

A VIRTUAL REALITY-BASED TRAINING ENVIRONMENT DESIGNED FOR
HANDS-ON EXPERIENCE OF SOFTWARE DEVELOPMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ULAŞ GÜLEÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

NOVEMBER 2018

Approval of the thesis:

**A VIRTUAL REALITY-BASED TRAINING ENVIRONMENT DESIGNED FOR
HANDS-ON EXPERIENCE OF SOFTWARE DEVELOPMENT**

submitted by **ULAŞ GÜLEÇ** in partial fulfillment of the requirements for the degree
of **Doctor of Philosophy in Computer Engineering Department, Middle East
Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzin
Head of Department, **Computer Engineering**

Prof. Dr. Veysi İşler
Supervisor, **Computer Engineering, METU**

Assist. Prof. Dr. Murat Yılmaz
Co-supervisor, **Computer Engineering, Çankaya Uni.**

Examining Committee Members:

Prof. Dr. Ali Hikmet Doğru
Computer Engineering, METU

Prof. Dr. Veysi İşler
Computer Engineering, METU

Assoc. Prof. Dr. Aysu Betin Can
Information Systems, METU

Assoc. Prof. Dr. H. Hakan Maraş
Computer Engineering, Çankaya University

Assist. Prof. Dr. Ayça Tarhan
Computer Engineering, Hacettepe University

Date:

12.11.2018

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Ulaş Güleç

Signature :

ABSTRACT

A VIRTUAL REALITY-BASED TRAINING ENVIRONMENT DESIGNED FOR HANDS-ON EXPERIENCE OF SOFTWARE DEVELOPMENT

Güleç, Ulaş

Ph.D., Department of Computer Engineering

Supervisor : Prof. Dr. Veysi İşler

Co-Supervisor : Assist. Prof. Dr. Murat Yılmaz

November 2018, 126 pages

This thesis study proposes an environment that provides an interactive virtual reality experience for individuals about the tasks of software development process starting from requirement analysis through software testing. The environment transports participants to the virtual world of a software development organization where they experience development problems. In this environment, the participant takes on the role of a novice software developer being recruited into a virtual software development organization who should work alongside five virtual characters, played by artificial intelligence. This virtual world has a unique time-line where some virtual characters serve as the company guide. This exclusive viewpoint draws participants from the 2D separation of the classical experience and places them into the virtual world of software development. Therefore, participants have a chance to experience the problems occurred in software development process created for simulation. To understand the effectiveness of the system it was tested with 32 students who are studying at computer engineering department. According to the results obtained from the tests, the designed training platform is a useful tool that can be efficiently used in the training of individuals about the software development process.

Keywords: Software Development Process, Virtual Reality, 3D Environments, Software Development Life Cycle

ÖZ

YAZILIM GELİŞTİRMENİN UYGULAMA DENEYİMİ İÇİN TASARLANMIŞ SANAL GERÇEKLİK TABANLI EĞİTİM ORTAMI

Güleç, Ulaş

Doktora, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Veysi İşler

Ortak Tez Yöneticisi : Yrd. Doç. Dr. Murat Yılmaz

Kasım 2018 , 126 sayfa

Bu tez çalışması kişilere, gereksinim analizinden yazılım testine kadar devam eden süreçte yazılım geliştirme görevlerine dayanan bir interaktif sanal gerçeklik tecrübesi deneyimleyebilecekleri bir ortam sunar. Bu ortam katılımcıları, yazılım geliştirme sorunları yaşanan gerçek hayat projelerinin simülasyonunun gerçekleştiği sanal bir dünyaya taşır. Burada katılımcı, yapay zeka ile kontrol edilen beş karakterle birlikte çalışması gereken işe yeni başlamış bir yazılım geliştiricisi rolündedir. Böylelikle katılımcılar, gerçek olaylara dayanan sanal bir yazılım projesinde gezinirken, bu sanal karakterler gelişen olaylarla ilgili görüşlerini katılımcılara sunar. Bu sayede SDVRE, katılımcılara 2B yaşanan klasik tecrübenin daha ilerisinde bir tecrübe yaşatarak 3B sanal bir dünyada gerçek hayattakine benzer bir yazılım geliştirme dünyası sunar. Bu sayede katılımcılar, simülasyon için yaratılmış olayları deneyimleyebilme imkanına sahip olur. Geliştirilen ortamın başarısını anlamak için sistem, bilgisayar mühendisliği bölümünde öğrenim gören 32 öğrenci ile test edilmiştir. Testlerden elde edilen sonuçlara göre, tasarlanan eğitim platformu, bireylerin yazılım geliştirme süreci hakkında eğitilmesinde kullanılabilecek verimli bir araçtır.

Anahtar Kelimeler: Yazılım Geliştirme Süreci, Sanal Gerçeklik, 3B Ortamlar, Yazılım Geliştirme Yaşam Döngüsü

To My Family...

ACKNOWLEDGMENTS

Writing this part of my thesis shows that I'm nearing the end of this long marathon, and when I look back, I'm proud of completing this really difficult but also very beneficial and enjoyable process. At the same time, I feel very fortunate that I have been taking several different courses from valued professors, working with successful students and finding the opportunity to take part in valuable projects. This was really an unforgettable experience for me.

First and the foremost, I would like to thank my sincere gratitude to my thesis advisors, Prof. Dr. Veysi İŞLER and Dr. Instructor Murat YILMAZ since they provide me this opportunity and constantly mentor me through this process. They taught me how to become an advisor in future by publishing valuable publications, developing necessary projects, excellently communicating with me and giving the vital comments about my process. In addition to their academic skills, they are role models with their social lives. Being the student of them is always a great honour for me in my whole life.

I would like to also thank my Thesis Monitoring Committee members, Prof. Dr. Ali Hikmet DOĞRU and Assoc. Dr. Hadi Hakan MARAŞ, for their directive advices and excellent feedbacks. Due to these valuable comments, this study was formed as the current structure since their comments have provided me to look the problems in different perspectives.

My biggest chance in this life is my family. My father, Erdoğan GÜLEÇ, is a very special person for me. He is my master, my leader, my protector from challenging and undesirable situations, my resource of trust and pride... Shortly, he is my idol. My mother, Gülay GÜLEÇ, is the definition of smile for me. She is the most self-sacrificing person in all over the world. She is my best friend, my chef, my confidant... In short, she is the reason of my existence. My brother, Çağdaş GÜLEÇ, is the most valuable gift that I have taken in my whole life. He is the most enjoyable person in all over the world and the most important member of our "*band of brothers*". And finally, my sweetheart wife, Gamze GÜLEÇ, is the life companion of me. She is the owner of my heart, my supporter, my luck, the resource of my good feelings, the most special person in my life... Shortly, she is my everything. During this study, they always stood beside me in every challenging situations. I could not complete this study without their endless support and patience and could not pay their labours on me whatever I do. Hence, I would like to dedicate this study to my family.

TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vi
ACKNOWLEDGMENTS	viii
TABLE OF CONTENTS	ix
LIST OF TABLES	xii
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xv
CHAPTERS	
1 INTRODUCTION	1
2 BACKGROUND AND RELATED WORKS	5
2.1 Definitions of Software, Software Development and Software Engineering	5
2.2 Software Development Methodologies	9
2.3 Importance of Using Virtual Reality Applications in Education	15
2.4 Tools Used in Software Engineering Training and Simulations	17
2.5 Summary	19
3 PROPOSED VR TRAINING ENVIRONMENT	23

3.1	Scenario Generator	23
3.2	Virtual Office Environment	31
4	TEST AND EVALUATION METHODOLOGY	47
4.1	Qualitative Research Methodology	47
4.2	Quantitative Research Methodology	48
4.3	Mixed Research Methodology	49
4.4	Overall Research Design	51
4.5	Mechanism of the System	51
4.6	Participants	56
4.7	Threats to Validity	58
5	ANALYSIS AND TEST RESULTS	61
5.1	Pre-Test	62
5.2	Post-Test	65
5.3	Comparison Between Pre-Test and Post-Test Results	68
5.4	PQ and ITQ	73
5.5	Validation Interviews	75
6	CONCLUSIONS AND FUTURE WORKS	79
6.1	Discussion	79
6.2	Validation of the Proposed Training Environment	81
6.3	Limitations	82
6.4	Revisiting the Research Questions	83

6.5	Conclusion	85
6.6	Future Work	86
REFERENCES		87
APPENDICES		
A	TRAINING PROGRAM SCENARIOS	97
B	PRE-TEST SCENARIO	103
C	POST-TEST SCENARIO	105
D	T-TABLE	109
E	PRESENCE QUESTIONNAIRE	111
F	IMMERSIVE TENDENCIES QUESTIONNAIRE	117
G	GRADING POLICY	121
CURRICULUM VITAE		123

LIST OF TABLES

TABLES

Table 2.1	Comparison Table with the Existing Studies in the Literature	21
Table 4.1	Threats to Validity for Empirical Research in Software Engineering .	59
Table 5.1	Pre-Test Results of the Members in the Experimental Group	63
Table 5.2	Average Scores of Both Groups for All Parts in Pre-Test	63
Table 5.3	Pre-Test Results of the Members in the Control Group	64
Table 5.4	Post-Test Results of the Members in the Experimental Group	67
Table 5.5	Average Scores of Both Groups for All Parts in Post-Test	67
Table 5.6	Post-Test Results of the Members in the Control Group	68
Table 5.7	The Differences between Pre-Test and Post-Test Results	72
Table 5.8	PQ Scores of the Participants over 100	76

LIST OF FIGURES

FIGURES

Figure 3.1	Use Case Diagram of System Administrator	24
Figure 3.2	Project Definition Segment	25
Figure 3.3	Requirement Analysis Segment	26
Figure 3.4	Design Segment	27
Figure 3.5	Implementation Segment	28
Figure 3.6	Input Screen	29
Figure 3.7	Test Segment	30
Figure 3.8	Maintenance Segment	31
Figure 3.9	Empty Content of the XML File	32
Figure 3.10	Use Case Diagram of Participant	33
Figure 3.11	Opening Screen of Virtual Office Environment	34
Figure 3.12	Starting Screen of Simulation	34
Figure 3.13	Team Member in Requirement Analysis Phase	35
Figure 3.14	First Dialogue Box of NPC in the Requirement Analysis Phase	35
Figure 3.15	Project Description Document	36
Figure 3.16	Opening Project Description Document	36
Figure 3.17	Reading Project Description Document	37
Figure 3.18	Finishing Project Description Document	37
Figure 3.19	Requirement Analysis Phase in Virtual Office	38
Figure 3.20	First Dialogue Box of NPC in the Design Phase	39

Figure 3.21 Design Phase in Virtual Office	39
Figure 3.22 First Dialogue Box of NPC in the Implementation Phase	40
Figure 3.23 Implementation Phase in Virtual Office	41
Figure 3.24 First Dialogue Box of NPC in the Test Phase	42
Figure 3.25 Test Phase in Virtual Office	43
Figure 3.26 First Dialogue Box of NPC in the Maintenance Phase	44
Figure 3.27 Maintenance Phase in Virtual Office	44
Figure 4.1 Overall Research Design	50
Figure 4.2 Work Flow of the System	56
Figure 4.3 Time Line of the Study	57
Figure 5.1 Successful Students in the Experimental Group in Pre-Test	65
Figure 5.2 Successful Students in the Control Group in Pre-Test	66
Figure 5.3 Successful Students in the Experimental Group in Post-Test	69
Figure 5.4 Successful Students in the Control Group in Post-Test	70
Figure 5.5 The Difference Between Pre-Test and Post-Test Results of Both Groups	71
Figure 5.6 ITQ Scores of the Participants	74
Figure 5.7 PQ Scores of the Participants	75

LIST OF ABBREVIATIONS

VR	Virtual Reality
SDLC	Software Development Life Cycle
PQ	Presence Questionnaire
ITQ	Immersive Tendencies Questionnaire
NPC	Non Player Character
XML	Extensible Markup Language
HTML	HyperText Markup Language
METU	Middle East Technical University
MUDEK	Association for Evaluation and Accreditation of Engineering Programs
SLR	Systematic Literature Review
NIST	National Institute of Standards and Technology

CHAPTER 1

INTRODUCTION

Software is a set of instructions designed to fulfil specific requirements of a user [1]. However, defining its requirements and implementing of software is a very complicated process [2]. Consequently, many software projects fail to deliver acceptable outcomes due to dynamic changes in requirements [3]. A reason for this is that novice software developers cannot show a variety of skills especially during their on-boarding process [4]. To bridge this gap, it is essential to provide hands-on experience in a virtual environment for novice practitioners where they can practice the real life problems without having actual risks.

Virtual Reality (VR) is a technology that generates 3D environments in which users can highly interact with various input and output equipments [5]. The aim of this technology is to immerse a user in a computer-generated environment and produce the feeling that they are physically in this environment [6]. Hence, it provides a beneficial and immersive training environment that includes real-life conditions for individuals [7]. Virtual environments are used to increase the level of knowledge and experience of people working in many different fields, and to experience the problems and events that they may encounter in real life [8]. Due to this property, in the last two decades, VR becomes a very popular area [9] that provides opportunities in different domains [10].

There are several different VR applications in different domains including education [11], sports [12], psychology [13] and medicine [14]. With the progress of technology in VR over time, both the number of VR projects, the number of companies that deal with VR projects and the number of domains where VR projects are completed are also increasing rapidly [15]. Many businesses are investing in this area to

take advantage of the benefits of VR technology [16].

These studies show that VR is an effective training tool that can be used to increase individuals' level of knowledge and experience in different working spaces. Hence, the aim of this study is to increase the experience and knowledge levels of novice software engineers about the tasks related to software development process by designing a 3D virtual office environment. By using the potentials of a virtual office, a task-based software engineering training can be conducted independently from customers and instructors outside traditional office environments. Ultimately, an evidence-based training in a virtual office environment promotes self-directed learning with case-based assessments which is likely to improve performance and skills of students. Therefore, the proposed VR environment is an opportunity for participants to practice software development tasks repeatedly to improve their practical skills before gaining real-life experiences.

The research questions of this thesis are determined to guide the research as follows:

- **RQ 1:** *Can the proposed training environment increase performance of students on software engineering tasks (e.g. requirement capturing, coding, testing, etc.)?*
- **RQ 2:** *Can the proposed training environment motivate the students for exercising the tasks related to the each phase of SDLC?*

The overall structure of this thesis takes the form of six chapters; including this introduction chapter. The remaining parts of this thesis are structured as follows:

Chapter 2 firstly creates the background of the research by giving the definitions of software and software development, illustrating the importance of software engineering, software development processes and methods, and criticizing why there exist several different software development methodologies. Then, this chapter also explains the importance of VR applications in the education domain and details the studies which were developed to train the individuals about the software development process.

Chapter 3 details the system description by presenting the interfaces, the users and

their functionalities. In addition, this chapter also mentions about the technologies used for the applications in the scope of this research.

Chapter 4 begins by describing the research techniques in the literature. After that, the research technique used in this study is demonstrated with the reasons of selection. In addition, this chapter also includes the mechanism of the proposed system, information about the participants of the study and the threats which may affect the results of the study in a negative way.

Chapter 5 mentions how the developed system was tested by the participants and how the results obtained from the tests were analyzed by using statistical tests. The visual elements are also used to show the results more clearly. In addition, the opinions of the participants about the system are also illustrated in this chapter.

Chapter 6 gives a summary of the overall study. Then, it tells the conclusions and future works of the study. Finally, the dissertation is concluded by explaining the future works.

CHAPTER 2

BACKGROUND AND RELATED WORKS

The literature review has been presented in two different parts. The aim of the first part is to create the background information of the study since this study aims to produce a virtual training environment to teach the basics of the software development process to novice software engineers. To accomplish this aim, this part begins by explaining what software, software development and the phases of software development are, and it discusses the importance of software engineering, software development processes and methods, and why many software development methods are needed. Then, this part also compares the existing software development methodologies in order to show the reasons why many software projects fail although there exist several different software development methodologies.

After determining the missing points of the existing software development methodologies, the second part of the literature review begins to find the solution that improves the effects of the software development process on projects. In this part, the importance of using VR applications for training purpose is indicated by giving example studies in the literature. Furthermore, tools used for software engineering training are explained in detail by comparing with each other. Based on the comparison, the missing points of the existing tools in the literature are listed. Finally, a brief summary of this chapter concludes the literature review of the study.

2.1 Definitions of Software, Software Development and Software Engineering

Software is an expandable, functional and programmable product that can run on a computer [17]. Although the definition of software appears to be simple, the structure

of software including analysis, design, development and testing steps is very complicated [18]. Hence, software is a product obtained after several long processes. At this stage, the definition of software development becomes more meaningful since software does not have a simple structure.

In 1971, Weinberg [19] described software development as a human activity that consists of design, creation, implementation and maintenance phases. It can be also defined as a combination of several significant phases that lead to transforming customer requirements to a software product [20]. These phases are:

- **Requirement Analysis:** It is a phase that describes for what reason the system is going to be created [21]. In addition, this phase also converts the needs of customers to system specifications [22]. Pereira and Soares [23] point out the importance of the requirement analysis phase in their study. They consider that if the requirements of a system cannot be accurately determined, it causes to develop useless products. In support, based on the CHAOS report released by the Standish Group in 2015 [24], the requirement analysis has become more critical nowadays since the failure rates in IT projects are very high due to determining the wrong requirements. This issue is essential for software companies because, if an error in a system is detected at the last stages of software development life cycle (SDLC), it is more costly for companies to repair it at the final stages than to repair it at the beginning stages. Therefore, the requirement analysis plays a key role in the development of software to develop more accurate and less costly products since it is the first phase of a SDLC.
- **Design:** This phase contains a set of activities that should be accomplished to develop a system [25]. It is also a decision-making phase for selecting the technique to complete the modules in the system [26]. This phase plays a key role in SDLC. According to Karimi [27], the number of logical errors between 36% and 74% is detected in the design phase. Although each phase is significant to develop the projects properly, the design phase is more important than the other phases since it affects the future of the systems [28]. If a system is designed well, it fulfils the future expectations from the system.
- **Implementation/Coding:** This phase aims to classify and construct the cod-

ing and algorithm part of the system by taking into account of constraints and demands obtained from the requirement analysis phase [29]. In addition, the programming parts of the systems are also conducted in this phase based on the tasks determined in the design phase [30]. Due to these properties of the implementation phase, it directly affects the quality of the end product.

- **Testing:** It is a phase of software development considered as a frequently used verification technique that finds errors in a system [31]. In addition to this definition, testing is a combination of both verification and validation methods that consist of test plans, designs and procedures [32]. This phase is a significant step in SDLC since it improves the quality of the product by finding and fixing the missing points of the system [33]. In support, Ralph and Kelly [34] administrated an interview with 191 professionals in software engineering in order to understand the factors that affect the success of the software product. According to results of the interview, participants feel safer after the product passes the testing phase since they trust that all errors of the system are straightened in testing stage.
- **Maintenance:** Maintenance phase provides changes, corrections and improvements to the system after it is delivered to customers [35]. There are several different types of improvements and changes such as corrective, adaptive and perfective in the maintenance phase to make the system work properly [36, 37]. Schneidewind [38] points out that keeping the system appropriately running after delivering to the customer is an important and critical stage for the companies since it reflects the success of the software produced by applying whole stages of software development. If the customer cannot be satisfied in this phase, the entire work becomes useless. In support, April and Abran [39] agree with the importance of the maintenance phase since customer expectations are met at this stage.

These studies in the literature show that each phase of software development plays a critical role in developing useful and successful products. Hence, software development is an important issue such that its phases should be thought of and carried out as engineering concepts in order to produce measurable, evaluable, reusable and

replicable products which have concrete outputs [40]. Although it is an essential issue, most software projects fail [3]. According to the National Institute of Standards and Technology (NIST), the annual loss of unsuccessful software projects to the US economy was about \$ 59 Billion in 2002 [41]. When today's data is analysed, the total worldwide financial loss of failed software in 2017 was around 1.7 trillion dollars [42]. These numerical values illustrate that most of the software companies waste both time and money for developing completed software. In order to avoid such undesirable circumstances, the software development process should be designed based on an engineering point of view since the end product requires systematic development.

The systematic development within the engineering perspective creates the "*software engineering*" concept. Software engineering is a set of engineering activities that manages the phases of SDLC in order to produce high quality software products [43]. It is a discipline that tries to solve problems occurred in software development by organizing the phases of SDLC [44]. Therefore, it improves the performance of software within a systematic scheduling to achieve software quality [45]. Software quality measures both the software development process and the software product based on a number of factors [46]. The studies [47, 48, 49, 50] in the literature analyse and list these factors as: *Correctness, Reliability, Efficiency, Integrity, Usability, Maintainability, Flexibility, Testability, Portability, Reusability, Functionality, Operability, Compatibility, Modifiability*

The studies in the literature illustrate that software should be developed within a methodology that is systematically designed to meet the software quality criteria. However, the methodology can differ based on the properties of software and company resources. Therefore, companies need to manage the phases of SDLC according to their resources and the requirements of their customers in order to provide those software quality criteria. The next section is going to explain the different software development methodologies in detail.

2.2 Software Development Methodologies

A software development methodology is a method that applies the phases of SDLC in different way based on the organizations' resources, and the customers' requirements. Hence, there are several different software development methodologies which help organizations to produce cost-effective and successful projects. The commonly used methodologies are:

- **Waterfall Development:** Royce [51] developed this model based on his experiences on the development of software projects. This model suggests that each activity in the process of software development should be planned and scheduled before performing on them. It has a specific order to implement the phases of SDLC. The next process cannot start until the current phase is completed. The outputs or findings of a phase is going to be the inputs of the next phase. It is suitable when the requirements are well defined, the scope of the project is not complex and the requirements do not need to be changed rapidly.
- **Prototype Development:** Sommerville [52] describes this approach as a prototyped version of a software product is produced to show a set of features of the software product to the user. The goal is to acquire a certain degree of understanding about design options and the problems occurring in the development of a product. Since users can utilize the prototype and understand its deficiencies more easily, software developers have an opportunity to design and implement the product based on users' feedback. Thus, the customers have a chance to test the prototype for better improvement.
- **Iterative and Incremental Development:** A detailed study of iterative and incremental development by Victor [53] reports that projects are developed iteratively and incrementally in this type of software development model. "*Iteratively*" and "*incrementally*" mean that the phases of SDLC are in a loop until the user accepts the project. The products that are created after the implementation phase are shown to users to obtain their opinions about the system. Then, it is updated based on these opinions. This process continues until all users requirements are achieved.

- **Spiral Development:** Boehm [54] defines this model as an evolutionary software model that uses risk assessment techniques for software components. This model has loops which visit the software development phases. Each loop starts with an objective definition in terms of performance and functionality and continues to plan how the objective can be achieved. The following iteration attempts to solve the risks gathered from the previous step.
- **Rapid Application Development:** is an adaptive approach that frequently uses risk mitigation techniques such as prototyping [55]. In addition, it also increases the productivity by focusing on changes reducing the total delivery time and verifying the solution that works for end users since they are a part of the development team during the software development process [56]. Therefore, agile methods are used for rapid application development where developers and customers collaborate beginning from the design phase of a software project.
- **V-Shaped Model:** This is a type of waterfall development model that differs in verification and validation techniques [57]. Testing of the product is completed with the phases of SDLC simultaneously. This means that each phase has a testing operation.
- **Agile Development:** The goal is to minimize the total cost, to produce high quality products and to decrease total project time by adapting any updates and changes iteratively [58]. Although there are different methodologies that can be considered as Agile Software Development such as Scrum [59], Kanban [60] and Extreme Programming [61], the general idea is to meet user requirements and expectations whenever required or desired. In this method, the collaboration between team members, the communication between organization members and customers and testing are very significant concepts rather than classical phases of software development. The short-term objectives are determined rather than the long-term objectives.

There exist many different software development methodologies in the literature since none of them can be applicable to all types of software projects [62]. This means that the software development methodologies have both advantages and disadvantages since their components differ from each other [63, 64].

There are various studies in the literature which compare the existing software development methodologies. In the first study, Saxena and Upadhyay [65] compared waterfall and prototype development methodologies. In this study, the total development time, expenditure to develop the project and user satisfaction were designated as factors that determine whether a project is successful. In the light of these factors, both methods have a number of advantages and disadvantages similarly to the other software development models. The advantages of the waterfall development model can be listed as having well defined stages, being easy to understand and easy to use, and having a simple implementation process. In addition to these advantages, this model requires minimum resources to complete a project in comparison to the other models. On the other hand, this model also has a number of disadvantages, the first of which is that the system cannot be used for requirements not occurring at the beginning of the development stages. Another disadvantage is that this model does not allow users to be involved in the development process of a system from the beginning. In the prototype development model, users are included in the project development process from the beginning of the project. This means that the users can continuously give feedbacks about the project. Therefore, errors may be easily detected. In summary, the waterfall development model is suitable for well-defined projects which require minimal updates when the prototype development model should be used to develop more complex systems due to its dynamic and flexible structure.

Drury-Grogan and Kennedy [66] compare the waterfall and agile development models in their study. According to this study, the waterfall model enables project members to focus on the project tasks, which occur in a linear sequence. A new task starts after an existing task is completed. Thus, the aim of project members is to complete the tasks assigned to them within the time schedule. For this reason, the project can be easily managed. Although this can be seen as an advantage, it also brings some disadvantages. The first disadvantage is that the different teams are generally assigned to each phase of SDLC in the waterfall development model. This situation causes a continuous lack of sharing of knowledge and information among team members due to every team member focusing on his own respective task. Thus, there is no strong interaction between team members. Another disadvantage is that the project is tested in the last stages of SDLC. This is a significant deficiency of this method since al-

most every project requires reworking. In addition, if a company wishes to rework a project to rectify problems, the operation would be costly. The agile development model fills these lacunae in the waterfall development model. It allows for an iterative development of the project since the process of this method is dynamic. Therefore, any new demands from customers may be more easily performed. Short-term tasks are assigned to small teams which have high-level interactions between members.

A significant analysis and discussion on comparing the software development methods was presented by Moniruzzaman and Hossain [67]. This study illustrates the deficiencies of existing traditional software development methods by comparing them with the agile development model. According to this study, traditional software development models feature the importance of planning instead of adapting customer requirements, which seem to occur after a project starts. This situation makes the projects more manageable; however, the studies in the literature [68, 69] indicate that easy management of a project is not meaningful if it does not satisfy customer expectations. Traditional methods attempt to guarantee to finish a product based on the requirements determined at the beginning of the project, whereas the aim of the agile development model is to integrate the customers' new requirements into the system rapidly. Consequently, more qualified products can be produced at the end of the development process. Additionally, the agile development model brings other advantages. The first additional advantage is that it divides the objectives into the small tasks. Thus, the time needed to complete a task is short. This provides rapid development in addition to a rapid testing environment. Another advantage is that customers are included in many aspects and steps of the project development process. This produces opportunities for customers and developers since customers can see any missing or undesirable parts of the projects, thereby enabling developers to deal immediately with any deficiencies in a project. The last advantage is that this model reduces the total reworking cost since the errors in the system are detected in the early stages of the software development process.

