Towards a Serious Game to Teach ISO/IEC 12207 Software Lifecycle Process: An interactive learning approach

Ufuk Aydan[1,2] and Murat Yilmaz[1,2] and Rory V. O'Connor[3]

[1] Çankaya University, Game Research & Development Laboratory, Turkey
aydanufuk@gmail.com
[2] Çankaya University, Department of Computer Engineering, Turkey
myilmaz@cankaya.edu.tr
[3] School of Computing, Dublin City University, Ireland
roconnor@computing.dcu.ie

Abstract. ISO/IEC 12207 training is a key element to provide an ability to software development organizations for selecting a set of required processes, measuring the performance of these processes, and continuously improving them. Traditionally, such training is either performed by an expert individual to the software quality management personnel most likely in form of a seminar in a classroom environment. This may also be given by a suitability qualified professional, such as a registered auditor. However, software requirements are usually subject to change, and therefore such training is not enough to teach the substantial details of the entire standard. This has led to increased reports of complications, which demotivates organization to use this standard. To improve the quality of training, a 3D serious game was proposed for the software practitioners. The preliminary idea here is that the game employs 3D office landscape to provide a realistic virtual environment ensuring that the training will be based on real word like situations. Before building a prototype for our serious game, we consulted five industrial experts whose works are related with ISO standards. To give these practitioners an opportunity to explore the conceptual design and raise some potential problems, the semi-structured interview method was used. Based on the suggestions of experts, dynamics and mechanics of the proposed game were updated. Taken together, initial results suggest that a serious game for teaching ISO/IEC 12207 should be useful for individuals who are interested to learn more about the standard.

1 Introduction

ISO/IEC 12207 is an international software engineering standard which defines the software engineering processes and activities, which are associated with software life cycle process from conception to the end of product [1]. This standard defines a set of suitable roles for software practitioners and follows the plan-do-check-act cycle for improving the quality of the product [2].

ISO/IEC 12207 is not a specific standard for a product. Moreover, it does not measure the quality, it does not define specifically how to do activities and tasks, and it does

not prescribe to specific methods, practices or tools. Its modular structure is suitable for tailoring. Therefore, an organization can customize the necessary parts of the standard that are planned to be used based on the requirements of a software project [3]. Because of the high modularity of the standard, it is more easily to deal with factors that are affecting the software development such as complexity, schedule, cost, etc. In addition to that ISO/IEC 12207 can act as an *inventory* of processes, which give different perspectives to specific parts of the software life cycle process. These processes are categorized as organizational processes (i.e. management, infrastructure, improvement, training), supporting processes (i.e. documentation, configuration management, quality assurance, verification, validation, joint review, audit, problem resolution), and primary processes (i.e. acquisition, supply, development, operation, maintenance) [4].

ISO/IEC 12207 avoids strangling itself with this list of process descriptions by giving guidance on how to adopt a role, and it gives sample criteria for how to choose the processes, activities and tasks that are needed. It offers the concept of *views* to help identify the processes attached to a role. There are five views: (i) contract view: acquisition process, for the acquirer; supply process, for the supplier; (ii) engineering view: development process, for producing the products(s); maintenance process for modifying the software and keeping it current; (iii) operating view: the operation process for what to do to operate the software; (iv) quality management view: there are six processes for this: joint review process, audit process, verification process, validation process, quality process, and problem resolution process; (v) management view: the management process is to be used by any organization for managing its process(es) [3]. ISO/IEC 12207 Software Lifecycle processes can be maintained by 7 main phases by any organization which have capability to support the standard's views and ability to handle software engineering requirements. These main phases are; Requirements Analysis, Specification, Design, Coding, Verification & Validation, Installation, Maintenance & Support.

