A TASK FLOW DESIGN TOOL FOR SERIOUS GAMES: AN EXTENDED VERSION OF UML-AD (UML-ADE)

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

EDA TOPALOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

IN

INFORMATION TECHNOLOGY SERVICE MANAGEMENT

JULY 2014

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. İbrahim Akman

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Murat Koyuncu

Head of Department

This is to certify that we have read the thesis "A Task Flow Design Tool For Serious Games: An Extended Version of UML-AD (UML-ADE)" submitted by Eda Topaloğlu and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Nergiz Ercil Çağıltay

Supervisor

Examining Committee Members

Assoc. Prof. Dr. Murat Koyuncu _____

Assoc. Prof. Dr. Hadi Hakan Maraş _____

Asst. Prof. Dr. Nergiz Ercil Çağıltay _____

Asst. Prof. Dr. Atilla Bostan _____

Asst. Prof. Dr. Gökhan Şengül _____

Date: 18.07.2014

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Eda Topaloğlu

Signature:

ABSTRACT

A TASK FLOW DESIGN TOOL FOR SERIOUS GAMES: AN EXTENDED VERSION OF UML-AD (UML-ADE)

Topaloğlu, Eda

M.S., Information Technology Service Management

Supervisor: Asst. Prof. Dr. Nergiz Ercil Çağıltay

July 2014, 77 pages

By improving technology, serious game is widely used by several areas such as defense, science, exploration, health care, emergency management, military management, city planning, engineering, religion and politics. However, development and design of these serious games is more important than playing them. Therefore, in order to create successful serious games, involving domain experts and end-users into the development process is important and necessary. During the development process, communication among those people is a very critical factor to better design and development of the game features according to the expected outcomes. In this study, UML Activity Diagram (UML-AD) is analyzed to represent the scenario flow of a serious game and document the serious game design. Additionally, for the design of serious game, an extension of UML-AD, named as UML-Activity Diagram Extended (UML-ADE), is proposed. The main purpose of the UML-ADE model is to improve the level of understandability of the UML-ADE with both the domain-experts, technicians and end-users. Therefore, the understandability level of the proposed UML-ADE model by technicians is investigated, evaluated experimentally. Results of this study show that the understandability level of proposed UML-ADE model is significantly higher than that of UML-AD.

Keywords— Serious Game; Unified Modelling Language; Activity Diagram; Activity Diagram Extended; Defect Difficulty Level; Defect Detection Performance

ÖZ

CİDDİ OYUNLAR İÇİN İŞ AKIŞ TASARIM ARACI: GENİŞLETİLMİŞ UML-AD (UML-ADE)

Topaloğlu, Eda

Yüksek Lisans, Bilgi Teknolojileri Hizmet Yönetimi

Tez Yöneticisi: Yrd. Doç. Dr. Nergiz Ercil Çağıltay

Temmuz 2014, 77 sayfa

Gelişen teknolojiyle birlikte ciddi oyunlar, savunma, bilim, keşif, sağlık hizmetleri, acil durum yönetimi, askeri yönetim, şehir planlama, mühendislik, din ve siyaset gibi bir çok alanda yaygın olarak kullanılmaktadır. Ancak, ciddi oyunların geliştirilmesi ve tasarımı onları oynamaktan daha önemlidir. Bu nedenle, başarılı bir ciddi oyun geliştirebilmek için, yazılım geliştirme sürecinde alan uzmanları ve son kullanıcıların dahil edilmesi önemli ve gereklidir. Yazılım geliştirme süreci boyunca, istenen sonuçlara göre iyi bir oyun tasarımı ve gelişimi sağlamak için, insanların birbirleri ile olan iletişimi kritik bir faktördür. Bu çalışmada, ciddi oyundaki senaryo akışını göstermek ve tasarımını dokümante etmek amacıyla UML Aktivite Diyagramı (UML-AD) analiz edilmiştir. Ayrıca, ciddi oyun tasarımı için UML-AD genişletilmiş ve Genişletilmiş UML Aktivite Diyagramı (UML-ADE) önerilmiştir. UML-ADE modelinin temel amacı, alan uzmanları, teknik kişiler ve son kullanıcılar ile birlikte UML-ADE'nin anlaşılabilirlik seviyesini geliştirmektir. Bu kapsamda, teknik kişiler tarafından UML-ADE modelinin anlaşılabilirliği araştırılmış ve deneysel yöntemle değerlendirilmiştir. Bu çalışmanın sonucu UML-ADE modelinin anlaşılabilirliğinin UML-AD modelinden yüksek olduğunu göstermektedir.

Anahtar Kelimeler— Ciddi Oyun; Birleşik Modelleme Dili, Aktivite Diyagram; Genişletilmiş Aktivite Diyagram; Hata Zorluk Derecesi; Hata Bulma Performansı

To My Family

ACKNOWLEDGMENTS

First and foremost, I express sincere appreciation to my supervisor Asst. Prof. Dr. Nergiz Ercil Çağıltay for her guidance, continuous support and insight throughout my thesis study.

I would like to thank Asst. Prof. Dr. Gül Tokdemir and Asst. Damla Topallı for their valuable support and guidance, helping to improve my thesis work.

I would also express my appreciation to examination committee members; Assoc. Prof. Dr. Murat Koyuncu, Assoc. Prof. Dr. Hadi Hakan Maraş, Asst. Prof. Dr. Nergiz Ercil Çağıltay, Asst. Prof. Dr. Atilla Bostan, Asst. Prof. Dr. Gökhan Şengül, for their valuable time, comments and suggestions.

I would also like to thank Atılım University students' and colleagues, who willingly participated in my study, for their valuable time and contributions.

This study is conducted for improving the scenario designs of the educational materials which are developed for neurosurgery education project (ECE: Tubitak 1001, Project No: 112K287) purposes. I would like to thank the support of TUBITAK 1001 program for realizing this study.

Last but not the least; I would like to express my appreciation to my parents for their endless love, support and patience. They have always believed in me and encouraged me all through my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AD	-	Activity Diagram
ADE	-	Activity Diagram Extended
DF	-	Defect Difficulty Level
M-UML	-	Mobility of Unified Modeling Language
OCL	-	Object Contraint Language
PP	-	Participant Performance
SFD	-	Scenario Flow Diagram
UML	-	Unified Modeling Language

CHAPTER 1

INTRODUCTION

History of game is based on ancient times. From the archaeologists' studies we understand that games were part of social lives in earlier time [1]. Nowadays, there are millions of game variations such as racing games, adventure games, serious games, health games, intelligence games, sport games, music games, action games, strategic games, puzzle games and games for kids, love and fun games [2][5].

Game is an activity that people of all ages play [3]. In today's world, computer games, especially for children have become one of the most interesting and entertaining technology environments [4]. On the other hand, the Internet becomes an indispensable part of our lives. Playing via the Internet, online multi-user computer games with people of different ages, cultures and geographies in order to ensure interaction and communication has played an important separation [4]. Internet access and applications has a great contribution to diverse application areas and the spread of computer games [4]. Games nowadays provide environments such that thousands of people come together for a common goal of virtual worlds [4].

Most of the children playing computer games consider computer games as an activity of leisure entertainment [4]. According to Mangir (1993), games have potential to improve children ability, creative potential of mind, the language skills and social skills as well as emotional and motor skills [10]. In addition, children learn the basic social rules such as obeying game rules, sharing with friends, helping each other and establishing positive relationships with the environment, rules, and taking responsibility to respect the rights of others [10]. For adults, game is an entertainment to relax themselves during their leisure times [3]. Besides these features of computer games, in the literature, several studies show benefits of game-based learning environments [4]. These studies show how computer games are

represented educational content by using the most effective and appropriate learning environments to integrate adaptation and concentration [4].

Furthermore, by improving technology, some games are developed for patients who lose a variety of movement skills. These games are effective to rehabilitate the patients and improve patients' cerebral function. Hence, these games are designed not only for entertainment and learning but also rehabilitation purposes [50]. These groups of games are called as serious games that are designed for entertainment and important purposes [5]. Another definition of 'serious gaming' offered by Zyda (2005) is, "*a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives*" [5]. When the studies in the literature are analyzed, many different types of serious games can be found as follows [5][2]:

- Educational games
- Healthy games
- Edutainment
- Training and Simulation
- Games for Good
- Games for Change
- Virtual Reality
- Alternative Purpose Games
- Digital Game-Based learning
- Immersive Learning Simulations
- Social Impact Games
- Persuasive Games
- Synthetic Learning Environments

Serious game is used at several areas such as defense, scientific, exploration, health care, emergency management, military management, city planning, engineering, religion, and politics [6][58]. It can also be used at every level of education, at all kinds of schools and universities around the world [6]. Serious games have potential to improve lots of different development skills such as strategic thinking, planning,

communication, collaboration, group decision making, negotiating and data-handling skills [19]. Games provide opportunities for learners to experiment and to make mistakes in a protect environment without risk of life, limb or identity [5]. Also, studies show that, serious games increase inventive thinking, high productivity and effective communication [7]. In some serious games, players work together to accomplish tasks and goals as a virtual team which unfold over time as a story and create a permanent memory for the players [5]. We can group application areas of these serious games in two categories: educational and health. Below the games according to these categories are summarized.

1.1. Educational Serious Game

Educational serious game can be called as edutainment [8]. It is an effective tool for improving learning in both adults and children. All digital games, simulations and virtual worlds involve learning, but there is in balance between learning what the game is designed to teach and learning the game [8]. Serious game is the accepted term for games with an educational intent. They need to be engaging, although not necessarily fun, while the learning can be implicit or explicit. Educational games are designed to teach students about certain subjects, expand concepts, reinforce development, understand an historical event or culture, or assist them in learning a skill as they play [8]. These games provide an immediate and challenging visual feedback within a fun safe virtual environment [8].

1.2. Health Serious Game

The number of serious games in the health sector is also growing day by day. These games have great potential for patients to prevent and rehabilitate them [9]. Serious games are pedagogical platforms [9]. Therapy of cancer, diabetes, asthma, burns, and brain injuries profits from serious games. Serious games in health help to increase motivation, reduce anxiety on patients by entertaining and promote physical activity. Video game technologies are increasingly used nowadays. In health care, they provide examples of innovative ways to improve skills for patients [9][59]. Some of the advantages of serious health games can be summarized as below:

- Training and simulation

- Cognitive functioning
- Control mental and emotional states
- Recovery and rehabilitation
- Diagnosis and treatment of mental illness/mental conditions
- Distraction therapy

1.3. Problems of Serious Games

In contrast to the studies showing several advantages of serious games, some studies also report failure and uncertain situation in game-like environments [11]. Motivating students is one of the serious problems in education, because learning tasks can be very boring, complex, too easy, or repeated [11]. Another serious problem is enforcement. When students are enforced for gaming, it can create negative effects on students [11]. Studies also show that, there are some factors, affecting the success of those serious games. We believe the design characteristics of games are one of the important factors to improve their benefits in education and health fields.

Every game has rules and components, which define the game [3]. Everything that is in the rules is part of the game whereas everything that is not in the rules does not belong in the game [3]. Additionally, every game has a goal, which means the victory condition or requirement or the strategy needed to win the game [3]. Every game has different properties. For instance, in some games, narrative and storytelling are preferred. Because, it is believed that the narrative and storytelling are more permanent in children's memory. Educational video games are useful activities that supply problem solving, narrative feature, learning-by-doing, and ability [13]. These game design factors are important to improve benefits of games in the educational and health environments. In order to create successful serious games, it is important to involve domain experts who can define the specific features and requirements of the end-users [12]. There should be a balance between education and entertainment for the effectiveness of design educational games [13]. Hence, the input of domain experts is important from inception to design and from initial development to testing [13] processes of serious games. End-user Development refers to methods, techniques, and tools that support end users to create, adapt, or evolve the game

design [16]. End-user Development approves the importance of involvement of different domain experts in the design of effective interactive systems, because domain experts have specific knowledge of the domain which is needed for serious game software development [14] [15]. Also, domain experts should be active during design, development and evaluation phases of such software projects. Additionally, according to a study, different techniques and tools such as programming libraries and authoring tools can be used during the technical development process of educational games [17]. Authoring tools are aimed to make development process more accessible [17]. On the other hand, the technical development of serious games requires the involvement of programmers, visual arts, designers, sound designers and writers [18]. Programmers create the game codes [18]. Sound designers develop everything from music and ambient sounds to character speech and sound effects [18]. Writing involves producing high-level narrative and character treatments, developing back-story, and writing dialogue [18]. All game artists are skilled at basic visual design, including coursework in drawing and painting fundamentals, sculpture, anatomy, physiology, and life drawing [18]. Hence, we understand that in the development process of serious games, several people such as domain experts, end-users, and technical people should be involved in. During the development process communication among those people is a very important factor to better reflect their concerns in the game design and user interface.

Today, there are several tools that can be used to better reflect the serious game design and improve the level of communication among the technicians & domain experts such as UML, Game-Flow-Design, Meta-Design, Meta-Edit, Workflow, Story Flow Diagram, and Storyboard [20] [21] [22] [23] [24] [25]. However there are not many studies showing how to use these tools to better design serious games as well as evaluating if these tools are appropriate for representing the game design. Additionally, there are not many studies evaluating the effectiveness of these tools for serious game design. In this study these design tools are in general described and one of the tools that is used commonly for serious game design, namely the UML Activity Diagram to represent the scenario flow of a serious game is analyzed.

Additionally, the limitations of UML Activity Diagram (UML-AD) representation for the serious game design are discussed. Based on these limitations, an extended

version of UML-AD is proposed and named as UML Activity Diagram Extended (UML-ADE). The benefit of UML-ADE representations according to the UML-AD is also analyzed experimentally.

The next chapters of thesis are organized as follows. Chapter 2 reviews the literature about serious game design and tools that can be used to represent story/scenario flow of these games, and discusses their limitations to better represent this information. Chapter 3 describes the proposed methodology of this study and defines the proposed extended version of UML-AD (UML-ADE). Chapter 4 describes the research methodology of this study and discusses the proposed extended version of UML-AD (UML-ADE). Chapter 5 represents the results of this study to better understand the effect of UML-ADE for serious game design, to improve the level of communication among people involved in the game development process. And finally, Chapter 6 represents the discussion/conclusion of the study.

CHAPTER 2

BACKGROUND OF THE STUDY

In the literature there are several tools that can be used to document the serious game design and used for communication purposes between the domain experts and the technical developers. Among those, the widely used ones are described below.

2.1. Serious Game Design Tools

There are several tools developed to support creating better design for serious games. These tools are used for different purposes in game design such as representing the game flow, helping the domain experts to create their own games without coding and helping both the domain experts and the technical people for improving their communication level. Among those, the Game-Flow and Game-Flow-Design are the tools developed to design the flow in the game.

2.1.1. GameFlow

The aim of Game-Flow is to develop and validate a model of player enjoyment in games, which are based on flow [31]. This tool involves extending the 8 elements of flow to model player enjoyment in games by using the heuristics in the games usability and user-experience literature [31]. The 8 core elements are concentration, challenge, skills, control, clear goals, feedback, immersion, and social [31]. For each element, the Game-Flow model includes an overall goal and a set of central criteria that can be used to design and evaluate games with respect to player enjoyment [31].

2.1.2. Game-Flow-Design

Game-Flow-Design is a tool to provide a detailed design technique for modeling individual computer game levels [21]. The purpose of this tool is to outline the flow

of game play between scenes and within scenes in a given game level and modeling game in scripted games [21]. Computer game-flow design tool extends the use-case diagrams of UML technique that are commonly used to make them more applicable for the computer game design [21].

2.1.3. Meta-Design

Meta-Design supplies End User Development (EUD) activities, which create socio-technical conditions to adapt the users [22]. Recently a definition for Meta-Design has been proposed in [28] [29]: "*Meta-design characterizes objectives, techniques, and processes for creating new media and environments allowing 'owners of problems' (that is, end-users) to act as designers. A fundamental objective of meta-design is to create socio-technical environments that empower users to engage actively in the continuous development of systems rather than being restricted to the use of existing systems*".

2.1.4. Meta-Edit

Meta-Edit is a domain-specific language, which allows modeling tools without writing a single line of code [23]. Meta-Edit tool provides a meta-modeling language and tool suite to define the method concepts, their properties, associated rules, symbols, checking reports, and generators [23]. The tool increases productivity for product families.

2.1.5. IDE (Integrated Development Environment)

IDE is for authoring and debugging non-linear scenarios to train games [34]. IDE platform supplies software development and normally consists of a source code editor, a compiler and/or interpreter, build automation tools, usually a debugger, and visualization module for 3D environment [34]. It is believed that the IDE has a Visual Programming Language (VPL) and its respective logic engine (interpreter), a scenario-authoring tool (source editor), and a scenario testing and debugging tool (debugger) [34]. The games developed by using these tools have user interface design limitations with the capabilities of the tool. In other words, the professional programming flexibility can never be provided through these tools. Additionally, it requires more time and effort of domain experts.

In this concern, there are some tools that can be used by both the domain experts and the technicians to improve their level of communication considering the game design. For example Game-Flow-Design, e-Adventure, Workflow, and UML Activity Diagrams can be considered in this category.

2.1.6. <e-Adventure>

The <e-Adventure> tool is an instructor-oriented platform for the development of educational point-to-point adventure videogames [32]. The <e-Adventure> platform supports the educational features through its built-in assessment and adaptation mechanisms and allows instructors to define aspects of games, which are educationally relevant. [33]. <e-Adventure> platform also allows authors to specify assessment rules which are in different states and identifies relevant patterns of behavior to be matched on the state space of game [33].

2.1.7. Workflow

Workflow process includes how something gets done, how the applications work together and what the process might be for different kinds of projects [35]. In educational games, the workflow supposes availability of an extensible game framework to supply an easy design of word and logic educational games and includes the instructor role as a role of key importance of the way games will be incorporated into an instructional design. [36]. It is believed that to provide effective workflow, development of simple single-user educational games can easily be parameterized, instantiated and executed [36].

Among these tools the UML is one of the widely used one for several different purposes. For this reason in the next section this tool is described in detail.

2.2. UML (Unified Modeling Language)

UML system is used for modeling and simulation that presents a meta-model [20]. It is stated that UML is for visualizing, specifying, constructing, and documenting the products of a software-intensive system [26]. UML supports objects, relationships, use cases, scenarios, application, classes, and components [26]. Also, it is believed that the UML is based on Domain Specific Language (DSL) for the meta-model [20].