Another comparative study completed by Mishra and Dubey [70] illustrates the similarities and differences between the waterfall, V-Shaped, spiral and rapid application development methods by explaining both the advantages and disadvantages of each software development model. According to this study, the waterfall develop-

ment model cannot be applicable to a project in which requirements are dynamically changed or updated. The other models were discovered to fill this gap. For example, in the spiral development model, the developers start with a small set of requirements at the beginning of the project and, if necessary, add new requirements in the following stages. However, this dynamism causes delays in the delivery of projects if the manpower is not adequate to complete any complex systems. The V-Shaped model is an improved version of the waterfall development model in terms of the testing phase. Although testing occurs after each phase, the requirements cannot be easily included in the project development phase. The rapid application development model can have projects finish in a short-time period if the requirements are well-defined; however, the error rate may increase in the end product.

These studies in the literature illustrate that the existing software development methodologies cannot satisfy all non-functional requirements in which the end product should contain. This means that each of them has some disadvantages. Because of this reason, they may provide the requirements of the applications in a limited way. Hence, they need to be improved in order to enhance the quality of the projects. In accordance with this purpose, the literature has been scanned to find any studies that have been carried out to determine the factors that make the existing software development methodologies ineffective.

There are a number of studies [71, 72, 73] in the literature showing that the software development methodologies are generally hard to be trained for software development teams. Especially, the agile development needs to be combined with other techniques such as "*user centered design*" [74] or "*model driven development*" [75] in order to be successfully deal with development challenges. The reason for these challenges is that the software development methodologies are hard to understand without real-life practices. Although software development methods are well-understood by experienced managers, they cannot easily transfer this tacit knowledge since novice practitioners do not fully comprehend the technical details of software development [76]. Therefore, members in a software development team have to play different roles during the software development project since they try to close each other's deficiencies to bring out a successful software product. For this reason, roles in software team members become exchangeable since the ultimate goal is to deploy the software

product [77]. However, if an introvert person is involved in the project, this situation directly affects the quality of the software development process since the communication between the team members cannot be performed properly [78]. In particular, in some software projects, team members should self organize to successfully complete the software projects when their roles are not well-defined [79]. However, this situation can also directly affect the success of the project negatively, due to the fact that the workload within the team is not equal and that the self-organized people may not share information.

When the above paragraph is summarized to illustrate the main drawbacks of the existing software development models, the improvement of the following substances is important to enhance the impact of the models on the projects:

- To increase the knowledge and experience levels of the project members regarding the applied process.
- To better identify the roles of individuals in the project team.
- To give more detailed information team members about their duties in the project.
- To accelerate the adaptation process of new participants to the project.
- To choose more suitable people for the project.

As a result of this section, software development is a discipline that aims to standardize the tasks of software production. There are many different methodologies proposed for software development. These methodologies, however, possess the same sequential stages (i.e. requirements gathering, design, implementation, testing, and integration) for both traditional and agile approaches. Each software project includes these steps, without being dependent on the software development methodology. In other words, the tasks that arise in the phases of software development are not dependent on the software development process. Consequently, a software practitioner should know all these phases from both theoretical and practical point of view. However, apart from the methodology chosen for production, novice software practitioners should gain hands-on experience for all stages of software development.

Therefore, this study proposes a process-agnostic training environment that encapsulates experience-based training for novice software engineers. The next section will explain the benefits of VR applications when they are used to improve individuals' experience and knowledge levels.

2.3 Importance of Using Virtual Reality Applications in Education

In this study, a VR environment is created to provide a workspace where the participants can increase their both knowledge and experience levels about the basis of software development process. Hence, it is important to figure out the importance of the VR applications developed for training purpose. To achieve this, the literature review was conducted to find similar studies that aim to train individuals in different domains by using VR technology.

Fang and Teizer [80] developed a virtual environment to train both crane operators and ground personnel. The aim of this study is to increase the collaboration between crane operators and ground personnel by reducing the mistakes during a construction. The cranes, construction materials and buildings used in real environment during construction were modelled and integrated to the virtual environment. The training program is scenario-based and it is expected that people perform certain tasks in each scenario. The designed virtual environment is a multi-user environment and offers instant communication between users. In this way, the users can prepare themselves against the problems they may encounter in real life by experiencing these problems with the communication created in the virtual environment. This environment has been tested by construction graduate students. According to the results of this test, the practice in the virtual environment improves the skills of the participants. In another study by Bliss et al. [81], a virtual environment was designed to increase the navigation capabilities of firemen. In order to understand the effect of the virtual environment, participants divided into three different groups as blueprint, VR and no training. According to the results of this training program, the VR is an effective tool that can be used in education domain. Seymour et al. [82] conducted a study to illustrate the effect of VR when it is used as a training tool. The aim of this study is to improve the abilities of the medical students by creating a virtual environment. A group

of student ($N = 16$) was equally divided into two different groups, VR and non-VR. At the beginning of the study, a pre-test was administered to evaluate the knowledge levels of the students before training program. After the training program, a post-test was also administered to illustrate the development of the students. According to the results obtained from these tests, students who studied with VR developed themselves more than the students who worked with traditional methods. In the same vein, Kandalaft et al. [83] has a study to improve the social abilities, attention and functioning of young adults who are diagnosed with autism at high levels. In Second Life, an area including offices, buildings, shopping and coffee shops, restaurants, schools and parks was reserved for only this training program. This virtual environment was used by eight participants. The avatars in the virtual environment were modelled considering the appearance of the participants in order to increase the reality. The designed training program was scenario-based and the scenarios such as meeting new people, ordering food, conducting relationships with friends, making financial decisions were designed to increase participants' social skills. A pre-test and a post-test were administered to the participants in order to observe the development of the participants' skills. The results of these tests illustrate that virtual environments can be used as an important training tool which enhances individuals' abilities. Another study completed by Elledge et al. [84] contains a virtual environment to increase the abilities of medical students in treatment methods for maxillofacial emergencies. Throughout their education life, there is no environment in which students can gain experience in this regard. The students involved in this study used the virtual environment to fulfil for these shortcomings within a month of training. The virtual environment involves scenarios which consist of ten intervention directives frequently occurring in real life. A pre-test and a post-test consisting of twenty multiple choice questions were administered to determine the improvement on the skills of the students. The qualitative results obtained from the tests illustrate that there is a significant difference between the students' post test and pre-test scores in positive manner. Therefore, a virtual environment is an effective tool to train the individuals.

These studies in the literature show that VR is an appropriate tool used in the training of individuals in different working areas. The next section will mention about the training tools, including VR, developed for software engineering training.

2.4 Tools Used in Software Engineering Training and Simulations

There are several different studies in the literature that train individuals about software engineering concepts. While most of these studies are 2D or board/card games, and involve many features of serious games, 3D virtual environments are also designed to enable individuals to experience a more active learning process in some other studies.

An example study in this area was carried out by Baker et al. [85]. In this study, a card game was developed with the aim of increasing the level of experience of people about software development processes without involving real life risks. The game is multi-player and each player is given tasks similar to tasks in real life projects. The players must perform these tasks as soon as possible to meet the customer's requests in accordance with their budget. This game includes phases of the Waterfall Development Model as a software development methodology and the players give decisions related to project management as the project team leader. According to the results of the study, this card game is a beneficial training tool that can be used to support traditional methods since it creates a competitive atmosphere among the players which encourages people to play.

Bollin et al. [86] developed a simulation framework to experience different software development methodologies for the participants. This framework is a computer application that contains elements similar to flowchart components. Hence, the participants can develop the projects or decide on a topic by connecting these elements to each other. In this way, the participants have a chance to improve their experience levels on project development without living real experiences.

The idea of using games in software engineering education is also supported by Hailey et al. [87] who produced a game that helps individuals about how the requirements of a project should be gathered and analyzed. In this game, the players should manage and complete the software projects with different roles such as team leader, system analyst, designer or project manager. The game was supported by non-player characters who direct the players about their tasks. This game was tested with 92 students to figure out whether the game is an effective tool which supports individuals to increase their requirement analysis skills. Both pre and post test was administered

to see the difference between the participants' knowledge levels before and after the training program. According to the results obtained from this study, this game is very useful to increase the knowledge levels of the participants about software engineering since it attracts the attention of people.

Similarly, Rusu et al. [88] designed an interactive game that teaches the content of the maintenance phase of SDLC to the students. This content was divided into four categories: perfective, adaptive, preventive and corrective. The game dynamics of this game are very similar to game dynamics of classical tower-defence game. The bugs and errors in the system were represented as enemies and the projects were represented as towers. The players try to protect their towers from the enemies by using the available protection methods in the game. 18 students were selected to test the efficiency of the game. The results of this study indicate that the games can be an alternative learning technique which can be effectively used in the training of software engineering topics.

When the 3D applications developed in this area are to be examined, Aydan et al. [89] designed a serious game called "*Floors*" to teach the basics of ISO/IEC 12207:1995 in an enjoyable manner. For this game, a 3D virtual office environment supported by non-player characters was created to further impress the participants. This virtual environment was tested with 40 students. The students were divided into 2 equal groups as one control group and one experimental group. The students in the control group used traditional methods to learn the fundamentals of ISO/IEC 12207:1995 while the students in the experimental group used the designed game. The findings of this study illustrated that the level of knowledge about the ISO/IEC 12207:1995 of the participants in the experimental group improved considerably.

In the same vein, Ye et al. [90] used Second Life as a training tool where the faculty members have the ability to answer their students' questions about software engineering. The participants have avatars to represent themselves in the virtual environment. A virtual classroom similar to real classroom was designed in Second Life to provide communication between students and lecturers. This environment was tested with a group of students ($N = 25$). After the study, a survey was administered with the participants to obtain their opinions about the impact of the system on the users. The results

indicate that a virtual teaching strategy is advantageous in enhancing the knowledge levels of students about software engineering field.

Another similar study was performed by Rodriguez et al. [91]. In this study, a 3D virtual seminar room was designed to teach the practices of Scrum to software engineering students without having time and facility limitations. This virtual environment consists of virtual elements similar to real elements in a meeting room such as blackboard, charts and calendar. In this environment, the participants were allowed to move the virtual objects in order to increase the reality of the system. An example project was integrated into the system to demonstrate the flow of the Scrum methodology. The virtual environment was tested with 45 undergraduate students to measure the effect of the environment on the individuals. According to the results obtained from the tests, this virtual environment is a beneficial tool that can be used to explain the flow of the Scrum method to the students.

Parsons and Stockdale [92] conducted a study in which a virtual world was designed by using Open Wonderland to train the participants about the properties of agile software development methodology such as user stories, features in user stories and team collaboration. In this study, it is expected that the knowledge levels of participants about agile software process is increased by development of a simple project that is not related to software engineering. The findings of this study clarify that although the developed tool requires significant improvements such as reality of the project, whole project development process and more realistic project, it can be used for training individuals about software engineering processes.

2.5 Summary

According to the results obtained from the literature review, it is necessary to train software practitioners, especially novice ones, about the tasks related to software development process in order to increase the quality and success of the software projects. Although there exist a number of applications for this purpose in the literature, there are still some drawbacks of these systems that should be completed. Table 2.1 shows these deficiencies on a study basis.

When these studies are analyzed in detail, the main weaknesses can be summarized as:

- **Low/Limited Increased Reality:** The most important feature that affects the learning progress of individuals is that the training environments should be similar to real environments [93]. Although it is an essential issue for training tools, the card and 2D games cannot fully provide this feature since these platforms do not have enough hardware and software functions to detach the players from the real environment. Hence, this situation decreases the sense of presence which is one of the most significant factor that shows the success of the designed environment [94]. 3D environments can provide the sense of presence, however, the existing studies did not have enough user interaction functions. Hence, they did not measure their participants' level of sense of presence.
- **Missing Whole Project Development Processes:** As mentioned in the previous sections, the software is a product obtained after several different phases. Although the development of a software is a long process, the existing studies aim to increase the participants' level of knowledge about the specific phases. None of them can inform the users about the whole project development.
- **Limited Number of Stories:** The existing studies only focused on one project as an example test project. In addition, some of these projects were not related to the software engineering and the other ones were not large scale projects like in real life. Therefore, this case can be shown as an important drawback of the existing systems, because, analyzing only one project limits the development of individuals since the requirements of real life projects are very different from each other. Hence, the participants cannot face different problems occurring in the different types of the projects.

To sum up, this study purposes an interactive 3D virtual environment to help participants gain experience based on the tasks of SDLC by enhancing the drawbacks of the existing studies in the literature. Due to this environment, the participants have a chance to face development problems and conflicting situations with some of distinctive virtual personality characters without actual risks.

Table 2.1: Comparison Table with the Existing Studies in the Literature

Authors	Type of the Tool	High Reality	Whole Project Development Process	Different Stories
Baker et al. [85]	Card Game	✗	✓	✓
Bollin et al. [86]	Computer Application	✗	✓	✓
Hailey et al. [87]	Digital 2D Game	✗	✗	✗
Rusu et al. [88]	Digital 2D Game	✗	✗	✗
Aydan et al. [89]	3D Virtual Environment	✗	✗	✗
Ye et al. [90]	3D Virtual Environment	✓	✗	✗
Rodriguez et al. [91]	3D Virtual Environment	✗	✓	✗
Parsons and Stockdale [92]	3D Virtual Environment	✗	✗	✗

CHAPTER 3

PROPOSED VR TRAINING ENVIRONMENT

As was mentioned in the previous chapters, this study aims to train participants about the tasks occurred during software development in a virtual environment which is similar to real environment. Additionally, the proposed system also aims to eliminate the shortcomings of the similar studies in the literature (see Chapter 2). In this context, two complementary applications were developed, one as a desktop application to create software project scenarios and the other one is a virtual environment in which the participants can live the software development process of different software projects. For this reason, the following section introduces the details of the scenario generator. After that, the details of the virtual environment in which the participants can experience the development process of a software project scenario generated by the scenario generator program is explained.

3.1 Scenario Generator

In the literature, there are some studies developed for a similar purpose as this study. Although there exist similar studies in the literature, they have some drawbacks when they try to be used for training purpose (see Chapter 2).

"*Scenario Generator*" module was developed in order to make up for the lack of a limited number of stories. This module enables the system administrator, who may be an authorized person in the company, to enter the information, tasks and items that are related to the whole project development processes as shown in the use-case diagram of the system administrator (see Figure 3.1). Thus, this module has the ability to produce several different project scenarios, which include the whole project devel-

opment process, so that the second and third missing points of the existing studies in the literature can be eliminated due to these properties of the module.

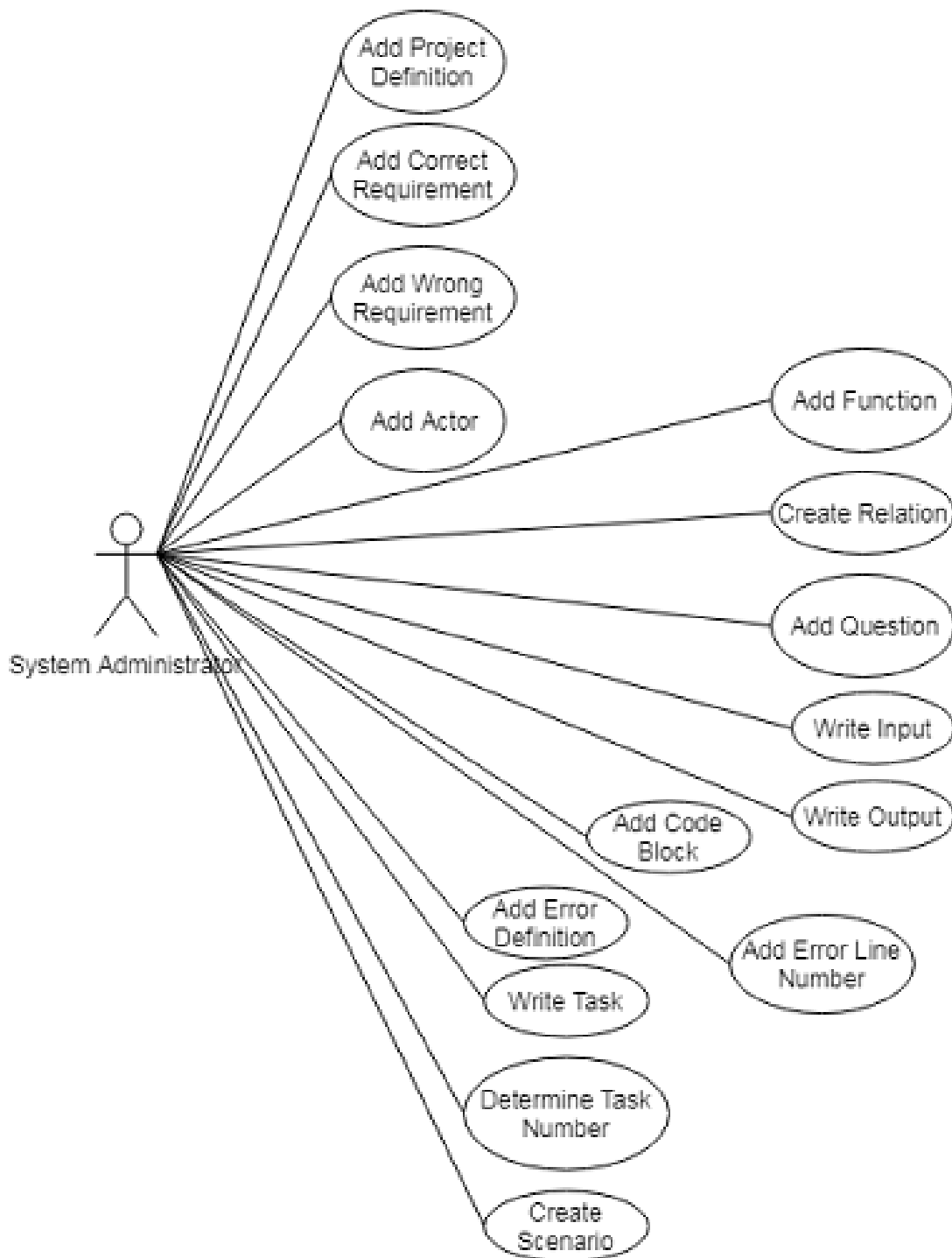


Figure 3.1: Use Case Diagram of System Administrator

When a system administrator runs the module, the main page appears on the screen as shown in Figure 3.2.

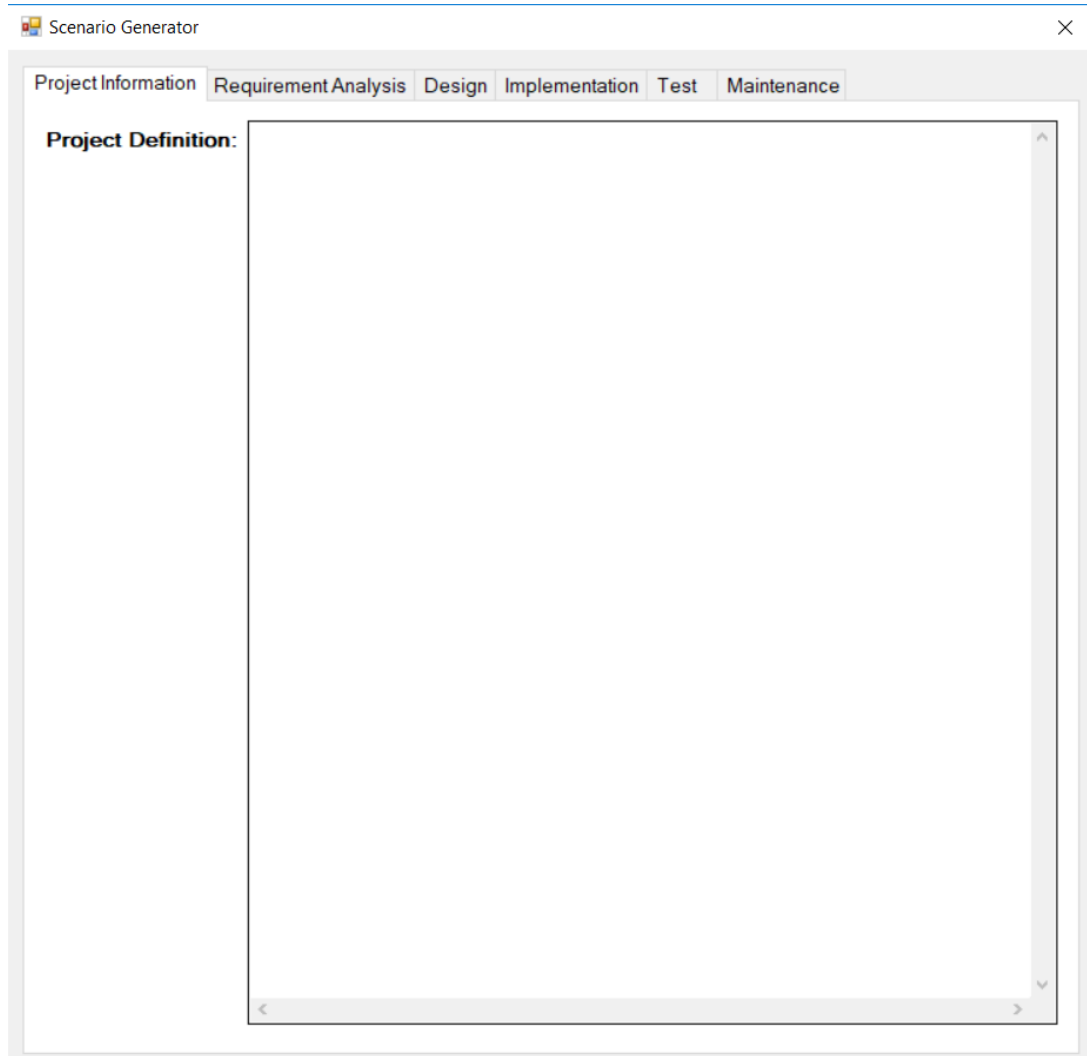


Figure 3.2: Project Definition Segment

Since the aim of this study is to experience participants the entire software development process, it is necessary to generate a scenario which contains the several different assignments related to each phase of SDLC. For this reason, this page consists of six different segments: "*Project Information*", "*Requirement Analysis*", "*Design*", "*Implementation*", "*Test*" and "*Maintenance*" respectively.

In the first segment the administrator should enter the project definition into the related text-box, which is shown in Figure 3.2. After writing the whole project definition, the administrator should pass the second segment in order to enter the requirements of the project. In this page, there are two different categories that are "*Real Requirements*" and "*Fake Requirements*" (see Figure 3.3).

Scenario Generator

Project Information Requirement Analysis Design Implementation Test Maintenance

Real Requirements	Fake Requirements
Requirement 1: <input type="text"/>	Requirement 1: <input type="text"/>
Requirement 2: <input type="text"/>	Requirement 2: <input type="text"/>
Requirement 3: <input type="text"/>	Requirement 3: <input type="text"/>
Requirement 4: <input type="text"/>	Requirement 4: <input type="text"/>
Requirement 5: <input type="text"/>	Requirement 5: <input type="text"/>
Requirement 6: <input type="text"/>	Requirement 6: <input type="text"/>
Requirement 7: <input type="text"/>	Requirement 7: <input type="text"/>
Requirement 8: <input type="text"/>	Requirement 8: <input type="text"/>
Requirement 9: <input type="text"/>	Requirement 9: <input type="text"/>
Requirement 10: <input type="text"/>	Requirement 10: <input type="text"/>

Figure 3.3: Requirement Analysis Segment

This study can be considered as a kind of serious game so that it is necessary to put some gaming concept into it. Because of this reason, there are two different requirement categories in this segment since the participant, who reads the project definition, should determine the correct requirements for the project from the pool, which consists of both correct and wrong requirements. Therefore, the system administrator can figure out whether the participant can improve his/her level of knowledge by distinguishing the correct requirements from the wrong ones. In addition, the participants have a chance to see their missing points since the virtual environment has the ability to give feedbacks to participants about their wrong choices during the simulation. As a result of this phase, the system administrator can add both correct and wrong requirements about the related project topic.

After recording the requirements of the project, the third segment asks the system administrator to enter both the actors and the functionalities of the system as shown in Figure 3.4. In addition, the system administrator can also create the relationships between the actors and the functionalities using this segment. With this property, novice software engineers can create use-case diagrams of software projects when they arrive at the design stage in the virtual office environment.

Scenario Generator

Project Information Requirement Analysis **Design** Implementation Test Maintenance

Please enter a user type: Save User

Please enter functionality: Save Function

Create Relationship

Users: Functions:

Create Relation

	Relation Number	User Type	Relation Name
*			

Figure 3.4: Design Segment

Figure 3.5 illustrates the implementation segment of the module. In this page, the system administrator records the problem that is expected from the trainee to solve by developing an algorithm in the virtual office environment. Thanks to this phase, the trainee’s coding ability can be measured since the trainee should write C# code to solve the question entered by the system administrator.

