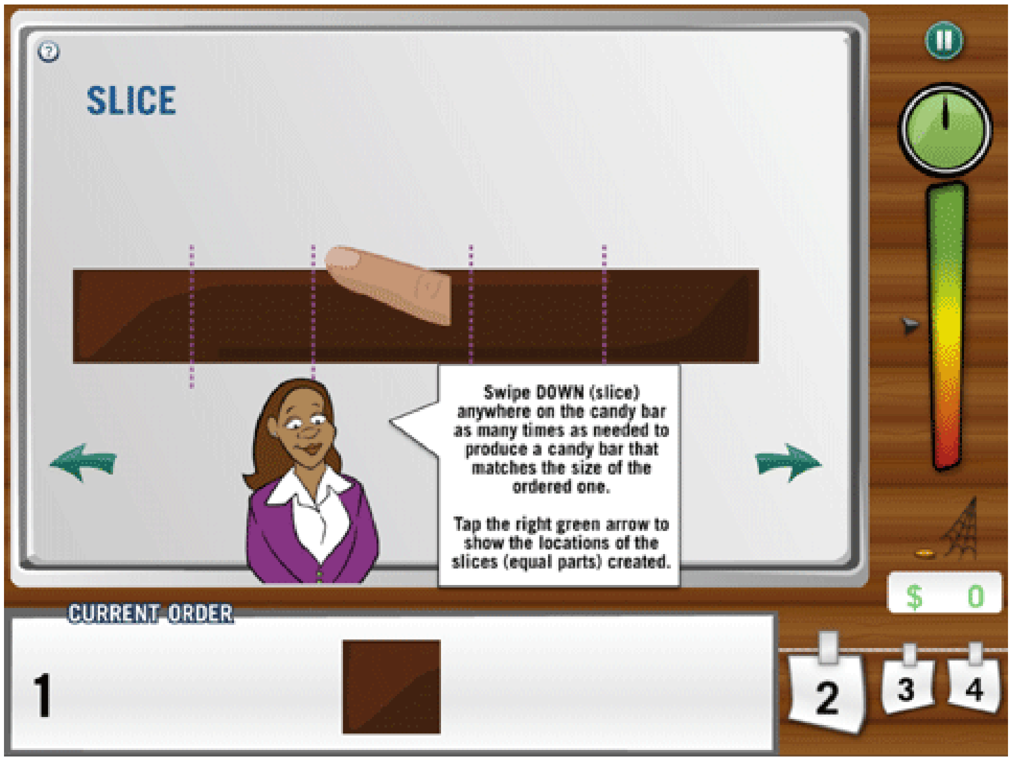


Figure 39. CandyFactory Educational Game for iPad game levels screen



This shows the candy bar ordered by a customer. You need to create and ship a candy bar with the same kind and size. Otherwise, the customer will be unhappy and Boss Cog will be angry!

Figure 40. CandyFactory Educational Game for iPad interactive tutorial for customer order screen



Swipe DOWN (slice) anywhere on the candy bar as many times as needed to produce a candy bar that matches the size of the ordered one.

Tap the right green arrow to show the locations of the slices (equal parts) created.

Figure 41. CandyFactory Educational Game for iPad interactive tutorial for slicing a candy screen

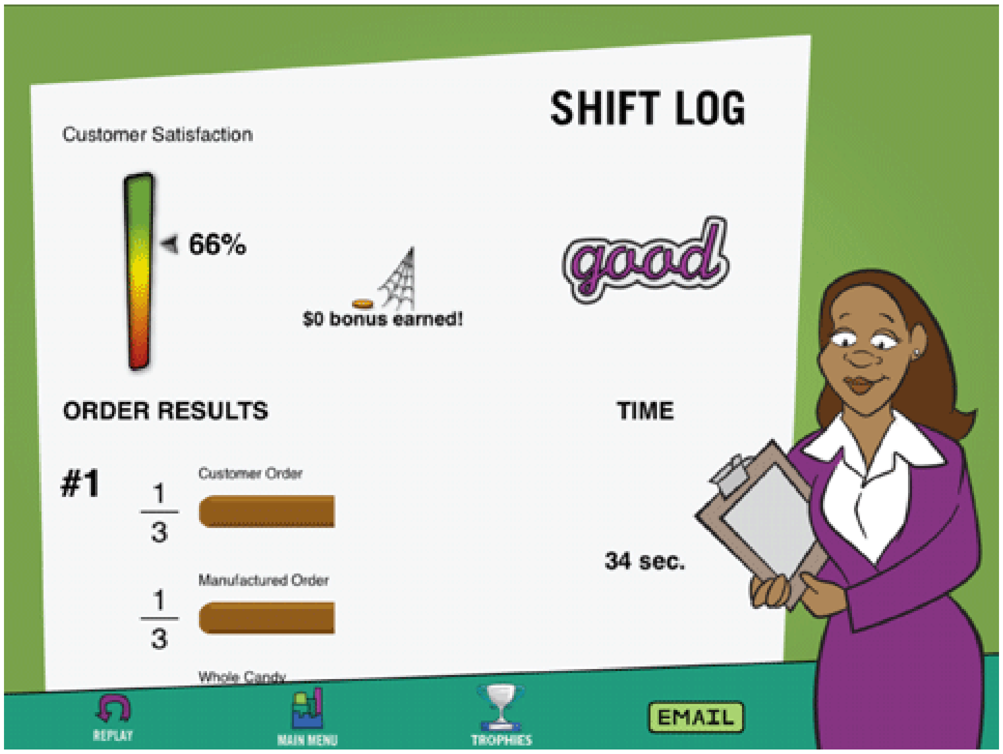
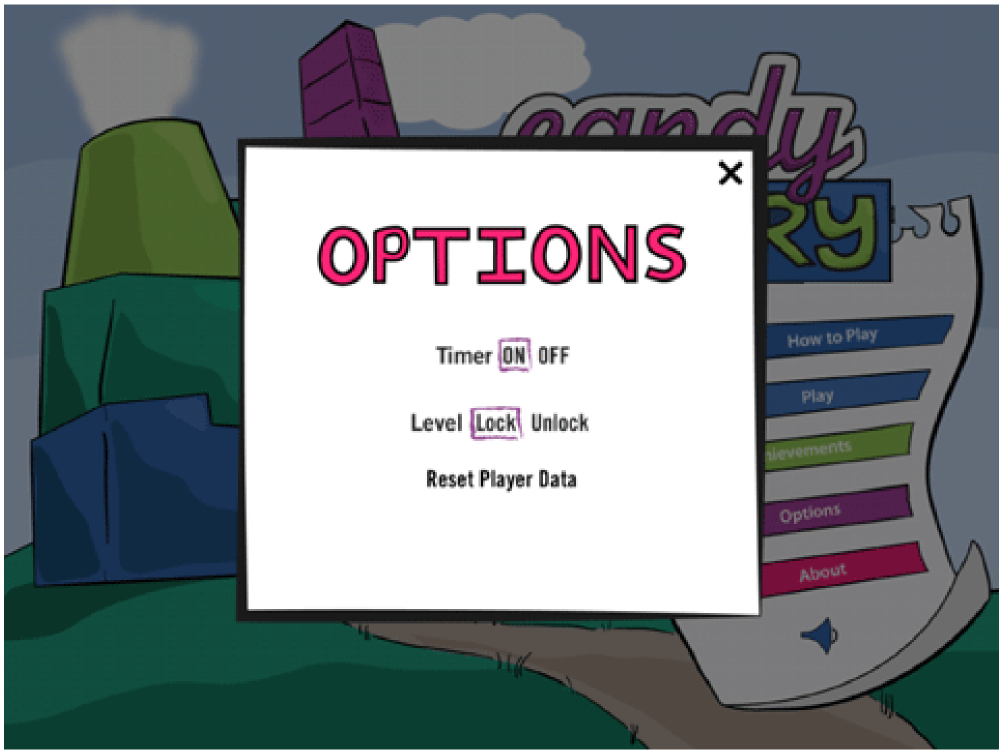


Figure 42. CandyFactory Educational Game for iPad shift log screen



OPTIONS

Timer [✓ ON] OFF

Level [✓ Lock] Unlock

Reset Player Data

Figure 43. CandyFactory Educational Game for iPad options screen

6.3 CandySpan

6.3.1 Description

"CandySpan is a simple and fun puzzle game that challenges your "working memory"" [LTRG 2014b]. This game plays a sequence of sounds, then the player is asked to repeat the given sequence of sounds, which gradually grows longer in length as the player successfully completes each task. CandySpan can be used by researchers and educators to measure users' current skill level. For example, players can play the "Try out your skills" as shown in Figure 44 to see how well their memories function. The CandySpan game is a universal application that supports iPhone, iPad, and iPod touches [Aslan et al. 2013a; LTRG 2014b].

The CandySpan application can be downloaded from Apple App Store [Aslan et al. 2013].

6.3.2 Functionality

The Main screen shows six different components: Play, Trophies, Information, Sound On/Off, and Settings buttons. Tapping on each button takes the player to the corresponding screen. For example, Play button takes the player to the Level selection screen, as shown in Figure 44. In this screen, each button starts a new game with a different level. For instance, tapping on the Easiest-Forward button makes the game begin in the easiest mode.

To successfully complete each task, players must repeat a sequence of touches on candy images as they see and hear:

- forward sequential order [Easiest, Forward]
- forward random order [Hard, Forward]
- reverse sequential order [Harder, Backward]
- reverse random order [Hardest, Backward].

The Level selection screen requires players to choose one of the six levels that are listed above. Upon selection of one of the levels, the game presents a How to Play tutorial. Figure 45 shows the tutorial screen. The tutorial walks players through each step and provides a detailed description and an animation of each task.

During the game play, if a player makes a mistake, the game warns the player and displays the Score screen. Figure 46 shows an example of a score list for a player. The score list includes the longest sequence completed, number of mistakes, level completion time, points earned, and the highest points earned for a player. The Score screen also allows players to replay, go to the home screen or the next level.

