Agent Oriented Software Engineering (AOSE) Approach to Game Development Methodology

By

Rula K. Al-Azawi

Faculty of Technology

De Montfort University, Leicester, UK

A thesis submitted in partial fufillment of the requirements of De Montfort University for the degree of

Doctor of Philosophy

Computer Sciences

February 2015

Abstract

This thesis investigates existing game development methodologies, through the process of researching game and system development models. The results indicate that these methodologies are engineered to solve specific problems, and most are suitable only for specific game genres. Different approaches to building games have been proposed in recent years. However, most of these methodologies focus on the design and implementation phase. This research aims to enhance game development methodologies by proposing a novel game development methodology, with the ability to function in generic game genres, thereby guiding game developers and designers from the start of the game development phase to the end of the implementation and testing phase.

On a positive note, aligning development practice with universal standards makes it far easier to incorporate extra team members at short notice. This increased the confidence when working in the same environment as super developers. In the gaming industry, most game development proceeds directly from game design to the implementation phase, and the researcher observes that this is the only industry in which this occurs. It is a consequence of the game industry's failure to integrate with modern development techniques.

The ultimate aim of this research to apply a new game development methodology using most game elements to enhance success. This development model will align with different game genres, and resolve the gap between industry and research area, so that game developers can focus on the important business of creating games. The primary aim of Agent Oriented

Agile Base (AOAB) game development methodology is to present game development techniques in sequential steps to facilitate game creation and close the gap in the existing game development methodologies.

Agent technology is used in complex domains such as e-commerce, health, manufacturing, games, etc. In this thesis we are interested in the game domain, which comprises a unique set of characteristics such as automata, collaboration etc. Our AOAB will be based on a predictive approach after adaptation of MaSE methodology, and an adaptive approach using Agile methodology.

To ensure proof of concept, AOAB game development methodology will be evaluated against industry principles, providing an industry case study to create a driving test game, which was the problem motivating this research. Furthermore, we conducted two workshops to introduce our methodology to both academic and industry participants. Finally, we prepared an academic experiment to use AOAB in the academic sector. We have analyzed the feedbacks and comments and concluded the strengths and weakness of the AOAB methodology. The research achievements are summarized and proposals for future work outlined.

Declaration

I declare that the work described in this thesis is original work undertaken by me for the degree of Doctor of Philosophy, at the Faculty of Technology, De-Montfort University, Leicester, United Kingdom.

No part of the material described in this thesis has been submitted for the award of any other degree or qualification in this or any other university or college of advanced education. This thesis is written by me and produced using LATEX.

Rula Khalid Al-Azawi

Leicester, United Kingdom. 2014

Publications, Other Output

The research towards this thesis has produced the following publications:

1. R. Al-Azawi, A. Ayesh, I. Kenny, and K. AL-Masruria,"**Analysis of Using Intelligent Technique in Games"** , in 3rd GAMEON-ARABIA'2012 Conference, (Muscat-Oman), pp. 83-87, EUROSIS Conference, 10-12 December 2012.

2. R. Al-Azawi and A. Ayesh, **"Comparing Agent Oriented Programming Versus Object Oriented Programming"** , in the 6th International Conference on Information Technology ICIT'13, IEEE Jordan Chapter,(Amman-Jordan), 8-10 May 2013.

3. R. Al-Azawi, A. Ayesh, I. Kenny, and K. AL-Masruria, **"Towards an AOSE: Game Development Methodology"**, in Distributed Computing and Artificial Intelligence, 10th International Conference. Advances in Intelligent and Soft-Computing series of Springer,(Salamanca- Spain), vol. 217, pp. 493-501, May 2013.

4. R. Al-Azawi, A. Ayesh and M. Al-Obaidi, **"Generic Evaluation Framework for Games Development methodology"**, The 3rd International Conference on Communications and Information Technology (ICCIT-2013): Digital Information Management and Security,(Beirut- Lebanon), pp. 55-60, IEEE Computer Society, 19-21 June 2013.

5. R. Al-Azawi, A. Ayesh, I. Kenny, and K. AL- Masruria,"**Generic Framework for Evaluation Phase in Game development Methodologies"**, in the IEEE Technically Cosponsored Science and Information SAI Conference 2013, (London- UK), pp. 237-243, IEEE Computer Society, October 7-9 2013.

6. R. Al-Azawi, A. Ayesh and M. Al-Obaidi, **" Towards Agent-based Agile Approach for Game Development Methodology"**, WCCAIS'2014 World Congress On Computer Applications and Information Systems, (Hammamet, Tunisia), IEEE Computer Society, 17-19 January, 2014.

7. R. Al-Azawi, A. Ayesh, I. Kenny, and K. AL-Masruria,"**Multi Agent Software Engineering (MaSE) and Agile Methodology for Game Development"**, 4th GAMEON-ARABIA Conference, EUROSIS Conference. (Muscat, Oman) , pp. 116-122.

Other Output

1. R. Al-Azawi and A. Ayesh,," **Agent Oriented Agile Based Game Development Methodology (AOAB) Workshop"**, was conducted in GameLab / Jordan, 26 July 2014.

2. R. Al-Azawi and A. Ayesh,," **Game Development Methodology -Agent Oriented Agile Based (AOAB) Workshop"**, was conducted at the GAMEON'2014 conference, 11 September 2014, University of Lincoln, Lincoln, UK.

3. R. Al-Azawi became the Head of the Judgment Committee in the Telecommunication Regulatory Authority for game competition between undergraduate university students in the Sultanate of Oman, which is a country located in the Middle East. Appendix C illustrates the competition poster and costs, with the game theme "My address is Omani".

Acknowledgments

To my supervisor **Dr.Aladdin Ayesh**, many thanks for being so kind, and so patient to my circumstances, as well as highly cooperative and so supportive. His guidance was crucial to this thesis and he has opened the door for me to the conference and research world, which I really find myself drawn to more than any other academic activity. Dr. Aladdin is my pacemaker. I appreciate his efforts and the support he extended to me to make this work possible.

To **Mr. Peter Hornort** from Staffordshire University, thanks for your efforts and encouragement to me when starting my PhD study, and thanks to your lovely wife "Karen". I have enjoyed your friendship.

To my local supervisor **Dr. Khalfan AL-Masruria**, thanks for all your efforts to encourage me and for answering my enquiries whenever needed.

To my second supervisor **Dr. Ian Kenny**, thanks for giving me the time and help I needed.

To **Mr. Nart and Mr. Hamza** from GameLab, many thanks for the proficient organization of the workshop conducted in Jordan, and for your support inviting professional people from within the game field.