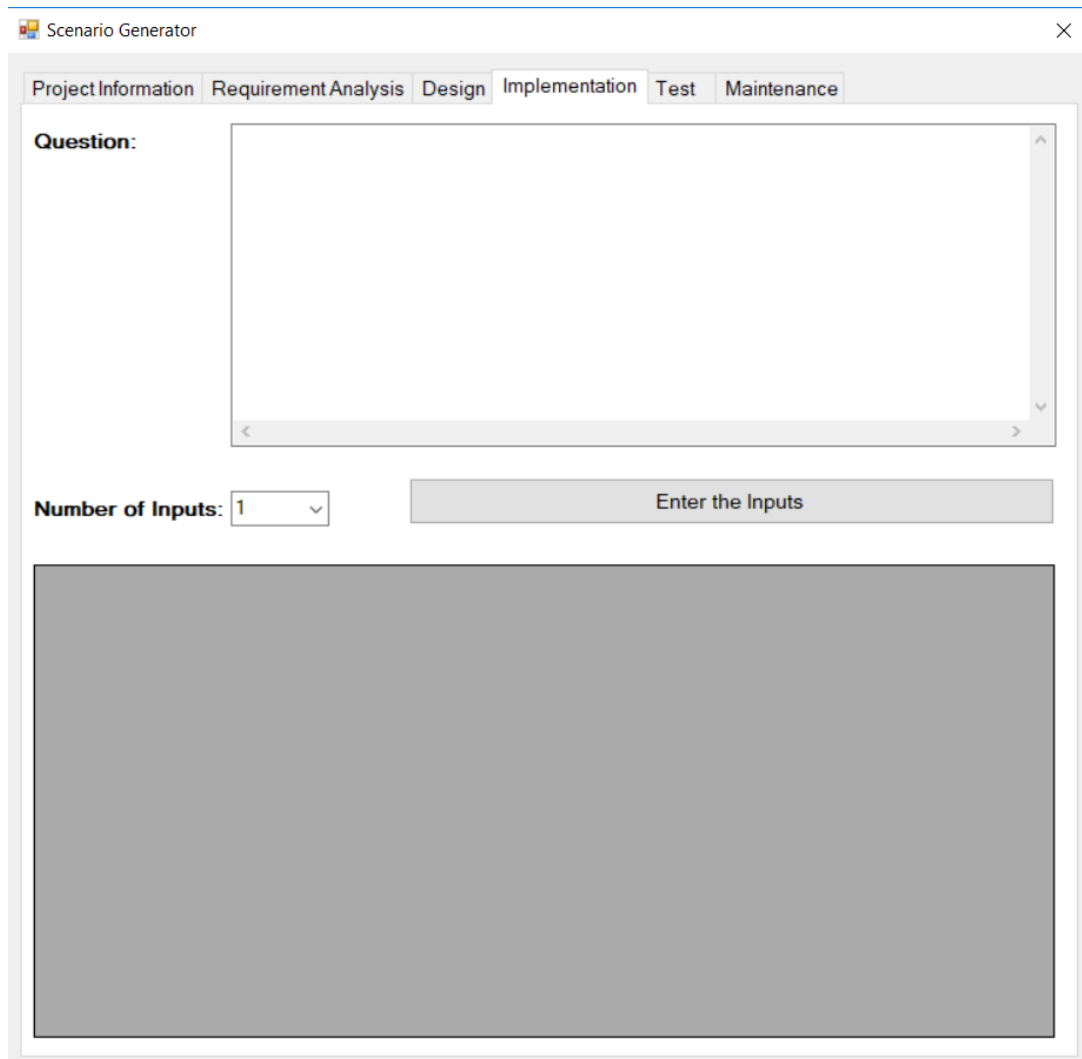


Figure 3.5: Implementation Segment

This code is going to be automatically executed in the virtual environment. Hence, it is necessary to add the inputs appropriate to the structure of the code and the outputs obtained from those inputs into the scenario in order to understand whether the participant writes the code correctly. For this reason, this page allows the system administrator to input the desired amount of input set in order to prevent the user from guessing and writing the output directly without writing code in the virtual office environment (see 3.6).

In the test phase of the software development process, the participant is expected to perform a code review in the virtual office environment. In order for the participant to be able to perform a code review, the system administrator must add a code block with

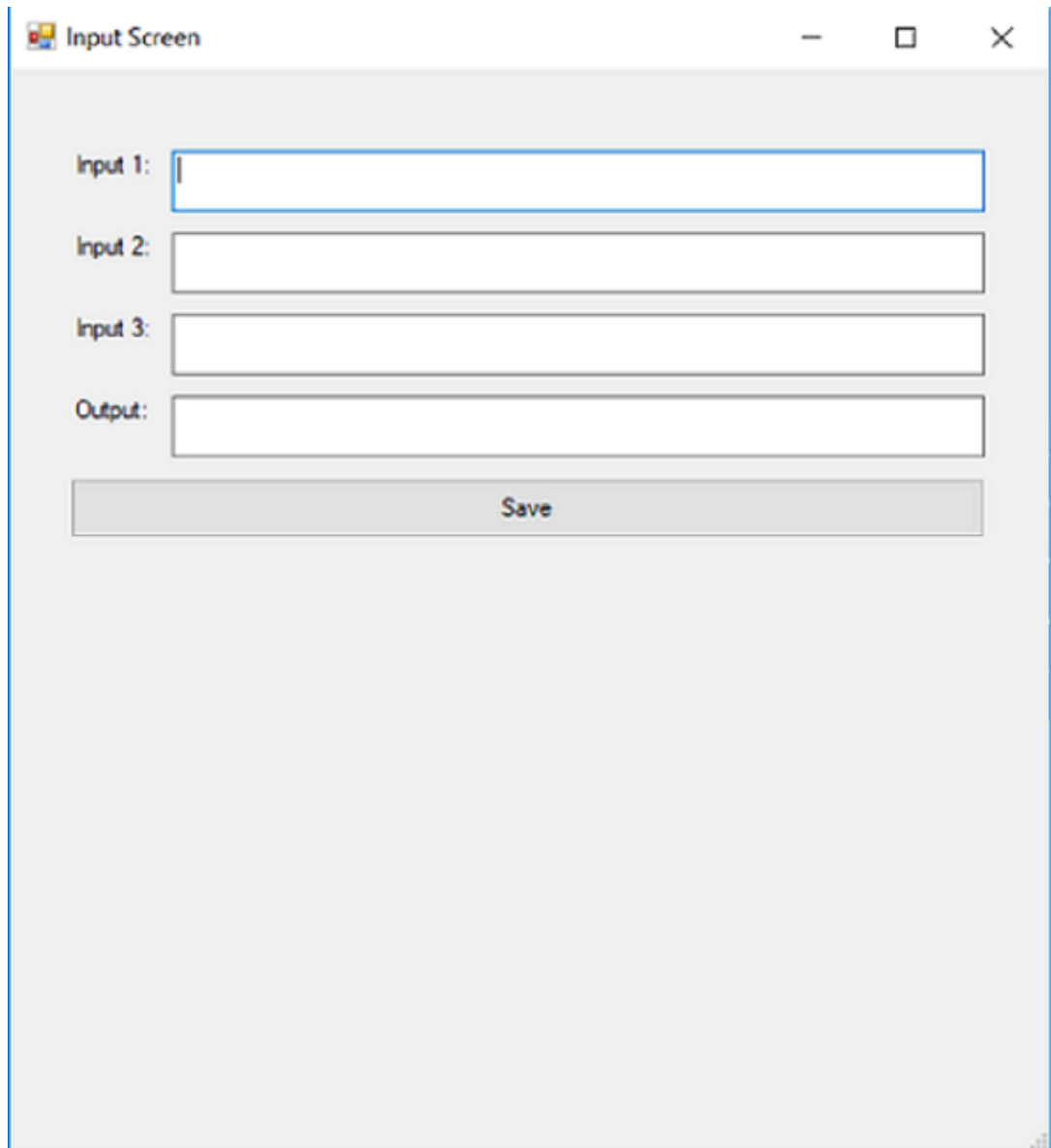
The image shows a software window titled "Input Screen" with standard Windows window controls (minimize, maximize, close) in the top right corner. The window has a light gray background. Inside, there are four input fields arranged vertically. The first field is labeled "Input 1:" and has a blue border with a cursor inside. The second field is labeled "Input 2:" and has a gray border. The third field is labeled "Input 3:" and has a gray border. The fourth field is labeled "Output:" and has a gray border. Below these fields is a wide, light gray button labeled "Save".

Figure 3.6: Input Screen

errors to the software development scenario. The screen in Figure 3.7 is designed to allow this work to be done.

After the system administrator writes the code that includes bugs and errors, s/he should determine the lines and the definitions of the errors. At this stage, there is a similar logic to the requirement analysis phase. This means that the system administrator can also add an error that is not in the code. In this case, the participant should choose and match the correct error lines and definitions from the pool that is constituted both correct and wrong errors in the virtual office environment.

Scenario Generator

Project Information Requirement Analysis Design Implementation **Test** Maintenance

Code Lines:

Enter the Error Definition:

Enter the Line Number that Contains Error:

☐ Dummy Error

Save

	Line Number	Error Definition	Type
*			

Figure 3.7: Test Segment

For the final segment, which contains the fields related to the maintenance operations of the project, the system administrator inputs both the definition and the order of the tasks which should be ordered by the participant in the virtual office environment. A screen as shown in Figure 3.8 is designed for the system administrator to enter this information.

After entering all information about the whole software project development process, an Extensible Markup Language (XML) file, which stores the whole information entered by the administrator, is automatically created if all necessary fields are filled. This file plays a critical role to standardize the scenarios related to software project development process. The content of this file should be easily understood by people

Scenario Generator

Project Information Requirement Analysis Design Implementation Test Maintenance

Task:

Task Order: 1 Save

Task Order	Task
*	

Create Scenario

Figure 3.8: Maintenance Segment

working in the field of software development. In addition, it has to be also understandable by the computer in order to visualize the entered scenario in the virtual environment. Due to these reasons, an XML file is created by this desktop application in order to create a common language for software project scenarios. The empty content of the XML file is illustrated as in Figure 3.9.

3.2 Virtual Office Environment

The virtual office environment was designed to teach the basics of software engineering concepts to novice software engineers. In this environment, they have a chance



Figure 3.9: Empty Content of the XML File

to live the whole software development process without actual risks. Therefore, the novice software engineers may increase both their knowledge and experience levels about software engineering concepts by interacting with the environment with several different functionalities as shown in the use-case diagram of the participant in Figure 3.10.

This virtual environment, designed in close proximity to the real life, aims to experience a software project from the requirements analysis phase to the maintenance phase. To accomplish this aim, an input, which contains the tasks related to the software engineering, is required. As was mentioned in the previous section, this input is provided by the "Scenario Generator" program as an XML file, which contains a scenario of a software project. Hence, it is necessary to read the information in this XML file in order to make the software project live in the virtual environment. For this reason, when the participant opens the virtual office environment, a window in Figure 3.11 appears on the screen in order to select the XML file generated by the "Scenario Generator" program.

Before starting the simulation, the participant has to select the XML file. Otherwise, the program gives a warning message that prompts the user to select the file. When the file is uploaded the system, the participant starts the simulation as shown in Figure 3.12.

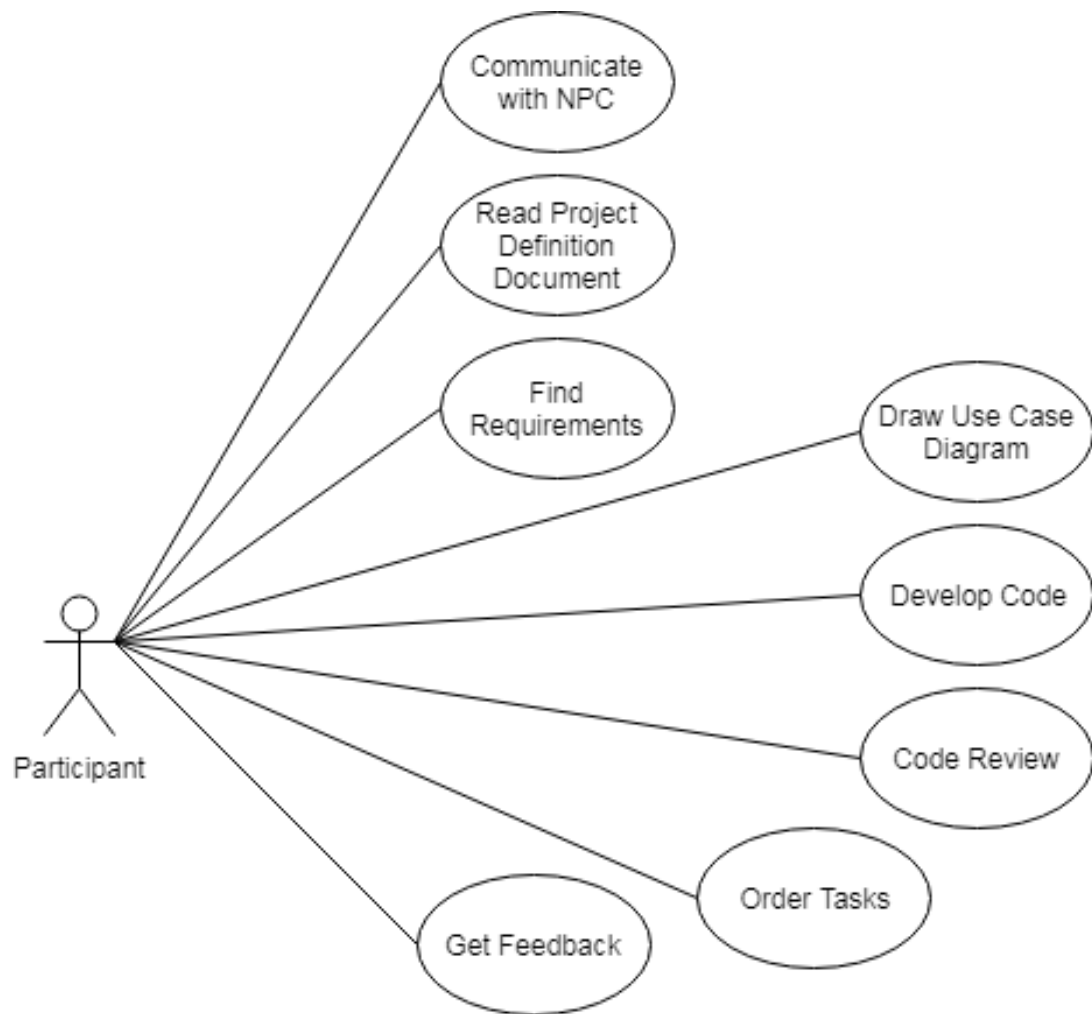


Figure 3.10: Use Case Diagram of Participant

As shown in this figure, the participant stands at the entrance of an office. In the upper left corner of the screen there is a clock to show the elapsed time in the virtual environment while there is a calendar to show the date in the right side of the screen. The date was set up as June 1 at the beginning of the simulation since the participant portrays the character who is a new worker of the virtual company. In addition, the time zone of the virtual environment is adjusted to equal an hour in virtual environment to a minute in real life. Hence, a working day in the virtual office lasts 9 minutes in real life. After passing 9 minutes in real life, the second working day in the virtual office will start with an update in the date.

As it can be easily seen from Figure 3.12, there are five different cubicles in the office. Each cubicle represents the each phase of SDLC. The cubicles on the left side of the



Figure 3.11: Opening Screen of Virtual Office Environment



Figure 3.12: Starting Screen of Simulation

screen are divided into two regions: the lower cubicle is for the requirement analysis phase and the upper cubicle is for the design phase. The cubicles on the right side of the screen are designed as the places where tasks relating to implementation, test and maintenance phases are sequentially performed from bottom to top. Each cubicle has a total of 4 employees, one of whom works on the same project as the virtual character played by the participant. Hence, this virtual environment has a total of 20 non-player characters (NPC). Five of them have a basic AI algorithm with case-based reasoning approach to guide the participant by determining the participants' progress in the environment.

The process of the project in the virtual environment starts when the participant's character goes to the character's first project partner in the first cubicle. In order to show that the employees are in the same project as the participant's character, names



Figure 3.13: Team Member in Requirement Analysis Phase

are added on the employees as shown in Figure 3.13. The participant has to press the "E" key on the keyboard to communicate with NPC. After pressing the "E" key, a dialogue box opens as shown in Figure 3.14.

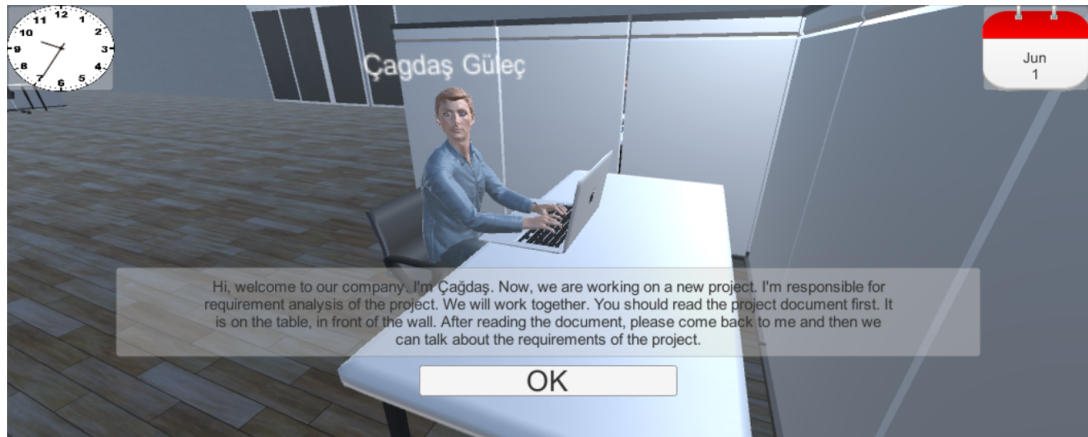


Figure 3.14: First Dialogue Box of NPC in the Requirement Analysis Phase

In this box, a message containing the tasks that the participant should complete is illustrated to the participant. Once the participant receives the message from the NPC in this manner, s/he continues to move in the environment to complete the task. As the participant is expected to perform the requirement analysis at the first stage, the NPC is asked to read the project description document from the participant in the first message. Hence, the participant should first read the project description document. To achieve this aim, s/he should find the project file. For this reason, a flashing light is added to the file to show the document in the virtual office environment as shown in Figure 3.15.

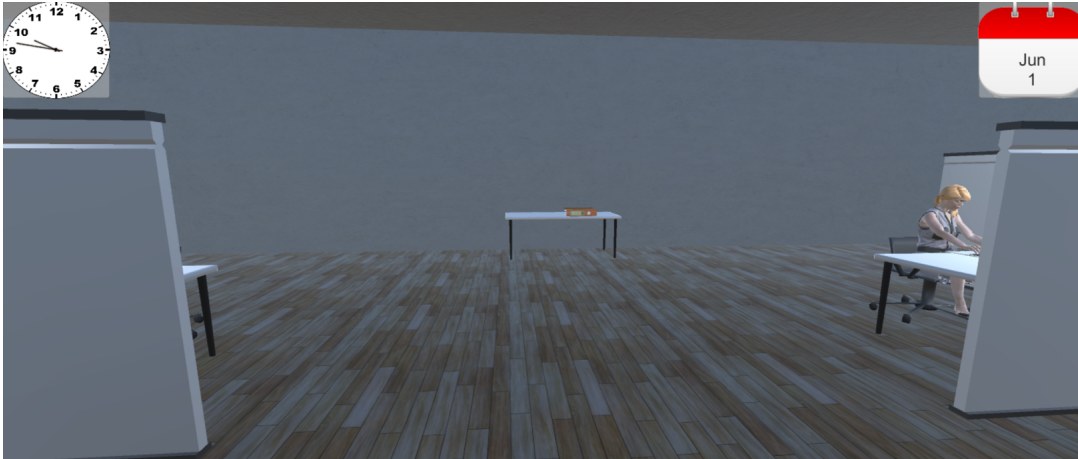


Figure 3.15: Project Description Document

When the participant moves to the side of the document as shown in Figure 3.16, a message is displayed on the screen indicating that the participant should press the "E" key to read the document.

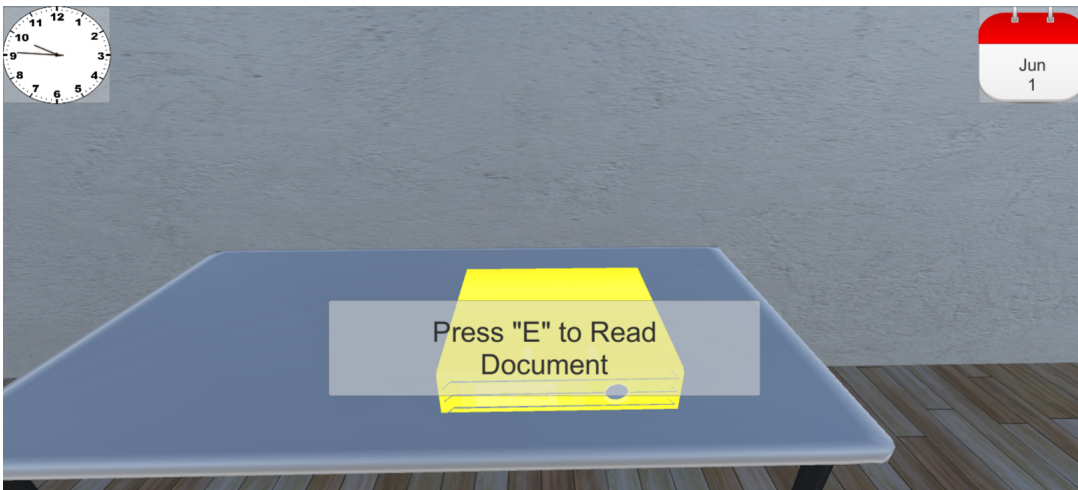


Figure 3.16: Opening Project Description Document

After pressing the "E" key to open the document, the panel shown in Figure 3.17 appears on the screen.

This screen contains the definition of the project that the system administrator specified in the scenario generation program. To increase the reality of the virtual office environment, both file model used in real life and paper texture as a background of the panel is added to the screen since the participant reads the project description from the papers inside of the file. The clock and calendar on the left and right upper

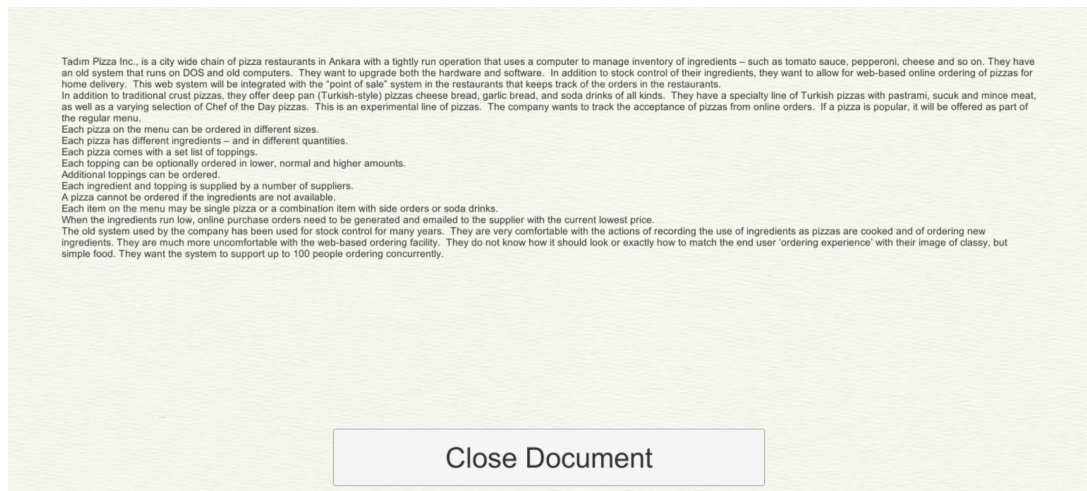


Figure 3.17: Reading Project Description Document

corners, which appears as the participant moves around in the office environment, are removed from this screen in order not to limit the display on the screen. After reading the project definition, the participant returns to his/her team member as shown in Figure 3.18. The NPC on this screen understands that the participant has read the project definition document and tells the participant that s/he can start the requirement analysis phase of the project.

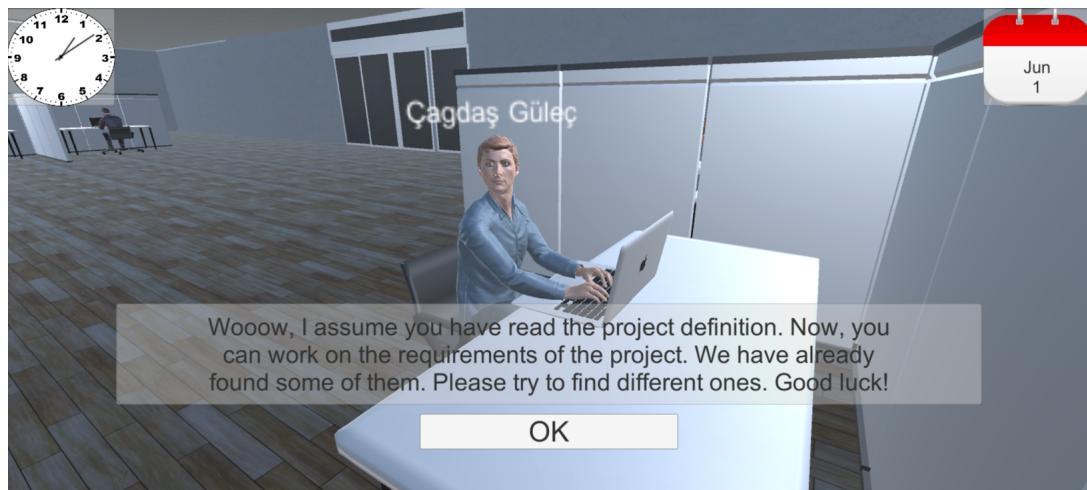


Figure 3.18: Finishing Project Description Document

When the participant starts the requirement analysis phase, the panel in Figure 3.19 appears on the screen. This screen contains two different sub-panels, being left and right, and two buttons, one being at the center of the screen and the other being at the bottom of the screen. In the left panel, both the correct and the wrong requirements

entered by the system administrator in the scenario generation program are listed as a button. In the right panel, some of the correct requirements entered by the system administrator in the scenario generation program are randomly selected and listed. The participant should find and transfer the correct requirements of the project from the list of all requirements. To do this operation, s/he has to click the button, which contains a requirement of the project. When the participant clicks the button, the background colors of the selected button changes from gray to green to show the participant's selection. The participant can choose several requirements one at a time. After selecting the requirements, the participant should click "*Transfer Items*" button to transfer the selected requirements to other list. When the participant considers that the requirement analysis operations are completed, s/he should click "*Done*" button to finish the requirement analysis phase.

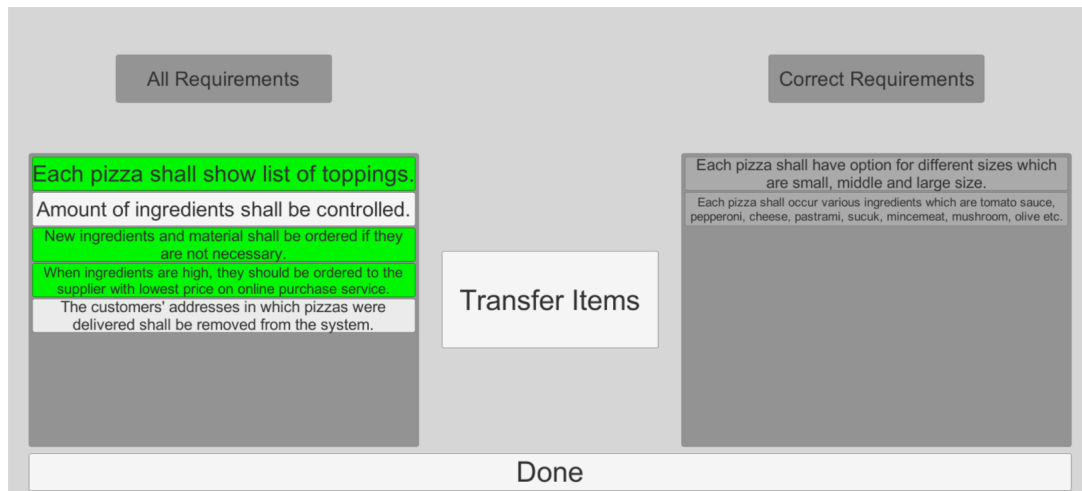


Figure 3.19: Requirement Analysis Phase in Virtual Office

After finishing this phase, the NPC, who is responsible from the requirement analysis phase of the project, informs the participant that this phase is over and that s/he should go to the design phase. After the participant gets this information, s/he goes to the NPC responsible for the design phase of the project (Figure 3.20).