Despite the fact that ISO/IEC 12207 is a well-structured and detailed technical text on complex subject, many professionals find it difficult to gain substantial information regarding to the standard. Games are found to be effective learning tools especially for teaching complex subjects. In particular, serious games are a kind of interactive computer application (i.e. computer simulations of real-life situations or processes) designed for educational purposes. As a serious game is designed to include an educational aspect [5], the learners can be challenged with possible scenarios that may found similar to real-word problem. However, a well-designed serious game should include game playing and a set of serious aspects (e.g. teaching, learning, communication, information, etc.) where such combination should be based on an utilitarian goals [6]. In fact, this scenario should be aligned with gaming objectives that implements the dramatic elements of a game such as story, sound, rules, graphics, etc.

In light of this remarks, the goal of the study is to investigate the possibilities of a game that is designed for teaching the primary concepts of ISO/IEC 12207. The rest of the paper is organized as follows. Section 2 presents the background of the study. It details of the ISO/IEC 12207 standard and the notion of serious games. Section 3 includes a discussion about the customization of the standard. Further, it details the applications of serious games in software engineering. Next, we discuss the tentative plans for an ideal game. Lastly, paper concludes with conclusion and future work.

2 Background

2.1 ISO/IEC 12207

Similar to the definition of ISO/IEC 12207, definition the life cycle starts with an idea or a specific need that is about software and ends with the retirement of the software or software service. The standard has an architecture which is built by set of interrelated processes, which are consequent to modularity and responsibility. While defining modularity under the conceptualization of the standard is about being unique with every processes and availability of being capable enough to handle all types of projects. The processes are modular; that is, they are maximally cohesive and minimally coupled to the practical extent feasible. An individual process is dedicated to a unique function [4]. Responsibility on the other hand means the responsibility of any party in the software life cycle process. To clarify every party which is associated with the life cycle has specific and well defined responsibility to take care. This is in contrast to a ”text book approach,” where the life cycle functions could be studied as topics or subjects, such as management, design, measurement, quality control, etc [4]. Additional to responsibility and modularity, there are some basic concepts that need to be defined explicitly. The standard defines Organization and Party. To define the difference between an organization and the party there is a need to look from standards view. In this standard, the terms ”organization” and ”party” are closely related. An organization is a body of persons with identified responsibilities and authorities organized for some specific purpose, such as a club, union, corporation, or society. When an organization, as a whole or a part, enters into a contract, it is a party. Parties may be from the same organization or from separate organizations. An individual is an example of an organization, if the individual is assigned responsibilities and authorities. An organization or a party derives its name from the process for which it is responsible. For example, it is called an acquirer when it performs the Acquisition Process. So, when the following terms are used in this standard, they do not have their generic meaning, but instead, refer to the organization or party responsible for executing the process with a similar name: acquirer, supplier, implementer, maintainer, and operator [7]. The processes are grouped into three broad classes: primary; supporting; and organizational (see Figure 1).

Primary processes are the prime movers in the life cycle; they are acquisition, supply, development, operation, and maintenance. Supporting processes are documentation, configuration management, quality assurance, joint review, audit, verification, validation, and problem resolution. A supporting process supports another process in performing a specialized function. Organizational processes are management, infrastructure, improvement, and training. An organization may employ an organizational process to establish, control, and improve a life cycle process.

This International Standard groups the activities that may be performed during the life cycle of a software system into seven process groups. Each of the life cycle processes within those groups is described in terms of its purpose and desired outcomes and lists activities and tasks which need to be performed to achieve those outcomes (see Figure 2).

(a) Agreement Processes two processes (subclauses 5.2.2.1.1 and 6.1)

