Received: 6 January 2021 | Revised: 15 March 2021 | Accepted: 6 September 2021

IET Software

DOI: 10.1049/sfw2.12047

ORIGINAL RESEARCH PAPER

Applying virtual reality to teach the software development process to novice software engineers

Ulas Gulec[1,2] | Murat Yilmaz[3] | Veysi Isler[2,4] | Paul M. Clarke[5]

[1]Department of Software Engineering, TED University, Ankara, Turkey

[2]Game and Simulation Group, Simovate Information, Ankara, Turkey

[3]Department of Computer Engineering, Faculty of Engineering, Gazi University, Ankara, Turkey

[4]Department of Computer Engineering, Hasan Kalyoncu University, Gaziantep, Turkey

[5]School of Computing, Dublin City University, Dublin, Ireland

Correspondence

Ulas Gulec, Department of Software Engineering, TED University, Ankara, Turkey.
Email: ulas.gulec@tedu.edu.tr

Abstract

Software development is a complicated process that requires experienced human resources to produce successful software products. Although this process needs experience from the individuals, it is hard to provide this experience without encountering real incidents during the software development process. To fill this gap, this study proposes a Virtual Reality Based Software Development Framework (VR-SODEF), which provides an interactive virtual reality experience for individuals learning about the tasks of software development starting from requirement analysis through software testing. In the VR-SODEF, the participant takes on the role of a novice software developer being recruited into a virtual software development organisation who should work alongside five virtual characters, played by artificial intelligence. This exclusive viewpoint draws participants from the 2D separation of the classical experience and virtually into the world of the software development itself. Participants experience the intense dramatic elements created for simulation and confront the challenges of virtual software practitioners in a somewhat uncompromising virtual simulation environment. To examine the efficiency of the VR-SODEF, it was tested on 32 computing students, with results indicating that virtual reality can be an effective educational medium, especially for skills that might traditionally be acquired through experience rather than traditional classroom-based teaching.

KEYWORDS

interactive learning environments, software development life cycle, software development process, software engineering education, virtual reality

1 | INTRODUCTION

The software development sector, one of the most popular workspaces nowadays, is a difficult sector in terms of producing successful products even if the definition of software seems easy [1]. According to the National Institute of Standards and Technology (NIST), the annual loss of unsuccessful software projects to the US economy is about $ 59 Billion in 2002 [2]. When today's data is analysed, the total worldwide financial loss of failed software in 2017 is around 1.7 trillion dollars [3]. These numerical values indicate that software companies waste their money trying to produce successful software products [4]. In order to prevent such undesirable situations, software practitioners are required to gain real-life practice since the most important factor, which makes software products fail, is the lack of experience and knowledge of the individuals who work in this area [5].

Although the training of software practitioners is an important issue to increase the success rate of the completed software projects, it is not an easy topic since the details of the software development is generally considered to be a complex task for individuals to be trained [6]. To overcome this argument, virtual reality (VR) provides a beneficial training environment, which is close to the real environment for individuals to gain experience without having actual risk [7]. There are various VR applications that are developed for educational purposes in different domains such as military [8], medical and healthcare [9], psychology [10], sport [11] and so on. Hence, this study aims to train novice software engineers on the software development process in a 3D virtual office environment that is similar to a real environment by eliminating the limitations of the existing studies in the literature. To achieve this aim, two complementary applications were developed to produce a dynamic and useful training framework for the novice software engineers. In addition, HTC Vive, which can create a realistic perception on the participants for the virtual environments, was used in order to visualise the 3D virtual office environment. The proposed system was tested with 32 students who are currently studying at a department of computer engineering, to answer the research questions of the study:

RQ1: *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 software development life cycle* (*SDLC*)?

The overall structure of this study is as follows: Section 2 details the studies that were developed to train the individuals about the software development process. The following section describes the research technique used in this study with the reasons of selection. In addition, this section also gives the information about the workflow of the proposed system and the participants of this study. The next section describes the system description by presenting the interfaces. Section 5 mentions how the developed system has been tested by the participants and how the results obtained from the tests were analysed by using statistical tests. In addition, the opinions of the participants about the system were also illustrated in this section. Finally, the last section gives a summary of the overall study and details the conclusions reached.

2 | BACKGROUND AND RELATED WORKS

Whereas once upon a time VR was considered to suffer from usability issues [12] running the risk that students might be unnecessarily exposed to information overload [13], the application of VR is established as a medium for learning [14] and VR has, furthermore, been shown to be effective for teaching in both higher [15] and elementary [16] education. Even though students may have to overcome the technical aspect of acquaintance with VR technology, they nevertheless report an overall positive experience [17] and improved performance [18] from VR-based learning. Indeed, recent publications suggest that an absence of technical familiarity with VR-related technology is also an inhibitor for some teachers [19] and that VR applications should offer experimental support [20]. A further recent publication supports the case that immersive VR-based games are effective for knowledge delivery because they can be highly engaging, which promotes improved cognitive learning [21]. Given these various findings from related literature, we believe that VR-based learning can play an important role in software development project and process-related education.

As outlined in the position paper [22] associated with this work, individuals in software development projects are continually challenged with process adaptation [23], a challenge we suggest must be greatly increased for novice software professionals. And while process-based knowledge is difficult to acquire other than through experience, we have also seen that knowledge can also be lost of projects simply through turnover of personnel, which it seems a particular challenge in the Open-Source Software Development domain [24]. There are, furthermore, many different types of practices, with some of these practices being carried forward from one software development approach to another as is the case with agile software development incorporating many pre-existing practices/processes [25]. It has also been shown that there are often different terms applied for the same practice and even different practice terms have multiple different meanings (instances of synonymy and polysemy), [26, 27] giving rise to a particularly difficult challenge in terms of education. This challenge is centred on the problem understanding: if different terms are used in sometimes confusing ways, how can educators assist trainees in understand the underlying phenomena and concerns? We suggest that this may only be resolvable through either actual experience or simulated learning. In studies in the literature, individuals benefit from the properties of 2D, card and board games, some of them have provided a more active learning concept to the individuals by developing 3D virtual environments [28, 29].

In the first study, a card game was developed to increase the experience levels of the individuals about software development processes [30]. This game can be played with multiple players and the role of each player is the project team leader. In this game, tasks related to the project management are given to the players and it is expected from them to complete these tasks within the limits of both time and money. The player who completes the tasks in the most appropriate way to the time and money constraints wins the game. The game was tested in order to understand its efficiency. The results of the tests indicate that this card game can be used as an educational tool in order to support traditional education techniques.

In the other study, a simulation framework was developed to help the individuals experience different software development methodologies by enabling them to make decisions about the issues related to project management as a project manager or a team leader [31]. To complete this operation, the players have to use the flowchart elements in this educational framework. Therefore, it is expected that the players increase their decision-making skills regarding the software development process.

Another study has produced a 2D game to describe how the requirements of a project should be determined [32]. This game provides different roles to the players such as system analyst, project manager, team leader or designer in order to determine the requirements of the project based on the given project definition. Players are expected to enhance their personal development by taking decisions about project management according to their roles in the game. This game was also tested with the students to understand its efficiency. According to the test results, this game is a useful tool to increase the knowledge levels of the individuals about software management.

In a similar way, a 2D game, which contains the similar game dynamics and elements to the classical tower-defence game, was developed to teach the tasks that occurred during the maintenance phase of software projects [33]. In this game, the players have the towers and they try to protect these towers by finding and solving the bugs in the software projects. This study has selected 18 students to understand the effect of the game on the students. The findings show that this game can be an alternative training tool that can be used to teach students the basics of the maintenance phase of the SDLC.

In addition to the above-mentioned 2D and card games, there are also 3D virtual environments developed to educate individuals about software engineering issues. An example study was completed by Aydan et al. [34]. In this study, a 3D virtual environment was designed to increase the knowledge levels of the individuals about the basics of ISO/IEC 12207:1995. This game was supported by non-player characters (NPCs) to inform the participants about their wrong choices. This game was also tested with the students and the test results illustrate that the knowledge level of the participants about the basics of ISO/IEC 12207:1995 was increased by using this environment.

In another similar study developed in this context, the Second Life virtual environment was used as an environment where students and lecturers could interact with each other [35]. In this environment, while both students and lecturers represent themselves with virtual characters, students can ask questions of their lecturers on software engineering. According to the results obtained from this study, the students have increased their knowledge levels about software engineering because they felt more comfortable asking questions since they hide their real identities by using virtual characters.

In another similar study, a 3D virtual environment was designed to describe the procedure of the Scrum methodology [36]. An example software project was integrated into the system to show the steps of the Scrum methodology. It was expected that the participants complete the steps of the Scrum methodology based on the tasks given in the project. In total, 45 undergraduate students were selected to test the system in order to measure the effect of the environment on the individuals. The results of this study show that this virtual environment is a useful tool that can be used to explain the flow of the Scrum methodology.