In the design phase, it is expected from the participant to create a use-case diagram of the project as an early design phase of the project [95, 96, 97]. As was mentioned in the previous section, the system administrator determines the actors, the functions and the relations between the actors and the functions of the project via using scenario generation program. In the virtual environment, the participant should create the



Figure 3.20: First Dialogue Box of NPC in the Design Phase

correct relations between the actors and the functions. As shown in Figure 3.21, both the actors and the functions are listed in the drop-down lists separately.

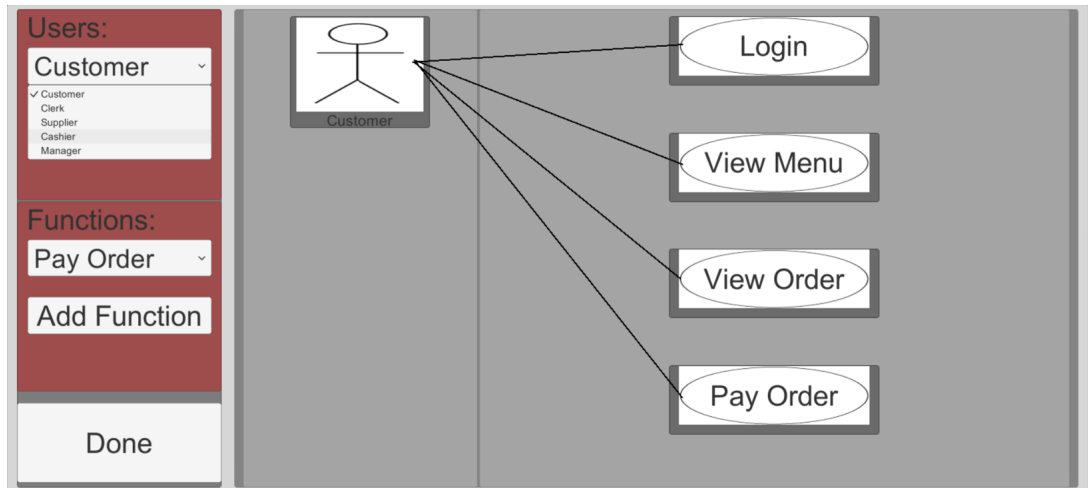


Figure 3.21: Design Phase in Virtual Office

In this screen, the participant can choose both the actors and functions via using drop-down lists. In order to create the use-case diagram, the participant should select the necessary actors and functions. For the actor selection operation, s/he selects the actors from the drop-down list and clicks the "Add User" button. When this button is clicked, the selected actor appears on the screen as a classical stick man. The same idea is valid for the function selection. After the actors and the functions are ready on the screen, the participant can create the relations between them. To do this, the participant should select one of the actors or one of the functions on the screen. When

an item is selected, a line starts to be drawn on the screen. The direction of this line depends on the direction of the mouse. This means that a line is being drawn towards the direction of the mouse when the participant drags the mouse until selects another item. However, there is a rule in here. This rule says that if the participant selects an actor, the other selected item should be a function or vice versa. In this way, the drawing of the use-case diagram is completed by establishing the relationships between the actors and the functions. If the participant wants to remove the added items, s/he should click the items two times. When an item is removed, the relation or the relations that are connected the removed item are also removed. After completing the diagram, the participant should click the "*Done*" button to complete the design phase of the project.

When the design phase is completed, the NPC responsible for the design phase of the project warns the participant that s/he should start the implementation phase of the project. The participant who receives this warning goes to the NPC, who is in charge of implementation phase (Figure 3.22).



Figure 3.22: First Dialogue Box of NPC in the Implementation Phase

The NPC, who is responsible for this phase, first congratulates the participant for completing the first two stages and indicates to the participant the task to be done at this stage. In this phase, the participant is expected to code a function of the corresponding project with the C# programming language. In order for the participant to write the code, the panel in Figure 3.23 appears.

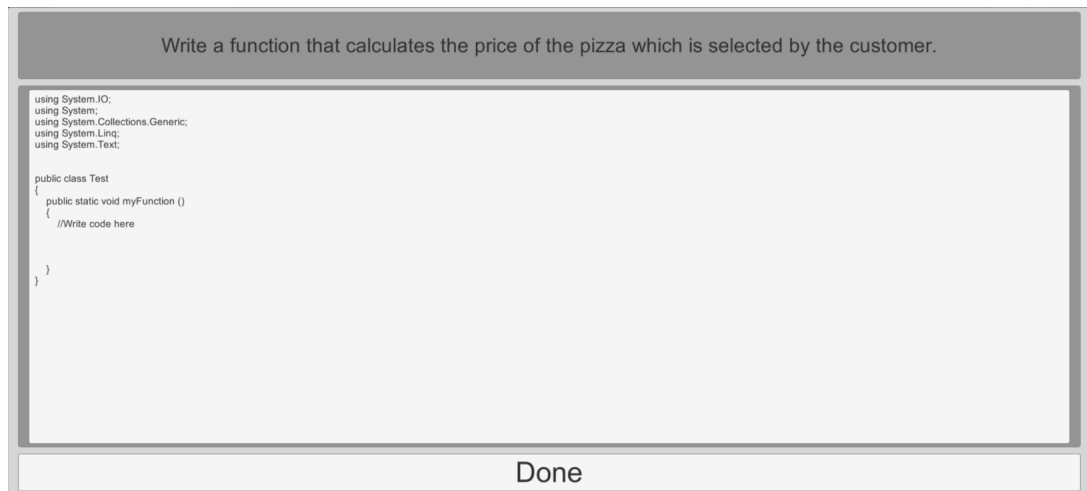


Figure 3.23: Implementation Phase in Virtual Office

In this panel, there is a label at the top of the panel which includes the text of the question that should be solved by the participant. In addition, there is a text-box at the center of the panel which enables the participant to write the C# code to solve the problem. In this section, a compiler, which is capable of executing the codes written in C# programming language, was designed to see whether the code written by the participant is correct. However, this compiler needs to have a specific structure in order to run the code. This means that both the class name and the function name should be fix to execute the code. For this reason, when the panel designed for the implementation phase is opened, the structure of the code is ready inside of the text-box. The participant can add new functions inside of the class, however, the functions named with "*myFunction*" should be considered as a main function. Thus, it should not be changed by the participant. The code written by obeying these rules can be executed by the virtual environment without having any problems. This compiler checks whether there is an error (run time or compile time) in the code. However, it can not check whether the code being written is working properly for the purpose. To overcome this problem, some necessary string operations were applied to the code written by the participant before it was sent to the compiler. These operations can be explained as follows:

- The code, which is requested from the participant to solve the problems, mostly requires input. Hence, the participant should write "*Console.ReadLine()*" command to get the inputs. However, it is impossible to get inputs from the user of

the program. For this reason, "*Console.ReadLine()*" commands in the code exchange with the input values entered by the system administrator in the scenario generation program to test the code before sending it to the compiler.

- The compiler runs the code with the inputs entered by the system administrator. In most of the programs, the coders write "*Console.WriteLine()*" command to illustrate the output of the program. Hence, the value written in the "*Console.WriteLine()*" command is read as the output of the program. After that, this read value is compared with the output given by the inputs entered by the system administrator. Therefore, it can be easily understood whether the code written by the participant is correct or not.

After writing the code, the participant should click the "*Done*" button to complete the implementation phase of the project. When the implementation phase is completed, the NPC responsible for the implementation phase of the project warns the participant that s/he should start the test phase of the project. The participant who receives this warning goes to the NPC, who is in charge of this phase, in order to start the test phase (Figure 3.24).

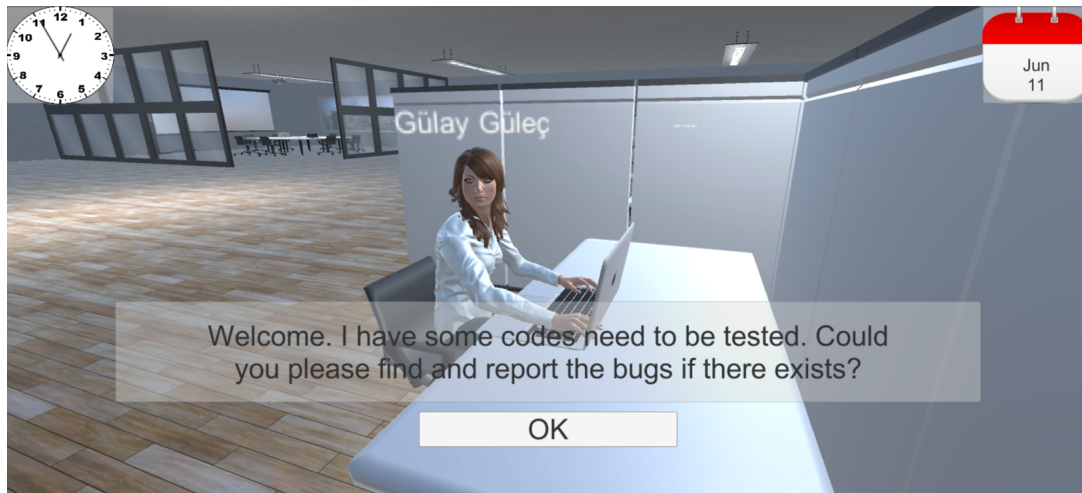


Figure 3.24: First Dialogue Box of NPC in the Test Phase

In this phase, the NPC guides the participant that they have a code block which is required to be tested with the code-review technique since this technique plays a critical role in the testing phase of SDLC [98, 99, 100]. The participant, who takes the mission from the NPC, starts to do code review on the code which is shown in

Figure 3.25.

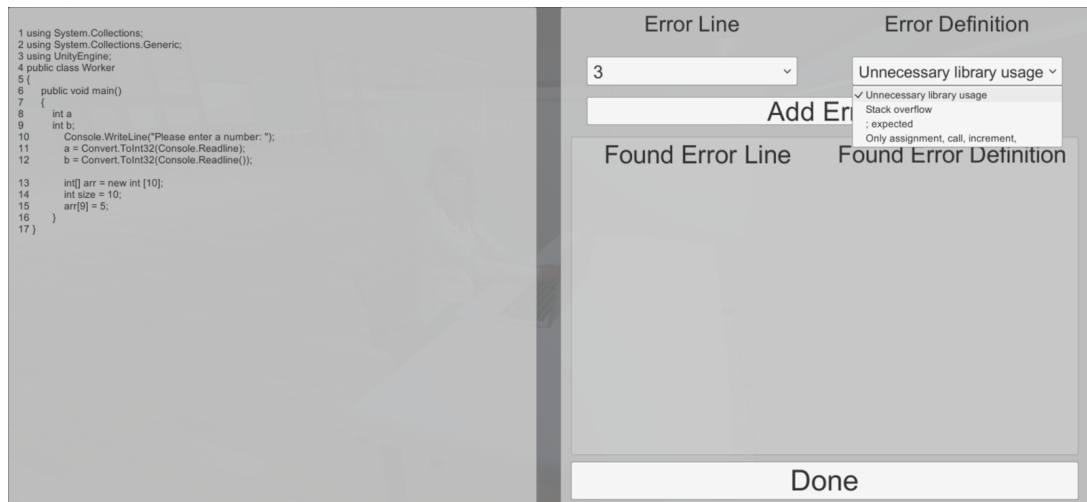


Figure 3.25: Test Phase in Virtual Office

This panel is constituted by two different sub-panels. In the left sub-panel, the code block that is entered by the system administrator in the scenario generating program and which should be tested by the participant is displayed to the participant in a label. The number of code lines is automatically generated and integrated into the string which includes the code block. In the right sub-panel, there are two different drop-down lists, which are designed for the error line and definition respectively. These drop-down lists include both correct and wrong errors. Thus, the participant should find the correct ones and also match them correctly. Hence, there are two challenges in this phase. When the participant wants to add the selected error lines and definitions, s/he should click "*Add Error*" button. When this button is clicked, the selected items in the drop-down lists are listed in the below section. If the participant wants to remove the added items, s/he should click the items two times. After finding the errors, the participant should click the "*Done*" button to complete the test phase of the project.

When the test is completed, the NPC responsible for the test phase of the project warns the participant that s/he should start the maintenance phase of the project. The participant who receives this warning goes to the NPC, who is in charge of this phase, in order to start the maintenance phase (Figure 3.26).

The NPC, who is responsible from the maintenance phase of the project, congratulates

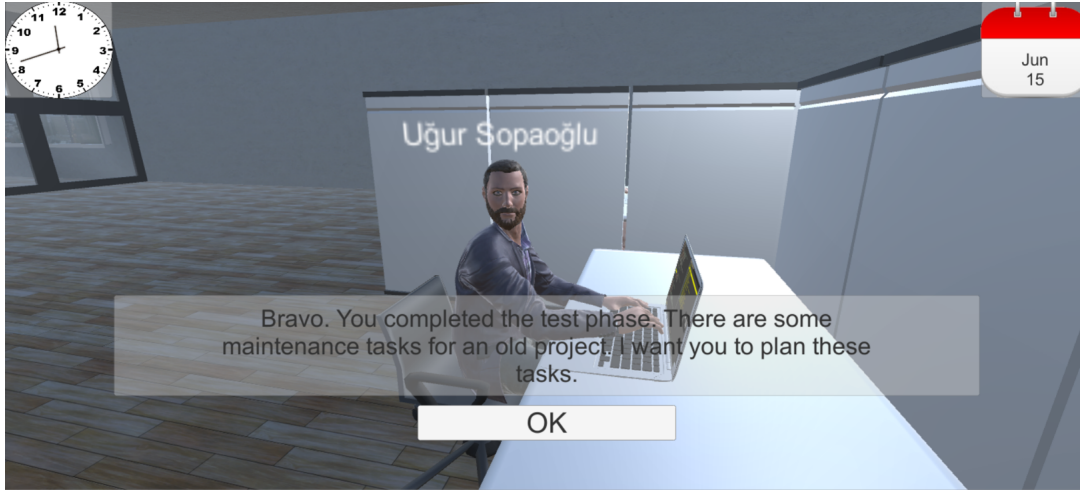


Figure 3.26: First Dialogue Box of NPC in the Maintenance Phase

that the participant has arrived at the last stage and indicates the participant in this phase of the tasks required to do so. In this phase, participant is expected to put the tasks added to the system into the correct order. As shown in Figure 3.27, the tasks are listed in the left of the panel. This structure is similar to the structure of the requirement analysis phase. The only difference is that the participant cannot choose more than one task at the same time to transfer the selected item to the other list. The reason for this constraint is that the participant should constitute the order of the tasks. Hence, a task should be selected one at a time.

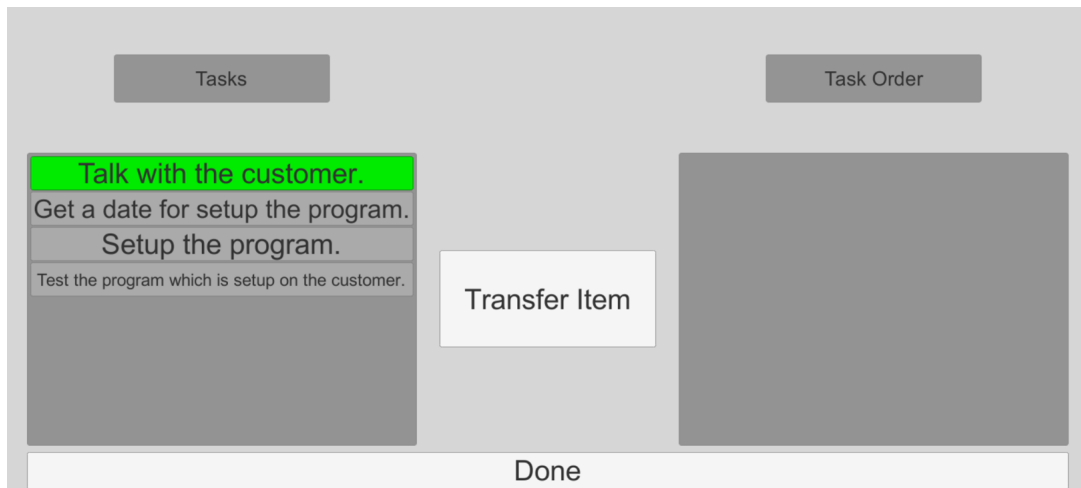


Figure 3.27: Maintenance Phase in Virtual Office

When the participant considers that this phase is completed, s/he should click "*Done*" button to finish this phase. After finishing this phase, the NPC tells the participant

that the simulation is over.

When the simulation is over, the evaluation process starts for the participant. This process is important for the participants, because, the aim is to increase the level of knowledge of the participants. To do this, a file, which consists of the true answers of the tasks and the time that is spent by the participant in the each phase of SDLC, is generated at the end of the simulation. The participants have a chance to see where they made mistakes, what the correct answers were and how much time they spent to complete the task in each phase of SDLC through the information written in the feedback file. In this way their abilities, knowledge and experience levels can be improved.

CHAPTER 4

TEST AND EVALUATION METHODOLOGY

This chapter describes the test and evaluation methodology of the current study. First of all, this study is conducted by using the features of mixed research methodology since both qualitative and quantitative analysis are applied to understand whether the proposed environment works as expected or not. Hence, this chapter explains the qualitative, quantitative and mixed research techniques. Then, the overall research design is shown. After that, the systematic flow of the proposed environment is illustrated in detail. Finally, this chapter is finalized by providing information on the participants who use the system to increase their knowledge levels about software engineering and listing the threats that may affect the results of the study in a negative way.

4.1 Qualitative Research Methodology

Qualitative research methodology aims to get meaningful results through the text obtained by interviewing individuals and taking the opinions of them about the related research subjects [101]. In this technique, there are several different ways to get opinions from people such as conducting interviews, making surveys and recording the voice of the participants. This technique can be also called "*numberless research methodology*" since there is no numerical data and analysis in this research methodology.

The qualitative research part of this study is completed in two different stages as:

- To administer an interview to get the suggestions of the lecturers about the flow

of the system.

- To administer an interview to get the opinions of the participants about the proposed virtual environment.

For the first part of the qualitative analysis, the lecturers examined the general concept of the system. Since this study aims to train the tasks that should be completed at the each phase of SDLC, the content of the each task should be validated by the lecturers. The reason for this validation is that this framework can be considered as a kind of serious game. Hence, the content of the tasks should be educational as far as not boring.

In the second part of the qualitative analysis, an interview was administered with the participants to figure out the effect of the system. This interview is essential since the opinions of the participants determine whether this system can be used as a training tool for the tasks occurred in the each phase of SDLC.

4.2 Quantitative Research Methodology

Quantitative Research Methodology requires numerical values to reach the exact information by analyzing the numerical results of the tests administered with the participants [102]. In general, participants are divided into two groups, one is control group and the other is experimental group, to figure out the effect of the developed system. The participants in the control group continue to work in traditional way while the individuals in the experiment group work with the newly developed system [103]. At the beginning of the study, a preliminary test is conducted to measure participants' knowledge levels. At the end of the study, a post test is administered to measure the progress of the participants in different groups.

In order to complete the numerical analysis part of this study, a methodology similar to the one described above was conducted. This means that there are two different groups, one is control group and the other is experimental group, in this study. At the beginning of the study, it is necessary to determine the knowledge levels of the participants in each group. For this reason, a pre-test was administered with the participants

before starting the training. Then, participants in the control group studied on the software engineering topics for 6 weeks from traditional methods such as books, slides or lecture notes, while the experimental group worked on the same topic by using the designed environment without benefiting from traditional resources. Finally, a post-test was administered with the participants in each group to identify whether the participants' knowledge levels improved. When the numerical values of these tests were obtained, the suitable statistical tests such as *Two-Sample T-Test* was applied the numerical data to illustrate the difference in the knowledge levels of both group members in a statistical way if the difference exists. In addition to these statistical tests, *Presence Questionnaire (PQ)* and *Immersive Tendencies Questionnaire (ITQ)* were conducted with the participants in the experimental group to measure the realism of the proposed virtual environment by determining the presence levels of the participants.

4.3 Mixed Research Methodology

Mixed Research Methodology contains the properties of both qualitative and quantitative research methodologies [104]. In this technique, both numerical and non-numerical analysis are conducted. It can be also considered as a hybrid approach between the qualitative and quantitative analysis [105]. Due to this technique, the findings obtained in both analyzes should complement each other.

In this study, a quantitative research approach supported by validation interviews, which can be considered as a type of mixed research methodology, was utilized since the research methodology of this study contains the properties of both qualitative and quantitative research. A 3D virtual office environment was developed to train novice software practitioners about the software development process. This environment was tested with the students studied at computer engineering department as novice software engineers in order to figure out whether the designed system is successful. To achieve this issue, a pre-test (before the training period) and a post-test (after the training period) were administered to the participants in order to observe their progress about software development process. These tests have provided numerical values as quantitative data so that quantitative part of this study was completed by analysing the

numerical values with the statistical methods. In addition to these values, Presence Questionnaire (PQ) and Immersive Tendencies Questionnaire (ITQ) were applied to the participants to measure their presence levels in the designed virtual environment. With these questionnaires, the participants' presence levels were determined with the numerical values. Hence, these numerical values also provide the quantitative part of the research methodology of this study. For the qualitative part of the study, two different interviews were organized. In the first interview, a meeting was held with the lecturers in order to understand whether the system would be useful for the individuals working in the area of software development before the development of the virtual environment. In addition, a set of semi-structural interviews was also administered to the participants in order to obtain their opinions about the system after the training period. Therefore, the qualitative part of the research methodology was completed by analyzing the comments of the lecturers and the opinions of the participants of the study.

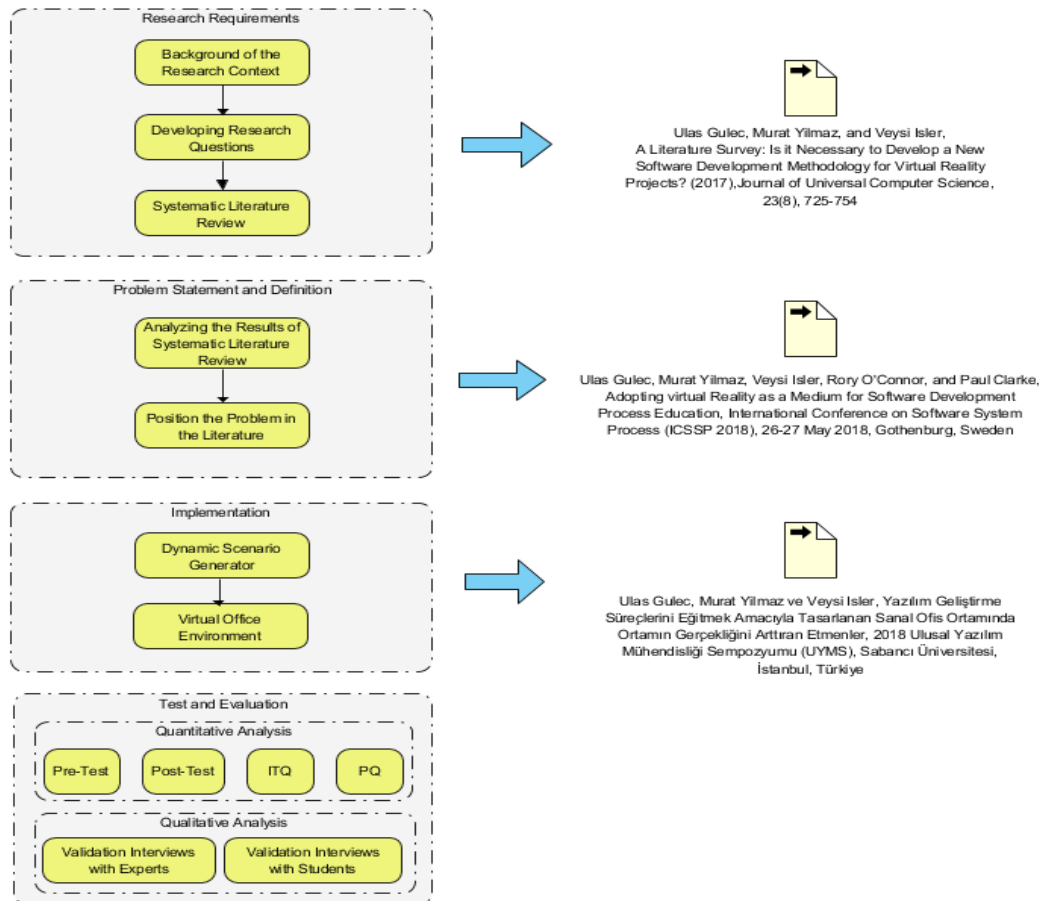


Figure 4.1: Overall Research Design

4.4 Overall Research Design

The overall research design of this study can be summarized as shown in Figure 4.1. It started by reviewing the literature and arranging the designing systematic literature review (SLR). The outcomes of the SLR [106] was published in Journal of Universal Computer Science. After completing the literature review, the problem was specified with respect to the findings obtained from the literature review. After determining the problem, a solution was proposed for this problem during the second phase of the research. The output of this phase was an international conference paper [107] as a position paper. For the proposed solution, a training environment was developed in the third phase of the study. Based on the results of this phase, a national conference paper [108] was published. In the final phase of the study, proposed training platform was tested with students from a private institute. In these tests, both qualitative and quantitative data was gathered and used to determine whether the proposed system is successful.

4.5 Mechanism of the System

As previously mentioned, a training platform was developed to train novice software engineers about the software development process without having actual risks. The aim is to increase the level of knowledge and experience of the novice software engineers about the software development process by eliminating the missing points of the existing studies in the literature, which have been developed for similar purpose. To achieve this aim, two complementary applications were developed in the scope of this study:

1. A desktop application for generating software project scenarios
2. A 3D virtual office environment

The purpose of the desktop application is to enable the authorized person/people in companies to create different software project scenarios, which contain the tasks related to the whole project development process including requirement analysis, de-

sign, implementation, test and maintenance phases. This application produces an XML file, namely with "*Scenario.xml*", that consists of the whole entered information related to the project scenario by creating an ontological meta-language, which can be understood by the individuals and computers. The tasks that should be completed by the participants are written inside of this file. The tag structure of the XML file is as follows:

- **<scenario> ... </scenario>**: This tag is the outermost tag of the file. It shows that the scenario begins and ends.
- **<project> ... </project>**: This tag in the "*Scenario*" tags indicates that the information about the project starts and finishes.
- **<definition> ... </definition>**: Project scenario starts with project definition. For this reason, this tag is the first tag in the "*Project*" tags and keeps the information of the project definition.
- **<requirements> ... </requirements>**: This tag, which is the second tag in the "*Project*" tags, has been created to show that information about the requirement analysis has begun to be retained. The designed system can be considered as a serious game, which teaches something to users in an enjoyable manner [109]. Hence, it involves some gaming concepts such that the novice software engineer should find the correct requirements from the whole requirement list. For this reason, this tag stores both the correct and wrong requirements of the project.
- **<realReqItem> ... </realReqItem>**: This tag is the first tag in the "*Requirements*" tags. It is designed to keep the correct requirements of the project.
- **<fakeReqItem> ... </fakeReqItem>**: This tag is the second tag in the "*Requirements*" tags. It is designed to keep the wrong requirements of the project.
- **<design> ... </design>**: The information about the design phase of the project is kept in this tag, which is the third tag in the "*Project*" tags. At this stage, it is expected from the novice software engineer to create the use-case diagram of the project by finding the correct actors and functions, and matching them

correctly. Hence, this part contains the information of the actors and functions of the system.