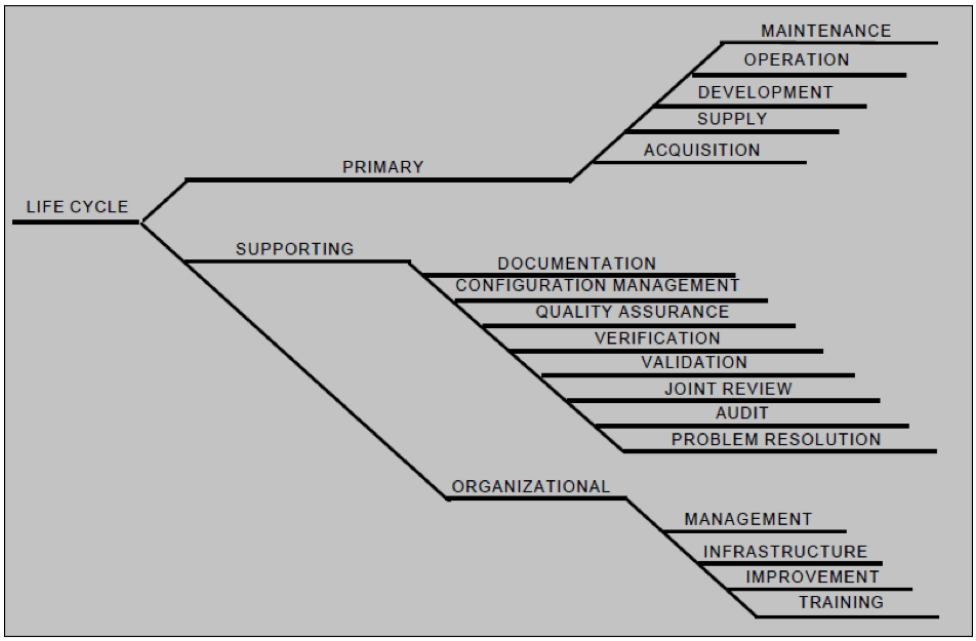

Fig. 1: *The Life Cycle Processes of ISO/IEC 12207*

(b) Organizational Project-Enabling Processes five processes (subclauses 5.2.2.1.2 and 6.2)
(c) Project Processes seven processes (subclauses 5.2.2.1.3 and 6.3)
(d) Technical Processes eleven processes (subclauses 5.2.2.1.4 and 6.4)
(e) Software Implementation Processes seven processes (subclauses 5.2.2.2.1and 7.1)
(f) Software Support Processes eight processes (subclauses 5.2.2.2.2 and 7.2)
(g) Software Reuse Processes three processes (subclauses 5.2.2.2.3 and 7.3) [2].

Basically, the structure of ISO/IEC 12207 software life cycle process was designed in terms of relative constituent parts. For instance, each process has its own activities that cover cohesive tasks.even tasks have tiney actions. A task consumes inputs (data, information, control) and produces outputs (data, information, control). It is expressed in the form of self-declaration, requirement, recommendation, or permissible action. For this purpose, the standard carefully employs certain auxiliary verbs (will, shall, should, and may) to differentiate between the distinct forms of a task. ”Will” is used to express a self-declaration of purpose or intent by one party, ”shall” to express a binding provision between two or more parties, ”should” to express a recommendation among other possibilities, and ”may” to indicate a course of action permissible within the limits of the standard. ”Must,” which denotes a mandatory action, is never used in the standard [4].

The standard implements the quality management principles which are the integral and indispensable parts of the total life cycle. Each process is circumvented with a plan-do-check-act (PDCA) cycle. Therefore, each process and the personnel who are