In the final study, Open Wonderland environment was used as a distance learning tool to train the participants about agile methodologies [37]. An example project, which is not related to software engineering, was integrated into the system and it was expected from the participants to increase their knowledge levels about the properties of the agile methodologies such as user stories, features in user stories and team collaboration while they were developing the project. The findings of this study illustrate that although the system needs significant improvements, it can be used for training individuals about agile methodologies.

When the existing studies in the literature were analysed in detail, it was observed that these studies have some limitations when they are used for educational purposes. These limitations are as follows:

- **Large disparity between the virtual world and the real environment:** In order for the virtual environments to be considered successful, the designed environment must be similar to the real environment [38]. Therefore, the learning acceleration of the individuals can be increased more since they feel themselves to be the real environment. Hence, this situation brings real life experience to individuals without actual risks. Although it is an essential issue for training tools, the card and 2D games cannot provide this feature since they do not have enough hardware and software functions. Hence, this situation decreases the sense of presence, which is the most significant factor that shows the success of the designed environment [39]. 3D environments can provide the sense of presence; however, the interaction between the users and environment is limited in the existing studies so that they did not measure their participants' level of sense of presence.
- **Focus of virtual world tasks is too narrow:** Software is a product obtained after several different phases in SDLC; however, 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 process. As a result of this situation, the users cannot deal with the problems occurred in each phase of SDLC and cannot experience the project as it is expected.
- **Insufficient numbers and variations of simulations:** Software developers in the sector have taken part in many different projects throughout their careers. In this way, they have a chance to increase their experience levels. Hence, the fact that individuals take part in different projects in gaining experience and solving the problems in those projects has a very important role. Although it is an essential issue, the existing studies have contained only one project as an example test project. Moreover, some of them were not related to software engineering and are not detailed projects as in real life. Therefore, this case can be shown as an important drawback of the existing system because analysing only one project limits the development of individuals since the requirements of real-life projects are very different from each other. Participants cannot face to different problems occurring in the different types of projects.

- **Absence of AI-enabled non-player characters:** If individuals are expected to gain experience on a subject, the frequent repetition on the related topic plays an important role in achieving this situation [40]. Although repetition is a critical factor that affects personal evolution, the platforms designed in the above studies require multiple players to experience the actions in the software development phases. If a person cannot find other players, s/he will not use the system. This is the main drawback of the existing systems since they are not continuously available.

3 | METHODOLOGY

3.1 | Mechanism of the system

As previously mentioned, this study proposes a system to train novice software engineers about the software development process without having any actual risks. The aim is to increase the level of knowledge and experience of the novice software engineers about the software development process in an environment similar to a real environment by eliminating the limitations of the existing studies in the literature, which have been developed for a similar purpose. To achieve this aim, two complementary applications were developed in the scope of VR-SODEF. These applications are **a desktop application for generating project scenarios** and **a 3D virtual office environment**.

The purpose of the desktop application is to enable the authorised people in companies to create different project scenarios, which contain the whole project development process including requirement analysis, design, implementation, test and maintenance phases. For this purpose, this application produces an Extensible Markup Language (XML) file 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.

After producing the XML file, it is necessary to animate the scenario written in the XML file in the virtual environment. In order to accomplish this goal, the XML file is parsed by the virtual 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. At this stage, the virtual office environment has the ability to simulate the project scenario since it keeps all the information about the software project in the XML file. The participants can live the project scenario in the virtual office environment by using HTC Vive. The systematic workflow of the study can be summarised as shown in Figure 1.

3.2 | Research methodology and participants

In this study, a quantitative research approach supported by validation interviews was used since this study contains the properties of both qualitative and quantitative research methodologies. For the quantitative part of the study, pre-test and post-test were administered to the participants to observe the progress of their knowledge about software engineering concepts. Then, the numerical values obtained from the tests were analysed by applying the statistical methods in order to figure out whether there is a significant difference between the results of pre-test and post-test. In addition, Presence Questionnaire (PQ) and Immersive Tendencies Questionnaire (ITQ), which are the most commonly used questionnaires to measure the presence levels of the individuals when they use the virtual system [41], were organised to measure the level of sense of presence of the participants. For the qualitative part of the study, a set of semi-structural interviews was organised with experts in the software engineering topic before starting the study to get their suggestions and opinions about the proposed system. In addition, a set of semi-structural interviews was also organised with the participants after the training period to learn their opinions about the system. In this way, both the

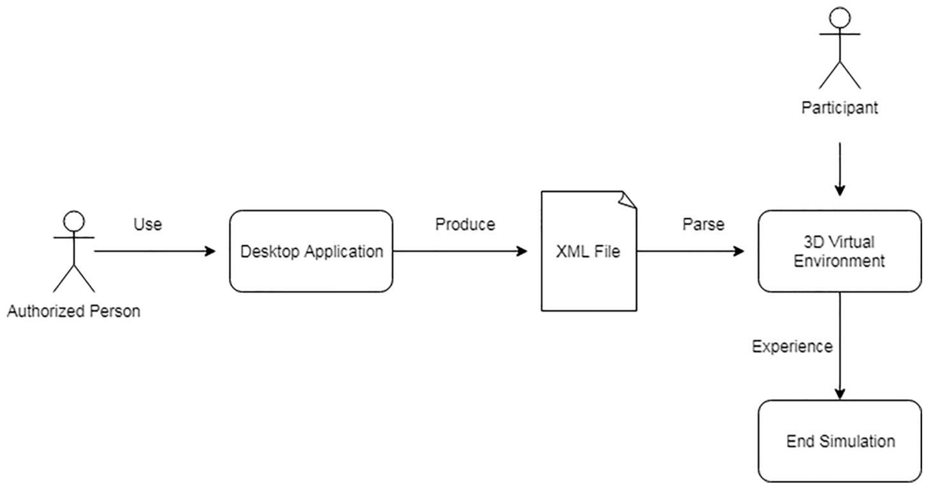


FIGURE 1 Workflow of the system

qualitative and quantitative data were obtained at the end of the study.

The proposed system was tested with 32 students, who have provided the below criteria:

- 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 of 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.

These students were randomly separated into an experimental group ($N = 16$) and a control group ($N = 16$). The students in the control group did not use the virtual office environment for the topics of software engineering. They have studied these concepts from traditional resources such as books, presentations and videos. The students in the other group have just used the virtual office environment to increase their knowledge levels about software development process. 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 the progress of their knowledge about software engineering concepts. The post-test gave another case-study to the students to complete the same issues as in 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.

3.3 | 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 [42]. These factors can also decrease the correctness and trustworthiness of the studies. As in many other studies in the literature, there may be some factors affecting the results of the study. These threads can be classified in four different categories as follows:

1. **Construct Validity:** Ability to measure the proposed idea in a correct way.
   - **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 and post-test).
2. **Internal Validity:** The research design should be internally consistent.
   - **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 and the participants in the control group may not study as expected.
   - **Instrumentation effect:** Any change in the system during the training period.
3. **External Validity:** The results obtained from this study should be directive for the researchers and should be generalised for target population.
   - **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.
4. **Reliability:** 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.

4 | DESIGN AND IMPLEMENTATION

As was mentioned in the previous section, two complementary applications were developed in the scope of this study. These applications are **Scenario Generator** and **3D Virtual Office Environment**.

The aim of these applications was to create and animate a software project scenario, which contains the tasks related to each phase of SDLC. Therefore, the novice software engineers have a chance to increase their level of knowledge and experience by completing the tasks in each phase of SDLC.

4.1 | Scenario generator

This program enables to enter the information, tasks and items that are related to the whole project development processes. It can produce lots of different project scenarios that include the whole project development process. In this way, '*Missing Whole Project Development Processes*' and '*Limited Number of Stories*' limitations of the existing studies in the literature can be eliminated.

When the program is opened, the main page appears on the screen as shown in Figure 2. This page consists of six different segments: '*Project Information*', '*Requirement Analysis*', '*Design*', '*Implementation*', '*Test*' and '*Maintenance*', respectively, since it is necessary to generate a software project scenario that contains the several different assignments related to each task of SDLC for participants to experience the entire software development process.