- **<user> ... </user>**: This tag is the first tag in the "*Design*" tags. It is designed to keep the actors of the project.
- **<function> ... </function>**: This tag is the second tag in the "*Design*" tags. It is designed to keep the functions of the project.
- **<relation from = " " to = " " />**: This tag is the third tag in the "*Design*" tags. It stores the relation information between the actors and the functions. Instead of the other tags, this tag does not store any information inside of itself. It has 2 properties, one of them "*from*" and the other one is "*to*", is to keep the necessary information. "*from*" tag is designed for the actor information and "*to*" tag is for the function information. In this way, the relations in the use-case diagram can be stored.
- **<implementation> ... </implementation>**: This tag is the fourth tag of the "*Project*" tags. At this phase, it is expected from the novice software engineer to develop an algorithm to solve a problem related to the project. For this reason, the problem statement should be recorded as a question inside of this tag. In addition, the algorithm developed by the novice software engineer should be automatically checked whether it properly works. To make this operation automatically, the necessary inputs and the outputs should be also added to the scenario since the virtual office environment has an ability to execute the written code, however, several input-output sets have to be entered into the scenario in order to prevent malicious behaviour such as calculating the result directly by hand and printing the screen.
- **<codeQuestion> ... </codeQuestion>**: This tag is the first tag in the "*Implementation*" tags. It shows the question text of the problem of the project.
- **<numberOfInputSet> ... </numberOfInputSet>**: This tag is the second tag in the "*Implementation*" tags. It illustrates how many input-output sets are entered by the system administrator.
- **<numberOfInput> ... </numberOfInput>**: This tag is the third tag in the

"Implementation" tags. It illustrates how many inputs are required to test the written code by the novice software engineer.

- **<numberOfOutput> ... </numberOfOutput>**: This tag is the fourth tag in the "Implementation" tags. It illustrates how many outputs should be compared with the output produced by the program developed by the novice software engineer.
- **<inputSet no = " "> ... </inputSet>**: This tag is the fifth tag in the "Implementation" tags. It consists of the values of both inputs and the outputs. This tag has also one property, which is designed to represent the number of the input-output set.
- **<input> ... </input>**: This tag is the first tag in the "Input Set" tags. It stores the values of the inputs.
- **<output> ... </output>**: This tag is the second tag in the "Input Set" tags. It stores the values of the outputs.
- **<test> ... </test>**: The information about the test phase of the project is kept in this tag, which is the fifth tag in the "Project" tags. In this phase, the novice software engineer has to find the bugs and errors of the code, already written before, related to the project. S/he should find the correct error line and the error definition, and match them correctly. A structure similar to the requirement analysis phase is also presented at this stage. In other words, this phase contains both the correct errors and the wrong errors of the code of the project. The novice software engineer has to find the correct ones from the whole errors.
- **<code> ... </code>**: This tag is the first tag in the "Test" tags. It is designed to store the code of the project. The system administrator writes the code that should be checked by the novice software engineer and record it to the file. When the written code is saved to XML file, the code lines are automatically created by the system and are recorded with the code text to the XML file. In this way, the novice software engineer can see the line numbers of the code.
- **<realErrors> ... </realErrors>**: This tag is the second tag in the "Test" tags and illustrates that the information of the correct errors starts and ends.

- **<realErrorLine> ... </realErrorLine>**: This tag is the first tag in the "*Real Errors*" tags. It keeps the line numbers of the correct errors that the code has.
- **<realErrorDefinition> ... </realErrorDefinition>**: This tag is the second tag in the "*Real Errors*" tags. It keeps the definitions of the correct errors that the code has.
- **<fakeErrors> ... </fakeErrors>**: This tag is the third tag in the "*Test*" tags and illustrates that the information of the wrong errors starts and finishes.
- **<fakeErrorLine> ... </fakeErrorLine>**: This tag is the first tag in the "*Fake Errors*" tags. It keeps the line numbers of the wrong errors.
- **<fakeErrorDefinition> ... </fakeErrorDefinition>**: This tag is the second tag in the "*Fake Errors*" tags. It keeps the definitions of the wrong errors.
- **<maintenance> ... </maintenance>**: This tag is the last tag of the "*Project*" tags. In this stage, there are several different tasks related to the maintenance phase. These tasks should be ordered by the novice software engineer. Hence, the information of the tasks, such as the order and the definition of the task, has to be stored between the "*Maintenance*" tags.
- **<tasks> ... </tasks>**: This tag is the first tag of the "*Maintenance*" phase. It shows that the information about the tasks of the project begins and ends.
- **<taskOrder> ... </taskOrder>**: An order number of the task is stored between these tags, which are the first tag in "*Tasks*" tags.
- **<taskDefinition> ... </taskDefinition>**: This tag is the second tag in "*Tasks*" tags. It keeps the definitions of the tasks.

After producing the XML file, it is necessary to animate the scenario written in the XML file in the virtual office environment. In order to accomplish this goal, the XML file is parsed by the virtual office environment at the beginning of the simulation. The information obtained after parsing the XML file is stored in the classes designed in the back-end of the virtual environment. Thus, the novice software engineer can live the project scenario written by the authorized person of the company in the virtual

office environment. The systematic work flow of the study can be summarized as shown in Figure 4.2.

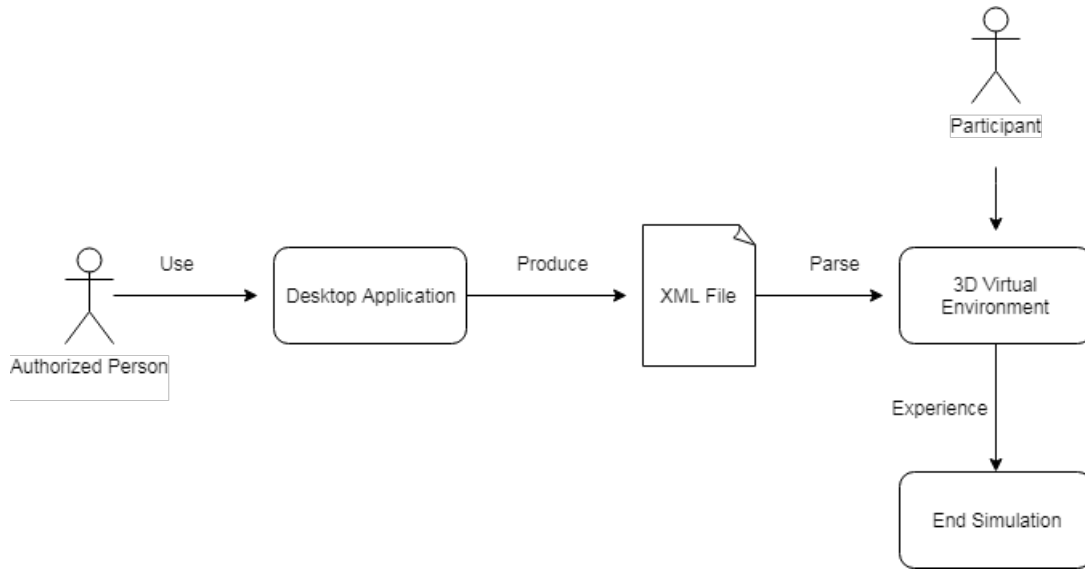


Figure 4.2: Work Flow of the System

4.6 Participants

During this study, 41 students, who are studying at the department of computer engineering, have volunteered to participate to this study in order to test our educational framework. However, 9 of these students were excluded from the study since they are not suitable for the criteria of this study, which are:

- The students should know C# programming language since they develop an algorithm using C# programming language in the implementation phase of the simulation.
- The students should not have taken a "*Software Engineering*" course previously. The reason for this restriction is that if a student has taken this course, we cannot distinguish how much the student knows the concept of software engineering at the beginning of the study since they take this course from different universities. In order to prevent biased results the students who have not taken "*Software Engineering*" course were selected.

The remaining 32 students were randomly divided into two groups, which are control and experimental groups. Thus, we obtained 2 groups of 16 people in order to understand whether the system was successful. The students in the control group did not use the virtual office environment. They have studied the software engineering concepts from the books, presentations or videos. The students in the experimental group just use the simulation environment. They were not allowed to use any other resources related to the software engineering concepts. The training time lasted 6 weeks. Before the training period, a pre-test was administered with the 32 students in order to measure their knowledge levels about software engineering. In this test, a case-study was given to the students and it was expected from them to find the requirements of the project given in the case-study, to create a use-case diagram of the project, to develop an algorithm for the problem in the project, to find the errors inside of a code block related to the project and to order the tasks for maintenance operations of the project. The results of this test were calculated, however they were not shared with the students. After six weeks, a post-test was administered with the students in order to observe their progress. The content of the post-test was similar to content of the pre-test. After obtaining the results, the statistical tests were applied to figure out whether there exists a significant difference between the knowledge levels of students in different groups about software engineering concepts. The time-line of this study is shown in Figure 4.3.



Figure 4.3: Time Line of the Study

We started to look for the students on June 30, 2018. At the end of this operation, 41 students were found, however, 9 of them were extracted from the study since they did not fulfil our limitations. After the remaining students were divided into 2 different groups, a pre-test was administered on July 23, 2018. In this way, the training period began and continued 6 weeks. Each week, meetings were arranged with the

students to enable them to experience different project scenarios on the system. In this way, they had a chance to face the problems occurred in 6 different projects. After each session, PQ was applied in order to measure their presence levels in the virtual office environment. At the end of the 6 weeks, a post-test was administered on September 4, 2018. After getting the numerical values, an interview was organized with the students to get their opinions about the system on September 7, 2018. After obtaining both the qualitative and quantitative data, an analysis process has started and continued until on September 19, 2018.

4.7 Threats to Validity

Each study in the literature may contain some threats to validity that may affect the results of the study in a negative manner [110]. These factors can also decrease the correctness and trustworthiness of the studies. They can be classified in four different categories as shown in Table 4.1.

Table 4.1: Threats to Validity for Empirical Research in Software Engineering

Construct Validity	<p><i>Ability to measure the proposed idea in a correct way.</i></p> <ul style="list-style-type: none"> • Qualitatively: To get the opinions of experts and participants about the proposed system and compare these opinions with the opinions of the researcher. (i.e. validation interviews) • Quantitatively: To measure the progress of the participants on the subject by applying tests. (i.e. pre-test & post-test)
Internal Validity	<p><i>The research design should be internally consistent.</i></p> <ul style="list-style-type: none"> • History effect: The participants may improve themselves about software engineering concepts by using other resources before the training period. • Testing effect: The participants may intentionally give wrong answers to the questions asked in the pre and post tests. In addition, the participants in the experimental group may also study the software engineering concepts from other resources. The students in the control group did not receive any tasks such as homework, projects etc. from us. Hence, their motivation may be decreased since there was less inclination to study the software engineering except post-test. • Instrumentation effect: Any change in the system during the training period.
External Validity	<p><i>The results obtained from this study should be directive for the researchers and should be generalized for target population.</i></p> <ul style="list-style-type: none"> • A conceptual replication: A study develops a similar training environment to teach the tasks related to each phase of SDLC or uses same dynamics to increase the learning process of the individuals.
Reliability	<p>The proposed training environment should be a beneficial tool for the individuals, especially for the students, to study the tasks occurred in the software development process by supporting the traditional techniques.</p>

CHAPTER 5

ANALYSIS AND TEST RESULTS

As was explained in the previous chapters, there are two groups, one is the control group and the other one is the experimental group, in this study in order to understand whether the developed environment is a useful tool to train individuals about software engineering concepts. During the training period, the members of the control group are not allowed to use the virtual office environment to study the software engineering topics. They can only benefit from the books, videos, presentations and other traditional resources for these subjects. On the other side, the individuals, who are in the experimental group, practice the software engineering concepts by using only the virtual office environment. To perform this rule, it was requested from the participants in the experimental group not to study from any books, videos, presentations or any other sources related to software engineering during the training period.

The training period applied to the individuals is 6 weeks and during this period, the people in the experimental group carried out the tasks related to 6 different projects (see Appendix A), a different project scenario each week. In order to figure out the efficiency of our system, it is necessary to observe the progress of the participants of the study. To accomplish this issue, a pre-test, which was organized at the beginning of the study, and a post-test, which was organized at the end of the study, were administered to the participants. In this way, the change in the knowledge level of the participants can be easily obtained by subtracting the results of the participants in both tests. After the tests, two sample t-test was applied to the numerical data obtained as a result of the exams in order to determine whether the results obtained yielded significant results in the statistical scope. In addition to these tests, PQ and ITQ were also applied to the participants in order to measure their involvement levels while they

were using the virtual office environment since this measurement illustrates the success of the virtual environments. In this chapter, the details of pre and post tests, PQ and ITQ, the opinions of the participants about the system and the statistical methods will be described in detail.

5.1 Pre-Test

At the beginning of the training period, a pre-test was administered to the selected group of students in order to determine their knowledge levels of software engineering. In this test, the students dealt with problems related to the sample project. The definition of the project was given to the students and it was requested from them to determine the requirements of the system, to detect the actors and functionalities of the system by drawing the use-case diagram of the project as the design of the system, to develop an algorithm to solve the problem related to the project scenario, to make a code-review of a code block developed for a problem occurred in the project scenario and to order the tasks revealed after delivering the project to the customer. Hence, this system consists of 5 parts: "*Requirement Analysis*", "*Design*", "*Implementation*", "*Test*" and "*Maintenance*".

The project scenario in the pre-test (see Appendix B) was taken from the exam organized in the "*Software Engineering*" course offered at the Department of Computer Engineering, Çankaya University in order to build a proper assessment tool. After determining the content of the pre-test, it was organized on July 23, 2018. 32 students, who are currently studying at computer engineering departments, participated in this exam. As was discussed in the previous part of this study, the students were equally divided into 2 groups, one is the control group and the other one is the experimental group. The pre-test results of both groups are shown in Table 5.1 for experimental group and Table 5.3 for control group. When the average scores the member in both groups were calculated in all parts of the test, the results are shown in Table 5.2.

When the quantitative data was analyzed, some important points appeared. The first important point is that the average scores of the participants of both groups in "*Requirement Analysis*" and "*Design*" parts are less than the participants' average scores

Table 5.1: Pre-Test Results of the Members in the Experimental Group

Students	Requirement Analysis	Design	Implementation	Test	Maintenance	Total
Student 1	4	0	10	14	8	36
Student 2	8	4	16	20	14	62
Student 3	2	0	12	14	4	32
Student 4	4	0	12	14	6	36
Student 5	8	4	18	20	14	64
Student 6	2	0	6	12	8	28
Student 7	0	0	6	8	6	20
Student 8	4	0	10	12	8	34
Student 9	0	0	6	6	4	16
Student 10	0	0	2	4	4	10
Student 11	2	0	14	12	10	38
Student 12	2	0	8	8	6	24
Student 13	6	0	10	8	8	32
Student 14	4	0	12	10	8	34
Student 15	2	0	10	8	6	26
Student 16	2	0	12	8	4	26

in the other parts. Thus, the participants have some trouble when they identify the requirements of the system and find the actors and the functionalities of the system. For the "*Implementation*" and "*Test*" parts, they already know the C# programming language so they did not face the problems they had in the first two phases of SDLC. For the last part, which is the "*Maintenance*" phase, the participants have showed an average success.

The second important point is that although the total average score of the participants

Table 5.2: Average Scores of Both Groups for All Parts in Pre-Test

Group	Requirement Analysis	Design	Implementation	Test	Maintenance	Total
Experimental Group	3.1	0.5	10.2	11.1	7.4	32.4
Control Group	2.9	0.1	9.6	10.2	7.9	30.6

Table 5.3: Pre-Test Results of the Members in the Control Group

Students	Requirement Analysis	Design	Implementation	Test	Maintenance	Total
Student 1	2	0	8	14	10	34
Student 2	4	0	8	10	8	30
Student 3	0	0	6	6	8	20
Student 4	6	0	10	8	12	36
Student 5	2	0	12	14	14	42
Student 6	2	0	10	8	10	30
Student 7	4	0	16	18	8	46
Student 8	4	0	14	12	10	40
Student 9	8	2	20	20	14	64
Student 10	0	0	6	4	4	14
Student 11	2	0	10	8	6	26
Student 12	2	0	6	8	4	20
Student 13	4	0	6	10	6	26
Student 14	2	0	8	4	2	16
Student 15	0	0	4	6	2	12
Student 16	4	0	10	12	8	34

in the experimental groups is a little bit higher than the total average score of the participants in the control group, the average scores of the participants in both groups are very close to each other in all parts of the exam. This means that the knowledge levels of the students in both groups about the software development process are similar at the beginning of the training program.

The last important point is the number of successful students in the pre-test. According to the grading policy (see Appendix G) applied at the courses given at the Middle East Technical University (METU), at least 60 points has to be collected to pass the course. When the results of the pre-test are analyzed by taking this criterion into consideration, there were 2 students who achieved the grade to pass this test in the experimental group as shown in Figure 5.1, and 1 student who achieved the grade to pass the pre-test in the control group as shown in Figure 5.2. In total, the number of students who could pass the test is 3 out of 32 students.

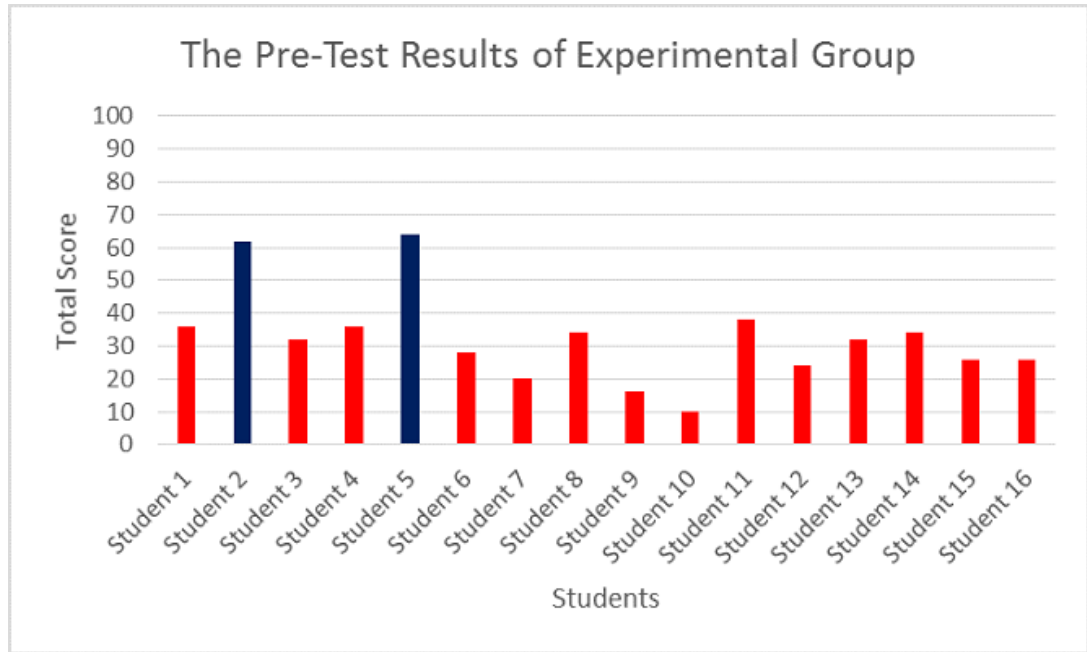


Figure 5.1: Successful Students in the Experimental Group in Pre-Test

5.2 Post-Test

At the end of the training period, a post-test was administered to the selected group of students in order to determine the participants' progress in their knowledge about software engineering. In this test, the students dealt with the problems related to the sample project as they did in the pre-test. Although the project scenario given in the post-test is different from the project scenario given in the pre-test, the structural content of the tests are the same. In other words, in the post-test, participants are expected to find solutions to the problems they encounter in all the software development processes related to a project scenario.

The project scenario in the post-test (see Appendix C) was also taken from the exam organized in the "*Software Engineering*" course offered at the Department of Computer Engineering, Çankaya University in order to provide consistency between the tests since the contents of the exams prepared of the "*Software Engineering*" course are periodically controlled by the Association for Evaluation and Accreditation of Engineering Programs (MUDEK). After determining the content of the post-test, it was organized on September 4, 2018. There were 32 students selected for the training program as experimental (16) and control groups (16). The post-test results of both

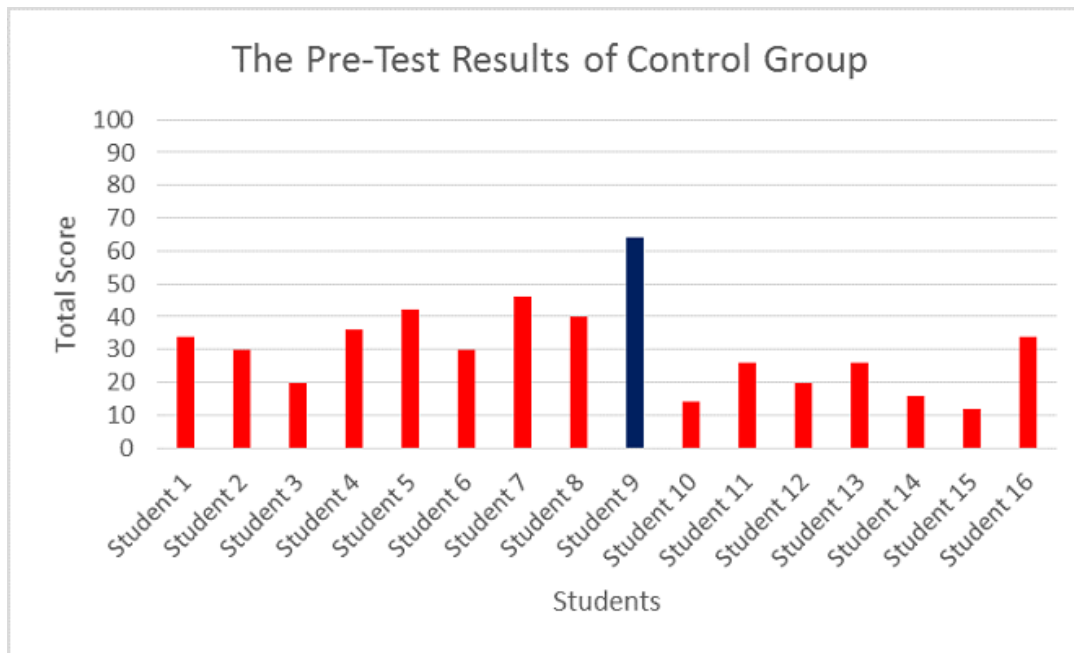


Figure 5.2: Successful Students in the Control Group in Pre-Test

groups are shown in Table 5.4 for the experimental group and Table 5.6 for the control group. The average scores the member in both groups were calculated in all parts of the test and the results are shown in Table 5.5.

When the post-test results are examined using the same process as in the pre-test, some important results arise. The first important point is that the average scores of the students in the "*Requirements Analysis*" and "*Design*" phases, which are missing in the pre-test, have significantly increased in the post-test. The grades of the students in the experimental group increased more than the grades of the students in the control group. In addition, the average scores in other parts of the post-test are higher than the scores obtained in the pre-test. This means that the training period was beneficial for the participants, especially the students who used the virtual office environment, as they have dramatically increased their abilities in software development process.

The second important point of this analysis appears when the results of the post-test are compared on a group basis. This examination illustrates that the average scores of the experimental group in all parts of the post-test, except "*Maintenance*" phase, are higher than the average scores of the control group. It means that the students in the experimental group had a more successful education period than the students in the

Table 5.4: Post-Test Results of the Members in the Experimental Group

Students	Requirement Analysis	Design	Implementation	Test	Maintenance	Total
Student 1	16	14	16	18	10	74
Student 2	12	16	18	20	12	78
Student 3	8	10	14	18	8	58
Student 4	10	12	18	16	8	64
Student 5	12	20	20	20	16	88
Student 6	10	8	6	14	8	46
Student 7	6	4	8	14	8	40
Student 8	10	12	8	12	8	50
Student 9	10	10	12	20	8	60
Student 10	6	2	8	12	4	32
Student 11	12	16	20	20	8	76
Student 12	8	10	12	12	6	48
Student 13	12	12	16	12	10	62
Student 14	8	10	14	14	6	52
Student 15	8	6	10	12	8	44
Student 16	8	6	12	10	6	42

control group.

The last important point is the number of successful students in the post-test. As was indicated in the previous section, the students have to get at least 60 points out of 100 in order to pass the course according to the grading policy of METU. When the students' numerical values are examined in the post-test, there were 7 students, who provided the minimum criteria that is required to be successful in the exam, in the experimental group out of 16 students as shown in Figure 5.3 and 4 students, who

Table 5.5: Average Scores of Both Groups for All Parts in Post-Test

Group	Requirement Analysis	Design	Implementation	Test	Maintenance	Total
Experimental Group	9.8	10.6	13.3	15.3	8.4	57.1
Control Group	8.5	6.4	10.6	12.1	10.8	48.4

Table 5.6: Post-Test Results of the Members in the Control Group

Students	Requirement Analysis	Design	Implementation	Test	Maintenance	Total
Student 1	12	12	12	14	12	62
Student 2	8	10	12	10	8	48
Student 3	6	6	10	12	10	44
Student 4	10	12	16	16	14	68
Student 5	10	4	12	16	16	58
Student 6	6	2	10	10	12	40
Student 7	10	10	14	20	12	66
Student 8	8	6	10	10	10	44
Student 9	16	12	18	18	16	80
Student 10	4	4	8	10	10	36
Student 11	6	2	6	8	12	34
Student 12	6	6	10	12	8	42
Student 13	8	2	8	8	8	34
Student 14	8	4	10	12	6	40
Student 15	6	6	6	8	8	34
Student 16	12	4	8	10	10	44

passed the test, in the control group out of 16 students as illustrated in Figure 5.4. In total, the number of students, who could pass the test, is 11 out of 32 students.