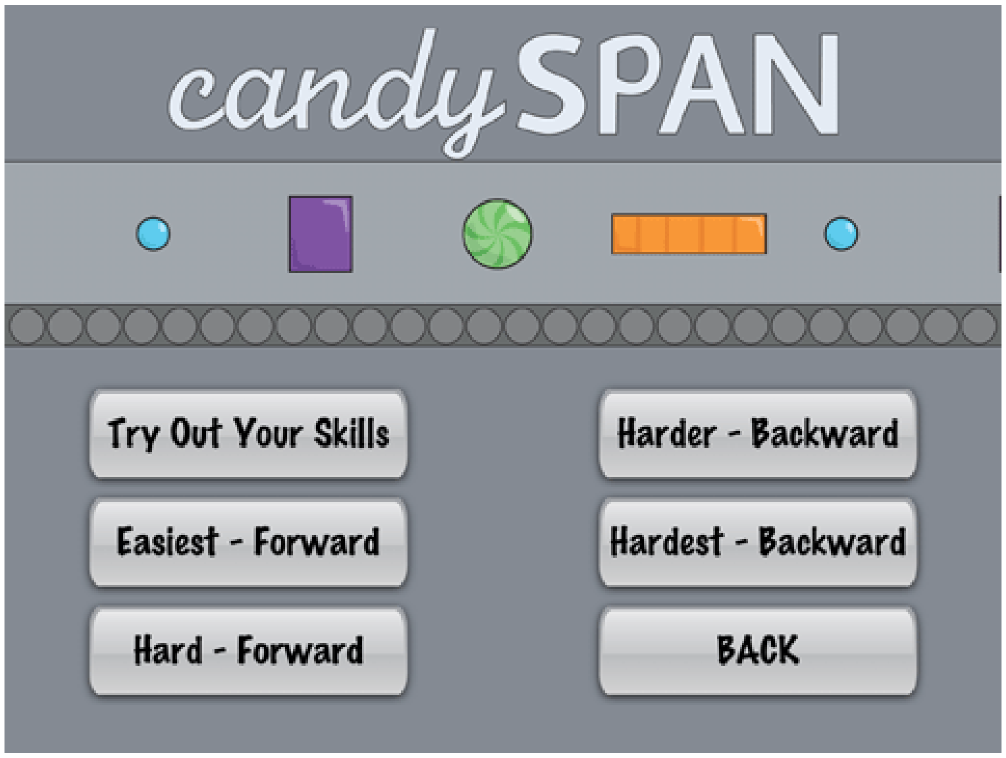
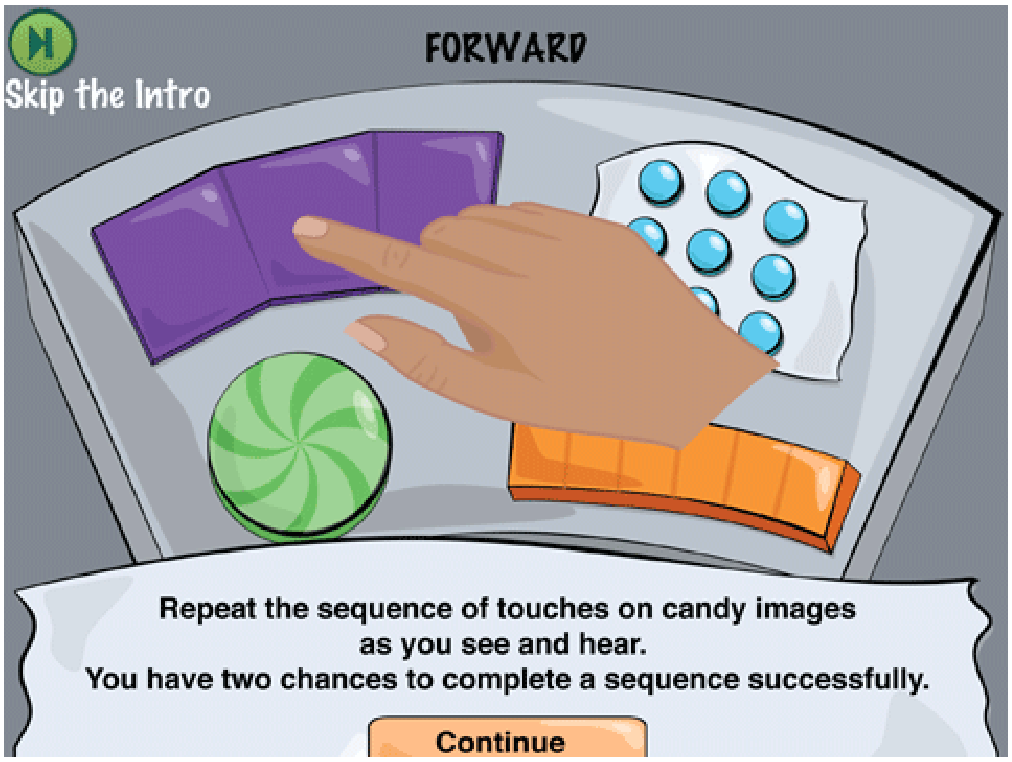
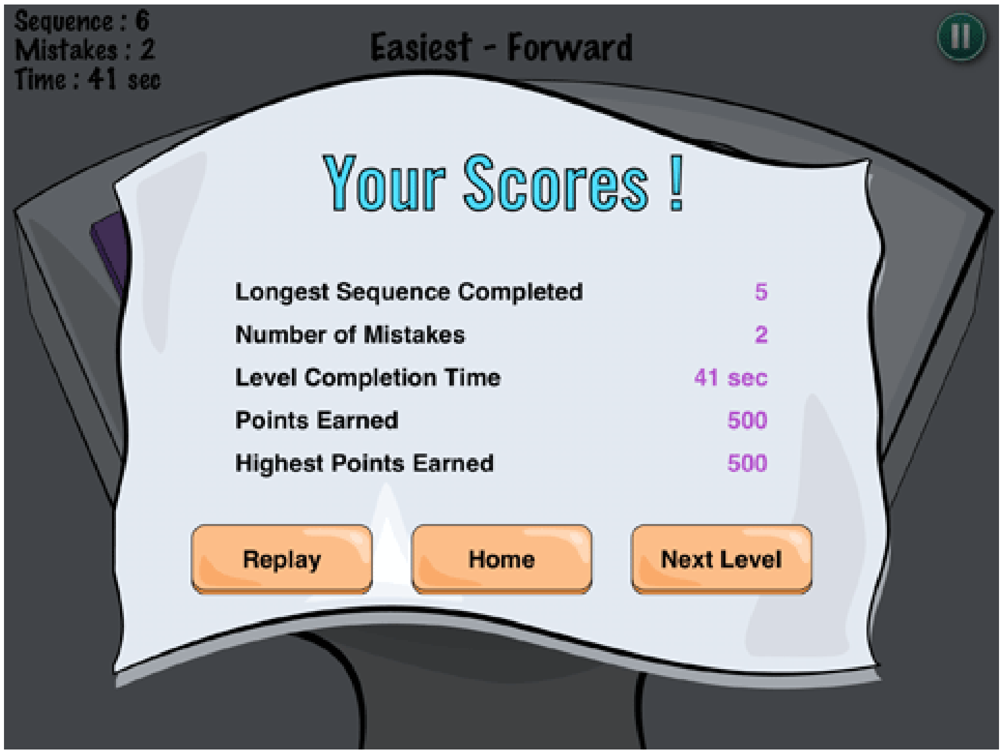


Figure 44. CandySpan level selection screen



Repeat the sequence of touches on candy images as you see and hear.
You have two chances to complete a sequence successfully.

Figure 45. CandySpan game tutorial screen



Longest Sequence Completed	5
Number of Mistakes	2
Level Completion Time	41 sec
Points Earned	500
Highest Points Earned	500

Figure 46. CandySpan player score screen

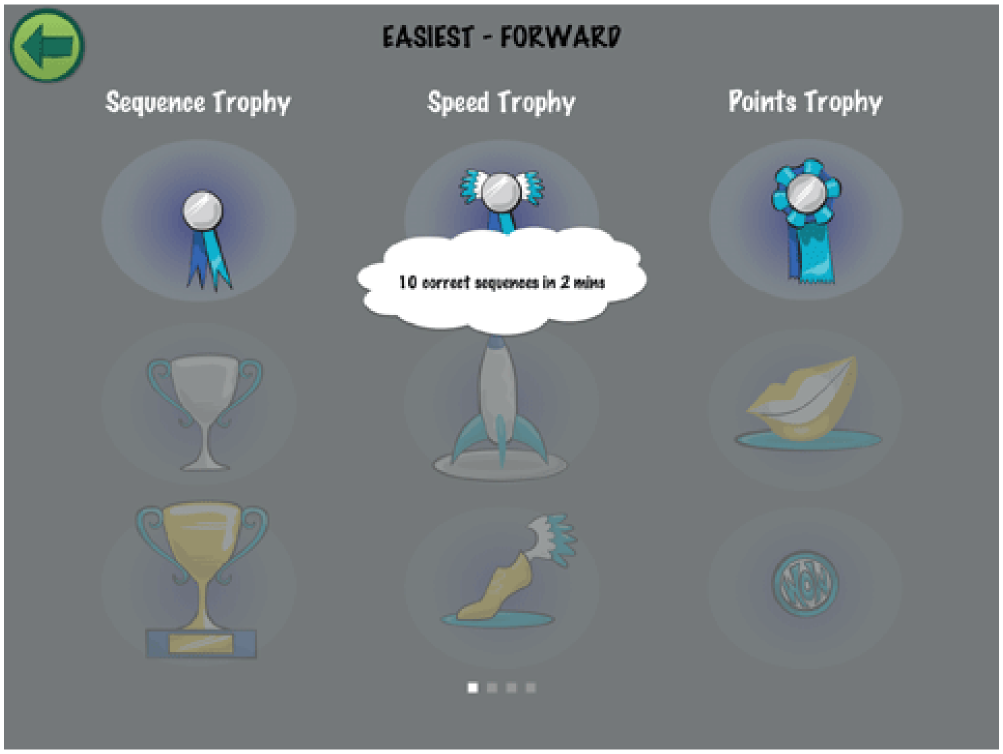


Figure 47. CandySpan trophy screen

6.4 CandyDepot

6.4.1 Description

CandyDepot is an educational game for the iPad. It is designed for middle school mathematics students and it focuses on units' coordination. "Students who coordinate two levels of units in an activity can solve simple multiplication problems, like 4x6, by counting by 6 four times. However, such students cannot necessarily understand situations involving more than one set of 6s. For example, they might not deduce that 4x6 and 3x6 is simply 7x6. In order to meaningfully understand such situations, students need to interiorize two levels of units as a structure in which they can think about four 6s and three 6s as objects that can be acted upon. Likewise, interiorizing three levels of units opens possibilities for students to meaningfully assimilate numerous situations across content domains: whole number multiplication, integer addition, fractions concepts, and algebraic reasoning" [Aslan et al. 2013b; LTRG 2014c].

The Candy Depot application is developed for all iPad devices and it can be downloaded from Apple App Store [Aslan et al. 2013b].

There are two levels:

- Level 1: "Players are allowed to choose how many bars go in each bundle and how many bundles go in each box. Players' performance will be rated by how efficiently they bundle each order" [Aslan et al. 2013b; LTRG 2014c].

- Level 2: "Players are only allowed to preview the orders that will be placed. Identifying common denominators and factors between the orders will help players choose optimal numbers of bars in each bundle and bundles in each box" [Aslan et al. 2013b; LTRG 2014c].

6.4.2 Functionality

The Home screen displays six different components: How to Play, Play, Achievements, Options, About and Sound On/Off as depicted in Figure 48. Each button takes players to the corresponding screen.

The game starts with President Carmelo's message. President Carmelo informs players that they are now responsible for distributing candy bars to Candy Inc. customers. To prove their worth, they must package and ship the correct number of candy bars in order to fill customer orders.

After the welcome message, the game takes the player to the main game play screen as shown in Figure 49. The player can see the incoming customer orders on the left side, shipment results on the right side, current packaging system at the bottom of the screen, and the shipment truck containing different numbers of bars, bundles and boxes in the middle as depicted in Figure 49.

In the main game play screen, players can drag and drop a number of different bars, bundles, and boxes to the truck and can remove them. When the player determines that the total number of bars matches with the customer order, the player scrolls down the truck door and the shipment takes place. The game displays an immediate feedback for the player's shipment and marks the shipment as either correct or incorrect in the shift performance table as depicted in Figure 49.

During the process of loading the truck, the player may need different types of bundles and boxes. Figure 50 and Figure 51 show the creation of different bundles and boxes. In the first process, players decide on the number of candy bars that would go into each bundle. In the next process, players decide on the number of bundles that would go into each box. The challenge is to efficiently ship each order using the fewest number of components—bars, bundles, and boxes—as possible. At each level, players need to ensure that they work quickly and efficiently to impress the Boss Cog!

After each shift, a detailed shift log is provided to players with feedback to assess their progress as depicted in Figure 52. The game also enables players to earn bonus cash and trophies for correct orders and fast turnaround. Player bonus cash can be used to customize your workspace, and change the types of packages you use. In addition, the game also allows players to share their score via email.