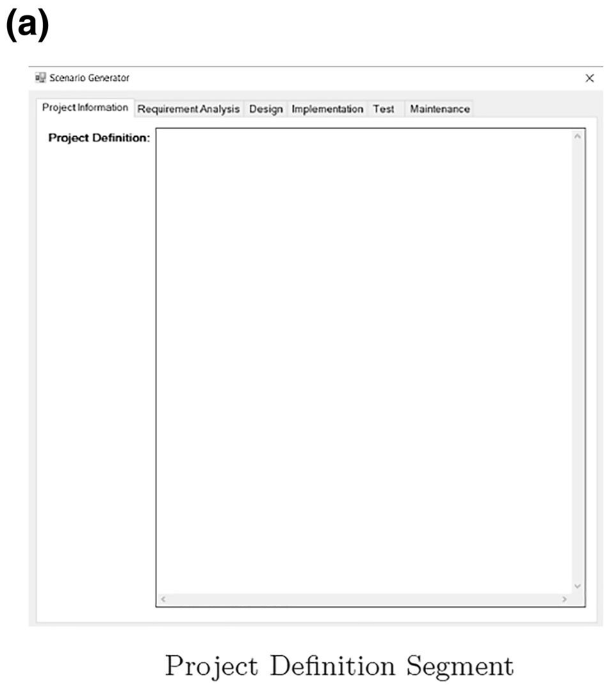
In the first segment, the user should enter the project definition into the related text-box as shown in Figure 2. After writing the project definition, the user should pass the second segment in order to enter the requirements of the project. In this page, there are two different categories that are '*Real Requirements*' and '*Fake Requirements*' (see Figure 2). The reason for having two different categories is to create a challenge for the novice software engineers that they find differentiate correct requirements of the projects from the wrong requirements. Thus, they have a chance to see their missing points since the virtual environment has an ability to give feedback to them about their wrong choices during the simulation.

After recording the requirements of the project, the third segment asks the user to enter both the actors and the functionalities of the system as shown in Figure 3. In addition, the user can also create the relationships between the actors and the functionalities using this segment. With this property, the novice software engineers can create use-case diagrams of the relevant project when they arrive at the design stage in the virtual office environment.

Figure 3 illustrates the implementation segment of the program. In this page, the user records the problem that is expected from the novice software engineer to solve by developing an algorithm in the virtual office environment. Thanks to this phase, the novice software engineer's coding ability can be measured since s/he should write C# code to solve the question entered by using the scenario generator program. This code is going to be automatically executed in the virtual office 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 novice software engineer has written the code correctly. For this reason, this page allows the user 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. As a result of this page, the information entered in this step enable the virtual office environment to execute the code written by the novice software engineer and to understand whether the written code is correct.

In the test phase of the software development process, it is expected of the novice software engineers to perform a code review in the virtual office environment. In order for the participant to be able to perform a code review, the user must add a code block with errors to the software development scenario. The screen in Figure 3 is designed to allow this work to be done. At this stage, there is a similar logic to the requirement analysis phase. This means that the user can also add an error that is not in the code. In this case, the novice software engineer 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.

For the final segment, which contains the fields related to the maintenance operations of the project, the user inputs both the definition and the order of the tasks, which should be



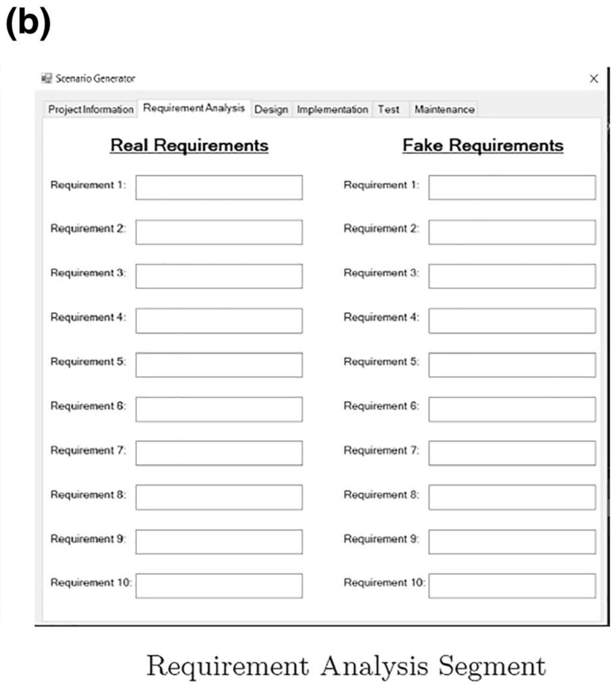


FIGURE 2 Screens of scenario generator

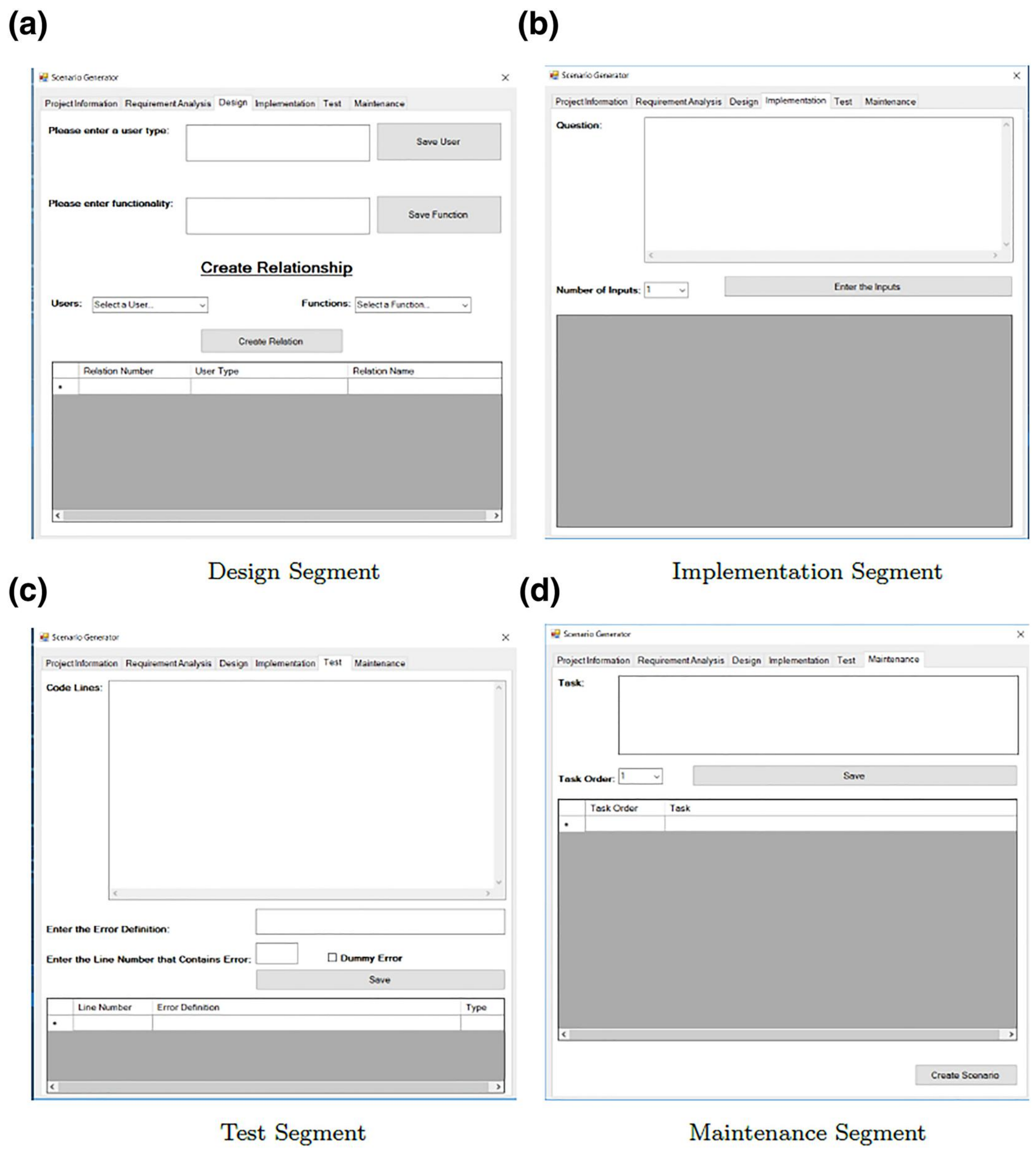


FIGURE 3 Screens of scenario generator

ordered by the novice software engineer in the virtual office environment. A screen as shown in Figure 3 is designed for the user to enter this information.

After entering all information about the whole software project development process, an XML file is automatically created if all necessary fields are filled. This file plays a critical role to standardise the scenarios related to the software project development process. The content of this file should be easily understood by the individuals working in the field of software development in order to be able to create a common language. In addition, it also has to be understandable by the computer in order to visualise the entered scenario in the virtual office environment. To achieve these goals, an ontological meta-language was developed to create a common language for the software project scenarios as a content of the XML file. After producing the XML file, it is necessary to animate the scenario written in the XML file in the virtual office environment. The next section will introduce the virtual office environment in detail.

4.2 | 3D virtual office environment

The virtual office environment supported by HTC Vive was designed to teach the basis of software engineering concepts to novice software engineers. In this environment, they have a chance to live the whole software development process without taking any actual risks. Therefore, the novice software engineers may increase both their knowledge and experience levels about software engineering concepts.

To accomplish this aim, an input, which contains the tasks related to the software project, 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 office environment. For this reason, the environment should read and parse the XML file to store the information related to software project.