UML tools support experimentation and evaluation of different design options [27]. UML model includes use case diagrams, class diagrams, object diagrams, component diagrams, deployment diagrams, activity diagrams, sequence diagrams, state machine diagrams, and collaboration diagrams [26][27].

UML representations have been used in serious game design successfully. For instance Cirulis and Ginters report that the UML has diagrams and graphical elements in Virtual and Augmented Reality (VR/AR) worlds [30]. According to them, UML choice allows developing an agent-based simulator easily for the object-oriented framework, engineering, maintenance and extension [30]. In this study agent- based simulations are used to integrate prepared learning scenarios in VR/AR world [30]. According to Cirulis and Ginters the UML is for modeling and planning purposes, and object oriented approach for programming and creation of VR scene [30].

2.2.1. Use Case Diagram

It is believed that the Use Case Diagram is a scenario between user and system that defines event flows as a text format [39]. This diagram is related with what the system does [39]. Use Case is preferred to document the behavior of software systems and specifies the requirements of software systems [40]. It is thought that Use Case is used to show the system behaviors to determine requirements and analyze stages [39].

2.2.2. UML Activity Diagram

It is believed that UML Activity Diagram (UML- AD) supports verification of flow of action models and translates an activity diagram into input format [37]. Also, the purpose of this tool is to model system-level functions and procedural flow that is part of larger activity [38]. It is thought that the UML Activity Diagram represents the performance of actions and sub activities that is a special case of state diagram [41]. Since the UML-AD is the most commonly used one, in this study, the features of UML-AD are analyzed in detail below.

An Activity Diagram can be defined by one of the Unified Modeling Language (UML) [42]. UML (Unified Modeling Language) is the last version that is used and

modified to know as an activity diagram and flow modeling in Figure 2.1 [48]. The Activity Diagram models the actual workflow and control flow of behavior in a system [43]. It is thought that the Activity Diagram is the expanded version of the use-case diagram and similar to flow chart, but the small difference is that the Activity Diagram has concurrent operations [44]. It is believed that the purpose of the Activity Diagram is to determine activity flow of a system, analyze use-case diagrams, define the sequence from one activity to another, or describe parallel, branched and concurrent flow of the system [44]. The Activity Diagrams describe work processes and functional processes that have division of work between user and system explicit and short description of function [45].

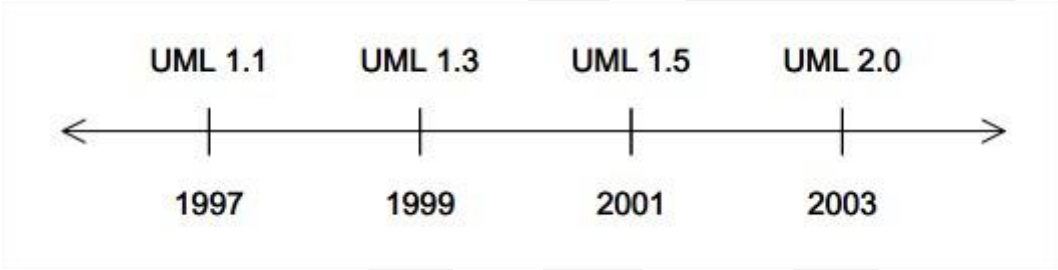


Figure 2.1 – History of UML

In the last two years UML diagram has been used in mobile systems for widely distributed and heterogeneous systems [54][55][56]. Location, mobile object, mobile location, move action and clone action are concepts of UML diagram in mobility. These concepts are built-in mechanisms that are defined by using UML stereotypes, tagged values, and Object Contraint Language (OCL) constraints [55][56]. For modeling mobility, two variants that are responsibility-centered and location-centered are included [56]. Also, UML modeling becomes increasingly important about modeling of agent-based systems [55]. Klein et al. [57] proposed an extension to UML for mobile agents. Mobile agent system is software and has several agents and technical and functional features to develop mobile-agent applications [55]. Mobile agents system provides agent programmers, native resources, agents' acceptance, and logical locations, code/data transferring and computational environment. UML diagram models the several aspects of a software system. Because the UML-AD shows the control flow between actions in activity, in UML-AD diagram, a mobile action/activity includes at least one mobile agent in a different platform [54]. Mobile agents and static locations are not well suited enough for

mobile computing modeling. In order to deal with this problem, UML diagram has been proposed. UML profile includes view of organizational, life cycle, interaction and mobility areas of mobile agents' model [55]. Therefore, UML is widely used in mobile agent systems as well.

Additionally, UML-AD is used as an effective design and Meta model in object oriented program testing, because use of code is complex and important to test, and software test techniques consider insufficient static view of code [46][60]. It is believed that this model supplies simplicity and understandable flow of logic and concurrent activities to generate test cases [46]. Below the main features of the activity diagrams are summarized [46] [47] [48].

2.2.3. Activity Diagram Nodes and Edges

Activity is a coordination of executions of subordinate behaviors. Subordinate behaviors are defined by using control and data flow model. Each activity diagram shows exactly one activity and an activity is modeled as a graph of activity nodes that are connected by edges. There are three different node types in an activity-action, object and control nodes.

2.2.4. Action Nodes

Action nodes are an individual step in an activity. These are basic elements that supply control or data flow to or from other nodes. They determine where the action appears within different namespaces. Action nodes receive an event, control, send signal and provide data. As seen in Figure 2.2, action nodes are represented by a rectangle.

Figure 2.2 – Action, accept event & send signal in UML - AD

2.2.5. Control Nodes

Control Nodes only controls the flow edges and illustrates the control flow of

the activity. It starts an activity node after the previous one is finished. In Figure 2.3, control nodes relationship is represented.

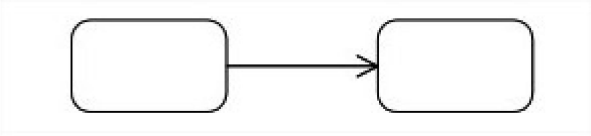


Figure 2.3 – Control Nodes in UML-AD

2.2.6. Object Nodes

It connects only object nodes and data. In Figure 2.4, object nodes relationship is represented.

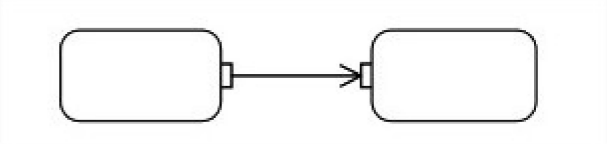
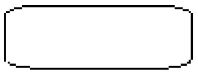


Figure 2.4 – Object Nodes in UML-AD

2.2.7. Details of Simple Control Flows

Simple control flows in UML-AD is used to represent flows in a design. Main factors in this representation are explained below according to the numbers shown in Figure 2.5.

Action: The action starts when a token has arrived at all its incoming flows. When it ends, tokens are sent on all the outgoing flows. In UML-AD, an action is represented by a rectangle like shape (). Action is a single step and has an input and output information. Action has the following components, body, language, local post conditions, and local preconditions.

- **Body** - Specifies the action in detail.
- **Language** - The language of the expression in Body.
- **Local Post conditions** - Constraints that must be satisfied when execution

ends. The goal achieved by the action.

- **Local Preconditions** - Constraints that must be satisfied before execution begins.

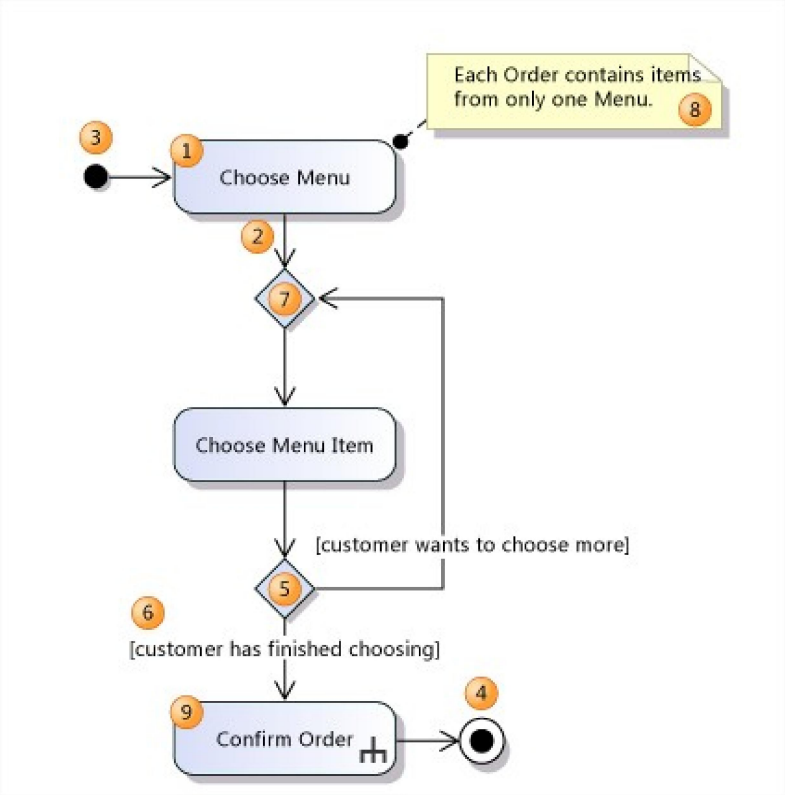


Figure 2.5 – Simple Control Flows in UML-AD

2 **Control Flow** is connector that shows the flow of control between actions. To interpret the diagram, a token flows from one action to the next. Control flow in UML-AD is represented by arrow shape ( ).

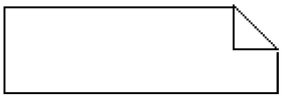
3 **Initial Node:** There is at least one initial node for each activity in UML-AD. When the activity starts, a token flows move to the initial node. Initial node means that the flow starts when the activity is called. Also, in an activity, there can be more than one initial node. Initial node in an UML-AD is represented by a black-circle shape ( ).

4 **Activity Final Node:** In an UML-AD, there can be more than one activity final node. It represents the end to the activity. All flow activity stops. If several parallel flows are presented within an activity, all flows are stopped at the time and the activity final node is reached. Activity final node in an UML-AD is represented by a white-black-circle shape ().

5 **Decision Node** is a conditional branch in a flow. A decision node has one incoming flow and multiple outgoing flows. It accepts tokens in incoming edges. Each outgoing flow has a condition attached to it, which is written in brackets. If a condition is true, the flow proceeds along the appropriate outgoing flow. If no other condition is true, an 'else' outgoing flow can be defined along which the flow can proceed. Decision node in an UML-AD is represented by a diamond shape ().

6 **Guard** is used on the outgoing flows of a decision node. Guard in an UML-AD is represented by brackets ([...]). In the brackets, flow information is written.

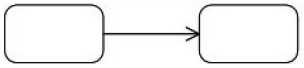
7 **Merge Node** is required to merge flows that were split with a decision node. A control node brings multiple incoming edges and one out coming edge. It is not used to synchronize concurrent flows. Also, it brings together multiple alternate flows. Merge node in UML-AD is represented by a diamond shape ().

8 **Comment** provides additional information. Comment in an UML-AD is represented by a rectangle shape which has a fold ()

9 **Call Behavior Action:** An action that is defined in more detail on another activity diagram. It can be represented with different levels of detail. Call behavior

action in an UML-AD is represented by a rectangle like shape, which has warning sign for detail diagrams (). Call behavior action has the following components as Is Synchronous and Behavior.

- **Is Synchronous** - If true, the action waits until the activity terminates.
- **Behavior** - The activity invoked.

Call Operation Action: An action that calls an operation on an instance of a class. User can use to invoke operations in a model. Each call operation has a unique name that is synchronized with the operation. Call operation action in an UML-AD is represented by 2 rectangles like shapes with transition ().

2.2.8. Concurrent Flows

Concurrent flows in UML-AD are used to represent flows in a design. Main factors in this representation are explained below according to the numbers shown in Figure 2.6.

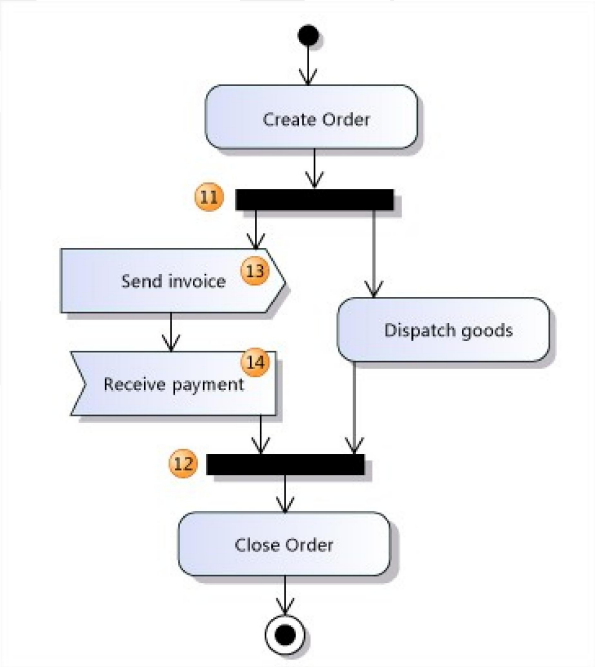
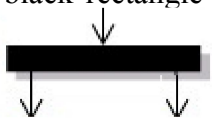
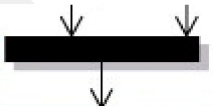
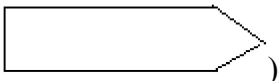
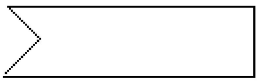


Figure 2.6 – Concurrent Flows in UML-AD

11 **Fork Node** divides an incoming flow into multiple concurrent outgoing flows. A fork has one input and two or more outputs. It is shown as a horizontal and vertical line. Fork node in an UML-AD is represented by a black rectangle shape which, has one input arrow and at least two output arrows ().

12 **Join Node** combines concurrent flows into a single flow. Join node has two or more incoming and one out going flows. It synchronizes multiple flows. Join node is a consolidation of two or more parallel flows. It is shown as a horizontal and vertical line. When the synchronization takes places, the flow proceeds only after all incoming flows. Join node in an UML-AD is represented by a black rectangle shape, which has more than one input arrow and one output ().

13 **Send Signal Action** sends a message or signal to another activity or to a concurrent thread in the same activity. According to the flow it originates from node in the activity diagram. Send signal action in an UML-AD is represented by right arrow like shape ().

14 **Accept Event Action:** Accepting events is an important element for business processes in activity diagrams. It waits for a message or signal before the action can continue. After, the event is accepted. Accept event action in an UML-AD is represented by left arrow like shape ().

2.2.9. Data Flows

Data flows in UML-AD are used to represent flows in a design. Main factors in this representation are explained below according to the numbers shown in Figure 2.7.

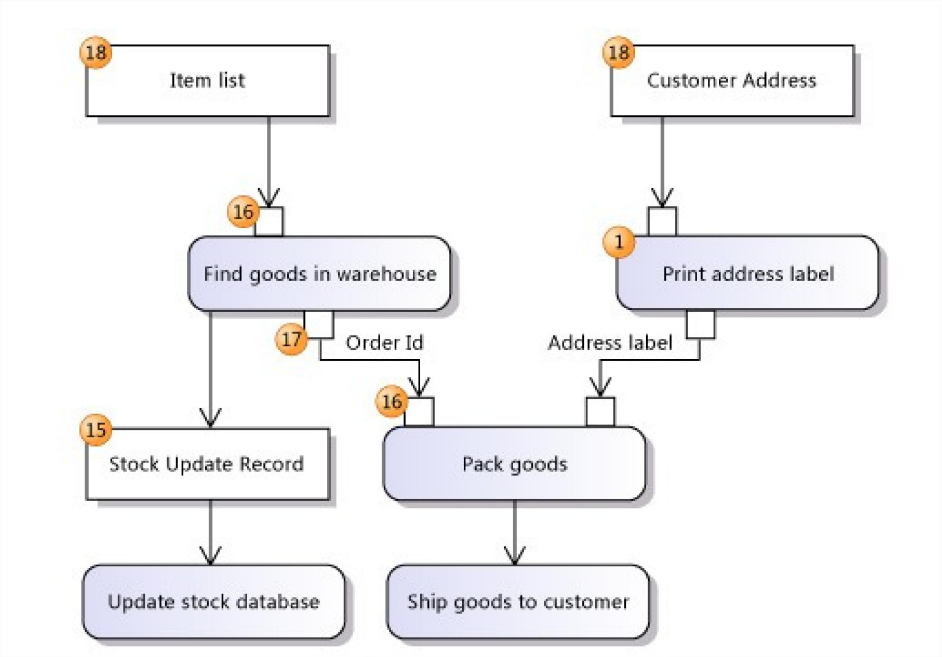
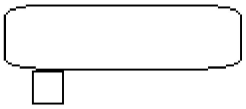


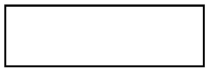
Figure 2.7 – Data Flows in UML-AD

15 **Object Nodes:** An object node describes the data participating in an activity diagram and indicates an instance of classifier. Object nodes are used where objects are flowing form and to somewhere. Object node in an UML-AD is represented by a rectangle shape ( ).Object nodes have components as ordering, selection, upper bound, and type.

- **Ordering** - How multiple tokens are stored.
- **Selection** - Invokes a process, which can be defined in another diagram that filters the data.
- **Upper Bound** - 0 indicates that data must pass directly along the flow; * indicates that data can be stored in the flow.
- **Type** - the type of objects stored and transmitted.

16 **Input Pin:** Input pin shows the input parameters on an action. It represents data that an action can receive when it executes. Input pin in UML-AD is represented by a rectangle like shape within a small input square ( ).

17 **Output Pin:** Output pin shows the output parameters on an action. It represents data that an action produces when it executes. Object flow edges deliver the output values to other actions. Output pin in an UML-AD is represented by a rectangle like shape within a small output square ().

18 **Activity Parameter Node:** It represents which data can be received or produced by the activity. It includes input and output parameters in an activity. Activity parameter node in an UML-AD is represented by a rectangle shape ().

2.2.10. Action Notation

In UML-AD, the system tasks and player tasks can be represented by action notation as shown in Figure 2.8.