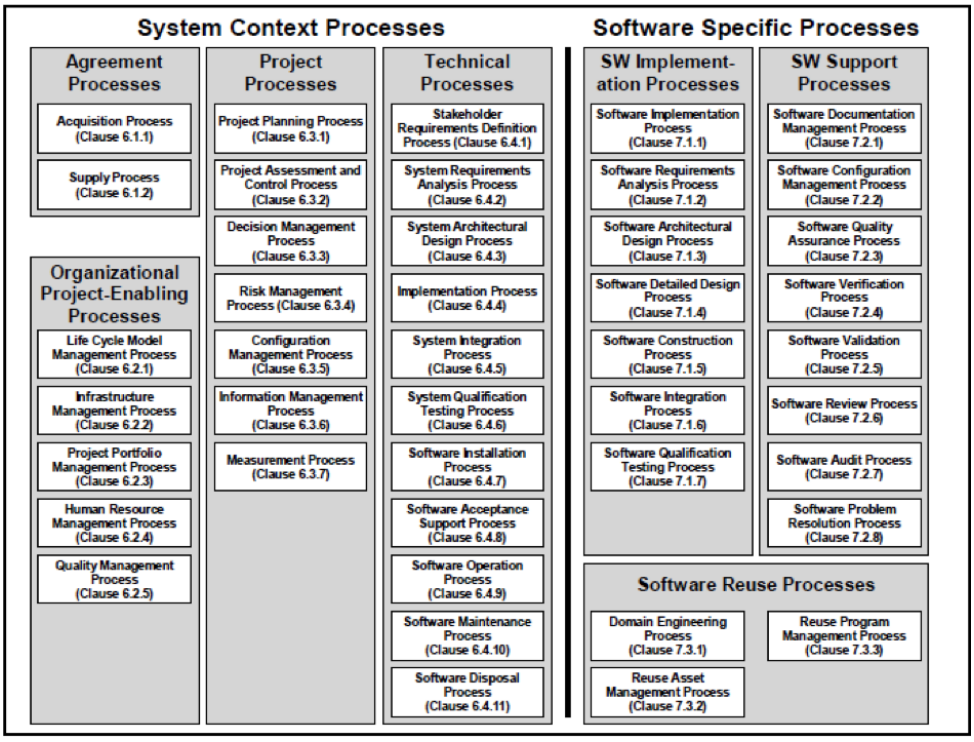

Fig. 2: *Life Cycle Process Groups*

the responsible of related processes must be aware of their particular responsibility in verification and validation phases.additionally, evalutaions of particular tasks have to be carried out by organization. The processes are conducted by the acquirer, the supplier, or an independent party to verify and validate the products in varying depth depending on the project. These evaluations do not duplicate or replace other evaluations, but supplement them [7].

ISO/IEC 12207 requires outputs and these outputs have to be documented, but there is no specific or predefined format for any types of output to be documented. The organization can use their documentation methods also get benefit from standards. In addition to documentation base-lining is important issue. The standard requires the baselines of software related tasks and activities such as software requirements, software design, and the code. Base-lining, when used judiciously, is an effective means for establishing confidence in milestones and controlling costs and schedules by inhibiting unnecessary, unplanned, or open changes to requirements, design, and code [4]. Particularly base-lining is occurred while joint review or an audit processes in order to clarify and solidify acquirer-supplier understandings. However, it is not necessary for projects to perform base-lining which is the responsibility of the Development Process. It is not related with Configuration Management Process and it is not a must. Consequently, ISO/IEC 12207 covers the total software development life cycle and it is relatively so-

phisticated and sometimes can be complicated especially from the point of different and independent parties in the life cycle. Reading the standard outside the context of an organization's objectives and a project's needs may negatively impact proper interpretation of the standard. For effective and productive use of the standard, the following prerequisites (in that order) should be met:

(a) Trained personnel;
(b) Familiarity with the organization's policies;
(c) Familiarity with the project's environment;
(d) An understanding of the standard [4].

2.2 Customization of ISO/IEC 12207

Although ISO/IEC 12207 software lifecycle standard shows a set of agreements of experts on some procedures for software development, there is no one-size-fits-all type of selection and tailoring of processes. However, the responsibilities of both acquirer and supplier are considerably important in the tailoring process. According to the *acquisition process* defined by ISO/IEC 12207 [8], success of the supplier depends largely on explicit definition of the acquirer's expectations, in terms of system requirements and with respect to the software development process.