To **Mr. Philippe** the EUROSIS conferences organizer, I am grateful to your cooperation and support in publishing two papers for different International EUROSIS conferences. Many thanks for your support and presentation of our workshop in parallel with the GAMEON conference in the UK.

Dedication

To **My mother**

Thanks for your encouraging me throughout my academic life and thanks for your continuous prayer.

To the loving memory of **My father**

I remember your words that *”your scientific degree and your money is your weapon in the life”.*

To **My family:**

My husband

Many thanks for your patience, your constant encouragement and support. Thanks for introducing me to the great person and supervisor Dr.Aladdin.

My daughters: Maysam and Rahef

Sorry for being away from you and spending so much time stuck to the laptop. All that I have done was to bring you a better life.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABC	Agent Based Computing
ABLE	Agent Building and Learning Environment
ACL	Agent Communication Language
AEF	AOAB Evaluation Framework
AI	Artificial Intelligence
AOAB	Agent Oriented Agile Based Game Development Methodology
AOD	Agent Oriented Development
AOM	Agent-Oriented Methodology
AOSE	Agent-Oriented Software Engineering
A-UML	Agent-UML
AV	Agent/Role View
BDI	Belief Desire Intention
BGE	Blender Game Engine
DV	Domain View
CIS	Computing Science

CS	Computer Science
DV	Domain view
FIPA	Foundation for Intelligent Physical Agents
FPS	First Person Shooter
FSM	Finite State Machines
FuSM	Fuzzy State Machine
GA	Genetic Algorithm
GDD	Game Development Document
GDMF	Game Development Methodology Framework
GEHF	Game Evaluation Heuristics Framework
GOAP	Goal Oriented Action Plan
GTV	Goal/Task View
GUI	Graphical User Interface
HEP	Heuristics for Evaluating Playability
HCI	Humen Computer Interaction
IE	Interactive Entertainment
IS	Information System
IV	Interaction View
JADE	Java Agent Development
JATLite	Java Agent Template

MACR	Multi agent and Cooperative Robotics
MAS	Multi Agent System
MC	Mobile Computing
MLs	Modeling Languages
NCP	Non Player Character
OAA	Open Agent Architecture
OA	Organizational agents
OOD	Object Oriented Development
OOM	Object Oriented Methodology
OV	Organization View
QA	Quality Assurance
QEF	Quality Evaluation Framework
RUP	Rational Unified Process
RTS	Real Time Strategy games
RPG	Role Playing Games
SOC	Service-Oriented Computing
UDK	Unreal Development Kit
UML	Unified Modeling Language

Chapter 1

Introduction

1.1 Motivation and Problem Formulation

Many researchers combine agents and games. This is because many games use agents or multi agents to represent players and Non-Play Characters (NPC) in games. In reality, most game development companies do not follow a specific lifecycle when developing games. They mainly deal with standard software engineering methodologies such as Waterfall or Agile methodology. From a search of the literature, we observed that it is not easy to find a *"common solution for a common problem in games"*. Creating such a methodology would, however, make the communication between developers easier and documentation more understandable, easy to follow, covering all the game elements.

The literature in reviews this study focuses on finding the best game development methodologies. We encountered limitation when searching for generic game development methodologies. Most existing methodology has been created to solve specific problems, such as 'INGENIAS', which merge Multi Agent System (MAS) and Role Playing Games (RPG) [67]. Furthermore, there is no standard methodology covering the entire game life cycle, which is easy to use and generic to the most game genres. Furthermore, we found that in some cases there is a gap between the industrial and academic sectors. After recognizing these problems, we began a survey of existing AOSE methodologies; specifically Gaia, MESSAGE, Promethus, MAS-CommonKAS, O-MaSE, Tropos and MaSE. We compiled a comparative study of these methodologies, selecting MaSE as part of the AOAB methodology. MaSE covers the predictive approach, while Agile covers the adaptive approach. Finally, we needed to prove our AOAB methodology by presenting different evaluation methods that would be useful to evaluate game development methodologies.

1.2 Thesis Objective

During my PhD research work, which took place around three years ago, the author conducted an intensive literature review of the game types and classifications, searching the most popular game development methodologies. A critical analysis was conducted to find the weaknesses in current game development methodologies.

Using software engineering methodologies and other Object Oriented methodologies were not believed to be suitable options, and so a new customized game development methodology was required. However, existing methodologies provide a good grounding. AOAB is a new generic game development methodology and is easy to use, fitting with different game genres. AOAB provides start to end development phases and steps, which provide the game designer and developer with a sequence of steps to follow. Furthermore, AOAB also bridges the gap between the academic developer and the industry sector. As a first step, we analyzed the agent in general, prepared a critical evaluation framework to compare AOSE methodologies and the MaSE methodology selected for adaptation to the AOAB methodology.

The second step, required focus on current software engineering methodologies, which have been used in game development methodology. After a critical analysis of the problems encountered with previous game development methodologies, we selected the Agile methodology to created an Agent Oriented Agile Base Game Development Methodology (AOAB). Finally, we focused on the individual AOAB phases: the requirement phase, creating Game Design Documents (GDD), analysis, design, implementation and an evaluation phase in AOAB. The evaluation phase focused on expert and end user evaluators, who used different criteria such as playability, quality, usability and enjoyment to evaluate games in each iteration before the game's final release.

1.3 Research Question

The main research question of this thesis is:

"Can we create a game development methodology that could be generic, standard and easy to use for different game genres?"

The main objective of the thesis is to propose a game development methodology, namely AOAB methodology. The life cycle of AOAB game creation is clear, and the overall goal of the game is easy to identify. The documentation of the game is hierarchical, and organised by each iteration. Moreover, the GDD template provides most of the information needed by organisations, from a structural perspective. This investigative work began in 2011 and, from previous work, it is observed that most game developers use Agile methodology in general and, in some cases, create their own methodology that fits with their requirements or specific game genres.

Initially, a literature review was carried out to determine current techniques and methodologies. A strong relation was observed between games and agents. The focus was then on gaining a clear idea of the common and special characteristics of building a game. The proposed AOAB methodology aims to be as standard and generic as possible, and combines agent and Agile methodologies. This unique combination requires wide knowledge of a variety of technologies and sciences. However, the research does not covers all aspect related to agent, Agile and game. For example, the research did not address agent communication language.

Once a complete methodology was established, AOAB was evaluated by sharing the methodology with different types of peoples through conducting workshops and participating in a conference in order to obtain feedback and evaluate the methodology in a real life context. The industry experiment removes the gap between the academy and industry sectors. In the workshops, participants were invited from both academia and industry as well as some Indies, to participate in the workshops and complete the questionnaire to provide their opinion of AOAB. It was clear from this feedback that AOAB fills most of the gaps between industry and academia. On the other hand, AOAB is not suitable for building small games, as it is designed to use more than one iteration and covers many details that are not necessary when building small games. AOAB is designed to facilitate the easy creation of a new version of game following game release by creating strong documentation. In the academic case study, students were happy to create diagrams in their work, rather than just textural descriptions, and AOAB outlines simple steps to make this easy for students to do.