Figure 2.8 – UML-AD Action Notation

2.2.11. Swimlane in UML-AD

UML-AD has also a notation that is swimlane. Swimlane means that activities are prepared into vertical and horizontal zones with lines. A swimlane supplies to group activities by the same actor on an activity, or to groups in a single thread. Each action is assigned to the one swimlane. Swimlane has unique name and presents different actor within the same flow [61]. With using swimlanes, activities are mapped and new associations that can be documented in class model are identified in Figure 2.9, UML-AD vertical swimlane notation is shown which are separated into 3 groups in an activity.

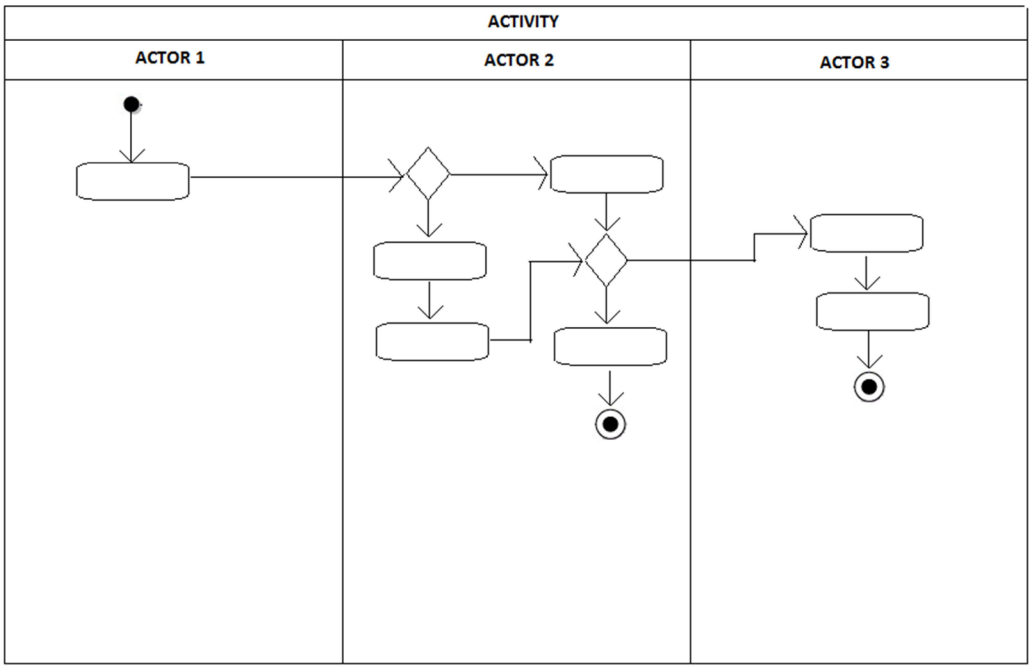


Figure 2.9 – UML-AD Vertical Swimlane Notation for 3 Actors

In Figure 2.10, UML-AD horizontal swimlane notation which is separated into 3 groups in an activity is shown.

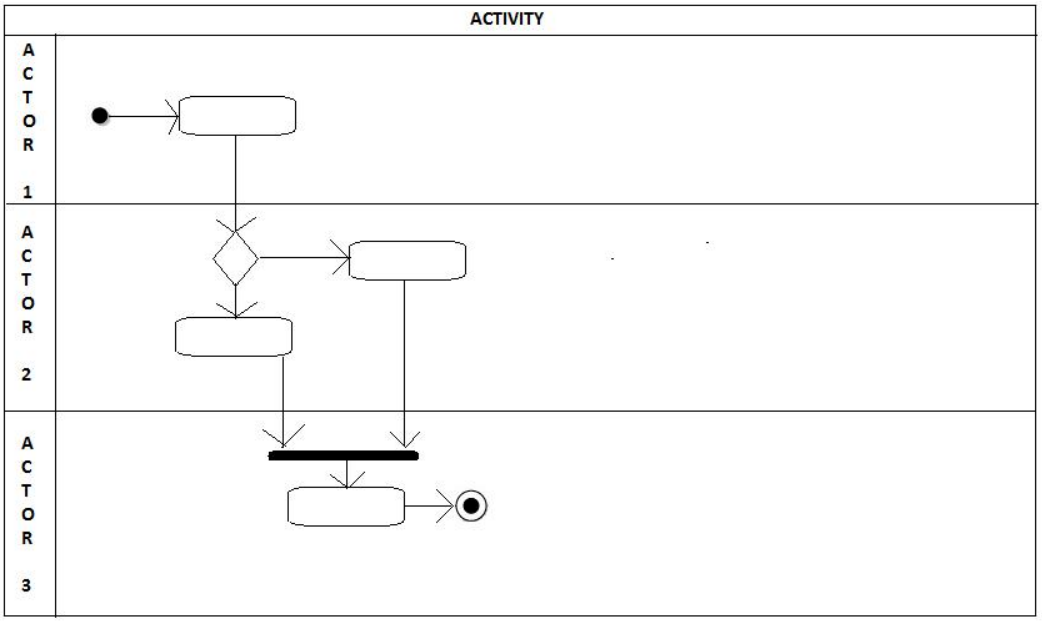


Figure 2.10 – UML-AD Horizontal Swimlane Notation for 3 Actors

Swimlanes divide activity diagrams into sections such as organizations, departments, components, or threads [62]. Therefore, swimlane is a useful method between people

that combine the UML-AD's depiction. However, in swimlane, when the numbers of actors increase, sections increase and diagram understandability can be difficult. Also, to draw a complex diagram can be difficult. Generally, swimlanes are preferred in the organizational responsibility for activity diagrams, but they are not effective and sufficient for designing advanced organizational relationships [61].

CHAPTER 3

PROPOSED METHODOLOGY

In this chapter, proposed methodology of this study is defined. The purpose of this methology is to separate system task and player task in game design documentation and to propose a new model that is UML Activity Diagram Extended (UML-ADE). Because in this study, it is determined that he UML-AD in serious game design has some limitations.

End-user Development refers to methods, techniques, and tools that support end users to create, adapt, or evolve the game design [53]. End-user Development approves the importance of involving different domain experts in design of effective interactive systems, because domain experts have different kinds of knowledge, which is needed to software development skills [14][15]. In other words, to reach a success point in the serious game implementations, it is important to involve domain experts and end-users into the development process. Hence, domain experts are important from inception to design and from initial development to testing [13]. Accordingly, the domain experts should be active in design, development and evaluation of software development processes.

Flow diagrams are necessary for documentation of serious game. The aim of the using flow diagrams in serious game is to better reflect the serious game design, describe serious games in a useful manner and improve the level of communication among the technicians & domain experts.

As described above, the UML-AD representation is a powerful tool to visually design the serious game features. However, in the serious games, measuring performance, success of players and assessment are very important. The player's performance within the game play is usually measured through his/her progress

while performing the task assigned for the player. This information is used for both guiding the player to improve his/her performance during game play as well as to help the domain experts and educators better understand the learning process or success level of the players according to the pre-defined game objectives. Hence, the player task design is a critical issue for the serious game design. Most of the measurements usually based on the players' performance during completing requirements defined for each task to fulfill the objectives of each task. This issue is more important for the domain experts to better evaluate the players' performance during the game play.

On the other hand, in the game design, besides the tasks assigned for the player, there are some tasks that need to be performed by the computer system itself. In the UML-AD, these tools are all represented by the same notation. This can be represented by using the swimlanes of UML-AD. However, we believe that this representation is confusing especially for the domain experts and this critical issue should be represented by a specific notation without modifying the UML-AD structure. In other words, we believe that, player tasks and system tasks should be separated in the serious game design. Our main hypothesis is that, by separating player's tasks and system tasks in the game design documentation, the understandability level of these documents can be improved. For improving the understandability level of these documents, player tasks and system tasks should be separated in the serious game design. Hence, in this study, parallel to the existing notations used in the UML-AD, in order to represent the player tasks with a specific notation. We have used double circled action representations for the UML-ADE as shown in Figure 3.1.

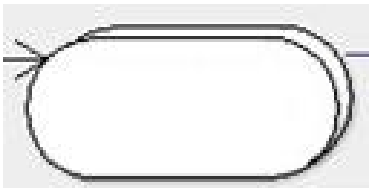


Figure 3.1 – UML-ADE Player Task Notation

In other words, in the UML-ADE model we have used the standard UML-AD action notations (Figure 3.2) to represent the system tasks and double circled action representation (Figure 3.1) to represent the player tasks.

Figure 3.2 – UML-AD Player Task Notation

In this study, the impacts of proposed UML-ADE model for the understandability of these diagrams (UML-AD and UML-ADE) by technicians are evaluated experimentally. The next chapter of this study describes the research methodology used for this experimental study.

CHAPTER 4

RESEARCH METHODOLOGY

In this chapter, the research methodology of the study is presented. The aim of the methodology is to give the work plan of the research. It is an investigation of finding solutions to collect new and useful information. This chapter covers measurable and testable data collection and research flows. In this study, our main assumption is to analyze and compare UML-AD and UML-ADE models and to better understand the level of understandability of the UML-ADE Model according to the UML-AD Model. Accordingly, this study is prepared to better understand reviewers' defect detection performance on UML-AD and UML-ADE Models. For this study, two scenarios developed for surgical simulation modules have been prepared. Each scenario is prepared by in two versions using UML-AD and UML-ADE. Additionally five defects have been seeded into these diagrams and the 72 participants are asked to detect these defects according to the description document of the scenarios. The details of the research model of this study explained below.

4.1. Research Questions

In this study we hypothesize that, the detecting defects level of UML-ADE is higher than that of UM-AD. We claim that if reviewers detect more defects in UML-ADE than that of UML-AD, then it could indicate that their detecting defect level of UML-ADE is higher than that of UML-AD. Accordingly, 3 research questions are set for this study.

RQ1. Can reviewers detect more defects in UML-ADE than UML-AD?

RQ2. Does the defects seeded in UML-ADE is easy to detect than UML-AD?

RQ3. Is reviewers' defect detection performance higher on UML-ADE than that of UML-AD?

4.2. Instruments

Several instruments are prepared for this study. These documents are the user requirements document of the scenarios, conceptual data model diagrams represented in UML-AD and UML-ADE models for each scenario, instructions document, defects report form, three questionnaires, experimental model diagrams are used. Below the details of these documents are explained in detail.

4.2.1. The User Requirements Documents for the Scenarios

For this study, two scenarios, which are developed for endoneurosurgery simulation for educational purposes have been used. For each scenario, a user requirements document is prepared. The original Turkish version of this document for Scenario-1 (Appendix C), for Scenario-2 (Appendix D) and the translated versions in English for Senario-1 (Appendix A) and Scenario-2 (Appendix B) are provided in the appendices.

4.2.2. Conceptual Data Model Diagrams

Additionally, two versions of each scenario are prepared by using UML-AD and UML-ADE notations. The original Turkish version of the UML-AD for scenario-1 (Appendix E), UML-AD for Scenario-2 (Appendix G), UML-ADE for Scenario-1 (Appendix F) and UML-ADE for Scenario-2 (Appendix H) are provided in the appendices. Similarly, the translated English version of the UML-AD for Scenario-1 (Appendix I), UML-AD for Scenario-2 (Appendix K), UML-ADE for Scenario-1 (Appendix J) and UML-ADE for Scenario-2 (Appendix L) are provided in the appendices.

4.2.3. Defect Report User Interface

In order to collect appropriate data to better understand the participants' performance during the experimental study, a specific computer program has been used. This program is named as Defect Report System. The system is developed as a web-based program, which is prepared to record detailed information such as student demographics, scenario code being studied. Additionally, their defect detection process is also recorded. There were five defects and two scenarios designed for the

experimental study. In each time, the reviewer detects a defect records it in to the provided boxes as shown in Figure 4.1. By this way, in which order the reviewer detects a defect, how long s/he works for detecting that specific defect as well as his/her description about the detected defect are recorded for each detected defect of two scenarios.

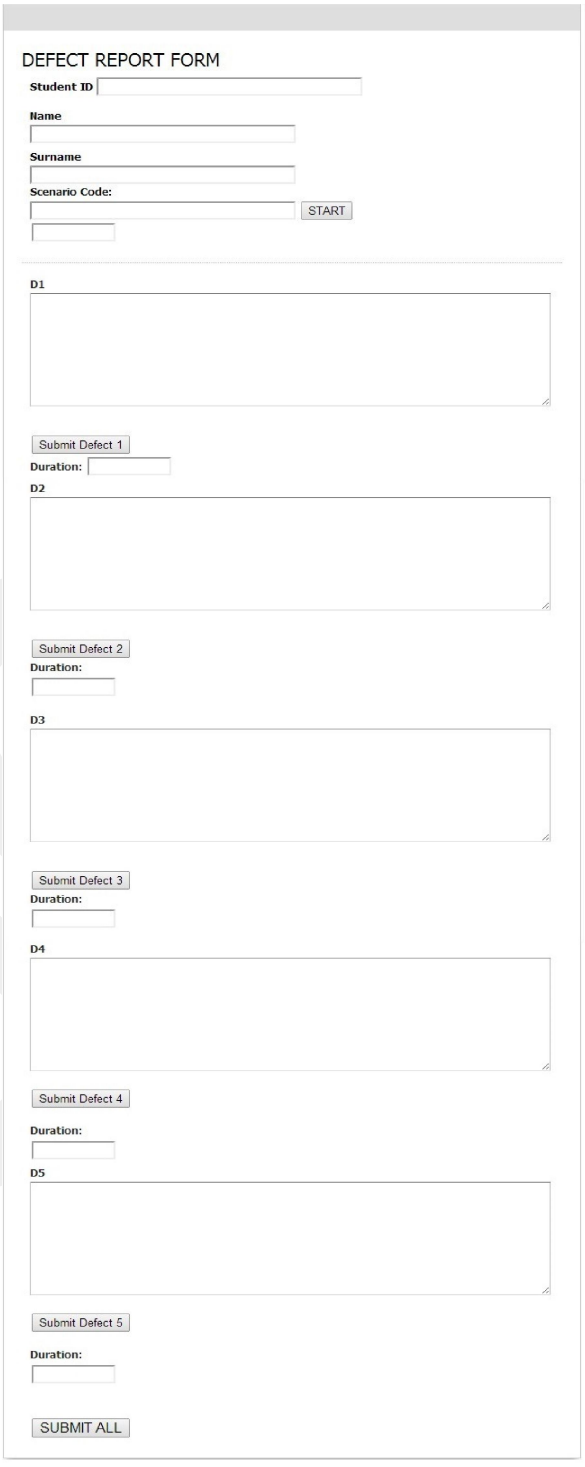


Figure 4.1 – Defect Report User Interface

4.2.4. Questionnaires

The questionnaires are prepared separately for UML-AD (Turkish version Appendix O, English version Appendix M) and UML-ADE (Turkish version Appendix P, English version Appendix N). The aim of these questionnaires is to better understand the reviewers' feedback and opinion in this study. The questionnaires are prepared specifically for this study and the items of this questionnaire are tested according to clarity by 4 experts in software engineering before using in this study.

4.2.5. Notation Explanation Document

Two Notation Explanation Documents for UML-AD (Turkish version Appendix T and English version Appendix R) and UML-ADE (Turkish version Appendix U and English version Appendix S) diagrams are prepared.

4.2.6. Experimental Data Model Diagrams

For the experimental study, five defects for each scenario were seeded in both the UML-AD and UML-ADE representations of each scenario. Defects are put randomly into the each scenario.

Senario-1: The lists of these defects that are seeded in scenario 1 are listed in Table 4.1.

Table 4.1 – Conceptual Model Defects for Scenario 1

DEFECTS	PROBLEM DEFINITION	DESCRIPTION
H01	Wrong Action State	Action state is not written true. In this scenario, there is no partial time.
H02	Wrong Action State	Action state is not written true. In this scenario, user should carry away the sphere to the outside.
H03	Wrong Transition	If the user does not carry away the sphere to the outside, total time should be controlled.
H04	Wrong Transition	If the user carries away the sphere to the box, point should be increased.
H05	Irrelevant Final state	There cannot be a final state. Scenario should control the scenario time.

These defects are seeded in both UML-AD and UML-ADE for Scenario 1 are provided and can be explained as follows:

H01: As stated by scenario requirement document, there is no partial time. There is only total time. In this action state, total time calculation should start.

H02: According to the scenario requirement document, user does not carry away the sphere to the box. There is a wrong action state. User should carry away the sphere to the outside. Carrying away the sphere to the outside is critical situation. If this critical situation is successful, user has a point.

H03 - H04: According to the scenario requirement document, if the user carries away the sphere to the outside, score increases. If not, again time control is done and scenario returns to the beginning. Location of H03 should change with location of H04.

H05: As stated by scenario requirement document, after the user gains 1 point, scenario time should be controlled. There is no final state. Scenario time finishes, score is calculated and then final state is put.