There is a common point of view to the quality of any software product largely depends on the software development process. Additionally, all successful companies in any size follow well-defined activities and tasks in a timely manner. Another factor that increase the success of further development and reduce the current project's problems is receiving adequate feedback from both user and prior projects. To accomplish these factors and to improve the development process, companies invest on improvement activities, but great numbers of companies rely on the success of the employment of ad-hoc processes, which relies on individual's skills. This approach is very difficult to repeat, and has a detrimental effect on product quality, maintainability, cost, and timeliness [8]. Moreover too many reasons can be stated for the failure of process because of the absence of enough knowledge, lack of customer feedback and market related problems.

To overcome risks, problems, and inadequate development process and to improve the quality, there is a need to follow some standardized methods. Due to these reasons ISO/IEC 12207 is one of the recent and valid to use standard that includes the steps that should be followed by contractors. Tailoring process shows itself here more clearly at the acquisition process and acquirer has a chance to tailor ISO/IEC 12207. This approach is totally beneficial to the suppliers side because it creates guidance to how to follow, proceed and determine quality related activities that carry the project to the success with reduced risks and problems. Under tailoring circumstances such as novelty, size, budget, risks, technology, time etc. shall be inspected.

3 Concept of Serious Games and Examples

An applied game or serious game is a game designed for a primary purpose other than pure entertainment [9]. Serious games are designed with the intention of improving

some specific aspect of learning, and players come to serious games with that expectation. Serious games are used in emergency services training, in military training, in corporate education, in health care, and in many other sectors of society. They can also be found at every level of education, at all kinds of schools and universities around the world. Game genre, complexity, and platforms are as varied as those found in casual games. Play is an important contributor to human development, maturation, and learning, which is a mandatory ingredient of serious games [5].

In other words, serious games are kind of simulations of real-world events or processes that are addressed to comprise particular problems. Serious games sometimes despitefully sacrifices being funny. However, they can be entertaining and funny, if their main purpose covers game elements well. Substantially, they have many attributes which have been seen in the case of different examples. For instance, serious games allow user to realize different learning environment with entertaining elements. Another example of attributes is stating how actions affect the context. Players can create artifacts or complete tasks within in the orders of a serious game serves and without the effects of real world problems and stress. This can be interpreted to resembling sand box type games. Moreover, serious games allow users have an active role while accomplishing the main goal. Applied games are a very powerful tool, because they have the ability to change behavior [1]. Furthermore, serious games protect users from repetitive actions and manners while learning certain subjects. Because particular tasks and clearly stated objectives of serious games make player easier to follow certain pathways and play the role of necessary behaviors. Closely linked to behavior is learning, which is also something that applied gaming can assist with, especially for learning how to do things [2].

There are numerous works which are related about serious games and applications. However in the literature there are only several serious games which are related to software project management. These are; Problems and Programmers [10], SIMSOFT [11], SimSE [12], SESAM [13], DELIVER [14], ProDec [15].

Problems and Programmers [10], an educational card game that simulates the software engineering process and is designed to teach those process issues that are not sufficiently highlighted by lectures and projects. Problems and Programmers is a teaching tool, and as such its purpose is to educate [10]. Among computer based digital variants of serious games, Problems and Programmers is created as a physical card game. The complete game state is present on the table; the results of actions are immediately visible; and, because Problems and Programmers is set up as a competitive game, interactive games ensues in which students learn from each other [16]. The game is a multiplayer set up where two or more players try to complete a software project. The first and foremost goal of Problems and Programmers is to enrich a students' understanding of the software process [16] These attributes strengthen the main goal of the game and reveal the games powerful sides such as covering proper use of software engineering techniques. Providing clear and instant feedback pf players choices and decisions towards phases. Encouraging interactions among different players and evaluating their perspectives. Because different players follow different strategies, more than one strategy is exposed per game. This allows players to not only evaluate their own strategy but to also discuss and compare strategies followed by others. As a result,

players learn from each other, which enhances the educational value of Problems and Programmers [16] .