5.3 Comparison Between Pre-Test and Post-Test Results

When the results obtained from both pre-test and post-test are analyzed, it is easily observed that there are some differences between these results. The first difference is that the number of successful students in the post-test is higher than the number of successful students in the pre-test. In the pre-test, a total of 3 students, 2 from the experimental group and 1 from the control group, received the grade to pass the test. In the post-test, this number reached 11 students, including seven from the experimental group and four from the control group. These numerical values show that both training methods are beneficial to improve the level of knowledge of the participants since the number of successful students in the post-test for each group is increased with respect to the number of successful students in the pre-test. However,

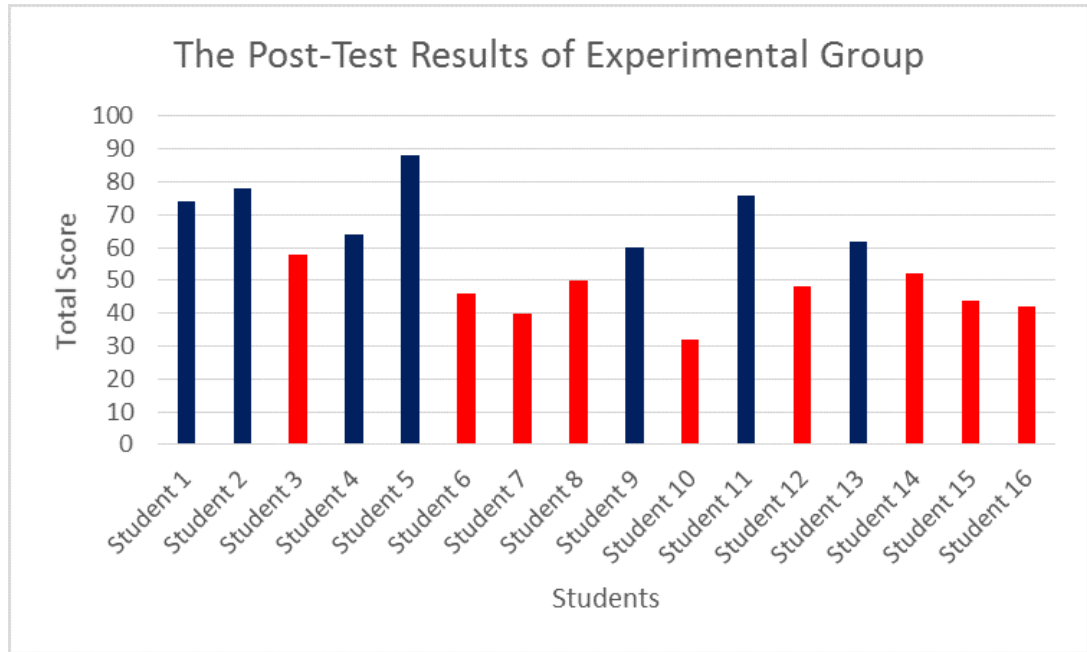


Figure 5.3: Successful Students in the Experimental Group in Post-Test

since the number of successful students in the experimental group is higher than the control group, it is seen that the virtual office environment is better than traditional methods. Another difference is that the average scores of the participants in all parts of the post-test is higher than the participants' average scores in all parts of the pre-test as shown in Figure 5.5. This observation also supports the idea that both training methods are useful for individuals. However, the important finding is that the average scores of the students in the experimental group are higher than the average scores of the students in the control group in all parts of the post-test except the "*Maintenance*" part. This means that the students, who were trained with the virtual office environment, have increased their knowledge levels more than the students, who have studied the software engineering concepts by using traditional methods. Hence, our training environment is more beneficial than the traditional methods.

In order to prove this claim statistically, *two sample t-test* have been used since the groups were randomly constituted. The definitions of the variables used in the formula of *two sample t-test* are listed as follows:

- μ_1 : the population mean of the differences between pre and post-test for the experimental group

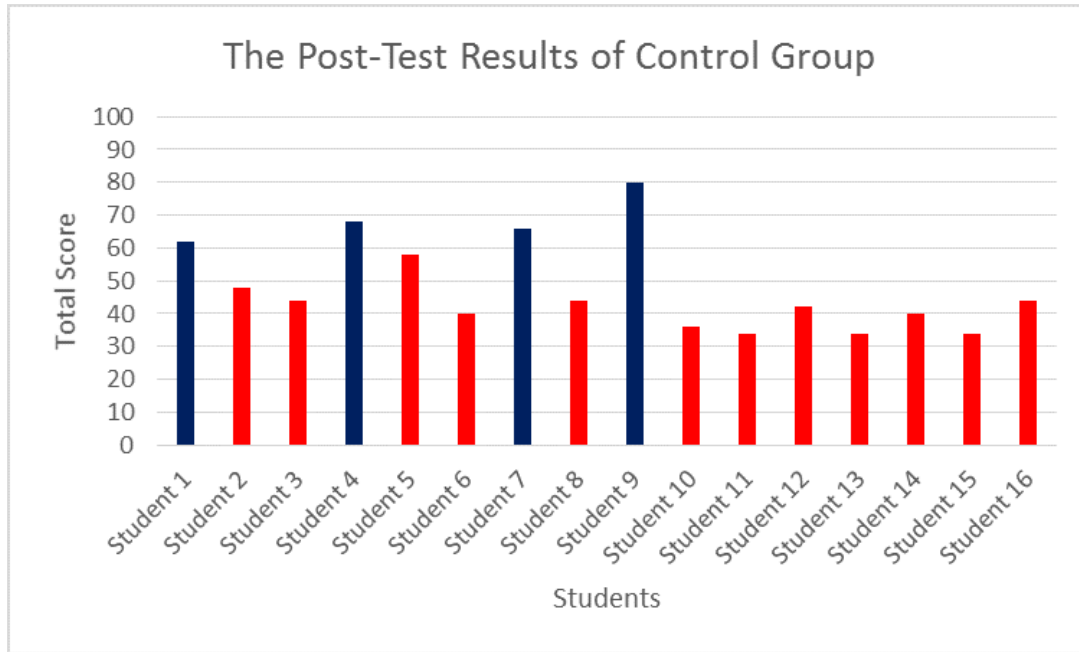


Figure 5.4: Successful Students in the Control Group in Post-Test

- μ_2 : the population mean of the differences between pre and post-test for the control group
- n_1 : the sample size of the experimental group
- n_2 : the sample size of the control group
- \bar{X}_1 : the sample mean of the differences between pre and post-test for the experimental group
- \bar{X}_2 : the sample mean of the differences between pre and post-test for the control group
- s_1^2 : the sample variance of the differences between pre and post-test for the experimental group
- s_2^2 : the sample variance of the differences between pre and post-test for the control group
- T: test statistic
- t : critical value
- p : probability value of differences

Two Sample T-Test Statistic Formula:

$$T = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2} \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}}$$

The hypothesis of the study should be identified before calculating the test statistic denoted by "T". This study states the null hypothesis, which is illustrated as " H_0 ", as the difference between the population's means of the experimental and control groups is equal to each other. Hence, the alternative hypothesis indicates that the population mean of the experimental group is greater than the population mean of the control group. The statistical representations of these hypothesizes are shown as follows:

$$H_0: \mu_1 = \mu_2$$

$$H_a: \mu_1 > \mu_2$$

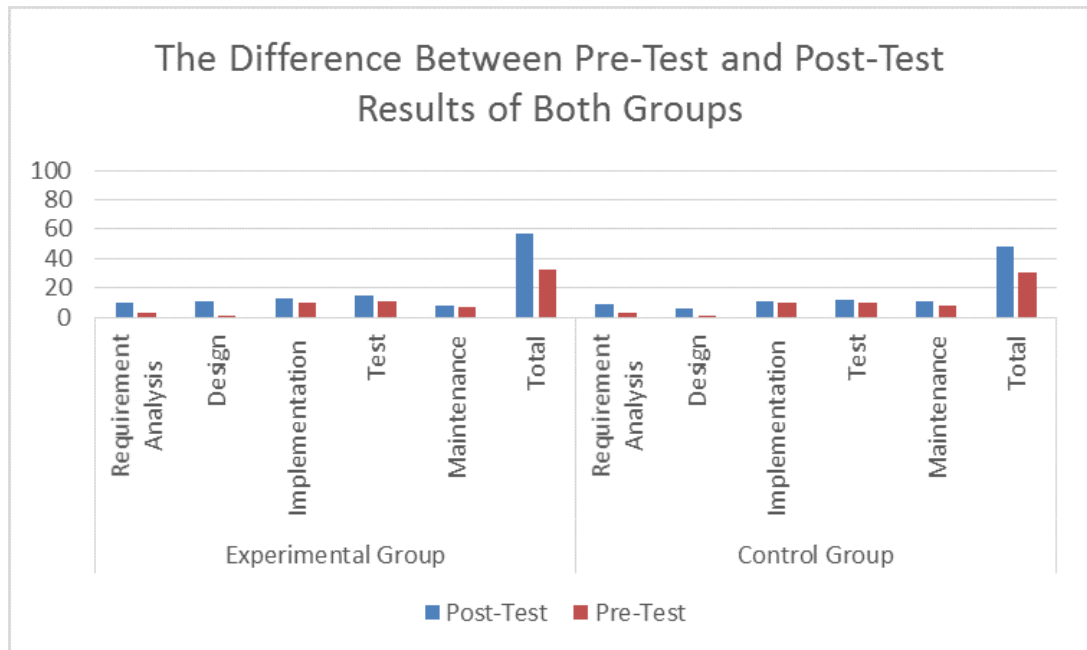


Figure 5.5: The Difference Between Pre-Test and Post-Test Results of Both Groups

The value of "T" should be calculated at first in order to decide which hypothesis will be accepted. To do this operation, the numerical data, which represents the differences between pre-test and post-test scores of both groups (see Table 5.7), have been entered to Minitab in order to figure out whether there is a significant difference between the groups.

Table 5.7: The Differences between Pre-Test and Post-Test Results

Experimental Group	Control Group
38	28
16	18
26	24
28	32
24	16
18	10
20	20
16	4
44	16
22	22
38	8
24	22
30	8
18	24
18	22
16	10

According to the results obtained from Minitab, when the value of significance level (α) was selected as 0.05 and assuming the variances of the groups as equal to each other, the value of test statistic (T) is 2.36 and " p " value is 0.03. For the critical value (t), t -Table (see Appendix D) plays a critical role to calculate t . Hence, according to t -Table, this value is equal to 2.04 when the number of population was selected as 30 ($n_1 + n_2 - 2$).

The numerical data obtained from the t -test provide important findings to prove whether the study was successful or not in a statistical way. There are two different ways to understand which of the above mentioned hypotheses will be accepted by using this numerical data. In the first way, the calculated " p " value, which is equal to 0.03, indicates that the null hypothesis has to be rejected since the value of " p " is smaller than 0.05. In such cases, the statistic states to reject the null hypothesis. In the second way, our test statistic value (T), which is calculated as 2.36, is greater than the critical value (t), which is equal to 2.04 with respect to the t -Table. In such cases, the statistic also states to reject the null hypothesis. As a result of this statistical analysis, we accept the alternative hypothesis by rejecting the null hypothesis within a 99.5%

confidence interval. This means that the population mean of the differences between pre-test and post-test of the experimental group is greater the population mean of the differences between pre-test and post-test of the control group. Thus, according to the statistical results, we can say that the members of the experimental group increase their knowledge levels more than the members of the control group.

5.4 PQ and ITQ

The sense of presence is one of the critical factors to establish successful virtual environments [111]. In order to provide this sense to the participants, the designed virtual environments should be close to the real environments. As a result, the virtual environments can be considered a successful tool. For this reason, it is important to measure the participants' level of sense of presence after using the virtual environment in order to detect whether the virtual environment is successful.

PQ and ITQ are the most popular questionnaires to measure both the participants' immersion levels and how much the individual tends to be immersed [112]. Hence, this study has benefited from both PQ (see Appendix E) and ITQ (see Appendix F). Before applying these questionnaires, items 23 and 24 in PQ, which are unrelated to our study, were extracted from the PQ since our system does not include any haptic mechanism, hence, it is not meaningful to measure these questions.

Before the training program, ITQ was organized to understand how much the participants tend to be immersed. In this questionnaire, the participants have answered 18 questions. According to the question content, they rated each question at a value between 1 and 7 to indicate how much the problem was appropriate for them. The highest score that can be obtained from this test is 126. After calculating the score of each participant, the results are obtained as shown in Figure 5.6.

When the scores illustrated in Figure 5.6 are evaluated over 100, there are 4 students who tend to be immersed by 90 percent and above; 1 student who tends to be immersed between 80 and 90 percent; 8 students who tend to be immersed between 70 and 80 percent; and 3 students who tend to be immersed below 70 percent. This means that the participants tend to be immersed at an average of 74.55 percent. Ac-

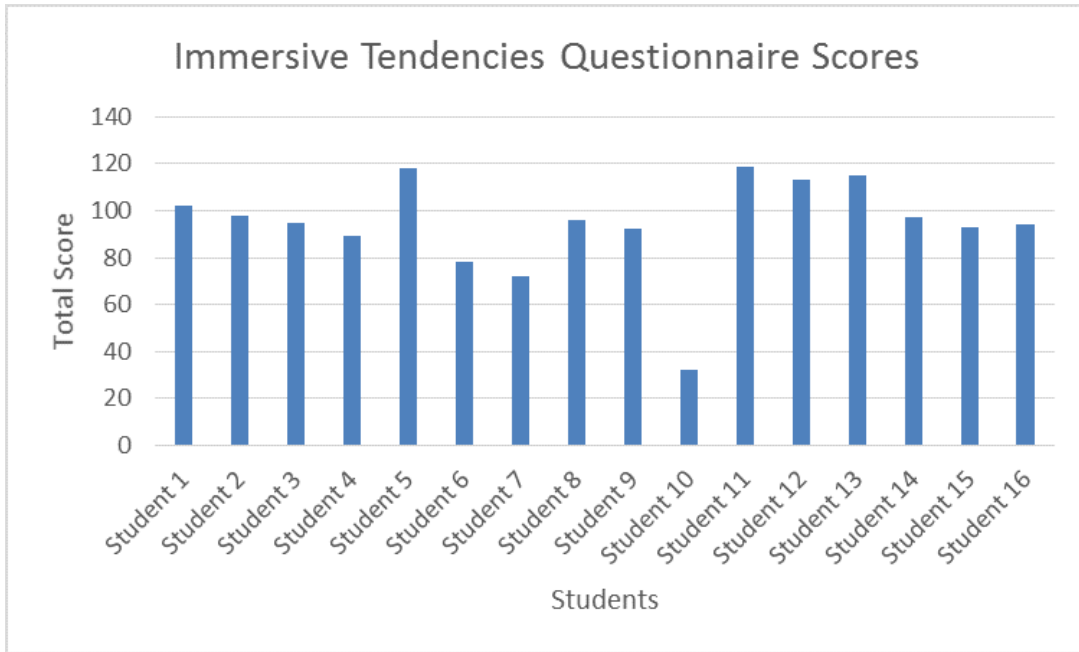


Figure 5.6: ITQ Scores of the Participants

tually, this result was expected because the participants' age range is between 20 and 22 so they can be easily affected by films, videos, games or any other technological tools since they grow up with technological devices.

During the training period, PQ was organized with the students after each VR session in order to measure their sense of presence. In this questionnaire the participants have answered 22 questions. According to the question content, they rated each question at a value between 1 and 7 to indicate their level of immersion based on the properties and functionalities of the virtual environment. The highest score that can be obtained from this test is 154. After calculating the score of each participant in each VR session, the results are obtained as shown in Figure 5.7.

When the scores of PQ shown in Figure 5.7 are evaluated over 100, the immersion levels of the participants can be listed as in Table 5.8.

When the values of PQ are examined, some important points are obtained from this analysis. First of all, all students except one have felt themselves at least 70 percent included in the virtual environment. Six students felt a strong immersion sense to our virtual office environment at 85 percent and above. The second important point is that although the PQ scores obtained from the last session is less than the PQ scores

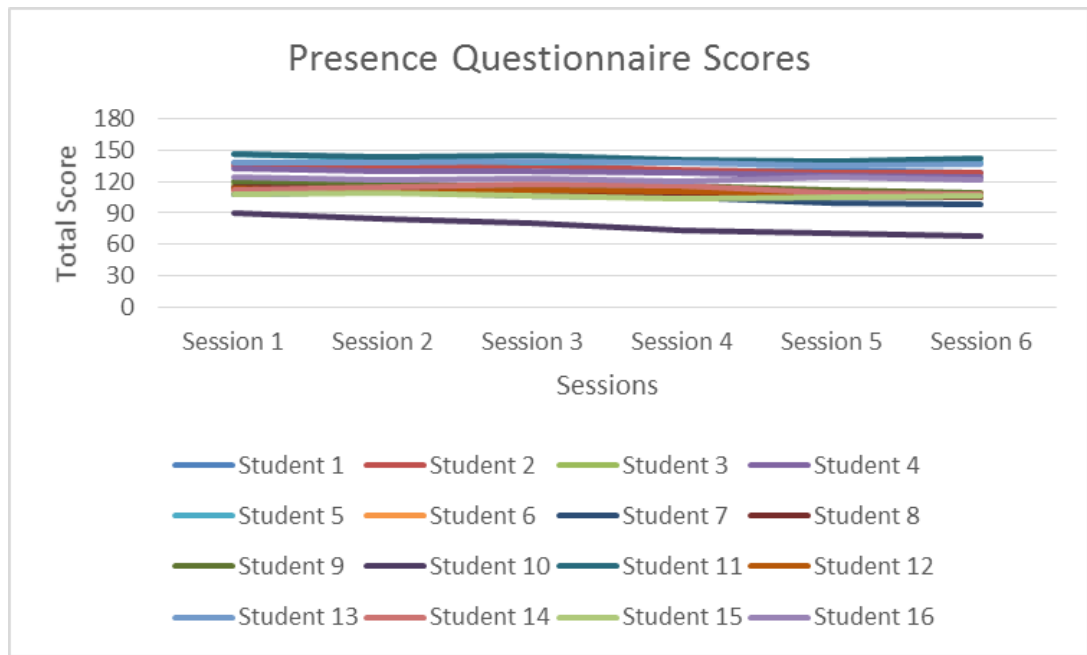


Figure 5.7: PQ Scores of the Participants

obtained from the first session, the difference between these scores are close to each other. This means that even when the system is used frequently by the participants, the effect of the system on them remains almost the same. Lastly, the average PQ score of the participants in whole training period is 77.06 percent. This score shows that the participants have likened the virtual office environment to the real office environment about 77.06 percent. As a conclusion, our virtual office environment has the ability to create an atmosphere similar to the real office environment atmosphere so that the participants can feel like they are in the real office environment. In this way the participants' level of knowledge and experience can be increased more since the sense of presence is the most important factor that shows the success of the virtual environments when they are used for educational purposes.

5.5 Validation Interviews

The quantitative part of the study was completed by administrating pre-test and post-test, and organizing PQ and ITQ with the participants. The numerical values obtained from these tests were also analyzed by applying *two sample t-test* in order to get valid results in terms of statistic. As was mentioned in the "*Test and Evaluation Methodol-*

Table 5.8: PQ Scores of the Participants over 100

Students	Session 1	Session 2	Session 3	Session 4	Session 5	Session 6
Student 1	89.61	86.36	87.66	84.42	85.06	83.77
Student 2	87.66	87.01	88.31	85.71	83.12	83.77
Student 3	76.62	74.68	74.03	72.73	70.13	68.18
Student 4	86.36	84.42	84.42	83.77	81.82	81.17
Student 5	94.81	92.21	90.26	90.91	89.61	90.26
Student 6	72.73	71.43	72.08	73.38	70.78	71.43
Student 7	70.13	72.08	68.83	68.18	64.94	63.64
Student 8	74.68	75.32	70.13	71.43	68.83	68.18
Student 9	78.57	77.27	74.03	75.32	72.73	71.43
Student 10	58.44	55.19	51.95	48.05	45.45	44.16
Student 11	95.45	93.51	94.16	91.56	90.91	92.21
Student 12	74.03	71.43	72.73	72.08	70.13	69.48
Student 13	89.61	90.26	90.91	89.61	88.31	88.96
Student 14	72.73	74.68	75.97	75.32	71.43	70.13
Student 15	70.13	70.78	68.83	67.53	68.18	69.48
Student 16	81.17	79.22	79.87	77.92	80.52	79.22

ogy" chapter, *"Mixed Research Methodology"* was used for the research methodology of the study, hence, it was necessary to complete the qualitative part of the study. To achieve this issue, a set of semi-structural interviews was organized with the lecturers in order to obtain their comments about the system. The summary of these interviews are as follows:

- *"The fact that the practical knowledge given in the course can be tested in an environment close to real life is a very positive feature in terms of the students' development."*
- *"A limited number of practical environments such as intern-ship and graduate projects is increasing via using this tool."*
- *"The students can increase their experience level by working in many different project scenarios."*
- *"As projects developed in the market create risks for individuals, there is pressure on people during project development. Such a platform that aims to edu-*

cate people without having this pressure will play a positive role in the development of individuals."

In addition to these comments, a set of semi-structural interviews was also organized with the participants in order to learn their opinions about the system. The summary of these interviews are as follows:

- *"That was an amazing experience for me. I am a student and i have worked as an intern last year. I was so shy to ask people what should I do who shall I talk with etc. Most probably I will experience the same emotions during the first month of my job. In my opinion these programs can help us get used to work. Sometimes it is hard to ask people so many questions. Thanks to this applications we can complete the tasks with no need to being annoying."*
- *"I have some problems in communicating with other students. I'm sure that the same situation will occur when I start to work. If the companies use this program in the orientation period by demonstrating the current workers and their duties, this will be so helpful for me since there is no need to ask anyone for help."*
- *"Before working in real life, I gained the confidence to experience the fields that I will work in the future. I know this is virtual, but I didn't feel that much when I was using it. I think that if we could move instead of using a keyboard to move in a virtual office environment, the realism of the system would increase."*
- *"Learning some things in real life can sometimes create unwanted results. This training platform has enabled us to gain experience about software projects without experiencing all these problems. It is a successful training tool because it provides opportunities to use this system frequently without the need for space for training and trainers. The warnings given about the errors are very important in terms of personal development."*

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

6.1 Discussion

This thesis study proposes a training environment to the novice software engineers in order to increase both their level of knowledge and experience in a similar environment to the real environment without having actual risks. For this purpose a virtual office environment was developed to teach the tasks occurred in the software development process to the individuals who will work in this area. In addition, a desktop application was also developed in order to make the virtual office environment more dynamic since it enables system administrators to create different software project scenarios. However, before developing these tools a literature review was conducted to develop the background of the study. Firstly, it was begun to give the definitions of software, software development and software engineering. After defining these concepts, the reasons why it is necessary to develop the software projects within a certain logical framework were explained to show the importance of the software development methodologies. Then, it was explained why there are several different software development methodologies in the literature and both the advantages and disadvantages of the existing software development methodologies were illustrated. After that, although there exist several different software development methodologies to produce successful software products, it was stated that most of the software projects are finished by not providing the requirements of the customers. Hence, it continued to illustrate the negative impacts of the unsuccessful software projects on the economy of both the governments and companies. After finding this problem, the reasons that cause the software projects to fail were researched and it was found that

the most important reason is the lack of knowledge and experience of the individuals, who work in this area. In the last stage of the literature survey, the studies, which aim to teach software engineering concepts to individuals, were analyzed in detail.

After analyzing the studies in the literature, it was determined that there are 3 main missing points of the existing studies. These are:

- **Low/Limited Increased Reality**
- **Missing Whole Project Development Processes**
- **Limited Number of Stories**

As was mentioned before, the aim of this thesis is to develop a training tool for individuals about the software development process by eliminating the missing points of the existing studies in the literature. To achieve this aim, 2 complementary applications were developed in the scope of this dissertation. These are: **Scenario Generator** and **Virtual Office Environment**.

The below items describe how the programs developed in this study solved the missing points of the studies in the literature.

- **Low/Limited Increased Reality:** This missing point was solved by the "*Virtual Office Environment*". The 3D virtual office environment with a head-mounted display produces an interactive training environment which is similar to the real office environment. The participants can interact with the workers, who are NPCs working in the same company, in order to learn their missions in the software project. In addition, the participants can also interact with the elements in the virtual office environment such as opening and reading the project definition document. In this way the participants have the opportunity to test and develop themselves in a virtual office environment, which is designed to be closest to the real office environment, by dealing with project development problems. The effect of the system on the participants was measured by PQ and ITQ frequently used questionnaires in the literature.
- **Missing Whole Project Development Processes:** This missing point was solved

by "Virtual Office Environment" and "Scenario Generator". "Scenario Generator" program has the ability to produce an XML file, which contains the missions related to the each phase of SDLC, by constructing an ontological meta-language based on its own unique tag-structure. Thanks to this program, which is used by the project manager or the team leader, the participants have the chance to be involved in every phase of the software development process. "Virtual Office Environment" parses the XML file produced by "Scenario Generator" and makes the scenario written in the XML file come alive. Hence, the participants can experience the project scenario generated by the authorized person of the company via the virtual office environment.

- **Limited Number of Stories:** This missing point was solved by "Scenario Generator". This program has the ability to produce an XML file, which contains the missions related to each phase of SDLC. Hence, the project managers can easily produce several different project scenarios by using this program.

6.2 Validation of the Proposed Training Environment

The potential threats of this study, which may affect the validity of the study, were identified in Chapter 4. In order to deal with these threats, some procedures were performed as follows:

- **Construct Validity \Rightarrow Qualitative Measure:** A set of semi structural interviews was organized by both the lecturers and the students.
- **Construct Validity \Rightarrow Quantitative Measure:** A pre-test and a post-test were administered with the students to determine their progress levels during the training period. In addition, PQ and ITQ were also applied to understand the success of the designed VR environment.
- **Internal Validity \Rightarrow History Effect:** A pre-test was administered with the students for both groups at the beginning of the study in order to determine their knowledge levels before starting the training period. Therefore, the history effect was eliminated since their knowledge levels were almost the same at the beginning of the study.

- **Internal Validity \Rightarrow Testing Effect:** In order to eliminate this threat, the first three students in both tests were given a gift. Therefore, the students tried to give correct answers to the questions in order to be among the first three students in the tests. In addition to this procedure, for determining the motivation of the students in the control group, they were asked how many hours they spent to study software engineering topics in a week. They have indicated that they have studied the software engineering concepts for an average of 78 minutes per week. Hence, these students spent as much time as the students in the experimental group to study software engineering subjects.
- **Internal Validity \Rightarrow Instrumentation Effect:** refers any change in the system during the training period. The designed system did not change during the training period.
- **External Validity \Rightarrow A Conceptual Replication:** This study is based on the literature that has been rigorously reviewed to assess potential needs of a virtual training platform. As a result, some important and novel features were identified. After conducting this exploration, we have added a set of features to our training platform to develop more efficient and immersive training environment for the individuals. Therefore, a novel platform was designed to teach the tasks related to software development process.
- **Reliability:** The students have tested the training platform. During the training period, they have participated the necessary tests that show the success of the designed platform. In order to address the reliability issues of the study, valid statistical tests applied to the results obtained from these tests. In addition, a set of semi-structural interviews was also conducted with the lecturers to obtain their thoughts about our system as an expert point of view.