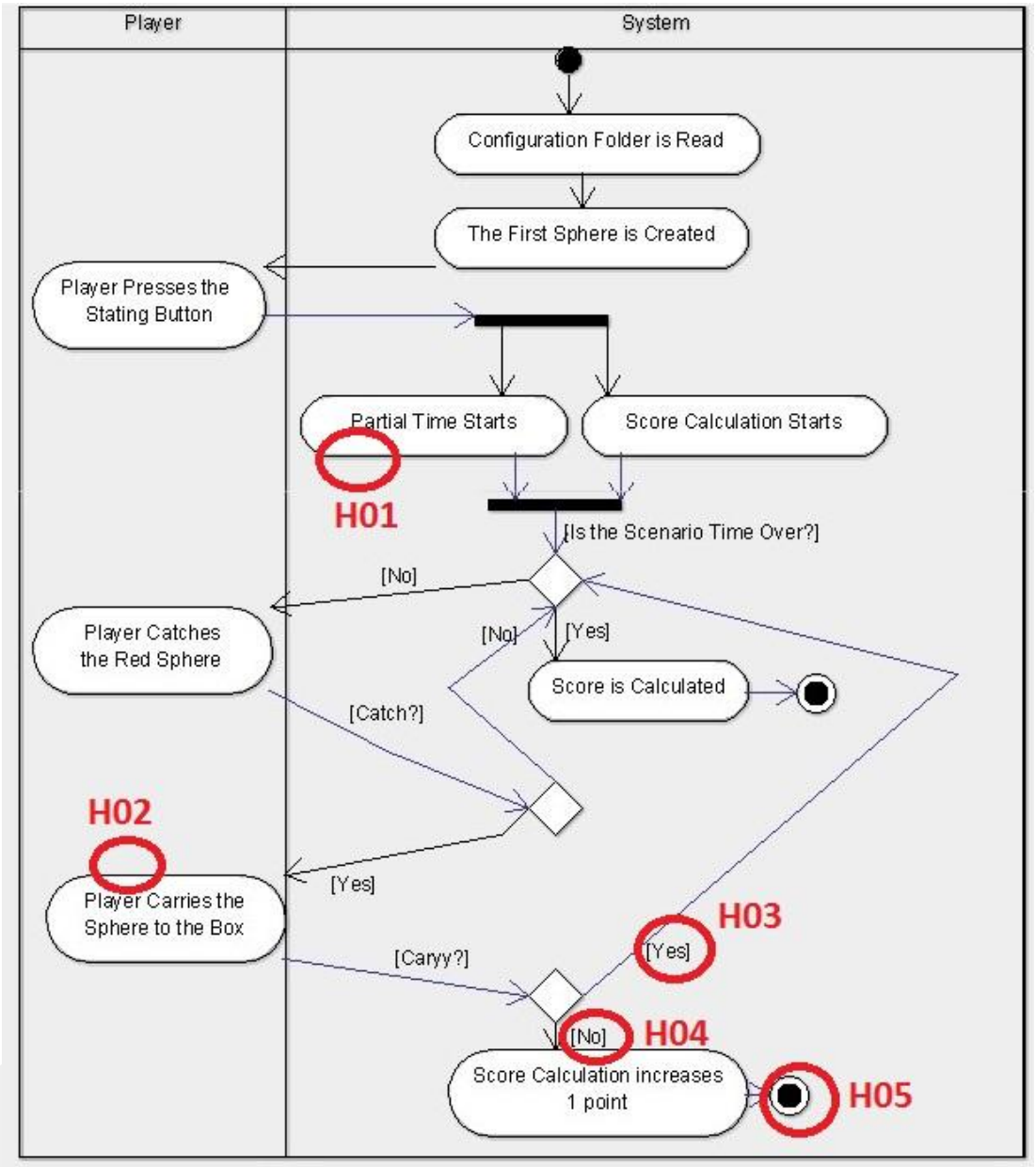


Figure 4.2 – UML-AD Model with Defects

Accordingly, both the UML-ADE and UML-AD versions of scenario-1 representations were modified as seeded these defects. The modified and defected versions of UML-AD diagram for scenario-1 is given in Figure 4.2. Similarly, the modified and defected versions of UML-ADE diagram for scenario-1 is given in Figure 4.3.

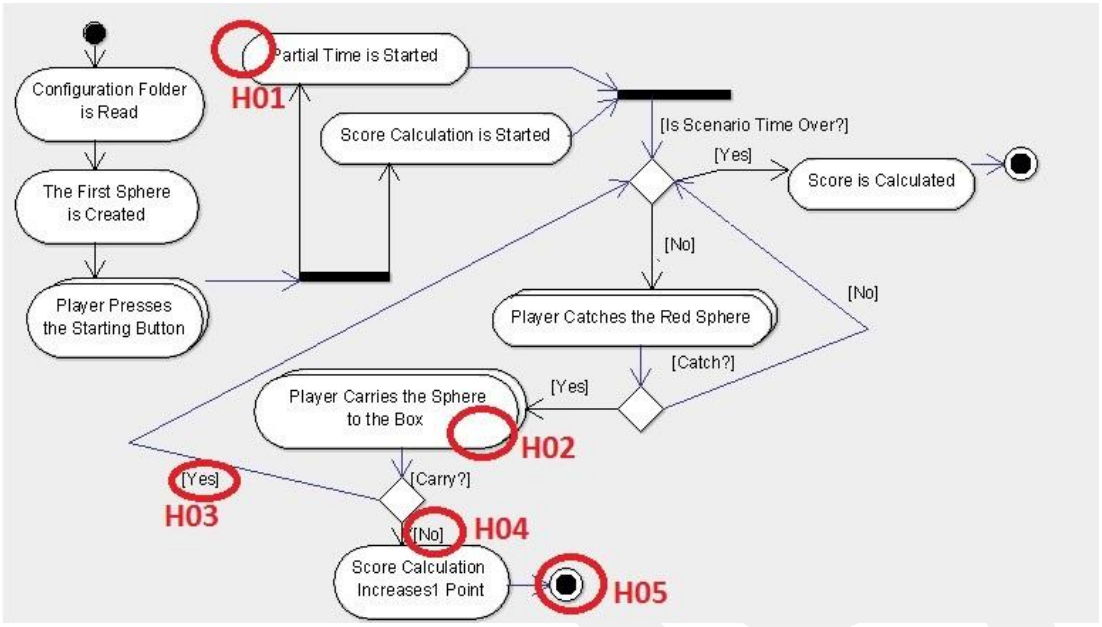


Figure 4.3 – UML-ADE Model with Defects

Scenario 2: The lists of these defects that are seeded in scenario 2 are listed in Table 4.2.

Table 4.2 – Conceptual Model Defects for Scenario 2

DEFECTS	PROBLEM DEFINITION	DESCRIPTION
H01	Missing Transition	Transition arrow should not be null. If there is a transition arrow, condition should be defined. It also must be "No" condition.
H02	Wrong Action State	User cannot remove the sphere with left haptic. User can remove the sphere with right haptic.
H03	Wrong Action State	Besides the partial time of finding box, it must be the partial time of removing sphere.
H04	Missing Transition	If there is a transition arrow, condition should be defined. It also must be "No" condition.
H05	Wrong Action State	Partial time of removing sphere must be reset.

Below detailed descriptions of each defect seeded in UML-AD and UML-ADE for Scenario 5 is provided and can be explained as follows:

H01: According to the scenario requirement document, conditions are determined. Transition arrow should not be null. If there is a transition arrow, condition should be defined. “No” condition should be written.

H02: According to the scenario requirement document, user cannot remove the sphere with left haptic. User should find red spheres in the boxes with left haptic. User can remove the spheres with right haptic.

H03: As mentioned by scenario requirement document, besides the partial time of finding box, it must be the partial time of removing sphere, because partial time of finding box has already started.

H04: According to the scenario requirement document, conditions are determined. Transition arrow should not be null. If there is a transition arrow, condition should be defined. “No” condition should be written.

H05: As mentioned by scenario requirement document, partial time of removing sphere must be reset. Partial time of removing sphere cannot start.

Accordingly, both the UML-ADE and UML-AD versions of scenario 2 representations were modified as seeded these defects. The modified and defected versions of UML-AD diagram for scenario-2 is given in Figure 4.4. Similarly, the modified and defected versions of UML-ADE diagram for scenario-2 is given in Figure 4.5.

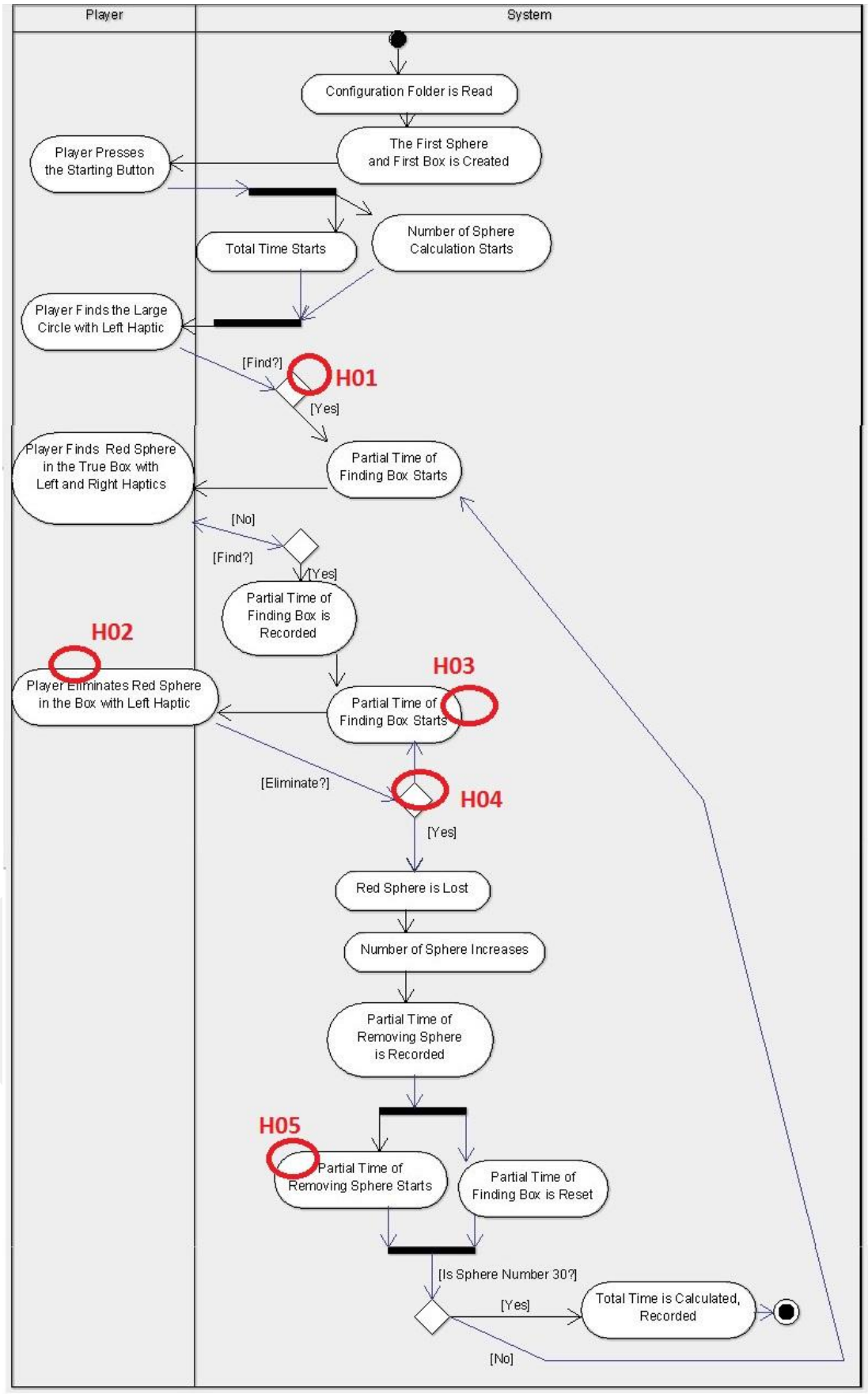


Figure 4.4 – UML-AD Model with Defects

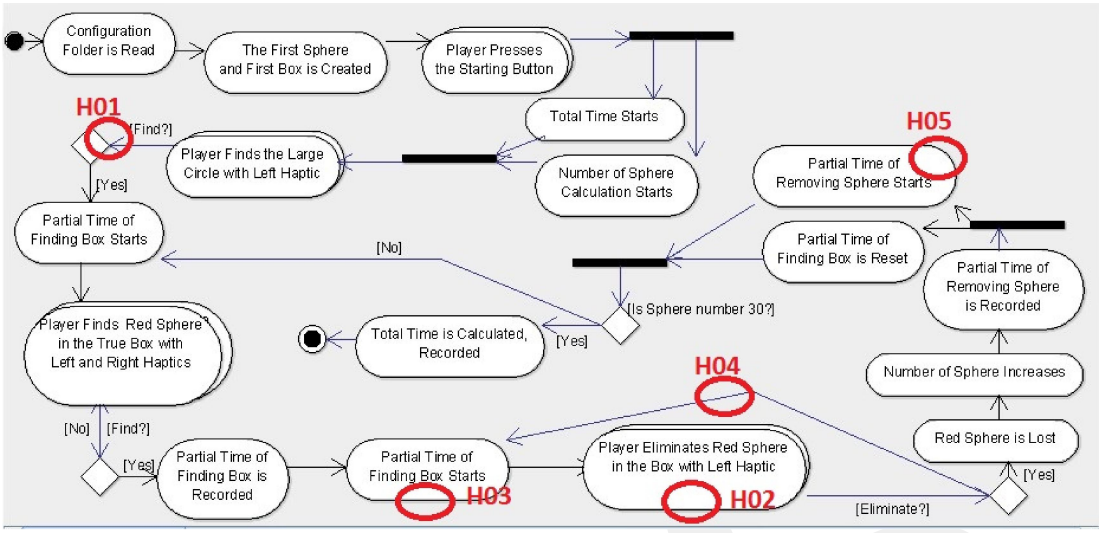


Figure 4.5 – UML-ADE Model with Defects

It should also be recorded that scenario 2 is created as more complex according to the scenario 1.

4.3. Research Procedures

This study is organized as an experimental study. The experiments are conducted in Turkish which is the native language of participants. We test and understand that the number of defects detected by each reviewer is higher in UML-ADE than UML-AD for each scenario. The research procedure is shown in Figure 4.6.

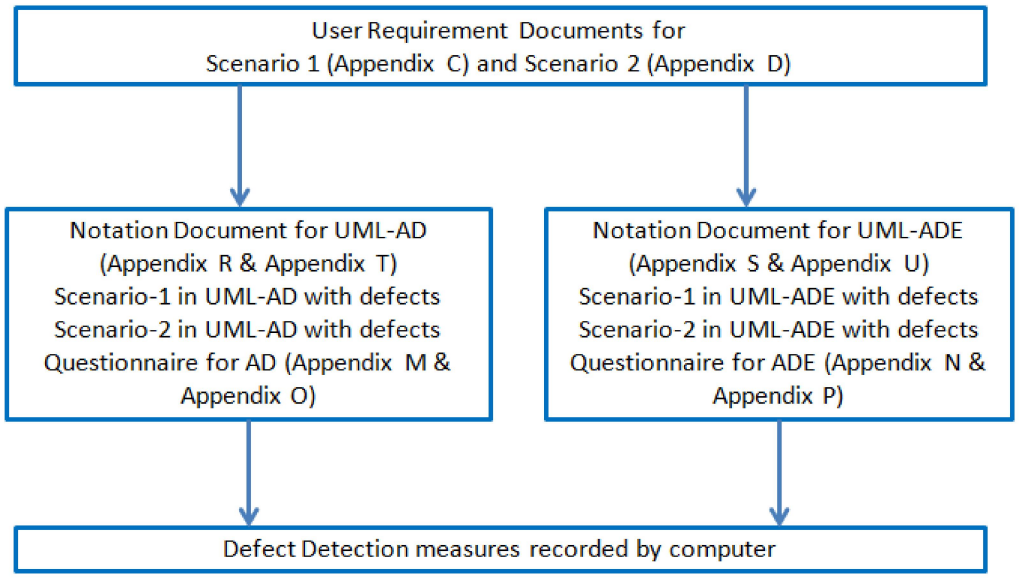


Figure 4.6 – Diagram for research procedure

As seen from Figure 4.6, before one week from the experimental study, the participants of this study are provided the user requirements documents prepared for each scenario. Hence the participants had a chance to review the requirements document and understand the main system requirements. For the experimental study, the 72 participants were randomly divided into two groups and each participant is provided with the instructions document for the study. During the experimental study, the first group is provided with the UML-AD representations of each scenario with 5 defects seeded on them. Similarly, the second group is provided with the UML-ADE representations of each scenario with the same 5 defects seeded on them. The participants are asked to examine the UML diagrams according to the requirements document and detect the defects. When the participant detects a defect, s/he asked to record it on the defect report user interface.

In order to better understand the reviewers' performance during the defect detection process, the performance of each reviewer is calculated according to the "Defect detection performance formula, Formula 2" provided by Cagiltay et al. (2013) [52]. Similarly, the recognized difficulty level of each defect is calculated by using the "Defect Detection Difficulty Level Formula, Formula 1" provided by Cagiltay et al. (2013) [52]. Hence, in order to answer the research questions, the data collected through this research is analyzed both descriptively and statistically.

4.4. Participants

For the experimental study, 72 reviewers were participated voluntarily. Those participants were students in their fourth year in the departments of Computer Engineering, Software Engineering and Information Systems Engineering. Detailed information about the participants for the experiment is shown in Table 4.3.

Table 4.3 – Participants of Experiment

	UML Name		
Gender	**UML-AD**	**UML-ADE**	**Total**
Female	8	9	17
Male	27	28	55
Total	35	37	**72**

As seen from Table 4.3, 72 participants have been voluntarily involved in this study. For the UML-AD, there were 35 participants. For the UML-ADE, there were 37 participants. In this study, there is no gender separation into the groups. In each diagram, participants have responsible for Scenario 1 and Scenario 2. Because the scenarios were given randomly to the participants, there is a slipping for 1 participant from UML-AD to UML-ADE. In the next section all results found through this study are provided.

4.5. Measures for UML-AD and UML-ADE

The main purpose of these measures is to analyze the defect detection process of UML-AD and UML-ADE. Accordingly, the two measures which were developed by Cagiltay et al. [52] and proposed to supply better understand performance of reviewers and the difficulty levels of defects are used in this study to better understand the understandability levels of UML-AD and UML-ADE. These two measures are named as the defect detection difficulty level and defect detection performance [52]. The defect detection difficulty level (DF) measures how difficult a defect to be detected according to the other defects for UML-AD and UML-ADE. Defect detection performance (PP) measure is also used to understand the performance of UML-AD and UML-ADE during the defect detection process [52].

4.5.1. Defect Difficulty Level

There are several measures for the information and performance of subjects. However, in this study, any past information cannot be asked to the reviewers, because these measures cannot be implemented into the formulas in this study.

In this study, we mention that familiarity and ability of the reviewers have been included through two factors in Cagiltay et al. [52]'s study. These factors are time spent which detects a defect and identifying the defect at the first place or at a later time. In Formula 1, the defect detection difficulty level measure is proposed.

$$DF_j = \frac{D_j \bullet O_j}{S_j}$$

DF_j: Defect detection difficulty level of the j^{th} defect

D_j : Average duration spent by all participants for finding defect j

O_j : Average score of all participants for detecting j^{th} defect

S_j : Success rate of detecting defect j (Number of people who detected defect j/Total number of participants)

Formula 1 – Defect Detection Difficulty Level Formula [52]

This formula is applied for each scenario. In this formula, Oj refers to the score of all participants gained from the defect detection order. D_j refers to the duration spent of all participants during the finding for each defect. S_j means the success of the finding for each defect. The defect detection order may also be related to the order. The defect information about the defect is presented in the user requirement document.