SimSE is an interactive, graphical, educational soft-ware engineering simulation game designed to teach students the process of software engineering [12],. SimSE is a single-player game in which the player takes on the role of project manager of a team of developers. As the player manages the process to complete (a particular aspect of) a software engineering project, they can, among other things, hire and fire employees, assign tasks to them, monitor their progress, and purchase tools [12],. The most likely the reason of creating educational software engineering applications is students from every educational level are exposed to theoretical concepts of software engineering. There is no sufficient or relatively sufficient project which converts lectures into practices. SimSEs main goal is fulfilling this absence by providing 2D graphical virtual office where software engineering processes take place. This office scene includes many office staff such as computers, desks, employees, and artifacts. Moreover, it is an interactive which means, it should operate on a step-by-step basis, accepting user input and providing feed-back constantly throughout the simulation [12],. This approaches clarify the players actions with the support of selection of the available moves and steps the player will proceed. Perhaps the most discriminative specification of SimSE among other variants is including a model builder which allows user to create employees, artifacts, tools, projects, and customers without any need of programming skills. Via this modeler SimSE supports customization of the software processes because real-world scenarios about software processes vary with different organizations.

Project Decision (ProDec) is a simulation-based serious game created with the intention to train and assess students in software project management [15]. The main objective is to take advantage of the engaging nature of games to place the learners in a virtual organization where they can manage software projects and solve real-life problems in a risk-free environment [15]. The main goal of the game is to remain aware of planning, controlling, and managing a software project. The game is over to the extent permitted by the amount of budget the players have and allocated time for the project. While gameplay players need to plan to deal with obstacles which are created by unplanned events. ProDec is intended to be a collaborative game, that is, it is a game to be played by teams of players [15]. This means that the group of players works collaboratively to win the game not to compete among them [15]. After any game play, ProDec offers a complete report including the logs representing every decision the players made and the result of applying the assessment criteria provided by the trainer at the beginning of the game play [15].

Simsoft [11] is a kind of serious game which consists of two game boards, a printed board and a digital board. A0-sized printed game board around which the players gather to discuss the current state of the project and to consider their next move. The board shows the flow of the game while plastic counters are used to represent the staff of the project. Poker chips represent the teams budget, with which they can purchase more staff, and from which certain game events may draw or reimburse amounts depending on decisions made during the course of the game [11]. Additional to the physical board there is also a Java-based game board where players can see the current and historical state of the project through a series of simple reports, messages, and other information

and also can adjust the projects settings, for example to recruit new staff, before advancing the games time to create the state of the project [11]. The main goal of the game is completing the given project on time and with significant poker chips left over. Simsoft players are formed into teams of two or three or more and they are given a scenario that describes the requirements for a small software development project. Taking the role of project manager, the team must manage the project from start-up to final delivery [11]. SIMSOFT mainly focuses on human resource management, with an emphasis on how the ability of the staff affects the outcomes of the project.

In addition, SESAM is a natural language based serious game which motivates players to gain software project management techniques. SESAMs environment consists of natural language interface, records about program and statistics about project. Lastly, DELIVER is another type of serious game which consists of a printed board. It helps students to develop controlling projects performances. Its main ambition is totally motivate students in their learning progression.

Defining the solution of efficiency of managerial skills in software project management to a game environment may lead disagreement in choosing such a way which is playing a game is capable enough to get concepts is software engineering. Good video games incorporate good learning principles, principles supported by current research in Cognitive Science [17]. If a period of learning has a chance to reform in a more enjoyable way, every known problems solutions become easier to everyone. Why? If no one could learn these games, no one would buy them- and players will not accept easy, dumbed down, or short games [18]. If anything seems motivating and entertaining everyone easily enjoy learning process of any concept.