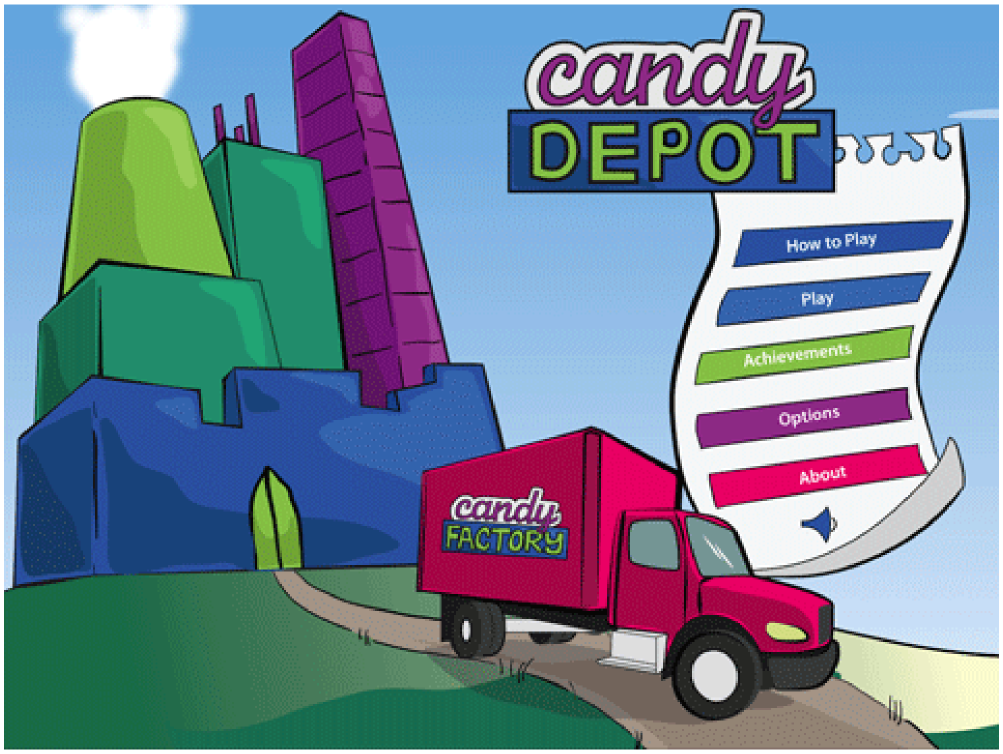


Figure 48. Candy Depot home screen

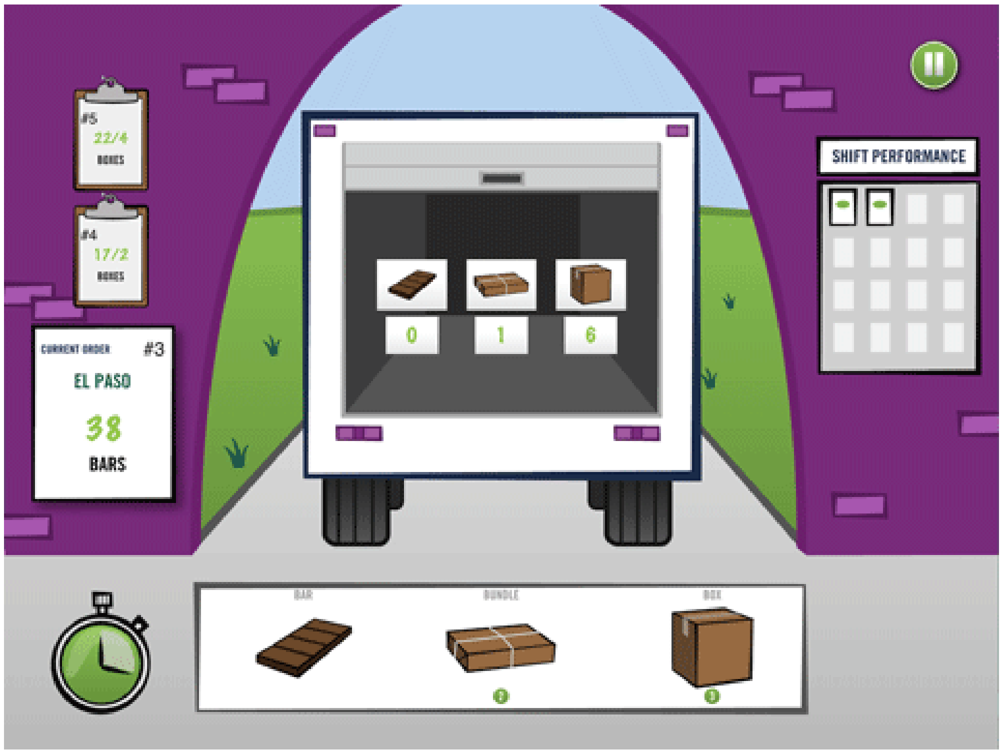


Figure 49. Candy Depot main game play screen

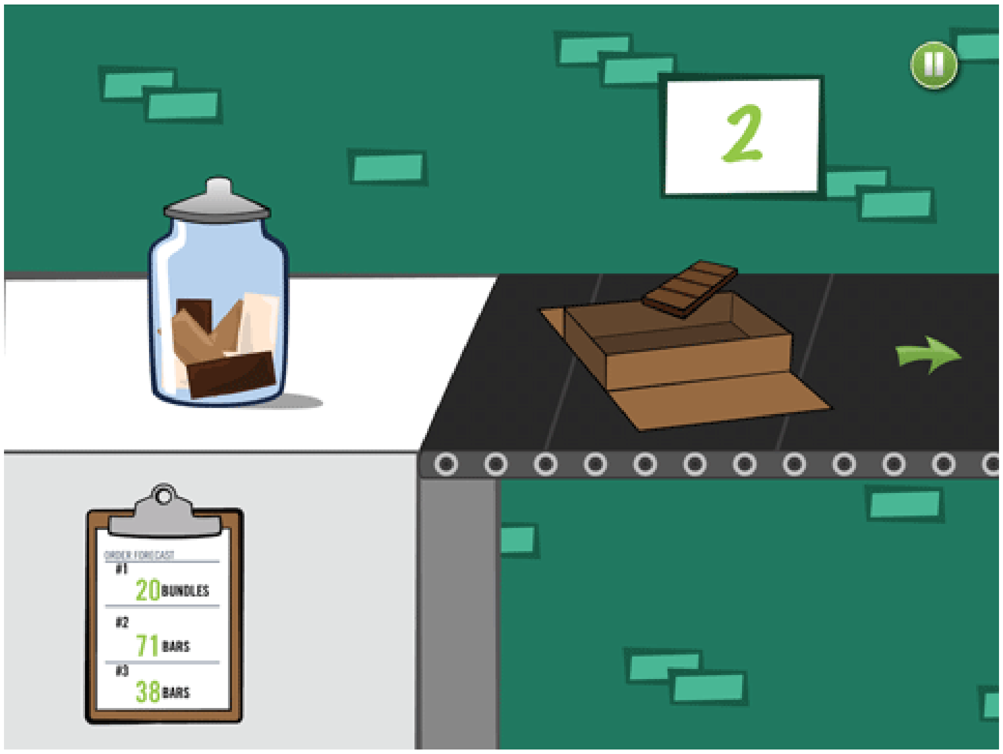


Figure 50. Candy Depot bundling screen

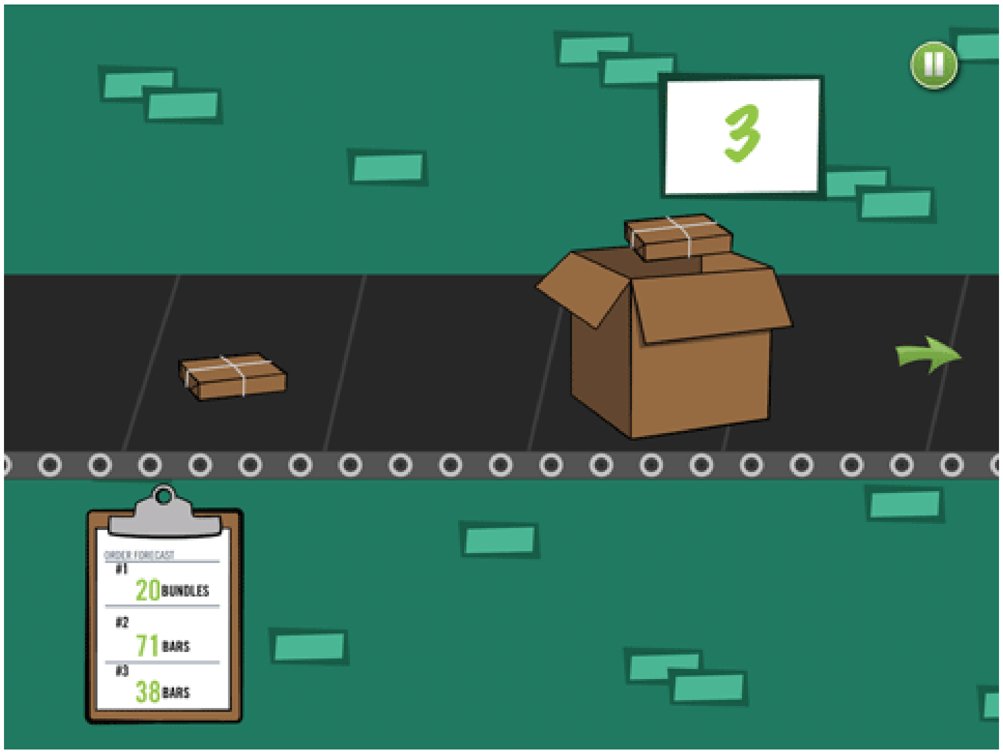
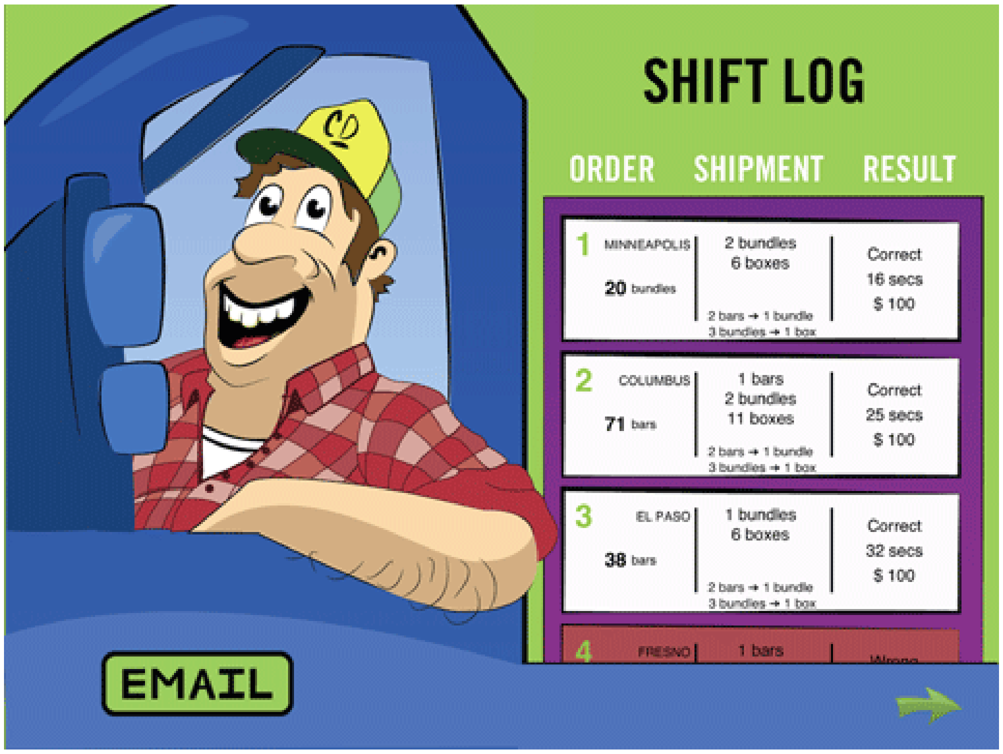


Figure 51. Candy Depot boxing screen



SHIFT LOG

ORDER	SHIPMENT	RESULT
1 MINNEAPOLIS 20 bundles	2 bundles 6 boxes 2 bars → 1 bundle 3 bundles → 1 box	Correct 16 secs $ 100
2 COLUMBUS 71 bars	1 bars 2 bundles 11 boxes 2 bars → 1 bundle 3 bundles → 1 box	Correct 25 secs $ 100
3 EL PASO 38 bars	1 bundles 6 boxes 2 bars → 1 bundle 3 bundles → 1 box	Correct 32 secs $ 100
4 FRESNO	1 bars	Wrong

Figure 52. Candy Depot shift log screen

6.5 CandyBot

6.5.1 Description

CandyBot is an educational game that focuses on teaching fractions and functions. The CandyBot game combines the functionality of the CandyFactory and the CandyDepot games. The game allows players to train a robot to efficiently satisfy customer orders that are types of improper fractions and mixed numbers. The game emphasizes visuals, animations, and actions to convey the game's story. Therefore, the game requires minimal reading and motivates players to practice and improve their algebraic reasoning. The game also invites players to automatize work done at the CandyFactory and the CandyDepot by programming a robot [LTRG 2014d; Aslan et al. 2015].

The CandyBot application is developed for all iPad devices and it can be downloaded from the Apple App Store [Aslan et al. 2015].

6.5.2 Functionality

Figure 53 shows six components: How to Play, Play, Achievements, Options, About, and Sound On/Off buttons. Each component takes the player to the corresponding game screen. For example, Play button takes the player to the main game play screen as shown in Figure 54.

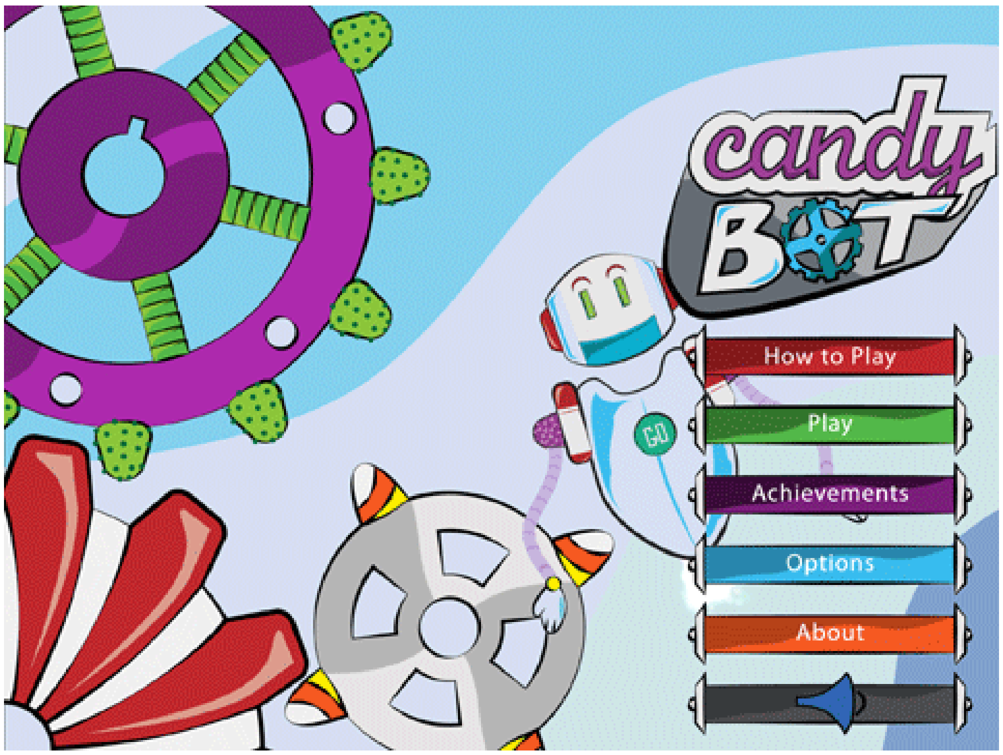


Figure 53. CandyBot home screen

The game starts with President Carmelo's welcome message informing players to be responsible for automating production by programming the company's new robot. In order to be successful, players have to determine the best way to program the robot to slice and to bundle various units.