4.5.2. Participant Performance

In this measurement, the defect detection performance of each participant i (PPi) is calculated. PPi is the cumulative difficulty level of defects detected by P_i. To calculate participant performance, the defect detection difficulty level value of each defect (DFj) and a performance measure formula are used [52]. This formula is shown in Formula 2.

$$PP_i = \frac{\sum_{j=1}^{n} DF_j}{\sum_{k=1}^{s} DF_k}$$

PP_i: Defect Detection Performance of the i^{th} participant

DF_j: Difficulty level of the j^{th} defect calculated by Formula 1

n: Total number of defects detected by participant i

s: Total number of defects seeded in the UML-ADE/UML-AD

Formula 2 – Defect Detection Performance Formula [52]

CHAPTER 5

RESULTS

The results of the study are analyzed to answer the research questions. Accordingly, firstly the reviewers' performance during the defect detection process is analyzed. Hence, in order to better understand the reviewers' defect detection performance on both UML-ADE and UML-AD representations, we have evaluated number of defects detected by each participant, the defect difficulty value for each defect calculated by the formulas provided in [51][52].

5.1. Detected Defects in UML-ADE and UML-AD

An independent sample t-test was conducted to evaluate the hypothesis that the reviewers' work on the scenario-1 representation of UML-ADE detect more defects than that of the ones work on UML-AD. The test was significant, df=70, p=0.011. Reviewers working on UML-ADE detected more defects (M=3.19, SD=1.49) than the reviewers working on UML-AD (M=2.26, SD=1.52). Figure 5.1 shows the distribution of both groups.

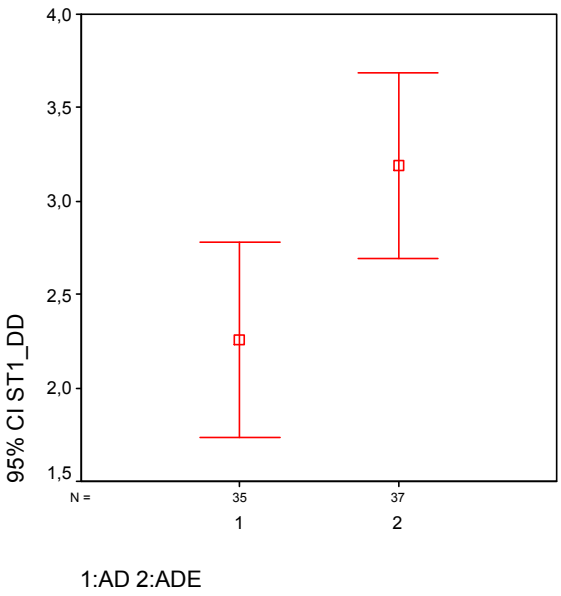


Figure 5.1 – Detected Defects Distribution in Both Groups for Scenario-1

An independent sample t-test was conducted to evaluate the hypothesis that the reviewers' work on the scenario-2 representation of UML-ADE detects more defects than that of the ones work on UML-AD. The test was significant, df=70, p=0.022. Reviewers working on UML-ADE detected more defects (M=2.77, SD=1.49) than the reviewers working on UML-AD (M=2.00, SD=1.21). Figure 5.2 shows the distribution of both groups.

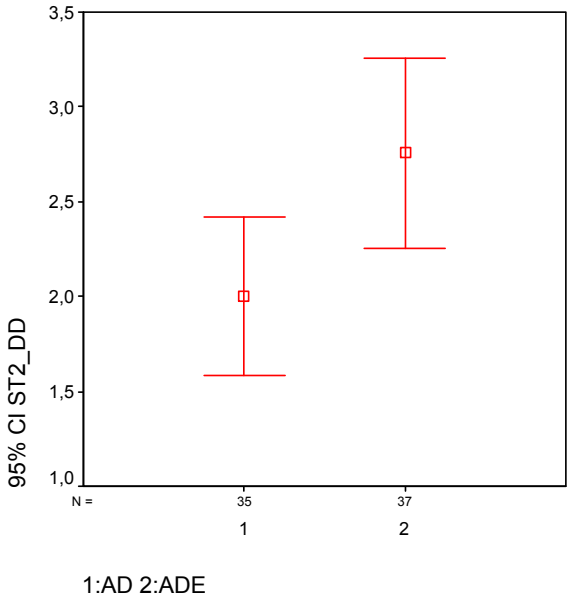


Figure 5.2 – Detected Defects Distribution in Both Groups for Scenario-2

5.2. Difficulty Levels of Defects

In order to answer this research question, the recognized defect difficulty levels of each defect is calculated by the Formula 1 provided in [52]. The values of these calculated defect difficulty levels for scenario 1 and scenario 2 are provided in Table 5.1 and Table 5.2 respectively.

Table 5.1 – Defect Difficulty Levels Scenario 1

Defect	UML-AD	UML-ADE
D1	2812	1221
D2	1728	1313
D3	1085	768
D4	1647	941
D5	1922	841
Average	1839	1017

As seen from Figure 5.1, the average recognized difficulty level value for the defects seeded in UML-AD of scenario 1 (1839) is higher than that of UML-ADE (1017). This result indicates that the reviewers recognized the same defects seeded in the UML-AD as harder to detect according to the same defects seeded in the UML-ADE.

Table 5.2 – Defect Difficulty Levels Scenario 2

Defect	UML-AD	UML-ADE
D1	733	439
D2	3049	1728
D3	1836	968
D4	1202	1374
D5	6880	2998
Average	2740	1501

Similarly, when recognized difficulty levels of defects seeded in UML-ADE and UML-AD representations of scenario 2 as seen from Table 5.2, the reviewers recognized the defects seeded in UML-ADE (average 1501) easily according to the ones seeded in UML-AD (average 2740). Because of the very limited number of defects, a statistical analysis was not conducted for this data.

5.3. Reviewers' Defect Detection Performance

An independent sample t-test was conducted to evaluate the hypothesis that for the design of scenario-1, the reviewers' defect detection performance on UML-ADE is higher than that of UML-AD by the Formula 2 provided in [52]. The test was significant, df=70, p=0.007. Reviewers working on UML-ADE performed higher (M=.62, SD=.30) than the reviewers working on UML-AD (M=.42, SD=.30). Figure 5.3 shows the distribution of both groups.

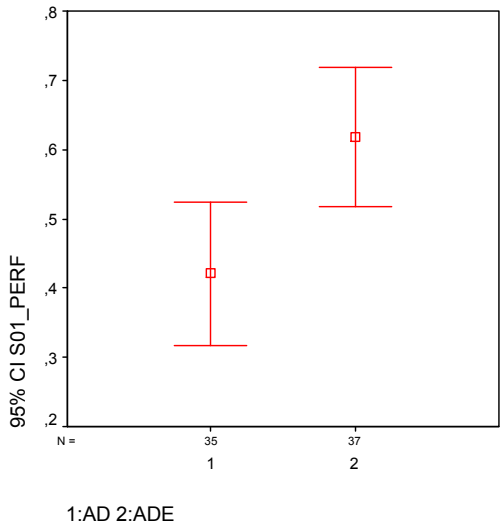


Figure 5.3 – Defect Detection Performance Distribution in Both Groups for Scenario-1

An independent sample t-test was conducted to evaluate the hypothesis that for the design of scenario-2, the reviewers' defect detection performance on UML-ADE is higher than that of UML-AD. The test was significant, df=70, p=0.007. Reviewers working on UML-ADE performed higher (M=.49, SD=.32) than the reviewers working on UML-AD (M=.29, SD=.27). Figure 5.4 shows the distribution of both groups.

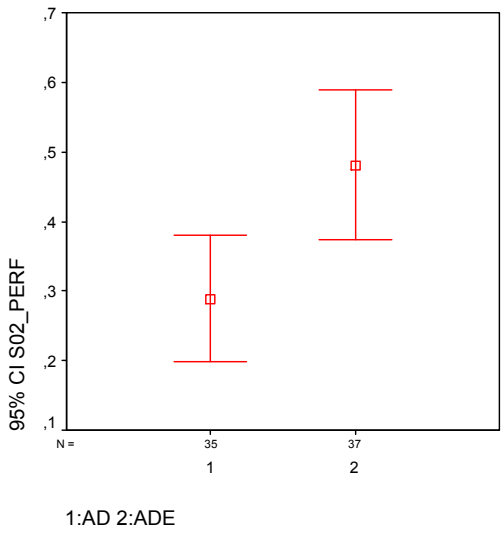


Figure 5.4 – Defect Detection Performance Distribution in Both Groups for Scenario-2

The questionnaire results also support these results. As seen from Table 5.3, the reviewers found the UML-ADE representations easier to understand (3.79) than UML-AD (3.07). Similarly their responses on the level of their understanding of game that is represented with the UML-ADE notations is higher (3.79) than that of UML-AD (3.37). Their responses on the complexity of the representations in UML-AD is higher (2.77) that that of UML-ADE (2.47).

Table 5.3 – Questionnaire Responses

Questionnaire Item	UML-ADE	UML-AD
"I think ..... diagram given to us is easy to understand"	3.79	3.07
"How complicated is the given ..... diagram, evaluate from 1 to 5 (1: very easy - 5: very difficult)"	2.47	2.77
"I think I understand the system very well by looking to the given ....."	3.79	3.37

Additionally, the True Positive (TP) and False Negative (FN) values are also calculated to better understand the performance of the reviewers. TP indicates the correctly identified defects by each reviewer and FN value indicates the incorrectly rejected defects by each reviewer. Accordingly, True positive rate, which is also called sensitivity measure indicating how well a reviewer can detect a defect, is also calculated by using the following formula [63].

An independent sample t-test was conducted to evaluate the hypothesis that the sensitivity value for the reviewers who work on the UML-ADE version of the system design is higher than that of the reviewers who work on the UML-AD version. The test was significant, $t(142) = 2.26$, $p= 0.26$. Sensitivity value for the reviewers who work on the UML-ADE version ($M=0.69$, $SD=0.30$) on the average is higher than that of the reviewers who work on the UML-AD version ($M=0.57$, $SD=0.33$). In other words, the probability of the defect detection rate of the reviewers working on the UML-ADE version is significantly higher than that of the ones working on UML-AD.

CHAPTER 6

DISCUSSIONS / CONCLUSION

In this study, an enhanced version of UML-AD representation is proposed in order to improve the level of understandability of these diagrams. Main assumption of this study is that UML-ADE model can be more understandable and easier than the UML-AD model. Accordingly, through two real-life scenarios this assumption is tested experimentally. The research findings of this study can be summarized as below:

- Reviewers' who are working on UML-ADE model could detect more defects and understand the system easily.
- Reviewers' recognition on the difficulty level of defects in UML-ADE is lower than that of UML-AD.
- Reviewers' performance in the Scenario 1 is higher than that of the Scenario 2. Since the Scenario 2 was designed as more complicated according to the Scenario 1, this was an expected result.

In general the findings of this study can be summarized as in Table 6.1. As seen from this table, the mean of number of detected defects for scenario 1 (2.26 for UML-AD, 3.19 for UML-ADE) is higher than that of Scenario 2 (2.00 for UML-AD, 2.77 for UML-ADE). This results show the answer of RQ1.

Table 6.1 – Summary of Results

		Scenario-1	Scenario-2
Number of Detected defects	UML-AD	2.26	2.00
	UML-ADE	3.19	2.77
Defect Detection Performance of the Reviewers	UML-AD	0.42	0.29
	UML-ADE	0.62	0.49
Recognized Defect Difficulty Levels	UML-AD	1839	2740
	UML-ADE	1017	1501

Similarly, as seen from Table 6.1, it can be seen that, the recognized defect difficulty levels for Scenario 2 (UML-AD: 2740, UML-ADE: 1501) are higher than that of Scenario 1 (UML-AD: 1839, UML-ADE: 1017) for both UML representations. Parallel to these results the recognized difficulty levels of defects in scenario 2 (2740 for UML-AD, 1501 for UML-ADE) is higher than that of Scenario-1 (1839 for UML-AD, 1017 for UML-ADE). As reported earlier, this is because the second scenario was designed as more complicated according to the first one. Hence when the scenario becomes more complicated the recognized defect difficulty level values become higher and reviewers' performance becomes lower.

Another important result of this study shows that, reviewers' performance for the UML-ADE representations of both scenarios is higher than that of UML-AD representations of both scenarios. This indicated that the UML-ADE representation is easier to detect defect according to the UML-AD representation for this study. The results show the answer of RQ3 promise to use the UML-ADE representation for describing the tasks of the users and systems separately for improving the understandability level of the UML Activity diagram representations of software systems.

6.1. Limitations and Future of Study

In this study, the scenarios are conducted with the 4th year students of Computer Engineering, Software Engineering and Information Systems Engineering. Hence this study reflects the performance of technicians on these representations. On the other hand the performance of domain experts in these representations should also be

studied. Additionally, this study is conducted on scenarios that are developed for the surgical simulation purposed. Hence as future studies, the hypothesis can also be analyzed on different domains.

Also, in this study, 72 participants are used who are in fouth year of Computer Engineering, Information System Engineering and Software Engineering in Atılım University because of not reaching enough domain experts.

REFERENCES

[1] Radoff J., "History of social games." *Jon Radoff's Internet Wonderland*, 2010.

[2] Grace L., "Game Type and Game Genre." *Retrieved February* 22 (2005): 2009, http://aii.lgrace.com/documents/Game_types_and_genres.pdf

[3] Kramer W., "What makes a game good." *The Games Journal*, 2000.

[4] Doğusoy B., İnal Y., "Çok Kullanıcılı Bilgisayar Oyunları ile Öğrenme." *VII. Ulusal Fen Bilimleri ve Matematik Eğitimi Kongresi GÜ, Gazi Eğitim Fakültesi*, pp. 7-9, 2006.

[5] Arevolo W., Latham L., "Key Reasons Why You Should Consider a 'Learning by Gaming' Strategy", Gartner research, 2006.

[6] Derryberry. A., "Serious games: online games for learning", 2007.

[7] Wagner, M., "What are the advantages and disadvantages of serious games", December, 2005.

[8] Ulicsak M., "*Games in Education: Serious Games: A Futurelab Literature Review"*, FutureLab, 2010.

[9] Wiemeyer, J., & Kliem, A., "Serious games in prevention and rehabilitation—a new panacea for elderly people?", *European Review of Aging and Physical Activity*, *9*(1), pp. 41-50, 2012.

[10] Mangır M., Aktaş Y., "Çocuğun gelişiminde oyunun önemi"., *Yaşadıkça Eğitim Dergisi* 26(16), pp. 14-19, 1993.

[11] Leen, E., "Changing attitudes with new learning methods Serious games and e-learning", *CAVA Conference, Innovation in Learning Institute University of Erlangen-Nuremberg*, March, 2012.

[12] International Scholarships & Fellowships, 3 funded PhD Positions in the Domain of Serious Games, October, 2011, http://www.scholarships-links.com/getpdf.php?id=2867

[13] Marchiori, J. E., Serrano A., Blanco, A., Martínez-Ortíz, I., Fernández-Manjón, B., "Integrating domain experts in educational game authoring: a case study". In *Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), 2012 IEEE Fourth International Conference,* pp. 72-76, IEEE, March, 2012.

[14] Fischer, G., Giaccardi, E., "Meta-design: A framework for the future of end-user development". In End user development, pp. 427-457, Springer Netherlands, 2006.

[15] Fogli, D., "End-User Development for E-Government Website Content Creation", *End-User Development*. Springer Berlin Heidelberg, pp. 126-145, 2009.

[16] Costabile, M. F., Dittrich, Y., Fischer, G., Piccinno, "A End-User Development ", In *Proceedings of the 3rd International Symposium on EUD (IS-EUD 2011), Torre Canne, Italy, June 7* (Vol. 10)., June, 2011.

[17] Troyer, D. O., Paret, E., "Challenges in Designing Domain-Specific Modeling Language for Educational Games", http://wise.vub.ac.be/sites/default/files/publications/DSML_submitted.pdf

[18] Hunicke, R., "Put Your Game Development Education to Work", Game Developer's 2003 Game Career Guide, 2003.

[19] Kirriemuir, J., McFarlane, A., "Literature Review in Games and Learning", 2004, Report 8.

[20] Kleins, A., Teilans, A., Merkuryev, Y., Krasts, O., "A Metamodel Based Approach for UML Notated Domain Specific Modelling Language", *Computer Modelling and Simulation (UKSim), 2011 UkSim 13th International Conference on. IEEE, 2011.*

[21] Taylor, M. J., Gresty, D., Baskett M., "Computer Game-Flow Design", *Computers in Entertainment (CIE)*, 4(1): 5, January, 2006.

[22] Costabile, M. F., Fogli, D., & Mussio, P., & Piccinno, A., "A meta-design approach to End-User Development", *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*. IEEE, pp. 308-310, 2005.

[23] Tolvanen, J. P., Rossi, M., "MetaEdit+: Defining and Using Domain-Specific Modeling Languages and Code Generators", *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. ACM, pp. 92-93, October, 2003.

[24] Ruohomäki, V., "A simulation game for the development of administrative work processes", *The Simulation and Gaming Yearbook* 3, pp. 264-270. 1995, http://lib.tkk.fi/Diss/2002/isbn9512260948/article3.pdf

[25] Moreno-Ger, P., Blesius, C., & Currier, P., & Sierra, J., L., & Fernández-Manjón B., "Rapid Development of Game-like Interactive Simulations for Learning Clinical Procedures", *En Proceedings of Game Design and Technology Workshop and Conference,* 2007, http://www.e-ucm.es/drafts/e-UCM_draft_86.pdf

[26] Mishra, S., "Visual Modeling & Unified Modeling Language (UML): Introduction to UML", November, 2008.

[27] Tenzer, J., Stevens, P., "GUIDE: Games with UML for interactive design exploration", *Knowledge-Based Systems*, *20*(7), pp. 652-670, 2007.

[28] Fischer, G., Giaccardi, E., & Ye, Y., & Sutcliffe, A. G., & Mehandjiev, N., "Meta-Design: a Manifesto For End-User Development", *CACM*, 47(9), pp. 33-37, 2004.