1.4 Research Methodology

Selection of a particular research methodology is an important decision that will directly affect the achievement of the research objective and the research outcome. There are many research methodologies available to meet the research aims, complicating the possibility of selecting the methodology that will most effectively meet our requirements and the demands of scientific investigation. Despite making a decision regarding method selection, it is often necessary to use a combination of methods to fully understand a problem [56]. However, no standard procedures or rules can be applied to design the research, although there is a set of considerations that can be used as guidelines to align the research aims and methodologies. The classification of the research question(s), the data collection methods, the nature of the problem statement, the research subjects, and the available resources are all criteria providing important input into the research design process [101].

Our research methodology comprises the following main steps:

1. Literature review and problem definition of current game development methodologies and a critical analysis of AI in games.
   Deliverable: Publish two conference papers.

2. Perform intensive analysis of AOSE methodologies and select a suitable methodology MaSE to adapt to the game development methodology by using a common evaluation

framework.

Deliverable: Publish the third conference paper.

3. Intensive work on the evaluation phases in AOAB. We have identified that the evaluation game is an important component of the game industry, specifically prior to game release. We have produced a generic evaluation framework that encompasses important measures such as playability, usability, enjoyment...etc.

   Deliverable: Publish the fourth and fifth conference papers and submit first journal paper.

4. Study the current game development methodologies, which are based on software engineering methodologies; select Agile game development methodology to adapt the proposed methodology.

   Deliverable: Publish the sixth conference paper.

5. Finalize all AOAB phases and produce the final version of AOAB.

   Deliverable: Publish the sixth and seventh conference papers and submit second journal paper.

6. Evaluate the AOAB using suitable evaluation methods, such as a case study, experiment and presenting a workshop for both the academic and industry sectors. The evaluation section will describe the research methods for use in the evaluation, describing why these particular methods are appropriate for the evaluation.

Havner et al [68] designed a research framework model for an information science framework as shown in Figure 1.1. His Model was established in two complementary stages. Design science addressing research through the construction and evaluation of artifact design to fulfill the requirements identified. Behavioral science addresses research through the development and justification of requirements.

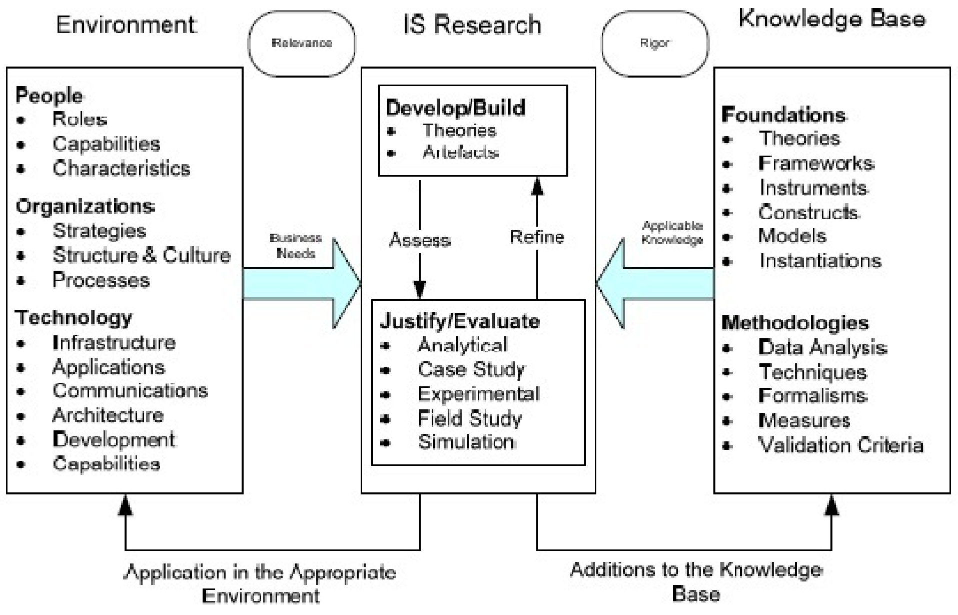


Figure 1.1: Information System Research Framework for Hevner[68]

Figure 1.2 depicts a map of our research plan, establishing requirements and objectives in the context of the Haven information system framework.