For example, Figure 54 shows the creation of 6 3/7 bars of a customer's order. For this customer's order, the player has to program the robot to take 1 (whole) bar, 1 bundle containing 5 bars, and 3 of 1/7 (fraction) bars, which outputs the total number of candy bars. In this step, the player is only allowed to choose the identity function, which is a function that always returns the same value that was used as its argument. Therefore, the output will be equal to the input [LTRG 2014d].

At the end of each game, the game displays a log that lists a shipment report for the shift. Figure 55 shows an example of a shipment report and it includes input, function, output, order, and a result for each customer order. The shift log screen also allows players to share their assessment report (shift log report) via e-mail as shown in Figure 56.

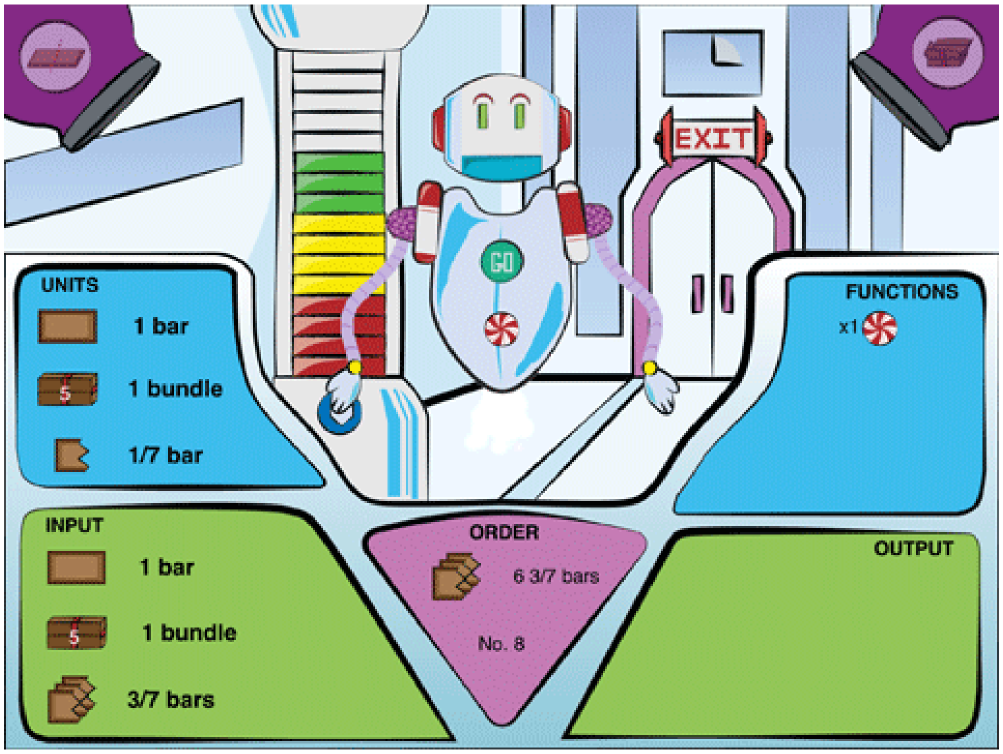
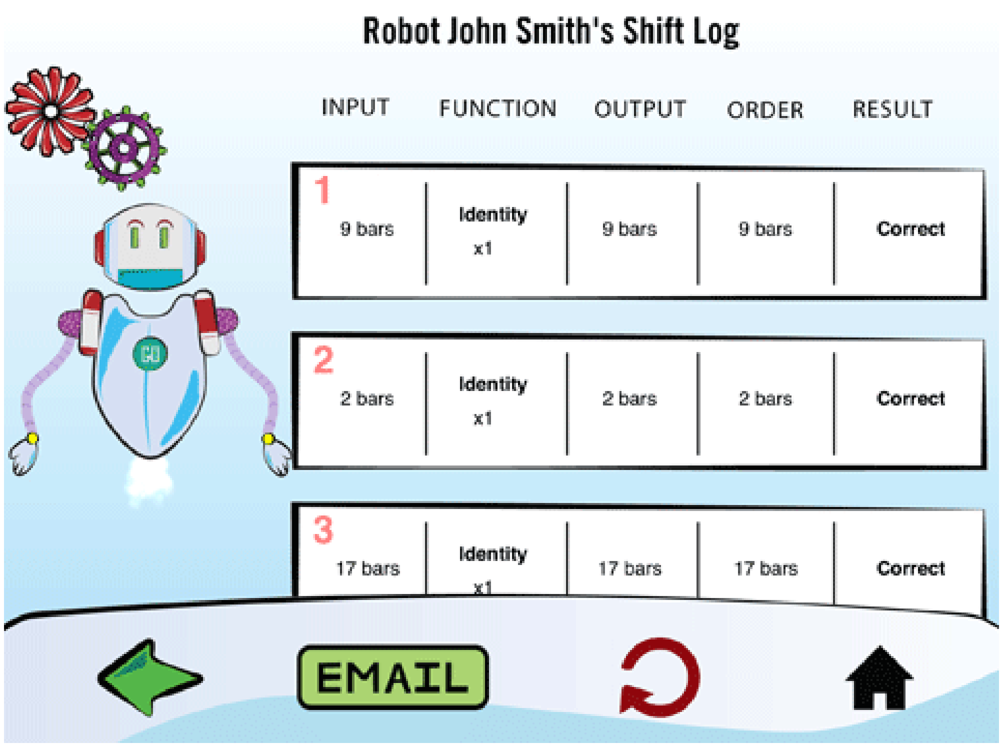
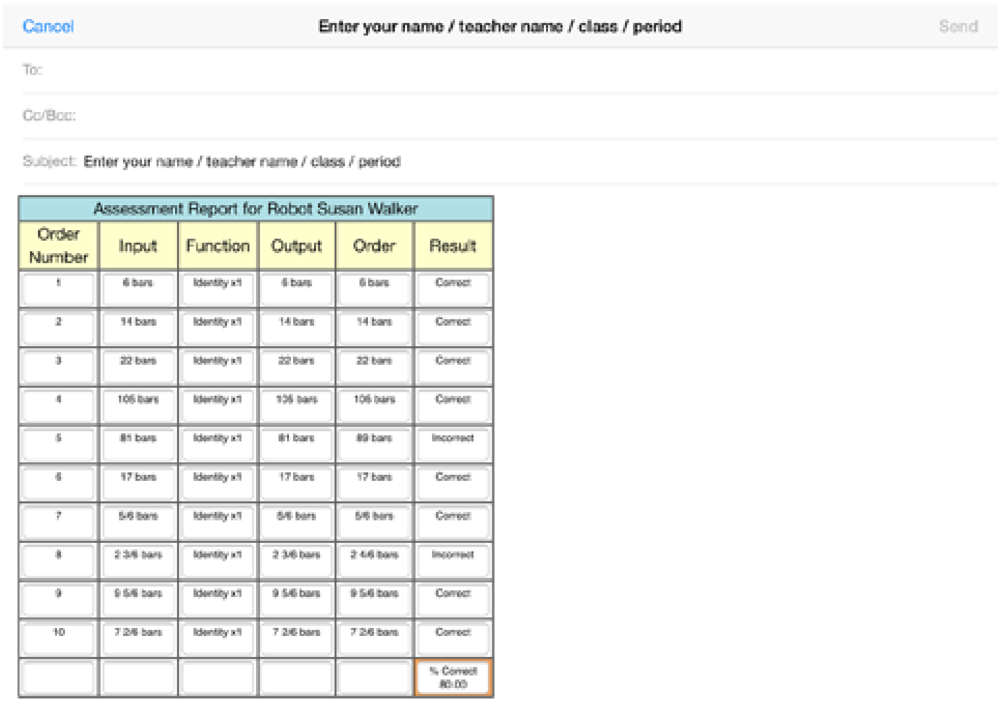


Figure 54. CandyBot main game play screen


Robot John Smith's Shift Log

	INPUT	FUNCTION	OUTPUT	ORDER	RESULT
1	9 bars	Identity x1	9 bars	9 bars	Correct
2	2 bars	Identity x1	2 bars	2 bars	Correct
3	17 bars	Identity x1	17 bars	17 bars	Correct

Figure 55. CandyBot shift log screen


Cancel Enter your name / teacher name / class / period Send

To:

Cc/Bcc:

Subject: Enter your name / teacher name / class / period

Assessment Report for Robot Susan Walker

Order Number	Input	Function	Output	Order	Result
1	6 bars	Identity x1	6 bars	6 bars	Correct
2	14 bars	Identity x1	14 bars	14 bars	Correct
3	22 bars	Identity x1	22 bars	22 bars	Correct
4	105 bars	Identity x1	135 bars	105 bars	Correct
5	81 bars	Identity x1	81 bars	89 bars	Incorrect
6	17 bars	Identity x1	17 bars	17 bars	Correct
7	5/6 bars	Identity x1	5/6 bars	5/6 bars	Correct
8	2 3/6 bars	Identity x1	2 3/6 bars	2 4/6 bars	Incorrect
9	9 5/6 bars	Identity x1	9 5/6 bars	9 5/6 bars	Correct
10	7 2/6 bars	Identity x1	7 2/6 bars	7 2/6 bars	Correct
					% Correct 80.00

Figure 56. CandyBot assessment report by e-mail screen

CHAPTER 7: CONCLUDING REMARKS AND FUTURE WORK

7.1 Concluding Remarks

Development of a DEG in the form of software for DEG-based learning poses significant technical challenges for educators, researchers, game designers, and software engineers. The nature of the problem of developing DEG is multi-faceted; thus, requiring unifying methodologies for development and software quality evaluation, and should not be performed in an *ad hoc* manner. This dissertation presents such methodologies named: GAMED and IDEALLY.

We developed GAMED and IDEALLY based on experience and knowledge we have gained in creating and publishing four digital educational games that run on the iOS mobile devices: CandyFactory, CandySpan, CandyDepot, and CandyBot. These two methodologies provide a quality-centered structured approach for development of digital educational games and are essential for accomplishing demanding goals of game-based learning.

Although there have been many studies published on educational game development, a structured and effective methodology has been absent. Hence, GAMED provides such a methodology to overcome the development complexity and to guide the developers throughout the entire life cycle.

Evaluation of a DEG is an important activity focusing on the accurate assessment of a given concept. It is a qualitative concept that cannot be directly measured or assessed. Therefore, the IDEALLY methodology provides a hierarchy of 111 indicators consisting of 21 branches and 90 leaf indicators. These come in the form of an acyclic graph for the measurement and evaluation of digital educational game software quality. As a case study, we provide step-by-step instructions for employing IDEALLY to assess the quality of the CandyBot game.

Classifications provided in the literature are inadequate for the game designers, engineers and practitioners. Moreover, the current terminology in the game domain involves many inappropriate terms. To that end, we present a taxonomy of games that focuses on the characterization of games and provides an appropriate classification based on identified characteristics.

Developing entertaining and engaging educational games are essential for students to truly benefit from it. Games that are developed as part of this study provide a game environment, which is not only captivating for the students, but also helps them to learn different mathematical concepts in an enjoyable manner with sustained interest. These games are designed by taking the user needs into consideration. Detailed research was compiled before developing these games in the following areas: inventory of the currently available games in the Apple App Store and on the Internet, in depth discussions with teachers and students.

We developed educational games that have clear objectives and rules. These games provide an interactive and realistic game environment, which help students gain hands-on learning experience.

7.2 Summary of Contributions

This research had three related objectives to achieve. The first objective was to create a methodology for the development of digital educational games. The second objective was to develop a methodology for software quality evaluation of digital educational games. The third objective was to create a taxonomy of games. These objectives were met in this dissertation, and resulted in the following contributions to the state of the art.

7.2.1 Digital Educational Game Development Methodology (GAMED)

We created GAMED methodology for the development of digital educational games. GAMED is embedded within a lifecycle that we developed. The lifecycle represents a framework for organization of the processes, work products, quality assurance activities, and project management activities required to develop, use, maintain, and evolve a game software application from birth to retirement. GAMED methodology brings modularization and structure to game development by providing guidance to the game designers, software engineers, and project managers, such as what process to follow and what products to deliver at the end of each process [Aslan and Balci 2015]. GAMED methodology has been adopted during the implementation of CandyDepot and CandyBot games.