After parsing the XML file, the novice software engineer starts the simulation as shown in Figure 4. As shown, the novice software engineer stands at the entrance of the office. In the upper left corner of the screen, there is a clock to show the elapsed time in the virtual office environment while there is a calendar to show the date in the right-hand side of the screen. The date was set up as June 1 at the beginning of the simulation since the novice software engineer portrays the character who is a new worker of the virtual company. In addition, the time zone of the virtual office environment is adjusted to equal an hour in virtual office environment to a minute in real life. Hence, a working day in the virtual office lasts 9 min in real life. After passing 9 min 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 the above figure, there are five different cubicles in the virtual office. Each cubicle represents each phase of SDLC. The cubicles on the left-hand side of the 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-hand 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 four employees, one of whom works on the same project as the virtual character played by the novice software engineer. Hence, this virtual office environment has a total of 20 NPCs. Five of them have a basic AI algorithm with case-based reasoning approach to guide the novice software engineer by determining his/her progress in the environment.

The process of the project in a virtual office environment begins when the novice software engineer goes to his/her project partner in the first cubicle. After that, a dialogue box opens as shown in Figure 5. In this box, a message containing the tasks, which should be completed by the novice software engineer, is illustrated. Once the novice software engineer receives the message from the NPC in this manner, s/he starts to move in the environment to complete the given task. As it is expected from the novice software engineer to perform the requirement analysis at the first stage, the NPC asks the novice software engineer whether s/he reads the project description document; hence, the novice software engineer 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 5. After finding the file, the panel appears on the screen (see Figure 5) to enable the novice software engineer to read the project definition. This screen contains the definition of the project that is generated by using 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 novice software engineer reads the project description from the papers inside the file. After reading the project definition, the novice software engineer returns to his/her team member to get a new task. The NPC understands that the novice software engineer has read the project definition document and tells the novice software engineer that s/he can start the requirement analysis phase of the project. When the novice software engineer starts the requirement analysis phase, the panel in Figure 5 appears on the screen. This screen contains two different sub-panels, being left and right. In the left panel, both the correct and the wrong requirements are listed while some of the correct requirements are randomly selected and listed in the right panel. In this phase, the novice software engineer should find and transfer the correct requirements of the project from the list of all requirements.

After finishing the requirement analysis phase, the NPC informs the novice software engineer that this phase is over and that s/he should go to the design phase. After the novice software engineer gets this information, s/he goes to the NPC responsible for the design phase as shown in Figure 6.

In the design phase, it is expected of the novice software engineer to create a use-case diagram of the project as an early

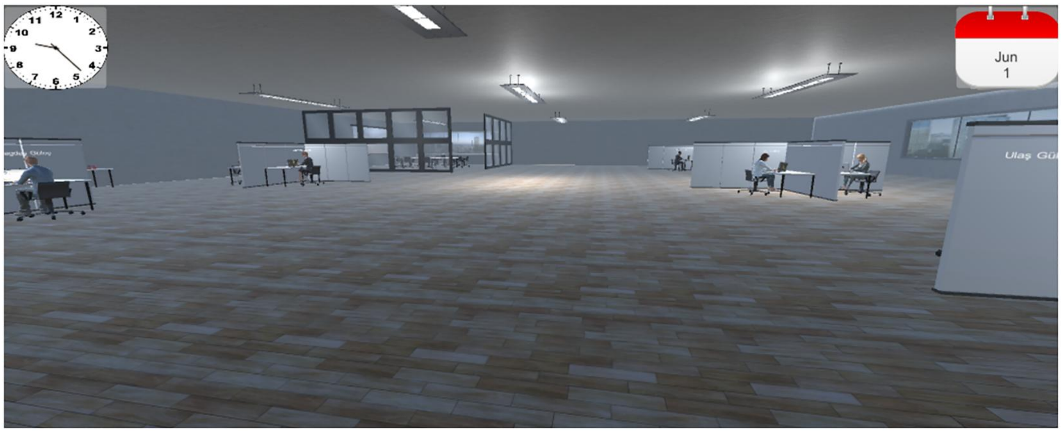
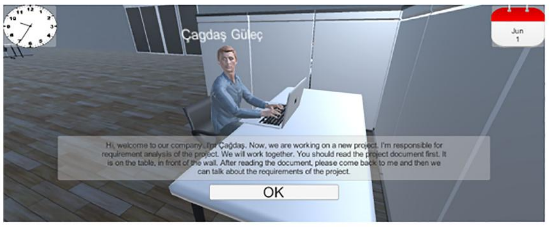


FIGURE 4 Starting screen of simulation

(a)

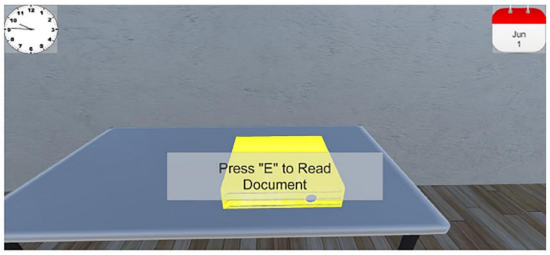


Hi, welcome to our company. I'm Çağdaş. Now, we are working on a new project. I'm responsible for requirement analysis of the project. We will work together. You should read the project document first. It is on the table, in front of the wall. After reading the document, please come back to me and then we can talk about the requirements of the project.

OK

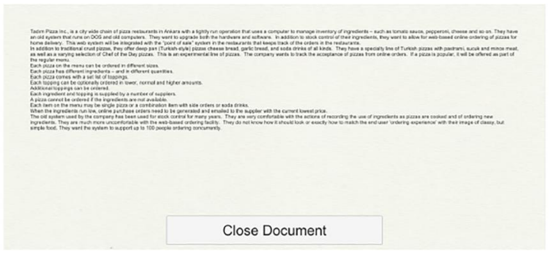
First Dialogue Box of NPC in the Requirement Analysis Phase

(b)



Opening Project Description Document

(c)

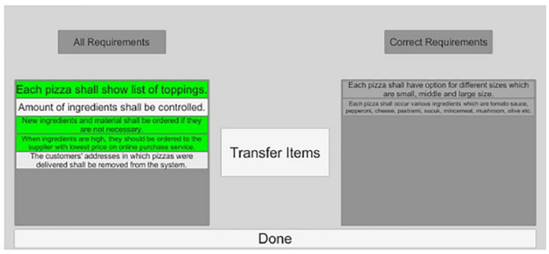


Tadım Pizza Inc., is a city wide chain of pizza restaurants in Ankara with a tightly run operation that uses a computer to manage inventory of ingredients – such as tomato sauce, pepperoni, cheese and so on. They have an old system that runs on DOS and old computers. They want to upgrade both the hardware 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 do not 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.

Close Document

Reading Project Description Document

(d)



All Requirements

🟩 Each pizza shall show list of toppings.
Amount of ingredients shall be controlled.
🟩 New ingredients and material shall be ordered if they are not necessary.
🟩 When ingredients are high, they should be ordered to the supplier with lowest price on online purchase service.
The customers' addresses in which pizzas were delivered shall be removed from the system.

Transfer Items

Correct Requirements

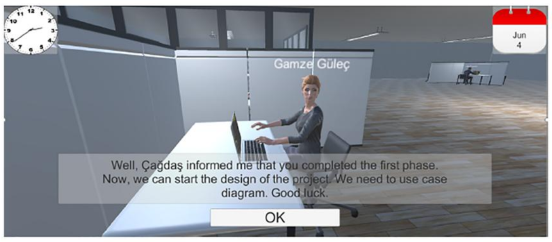
Each pizza shall have option for different sizes which are small, middle and large size.
Each pizza shall occur various ingredients which are tomato sauce, pepperoni, cheese, pastrami, sucuk, mincemeat, mushroom, olive etc.

Done

Requirement Analysis Phase in Virtual Office

FIGURE 5 Screens of requirement analysis phase in the virtual office environment

(a)

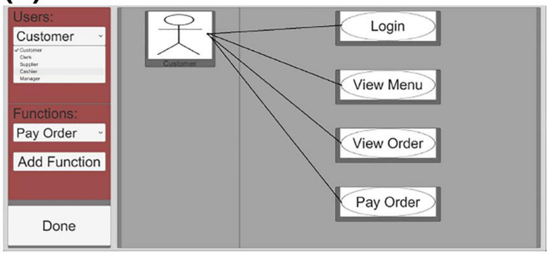


Well, Çagdaş informed me that you completed the first phase. Now, we can start the design of the project. We need to use case diagram. Good luck.

OK

First Dialogue Box of NPC in the Design Phase

(b)



Design Phase in Virtual Office

FIGURE 6 Screens of design phase in the virtual office environment

design phase of the project [43, 44]. As was mentioned in the previous section, the actors, functions and relations between the actors and the functions of the project were determined via using the scenario generation program. In the virtual office environment, the novice software engineer should create the correct relations between the actors and the functions. As shown in Figure 6, both the actors and the functions are listed in the drop-down lists separately. In order to create the use-case diagram, the novice software engineer should select the necessary actors and functions.