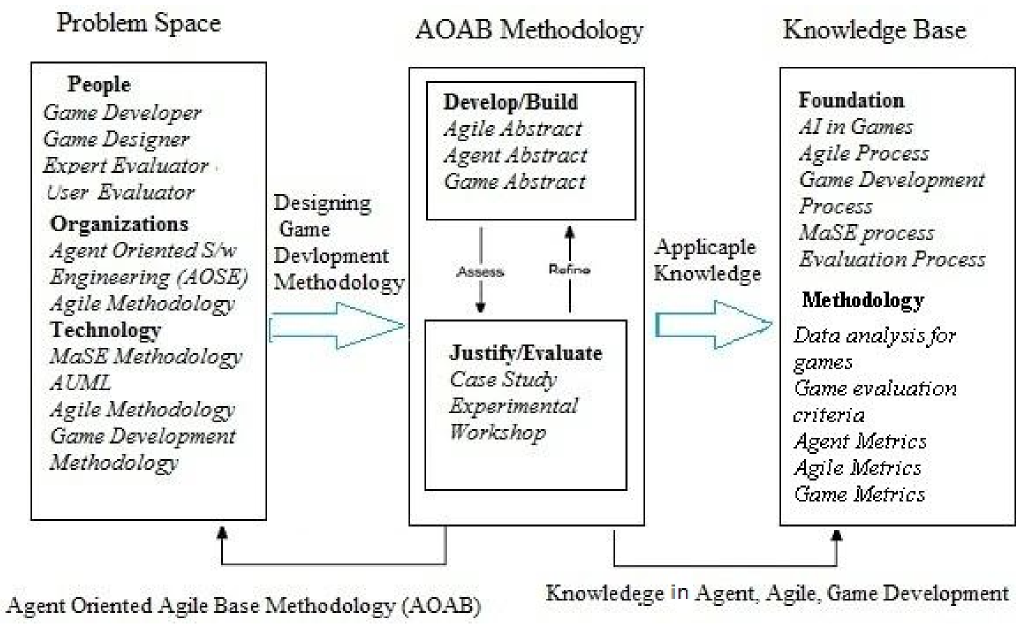


Figure 1.2: Our Research Methodology in Context of Hevner Framework

1.5 Contribution of Thesis

In this thesis, novel game development is introduced based on Agile and MaSE methodology. This thesis presents five main contributions. The first contribution is to enhance the game development methodology by providing a novel game development methodology, termed the AOAB methodology. AOAB is a hybrid methodology, which covers both adaptive and predictive approaches. Agile represents an adaptive approach, and MaSE represents a predictive approach. The second contribution proves the validity of the AOAB methodology, helping the developer and programmer to use different evaluation methods. The aim of the AOAB methodology is to ensure it is general and usable with different game genres. The third contribution is to investigate the effect of a proposed new methodology to create cooperation between the game industry and researchers by adding additional features such as time management, team work policy, and project management. In this thesis, we propose an investigation of how the game designer and developer deals with game creation from the first prototype until the point of the final game release. The fourth contribution is to introduce new and complete coverage of the evaluation phase into game design. The evaluation phase prior to final game release plays an important role throughout the game industry. We consider these problems in association with the evaluation phase, because it is an important component of game creation and can resolve problems prior to final game release. Furthermore, the game development life cycle focuses on combining all the required Agent-UML (AUML) diagrams to facilitate the work and provide for straight forward documentation in the game update, or in the creation of new versions of games. The final contribution is to create a generic Game Design Document (GDD) template to suit different game genres for use by Indie game developers and expert game developers.

In this research, the work done to ensure the validity of the contribution is summarized as follows:

- Create a novel game development methodology, which is a hybrid between the adaptive and predictive approach.
- Evaluate the AOAB using different evaluation methods according to general and standard methodologies.
- The AOAB should be suitable for industrial and academic use.
- The AOAB needs to cover the full life cycle and direct more attention to the methodology evaluation phase.
- Create a generic GDD template suitable for different game genres and for all game developers.

The proposed AOAB game development methodology is evaluated in many ways to investigate its strengths and weaknesses. The results provide an outline for future work.

1.6 Thesis Outline

This thesis is comprised of ten chapters, including this one. It is also arranged as follows:
Chapter Two: This contains a critical analysis of the current literature regarding computer gaming, and presents the relationship between the game and the agent. The chapter reviews different game genres and classifications according to different criteria. The primary goal of the chapter is to analyze the AI techniques used in the game by conducting a critical review of known AI requirements in games. The second part of the chapter surveys the game engine and architecture, which helps us to select the best game engine for our work.
Chapter Three: This chapter contains the literature of agents and intelligent agents in games. Both this and the previous chapter provide information to be used in the creation of a game development methodology. The review focuses on agent types, architecture and agents in games. Furthermore, it presents the compression between agent and object. In this chapter, we present AUML and select Agenttool3 software as part of our work.

Chapter Four: This chapter presents the starting point for our AOAB methodology creation. In the first part of the chapter, we present the Agile methodology, which is usually used in game development. The second part of the chapter focuses on a survey of existing AOSE methodologies; such as Gaia, Tropos, MaSE and Prometheus. We also make a comparative study between MaSE and Tropos to select the best one for use in the game development methodology. The final part of this chapter focuses on the game development methodology framework (GDMF), which is based on a critical analysis and evaluation framework to select AOSE methodology that will match game methodology requirements.

Chapter Five : The first part of this chapter presents an extensive analysis and development of AOAB, establishing problems with the current game development methodology. The second part of this chapter focuses on the AOAB suggested, which was adopted from MaSE and Agile methodologies. The final part of this chapter presents the completed phases of AOAB with a full description.

Chapter Six : This chapter presents a comparative study of the different evaluation methods available to evaluate new methodologies. The evaluation methods are then used to evaluate and assess the AOAB methodology, which covers three aspects of the evaluation. First by employing a survey approach, and second, by pursuing a formal experimental approach. Finally, by following a case study approach. Furthermore, we have created a procedural framework to evaluate AOAB and combine it with different evaluation time scales. The next two chapters will cover the evaluation methods to evaluate the AOAB in detail.

Chapter Seven : This chapter offers a case study approach of the industry sector, including introduction of full details of the game, beginning with creating a concept paper and GDD, and continuing until the game's final release. Furthermore, this chapter provides proof of use of the AOAB methodology, with its deployment techniques in the industry sector. In this chapter, we use 3Dunity as a game engine and Agenttool3 as AUML software, to create a complete and comprehensive game.

Chapter Eight : This chapter covers the second and third evaluation methods for evaluating AOAB. The first part covers formal experimental study for undergraduate students in two groups, working to create a mobile game. The first group uses AOAB, while the second uses the Agile methodology. The second part of this chapter covers a survey approach by conducting two workshops. The participants of the workshops are drawn from academic and industry sectors. The results, as indicated in this workshop have involved critical analysis.

Chapter Nine : This chapter outlines the conclusion of AOAB, which includes strengths and weakness following the AOAB methodology evaluation. Furthermore, it includes a critical analysis of the AOAB methodology, based on the results from previous evaluation methods as covered in the two previous chapters.

Chapter Ten : This chapter concludes the overall work delivering recommendations for future enhancement and work.

Appendix A : Contains the template for the GDD. This template facilitates the work of the game designer and developer. In this template, we organize and cover all the important aspects that need to be specified as general for games of different genres.

Appendix B : Contains game evaluation criteria tables.

Appendix C : Contains the game competition poster.

Appendix D: Contains the industry section in detail. This section contains diagrams required for the analysis and design phase. It also includes a snapshot of implemented games, some Java Script codes and communication messages that have been provided to the player.

Appendix E: This contains the details of the mobile computing course, which comprises part of the academic evaluation.

Appendix F: This contains all the materials used in both workshops, such as: the workshop details, invitation letters, questionnaires, pictures, the software tools required for the workshops and finally a feedback table to conclude the positive and negative comments made by the participants.

Chapter 2

The Structure of Computer Games

2.1 Chapter Overview

Computer games are an increasingly popular application within the domain of AI research. Computer gaming is also a big and growing industry, with a huge variety of games currently available. Recently AI has been used to resolve the issues that affect classical gaming. Computer game design involves the design of behavior, platforms, and decision making processes. AI fields have been used in new generation computer games, such as autonomous agents, decision making tools, scheduling, path finding and learning. The success or failure of games now depends on integrating some form of AI into the games. The game developers face a challenge when incorporating AI into the games. The main goal of AI in games is not only to find the optimal behavior required to win the game, but also to make the game believable, enjoyable and as fun as possible. In this chapter, we will begin with game definition and classification. We will then cover the AI in gaming in detail. Game engines are an important component of game creation. We have presented the most popular game engine, and this chapter will cover game architecture and game engineering.