7.2.2 Digital Educational Game Software Quality Evaluation Methodology (IDEALLY)

Evaluation of a DEG is essential for DEG software quality. In this work, we developed the IDEALLY methodology for software quality evaluation of digital educational games. IDEALLY enables measurement and evaluation of qualitative and quantitative characteristics of a game. It provides relative criticality weighting, enables the use of Subject Matter Experts (SMEs) in evaluating the leaf indicators, utilizes SME knowledge, presents graphical representation of the evaluation results and interprets the evaluation results by using a hypertext assessment report. Using IDEALLY, we provided step-by-step instructions to assess the quality of the CandyBot game in a case study.

7.2.3 Taxonomy of Games

Based on the identified characteristics of games, we created a taxonomy to provide guidance for digital educational game designers and developers.

7.2.4 Digital Educational Games Published

7.2.4.1 CandyFactory Educational Game

Table 8. Summary of CandyFactory Educational Game

Software Name:	CandyFactory Educational Game
Description:	A digital game that teaches the concept of fractions to middle school students based on splitting operations with partitioning and iterating.
Platform:	For all iOS devices
Release Date:	July 14, 2011 Download
Developed By:	Serdar Aslan, Anderson Norton, and Osman Balci
Number of Downloads:	231,000 as of September 2016
Award:	National Educators Association Award for Innovative App, $1000 awarded to LTRG, for the design of *CandyFactory*, 2012

7.2.4.2 CandyFactory Educational Game for iPad

Table 9. Summary of CandyFactory Educational Game for iPad

Software Name:	CandyFactory Educational Game for iPad
Description:	A digital game that teaches the concept of fractions. The original "CandyFactory Educational Game" is redesigned and implemented.
Platform:	iPad, iPad Mini, and iPad Pro
Release Date:	April 30, 2012, V2.0 Download
Developed By:	Serdar Aslan, Anderson Norton, and Osman Balci
Number of Downloads:	94,400 as of September 2016
Award:	Awarded as "Top Rated App" by Balefire Labs, in 2014

7.2.4.3 CandySpan

Table 10. Summary of CandySpan

Software Name:	CandySpan
Description:	A simple, fun puzzle game that challenges your "working memory", which can be used by researchers and educators to measure your current skill level.
Platform:	iPhone, iPad, and iPod Touch
Release Date:	January 20, 2013 Download
Developed By:	Serdar Aslan, Kirby Deater-Deckard, and Osman Balci
Number of Downloads:	23,400 as of September 2016

7.2.4.4 CandyDepot

Table 11. Summary of CandyDepot

Software Name:	CandyDepot
Description:	An educational game that focuses on a critical mental activity referred to as units coordination and is intended for middle school mathematics students.
Platform:	iPad, iPad Mini, and iPad Pro
Release Date:	August 27, 2013 Download
Developed By:	Serdar Aslan, Anderson Norton, and Osman Balci
Number of Downloads:	62,500 as of September 2016

7.2.4.5 CandyBot

Table 12. Summary of CandyBot

Software Name:	CandyBot
Description:	An educational game that teaches Algebraic reasoning and symbolizing with linear functions for middle school mathematics students.
Platform:	iPad, iPad Mini, and iPad Pro
Release Date:	April 2, 2015 Download
Developed By:	Serdar Aslan, Anderson Norton, Osman Balci, and Steven Boyce
Number of Downloads:	55,600 as of September 2016
Award:	Awarded the best educational App of 2015 by Balefire Labs.

7.3 Future Research

The GAMED methodology described in this dissertation is used to develop CandyFactory Educational Game for iPad, CandySpan, CandyDepo and CandyBot games. GAMED methodology focuses on the development of digital educational games. However, it can serve beyond educational games. Further research can be conducted to explore how GAMED can answer the following questions:

- How well the methodology can be adopted by other DEGs?
- How can the methodology be used to develop non-educational games, such as action, and adventure games?

The evaluation of a DEG is a qualitative concept that cannot be directly measured or assessed. Therefore, a set of indicators are created to measure it indirectly. The list of indicators that are described as part of the IDEALLY methodology can be modified since we have an example list of indicators. We believe that creating a complete list of indicators may enable developers to assess the quality of a DEG more effectively. Thus, further research can be conducted to explore different quality indicators for DEGs.

Based on the given time and scope we were only able to apply the IDEALLY methodology to the CandyBot game. Thus further research can be conducted to apply IDEALLY methodology to other digital educational games.

The games that are published as part of this dissertation can adopt the personalized learning ideology by using a robust game-based, mobile device-centric platform. This allows for continual embedded assessment, which keeps track of each individual player's learning curve and modifies the root of the game accordingly.

Moreover, the recent developments in cloud computing and cloud platforms enable developers to simply deploy backend game services, gather game analytics, and send a variety of different notifications. Hence, feature research can be conducted to explore how one can import these games to the cloud platforms and take all advantages as listed above.

BIBLIOGRAPHY

Aarseth, E. (2001), "Computer Game Studies, Year One," Game Studies: *The International Journal of Computer Game Research*, 1(1), 2001.

Adams, E. and R. Andrew (2010), "Fundamentals of Game Design," 2nd ed. New Riders Publishing Thousand Oaks, CA, USA.

Adams, E. and R. Andrew (2003), "Andrew Rollings and Ernest Adams on game design," New Riders Publishing.

Adams E. (2013), "Fundamentals of game design," Pearson Education. p. 57.

Ahmed, D. T., S. Shirmohammadi, and J. C. Oliveira (2009), "A hybrid P2P communications architecture for zonal MMOGs," Multimedia Tools and Applications 45, 313–34 Berson, A. (1996), "Client/server architecture," McGraw-Hill, Inc.

Alberts, C. J. and A. J. Dorofee, (2010), "Risk Management Framework," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Almeida, M. S. and F. S. da Silva (2013), "Requirements for game design tools," Proceedings of SBGames 2013.

Alvarez J., D. Djaouti, and R. Ghassempouri (2006), "Morphological study of videogames," *CGIE'06 conference*, Australia.

Ambinder, M. (2009), "Valve's approach to playtesting: The application of empiricism," In Game Developer's Conference.

Amory, A. (2001), "Building an educational adventure game: Theory, design and lessons," *Journal of Interactive Learning Research*, 129(2/3), 249–264.

Amory, A. (2006), "Game object model version II: a theoretical framework for educational game development," *Educational Technology Research and Development*, *55*(1), 51-77.

Apple (2014), "Using Design Patterns," Apple Incorporated, Cupertino, CA.

Arvola, M. and H. Artman (2007), "Enactments in interaction design: How designers make sketches behave," Artifact, 1(2), 106-119.

Ash, K. (2011), "Digital gaming goes academic," Education week, 30 (25), 24-28.

Aslan, S. (2011), "Game-based Improvement of Learning Fractions Using iOS Mobile Devices," Master's Thesis etd-04282011-130352, Digital Library and Archive, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Aslan, S. and O. Balci (2015), "GAMED: Digital Educational Game Development Methodology," *Simulation: Transactions of the Society for Modeling and Simulation International* 91, 4 (Apr.), 307-319, doi: 10.1177/0037549715572673.

Aslan, S., A. Norton, and O. Balci (2011), "CandyFactory Educational Game," Apple App Store

Aslan, S., A. Norton, and O. Balci (2012), "CandyFactory Educational Game for iPad," Apple App Store

Aslan, S., K. Deater-Deckard, and O. Balci (2013), "CandySpan," Apple App Store

Aslan, S., A. Norton, and O. Balci (2013), "CandyDepot," Apple App Store

Aslan, S., A. Norton, S. Boyce and O. Balci (2015), "CandyBot," Apple App Store

Balci, O. (1997), "Principles of simulation model validation, verification, and testing," *Transactions of the Society for Computer Simulation International* 14 (1): 3-12.

Balci, O. (2001), “A Methodology for Certification of Modeling and Simulation Applications,” *ACM Transactions on Modeling and Computer Simulation (TOMACS) 11*, 4 (Oct.), 352-377.

Balci, O. (2012), “A Life Cycle for Modeling and Simulation,” *Simulation: Transactions of the Society for Modeling and Simulation International 88*, 7, 870–883.

Balci, O. and W. F. Ormsby (2008), “Network-Centric Military System Architecture Assessment Methodology,” *International Journal of System of Systems Engineering 1*, 1-2, 271-292.

Balci, O., R. E. Nance, J. D. Arthur, and W. F. Ormsby (2002), “Expanding our horizons in verification, validation, and accreditation research and practice,” In *Pro- ceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes. Piscataway, NJ: IEEE.

Balci, O., R. J. Adams, D. S. Myers, and R. E. Nance (2002), “A Collaborative Evaluation Environment for Credibility Assessment of Modeling and Simulation Applications,” In *Proceedings of the 2002 Winter Simulation Conference* (San Diego, CA, Dec. 8-11). IEEE, Piscataway, NJ, pp. 214-220.

Baldwin M. (2005), “Game Design Document Outline,” Baldwin Consultant.

Barnhardt, P. (2011), “Game design tips 2: That Pesky Character development,” AXS Digital Group LLC.

Bartle, R. A. (2004), “Designing virtual worlds,” New Riders.

Bates, B. (2004), “Game Design,” Premier Press; 2nd Revised edition, p. 450.

Baughman, N., M. Liberatore and B. Levine (2007), “Cheat-proof playout for centralized and peer-to-peer gaming,” IEEE Trans. on Networking (TON) 15, 1–13.

Bell, R. and T. Hopper (1999), “A ‘T.A.C.T.I.C’ approach to teaching territory Games,” Wolfville, Nova Scotia: *CAHPER '99 national conference*.

Bell, R. C. (1980), “Discovering old board games,” (2nd ed.) Princes Risborough, Bucks.: Shire Publications.

Bernhaupt, R., W. Ijsselsteijn, F. F. Mueller, M. Tscheligi, and D. Wixon (2008), “Evaluating user experiences in games,” In *Proceedings of ACM CHI 2008 Conference on Human Factors in Computing Systems*, pp. 3905-3908.

Bethke, E. (2003), “Game development and production,” pp 15-32, Wordware Publishing, Inc.

Bettner, P. and M. Terrano, (2001), “Network Programming in Age of Empires and Beyond” UBM Tech, London, England.

Black, P. J. and D. Wiliam (1998), “Inside the black box: Raising standards through classroom assessment,” London, UK: King’s College London School of Education.

Blizzard (2014), “World of Warcraft,” Blizzard Entertainment Incorporated, Irvine, CA,

Blow, J. (2004), “Game development: Harder than you think,” *Queue, 1*(10), 28.

Boehm, B. (1986), “A spiral model of software development and enhancement,” ACM SIGSOFT Software Engineering Notes, 11(4), 14-24.

Boehm, B. W. (1988), “A spiral model of software development and enhancement,” Computer, 21(5), 61-72.

Carpenter, A. (2003), “Applying Risk Analysis To Play-Balance RPGs,” UBM Tech, London, England.

Boehm, B. and V. Basili (2001), “Software defect reduction top 10 list,” *IEEE Computer*, 34(1):135–137, Jan. 2001.

Booch, G., I. Jacobson and J. Rumbaugh (1999), Unified Modeling Language Users Guide, Addison Wesley Longman, Inc.

Brathwaite, B. and I. Schreiber (2009), "Challenges for game designers," Course Technology/Cengage Learning.

Bridgeland, J. M., J. J. Bilulio, and K. B. Morison (2006), "The silent epidemic: Perspectives of high school dropouts," *Bill and Melinda Gates Foundation*, Seattle, WA.

Buschmann, F., K. Henney, and D. Schimdt (2007). *Pattern-oriented Software Architecture: On Patterns and Pattern Language* (Vol. 5). John Wiley & Sons.

Byrne, E. (2005), "Game Level Design," Cengage Learning.

Caillois R. (1958), "Les jeux et les hommes," Gallimart, Paris.

Callele, D., E. Neufeld, and K. Schneider (2006), "Emotional requirements in video games," *14th IEEE International Requirements Engineering Conference (RE'06).*

Callele, D., K. Neufeld and K. Scnedier (2005), "Requirements Engineering and the Creative Process in the Video Game Industry," *Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05)*.

Chan, H., A. Fern, S. Ray, N. Wilson, C. Ventura (2007), "Online Planning for Resource Production in Real-Time Strategy Games," In ICAPS (pp. 65-72).

Chang, M., M. Evans, S. Kim, A. Norton, K. Deater-Deckard, O. Balci, and Y. Samur (2012), "The Influence of an Educational Video Game on Mathematical Engagement," Submitted for publication.

Charette, R. N. (2005), "Why software fails," *IEEE Spectrum*, September 2005.

Chen, J. (2007), "Flow in games (and everything else)," Communications of the ACM, 50(4), 31-34.

Chigani, A. and O. Balci (2012), "The Process of Architecting for Software / System Engineering," *International Journal of System of Systems Engineering 3*, 1, 1-23.

IBM (2012), "Big data analytics for video, mobile, and social game monetization," International Business Machines Corporation (IBM), Armonk, NY.

Clanton, C. (1998), "An interpreted demonstration of computer game design," in *CHI 98 summary: Human Factors in Computing Systems. Chi 98*, pp. 1-2.