[29] Fischer, G., Giaccardi, E., "Meta-Design: A Framework for the Future of End-User Development", To appear in Lieberman, H., Paternò, F., Wulf, V. (Eds), *End User Development - Empowering people to flexibly employ advanced information and communication technology*, Kluwer Academic, Dordrecht, The Netherlands, 2005.

[30] CIRULIS, A., & GINTERS, E., "Control of Simulation Elements in Virtual World", *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*. Eds. N. A. Baykara, and N. E. Mastorakis. No. 11. World Scientific and Engineering Academy and Society, June, 2009.

[31] SWEETSER, P., & WYETH, P., "GameFlow: A Model for Evaluating Player Enjoyment in Games", *Computers in Entertainment (CIE)* 3(3): 3-3. July, 2005.

[32] Moreno-Ger, P., & Martínez-Ortiz, I., & Sierra, J. L., & Fernández-Manjón, B., "A Content-Centric Development Process Model", *IEEE Computer,* vol. 41, pp. 24-30, 2008.

[33] Torrente, J., & Moreno-Ger1, P., & Fernández-Manjón, B., & Sierra, J., L., "Instructor-oriented Authoring Tools for Educational Videogames", *Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on*. IEEE, 2008.

[34] Erdem, A. T., & Utku, B., & Abacı, T., & Eroğlu Erdem, Ç., "Advanced Authoring Tools for Game-Based Training", Formerly with Momentum Digital Media Technologies, *Proceedings of the 2009 Summer Computer Simulation Conference*. Society for Modeling & Simulation International, 2009.

[35] Golding, M., & Ray, J., Sams Teach Yourself Adobe® Creative Suite 3, All in One, "The Game Plan: Developing a Workflow", Chapter 3, pp. 45-65, 2008.

[36] Bontchev, B., "A Framework for Educational Word Games", *Proceedings of the Int. Conf. on Intelligent Computational Systems (ICICS'2012), ISBN,* pp. 978-81., January, 2012.

[37] Eshuis, R., & Wieringa, R., "Tool support for verifying UML activity diagrams", Software Engineering, IEEE Transactions on (Volume:30, Issue: 7), pp. 437-447, July, 2004.

[38] Bell, D., "UML basics Part II: The activity diagram", *IBM Global Services Relational Software*, 2003.

[39] Tuncel, S., "UML Unified Modeling Language", March, 2013, https://dosya.sakarya.edu.tr/Dokumanlar/2013/424/58009950_05_ym_umlusecase.pdf

[40] Cockburn, A., "Writing Effective Use Cases, Pearson Education", 2001.

[41] UML Revision Task Force, "OMG Unified Modeling Language Specification", version 1.4, (Final Draft) February, 2001.

[42] "OMG Unified Modeling Language Superstructure Specification", version 2.1.1. Document formal/2007-02-05, Object Management Group, February 2007.

[43] Flater, D., & Martin, P., & Crane M., "Rendering UML Activity Diagrams as Human-Readable Text", In IKE, pp. 207-213, November, 2007.

[44] Preeti, D. G., "Activity Diagrams - Advantages, Disadvantages and Applications of Use", July, 2011.

[45] Paech, B., "On the Role of Activity Diagrams in UML", *The Unified Modeling Language.«UML»'98: Beyond the Notation*. Springer Berlin Heidelberg, pp. 267-277. 1998.

[46] Patel, P., & Patil, N., N., "Test case formation using UML activity diagram", Proceedings of *Communication Systems and Network Technologies (CSNT), 2013 International Conference on*. IEEE, 2013.

[47] Siebenhaller, M., & Kaufmann, M., "Drawing Activity Diagrams", *Proceedings of the 2006 ACM symposium on Software visualization*. ACM February, 2006.

[48] Conrad, B., "UML 2 Activity and Action Models", in Journal of Object Technology, vol. 2, no. 4, pp. 43-54, July-August, 2003.

[49] Bohm, C., & Jacopini, G., "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules", Communications of the ACM, 9(5), pp. 336-371, 1966.

[50] Cagiltay, E. N., Tokdemir, G., "Story/Scenario Flow Diagrams: A Simple Modeling Tool for Learning via Task Driven Play", 2012.

[51] Cagiltay, N., & Topallı, D., A Database Design Methodology For Complex Systems, 2013.

[52] Cagiltay, N.E., Tokdemir, G., Kilic, O., Topalli, D., "Performing and Analyzing non-Formal Inspections of Entity Relationship Diagram (ERD)", *Journal of Systems and Software*, 2013. DOI: 10.1016/j.jss.2013.03.106, in Press, http://www.sciencedirect.com/science/article/pii/S0164121213001015

[53] Costabile, M. F., Fogli, D., Mussio, P., & Piccinno, "A. End-user development: The software shaping workshop approach". In End user development, pp. 183-205, Springer Netherlands. 2006.

[54] Saleh K., El-Morr C., "M-UML: an extension to UML for the modeling of mobile agent-based software systems", Information and Software Technology, pp. 219-227, July, 2003.

[55] Belloni A. E., Marcos A. C., "Modeling of Mobile-Agent Applications with UML", In *Proceedings of the Fourth MA-UML XMI UML XMI OO Code Code PSM PIM Argentine Symposium on Software Engineering (ASSE 2003)* , Vol. 32, pp. 1666-1141.

[56] Baumeister H., Koch N., Kosiuczenko P., Wirsing M., "Extending Activity Diagrams to Model Mobile Systems", *Objects, Components, Architectures, Services, and Applications for a Networked World*. Springer Berlin Heidelberg, pp. 278-293, 2003.

[57] C. Klein, A. Rausch, M. Sihling, Z. Wen, "Extension of the unified modeling language for mobile agents", in: K. Siau, T. Halpin (Eds.), Unified Modeling Language: Systems Analysis, Design and Development Issues, Idea Group Publishing, 2001, pp. 116–128, (Chapter VIII).

[58] Rego, P., Moreira, P. M., & Reis, L. P., "Serious games for rehabilitation: A survey and a classification towards a taxonomy", In *Information Systems and Technologies (CISTI), 2010 5th Iberian Conference on* (pp. 1-6). IEEE, June 2010.

[59] Göbel, S., Hardy, S., Wendel, V., Mehm, F., & Steinmetz, R., "Serious games for health: personalized exergames", In *Proceedings of the international conference on Multimedia*, pp. 1663-1666, ACM, October, 2010.

[60] OMG, UML Unified Method: Notation Guide, Version 1.3, 1999.

[61] Schmidt, J. W., Wienberg, A., "A Comparison of Event-driven Process Chains and UML Activity Diagram for Denoting Business Processes", *Harburg: Technische Universität Hamburg-Harburg*, 2001.

[62] Rodrigues, R. W., "Formalising UML activity diagrams using finite state processes", In *Proc. of the 3rd Intl. Conf. on the Unified Modeling Language, York, UK*, October 2000.

[63] Peng, Y., Wanga, G., Wangb, H., “User preferences based software defect detection algorithms selection using MCDM”, Information Sciences 191 (2012) 3–13, 2012.

APPENDIX A: User Requirements Document For Scenario 1 (English)

This scenario passes in a room. Player acts backward and forward with using 1 haptic device in the room. Configuration folder is read by system. In this folder, there are sphere size, distance, wall shape, lines, and sphere color frames. Also, the first sphere is created by system. Processes that are in the scenario start within the player press the starting button. Player's the main task is to collect red spheres from the different point of the room and carry them to the outside. In this scenario, scenario time and score calculation is defined. Scenario time is total times that all identified spheres are carried to the outside after they are caught. The score calculation increases with the player carries the every sphere to the outside. Scenario time and score calculation starts when the scenario starts. Scenario time control is done. If the scenario time does not finish, player should catch the every sphere in the room. If the player catches the every sphere in the room, s/he should carry it to the outside. When the player carries the sphere to the outside, score calculation increases 1 point and scenario time is control again. If the total time is not over that is for scenario, scenario processes continues repeatedly with returning the beginning. If the total time is over that is for the scenario, total score is calculated for the player catches the sphere and carries it to the outside and the scenario finishes.

APPENDIX B: User Requirements Document For Scenario 2 (English)

This scenario passes in a room. Player acts backward and forward with using 2 haptic devices in the room. Configuration folder is read by system. In this folder, there are sphere size, distance, wall shape, lines, and sphere color frames. Also, the first sphere and first box are created by system. Processes that are in the scenario start within the player press the starting button. Player's the main task is to find red spheres from the boxes that have different angles with left or right haptic and eliminate them.

In the scenario, when the game starts, total time and number of sphere calculation also starts. Total time continues until the player success to eliminate the 30 spheres from the boxes that have different angles. Player should find the large circle with left haptic. This large circle helps player to eliminate the sphere with right haptic. If the player does not find the large circle, this process is repeated until the circle is found. If the player finds the large circle, partial time of finding box and player should find the spheres from the boxes that have different angles. When the player does not find process continues until sphere is found. If the player finds the spheres from the boxes that have different angles, partial time of finding box is recorded and partial time of removing sphere starts. Partial time of removing sphere is time that the player eliminates the red sphere with the right haptic in the box. When the sphere is eliminated each, partial time of removing the sphere is reset. Player should eliminate the red sphere in the true box with right haptic. If the player eliminates the red sphere, the red sphere is lost; the number of sphere increases, partial time of removing the sphere is recorded and reset, partial time of finding box is reset, and until the number of sphere is 30, the red sphere occurs another box. If the player does not eliminate the red sphere, the sphere cannot be lost, partial time of removing sphere starts and player eliminates the red sphere in the true box with right haptic. When the player successes to eliminate the 30 spheres, total time are recorded and scenario is completed. If the player does not success to eliminate the 30 spheres, scenario returns the 'partial time of finding box' process and scenario processes continues repeatedly.

APPENDIX C: User Requirements Document For Scenario 1 (Turkish)

Bu senaryo bir oda içinde geçmektedir. Oyuncu oda içinde 1 dokunsal cihazı (haptic) kullanarak ileri geri hareket edebilir. Sistem tarafından konfigürasyon dosyası okunur. Bu dosyanın içinde küre boyutu, mesafe, duvar biçimi, çizgiler, küre rengi, kare boyutu gibi parametreler yer alır. Ayrıca sistem tarafından ilk küre oluşturulur. Senaryodaki işlemler oyuncunun başlama tuşuna basmasıyla başlar. Oyuncunun temel görevi oda içinde farklı noktalarda çıkan kırmızı küreleri toplayıp alanın dışına çıkarmaktır. Senaryoda senaryo süresi ve puan hesaplaması tanımlanmıştır. Senaryo süresi, senaryo içinde tanımlı olan tüm kürelerin yakalanarak alan dışına çıkarılması için geçen toplam süredir. Puan hesaplaması ise, oyuncunun her küreyi alan dışına çıkarmasıyla artmaktadır. Senaryonun başlamasıyla birlikte senaryo süresi ve puan hesaplaması da başlar. Süre kontrolü yapılır. Eğer senaryo süresi bitmemiş ise, oyuncu oda içerisinde beliren her küreyi yakalamalıdır. Oyuncu oda içerisinde beliren her küreyi yakaladığı takdirde, küreyi alan dışına götürmelidir. Oyuncunun küreyi alan dışına götürebilmesi durumunda, puan hesaplaması 1 artar ve senaryo süresi yeniden kontrol edilir. Eğer oyuncu küreyi yakalayamazsa veya küreyi yakalayıp alan dışına götüremezse, başa dönülerek tekrar süre kontrolü yapılır. Eğer senaryo için ayrılan toplam süre bitmemiş ise, senaryo işlemleri başa dönerek tekrarlı bir şekilde devam eder. Senaryo için ayrılan toplam süre bittiğinde, oyuncunun küreyi yakalayıp alan dışına çıkarma işlemleri için topladığı puan hesaplanır ve senaryo sonlandırılır.

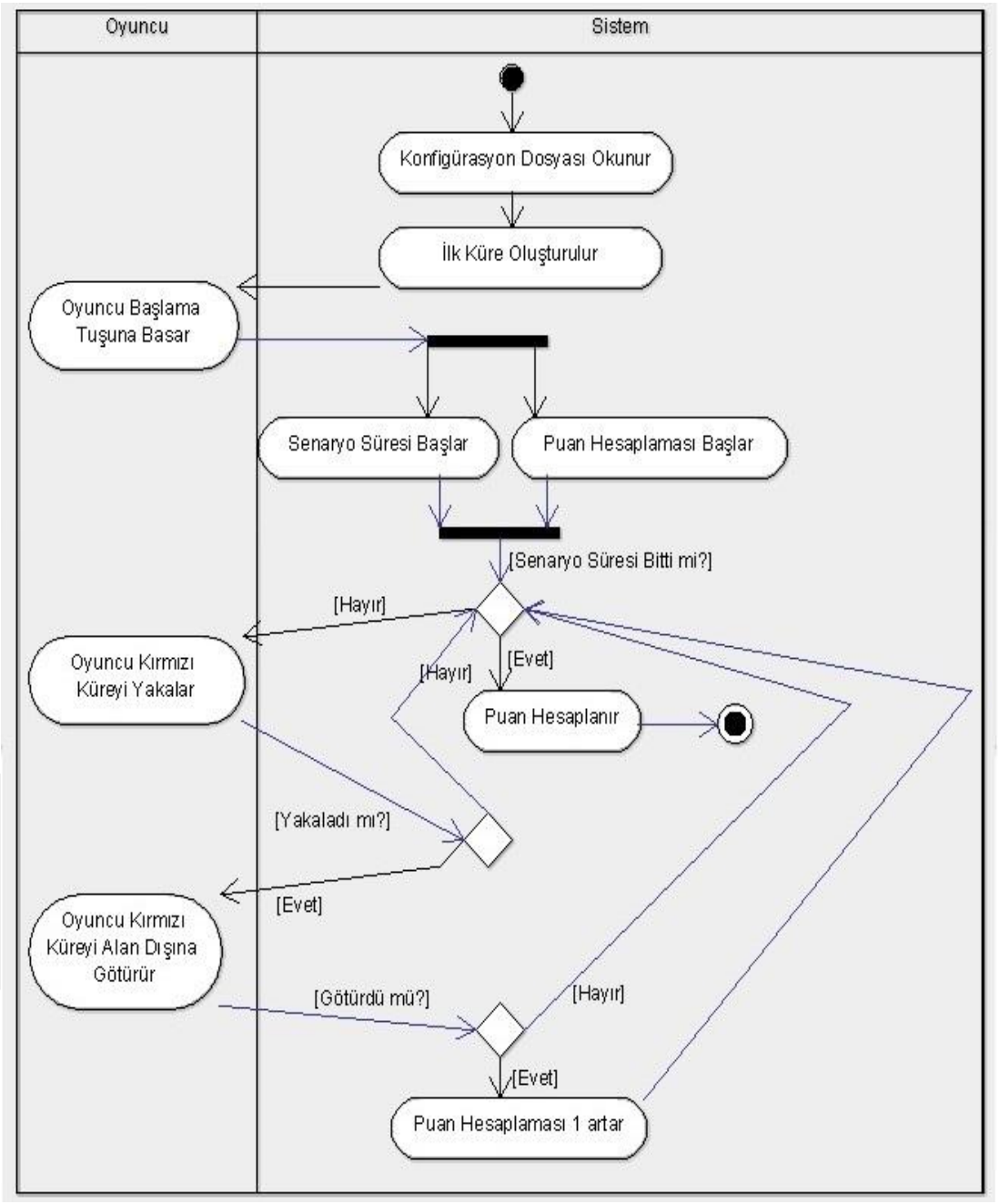
APPENDIX D: User Requirements Document For Scenario 2 (Turkish)

Bu senaryo bir oda içinde geçmektedir. Oyuncu oda içinde 2 dokunsal cihazı (haptic) kullanarak ileri geri hareket edebilir. Sistem tarafından konfigürasyon dosyası okunur. Bu dosyanın içinde küre boyutu, mesafe, duvar biçimi, çizgiler, küre rengi, kare boyutu gibi parametreler yer alır. Ayrıca sistem tarafından ilk küre ve ilk kutu oluşturulur. Senaryodaki işlemler oyuncunun başlama tuşuna basmasıyla başlar. Oyuncunun temel görevi sol veya sağ haptic ile farklı açılara sahip kutular içerisinden çıkan kırmızı küreleri bulup, sağ haptic ile yok etmektir.