2.2 Game Definition and Classification

We define a game in general as *"structured playing usually undertaken for enjoyment and sometimes used as an educational tool"*. Many other authors define a game as
"A game as an activity that must have the following characteristics: fun, separate, uncertain, non-productive, governed by rules, fictitious" [14].
"A game is a system that players engage in an artificial conflict, defined by rules, those results in a quantifiable outcome"[118].
"A game is a form of art in which participants; termed players, decisions making in order to control resources through game tokens in the pursuit of a goal"[44].
"A game is an activity among one or more independent decision-makers seeking to achieve their objectives in some limiting context" [16].

"A game is a form of play with goals and structure" [96].

However, Crawford's definition is the most suitable definition of a game, because he mentions the relationship between the game and its agents. A game according to his definition is *"An interactive, goal-oriented activity, with active agents to play against, in which players, including active agents and NPC can interfere with each other"*[45]. We have observed that games could be defined in a variety of different ways. Furthermore, games could also be classified into many types. A huge variety of games are currently available. Devising a single game classification is a complex process, because there are many different categories of classification, as shown in Figure 2.1 [55][43][92]. Decisions regarding choices of classification are based on different criteria as follows:

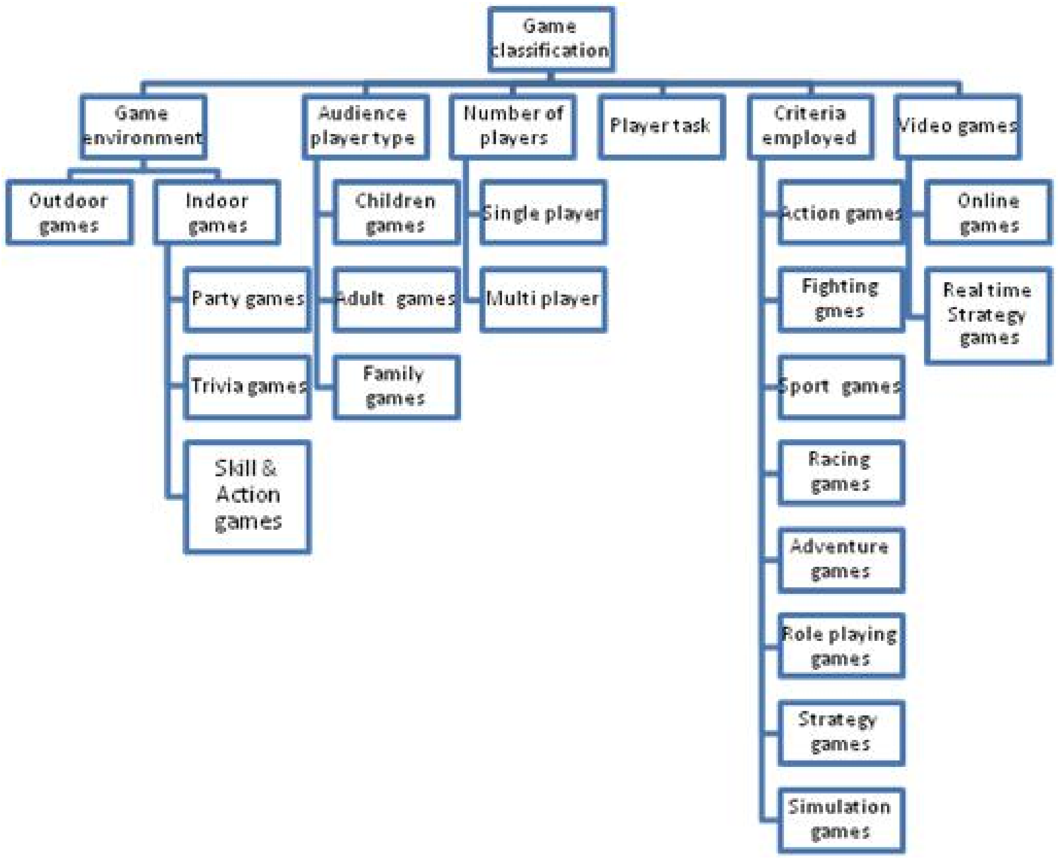


Figure 2.1: Game Classification

- Depend on the game environment such as Outdoor games (play activity) and Indoor games, which include a table game, such as a Board game and Card games and /or into one of these three sub categories:
  - Party games that require physical action or stunts
  - Trivia games that require questions and answers
  - Skill and Action games that require players to throw, roll, balance and build
- Depend on the audience for whom the game is targeted; i.e. children's games, family games or adult games.
- Depend on the number of players, such as single player e.g. Puzzle, or two players or multi players.
- Depend on what the player does. This is referred to as game play.
- Depend on criteria employed such as:
  - Action games: games played through a series of levels (set in a virtual world) with a variety of enemies. In most action games the player is the first person shooter (FPS) as in games such as Street fighter and Mortal Kombat.
  - Sports Games: games that portray different kinds of sports, creating virtual representations of sporting activities. Some simulation games simulate a sport's environment such as Astro Race.
  - Adventure Games: games that tend to be plot based. Adventure games involve activities such as exploration, information gathering and problem solving. [77]. Puzzle games are also an example of adventure games. Solving puzzles can unlock access to new areas in the game world.
  - Strategy Games: games that depend on creating a military-type battle scenario. The design is not focused only on characters, but also on resources (building

of defensive, offensive units and troops) and the need to organize and manage battles. Star Craft II: Wings of Liberty is a popular strategy game.

– Role Playing Games (RPGs): these games assume the role of characters acting in a fictional setting. The main RPG is played with a handful of participants, mostly face-to-face. RPGs generally require the player to undertake a quest. Dragon Quest and Pokemon are examples of an RPG.

– Real Time Strategy Games (RTS): This is a game in which the players focus on logistics and resource production, and manage combat and war. These games usually involve quick decision making and fast reflexes. Players need to command armies and gather resources at the same time. This can prove immensely challenging, and most RTS games have developed many in game tools to help players deal with this task. Common examples of RTS games are Age of Empires, Age of Empires 3 and Empire Earth.

2.3 Artificial Intelligence in Games

Many AI techniques and concepts are used with games. The most popular techniques include the finite state machine, scripting fuzzy logic agent, etc., and a few game developers use decision trees, neural networks and genetic algorithms [126]. We need to present the most important AI criteria and explain how they are linked to game requirements. In general, game AI focuses on creating the appearance of intelligence in many ways. Table 2.1 shows the main AI criteria for games and explains how they are linked to AI techniques [24].

Table 2.1: AI Requirements Criteria.