Clark, D., B. C. Nelson, P. Sengupta, and C. D'Angelo (2009), "Rethinking science learning through digital games and simulations: genres, examples, and evidence," *In National academies of sciences learning science: Computer games, simulations, and education conference*, Washington, DC.

CMMI Product Team (2010), "CMMI for Development, Version 1.3, " Carnegie Mellon Univ., Software Engineering Inst., Pittsburg, PA, Tech Rpt.

Co, P. (2006), "Level design for games: creating compelling game experiences," New Riders, Berkeley.

Cocos2d-swift (2014), "Cocos2d-swift," Cocos2d-swift community.

Coffey, H., (2009), "Digital game-based learning," University of North Carolina at Chapel Hill School of Education, Chapel Hill, NC.

Collins, J. (1997), "Conducting In-house Play Testing," UBM Tech, London, England.

Conole, G. and M. Oliver (1998), "A pedagogical framework for embedding C&IT into the curriculum," Research in Learning Technology 6 (2).

Cook, D. (2006), "Managing game design risk: Part I," Lost Garden.

Cover, J. G. (2010), “The Creation of Narrative in Tabletop Role-Playing Games,” McFarland & Company. pp. 6.

Dalton, J. P. (2000), “Online training needs a new course,” *The Forrester report*

Davis, J. P., K. Steury, and R. Pagulayan (2005). A survey method for assessing perceptions of a game: The consumer playtest in game design. Game Studies, 5(1).

De Freitas, S. and M. Oliver (2006), “How can exploratory learning with games and simulations within the curriculum be most effectively evaluated?,” *Computers and Education*, 46, 249–264.

Desurvire, H., C. and Wiberg (2008), “Master of the game: assessing approachability in future game design," in *Proceedings of ACM CHI 2008 Conference on Human Factors in Computing Systems*, pp. 3177-3182.

Deubel, P. (2006). “Game on!” T.H.E. Journal (Technological Horizons in Education), 33 (6), pp. 30-35.

DFC Intelligence (2012), “Worldwide Market Forecasts for the Video Game and Interactive Entertainment Industry,” DFC Intelligence, San Diego, CA.

Dickey, M. D. (2005), “Engaging by design: How engagement strategies in popular computer and video games and inform instructional design,” *Educational Technology Research and Development*, 53, 67–83.

DoDAF (2009a), “DoD Architecture Framework Version 2.0 Volume I: Introduction, Overview, and Concepts - Manager’s Guide,” Architecture Framework Working Group, Washington, DC.

DoDAF (2009b), “DoD Architecture Framework Version 2.0 Volume II: Architectural Data and Models - Architect’s Guide,” Architecture Framework Working Group, Washington, DC.

DoDAF (2009c), “DoD Architecture Framework Version 2.0 Volume III: DoDAF Meta-model Physical Exchange Specification - Developer’s Guide,” Architecture Framework Working Group, Washington, DC.

Dondlinger, M. J. (2007), “Educational video game design: A review of the literature,” Journal of Applied Educational Technology, 4(1), 21-31.

Dondlinger, M. J. (2007), “Educational video game design: A review of the literature,” *Journal of Applied Educational Technology*, *4*(1), 21-31.

Ebner, M. and A. Holzinger (2007), ‘Successful implementation of user- centered game based learning in higher education: An example from civil engineering," *Computers & Education*, vol. 49, pp. 873-890.

Ehrmann, S. (1999), “Studying teaching, learning and technology: a tool kit from the Flashlight programme,” Active Learning, 9, 36–39.

Elias, G. S., R. Garfield, and K. R. Gutschera (2012), “Characteristics of Games,” The MIT Press.

Ellington H., E. Addinal, and F. Percival (1982), “A handbook of game design,” Kogan Page, London.

Ellis, A. B. (2007), “The influence of reasoning with emergent quantities on students’ generalizations,” *Cognition and Instruction*, *25*, 439 – 478.

Ellis, M. (1983), “Similarities and differences in games: A system for classification,” *AIESEP conference*, Rome, Italy.

ESA (2007), “Essential Facts about the Computer and Video Game Industry,” Entertainment Software Association (ESA), Washington, DC.

Epic Games (2014), “Welcome to the Unreal Engine 4,” Epic Games Incorporated, Rockville, MD.

Ermi, L. and F. Mäyrä (2007), "Fundamental components of the gameplay experience: Analysing immersion," Worlds in Play: International Perspectives on Digital Games Research. New York: Peter Lang Publishers, 37-53.

Esbensen, D. (2005), "Online Game Architecture: Back-end Strategies," *Gamasutra, March, 10*, 2005.

Esposito N. (2005), "A Short and Simple Definiton of What a VideoGame Is", *Proceedings of DiGRA 2005 Conference*: Changing Views.

EVE (2014), "EVE Online," CCP Games, Reykjavik, Iceland

Fällman, D. (2003), "Design-oriented human- computer interaction," In Proc. of the SIGCHI Conference on Human Factors in Computing Systems CHI '03. ACM, p.225-232.

Federoff, M. A. (2002), "Heuristic And Usability Guidelines For The Creation And Evaluation Of Fun In Video Games," Indiana University.

Felicia, P., Ed. (2011), "Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches," IGI Global, Hershey, PA.

Feng, W. C., F. Chang, W. C. Feng, and J. Walpole (2002), "Provisioning on-line games: a traffic analysis of a busy counter-strike server," In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement* (pp. 151-156). ACM.

Fladen, E. and K. Blashki (2005), "Learning = playing: Interactive learning and game-based design principles," *22nd acsilite annual conference*, Brisbane, Australia.

Flanagan, M. and H. Nissenbaum, (2007), "A game design methodology to incorporate social activist themes," *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 181-190.

Flood, K. (2003), "Game Unified Process," GameDev.Net LLC, Denton, TX

Flynt, J. P. and O. Salen (2004), "Software Engineering for Game Developers," Software Engineering Series. Course Technology PTR, 1 ed. edition, November 2004.

Fullerton, T. (2008), "Game design workshop: a playcentric approach to creating innovative games," Taylor & Francis US.

Gabler, K., K. Gray, M. Kucic, and S. Shodhan (2005), "How to prototype a game in under 7 days: Tips and tricks from 4 grad students who made over 50 games in 1 semester," UBM Tech, London, England.

Gaikai, Inc. (2014), "Gaikai," Sony Computer Entertainment, Tokyo, Japan.

Gal, V. (2002), "Writing for Video Games," Proceedings Laval Ritual (IVRC).

Gamma, E., R. Helm, R. Johnson and J. Vlissides (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA.

Gee, J. P. (2003), "What video games have to teach us about learning and literacy," *ACM Computers in Entertainment*, 1(1), 1–4.

Gee, J. P. (2007), "Good video games and good learning," New York: Peter Lang.

Gee, J. P. (2009), "Deep learning properties of good digital games: How far can they go?," *In U. Ritterfeld, M. Cody, & P. Vorderer (Eds.) Serious games: Mechanisms and effects*, (pp. 67–82). New York, NY: Routledge.

Gerling, K. and M. Masuch (2011), "When gaming is not suitable for everyone: Playtesting wii games with frail elderly," In 1st Workshop on Game Accessibility: Xtreme Interaction Design (GAXID'11).

Gershenfeld, A., M. Loparco, and C. Barajas, (2003), “Game plan: the insider’s guide to breaking in and succeeding in the computer and video game business,” St. Martin’ s Grin Press, New York, 2003.

Goodman, J. and C. Verbrugge (2008), “A peer auditing scheme for cheat elimination in MMOGs,” In *Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES).* ACM.

Grace, L. (2005), “Game Type and Game Genre,” researchgate.net, Cambridge, MA.

Grammenos, D. (2008), "Game over: learning by dying," in Proceedings of ACM CHI 2008 Conference on Human Factors in Computing Systems, pp. 1443-1452.

Grand, J., and A. Yarusso (2004), “*Game Console Hacking: Xbox, PlayStation, Nintendo, Game Boy, Atari, & Sega*,” Syngress.

Gregory, J. (2009), “*Game engine architecture*,” CRC Press. Chicago

Griffiths, M. (2002), “The educational benefits of videogames,” *Education and Health Journal*, 20 (3), pp. 47-51.

Groff, J., C. Howells, and S. Cranmer, (2010), “The impact of console games in the classroom: Evidence from schools in Scotland,” UK: Futurelab.

Gunn, E. A. A., B. G. W. Craenen, and E. Hart (2009), “A Taxonomy of Video Games and AI,” In AISB.

Gunter, G.A., R. F. Kenny, and E. H. Vick (2008), “Taking educational games seriously: using the RETAIN model to design endogenous fantasy into standalone educational games”, Educational Technology Research and Development, 56 (5/6) (2008, October), pp. 511–537.

Hackenberg, A. J. (2010), “Students’ reasoning with reversible multiplicative relationships,” *Cognition and Instruction*, *28*(4), 383–432.

Hamann, W. (2003) “Goodbye postmortems, hello critical stage analysis,” Gamasutra - The Art & Business of Making Games, July 2003.

Hammersly, T. (2009A), “Planning for fun in game programming – Part 1,” UBM Tech, London, England.

Hammersly, T. (2009B), “Planning for fun in game programming – Part 2,” UBM Tech, Londong, England.

Han, Z., and Z. Zhang (2008), “Integration of Game Elements with Role Play in Collaborative Learning - A Case Study of Quasi-GBL in Chinese Higher Education,” Edutainment, pp. 427-435.

Hannu, K., and M. I. K. Elina (2006), “Playability heuristic for mobile games,” in *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, Helsinki, Finland.

Harrigan, P., and Wardrip-Fruin Noah (2007), “Second Person: Roleplaying and Story in Playable Media,” MIT University Press.

Hays, R.T. (2005), “The effectiveness of instructional games: a literature review and discussion,” *Technical report 2005-004*, Naval Airfare Center Training Systems Division Orlando, FL.

Helgeson, M. (2012), “The Halo Series Has Sold 50 Million Units to Date, ” The GameStop Corporation, Grapevine, TX.

Ho, P. C., S. M. Chung, and M. H. Tsai, (2006), “A case study of game design for e-learning,” In *Technologies for E-Learning and Digital Entertainment* (pp. 453-462). Springer Berlin Heidelberg.

Holmquist, L. E. (2005), “Prototyping: Generating ideas or cargo cult designs?,” Interactions, 12(2), 48-54.

Hoodat, H. and H. Rashidi (2009), “Classification and analysis of risks in software engineering,” World Academy of Science, Engineering and Technology, 56, 446-452. Houde, S. and C. Hill (1997), “What do prototypes prototype,” Handbook of human-computer interaction, 2, 367-381.

Hopper, T. and R. Bell (2000), “Game classification system: Teaching strategic understanding and teactical awareness,” CAHPERD, Vol 66 (4), pp 14-19.

Huang, C. Y., C. H. Hsu, Y. C. Chang, K. T. Chen (2013), “Gaming Any where: An open cloud gaming system,” In *Proceedings of the 4th ACM Multimedia Systems Conference* (pp. 36-47). ACM.

Huizinga, J. (1955), “Homo Ludens,” Beacon Press, 1955.

Hullett, K., N. Nagappan, E. Schuh, and J. Hopson (2011), “Data analytics for game development: NIER track,” In Software Engineering (ICSE), 2011 33rd International Conference on (pp. 940-943). IEEE.

Hunicke, R., M. LeBlanc, and R. Zubek (2004), “MDA: A formal approach to game design and game research,” In Proceedings of the AAAI Workshop on Challenges in Game AI (pp. 04-04).

Hwang, G.J., and P.H. Wu (2010), “Advancements and trends in digital game-based learning research: a review of publications in selected journals from 2001 to 2010,” *British Journal of Educational Technology*, Vol 43 No 1.

Iacucci, G., K. Kuutti, and M. Ranta (2000), “On the Move with a Magic Thing: Role Playing in Concept Design of Mobile Services and Devices,” In DIS2000 (193-202). NY, USA: ACM.

Ibrahim, R., R. Yusoff, H. Mohammed, and H. Jaafar (2011), “Students Perceptions of Using Educational Games to Learn Introductory Programming,” *Canadian Center of Science and Education*, Vol. 4, No.1.

Ifenthaler, D., D. Eseryel, and X. Ge, Eds. (2012), “Assessment in Game-Based Learning: Foundations, Innovations, and Perspectives,” Springer, New York, NY.

Isbister, K. and Schaffer, N. (2008), “Game usability: Advancing the player experience,” Taylor & Francis US.

Izsak, A., E. Jacobsen, Z. de Araujo, and C. H. Orrill (2012), “Measuring mathematical knowledge for teaching fractions with drawn quantities,” *Journal for Research in Mathematics Education, 43*(4).

J. Minogue, L. A., S. Y. Holmes, and M.T. Cheng (2009), “Investigating the impact of video games on high school students' engagement and learning about genetics,” *Computers & Education*, vol. 53, pp. 74-85.