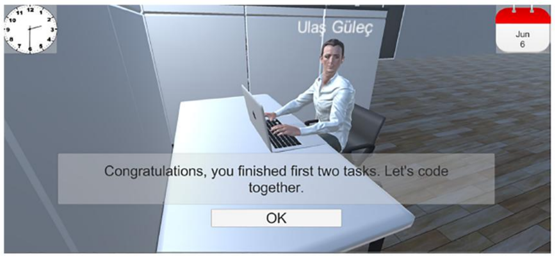
When the design phase is completed, the NPC responsible for the design phase of the project warns the novice software engineer that s/he should start the implementation phase of the project. The novice software engineer who receives this warning goes to the NPC, who is in charge of this phase, in order to start the implementation phase (Figure 7). At this stage, the novice software engineer is expected to write a code block with the C# programming language by using the virtual keyboard on the system as a solution to a problem related to the software project. In order for the participant to write the code, the panel in Figure 7 appears. 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 novice software engineer. In addition, there is a text-box at the centre of the panel that enables the novice software engineer 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 also designed to see whether the code written by the novice software engineers is correct.

When the implementation phase is completed, the NPC responsible for the implementation phase of the project warns the novice software engineer that s/he should start the test phase of the project. The novice software engineer who receives this warning goes to the NPC, who is in charge of this phase, in order to start the test phase (Figure 8). In this phase, the NPC tells the novice software engineer 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 [45, 46]. The novice software engineer, who takes the mission from the NPC, starts to do code review on the code, which is shown in Figure 8. This panel is constituted by two different sub-panels. In the left sub-panel, the code block that is entered by using the scenario generating program and which should be tested by the novice software engineer is displayed in a label. 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 novice software engineer should find the correct ones and also match them correctly. Hence, there are two challenges in this phase.

When the test phase is completed, the NPC warns the novice software engineer that s/he should start the maintenance phase of the project. The novice software engineer who receives this warning goes to the NPC, who is in charge of this phase, in order to start the maintenance phase (Figure 9). The NPC, who is responsible for the maintenance phase of the project, congratulates the novice software engineer as s/he is in the final stage of the project and tells the novice software engineer his/her duties in this phase. In this phase, the novice software engineer is expected to put the tasks added to the system into the correct order. As shown in Figure 9, the tasks are listed in the left of the panel. This structure is similar to the structure of the requirement analysis phase. The novice software engineer has to correctly order the tasks by selecting each of them in this phase.

After finishing this phase, the NPC tells the novice software engineer that the simulation is over. When the simulation is over, the evaluation process will be started for the novice software engineer. It is an important issue because the aim is to increase the level of knowledge of the novice software engineer about software development process. To do this, a file, which consists of the true answers of the tasks and the time that is spent by the novice software engineer in the each phase of SDLC, is generated at the end of the simulation. The novice software engineers 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

(a)

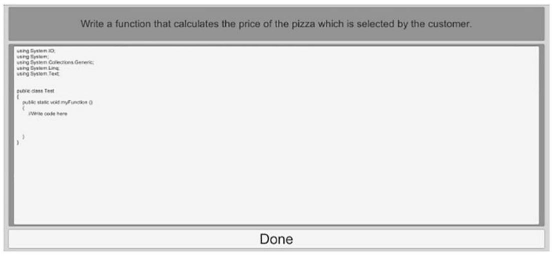


Congratulations, you finished first two tasks. Let's code together.

OK

First Dialogue Box of NPC in the Implementation Phase

(b)



Write a function that calculates the price of the pizza which is selected by the customer.

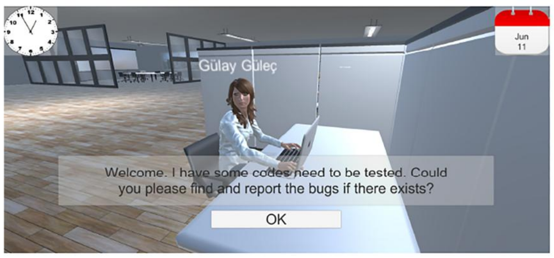
```
using System.IO;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


public class Test
{
   public static void myFunction ()
   {
      //Write code here



   }
}
```

Done

Implementation Phase in Virtual Office

FIGURE 7 Screens of implementation phase in the virtual office environment

(a)

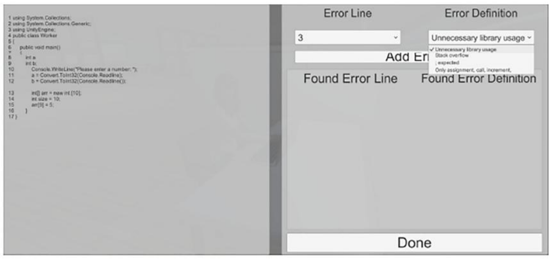


Welcome. I have some codes need to be tested. Could you please find and report the bugs if there exists?

OK

First Dialogue Box of NPC in the Test Phase

(b)



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 public class Worker
5 {
6     public void main()
7     {
8         int a;
9         int b;
10           Console.WriteLine("Please enter a number: ");
11           a = Convert.ToInt32(Console.Readline);
12           b = Convert.ToInt32(Console.Readline());