6.3 Limitations

Although the potential threats, which may affect the results of this study in a negative manner, were eliminated, one of them cannot be eliminated since it depends on the declaration of individuals. Hence, this threat may limit the validity of the study. It

can be listed as:

- The participants in the experimental group may also study the software engineering concepts from other resources.

To deal with this threat, the students in the experimental group were warned informed several times not to review the software engineering topics from other resources. They have declared confirmed that they did not study these concepts from other resources, however they may have studied.

6.4 Revisiting the Research Questions

In this section, the research questions from Chapter 1 are discussed in the light of the results obtained from the user experiences. The aim of this study is to increase the experience and knowledge levels of novice software engineers about the tasks related to software development process by designing a 3D virtual office environment. By using the potentials of a virtual office, a task-based software engineering training can be conducted independently from customers and instructors outside traditional office environments. By taking into account of this aim, this study has a total of 2 research questions:

RQ 1: *Can the proposed training environment increase performance of students on software engineering tasks (e.g. requirement capturing, coding, testing, etc.)?*

In order to response to the first research question, software engineering discipline requires self directed training where VR-based environment is an opportunity to get trained regarding best practices without having hands-on experience from the field. We found that using VR simulation for training of software engineering tasks significantly improves skills of software engineering trainees. This confirms that VR simulation training is a complementary tool to conventional training. In order to validate this claim, the proposed system was tested with the students. The students were randomly divided into two groups, which are control and experimental groups. The students in the control group studied the software engineering concepts from traditional resources while the students in the experimental group used our training

platform for these topics. In order to understand the students' progress, a pre-test and a post-test were administered. The results of these tests indicated that the students in the experimental group have increased their knowledge levels more than the students in the control group.

In addition, the students cannot find the opportunity to take place in the real-life projects frequently. However, the students had an opportunity to practice the tasks occurred in different types of software projects by using this training platform since this platform has the ability to produce and animate unlimited number of project scenarios. In addition, artificial intelligent NPCs also help the students about the tasks of the software development process. Therefore, the students could repeat the concepts in the field of software engineering without having real-life constraints. As a result, this environment provided an accelerated learning mechanism for them and their knowledge levels were increased. In addition, the lecturers also confirmed that students are able to transfer the tacit knowledge to explicit knowledge during the interviews.

In such training environments, the realism of the virtual environments is one of the most important aspects that highlights the success of designed environments. In general, this realism is directly proportional to the feeling of being there. For this reason, it is necessary to measure the sense of presence of the participants. To achieve this purpose, PQ was administered with the students in the experimental group. According to the results obtained from PQ, the students have felt a strong immersion. This means that our 3D virtual office environment is similar to real office environment. Hence, the developed system is evaluated as successful.

RQ 2: *Can the proposed training environment motivate the students for exercising the tasks related to the each phase of SDLC?*

Motivation plays a crucial role for the success of the individuals on a topic [113]. Hence, training environments should motivate the participants. In this study, the students have used our virtual office environment six times. After each VR session, PQ was administered. In here, the important point is that the difference between the score of the first PQ and the score of the last PQ is almost equal for each student in the experimental group. This shows that our training platform has the ability to motivate

the students since both their PQ scores did not decrease and their knowledge levels increased.

6.5 Conclusion

As can be seen, our training framework tried to eliminate the missing points of the existing studies in the literature by gathering all properties into one environment. Different testing methods were organized in order to learn whether this training platform achieves this aim. For this purpose, 32 students, who are currently studying at the computer engineering department, have tested our training platform. They were randomly divided into 2 groups which are experimental and control groups. The members in the control group could not use our training platform for the software engineering topics. They were only allowed to use traditional resources such as books, videos or presentations. The members in the experimental group were only allowed to join our training environment to study defined software engineering problems. It was strongly stated to them that they are not allowed to use any other resources to study these subjects. A pre-test at the beginning of the training period and a post-test at the end of the training period were administered in order to figure out the effects of both training strategies. The results indicate that there is no significant difference between the knowledge levels of both groups at the beginning of the program. At the end of the training program, the knowledge levels of the program have dramatically increased for both groups. However, the members in the experimental group have increased their knowledge levels more than the members in the control group. This means that our training platform provides a beneficial educational tool to teach the software development process to individuals who will work in this area. To support this idea, a set of semi-structural interviews was also organized with the participants in order to learn their opinions about our training framework. According to the results obtained from these interviews, the participants have evaluated our platform as a valuable training tool.

The success of our training platform was also tested by applying PQ and ITQ with the participants to measure their sense of presence levels since presence is the most important factor that shows the success of the virtual environments. After each VR

session, PQ was administered to the participants to observe their sense of presence levels. These measurements specify that most of the participants have a strong "*presence*" feeling for our virtual office environment. When the average of all participants' scores in six sessions is taken, a success rate of 77 percent is observed. This means that our virtual office environment has the ability to create real office environment atmosphere 77% success. As a conclusion, the findings of this thesis demonstrate that our virtual reality training platform can be efficiently used in the training of participants about the tasks occurred in the software development process.

6.6 Future Work

For the future works of this study, it is planned to increase the AI mechanism of the NPCs in the virtual office environment by using deep learning algorithms. Hence, the NPCs may produce their own dialogues to communicate with the participant about the development process of the project or to automatically give duties of the project to the participant without reading the XML file. In this way, the work load of the project managers, who create the project scenario, may be decreased. In addition, an on-line module in which it allows the multi-player concept is also planned to integrate to the system. Hence, the participants have a chance to develop the projects with both the real individuals and NPCs. Finally, text-to-speech property will be also included into the system as a functionality of the participants. In this way, the participants can use the system more dynamically since they can dynamically enter the speeches and the NPCs have ability to understand these speeches. This would increase the sense of presence of the participants.

REFERENCES

- [1] P. Suber, “What is software?,” *The Journal of Speculative Philosophy*, pp. 89–119, 1988.
- [2] L. J. Osterweil, “What is software?,” *Automated Software Engineering*, vol. 15, no. 3-4, pp. 261–273, 2008.
- [3] E. J. Braude and M. E. Bernstein, *Software engineering: modern approaches*. Waveland Press, 2016.
- [4] A. Oram and G. Wilson, *Making software: What really works, and why we believe it*. " O'Reilly Media, Inc.", 2010.
- [5] S. Jayaram, H. I. Connacher, and K. W. Lyons, “Virtual assembly using virtual reality techniques,” *Computer-aided design*, vol. 29, no. 8, pp. 575–584, 1997.
- [6] I. Heldal, “Supporting participation in planning new roads by using virtual reality systems,” *Virtual Reality*, vol. 11, no. 2-3, pp. 145–159, 2007.
- [7] A. Z. Sampaio and O. P. Martins, “The application of virtual reality technology in the construction of bridge: The cantilever and incremental launching methods,” *Automation in construction*, vol. 37, pp. 58–67, 2014.
- [8] L.-K. Cheng, M.-H. Chieng, and W.-H. Chieng, “Measuring virtual experience in a three-dimensional virtual reality interactive simulator environment: a structural equation modeling approach,” *Virtual Reality*, vol. 18, no. 3, pp. 173–188, 2014.
- [9] Z. Merchant, E. T. Goetz, L. Cifuentes, W. Keeney-Kennicutt, and T. J. Davis, “Effectiveness of virtual reality-based instruction on students’ learning outcomes in k-12 and higher education: A meta-analysis,” *Computers & Education*, vol. 70, pp. 29–40, 2014.
- [10] R. Schroeder, *Possible worlds: the social dynamic of virtual reality technology*. Westview Press, Inc., 1996.
- [11] G. Lorenzo, A. Lledó, J. Pomares, and R. Roig, “Design and application of an immersive virtual reality system to enhance emotional skills for children with autism spectrum disorders,” *Computers & Education*, vol. 98, pp. 192–205, 2016.

- [12] L. Donath, R. Rössler, and O. Faude, “Effects of virtual reality training (exergaming) compared to alternative exercise training and passive control on standing balance and functional mobility in healthy community-dwelling seniors: a meta-analytical review,” *Sports medicine*, vol. 46, no. 9, pp. 1293–1309, 2016.
- [13] W. Peñate Castro, M. J. Roca Sanchez, C. T. Pitti González, J. M. Bethencourt, J. A. de la Fuente Portero, and R. Gracia Marco, “Cognitive-behavioral treatment and antidepressants combined with virtual reality exposure for patients with chronic agoraphobia,” *International Journal of Clinical and Health Psychology*, vol. 14, no. 1, 2014.
- [14] E. Yiannakopoulou, N. Nikiteas, D. Perrea, and C. Tsigris, “Virtual reality simulators and training in laparoscopic surgery,” *International Journal of Surgery*, vol. 13, pp. 60–64, 2015.
- [15] J. Gregory, *Virtual reality*. Cherry Lake, 2017.
- [16] E. Pantano, “Innovation drivers in retail industry,” *International Journal of Information Management*, vol. 34, no. 3, pp. 344–350, 2014.
- [17] W. S. Humphrey, “The software engineering process: definition and scope,” *ACM SIGSOFT Software Engineering Notes*, vol. 14, no. 4, pp. 82–83, 1989.
- [18] S. McConnell, “Who needs software engineering?,” *IEEE Software*, vol. 18, no. 1, pp. 5–8, 2001.
- [19] G. M. Weinberg, *The psychology of computer programming*, vol. 932633420. Van Nostrand Reinhold New York, 1971.
- [20] O. Salo and P. Abrahamsson, “Empirical evaluation of agile software development: The controlled case study approach,” *Product Focused Software Process Improvement*, pp. 408–423, 2004.
- [21] I. A. Zualkernan and W.-T. Tsai, “Are knowledge representations the answer to requirement analysis?,” in *Computer Languages, 1988. Proceedings., International Conference on*, pp. 437–443, IEEE, 1988.
- [22] R. Thackeray and G. V. Treeck, “Applying quality function deployment for software product development,” *Journal of Engineering Design*, vol. 1, no. 4, pp. 389–410, 1990.
- [23] C. S. Pereira and A. L. Soares, “Improving the quality of collaboration requirements for information management through social networks analysis,” *International Journal of Information Management*, vol. 27, no. 2, pp. 86–103, 2007.

- [24] K. Kannan *et al.*, “An approach for decomposing requirements into analysis pattern using problem frames (drap-pf),” in *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pp. 2392–2396, IEEE, 2015.
- [25] V. Seidita, M. Cossentino, and S. Gaglio, “Using and extending the spem specifications to represent agent oriented methodologies,” in *International Workshop on Agent-Oriented Software Engineering*, pp. 46–59, Springer, 2008.
- [26] P. S. Sajja, *Essence of Systems Analysis and Design: A Workbook Approach*. Springer, 2017.
- [27] J. Karimi, *Computer aided process organization in software design*. PhD thesis, The University of Arizona., 1983.
- [28] T. G. Grbac, Ž. Car, and M. Vuković, “Requirements and architecture modeling in software engineering courses,” in *Proceedings of the 2015 European Conference on Software Architecture Workshops*, p. 36, ACM, 2015.
- [29] F. Alonso, J. Fuertes, C. Montes, and R. Navajo, “A quality model: How to improve the object-oriented software process,” in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, vol. 5, pp. 4884–4889, IEEE, 1998.
- [30] S. A. Dart, R. J. Ellison, P. H. Feiler, and A. N. Habermann, “Software development environments,” *Computer*, vol. 20, no. 11, pp. 18–28, 1987.
- [31] B. Meyer, “Seven principles of software testing,” *Computer*, vol. 41, no. 8, pp. 99–101, 2008.
- [32] I. Burnstein, T. Suwanassart, and R. Carlson, “Developing a testing maturity model for software test process evaluation and improvement,” in *Test Conference, 1996. Proceedings., International*, pp. 581–589, IEEE, 1996.
- [33] J. Uddin, R. Ghazali, M. M. Deris, R. Naseem, and H. Shah, “A survey on bug prioritization,” *Artificial Intelligence Review*, vol. 47, no. 2, pp. 145–180, 2017.
- [34] P. Ralph and P. Kelly, “The dimensions of software engineering success,” in *Proceedings of the 36th International Conference on Software Engineering*, pp. 24–35, ACM, 2014.
- [35] T. Mens, “Introduction and roadmap: History and challenges of software evolution,” in *Software evolution*, pp. 1–11, Springer, 2008.
- [36] Y. Fu, M. Yan, X. Zhang, L. Xu, D. Yang, and J. D. Kymer, “Automated classification of software change messages by semi-supervised latent dirichlet allocation,” *Information and Software Technology*, vol. 57, pp. 369–377, 2015.

- [37] E. B. Swanson, "The dimensions of maintenance," in *Proceedings of the 2nd international conference on Software engineering*, pp. 492–497, IEEE Computer Society Press, 1976.
- [38] N. F. Schneidewind, "The state of software maintenance," *IEEE Transactions on Software Engineering*, vol. 3, pp. 303–310, 1987.
- [39] A. April and A. Abran, *Software maintenance management: evaluation and continuous improvement*, vol. 67. John Wiley & Sons, 2012.
- [40] S. McConnell, "The art, science, and engineering of software development," *IEEE Software*, vol. 15, no. 1, pp. 120–118, 1998.
- [41] S. Planning, "The economic impacts of inadequate infrastructure for software testing," *National Institute of Standards and Technology*, 2002.
- [42] A. Kiral and T. Ayyildiz Ercelebi, "Comparison of software quality metrics: A case study," in *Proceedings of 12th Turkish National Software Engineering Symposium (UYMS)*, 2018.
- [43] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of software engineering*. Prentice Hall PTR, 2002.
- [44] N. D. Singpurwalla and S. P. Wilson, *Statistical methods in software engineering: reliability and risk*. Springer Science & Business Media, 2012.
- [45] B. A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in *Proceedings of the 26th international conference on software engineering*, pp. 273–281, IEEE Computer Society, 2004.
- [46] M. Jørgensen, "Software quality measurement," *Advances in engineering software*, vol. 30, no. 12, pp. 907–912, 1999.
- [47] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in software quality. volume i. concepts and definitions of software quality," tech. rep., DTIC Document, 1977.
- [48] B. W. Boehm, *Characteristics of software quality*, vol. 1. North-Holland, 1978.
- [49] R. B. Grady, *Practical software metrics for project management and process improvement*. Prentice-Hall, Inc., 1992.
- [50] R. G. Dromey, "Cornering the chimera," *IEEE Software*, vol. 13, no. 1, p. 33, 1996.
- [51] W. W. Royce, "Managing the development of large software systems," in *proceedings of IEEE WESCON*, vol. 26, pp. 328–338, Los Angeles, 1970.
- [52] I. Sommerville, *Software Engineering*. Addison-Wesley, 2010.

- [53] R. Victor, "Iterative and incremental development: A brief history," *IEEE Computer Society*, pp. 47–56, 2003.
- [54] B. Boehm, "A spiral model of software development and enhancement," *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, 1986.
- [55] G. Coleman and R. Verbruggen, "A quality software process for rapid application development," *Software Quality Journal*, vol. 7, no. 2, pp. 107–122, 1998.
- [56] C. Schmittner, Z. Ma, and E. Schoitsch, "Combined safety and security development lifecycle," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pp. 1408–1415, IEEE, 2015.
- [57] R. Thakurta and F. Ahlemann, "Understanding requirements volatility in software projects-an empirical investigation of volatility awareness, management approaches and their applicability," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pp. 1–10, IEEE, 2010.
- [58] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro, "Automated error analysis for the agilization of feature modeling," *Journal of Systems and Software*, vol. 81, no. 6, pp. 883–896, 2008.
- [59] H. Takeuchi and I. Nonaka, "The new new product development game," *Harvard business review*, vol. 64, no. 1, pp. 137–146, 1986.
- [60] D. Anderson, "The principles of the kanban method," *David J. Anderson & Associates*, 2010.
- [61] K. Beck, *Extreme programming explained: embrace change*. addison-wesley professional, 2000.
- [62] G. Kannabiran and K. Sankaran, "Determinants of software quality in offshore development—an empirical study of an indian vendor," *Information and Software Technology*, vol. 53, no. 11, pp. 1199–1208, 2011.
- [63] M. Huisman and J. Iivari, "Deployment of systems development methodologies: Perceptual congruence between is managers and systems developers," *Information & Management*, vol. 43, no. 1, pp. 29–49, 2006.
- [64] K. Petersen and C. Wohlin, "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case," *Journal of systems and software*, vol. 82, no. 9, pp. 1479–1490, 2009.
- [65] A. Saxena and P. Upadhyay, "Waterfall vs. prototype: Comparative study of sdlc," *Imperial Journal of Interdisciplinary Research*, vol. 2, no. 6, 2016.
- [66] M. L. Drury-Grogan and D. M. Kennedy, "Highlighting communication activities and inefficiencies between agile vs. waterfall methods: An agent based

- model of knowledge sharing,” in *8th Pre-ICIS International Research Workshop on Information Technology Project Management (IRWITPM 2013)*, p. 46, 2013.
- [67] A. Moniruzzaman and D. S. A. Hossain, “Comparative study on agile software development methodologies,” *arXiv preprint arXiv:1307.3356*, 2013.
 - [68] S. Nerur, R. Mahapatra, and G. Mangalaraj, “Challenges of migrating to agile methodologies,” *Communications of the ACM*, vol. 48, no. 5, pp. 72–78, 2005.
 - [69] T. Dyba and T. Dingsoyr, “What do we know about agile software development?,” *IEEE software*, vol. 26, no. 5, pp. 6–9, 2009.
 - [70] A. Mishra and D. Dubey, “A comparative study of different software development life cycle models in different scenarios,” *International Journal of Advance Research in Computer Science and Management Studies*, vol. 1, no. 5, pp. 64–69, 2013.
 - [71] L. Van Velsen, J. Wentzel, and J. E. Van Gemert-Pijnen, “Designing ehealth that matters via a multidisciplinary requirements development approach,” *JMIR research protocols*, vol. 2, no. 1, 2013.
 - [72] B. Losada, M. Urretavizcaya, and I. Fernández-Castro, “A guide to agile development of interactive software with a "user objectives"-driven methodology,” *Science of Computer Programming*, vol. 78, no. 11, pp. 2268–2281, 2013.
 - [73] S. Blomkvist, “Towards a model for bridging agile development and user-centered design,” *Human-centered software engineering-integrating usability in the software development lifecycle*, pp. 219–244, 2005.
 - [74] A. P. Costa, L. P. Reis, and M. J. Loureiro, “Hybrid user centered development methodology: An application to educational software development,” in *International Conference on Web-Based Learning*, pp. 243–253, Springer, 2014.
 - [75] B. Losada, M. Urretavizcaya, J.-M. López-Gil, and I. Fernández-Castro, “Combining intermod agile methodology with usability engineering in a mobile application development,” in *Proceedings of the 13th International Conference on Interacción Persona-Ordenador*, p. 39, ACM, 2012.
 - [76] M. Ceschi, A. Sillitti, G. Succi, and S. De Panfilis, “Project management in plan-based and agile companies,” *IEEE software*, vol. 22, no. 3, pp. 21–27, 2005.
 - [77] J. Kollmann, H. Sharp, and A. Blandford, “The importance of identity and vision to user experience designers on agile projects,” in *Agile Conference, 2009. AGILE’09.*, pp. 11–18, IEEE, 2009.
 - [78] F. Tsui, O. Karam, and B. Bernal, *Essentials of software engineering*. Jones & Bartlett Learning, 2016.

- [79] O. Sohaib and K. Khan, "Integrating usability engineering and agile software development: A literature review," in *Computer design and applications (IC-CDA), 2010 international conference on*, vol. 2, pp. V2–32, IEEE, 2010.
- [80] Y. Fang and J. Teizer, "A multi-user virtual 3d training environment to advance collaboration among crane operator and ground personnel in blind lifts," in *Computing in Civil and Building Engineering (2014)*, pp. 2071–2078, American Society of Civil Engineers, 2014.
- [81] J. P. Bliss, P. D. Tidwell, and M. A. Guest, "The effectiveness of virtual reality for administering spatial navigation training to firefighters," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 1, pp. 73–86, 1997.
- [82] N. E. Seymour, A. G. Gallagher, S. A. Roman, M. K. O'Brien, V. K. Bansal, D. K. Andersen, and R. M. Satava, "Virtual reality training improves operating room performance: results of a randomized, double-blinded study," *Annals of surgery*, vol. 236, no. 4, pp. 458–464, 2002.
- [83] M. R. Kandalaft, N. Didehbani, D. C. Krawczyk, T. T. Allen, and S. B. Chapman, "Virtual reality social cognition training for young adults with high-functioning autism," *Journal of autism and developmental disorders*, vol. 43, no. 1, pp. 34–44, 2013.
- [84] R. Elledge, S. McAleer, M. Thakar, F. Begum, S. Singhota, and N. Grew, "Use of a virtual learning environment for training in maxillofacial emergencies: impact on the knowledge and attitudes of staff in accident and emergency departments," *British Journal of Oral and Maxillofacial Surgery*, vol. 54, no. 2, pp. 166–169, 2016.
- [85] A. Baker, E. O. Navarro, and A. Van Der Hoek, "An experimental card game for teaching software engineering processes," *Journal of Systems and Software*, vol. 75, no. 1, pp. 3–16, 2005.
- [86] A. Bollin, E. Hochmüller, and R. T. Mittermeir, "Teaching software project management using simulations," in *Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on*, pp. 81–90, IEEE, 2011.
- [87] T. Hainey, T. M. Connolly, M. Stansfield, and E. A. Boyle, "Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level," *Computers & Education*, vol. 56, no. 1, pp. 21–35, 2011.
- [88] A. Rusu, R. Russell, E. Burns, and A. Fabian, "Employing software maintenance techniques via a tower-defense serious computer game," *Edutainment Technologies. Educational Games and Virtual Reality/Augmented Reality Applications*, pp. 176–184, 2011.

- [89] U. Aydan, M. Yilmaz, P. M. Clarke, and R. V. O'Connor, "Teaching iso/iec 12207 software lifecycle processes: A serious game approach," *Computer Standards & Interfaces*, vol. 54, pp. 129–138, 2017.
- [90] E. Ye, C. Liu, and J. A. Polack-Wahl, "Enhancing software engineering education using teaching aids in 3-d online virtual worlds," in *Frontiers in education conference-global engineering: knowledge without borders, opportunities without passports, 2007. FIE'07. 37th annual*, pp. T1E–8, IEEE, 2007.
- [91] G. Rodriguez, Á. Soria, and M. Campo, "Virtual scrum: A teaching aid to introduce undergraduate software engineering students to scrum," *Computer Applications in Engineering Education*, vol. 23, no. 1, pp. 147–156, 2015.
- [92] D. Parsons and R. Stockdale, "Cloud as context: Virtual world learning with open wonderland," in *Proceedings of the 9th World Conference on Mobile and Contextual Learning, Malta*, pp. 123–130, 2010.
- [93] J. Psotka, "Immersive training systems: Virtual reality and education and training," *Instructional science*, vol. 23, no. 5, pp. 405–431, 1995.
- [94] B. Çiflikli, V. İşler, and U. Güdükbay, "Increasing the sense of presence in a simulation environment using image generators based on visual attention," *Presence: Teleoperators and Virtual Environments*, vol. 19, no. 6, pp. 557–568, 2010.
- [95] Q. Chen, J. Grundy, and J. Hosking, "An e-whiteboard application to support early design-stage sketching of uml diagrams," in *Human Centric Computing Languages and Environments, 2003. Proceedings. 2003 IEEE Symposium on*, pp. 219–226, IEEE, 2003.
- [96] K. Goseva-Popstojanova, A. Hassan, A. Guedem, W. Abdelmoez, D. E. M. Nassar, H. Ammar, and A. Mili, "Architectural-level risk analysis using uml," *IEEE transactions on software engineering*, vol. 29, no. 10, pp. 946–960, 2003.
- [97] A. M. Fernández-Sáez, M. R. Chaudron, and M. Genero, "An industrial case study on the use of uml in software maintenance and its perceived benefits and hurdles," *Empirical Software Engineering*, pp. 1–65, 2018.
- [98] A. Bosu, J. C. Carver, C. Bird, J. Orbeck, and C. Chockley, "Process aspects and social dynamics of contemporary code review: insights from open source development and industrial practice at microsoft," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 56–75, 2017.
- [99] M. B. Zanjani, H. Kagdi, and C. Bird, "Automatically recommending peer reviewers in modern code review," *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 530–543, 2016.

- [100] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Review participation in modern code review," *Empirical Software Engineering*, vol. 22, no. 2, pp. 768–817, 2017.
- [101] R. Tesch, "Qualitative analysis: Analysis types and software tools," *London: Falmer*, 1990.
- [102] D. Muijs, *Doing quantitative research in education with SPSS*. Sage, 2010.
- [103] T. R. Black, *Doing quantitative research in the social sciences: An integrated approach to research design, measurement and statistics*. Sage, 1999.
- [104] R. B. Johnson, A. J. Onwuegbuzie, and L. A. Turner, "Toward a definition of mixed methods research," *Journal of mixed methods research*, vol. 1, no. 2, pp. 112–133, 2007.
- [105] J. M. Morse, *Mixed method design: Principles and procedures*. Routledge, 2016.
- [106] U. Gulec, M. Yilmaz, and V. Isler, "A literature survey: Is it necessary to develop a new software development methodology for virtual reality projects?," *Journal of Universal Computer Science*, vol. 23, no. 8, pp. 725–754, 2017.
- [107] U. Gulec, M. Yilmaz, V. Isler, R. V. O'Connor, and P. Clarke, "Adopting virtual reality as a medium for software development process education," in *Proceedings of the 2018 International Conference on Software and System Process*, pp. 71–75, ACM, 2018.
- [108] U. Gulec, M. Yilmaz, and V. Isler, "Factors that raise the reality of the virtual office environment designed to educate software development processes," in *Proceedings of 12th Turkish National Software Engineering Symposium (UYMS)*, 2018.
- [109] U. Ritterfeld, M. Cody, and P. Vorderer, *Serious games: Mechanisms and effects*. Routledge, 2009.
- [110] M. Yilmaz, *A software process engineering approach to understanding software productivity and team personality characteristics: an empirical investigation*. PhD thesis, Dublin City University, 2013.
- [111] W. Barfield and C. Hendrix, "The effect of update rate on the sense of presence within virtual environments," *Virtual Reality*, vol. 1, no. 1, pp. 3–15, 1995.
- [112] B. G. Witmer and M. J. Singer, "Measuring presence in virtual environments: A presence questionnaire," *Presence*, vol. 7, no. 3, pp. 225–240, 1998.
- [113] S. Yu and C. Levesque-Bristol, "Are students in some college majors more self-determined in their studies than others?," *Motivation and Emotion*, vol. 42, no. 6, pp. 831–851, 2018.