Jakobsson, M. and O. Sotamaa (2011), “Special Issue - Game Reward Systems,” Game Studies 11, 1, n.p.

Järvinen, A., S. Heliö, and F. Mäyrä (2002), “Communication and Community in Digital Entertainment Services,” Prestudy Research Report, pp. 11-12.

Johansson, M. and M. Arvola (2007), “A case study of how user interface sketches, scenarios and computer prototypes structure stakeholder meetings,” In Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1 (pp. 177-184). British Computer Society.

John, St. V. (2013), “Best Practices: Five Tips for Better Playtesting,” UBM Tech, London, England.

Johnson, L., R. Smith, H. Willis, A. Levine, and K. Haywood (2011), “The 2011 Horizon Report,” The New Media Consortium, Austin, Texas.

Jones, A., E. Scanlon, C. Tosunoglu, S. Ross, P. Butcher, and P. Murphy (1996), “Evaluating CAL at the Open University: 15 years on,” *Computers in Education*, 26(1–3), 5–15.

Joyce, A., P. Gerhard, and M. Debry (2009), "How are digital games used in schools: Complete results of the study," European Schoolnet.

Juul, J. (2003), "The Game, the Player, the World: Looking for a Heart of Gameness" In Level Up: Digital Games Research Conference Proceedings, edited by Marinka Copier and Joost Raessens, 30-45. Utrecht: Utrecht University, 2003.

Kabus, P., W. Terpstra, M. Cilia, and A. Buchmann (2005), "Addressing cheating in distributed MMOGs," In *Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)*. ACM.

Kanode, C.M. and H. M. Haddad (2009), "Software engineering Challenges in Game Development*,"* *2009 Sixth International Conference on Information Technology: New Generations*, Kennesaw State University.

Kanter, J. P. and K. Scott (1998), "Understanding thin-client/server computing," (Vol. 1, p. 998). Microsoft Press.

Karmali, L. (2013), "World of Warcraft Down to 7.7 Million Subscribers," IGN Entertainment Incorporated, San Francisco, CA.

Kasvi J. (2000). "Not Just Fun and Games - Internet Games as a Training Medium," Cosiga - Learning with Computerized Simulation Games. pp. 23-34.

Ke, F. (2009), "A qualitative meta-analysis of computer games as learning tools," In R. E. Furdig (Ed.) Handbook of Research on Effective Electronic Gaming in Education (pp. 1–32), New York: IGI Global.

Ketelhut, D. J. (2007), "The impact of student self-efficacy on scientific inquiry skills: an exploratory investigation in River City, a multi-user virtual environment," *The Journal of Science Education and Technology*, 99–111.

Kickmeier-Rust, M. D., B. Marte, S. Linek, T. Lalonde, and D. Albert (2008), "The effects of individualized feedback in digital educational games," *In T. Conolly & M. Stansfield (Eds.) Proceedings of the 2nd European Conference on Games Based Learning*, (pp. 227– 236), October 16—17, 2008, Barcelona, Spain. Reading, UK: Academic Publishing Limited.

Kiili, K. (2005), "Digital game-based learning: Towards an experiential gaming model," The Internet and higher education, 8(1), 13-24.

Kiili, K. (2005A), "Content creation challenges and flow experience in educational games: The IT-Emperor case," *The Internet and Higher Education*, vol. 8, pp. 183-198.

Kiili, K. (2005B), "Educational game design: Experiential gaming model revisited," Research Report 4. Tampere University of Technology.

Kim, C. (2012), "Designing around a core mechanic," UBM Tech, Londong, England.

Kirkley, S.E., S. Tomblin, and J. Kirkley (2005), "Instructional design authoring support for the development of serious games and mixed reality training," *Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*.

Kirkpatrick, D. L. (1994), "Evaluating Training Programs," San Francisco: Berrett-Koehler Publishers, Inc.

Kirriemuir, J. and A. McFarlane (2004), "Literature review in games and learning," Report 8. Bristol. Nesta Futurelab.

Klabbers, J. H. G. (2003), “The Gaming Landscape: A Taxonomy for Classifying Games and Simulations,” *Proceedings, Level Up Digital Games Research Conference*: 54-67, Utrecht, Netherlands.

Klopfer, E., S. Osterweil, and S. Katie (2009), “Moving learning games forward: Obstacles, Opportunities & Openness,” An Education Arcade white paper.

Knutsson, B., H. Lu, W. Xu, and B. Hopkins (2004), “Peer-to-peer support for massively multiplayer games,” In *Proc. of IEEE Int. Conf. on Computer Communications (INFOCOM)*. IEEE.

Koenig, A. D., J. J. Lee, M. Iseli, and R. Wainess (2010), “A Conceptual Framework for Assessing Performance in Games and Simulations,” (CRESST Report 771). Los Angeles, CA: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).

Konzack, L. (2002), “Computer Game Criticism: A Method for Computer Game Analysis,” *Proceeding of Computer Games and Digital Cultures Conference.*

Kulkarni, S., S. Douglas and D. Churchill (2010), “Badumna: A decentralised network engine for virtual environments,” *Computer Networks 54,* 12, 1953 – 1967.

Lacasa, P., L. Méndez, and R. Martínez (2008), “Bringing Commercial Games into the Classroom,” *Computers and Composition,* vol. 25, pp. 341-358.

Langdon, D., G. McKittrick, D. Beede, B. Khan, and M. Doms (2011), “STEM: Good jobs now and for the future,” ESA Issue Brief# 03-11. *US Department of Commerce*.

Lanubile, F., D. Damian, and H. Oppenheimer (2003), “Global Software Development: Technical Organizational and Social Challenges,” Software Engineering Notes.

Larman, C. and V. R. Basili (2003), “Iterative and incremental developments. a brief history,” Computer, 36(6), 47-56. Law, E. (2012), “Evaluation and validation methodologies for adaptive digital educational games,” in An Alien’s Guide to Multi-adaptive Educational Computer Games, M. D.

LTRG (2014a), “CandyFactory Educational Game for iPad,” Learning Transformation Research Group (LTRG), Virginia Tech, Blacksburg, VA.

LTRG (2014b), “CandySpan,” Learning Transformation Research Group (LTRG), Virginia Tech, Blacksburg, VA.

LTRG (2014c), “CandyDepot Educational Game for iPad,” Learning Transformation Research Group (LTRG), Virginia Tech, Blacksburg, VA.

LTRG (2014d), “CandyBot,” Learning Transformation Research Group (LTRG), Virginia Tech, Blacksburg, VA.

Lee, Y., S. Agarwal, C. Butcher and J. Padhye (2008), “Measurement and estimation of network QoS among peer Xbox 360 game players,” In *Passive and Active Network Measurement* (pp. 41-50). Springer Berlin Heidelberg.

Lenhart, A., J. Kahne, E. Middaugh, A. R. Macgill, C. Evans, and J. Vitak (2008), “Teens, video games, and civic,” Pew Internet & American Life Project 1615 L ST., NW – SUITE 700 WASHINGTON, D.C. 20036.

Lewandowski, S. M. (1998), “Frameworks for component-based client/server computing,” *ACM Computing Surveys (CSUR)*, *30*(1), 3-27.

Luton, W. (2009), “Making better games through iteration,” UBM Tech, London, England.

M. Prensky, (2001), “Digital game-based learning,” *McGraw-Hill*, Chapter 1, New York.

Malone, T. (1981), “Toward a theory of intrinsically motivating instruction,” Cognitive Science, 4, 333-369.

Malone, T. W. (1982), “Heuristic for designing enjoyable user interfaces: Lessons from computer games,” Norwood, NJ: Ablex Publishing Corporation, 1982.

Manker, J. (2011), “Game design prototyping,” In Games and Innovation Research Seminar 2011 Working Papers (p. 41).

Manker, J. (2012), “Designscape–A Suggested Game Design Prototyping Process Tool,” Eludamos. *Journal for Computer Game Culture*, 6(1), 85-98.

Manker, J. and M. Arvola (2011), “Prototyping in game design: Externalization and internalization of game ideas,” In Proceedings of the 25th BCS Conference on Human-Computer Interaction (pp. 279-288). British Computer Society.

McDevitt, D. (2010), “A Practical Guide to Game Writing,” UBM Tech, London, England.

Mazzone, E., D. Xu, and J. Read (2007), “Design in Evaluation: Reflections on Designing for Children's Technology,” *in Proceedings of the HCI'07 Conference on People and Computers XXI*, p. 38.

McFarlane, A., A. Sparrowhawk, and Y. Heald (2002), “Report on the educational use of game,” An exploration by TEEM of the contribution which games can make to the education process. London: DfES.

McLeod, J. (2013), “Game Rules,” http://www.pagat.com.

Medler, B., M. John, and J. Lane (2011), “Data Cracker: Developing a Visual Game Analytic Tool for Analyzing Online gameplay,” *CHI 2011.*

Microsoft (2014), “Microsoft Visio, ” Microsoft Corporation, Redmond, WA.

Microsoft, (2013), “Games on Xbox One – Better with Xbox Live Compute,” Microsoft Corporation, Redmond, Washington.

MineCraft Wiki, 2013, “MineCraft,” Mojang AB, Stocholm, Sweden.

Mitchell, B. L. (2012), “Game Design Essentials,” Wiley. com.

Niedenthal, S. (2007), “Real-time sweetspot: The multiple meanings of game company playtests,” DiGRA.

ModDB (2014), “Unreal Engine 4,” DBolical Pty Ltd., Melbourne, Australia.

Mohamed, H. and A. Jaafar (2009), “Challenges in the evaluation of educational computer games. In: 2010 International Symposium on Information Technology, ITSim (2010).

Mojang (2013), “Minecraft,” Mojang AB, Stockhom, Sweden.

Moore, M. E. and J. Novak (2010), “Game Industry Career Guide,” Delmar: Cengage Learning. ISBN 978-1-4283-7647-2.

Morgan S., Linder and C. J. Lokey (2009), “Postmortem: ngmoco/Demiurge Studios WordFu,” UBM Tech, London, England.

Mory, A. (2006), “Game object model version II: a theoretical framework for educational game development,” *Association for Educational Communications and Technology*.

Mueller, F. F. and M. R. Gibbs (2007), “Evaluating a distributed physical leisure game for three players,” in *Proceedings of OZCHI'07, the CHISIG Annual Conference on Human-Computer Interaction*, pp. 143-150.

Mulligan, J., B. Patrovsky and R. Koster (2003), “Developing online games: An insider’s guide,” Pearson Education.

Mwangi, R. W., R. Waweru and C. W. Mwathi (2011),"Integrating Ict with Education: Designing and Educational Computer for Teaching Functions in Undergraduate Mathematics," *Journal of Theoretical and Applied Information Technology*.

Najaran, M. T. and C. Krasic (2010), "Scaling online games with adaptive interest management in the cloud," In *Proc. Of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)*. ACM, 9.

National Science Foundation (2013), "How proficient are U.S. students in math and science?," National Science Board, Arlington, VA.

Neulight, N., Y. B. Kafai, L. Kao, B. Foley, and C. Galas (2007), "Children's participation in a virtual epidemic in the science classroom: Making connections to natural infectious diseases," *Journal of Science Education and Technology*, 16(1), 47-58.

Norton, A. (2008), "Josh's Operational Conjectures: Abductions of a Splitting Operation and the Construction of New Fractional Schemes," *Journal for Research in Mathematics Education 39* (4), 401-430.

Norton, A., and Wilkins, J.L.M. (2009), "A Quantitative Analysis of Children's Splitting Operations and Fraction Schemes," *Journal of Mathematical Behavior 28*, 150-161.

O'Brien, D., K. A. Lawless, and P. G. Schrader (2010), "A Taxonomy of Educational Games," In Baek, Y. (Ed.), Gaming for Classroom-Based Learning: Digital Role Playing as a Motivator of Study. (pp. 1-23).

Olive, J. and G. Çağlayan (2008), "Learner's difficulties with quantitative units in algebraic word problems and the teacher's interpretations of those difficulties," *International Journal of Science and Mathematics Education, 6*, 269–292. doi:10.1007/s10763-007-9107-6.

Oliver, M. (2000), "An introduction to the evaluation of learning technology," *Educational Technology & Society*, 3(4), 20–30.

Orca Computer (1999a), "*Evaluation Environment User's Guide*," Orca Computer, Inc., Blacksburg, VA.

Orca Computer (1999b), "*Evaluation Environment Subject Matter Expert User's Guide*," Orca Computer, Inc., Blacksburg, VA

Otoy, Inc. (2014), "octanerender," Otoy Incorporated, Los Angeles, CA.

Owston, R., H. Wideman, N. S. Ronda, and C. Brown (2009), "Computer game development as a literacy activity," *Computers & Education*, vol. 53, pp. 977-989.

Pagulayan, R. J., K. Keeker, D. Wixon, R. L. Romero, and T. Fuller (2003), "User-centered design in games," NJ: Lawrence Erlbaum Associates.

Papastergiou, M. (2009), "Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation," *Computers & Education*, vol. 52, pp. 1-12.