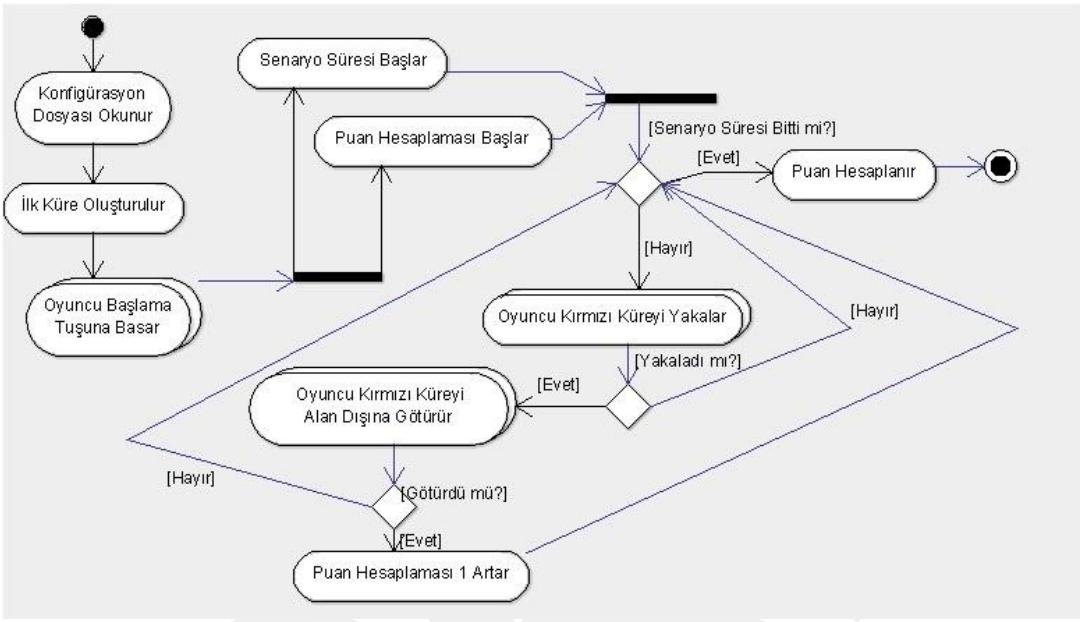
Senaryoda oyun başlayınca toplam süre ve küre sayısı hesaplaması başlar. Toplam süre, oyuncunun farklı açıdaki kutulardan çıkan 30 kırmızı küreyi yok etmeyi başarana kadar devam eder. Oyuncu sol haptic ile ekranda büyük bir çember bulmalıdır. Bu büyük çember, oyuncunun sağ haptic ile kutu içerisindeki küreyi yok etmesine yardımcı olur. Oyuncu büyük çemberi bulamaması durumunda, çember bulunana kadar bu süreç tekrarlanır. Oyuncu büyük çemberi bulursa, kutu bulma kısmi süresi başlar vefarklı açılardaki kutulardan çıkan küreleri bulması gerekir. Kutu bulma kısmi süresi oyuncunun sol ve sağ haptic ile doğru kutu içerisindeki kırmızı küreleri bulma süresidir. Oyuncu farklı açılardaki kutulardan çıkan küreleri bulamadığında, süreç bulana kadar devam eder. Oyuncu farklı açılardaki kutulardan çıkan küreleri bulduğunda ise, kutu bulma kısmi süresi kaydedilir ve küre yok etme kısmi süresi başlar. Küre yok etme süresi ise, oyuncunun sağ haptic ile baktığı kutunun içindeki kırmızı küreyi yok etme süresidir. Küre her yok edilişte küre yok etme kısmi süresi de sıfırlanır. Oyuncu sağ haptic ile baktığı doğru kutunun içindeki kırmızı küreyi yok etmelidir. Oyuncunun kırmızı küreyi yok etmesi durumunda, kırmızı küre kaybolur, küre sayısı artar, küre yok etme kısmi süresi kaydedilip sıfırlanır, kutu bulma kısmi süresi sıfırlanır ve küre sayısı 30 olana kadar, kırmızı küre başka bir kutudan çıkar. Oyuncunun kırmızı küreyi yok etmeme durumunda ise, küre kutudan kaybolmaz, küre yok etme kısmi süresi başlar ve oyuncu sağ haptic ile baktığı kutunun içindeki kırmızı küreyi yok eder. Oyuncu 30 küreyi başarılı bir şekilde kaybetmeyi başarırsa toplam süre kaydedilir ve senaryo tamamlanır. Oyuncu

30 küreyi başarılı bir şekilde kaybetmeyi başaramazsa, senaryo tekrar kutu bulma kısmi süresinin başlama aşamasına döner ve senaryo işlemleri tekrarlı bir şekilde devam eder.

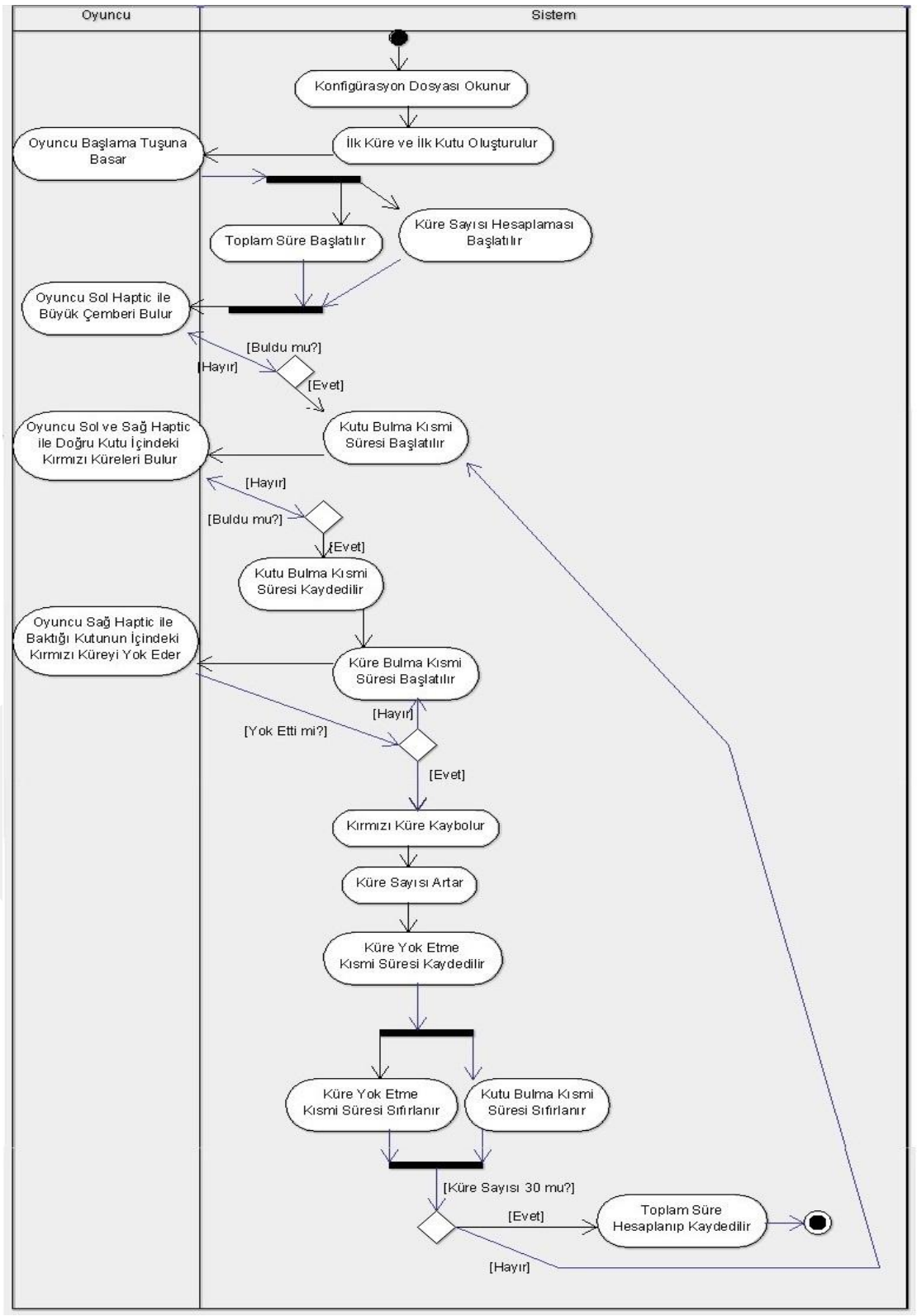
APPENDIX E: UML-AD Model Used For Scenario 1 (Turkish)



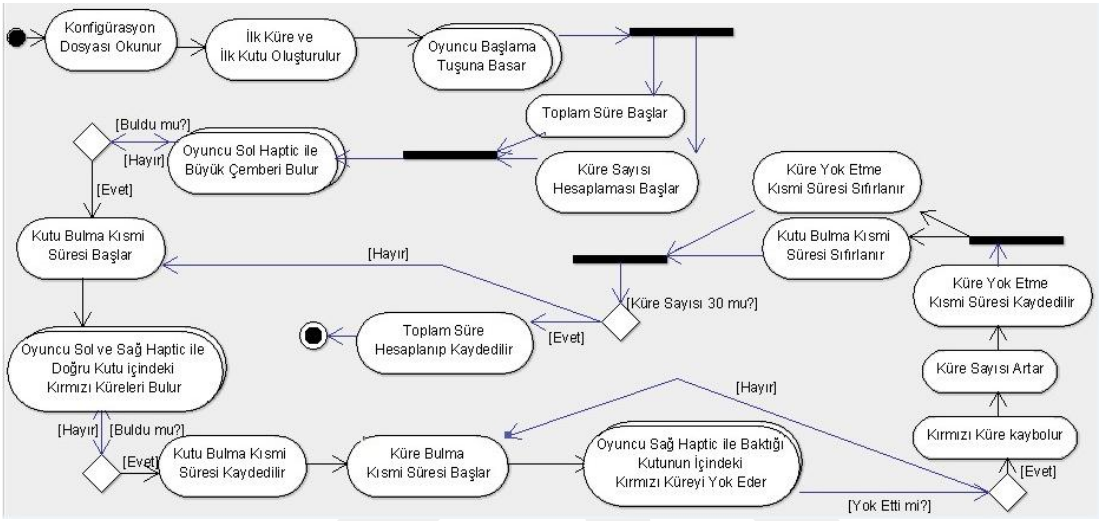
APPENDIX F: UML-ADE Model Used For Scenario 1 (Turkish)



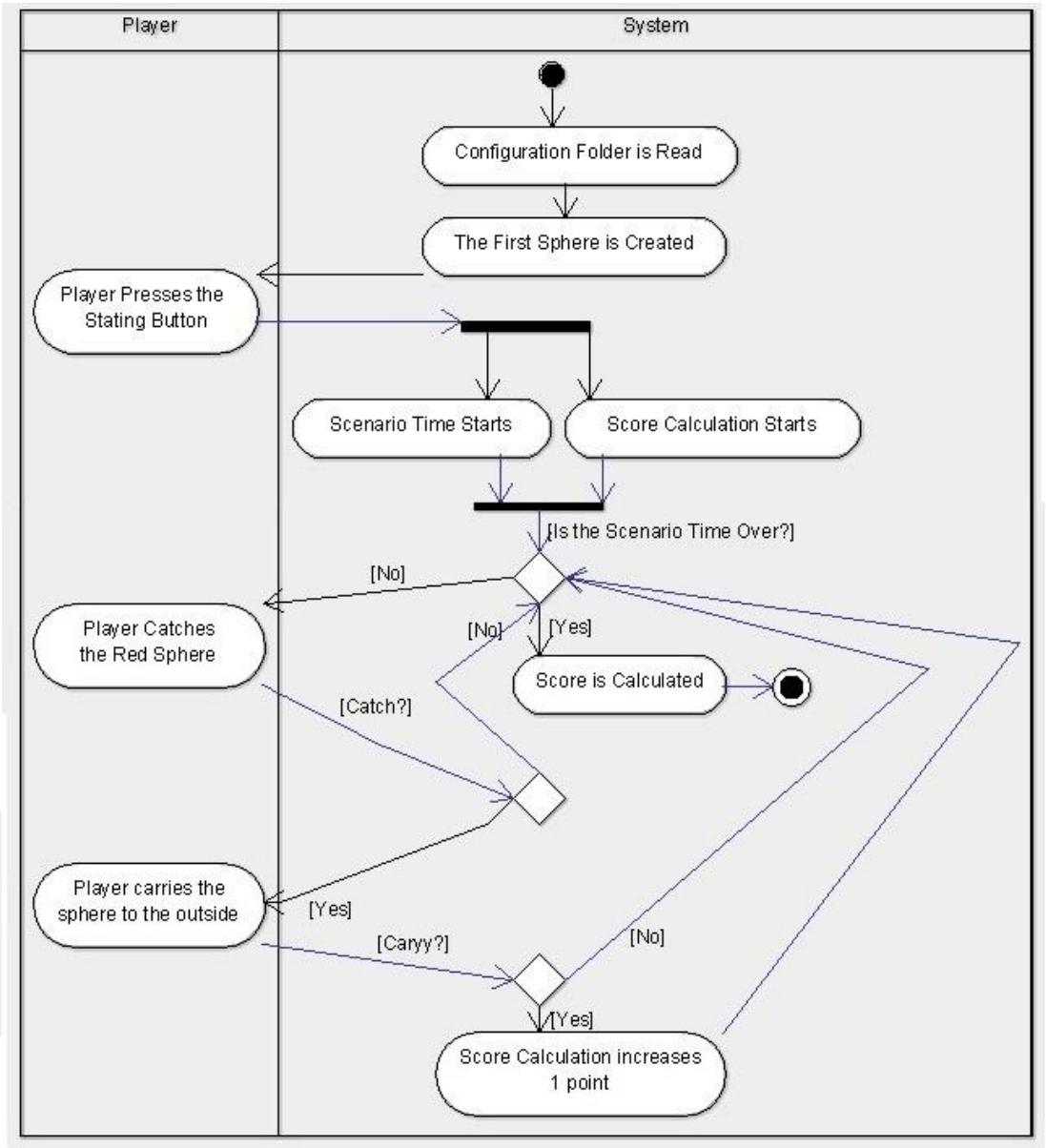
APPENDIX G: UML-AD Model Used For Scenario 2 (Turkish)



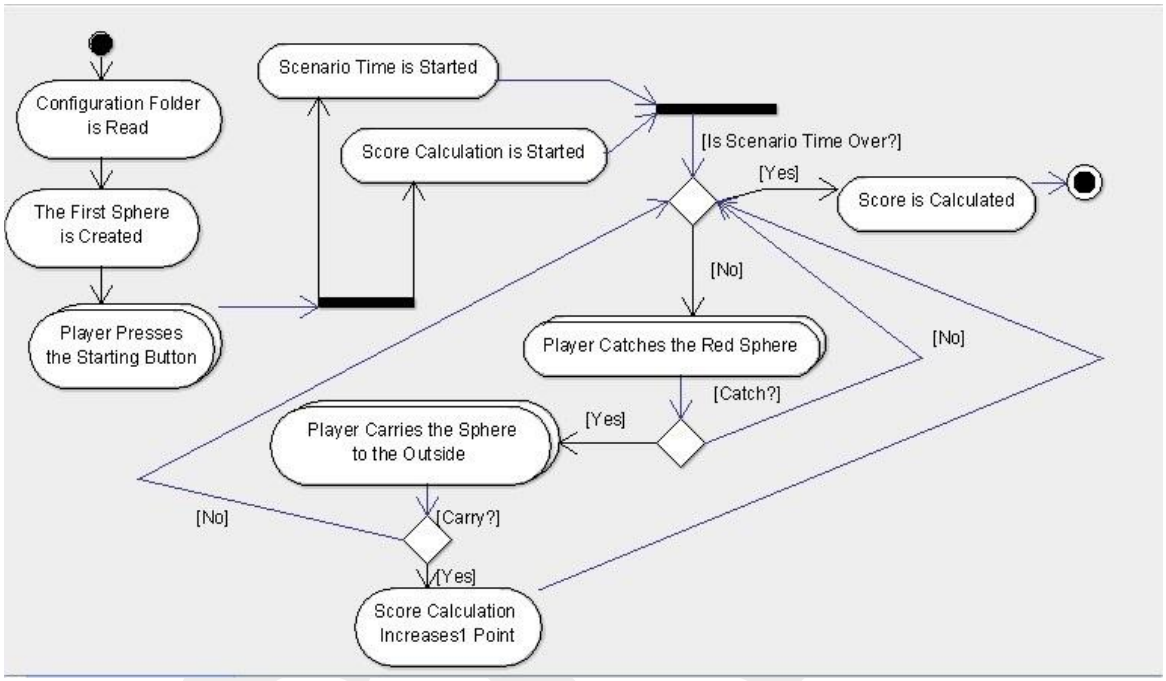
APPENDIX H: UML-ADE Model Used For Scenario 2 (Turkish)



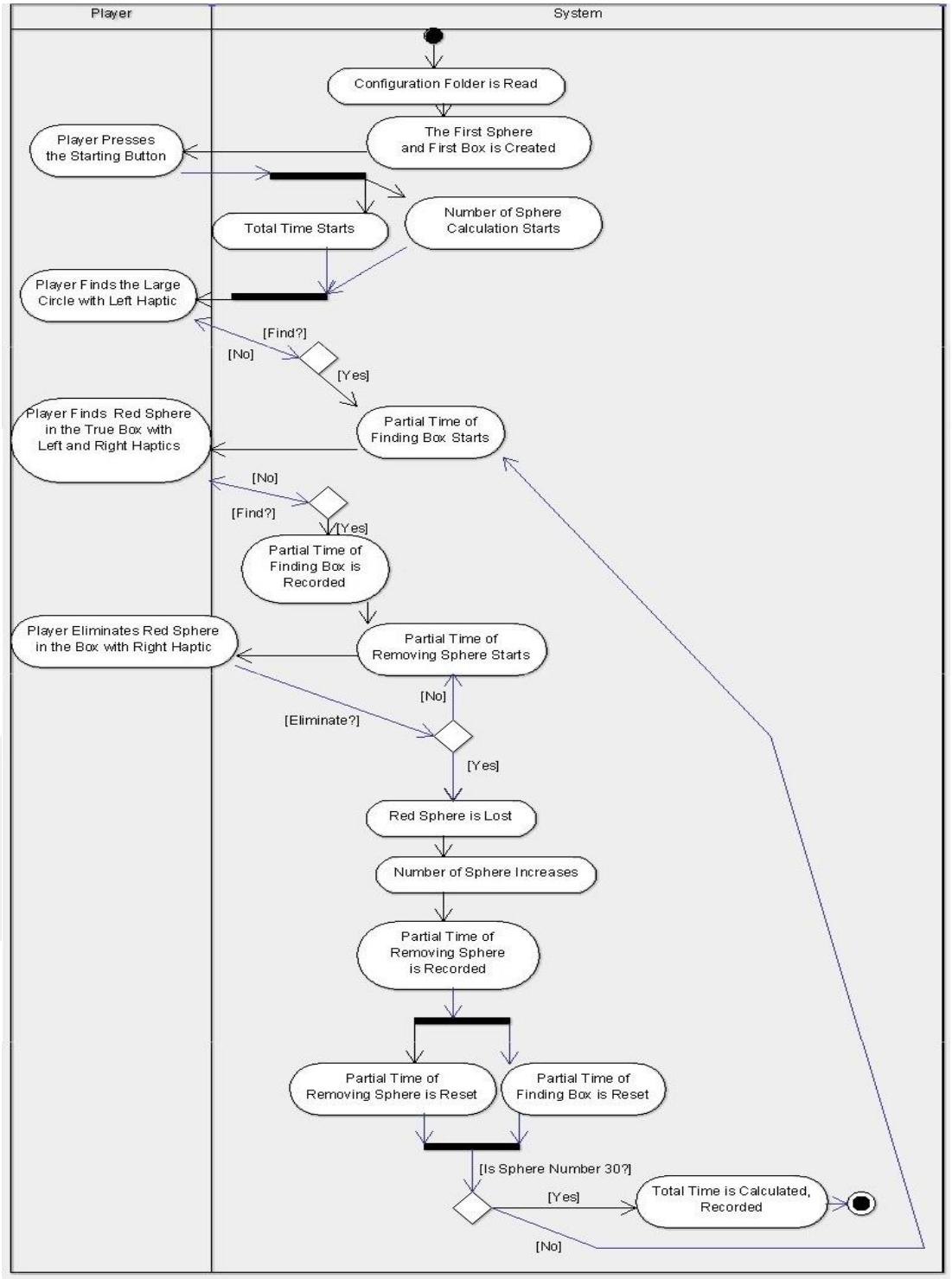
APPENDIX I: UML-AD Model Used For Scenario 1 (English)