AI requirement	AI Criteria
Manage game world	FSM, FuSM, Script
Optimal solution	GA

Path finding	A*, Streeing
Decision making	NN, Agent, Fuzzy logic
Learning	NN, GA
Developing game strategy	GA

This section explores the use of AI, as used with games:

1. Finite State Machine (FSM) in Games

   FSM is a set of a finite number of states, each set for input, output and state transition function. The FSM in games divides the game object's behavior into logical states. In this case, the object has only a single state for each different type of behaviour. FSM is realised by simple if-then statements, it uses graphical representations, which are part of the nine diagrams defined by the Unified Modeling Language (UML)[150].

   In a game's AI, the possible ways in which to use an FSM are endless. FSM could be used to specify module unit behaviors in an RTS. In addition, it could use be used to parse input from the human player or even to simulate the emotion of a NPC [133]. FSM is a natural choice for game developers when designing NPCs. It is used in most commercial computer games and video games like Age of Empires, and in FPS video games, such as Quake, Quake 2, Doom, and Half Life or in RTS game like Enemy Nations. The games that currently use FSM could also use other options, such as fuzzy state and Neural Network.

2. Fuzzy Logic in Games

   Boolean logic contains only true or false statements, while fuzzy logic allows intermediate values, such as 'rather hot' or 'very fast', to describe continuous or overlapping states that can arise in mathematics. Fuzzy logic can represent a concept using the smallest number of fuzzy values. The fuzzy logic used in decisions can be based on incomplete or erroneous data, which could not be used in Boolean logic. It is useful for decision making, behavior selection and input/output filtering.

Fuzzy logic is used to determine how frightening something is for the player, for the NPC to decide how much they reveal to the "player". The most common use of this technology on commercial games is in video games such as Platoon Leader; real time tactic action games such as SWAT 2; and strategy games such as CCTP. If we impose a nonlinear problem or no simple solution, or if NN is not applicable then fuzzy logic is appropriate.

3. Neural Network in Games

   Neural Networks (NN)contain a number of relative components linked to a system, and which will be able to produce output based on the identification of patterns in data. Many games that use neural networks can "learn" through experience or training and include the ability to make decisions.

   A neural network can also be improved continuously, which means the player will be challenged to change the style of the strategy of play, meaning they must not reuse the same strategy. Neural network has been used in adventure games such as Battle-cruiser 3000AD; racing games such as Dirt Track Racing, and strategy games such as Fields of Battle. The most interesting application of a neural network in AI is 'Battle-cruiser 3000AD'. In this game a neural network controls the NPC.

4. Genetic Algorithm in Games

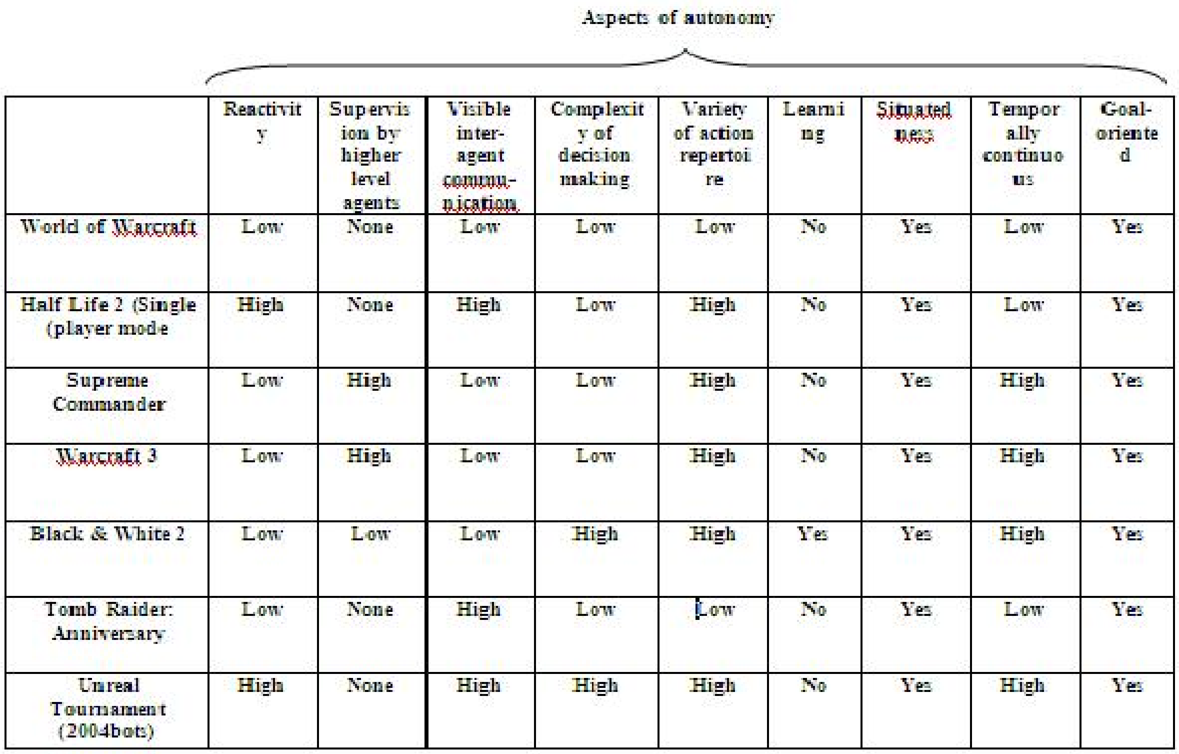
   Genetic Algorithm (GA) is used to find an optimal solution, and can be effectively used in machine learning to evaluate and find a solution to specific problems. GA starts with a small number of initial strategies, creating an entire population, seeking out solutions and evaluating each population to solve a problem. GA is a suitable solution when a problem or game contains a large enough search domain. It is also useful for solving nonlinear problems.

GA is used in RTS to adapt the computer strategy, to exploit the human player's weakness and define the behavior of individual units rather than a group. 'Cloak, Dagger, DNA' (CDD) are examples of GA use in RTS games. In CDD, the players share in the 'DNA' that is responsible for monitoring and storing performance. GA is used in RPG or FTS to evolve behaviors associated with characters and events.

5. Intelligent Agent in Games

   This is a software agent seen in an environment and acting in that environment to attain goals using an evaluative function, also called a heuristic function, to help the agent to decide on the best value. In addition, agents have studied many problems in AI. Games are ideal environments for agents because they use realistic environments, but there is a limitation in available information, as decisions must be made under time and pressure constraints. Generally, agents in games use a sets of FSMs and any other techniques or any combination of some or all of the techniques that resolve specific problems and make it possible to send messages.