Pappa, D. and L. Pannese (2010), "Effective Design and Evaluation of Serious Games: The Case of the e-VITA Project," WSKS (1) 2010: 225-237.

Pei-Chi, H., C. Szu-Ming, and T. Ming-Hsin (2006), "A case study of game design for e-learning," Springer Berlin, Heidelberg.

Peixoto, D.C.C., R.M. Possa, R.F. Resende, and C.I.P.S. Pádua (2011), "An overview of the main design characteristics of simulation games in Software Engineering education," *Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on*, pp.101-110, 22-24 May 2011.

Petrillo, F., M. Pimenta, F. Trindade, and C. Dietrich (2008), “Houston we have a problem: Asruvey of actual problems in Computre Games Development,” *Proceedings of the 2008 ACM Symposium on Applied Computing*, Fortaleza, Ceara, Brazil, March 16 - 20, 2008, SAC '08. ACM, New York, NY, pp707-711.

Pinelle, D., N. Wong, and T. Stach (2008), “Using genres to customize usability evaluations of video games,” pp. 129-136.

Prensky, M. (2003), “Digital Game Based Learning: Exploring the Digital Generation,” Educational Technology, U.S. Department of Education.

Prensky, M. (2007), “Digital Game-Based Learning,” Paragon House, St. Paul, Minnesota.

Pressman, R. S. (2010), “Software Engineering: A Practitioner’s Approach,” 7th Edition, McGraw- Hill.

Pressman R. S. and B. R. Maxim (2015), “Software Engineering: A Practitioner's Approach,” 8th Edition, McGraw-Hill Education, New York, NY.

R. Simon, W. Klaus, F. Marc, N. Heiko, P. Leo, and C. Georg (2007), “Peer-to-Peer-Based Infrastructure Support for Massively Multiplayer Online Games,” In *Proceedings of the 4th Consumer Communications and Networking Conference (CCNC)*, pages 763–767, IEEE.

OnLive (2014), “Onlive Cloud Gaming Service,” OnLive, Mountain View, CA.

Reimers, F. M. (2008), “Preparing students for the flat world,” Education Week,10/8/2008, Vol. 28 Issue 7, p24.

Rigby, S. and R. Ryan (2007), “The player experience of need satisfaction (PENS) model,” Immersyve, Inc.

Rimal, B. P., E. Choi and I. Lumb (2009), “A taxonomy and survey of cloud computing systems,” In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on* (pp. 44-51). IEEE.

Robertson, D. and D. Miller (2009), “Learning gains from using games consoles in primary classrooms: a randomized controlled study,” *Procedia - Social and Behavioral Sciences,* vol. 1, pp. 1641-1644.

Robertson, J. and C. Howells (2008), “Computer game design: Opportunities for successful learning,” *Computers & Education,* vol. 50, pp. 559- 578.

Rogers, S. (2010), “Level Up!: The Guide to Great Video Game Design,” John Wiley & Sons, p. 443.

Rollings, A. and E. Adams (2003), “Andrew Rollings and Ernest Adams on game design,” New Riders.

Rosenberg, D. (1999), “Use case driven object modeling with UML,” (pp. 1-4). K. Scott (Ed.). Massachusets: Addison-Wesley.

Rouse III, R. (2010), “Game design: Theory and practice,” Jones & Bartlett Learning.

Rouse R. (2004), “Game Design, Theory and Practice,” Wordware Publishing Inc.; 2nd Revised edition edition, p. 704.

Rupp, A. A., M. Gushta, R. J. Mislevy, and D. W. Shaffer (2010), “Evidence-centered design of epistemic games: Measurement principles for complex learning environment,” *Journal of Technology,* Learning, and Assessment, 8(4).

Ryan, T. (1999), “The Anatomy of a Design Document, Part1: Documentation Guidelines for the Game Concept and Proposal,” UBM Tech, London, England.

Rieche, S., K. Wehrle, M. Fouquet, H. Niedermayer, L. Petrak, and G. Carle. (2007), “Peer-to-peer-based infrastructure support for massively multiplayer online games,” *In Proc. of the 4th IEEE Consumer Communications and Networking Conference*, pages 763-767. IEEE, 2007.

Salazar, M. G., H. A. Mitre, C. L. Olalde, and J. L. G. Sánchez (2012), "Proposal of Game Design Document from software engineering requirements perspective," In Computer Games (CGAMES), 2012 17th International Conference on (pp. 81-85). IEEE.

Salen, K. (2007), "Gaming literacies: A game design study in action," *Journal of Educational Multimedia and Hypermedia*, 16(3), 301-322.

Salen, K. and E. Zimmerman (2003), "Rules of Play," MIT Press, 2003.

Salen, K. and E. Zimmerman (2004), "Rules of play: Game design fundamentals," Cambridge, MA: MIT Press.

Schach, S. R. (2002), "Object-Oriented and Classical Software Engineering," Fifth Edition, McGraw-Hill Higher Education, New York, NY.

Schell, J. (2008), "The Art of Game Design: A book of lenses," Taylor & Francis US.

SEI (2010), "Capability Maturity Model Integration (CMMI) for Development, Version 1.3," Software Engineering Institute (SEI), Carnegie Mellon University, Pittsburgh, PA.

Sellen, K. M., M. A. Massimi, D. M. Lottridge, K. N. Truong, and S. A. Bittle (2009), "The people-prototype problem: understanding the interaction between prototype format and user group," In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 635-638). ACM.

Sellers, M. (2005), "Designing the Experience of Interactive Play," In *Playing video games: Motives, responses, consequences.* Edited by Vorderer, P. & Bryant, J. Mahwah: Lawrence Erlbaum Associates.

Shaffer, D. W. (2006), "How computer games help children learn," New York: Palgrave Macmillan.

Shea, R., Liu, J., Ngai, E. H., & Cui, Y. (2013), "Cloud gaming: architecture and performance," *Network, IEEE*, *27*(4).

Shubik, M. (1983), "Gaming: A state-of-the-art-survey," In Ståhl, I (ed) Operational Gaming, Pergamon Press, Oxford, 1983.

Shute, V. J. and Y. J. Kim (2011), "Does playing the World of Goo facilitate learning?," In D.Y. Dai (Ed.), *Design research on learning and thinking in educational settings: Enhancing intellectual growth and functioning* (pp. 359–387). New York, NY: Routledge Books.

Smith, P.L. and T.J. Ragan (2005), "Instructional Design," 3rdEd., John Wiley & Sons, New Jersey.

Snyder, C. (2003), "Paper prototyping: The fast and easy way to design and refine user interfaces," Morgan Kaufmann.

Sommerville, I. (2007), "Software engineering," 9th Edition, Addison-Wesley.

Sommerville, I. (2011), *Software Engineering*, 9th Edition, Addison-Wesley / Pearson Education, Boston, MA.

Song, S. and J. Lee (2007) "RETRACTED: Key factors of heuristic evaluation for game design: Towards massively multi-player online role-playing game," *International Journal of Human-Computer Studies,* vol. 65, pp. 709-723.

Squires, D. and A. McDougall (1994), "Choosing and using educational software: A teachers' guide," London: The Falmer Press.

Stefan Marks, J. W. and B. W¨unsche (2007), "Evaluation of Game Engines for Simulated Surgical Training," in *GRAPHITE 2007*, Perth, Western Australia.

Steffe, L. P. (1992), "Schemes of action and operation involving composite units," *Learning and Individual Differences*, *4*(3), pp. 259-309.

Steinkueler, C. and M. Chmiel (2006), “Fostering scientific habits of mind in the context of online play,” ICLS 2006 proceedings of the 7th international conference on learning sciences.

Stuart, K. (2012), “Genius or elimination: where do good game ideas come from?,” Guardian News and Media Limited, London, England.

Suznjevic, M. and M. Matijasevic (2012), “Player behavior and traffic characterization for MMORPGs: a survey,” *Multimedia Systems*, 1–22.

Sykes, J. (2006), “A player-centred approach to digital game design,” Understanding Digital Games. Sage Publications, London, New York & New Delhi, 75-92.

Sykes, J. and M. Federoff (2006), “Player-centered game design,” In CHI'06 extended abstracts on Human factors in computing systems (pp. 1731-1734). ACM.

Talbert, M.L. (1995), “A methodology for the measurement and evaluation of complex system designs,” Ph.D. Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA, Dec.

Tan, C. T. and A. Johnston (2011), “Towards a Non-Disruptive, Practical and Objective Automated Playtesting Process,” In *Artificial Intelligence in the Game Design Process*.

Tang, S. and M. Hanneghan (2010), “A Model-Driven Framework to Support Development of Serious Games for Game-based Learning,” *Developments in E-systems Engineering (DESE)*, School of Computing & Mathematics Science, Liverpool John Moores University, Liverpool, UK.

Tapping America's Potential (TAP) Coalition (2008), “Gaining Momentum, Losing Ground,” Progress Report 2008, Business Roundtable, Washington, DC.

Tarr, P., H. Ossher, W. Harrison, and S. M. Sutton Jr (1999), “N degrees of separation: multi-dimensional separation of concerns,” In *Proceedings of the 21st international conference on Software engineering* (pp. 107-119). ACM.

The International Arcade Museum (1995), “International Arcade Museum,” The International Arcade Museum

Thompson, J., B. Berbank-Green, and N. Cusworth, (2007), “The computer game design course: principles, practices and techniques for the aspiring game designer," Thames & Hudson, London.

Thorpe, R., D. Bunker, and L. Almond (Eds.) (1986), “Rethinking games teaching,” Loughborough: University of Technology, Loughborough.

Transue, B. (2013), “Characteristics of Simulation Games in Edutainment,” http://bethtransue.wordpress.com/2013/02/13/characteristics-of-simulation-games-in-edutainment/.

Trefry, G. (2010), “Casual game design: Designing play for the gamer in all of Us,” Taylor & Francis US.

U.S. Department of Education, (2010), “Transforming American education: Learning powered by technology,” *National Education Technology Plan 2010.*

Unity Technologies (2014), “Create the game you love with Unity,” Unity Technologies, San Francisco, CA.

Urbaczewski, L. and S. Mrdalj (2006), “A Comparison of Enterprise Architecture Frameworks”, *Issues in Information Systems 7, 2*, 18-23.

Varvello, M., S. Ferrari, E. Biersack and C. Diot (2009), “Distributed avatar management for Second Life,” In *Proc. of Int. ACM SIGCOMM Workshop on Network & System Support for Games (NETGAMES)*. IEEE, 5:1–5:6.

Vogel, J. J., D. S. Vogel, J. Cannon-Bowers, C. A. Bowers, K. Muse, and M. Wright (2006), "Computer gaming and interactive simulations for learning: a meta-analysis," *Journal of Educational Computing Research*, 34(3), 229–243.

Wall, J. and V. Ahmed (2008), "Use of a simulation game in delivering blended lifelong learning in the construction industry - Opportunities and Challenges," *Computers & Education,* vol. 50, pp. 1383-1393.

Wang, T. H. (2008), "Web-based quiz-game-like formative assessment: Development and evaluation," *Computers & Education,* vol. 51, pp. 1247-1263.

Watson, W.R. (2007), "Formative research on an instructional design theory for educational video games," Thesis Doctor of Philosophy, IndianaUniversity.

Watt, A. and Policarpo, F. (2003), *3D Games, Vol. 2: Animation and Advanced Real-Time Rendering,* Addison-Wesley Longman Publishing Co., Inc..

Werner, P. and L. Almond, (1990), "Models of games education," JOPERD, 61(4), 23-27.

Wikipedia (2013b), "Sports game," Wikimedia Foundation, San Francisco, CA.

Wolf, M. J. P. and R. H. Baer (2002), "The Medium of the Video Game," University of Texas Press, June 2002

Wolters, J. (2012), "Playtesting: Preparation and Execution," http://www.joramwolters.com/playtesting-preparation-and-execution/

Xiaochun, X., Guanghui, X. and Yongsen, X. (2002), "Architectural Issues in Network-Centric Computing," *ACM SIGSOFT Software Engineering Notes 27, 1,* 53-57.

Yuan, B., E. Folmer, and F. C. Harris (2010), "Game Accessibility: a Survey," *Universal Access in the Information Society, 10 (1) (2010), pp. 81–100*

Zhang, S., and S. Goddard (2007), "A Software Architecture and Framework for Web-based Distributed Decision Support Systems," *Decision Support Systems 43*, 1133-1150..

Zhang, W., H. Mei, and H. Zhao (2006), "Feature-driven requirement dependency analysis and high-level software design," *Requirements Engineering, 11*(3), 205-220.

Zimmerman, E. (2006), "The game design reader," K. Salen (Ed.). Cambridge: MIT press pp. 21-22.

Zin, D. A. M., A. Jaafar, and W. S. Yue (2009), "Digital Game-based learning (DGBL) model and development methodology for teaching history," *WSEAS Transactions on Computers*, Issue 2, Volume 8.

Zynga Inc. (2014), "Farmville," Zynga Incorporated, San Francisco, USA.