13           int[] arr = new int [10];
14           int size = 10;
15           arr[9] = 5;
16     }
17 }
```

Error Line: 3
Error Definition: Unnecessary library usage
✓ Unnecessary library usage
Stack overflow
; expected
Only assignment, call, increment,

Test Phase in Virtual Office

FIGURE 8 Screens of test phase in the virtual office environment

way, their abilities, knowledge and experience levels can be improved. In order to understand whether this system is useful and beneficial for the novice software engineer, a set of tests was organised with the novice software engineers. The next section will introduce the applied tests and analysis of these tests.

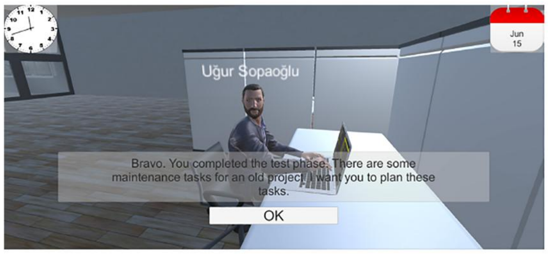
5 | ANALYSIS AND TEST RESULTS

5.1 | Pre-test

At the beginning of the training period, a pre-test was administered to the 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, 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 was taken from the exam organised 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, 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 Figure 10 for experimental group and Figure 11 for control group. When the average scores by the members in both groups were calculated in all parts of the test, the results are shown in Table 1.

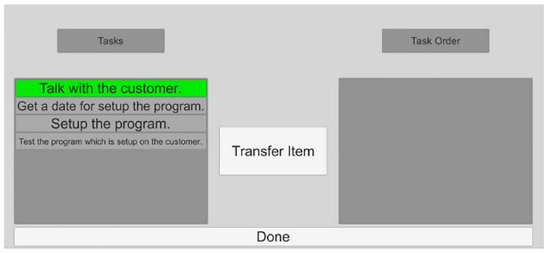
(a)



Bravo. You completed the test phase. There are some maintenance tasks for an old project. I want you to plan these tasks.

First Dialogue Box of NPC in the Maintenance Phase

(b)



Talk with the customer.
Get a date for setup the program.
Setup the program.
Test the program which is setup on the customer.

Maintenance Phase in Virtual Office

FIGURE 9 Screens of maintenance phase in the virtual office environment

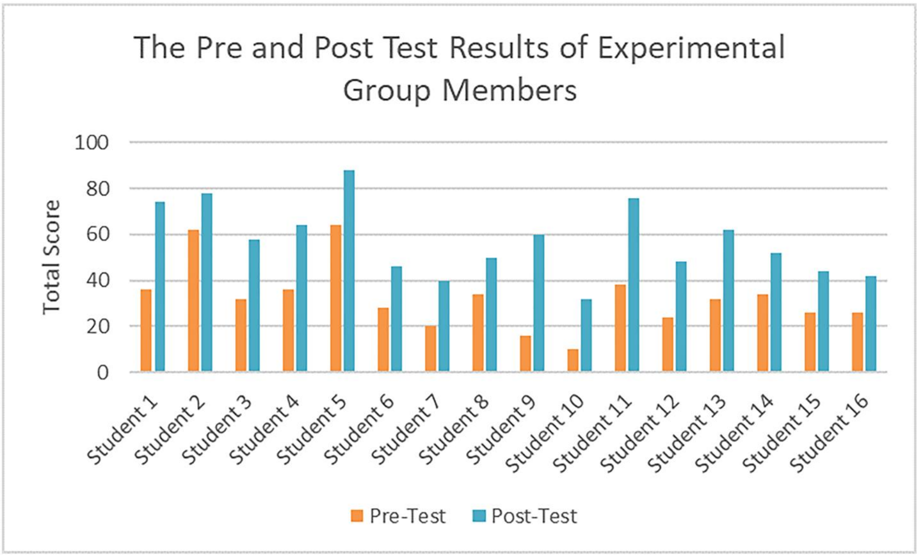


FIGURE 10 Total scores of the students for experimental group in pre- and post-test

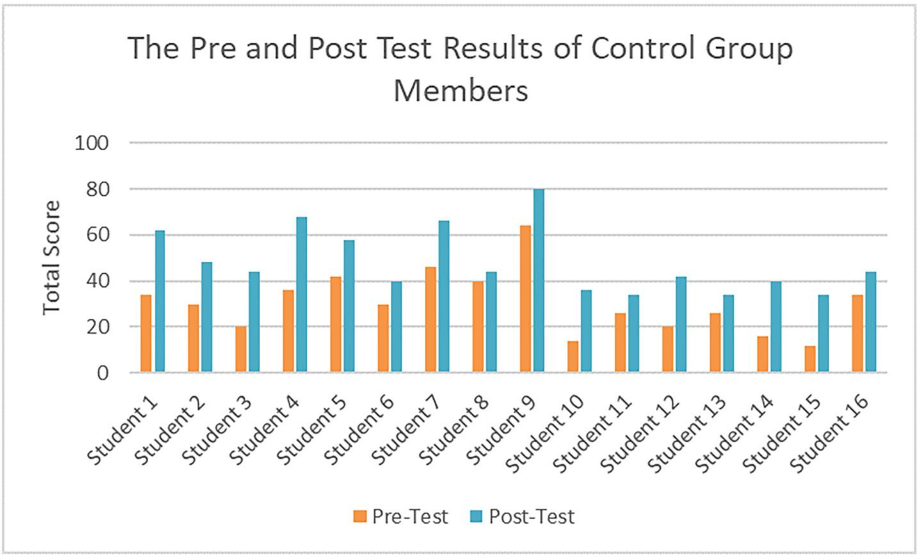


FIGURE 11 Total scores of the students for control group in pre- and post-test

TABLE 1 Average scores of both groups for all parts in pre-test

Group	Req. analysis	Design	Imp.	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

When the quantitative data was analysed, 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 in the other parts. Thus, the participants have some trouble when they identify the requirements of the system and find the actors and 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 in the experimental groups is 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 applied at the courses given in general, at least 60 points have to be collected to pass the course. When the results of the pre-test are analysed 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 10, and 1 student in the control group as shown in Figure 11. In total, the number of students, who could pass the test, is 3 out of 32 students.

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 was also taken from the exam organised 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 for 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 organised with 32 students selected for the training programs as experimental (16) and control groups (16). The post-test results of both groups are shown in Figure 10 for the experimental group and Figure 11 for the control group. The average scores by the members in both groups were calculated in all parts of the test and the results are shown in Table 2.

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 for 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 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. When the students' numerical values are examined in the post-test, there were seven students who provided the minimum criterion that is required to be successful in the exam from the experimental group of 16 students, as shown in Figure 4 and 10 students, who have passed the test, in the control group out of 16 students, as illustrated in Figure 11. 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 analysed, 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, 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 12. 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

TABLE 2 Average scores of both groups for all parts in post-test

Group	Req. analysis	Design	Imp.	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

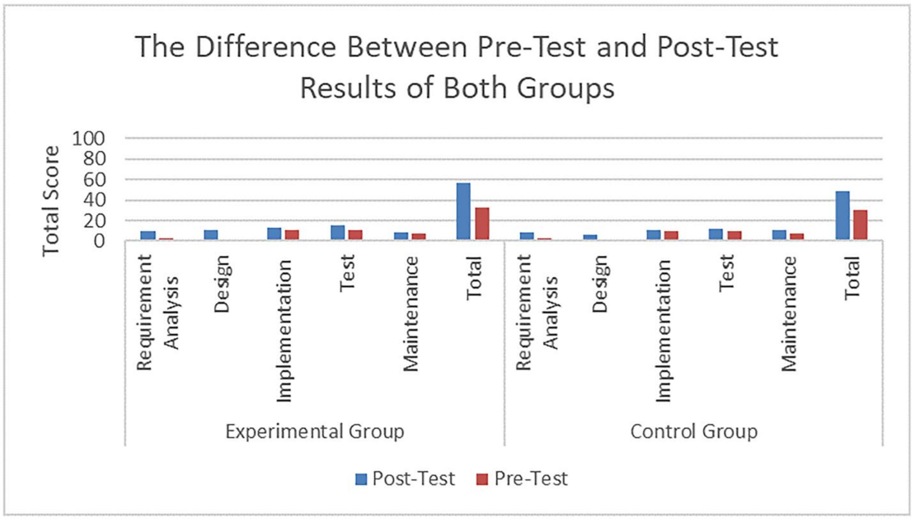


FIGURE 12 The difference between pre-test and post-test results of both groups

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-tests* have been used since the groups were randomly constituted. However, the hypothesis of the study should be identified before calculating the test statistic. This study states the null hypothesis as the difference between the population mean 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.

After defining the hypothesis of the study, the value of the test statistic 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 3), have been entered to Minitab in order to figure out whether there is a significant difference between the groups. 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 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 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.

TABLE 3 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

5.4 | PQ and ITQ

The sense of presence is one of the critical factors to establish successful virtual environments [47]. 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 [41]. Hence, this study has benefited from both PQ and ITQ. 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 organised 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 13.

When the scores illustrated in Figure 13 are evaluated over 100, there are four students who tend to be immersed by 90% and above; one student who tends to be immersed between 80% and 90%; eight students who tend to be immersed between 70% and 80% and three students who tend to be immersed below 70%. This means that the participants tend to be immersed at an average of 74.55%. Actually, 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 organised 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

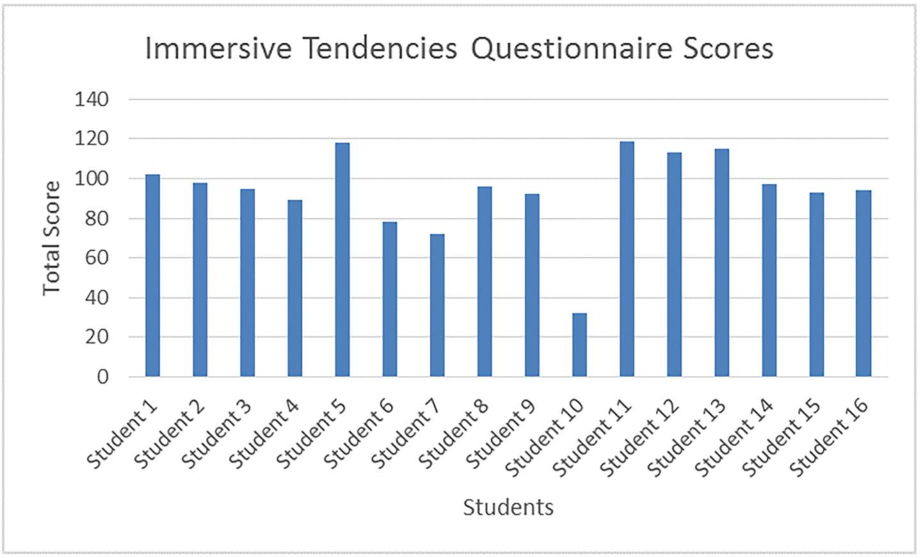


FIGURE 13 ITQ scores of the participants

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 14.

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% included in the virtual environment. Six students felt a strong immersion sense to our virtual office environment at 85% and above. The second important point is that although the PQ scores obtained from the last session is less than the PQ scores obtained from the first session, the difference between these scores is 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%. This score shows that the participants have likened the virtual office environment to the real office environment about 77.06%. 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 organising PQ and ITQ with the participants. The numerical values obtained from these tests were also analysed by applying *two sample t-tests* in order to get valid results in terms of statistic. As was mentioned in the Section 3, '*a quantitative research approach supported by validation interviews*' 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 organised with the experts in order to obtain their comments about the system. The summary of these interviews is 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*'.
- '*Thanks to this environment, we can practice the software development processes of different projects. This tool may be a supportive training tool to the internships and graduate projects since we have a chance to experience the software development processes during the tasks occurred in these activities*'.
- '*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 educate 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 organised 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*

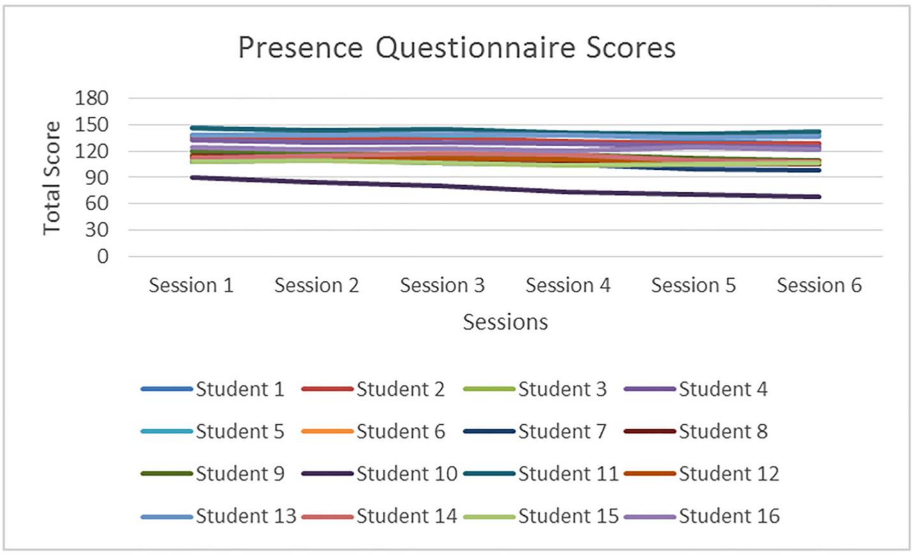


FIGURE 14 PQ scores of the participants

help us get used to work. Sometimes it is hard to ask people so many questions. Thanks to this application, we can complete the tasks with no need to being annoying'.

- *'I have some problems 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'.*

6 | CONCLUSIONS

6.1 | Validation of the proposed training environment

The potential threats of this study, which may affect the validity of the study, were identified in Section 3. In order to deal with these threats, some procedures were performed as follows:

- **Construct Validity ⇒ Qualitative Measure:** A set of semi-structural interviews was organised by both the lecturers and the students.
- **Construct Validity ⇒ 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 VR-SODEF.
- **Internal Validity ⇒ 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 of the students was eliminated since their knowledge levels were almost the same at the beginning of the study.
- **Internal Validity ⇒ 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, the students in the control group 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 min per week. Hence, these students spent as much time as the students in the experimental group to study software engineering subjects.
- **Internal Validity ⇒ Instrumentation Effect:** refers any change in the system during the training period. The designed system did not change during the training period.
- **External Validity ⇒ 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 provide the reliability 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.