APPENDIX A

TRAINING PROGRAM SCENARIOS

WEEK 1 SCENARIO

Traditionally, a voting machine has been defined by the mechanism the system uses to cast votes and further categorized by the location where the system tabulates the votes. Your company Elma Ltd. were asked to design a voting machine by Gantek A.Ş., on which voters can see a list of candidates and select one to vote for.

The voting machine should check that each voter is eligible to vote. The electoral registrar will also want to print a summary of the total votes for each candidate, and (separately) a list of the voters who have voted, and a list of those who haven't. In case of a dispute the machine should also list a complete record of who voted for whom, but only a judge can use this function. The machine shall be able to transmit 100 votes in every second to a cloud middleware. A voter shall be able to understand the user interface in a reasonable period of time (e.g. think about actual time for an individuals' voting process). None of the actors involved in the voting process should be able to link a ballot to a voter. Each and every ballot should directly be recorded and counted. All votes are considered equal.

WEEK 2 SCENARIO

You are hired as a software engineer from a very fast growing company called Argela, Inc. Innova was founded by Bülent Kaytaş who worked as a system engineer for Alcatel Lucent for more than ten years. Later, he decided to start a company to generate innovative technologies for the future of communication, which has grown rapidly and is now doing 10M Euros per year in sales of software which, has crafted a series of telecom products that lead the way in efficiency and intelligence. The services they provide are an elegant mix of compliance to industry standards, scalable, open and able to keep pace with all the latest advancements. The company now has 25 employees: 3 in management, 5 in marketing and sales, 5 in maintenance, and 12 in software development. On December 22, 2017, Argela successfully completed and delivered its "Development of Software Defined Network (SDN) Technologies, (MİLAT) Project" to the Turkish Under Secretariat for Defense Industries (SSM).

Argela has decided to implement an interactive web-based alternative to the telephone directory based solution that can be utilized in military, public safety and commercial communication infrastructures enabling dynamic cybersecurity using such a service with 256 bit AES encryption, anybody with access to the Internet shall be able to browse and search the company's list of telephone customers to find their name, address, and phone number. In addition, it shall be possible for the customers listed in the directory to extend this information with their email- and web-addresses. To access this functionality, customers must authenticate themselves by supplying a password provided by the telephone company. The system must also handle updates of the directory by company staff. For security reasons, this shall not be done from the web-interface, but only from workstations within the company's internal network.

The system should be able to handle 100 concurrent connections internally while should respond to 1000 participants as quick as possible. In addition, there is a legacy system that works with knowledge base, which was developed in COBOL where some important data sometimes should be double-checked instantly. Therefore a facade might be essential for providing some of the services. For security, an exact copy of selected calls need to be remain in the database for one year while recoding footprints on the server need to be less than 20 megabytes. All designed user interfaces system should be following the GUI Standard 12024 which certainly states the training time even for a difficult walkthrough happen on the application should not exceed half a day. SSM would like to have all development activities in a given a constantly evolving functional and technical landscape by easily adapting to changing requirements throughout the process with the visibility into the actual progress of projects that needs to be available.

WEEK 3 SCENARIO

Field service employee Bob tries to open the door Nr.777 via the locking mechanism at its side. Unfortunately, he breaks his key inside the lock without opening the door. He desperately shakes the door handle, which causes an alarm. Security officer Steven, who is working at the control center, receives a notification about the alarm. The notification consists of an audio signal and a camera position. Steven opens LaVis and enters the camera position. LaVis shows him the live picture of camera 1337 associated with the alarm. The live picture is overlaid with a 3D-Model of the room in which door 777 is flashing red. Steven opens the “Security Staff” menu that shows that three security staff members are available to deal with the alarm. LaVis shows their names, IDs and the position of nearby cameras. In addition, the distance to door 777 is also shown. Two of the security staff members are working at a passage nearby; the third one is further away. However, when Steven watches the videos of the associated cameras, he sees that both are quite busy at the moment. So he decides to send Thomas, the third person. Using Lavis’s drag and drop facility, He drags the icon for Tom to the location of the door alarm. As a result of this action, Toms’ iPhone rings. Tom opens the display and sees the alarm including the location of the door. When Thomas reaches the door, he meets Bob who is still shaking the door. Thomas asks for Bob’s Staff ID Card. Then he verifies the ID card by typing the ID number into the iPhone. The iPhone connects to the control center server and verifies the ID number. After Tom compares Bob’s face with the picture on the ID card he unlocks the door electronically using LaVis on his iPhone. He selects the door and clicks the “Unlock” icon. Bob can finally go through the door.

WEEK 4 SCENARIO

The software system supports a computerized banking network including both human cashiers and automatic teller machines (ATMs) to be shared by a consortium of banks. Each bank provides its own computer to maintain its own accounts and process transactions against them. Cashier stations are owned by individual banks and communicate directly with their own bank's computers. Human cashiers enter account and transaction data. Automatic teller machines (ATM) communicate with a central computer that clears transactions with the appropriate banks. An ATM accepts cash card, interacts with the user, communicates with the central system to carry out the transaction, dispenses cash, and prints receipts. The system requires appropriate record keeping and security provisions. The system must handle concurrent accesses to the same account correctly. The banks will provide their own software for their own computers. The cost of shared system will be appropriated to the banks according to the number of customers with cash cards.

WEEK 5 SCENARIO

A telephone company has decided to implement an interactive web-based alternative to the telephone directory. Using this service, anybody with access to the Internet shall be able to browse and search the company's list of telephone customers to find their name, address, and phone number. In addition, it shall be possible for the customers listed in the directory to extend this information with their email- and web-addresses. To access this functionality, customers must authenticate themselves by supplying a password provided by the telephone company. The system must also handle updates of the directory by company staff. For security reasons, this shall not be done from the web-interface, but only from workstations within the company's internal network.

WEEK 6 SCENARIO

Murat's Pizza Delivery wants to speed up the ordering process, reduce losses caused by misunderstandings on the phone and attract new customers. A new web-based pizza ordering system that allows customers to enter orders in their web browsers is supposed to solve all three issues. The system must be built for the WebObjects platform using the Xcode IDE and integrate in an existing Apache environment.

The ordering system must be easy to use, as customers of all ages and expertise levels are supposed to use it.

Customers may order pizzas with three different types of dough, thick or thin, and various toppings. Customers must be able to register for a customer account. A customer account stores address information and preferences, but no payment details for security reasons. Orders should be possible with or without a customer account. For privacy reasons, customer data must be stored in encrypted form only.

The system must be usable with all major web browsers (i.e. Internet Explorer, Firefox, Safari and Opera) and be able to handle at least 10 customers ordering at the same time.

The cook can request a list of all open orders. When he has finished making a pizza, he marks an order as “ready for delivery”.

A delivery note with the customer’s address, to be attached to the pizza by the cook, is printed automatically.

APPENDIX B

PRE-TEST SCENARIO

PRE-TEST SCENARIO

Tadım Pizza Inc., is a city wide chain of pizza restaurants in Ankara with a tightly run operation that uses a computer to manage inventory of ingredients – such as tomato sauce, pepperoni, cheese and so on. They have an old system that runs on DOS and old computers. They want to upgrade both the hardware and software. In addition to stock control of their ingredients, they want to allow for web-based online ordering of pizzas for home delivery. This web system will be *integrated* with the “point of sale” system in the restaurants that keeps track of the orders in the restaurants.

In addition to traditional crust pizzas, they offer deep pan (Turkish-style) pizzas cheese bread, garlic bread, and soda drinks of all kinds. They have a specialty line of Turkish pizzas with pastrami, sucuk and mince meat, as well as a varying selection of Chef of the Day pizzas. This is an *experimental* line of pizzas. The company wants to track the *acceptance of pizzas from online orders*. If a pizza is popular it will be offered as part of the regular menu.

Each pizza on the menu can be ordered in different sizes.

Each pizza has different ingredients – and in different quantities.

Each pizza comes with a set list of toppings.

Each topping can be optionally ordered in lower, normal and higher amounts.

Additional toppings can be ordered.

Each ingredient and topping is supplied by a number of suppliers.

A pizza cannot be ordered if the ingredients are not available.

Each item on the menu may be single pizza or a combination item with side orders or soda drinks.

When the ingredients run low, online purchase orders need to be generated and emailed to the supplier with the current lowest price.

The old system used by the company has been used for stock control for many years. They are very comfortable with the actions of recording the use of ingredients as pizzas are cooked and of ordering new ingredients. They are much more uncomfortable with the web-based ordering facility. They don't know how it should look or exactly how to match the end user 'ordering experience' with their image of classy, but simple food. They want the system to support up to 100 people ordering concurrently.

APPENDIX C

POST-TEST SCENARIO

POST-TEST SCENARIO

You are hired as a software engineer from a very fast growing company called Elma, Inc. Elma was founded by John Hopkins who worked as a system engineer for Dublin City University for more than ten years. Later, he decided to start a company, which has grown rapidly and is now doing 10M Euros per year in sales of school automation systems primarily to universities throughout the European Union. The company now has 25 employees: 3 in management, 5 in marketing and sales, 5 in maintenance, and 12 in development. The web automation system mainly consists of two modules: (i) an instructor module similar to an electronic version of a grade-book which is basically has a function of adding/removing grades for students' exams, (ii) a student module which is primarily for students to see their exam grades from the web.

You joined Elma 5 days ago and you have been told that you would be working as a Java programmer. However, the management now says: "During your interview, people from management found you capable of handling challenging tasks, we immediately start on a new features for our school automation soft- ware, so our project manager wants you to assist him in some of his software quality management activities. Perhaps, you may start with some basic requirement analysis. Next, we should be working on design and implementation of quality of our subsystem, and I hope you are a quick learner because we need to use your skills on a little bit of software testing. I am sure you will be delighted to work on the updates for our project charter and maybe later you need to do a little bit work on configuration management."

The next day, Marie Currie, who is the head of engineering management, calls you into her office. She says, "Our market research team consisting of all departments finished their study". Team suggested that we need to develop a new release which leverages the market advantage of our product. As an urgent matter, several of our customers demand that a new release should have features like a dial-up interfaces. For example: a student shall be able to dial up school phone and learn his or her grades. Mrs. Currie also tells, a competitor claims that their new product will support mobile devices for accessing grades. However, management decided to implement a phone interface alternative to mobile logins first. Any of the students are able to phone school shall be able to browse their exam results. Moreover, it shall be possible for students to get an email from the system when their exam results are ready for viewing. For some security reasons, all grading shall be done from a web-interface which is accessible only from university campus and can be altered only by both instructors and system administrators.

The next day Marie Currie calls you again for a new part of the project. She says, “As you might know, in several campus based accommodation in many universities are now using (disposable) key cards for dormitory doors (both for doors of individuals and for buildings). These cards and card-locks are usually more preferred than traditional keys because they are easier to maintain. In particular, it is a cheap and effective solution to deal with students who usually lost their keys. The notion behind this re-codeable cards is that the card codes the lock, for example if a card is lost or stolen the newly obtained card recodes the card-lock. After that, old key cards will not work anymore with the lock....”

She continues, “A card-lock unit usually works with battery which is under complete isolation, i.e. no network or any other connections with a computer. Nevertheless, it has an ability for storing a copy of code for the (current) key. That is the code which is also stored in the active key card. A special type of hardware which is connected to computer is usually needed for creating several sets of pseudo-random numbers which are operated as key for these locks. These card locks, however, can only be unlock by the current key code or otherwise with its successor key-code. The key codes are specifically generated by a special hardware as a sequence of pseudo-random numbers. Consequently, if someone loses a card, the next key-code number is generated and loaded in a new card. Whenever this card is used on the key-lock, the old card-code will be disabled automatically. One of the main benefits of this mechanism is that there is no connection required in between information services and locks in campuses unless the lock hardware and keys initially synchronized by using the same pseudo-random generator....”

For example in one form of a typical scenario, a student lost his or her key- card walks to IT services to get a new card which should have the next key for his or her room. As soon as the student walks to the room and uses his or her new card, the lock will be opened with the updated key-code and all previous key-cards will be disabled.

APPENDIX D

T-TABLE

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$	$t_{.9995}$
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	3.686	4.015
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.646	3.965
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.610	3.922
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.579	3.883
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.552	3.850
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.527	3.819
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.505	3.792
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.485	3.768
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.467	3.745
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.450	3.725
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.435	3.707
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.421	3.690
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.408	3.674
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.396	3.659
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.385	3.646
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704	3.307	3.551
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000	2.390	2.660	3.232	3.460
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990	2.374	2.639	3.195	3.416
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984	2.364	2.626	3.174	3.390
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962	2.330	2.581	3.098	3.300
Z	0.000	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576	3.090	3.291
	0%	50%	60%	70%	80%	90%	95%	98%	99%	99.8%	99.9%
	Confidence Level										

APPENDIX E

PRESENCE QUESTIONNAIRE

PRESENCE QUESTIONNAIRE

(Witmer & Singer, Vs. 3.0, Nov. 1994)*

Revised by the UQO Cyberpsychology Lab (2004)

Characterize your experience in the environment, by marking an "X" in the appropriate box of the 7-point scale, in accordance with the question content and descriptive labels. Please consider the entire scale when making your responses, as the intermediate levels may apply. Answer the questions independently in the order that they appear. Do not skip questions or return to a previous question to change your answer.

WITH REGARD TO THE EXPERIENCED ENVIRONMENT

1. How much were you able to control events?

NOT AT ALL			SOMEWHAT			COMPLETELY

2. How responsive was the environment to actions that you initiated (or performed)?

NOT RESPONSIVE			MODERATELY RESPONSIVE			COMPLETELY RESPONSIVE

3. How natural did your interactions with the environment seem?

EXTREMELY ARTIFICIAL			BORDERLINE			COMPLETELY NATURAL

4. How much did the visual aspects of the environment involve you?

NOT AT ALL			SOMEWHAT			COMPLETELY

5. How natural was the mechanism which controlled movement through the environment?

EXTREMELY ARTIFICIAL			BORDERLINE			COMPLETELY NATURAL

6. How compelling was your sense of objects moving through space?

NOT AT ALL			MODERATELY COMPELLING			VERY COMPELLING

7. How much did your experiences in the virtual environment seem consistent with your real world experiences?

NOT CONSISTENT			MODERATELY CONSISTENT			VERY CONSISTENT

8. Were you able to anticipate what would happen next in response to the actions that you performed?

NOT AT ALL			SOMEWHAT			COMPLETELY

9. How completely were you able to actively survey or search the environment using vision?

NOT AT ALL			SOMEWHAT			COMPLETELY

10. How compelling was your sense of moving around inside the virtual environment?

NOT COMPELLING			MODERATELY COMPELLING			VERY COMPELLING

11. How closely were you able to examine objects?

NOT AT ALL			PRETTY CLOSELY			VERY CLOSELY

12. How well could you examine objects from multiple viewpoints?

NOT AT ALL			SOMEWHAT			EXTENSIVELY

13. How involved were you in the virtual environment experience?

NOT			MILDLY			COMPLETELY
INVOLVED			INVOLVED			ENGROSSED

14. How much delay did you experience between your actions and expected outcomes?

NO DELAYS			MODERATE			LONG
			DELAYS			DELAYS

15. How quickly did you adjust to the virtual environment experience?

NOT AT ALL			SLOWLY			LESS THAN
						ONE MINUTE

16. How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?

NOT			REASONABLY			VERY
PROFICIENT			PROFICIENT			PROFICIENT

17. How much did the visual display quality interfere or distract you from performing assigned tasks or required activities?

NOT AT ALL			INTERFERED			PREVENTED
			SOMEWHAT			TASK PERFORMANCE

18. How much did the control devices interfere with the performance of assigned tasks or with other activities?

NOT AT ALL			INTERFERED			INTERFERED
			SOMEWHAT			GREATLY

19. How well could you concentrate on the assigned tasks or required activities rather than on the mechanisms used to perform those tasks or activities?

NOT AT ALL			SOMEWHAT			COMPLETELY

IF THE VIRTUAL ENVIRONMENT INCLUDED SOUNDS:

20. How much did the auditory aspects of the environment involve you?

|_____| |_____| |_____| |_____| |_____| |_____|
NOT AT ALL SOMEWHAT COMPLETELY

21. How well could you identify sounds?

|_____| |_____| |_____| |_____| |_____| |_____|
NOT AT ALL SOMEWHAT COMPLETELY

22. How well could you localize sounds?

|_____| |_____| |_____| |_____| |_____| |_____|
NOT AT ALL SOMEWHAT COMPLETELY

IF THE VIRTUAL ENVIRONMENT INCLUDED HAPTIC (SENSE OF TOUCH):

23. How well could you actively survey or search the virtual environment using touch?

|_____| |_____| |_____| |_____| |_____| |_____|
NOT AT ALL SOMEWHAT COMPLETELY

24. How well could you move or manipulate objects in the virtual environment?

|_____| |_____| |_____| |_____| |_____| |_____|
NOT AT ALL SOMEWHAT EXTENSIVELY

Last version : March 2013

*Original version : Witmer, B.G. & Singer, M.J. (1998). Measuring presence in virtual environments: A presence questionnaire. *Presence : Teleoperators and Virtual Environments*, 7(3), 225-240. Revised factor structure: Witmer, B.J., Jerome, C.J., & Singer, M.J. (2005). The factor structure of the Presence Questionnaire. *Presence*, 14(3) 298-312.

APPENDIX F

IMMERSIVE TENDENCIES QUESTIONNAIRE

IMMERSIVE TENDENCIES QUESTIONNAIRE

(Witmer & Singer, Version 3.01, September 1996)*

Revised by the UQO Cyberpsychology Lab (2004)

Indicate your preferred answer by marking an "X" in the appropriate box of the seven point scale. Please consider the entire scale when making your responses, as the intermediate levels may apply. For example, if your response is once or twice, the second box from the left should be marked. If your response is many times but not extremely often, then the sixth (or second box from the right) should be marked.

1. Do you easily become deeply involved in movies or tv dramas?

NEVER			OCCASIONALLY			OFTEN

2. Do you ever become so involved in a television program or book that people have problems getting your attention?

NEVER			OCCASIONALLY			OFTEN

3. How mentally alert do you feel at the present time?

NOT ALERT			MODERATELY			FULLY ALERT

4. Do you ever become so involved in a movie that you are not aware of things happening around you?

NEVER			OCCASIONALLY			OFTEN

5. How frequently do you find yourself closely identifying with the characters in a story line?

NEVER			OCCASIONALLY			OFTEN

6. Do you ever become so involved in a video game that it is as if you are inside the game rather than moving a joystick and watching the screen?

NEVER			OCCASIONALLY			OFTEN

7. How physically fit do you feel today?

NOT FIT MODERATELY FIT EXTREMELY FIT

8. How good are you at blocking out external distractions when you are involved in something?

[illegible]

9. When watching sports, do you ever become so involved in the game that you react as if you were one of the players?

NEVER OCCASIONALLY OFTEN

10. Do you ever become so involved in a daydream that you are not aware of things happening around you?

NEVER OCCASIONALLY OFTEN

11. Do you ever have dreams that are so real that you feel disoriented when you awake?

NEVER OCCASIONALLY OFTEN

12. When playing sports, do you become so involved in the game that you lose track of time?

NEVER OCCASIONALLY OFTEN

13. How well do you concentrate on enjoyable activities?

NOT AT ALL MODERATELY VERY WELL
WELL

14. How often do you play arcade or video games? (OFTEN should be taken to mean every day or every two days, on average.)

|_____|_____|_____|_____|_____|_____|_____|
NEVER OCCASIONALLY OFTEN

15. Have you ever gotten excited during a chase or fight scene on TV or in the movies?

|_____|_____|_____|_____|_____|_____|_____|
NEVER OCCASIONALLY OFTEN

16. Have you ever gotten scared by something happening on a TV show or in a movie?

|_____|_____|_____|_____|_____|_____|_____|
NEVER OCCASIONALLY OFTEN

17. Have you ever remained apprehensive or fearful long after watching a scary movie?

|_____|_____|_____|_____|_____|_____|_____|
NEVER OCCASIONALLY OFTEN

18. Do you ever become so involved in doing something that you lose all track of time?

|_____|_____|_____|_____|_____|_____|_____|
NEVER OCCASIONALLY OFTEN

Last version: March 2013

Original version : Witmer, B.G. & Singer, M.J. (1998). Measuring presence in virtual environments: A presence questionnaire. *Presence : Teleoperators and Virtual Environments*, 7(3), 225-240.

APPENDIX G

GRADING POLICY

GRADING POLICY

Letter Grade	Coefficient	Score intervals
AA	4,00	90-100
BA	3,50	85-89
BB	3,00	80-84
CB	2,50	75-79
CC	2,00	70-74
DC	1,50	65-69
DD	1,00	60-64
FD	0,50	50-59
FF	0,00	0-49
NA	0,00	*

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Güleç, Ulaş

Nationality: Turkish (T.C.)

Date and Place of Birth: 21.10.1989, Balıkesir

Marital Status: Married

Phone: 0 312 2331354

Fax: 0 312 2331024

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Department of Computer Engineering, Çankaya University	2015
B.S.	Department of Industrial Engineering, Çankaya University	2012
B.S.	Department of Computer Engineering, Çankaya University	2012
High School	Çankaya Anadolu Lisesi	2007

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
Sep 2012 - Mar 2013	Çankaya Teknoloji	Developer
Mar 2013 - Present	Department of Computer Engineering, Çankaya University	Lecturer

PUBLICATIONS

Articles Published in the International Journals (SCI, SCI-EXPANDED, SSCI)

- 1- M. Yilmaz, F. S. Tasel, **U. Gulec**, and U. Sopaoglu (2018). Towards a Process Management Life-Cycle Model for Graduation Projects in Computer Engineering, *PlosOne* (Accepted).
- 2- **U. Gulec**, M. Yilmaz, V. Isler, R. V. O'Connor and P. Clarke (2018). A 3D Virtual Environment to Standardize Training Methods of Soccer Referees, *Computer Standards & Interfaces* (Accepted).
- 3- M. Denizci Nazligul, M. Yilmaz, **U. Gulec**, A. E. Yilmaz, V. Isler, R. V. O'Connor, P. Clarke and M. A. Gozcu (2018). An Interactive 3D Virtual Environment to Reduce the Public Speaking Anxiety Levels of Novice Software Engineers, *IET Software* (Accepted).
- 4- **Ulas Gulec**, Murat Yilmaz and Veysi Isler (2017). A Literature Survey: Is it Necessary to Develop a New Software Development Methodology for Virtual Reality Projects, *Journal of Universal Computer Science*, 23 (8), pp. 725-754.
- 5- **Ulas Gulec** and Murat Yilmaz (2016). A Serious Game for Improving the Decision Making Skills and Knowledge Levels of Turkish Football Referees according to the Laws of the Game, *SpringerPlus*, 5:622.

International Conference Publications

- 1- **U. Gulec**, M. Yilmaz, V. Isler, R. V. O'Connor and P. Clarke (2018). Adopting Virtual Reality as a Medium for Software Development Process Education, International Conference on Software System Process (ICSSP 2018), 26-27 May 2018, Gothenburg, Sweden.
- 2- M. Denizci Nazligul, M. Yilmaz, **U. Gulec**, M. A. Gozcu, R. V. O'Connor and P. Clarke (2017). Overcoming Public Speaking Anxiety of Software Engineers Using Virtual Reality Exposure Therapy, Proceedings of the 24th European and Asian Con-

ference on Systems, Software and Services Process Improvement (EuroSPI 2017), 6-8 September 2017, Ostrava, Czech Republic.

3- Saran, M. and **Güleç, U.** (2014). Contribution of Intelligent Repeat Engine in Mobile Learning for Enhancing Students' Learning in Industrial Engineering Education, 2014 SOLSTICE eLearning and CLT Conference, Edge Hill University, Ormskirk, UK.

National Conference Publications

1- **U. Gulec**, M. Yilmaz ve V. Isler (2018). Yazılım Geliştirme Süreçlerini Eğitmek Amacıyla Tasarlanan Sanal Ofis Ortamında Ortamın Gerçekliğini Arttıran Etmenler, 2018 Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), Sabancı Üniversitesi, İstanbul, Türkiye.

2- **U. Gulec**, M. A. Gozcu, S. Dogan, N. Mesurhan, M. Yilmaz, V. Isler ve M. Dinc (2018). Simulacrum: Savaş Koşullarında Acil Tıbbi Müdahale ve İlk Yardım Simülasyonu, 2018 Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), Sabancı Üniversitesi, İstanbul, Türkiye. (**Best Paper Award**)

3- M. Yilmaz ve **U. Gulec** (2018). Yazılım Mühendisliği Dersi için Geliştirilmiş Ders Akış Modeli ve İlgili Alan Saha Çalışması, 2018 Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), Sabancı Üniversitesi, İstanbul, Türkiye.

4- **U. Gulec**, M. Yilmaz ve M. A. Gozcu (2017). Bireylerin Programlama Yeteneklerini ve Bilgi Seviyelerini Arttırmak Amacıyla Düşünölmüş Ciddi Oyun Tabanlı Öğrenme Çatısı – CENGO, 2017 Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), Alanya Hamdullah Emin Paşa Üniversitesi, Alanya, Türkiye.

5- M. Yilmaz, **U. Gulec**, R. V. O'Connor, P. Clarke ve E. Tüzün (2017). İşe Alıştırma (Onboarding) Süreçlerinin İyileştirilmesi için Düşünölmüş Bir Endüstriyel Vaka Çalışması, 2017 Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), Alanya Hamdullah Emin Paşa Üniversitesi, Alanya, Türkiye.

6- **U. Gulec**, M. Yilmaz ve M. A. Gozcu (2016). Futbol Hakemlerinin Eğitimi Amacıyla Tasarlanan Futbol Simülasyonunda Maçın Dinamizmini Sağlayan Etmen-

ler, 2016 Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), Çanakkale 18 Mart Üniversitesi, Çanakkale, Türkiye.

7- M. Yılmaz, S. Tasel, **U. Gulec** ve U. Sopaoglu (2016). Bilgisayar Mühendisliği Bitirme Projeleri için Düşünölmüş Bir Süreç Yönetim Modeli, 2016 Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), Çanakkale 18 Mart Üniversitesi, Çanakkale, Türkiye.

8- **U. Gulec** ve M. Yılmaz (2015). Futbol Hakemlerinin Karar Verme Yeteneklerini Geliştirmek İçin Düşünölmüş Ciddi Oyun Tabanlı Öğrenme Çatısı, 2015 Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), Yaşar Üniversitesi, İzmir, Türkiye.