4 Discussion on planning the ideal game

Typical software engineering methods and principles are not quite easy to learn and implement because of the lack of motivation and absence of fine learning period in lectures. In addition to this, technical developments and evolving industry demands change rapidly and there is a need for learning broader concepts about software engineering and management tasks, rather than simply getting general knowledge that was given in every introductory level software engineering course. The problem is not simple as being only hard-to-learn. The main point is being unable to make easier to understandable for everyone either managerial or technical levels of software engineering topics in the market. Due to this no consistency of an adequate education there are too many projects have been struggling to go further. For every six new large-scale software systems that are put into operation, two others are canceled. The average software development project overshoots its schedule by half; larger projects generally do worse. And some three quarters of all large systems are *operating failures* that either do not function as intended or are not used at all [19]. Undoubtedly, getting the big picture of today's current circumstances and state of the software engineering and project management are difficult, but obvious thing is the failures and problems are becoming more visible for everyone to recognize. To reduce failures and problems there are crucial jobs which are waiting for project managers to accomplish. Because project management skill are the key factors which have potential to overcome risks in development environment,

and this skills may lead the way of engineers, developers and even students who are the future responsible of life cycle of software. However, current view of project life cycles in any kind of software development implies that managerial skills need more practical experience to become mature enough to produce more stable projects with less risks and problems. From the student side of this problem can be re-stated with similar reasons, although being the future engineers and managers there are too many inconsistencies between market and syllabuses. A large difference exists between the software engineering skills taught at a typical university and the skills that are desired of a software engineer by a typical software development organization [20]. Apparently software engineering concept is given to students in a more theoretical sense. In contrast this should be more practical with supporting projects which resemble to real life scenarios. Even they come together there is a need for in depth procedure to follow. In particular, lectures allow only passive learning, and the size and scope of class projects are too constrained by the academic setting to exhibit many of the fundamental characteristics of real-world software engineering processes [20]. To deal with these problems one possible and feasible way is using serious game which is suitable for specific concept. Serious games are designed to teach especially educate players about desired topics in a well-defined environment [5].

To define the learning process in a clear sense, it can be repeated in any time without practicality, because in theoretical approach information is static not dynamic so there should be actions in several ways to change it to dynamic which helps to resolve the problem. No matter what the type is every game should clearly define some attributes while game play even before. This is more decisive when the serious gaming concept is under concern, because every player who is playing a serious game he/she ought to be conscious about what will he/she do, accomplish and learn. Desired commitment to the learning journey requires well defined and planned program. For instance without clearly defined characters or personalities in the game it is difficult to expect anyone to commit himself to a specific progress. Moreover the new virtual environment serves players living and acting with their commitment and this is the evident for most games success in any theme and ambition. In a software project development environment with a serious game it is necessary clarify exactly what are the characters, who are they, and what are the capabilities that they have. This clear sense is the primary factor for a player to commit himself in such an environment at first.

There is a degree of uncertainty around the terminology in being interactive with games. This shows a need to be explicit about exactly what is meant by the word interactive. Furthermore, this is more difficult for anyone at first to estimate the literal meaning of being interactive in the concept of serious game for ISO/IEC 12207. Plato in the Phaedrus famously complained that books were passive in the sense that you cannot get them to talk back to you in a real dialogue the way a person can in a face-to-face encounter [18]. Being interactive comes from interaction so any kind of game have to provide a dialogue to player. In fact if there is no such an interaction, how do players accomplish task? The instant feedback and continuous active reactions are the wanted factors to define the term being interactive. Undoubtedly, it is believed that every subject of in any science must be readable. This is the case over years and every person accomplishes many lectures and passes their exams via reading the related

resources. Texts and textbooks have to in a daily life, but in wider concepts such as engineering and medicine conceptualization needs to contain more practical program. To sustain ISO/IEC 12207 in a serious game environment the ambition of being interactive is possible with dynamic reactions between players and the game rather than text based learning.

There is an increasing desire about sandbox type games recently and this is another factor how a game should be. In games, sandboxes are safe environments where everyone can improve or destruct everything they create before without any stressful decision which is about the things can go wrong because it is isolated from outside world and therefore the effects of outside world can be minimized. This is an important factor for project managers to train software practitioners with respect to various bad experiences. Serious games have a potential to enhance this practice, because while playing a game usually players create different virtual careers based on their own interests and selections. They modify the ongoing progress according to their point of view to the concept, so any practitioner could be trained in a game setting, where they can learn from virtual situations where they make meaningful choices without negative outcomes.