6.2 | 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 follows:

- 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 informed several times not to review the software engineering topics from other resources. They have confirmed that they did not study these concepts from other resources; however, they may have studied.

In addition to above limitation, while the codified practices are consistent with established best practice, it is nevertheless the case that in making decisions regarding software projects, we are necessarily dealing with non-deterministic systems, which seem to be amethodical in nature [48] and which in fact may be so complex as to be an instance of a complex adaptive system [23]. Therefore, there is not likely to be any single absolutely and universally acceptable '*correct*' action for each and every scenario. To some extent, in managing complex software projects, we are dealing with some amount of uncertainty, and the logic-based reasoning implementation, which we have provided, assists in overcoming this challenge through the use of logical reasoning. We suggest that it is through the introduction of various logic-based alternative pathways—such as in our virtual environment—that the broader complexity of the software development process can be explored this enabling improved trainee outcomes.

6.3 | Reviewing the research questions

In this section, the research questions from Section 1 are discussed in the light of the results obtained from the user experiences.

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 useful adjunct 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 with the students. The results of these tests indicate 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 indicated that they have a chance 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 organised 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 can be 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 key role in the success of the individuals on a topic [49]. 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 with them. 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.

6.4 | Discussion and conclusion

This study proposes a training environment for novice software engineers in order to increase both their level of knowledge and experience in a similar environment to the real environment, and with the benefit of simulation-led learning rather than making mistakes in a live operational environment. For this purpose, a VR Based Software Development Framework (VR-SODEF), which contains HTC Vive, was produced. This VR-SODEF comprised of a run time configurable virtual office environment for use in teaching the software development process to individuals intending to work in this area. Prior to developing the VR-SODEF, a literature review was conducted to establish the background of the study, to identify the problems that lead to software project failure or non-completion, and to find related studies, which report on earlier attempts to reduce this problem. Firstly, the literature survey provided definitions for software, software development and software engineering. After defining these concepts, the reasons why it is necessary to develop software projects within certain logical frameworks was explored so as to show the importance of software development methodologies. The background exploration contributed an explanation as to why there are several different software development methodologies, each presenting with an assortment of benefits and limitations. And while different software development methodologies exist and can produce successful software products, it was shown in the literature that most software projects do not fully satisfy customer requirements and expectations, and where project failures arise, negative impacts arise for various parties: companies, individuals and even national governments. It was further found that one of the most prevalent reasons for software project failure is the lack of knowledge and experience of the individuals working on software development projects. The final stage of the literature survey involved a detailed analysis of works focused on teaching individuals about software engineering concepts.

After analysing the studies in the literature, it was determined that there are four main limitations related to earlier reported studies (i.e. where VR has been utilised for educational purposes). These are as follows:

- **Large disparity between the virtual world and the real environment**
- **Focus of virtual world tasks is too narrow**
- **Insufficient numbers and variations of simulations**
- **Absence of AI-enabled non-player characters**

The aim of this study was to develop an effective VR-based software development process training tool, which addresses the limitations of earlier related works. To achieve this aim, 2 different applications were developed in the scope of this study: A Scenario Generator and a Virtual Office Environment. These two applications combined to extend the effectiveness of VR-based software process education beyond previous works, specifically addressing identified limitations in earlier efforts in this domain. In terms of the disparity between the virtual world and the real environment, the 3D virtual office environment utilised a head-mounted display that produces an interactive training environment, which is similar to the real environment. In this environment, trainees can interact with virtual workers, who are NPCs working in the same company, and this has been shown to be useful in learning their role and mission on the software project. In addition, the participants can also interact with the elements in the virtual office environment such as accessing and reading the project definition document. This provides the participants with the opportunity to expose themselves to an office-like environment and learn to overcome project issues thorough repeated experience. The perceived effect for trainee participants was measured using establish state-of-the-art interviewing techniques for simulated presence evaluation (a PQ and an ITQ).

Steps were also take to address the issue in relation to the focus of virtual world tasks being too narrow through the implementation of the '*Virtual Office Environment*' and '*Scenario Generator*'. Specifically, the Scenario Generator program has the ability to produce an XML file, which contains the missions related to the each phase of SDLC, which is achieved through the construction of an ontological meta-language. Using this program as a virtual project manager or the team leader, trainees can simulate involvement in every phase of the software development process. The Virtual Office Environment application parses the XML file produced by the Scenario Generator, thereby bringing the scenario to life. Hence, the participant can experience the project scenario generated by the authorised person of the company via the virtual office environment. This XML-based scenario configuration is both flexible and easily extensible. This extensibility permits the introduction of a wide range of scenarios as desired by the program administrator/educator/trainer without the need to change existing program code or introduce new application code, thereby resolving the issue related to insufficiency in numbers or variations of simulations that was observed in previous related implementations.

Our Virtual Office Environment also includes 20 non-player characters (NPCs) that are controlled by cased-based reasoning logic. These NPCs may assign training/education missions to trainees, and provide automated feedback to the participants concerning their choices. Five of the NPCs have been seeded with extensive information on correct/recommended actions based on existing best practices, thereby providing the participants with virtual colleagues from which they can obtain appropriate and important experienced-based knowledge.

As can be seen, our training framework does not just develop a virtual learning environment, it does so by systematically improving upon the limitations noted in previously reported attempts. Different testing methods were organised in order to evaluate our training platform achieves and 32 students, currently studying at the Computer Engineering Department, participated in formal user testing. The students were randomly divided into experimental and control groups, with members of the experimental group having access to the software engineering training platform (but not to any other learning resources). Control group members were only allowed to use traditional resources such as books, videos or presentations. Tests were administered prior to and immediately following the training period, so as to establish the effectiveness of both training strategies. No significant difference between the knowledge levels of both groups was observed at prior to training commencement. However, following completion of the training program, software engineering knowledge levels witnessed a dramatic increase in both groups, with the experimental group demonstrating the best learning outcomes. This, we suggest, supports the case for further development of virtual learning platforms, though we further emphasise that it may be of significant importance to address the four limitations that we have identified from earlier works.