   The agent has the ability to take decisions and perform tasks to attain their goals as humans do. Every game that includes an AI can be said to be using some form of agent. In FTS or RPG, the monster would be an active agent and more suited to simply reacting to what is happening in the game. Strategy games need to be planned carefully, especially when planning a movement that will happen later in the game. A good architecture for an RTS agent is necessary to ensure success. In Empire Earth, the RTS contains several components called managers. There are managers for civilizations, buildings, units, resources, research and combat. The civilization manager is the highest level manager responsible for the player's development and for coordinating between other managers. The other managers are lower-level and responsible for sending requests and reports to each other. Weddle et al [142] creates Figure 2.2 which shows the links between the most popular agent characteristics and the most popular games.



	Aspects of autonomy								
	Reactivity	Supervision by higher level agents	Visible inter-agent commu-nication	Complexity of decision making	Variety of action repertoire	Learning	Situatedness	Temporally continuous	Goal-oriented
World of Warcraft	Low	None	Low	Low	Low	No	Yes	Low	Yes
Half Life 2 (Single (player mode	High	None	High	Low	High	No	Yes	Low	Yes
Supreme Commander	Low	High	Low	Low	High	No	Yes	High	Yes
Warcraft 3	Low	High	Low	Low	High	No	Yes	High	Yes
Black & White 2	Low	Low	Low	High	High	Yes	Yes	High	Yes
Tomb Raider: Anniversary	Low	None	High	Low	Low	No	Yes	Low	Yes
Unreal Tournament (2004bots)	High	None	High	High	High	No	Yes	High	Yes

Figure 2.2: Link Between Popular Agent Characteristics and Popular Games [142].

6. Machine Learning in Games

   This is a process of learning in games, which generally implies the adaptation of behavior for opponent players in order to improve performance. Machine learning may help to cover the search space in computer games and search efficiently for successful combinations of parameters[116]. Machine learning can either take place online or offline.

   Machine learning is also used in NPC. Machine learning can be employed to automate the creation of intelligent NPC behavior. Machine learning appears in old games such as Tic-Tac-Toe, Backgammon, Go, Othello, and Checkers. Recently, machine learning techniques have begun to appear in video games, as well as in fights, in first- and third person shooter games and also in strategy games.[29]

7. Fuzzy State Machine (FuSM) in Games

   There is a mix between fuzzy logic and FSM. Instead of only using an ”on” or ”off” state, we add ”almost on” or a ”little on”. FuSM are increasingly used in game play because they add interesting and varied responses to the NPC. Therefore, the player can interact with the NPC that can depict various degrees of ‘mad’, ‘wounded’ or ‘helpful’, and the player can also test different experiences and obtain different outcomes in similar situations when playing a game.

   FuSMs were used in FPS to make the enemy appear intelligent. Based on the elements in the battle situation, fuzzy logic is used to enable enemy characters to run away when losing a battle[116]. Also, they are used in CCTP, which is an RTS game. FuSM is used in RPG, to target the points of NPC and agents, and is ideal for controlling the behavior of game characters and for imbuing them with variable actions and reactions.[77]

Table 2.2 shows the links between the popular AI in games.

Table 2.2: AI In Games.

AI	Game type	Game name
FSM, FuSM	FPS- RTS	Doom, Quake, unreal, CCTP
GA	RTS	CDD
Agent	Action game RTS	S.W.A.T2, CCTP, Empire Earth
NN	Racing, Action, Adventure, Strategy games	Dirt Track Racing, Battle-cruiser 3000AD, Heavy Gear
GA	RTS	CDD
Fuzzy logic	Adventure, Strategy games	SWAT, CCTP

2.4 Game Engine

Game engine is a software system designed for the creation and development of video games. The core functionality that is typically provided by a game engine includes a rendering engine ("renderer") for 2-D or 3-D graphics, a physics engine for collision detection and response, and a scene graph for the management of both static and animation models, sound, scripting, artificial intelligence, networking, streaming, memory management, threading, etc. [98] There are many game engines available and the majority are user friendly. Modern games are frequently developed using game engines, which can be deployed on personal computers, game consoles, pocket PCs and mobile devices [110], and combine several technologies from the area of computer science, such as: graphics, artificial intelligence, network programming, languages and algorithms. Game engines are designed to manage multiple levels of programming expertise. They are divided into roll-your own versions (lowest level), mostly ready game engines (mid level) and point and click engines (highest level). In the following, we will explain the most popular game engine.

2.4.1 3DUnity

Point and click engines are becoming more common because they include a full tool chain that allows the user to point and click to create a game. 3DUinty is an example of this game engine. It is a free game engine development tool for individual and professional use, which is one of the best game engines on the market. It is very easy for developers to use and they can easily get to it. Because of the friendly interface it is relatively easy to understand almost all the sections within a few minutes of using it. There are many tutorials available online to find out about it. It has a very high graphic quality and always keeps improving. The graphics almost look real and have real time lighting and shadows, and more. 3DUnity has great realistic physics such as rigid body. Colliders and soft bodies help to make the game more real, making it easy to attach them to the game. With plug-ins in the asset store

it becomes much easier to make the game more realistic. It is easy for most users to use 3DUnity as any additional help required is provided by 3DUnity solutions on the asset store. Using 3DUnity one click is enough to publish the game to multiple platforms, such as Xbox 360, PS3, Wii, Mac, Pc, Linux, Web, Adobe Flash, Android and IOS. With all the platforms supported it is easy to distribute the game [9]. The Unity engine's scripting/programming is built on Mono, an open source .NET implementation. There are three programming language choices: C Sharp, Unity Script (which is basically Javascript), and Boo (a language intended to be similar to Python) [1].
3DUnity also allows the game to be developed in multiple platforms. The advantages are the following:

- Provide students with a degree of freedom in developing games for the platform of their choice.
- Learn about the strengths and constraints of different platforms, for example user interface, viewing screen size, resolutions, resources such as memory and processor power, storage for saving and loading the game, in game development [149].

For our game creation, we have selected 3DUnity as the game engine because it operates at a high level and can be published easily to many platforms.

2.4.2 Unreal Development Kit (UDK)

UDK is an example of a mostly ready game engine that is instantly ready for prime time, out of the box, with rendering, input, GUI and physics. The UDK engine was used for the AI, graphics and every other feature of the game aside from the physics. This was outsourced another engine called 'Havok' to power the physics in the game; this is an example use of middle ware. UDK has two version one is the free for non-commercial use, and for commercially usable. UDK is a somewhat daunting for people who are new to game development, although it is not impossible to learn how to use it, but it requires extra learning. It is anticipated that in the new UDK version the interface will be much better and more intuitive. The graphics are pretty mind blowing, the lights are advanced in the game engine, and can dynamically render the light maps and illuminations for runtime, and most of the other staff. This makes it possible to use hundreds of objects and textures from the game to create new levels, but importing one's own data is also an option. UDK supports two different programming languages: Kis-met, used for basic scripting, and script for other scripting. The developers can work only on PC, and their work can be published to iOS, Android, Xbox, PS3 and Windows [3].