5 Conclusions

This preliminary study investigates the need of a serious game from an industrial perspective. Initial results from the semi-structured interviews suggest that a serious game can improve the ability of learners of ISO/IEC 12207 standard. All five interviewees (n=5) agree that an interactive learning approach should make it easier to use and implement of the standard. Moreover, there is a consensus among the participants that such an approach will provide a better understanding of the documented concepts. One interviewee recommends that a serious game can use a simulated office environment and by creating a set of situations several complex concepts can be explained in an iterative way. One other recommends that a usability study should have to be conducted to address user perceptions of the product. In addition she suggested that the potenjtial for learning and engagement should somehow be measured. However, the major limitation of this study is the limited number of experts, therefore, caution must be applied. To evaluate the benefits of the proposed serious game, more research should be conducted on the functionality of the production process and ultimately with the end-product.

References

1. Tsui, F.F.: Essentials of software engineering. Jones & Bartlett Publishers (2014)
2. Futrell, R.T., Shafer, L.I., Shafer, D.F.: Quality software project management. Prentice Hall PTR (2001)
3. Jones, A.: Iso 12207 software life cycle processes fit for purpose? Software Quality Journal **5** (1996) 243–253
4. Singh, R.: International standard iso/iec 12207 software life cycle processes. Software Process Improvement and Practice **2** (1996) 35–50
5. Abt, C.: Serious games. University Press of Amer (1987)
6. Alvarez, J., Djaouti, D.: Introduction au serious game. Questions théoriques (2010)

7. ISO/IEC: Amendment to ISO/IEC 12207-2008 - Systems and software engineering Software life cycle processes. (2008)
8. Demirörs, O., Demirörs, E., Tarhan, A., Yildiz, A.: Tailoring iso/iec 12207 for instructional software development. In: Euromicro Conference. Volume 2., IEEE Computer Society (2000) 2300–2300
9. Khosrow-Pour, M.: Encyclopedia of information science and technology. Volume 1. IGI Global (2008)
10. Baker, A., Navarro, E.O., Van Der Hoek, A.: An experimental card game for teaching software engineering processes. Journal of Systems and Software **75** (2005) 3–16
11. Caulfield, C., Veal, D., Maj, S.P.: Teaching software engineering project management–a novel approach for software engineering programs. Modern Applied Science **5** (2011) p87
12. Navarro, E.O., van der Hoek, A.: Simse: An interactive simulation game for software engineering education. In: CATE. (2004) 12–17
13. Drappa, A., Ludewig, J.: Simulation in software engineering training. In: Proceedings of the 22nd international conference on Software engineering, ACM (2000) 199–208
14. von Wangenheim, C.G., Savi, R., Borgatto, A.F.: Deliver!–an educational game for teaching earned value management in computing courses. Information and Software Technology **54** (2012) 286–298
15. Calderón, A., Ruiz, M.: Prodec: a serious game for software project management training. In: ICSEA 2013, The Eighth International Conference on Software Engineering Advances. (2013) 565–570
16. Baker, A., Navarro, E.O., Van Der Hoek, A.: An experimental card game for teaching software engineering. In: Software Engineering Education and Training, 2003.(CSEE&T 2003). Proceedings. 16th Conference on, IEEE (2003) 216–223
17. Gee, J.P.: What video games have to teach us about learning and literacy. Computers in Entertainment (CIE) **1** (2003) 20–20
18. Gee, J.P.: What video games have to teach us about learning and literacy. Macmillan (2014)
19. Gibbs, W.W.: Software's chronic crisis. Scientific American **271** (1994) 72–81
20. Navarro, E.O., Baker, A., Van Der Hoek, A.: Teaching software engineering using simulation games. In: ICSIE04: Proceedings of the 2004 International Conference on Simulation in Education. (2004)