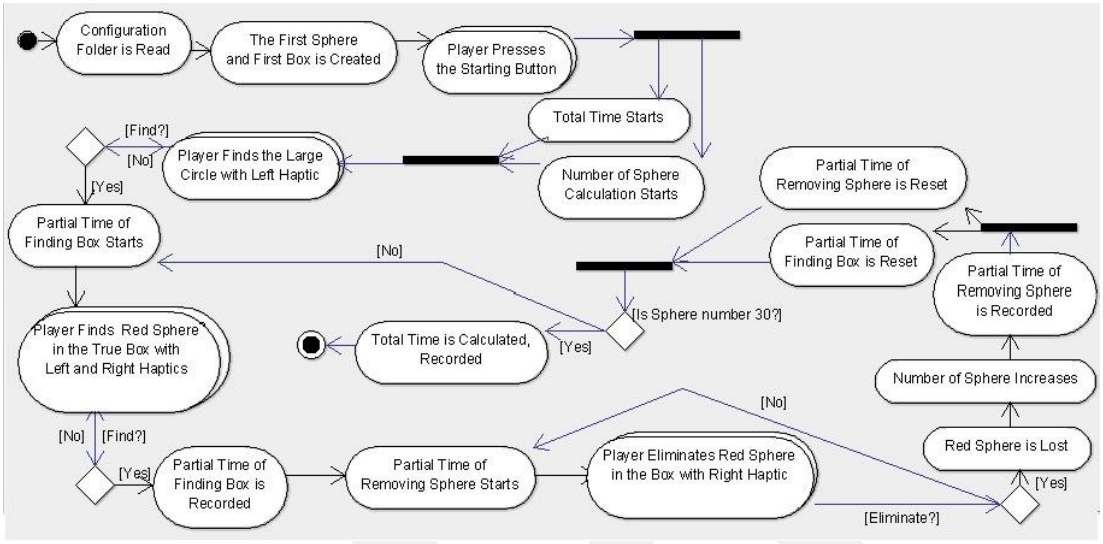
APPENDIX J: UML-ADE Model Used For Scenario 1 (English)



APPENDIX K: UML-AD Model Used For Scenario 2 (English)



APPENDIX L: UML-ADE Model Used For Scenario 2 (Engilish)



APPENDIX M: Questionnaire 1 Used In UML-AD (English)

QUESTIONNAIRE ABOUT UML-AD DESIGN

Some expressions about UML-AD diagram that you currently worked on are given below. About these expressions, choose one of the five alternatives (from 1 to 5).

		STRONGLY NOT AGREE 1	NOT AGREE 2	DON'T KNOW 3	AGREE 4	STRONGLY AGREE 5
1.	I think UML-AD diagram given to us is easy to understand.	☐	☐	☐	☐	☐
2.	How complicated is the given UML-AD diagram? Evaluate it from 1 to 5 (1: very easy - 5: very difficult)	☐	☐	☐	☐	☐
3.	I think I understand the system very well by looking to the given UML-AD.	☐	☐	☐	☐	☐

4. What is your opinion or suggesstion about UML-AD design or intelligibility?

_____
_____
_____
_____

5. If you design the Collection Spheres in the Room game (Scenario no. 1), how can you design? Draw.

6. If you design the Catching Spheres in the Box game (Scenario no. 2), how can you design? Draw.

APPENDIX N: Questionnaire 1 Used In UML-ADE (English)

QUESTIONNAIRE ABOUT UML-ADE DESIGN

Some expressions about UML-ADE diagram that you currently worked on are given below. About these expressions, choose one of the five alternatives (from 1 to 5).

		STRONGLY NOT AGREE 1	NOT AGREE 2	DON'T KNOW 3	AGREE 4	STRONGLY AGREE 5
1.	I think UML-ADE diagram given to us is easy to understand.	☐	☐	☐	☐	☐
2.	How complicated is the given UML-ADE diagram? Evaluate it from 1 to 5 (1: very easy - 5: very difficult)	☐	☐	☐	☐	☐
3.	I think I understand the system very well by looking to the given UML-ADE.	☐	☐	☐	☐	☐

4. What is your opinion or suggesstion about UML-ADE design or intelligibility?

_____
_____
_____
_____

5. If you design the Collection Spheres in the Room game (Scenario no. 1), how can you design? Draw.

6. If you design the Catching Spheres in the Box game (Scenario no. 2), how can you design? Draw.

APPENDIX O: Questionnaire 1 Used In UML-AD (Turkish)

UML TASARIMI HAKKINDA ANKET

Aşağıda üzerinde çalışma yaptığınız UML-AD diyagram ile ilgili bazı yargılar verilmiştir. Bu yargılar hakkında verilen seçeneklerden (1'den 5'e kadar) size uygun olan birini seçiniz.

		KESİNLİKLE KATILMIYORUM 1	KATILMIYORUM 2	KARARSIZIM 3	KATILIYORUM 4	KESİNLİKLE KATILIYORUM 5
1.	Verilen UML-AD Diyagramını kolay ve anlaşılabilir buldum.	☐	☐	☐	☐	☐
2.	Verilen UML-AD sizce ne kadar karışıktı, 1-5 arasında değerlendiriniz (1: az karışık - 5: çok karışık)	☐	☐	☐	☐	☐
3.	Verilen diyagrama bakarak geliştirilecek oyunu çok iyi anladığımı düşünüyorum.	☐	☐	☐	☐	☐

4. UML-AD diyagramın tasarımı ya da anlaşılabilirliği ile ilgili eklemek istedikleriniz / önerileriniz nelerdir?

_____
_____
_____
_____

5. Odadaki Küreleri Toplama oyununu (1. Senaryo) siz tasarlasanız nasıl bir tasarım olurdu? Çiziniz.

6. Kutudaki Cisimleri Yakala oyununu (2. Senaryo) siz tasarlasanız nasıl bir tasarım olurdu? Çiziniz.

APPENDIX P: Questionnaire 1 Used In UML-ADE (Turkish)

UML TASARIMI HAKKINDA ANKET

Aşağıda üzerinde çalışma yaptığınız UML-ADE diyagram ile ilgili bazı yargılar verilmiştir. Bu yargılar hakkında verilen seçeneklerden (1'den 5'e kadar) size uygun olan birini seçiniz.

		KESİNLİKLE KATILMIYORUM 1	KATILMIYORUM 2	KARARSIZIM 3	KATILIYORUM 4	KESİNLİKLE KATILIYORUM 5
1.	Verilen UML-ADE Diyagramını kolay ve anlaşılabilir buldum.	☐	☐	☐	☐	☐
2.	Verilen UML-ADE sizce ne kadar karışıktı, 1-5 arasında değerlendiriniz (1: az karışık - 5: çok karışık)	☐	☐	☐	☐	☐
3.	Verilen diyagrama bakarak geliştirilecek oyunu çok iyi anladığımı düşünüyorum.	☐	☐	☐	☐	☐

4. UML-ADE diyagramın tasarımı ya da anlaşılabilirliği ile ilgili eklemek istedikleriniz / önerileriniz nelerdir?

_____
_____
_____
_____

5. Odadaki Küreleri Toplama oyununu (1. Senaryo) siz tasarlasanız nasıl bir tasarım olurdu? Çiziniz.

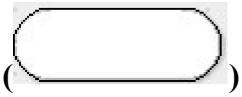
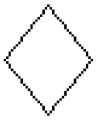
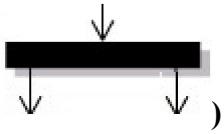
6. Kutudaki Cisimleri Yakala oyununu (2. Senaryo) siz tasarlasanız nasıl bir tasarım olurdu? Çiziniz.

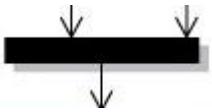
APPENDIX R: Notation Explanation Document For UML-AD (English)

UML-AD EXPLANATION DOCUMENT

UML-AD diagrams are drawn with using ArgoUML tool that is presented to you. In these diagrams, various control flows are used. Control flows mean the flow of a model. According to UML-AD control flows, you should analyze these diagrams.

UML-AD Control Flows:

1. () **Initial Node:** This is initial point. Activity starts with initial point.
2. () **Action:** This node is used to define the all actions. There is no showing message when from an activity to another activity flow.
3. () **Control Flow:** This node supplies connection between action. Control action is used to pass from one action to another action.
4. () **Decision Node:** This node is used to define a condition in an activity. There is a one incoming flow and multiple outgoing flows. There is a situation that is depended on every outgoing flow and to explain this situation, brackets are used.
5. () **Activity Final Node:** This node is final point. When the activity finishes, final point is reached. There can be more than one final node.
6. () **Fork Node:** One incoming flow is separated to more than one outgoing flow that is simultaneous.

() **Join Node:** More than one incoming flow can combine with one outgoing flow that is simultaneous. This node defines to combine the parallel actions and continue with a one action.

Available Defects: During the design of UML diagrams, for the scenario that is given for you, there are some defects about situations that is shown in below.

- Wrong Transition
- Missing Transition
- Irrelevant Finite State
- Wrong Action State
- Missing Final State

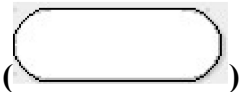
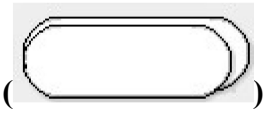
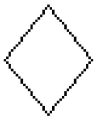
It is expected for you to find defects with marking them on diagrams and save them to the system. You should press “Submit Defect” button after you save every defect to the system. So, time of finding defects is shown in the screen. When the finding defect process is completed, you should press the “Submit All” button and complete this study.

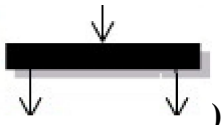
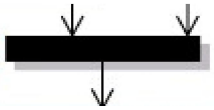
APPENDIX S: Notation Explanation Document For UML-ADE (English)

UML-ADE EXPLANATION DOCUMENT

UML-ADE diagrams are drawn with using ArgoUML tool that is presented to you. In these diagrams, various control flows are used. Control flows mean the flow of a model. According to UML-ADE control flows, you should analyze these diagrams.

UML-ADE Control Flows:

1. () **Initial Node:** This is initial point. Activity starts with initial point.
2. () **Action:** This node is used to define the all actions. There is no showing message when from an activity to another activity flow.
3. () **User Action:** This node is used to define actions that are done by user. There is no showing message when from an activity to another activity flow.
4. () **Control Flow:** This node supplies connection between action. Control action is used to pass from one action to another action.
5. () **Decision Node:** This node is used to define a condition in an activity. There is a one incoming flow and multiple outgoing flows. There is a situation that is depended on every outgoing flow and to explain this situation, brackets are used.
6. () **Activity Final Node:** This node is final point. When the activity finishes, final point is reached. There can be more than one final node.

7. () **Fork Node:** One incoming flow is separated to more than one outgoing flow that is simultaneous.
8. () **Join Node:** More than one incoming flow can combine with one outgoing flow that is simultaneous. This node defines to combine the parallel actions and continue with a one action.

Available Defects: During the design of UML diagrams, for the scenario that is given for you, there are some defects about situations that is shown in below.

- Wrong Transition
- Missing Transition
- Irrelevant Finite State
- Wrong Action State
- Missing Final State

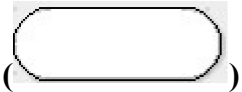
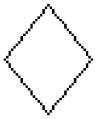
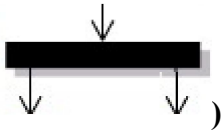
It is expected for you to find defects with marking them on diagrams and save them to the system. You should press "Submit Defect" button after you save every defect to the system. So, time of finding defects is shown in the screen. When the finding defect process is completed, you should press the "Submit All" button and complete this study.

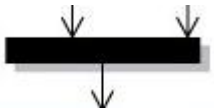
APPENDIX T: Notation Explanation Document For UML-AD (Turkish)

UML-AD AÇIKLAMA DOKÜMANI

Size sunulacak olan UML-AD diyagramları ArgoUML aracı kullanılarak çizilmiştir. Bu diyagramlarda çeşitli kontrol akışları kullanılmıştır. Kontrol akışları bir modelin akışını temsil eder. UML-AD kontrol akışlarına göre diyagramları analiz etmelisiniz.

UML-AD Kontrol Akışları:

1. () **Initial Node:** Başlangıç noktasıdır. Aktivitede akış başlangıç noktasıyla başlar.
2. () **Action:** Tüm eylemler belirtmek için kullanılır. Bir aktiviteden diğer aktivite akışında herhangi mesaj gösterilmez.
3. () **Control Flow:** Eylemler arasındaki bağlantıyı sağlar. Kontrol akışı, bir eylemden diğer eyleme geçişte kullanılır.
4. () **Decision Node:** Bir aktivitede koşul belirtmek için kullanılır. Bir tane gelen akım, birden fazla giden akım vardır. Her giden akıma bağlı bir durum vardır ve bu durumu belirtmek için köşeli parantez kullanılır.
5. () **Activity Final Node:** Bitiş noktasıdır. Aktivite bittiğinde bitiş noktasına ulaşılır. Birden fazla bitiş noktası olabilir.
6. () **Fork Node:** Bir gelen akım, birden fazla eş zamanlı giden akıma ayrılır. Eş zamanlı birden fazla giden akım, eylemlerin aynı anda yapıldığını belirtir.

7. () **Join Node:** Birden fazla gelen akım, tek bir tane eş zamalı giden akımla birleşir. Paralel olarak yapılan eylemlerin birleşip tek eylemle devam ettiğini belirtir.

Mevcut Hatalar: Size verilen senaryo için çizilmiş UML diyagramların tasarımı sırasında aşağıda belirtilen durumlar ile ilgili bazı hatalar yapılmıştır:

- Yanlış Geçişler (Wrong Transition)
- Eksik Geçişler (Missing Transition)
- Gereksiz Bitiş Noktası (Irrelevant Finite State)
- Yanlış Eylem Durumu (Wrong Action State)
- Eksik Bitiş Noktası (Missing Final State)

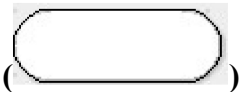
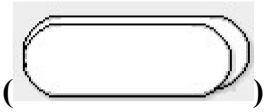
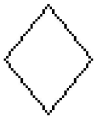
Sizlerden bu hataları bulup diyagramlar üzerinde işaretleyerek sisteme kaydetmeniz beklenmektedir. Her hatayı sisteme kaydettikten sonra "Submit Defect" butonuna basmanız gerekmektedir. Böylelikle hataları bulduğunuz süre ekranda görülecektir. Hata bulma işlemini tamamladığınızda "Submit All" butonuna basarak çalışmayı tamamlayınız.

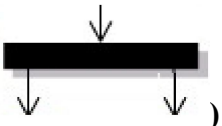
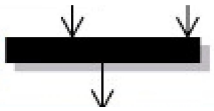
APPENDIX U: Notation Explanation Document For UML-ADE (Turkish)

UML-ADE AÇIKLAMA DOKÜMANI

Size sunulacak olan UML-ADE diyagramları ArgoUML aracı kullanılarak çizilmiştir. Bu diyagramlarda çeşitli kontrol akışları kullanılmıştır. Kontrol akışları bir modelin akışını temsil eder. UML-AD kontrol akışlarına göre diyagramları analiz etmelisiniz.

UML-AD Kontrol Akışları:

1. () **Initial Node:** Başlangıç noktasıdır. Aktivitede akış başlangıç noktasıyla başlar.
2. () **System Action:** Sistemin yaptığı eylemleri belirtmek için kullanılır. Bir eylemden diğer eylem akışında herhangi mesaj gösterilmez.
3. () **User Action:** Kullanıcının yaptığı eylemleri belirtmek için kullanılır. Bir eylemden diğer eylem akışında herhangi mesaj gösterilmez.
4. () **Control Flow:** Eylemler arasındaki bağlantıyı sağlar. Kontrol akışı, bir eylemden diğer eyleme geçişte kullanılır.
5. () **Decision Node:** Bir aktivitede koşul belirtmek için kullanılır. Bir tane gelen akım, birden fazla giden akım vardır. Her giden akıma bağlı bir durum vardır ve bu durumu belirtmek için köşeli parantez kullanılır.

6. () **Activity Final Node:** Bitiş noktasıdır. Aktivite bittiğinde bitiş noktasına ulaşılır. Birden fazla bitiş noktası olabilir.
7. () **Fork Node:** Bir gelen akım, birden fazla eş zamanlı giden akıma ayrılır. Eş zamanlı birden fazla giden akım, eylemlerin aynı anda yapıldığını belirtir.
8. () **Join Node:** Birden fazla gelen akım, tek bir tane eş zamanlı giden akımla birleşir. Paralel olarak yapılan eylemlerin birleşip tek eylemle devam ettiğini belirtir.

Mevcut Hatalar: Size verilen senaryo için çizilmiş UML diyagramların tasarımı sırasında aşağıda belirtilen durumlar ile ilgili bazı hatalar yapılmıştır:

- Yanlış Geçişler (Wrong Transition)
- Eksik Geçişler (Missing Transition)
- Gereksiz Bitiş Noktası (Irrelevant Finite State)
- Yanlış Eylem Durumu (Wrong Action State)
- Eksik Bitiş Noktası (Missing Final State)

Sizlerden bu hataları bulup diyagramlar üzerinde işaretleyerek sisteme kaydetmeniz beklenmektedir. Her hatayı sisteme kaydettikten sonra "Submit Defect" butonuna basmanız gerekmektedir. Böylelikle hataları bulduğunuz süre ekranda görülecektir. Hata bulma işlemini tamamladığınızda "Submit All" butonuna basarak çalışmayı tamamlayınız.