To further examine the effectiveness of the virtual training environment, a set of semi-structured interviews were also organised with the participants; the results from which indicate that the platform is perceived as being a valuable training tool. As a further measure of the effectiveness of the virtual training environment, PQ and ITQ were conducted; the results from which indicate that the participants felt that the virtual environment offered a 77% fit when compared with expectations of a real office atmosphere. Thus, we consider a significant success as virtual environments are merely virtual and, therefore, cannot presently be expected to provide a 100% fit with a real-world setting. As a conclusion, the findings of this study demonstrate that our VR training platform can be efficiently used in the training of participants about software engineering concepts; moreover, VR can be more effective than traditional training techniques in matters, which typically require experience for knowledge acquisition. For future work, it is planned to include software practitioners who are currently working in the software companies into the virtual environment to figure out the efficiency of the system on the experienced personnel.

DATA AVAILABILITY STATEMENT

Data available on request from the authors.

ORCID

Ulas Gulec https://orcid.org/0000-0002-6084-3590

REFERENCES

1. McConnell, S.: Who needs software engineering? IEEE Softw. 18(1), 5–8 (2001)
2. Tassey, G.: reportThe Economic Impacts of Inadequate Infrastructure for Software Testing, Tech. rep. National Institute of Standards and Technology (2002)
3. Platz, W.: Software Testing Tools for Continuous Testing Webpage. https://www.tricentis.com/wp-content/uploads/2018/01/20180119_Software-Fails-Watch_Small_Web.pdf (2018). Accessed 14 Oct 2018
4. Gulec, U., Yilmaz, M., Isler, V.: A literature survey: is it necessary to develop a new software development methodology for virtual reality projects? J. Univers. Comput. Sci. 23(8), 725–754 (2017)
5. Ceschi, M., et al.: Project management in plan-based and agile companies. IEEE Softw. 22(3), 21–27 (2005)
6. Clarke, P., O'Connor, R.V., Leavy, B.: A complexity theory viewpoint on the software development process and situational context. In: Proceedings of the International Conference on Software and Systems Process, pp. 86–90. ACM (2016)
7. Patle, D.S., et al.: Operator training simulators in virtual reality environment for process operators: a review. Virtual Real, 1–19 (2018)
8. Bhagat, K.K., Liou, W.-K., Chang, C.-Y.: A cost-effective interactive 3D virtual reality system applied to military live firing training. Virtual Real. 20(2), 127–140 (2016)
9. Huang, H.-M., Liaw, S.-S., Lai, C.-M.: Exploring learner acceptance of the use of virtual reality in medical education: a case study of desktop and projection-based display systems. Interact Learn. Environ. 24(1), 3–19 (2016)
10. Nazligul, M.D., et al.: Interactive three-dimensional virtual environment to reduce the public speaking anxiety levels of novice software engineers. IET Softw. 13(2), 152–158 (2018)
11. Stinson, C., Bowman, D.A.: Feasibility of training athletes for high-pressure situations using virtual reality. IEEE Trans. Visual Comput. Graph. 20(4), 606–615 (2014)
12. Virvou, M., Katsionis, G.: On the usability and likeability of virtual reality games for education: the case of VR-engage. Comput. Educ. 50(1), 154–178 (2008)
13. Wu, H.-K., et al.: Current status, opportunities and challenges of augmented reality in education. Comput. Educ. 62, 41–49 (2013)
14. Huang, H.-M., Rauch, U., Liaw, S.-S.: Investigating learners attitudes toward virtual reality learning environments: based on a constructivist approach. Comput. Educ. 55(3), 1171–1182 (2010)
15. Merchant, Z., et al.: Effectiveness of virtual reality-based instruction on students' learning outcomes in k-12 and higher education: a meta-analysis. Comput. Educ. 70, 29–40 (2014)
16. Blume, F., et al.: Do students learn better when seated close to the teacher? A virtual classroom study considering individual levels of inattention and hyperactivity-impulsivity. Learn. Instr. 61, 138–147 (2019)
17. Domingo, J.R., Bradley, E.G.: Education student perceptions of virtual reality as a learning tool. J. Educ. Technol. Syst. 46(3), 329–342 (2018)
18. Larmuseau, C., et al.: The relationship between acceptance, actual use of a virtual learning environment and performance: an ecological approach. J. Comput. Educ. 5(1), 95–111 (2018)
19. Pellas, N., et al.: Augmenting the learning experience in primary and secondary school education: a systematic review of recent trends in augmented reality game-based learning. Virtual Real, 1–18 (2018)
20. Ibáñez, M.-B., Delgado-Kloos, C.: Augmented reality for stem learning: a systematic review. Comput. Educ. 123, 109–123 (2018)
21. Feng, Z., et al.: Immersive virtual reality serious games for evacuation training and research: a systematic literature review. Comput. Educ. 127, 252–266 (2018)
22. Gulec, U., et al.: 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)
23. Clarke, P., et al.: Exploring the relationship between software process adaptive capability and organisational performance. IEEE Trans. Softw. Eng. 41(12), 1169–1183 (2015)
24. Rashid, M., Clarke, P.M., O'Connor, R.V.: A systematic examination of knowledge loss in open source software projects. Int J. Inf. Manag. 46, 104–123 (2019)
25. Clarke, P., O'Connor, R.V., Yilmaz, M.: In search of the origins and enduring impact of agile software development. In: Proceedings of the 2018 International Conference on Software and System Process, pp. 142–146. ACM (2018)
26. Clarke, P.M., et al.: Refactoring software development process terminology through the use of ontology. In: European Conference on Software Process Improvement, pp. 47–57. Springer (2016)
27. Sauberer, G., et al.: Do we speak the same language? Terminology strategies for (software) engineering environments based on the Elcat model-innovative terminology e-learning for the automotive industry. In: European Conference on Software Process Improvement, pp. 653–666. Springer (2017)
28. Greenwald, S.W., et al.: Comparing learning in virtual reality with learning on a 2D screen using electrostatics activities. J. Univers. Comput. Sci. 24(2), 220–245 (2018)
29. Barr, M.: Student attitudes to games-based skills development: learning from video games in higher education. Comput. Hum. Behav. 80, 283–294 (2018)
30. Baker, A., Navarro, E.O., Van Der Hoek, A.: An experimental card game for teaching software engineering processes. J Syst. Softw. 75(1), 3–16 (2005)
31. Bollin, A., Hochmüller, E., Mittermeir, R.T.: Teaching software project management using simulations. In: 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), pp. 81–90. IEEE (2011)
32. Hainey, T., et al.: Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level. Comput. Educ. 56(1), 21–35 (2011)
33. Rusu, A., et al.: Employing software maintenance techniques via a tower-defense serious computer game, edutainment technologies. In: Educational Games and Virtual Reality/Augmented Reality Applications, pp. 176–184. (2011)
34. Aydan, U., et al.: Teaching ISO/IEC 12207 software lifecycle processes: a serious game approach. Comput. Stand. Interfac. 54, 129–138 (2017)
35. Ye, E., Liu, C., Polack-Wahl, J.A.: Enhancing software engineering education using teaching aids in 3-D online virtual worlds. In: 37th Annual Frontiers in Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE'07, pp. T1E–8. IEEE (2007)
36. Rodriguez, G., Soria, Á., Campo, M.: Virtual scrum: a teaching aid to introduce undergraduate software engineering students to scrum. Comput. Appl. Eng. Edu. 23(1), 147–156 (2015)
37. Parsons, D., Stockdale, R.: 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)
38. Psotka, J.: Immersive training systems: virtual reality and education and training. Instr. Sci. 23(5), 405–431 (1995)
39. Makransky, G., Terkildsen, T.S., Mayer, R.E.: Adding immersive virtual reality to a science lab simulation causes more presence but less learning. Learn. Instr. 60, 225–236 (2017)
40. Buckingham, M.: What great managers do. IEEE Eng. Manag. Rev. 33(2), 3–3 (2005)
41. Witmer, B.G., Singer, M.J.: Measuring presence in virtual environments: a presence questionnaire. Presence. 7(3), 225–240 (1998)
42. Yilmaz, M.: A Software Process Engineering Approach to Understanding Software Productivity and Team Personality Characteristics: An Empirical Investigation. Ph.D. Thesis. Dublin City University (2013)
43. Goseva-Popstojanova, K., et al.: Architectural-level risk analysis using uml. IEEE Trans. Softw. Eng. 29(10), 946–959 (2003)
44. Fernández-Sáez, A.M., Chaudron, M.R., Genero, M.: An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles. Empir. Softw. Eng. 1–65 (2018)
45. Bosu, A., et al.: Process aspects and social dynamics of contemporary code review: insights from open source development and industrial practice at microsoft. IEEE Trans. Softw. Eng. 43(1), 56–75 (2017)
46. Zanjani, M.B., Kagdi, H., Bird, C.: Automatically recommending peer reviewers in modern code review. IEEE Trans. Softw. Eng. 42(6), 530–543 (2016)
47. Barfield, W., Hendrix, C.: The effect of update rate on the sense of presence within virtual environments. Virtual Real. 1(1), 3–15 (1995)
48. Clarke, P., O'Connor, R.V.: The situational factors that affect the software development process: towards a comprehensive reference framework. Inf. Softw. Technol. 54(5), 433–447 (2012)
49. Yu, S., Levesque-Bristol, C.: Are students in some college majors more self-determined in their studies than others? Motiv. Emot. 42(6), 831–851 (2018)

How to cite this article: Gulec, U., et al.: Applying virtual reality to teach the software development process to novice software engineers. IET Soft. 15(6), 464–483 (2021). https://doi.org/10.1049/sfw2.12047