2.4.3 XNA

Microsoft XNA is one of the best options in the game industry currently. It's a .NET programming package that lets the user prepare any small to medium sized game they are capable of coding, and then it allows the user to share their creation with the world on Xbox Live [15].
Brains is an A.I library for use with XNA games. It consists of a few building blocks to get the user quickly up and running with an A.I prototype that is simple to implement into the game. It is currently only 2D but would not require too much modification to support 3D. Brains A.I will also contain a list of Agent types. This type is used to provide autonomous behaviors to the game. An Agent stores some simple positioning properties, such as Position, Radius and the Cells the Agent is currently in. It also stores the desired orientation and the desired position of the Agent for use with a Locomotion controller. An Agent can store a set of feelers, which can be used to poke data around the world [6].

2.4.4 Blender 3D

Blender 3D is an open source freeware program maintained by the Blender Foundation. Learning Blender is not easy but it has limitless possibilities and provides an understanding

of the complexities of computer animation. Blender has developed over time and its development has evolved new releases (versions) of the program that have been made available. The program reached a stage where the developers called for a complete overhaul of the Graphical User Interface (GUI) [2]. Blender is used to make real-time interactive content and is written from scratch in C++ as a mostly independent component; it includes support for features such as Python scripting and OpenAL 3D sound.
Simulations of realistic physics behavior in blender are possible due to the Blender Game Engine (BGE) module. This module uses a system of graphical logic blocks (called logic bricks) to control behavior and visualize objects located in the 3D scene. All the object attributes and methods (functions) of the BGE are available for users through Blender Python API, and all of these can be used in the custom controllers implemented in the Python programming language[128].

2.4.5 Game Maker

Game maker is a game engine created by YoYo company and it uses the Android, Browser, iOS, Mac, PC and Windows Phone platforms. Game Maker is primarily used for 2D game creations with flexibility in design, and can be bad or good depending on user requirements. Its interface hides most of the coding involved and lets users input data rather than writing lines of code, but those who want to dive into coding can do so in advanced areas of the program. If the user is prepared to undertake programming then it is possible to customize the creation into many genres, but if the user is concerned about programming, then there will be restrictions on sidescroller and top down perspective games [7].

2.5 Game Architecture

One of the important considerations in the game research area is the architecture of the game that the developer will use. We have adopted general game architecture from [27] as shown in the Figure 2.3. This game logic holds the game's story. The audio and graphics are the modules that help the writer to narrate the story to the player. The event-handler and the input modules supply the game logic with the player's next action. The level data module is a storage module for details about static behavior, and the dynamics module configures the dynamic behavior of the game's characters [27].

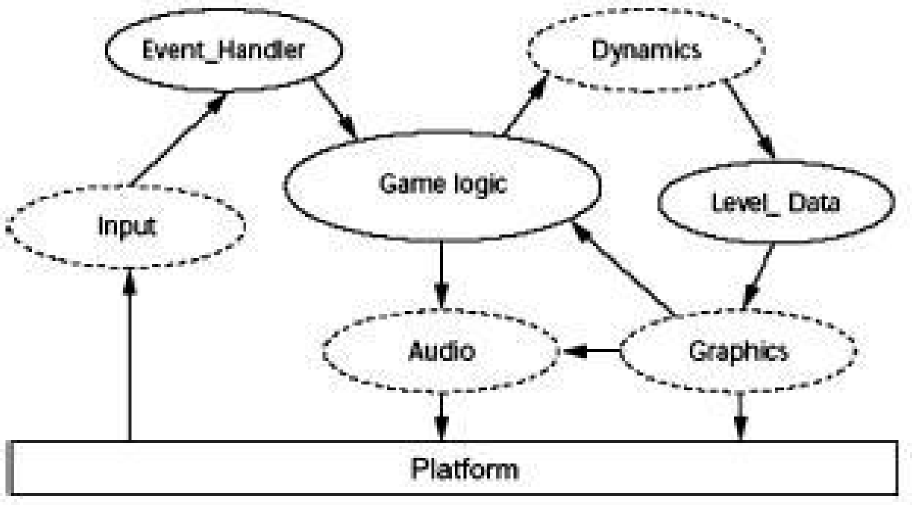


Figure 2.3: General Game Architecture[27].

2.6 Game Engineering

Contemporary game creation is an incredibly complex task, much more difficult than someone might initially imagine. This is because of the increased complexity combined with the multidisciplinary nature of the process of game development (art, sound, gameplay, control systems, artificial intelligence, human factors, among many others). The interaction with traditional software development creates a scenario that also increases this complexity. To make this connection, we need a methodology to take into account software engineering expertise in the field of gaming. As we know, the gaming industry is a very powerful sector within the entertainment industry, earning billions of dollars in profit and creating trillions of hours of fun [112].

There is a strong relationship between agent and games as explained previously in this chapter. The proposed AOAB will combine the concept of agent with the creation of a generic game development methodology. In the following two chapters, we will explain the agent in general and later we will explain AOSE methodologies in detail.

2.7 Summary

In this chapter, an analysis of the use of AI in commercial gaming and in research games is presented. The main goal when using AI in games is to simulate intelligent behavior and to make the games believable, challenging and fun at the same time. It also helps to find the most optimal action for the available information.

In terms of marketing, the focus is on features, such as graphics and physics known to include AI features in commercial games. Some of the AI techniques used were FSM, Agent and decision making, because they are simple. Other AIs need extra resources faster CPUs and extra run time, such as GA and NN. In addition, many games could be used with more than one set of AI techniques, depending on the game requirements.

Furthermore, some AI concepts could linked together, for example NN could be used with fuzzy logic. In conclusion, the aims is to enable academics and commercial experts to use standard design and implementation features in game AI.

Chapter 3

Agents Technology: An overview

3.1 Chapter Overview

Nowadays software agents have evolved to a stage at which they have the ability to act autonomously, proactively and cooperatively. These new developments make it possible to delegate intelligent agents for use in games. These agents can learn the preferences of a player and search for offers that match those preferences. The concept of an agent is become important in AI, games and computer science.

The aim of this chapter is to introduce the most important theoretical and practical issues related to agent. This chapter began with an explanation of agent technology, and the agent typology is presented with an explanation of the relationship between them. However, knowing the difference between agent and object will help the user attain a clear vision. One of the main and important elements used in the game is the Multi Agent System(MAS). Furthermore, we have presented a description of agent communication and the platform. Agent modeling language will be used in our AOAB methodology. The final section explains the agent oriented service and intelligent agents used in games.

3.2 Agent Technology Overview

The term agent has many definitions and yet no consensus has been reached about what it means. A simple definition of an agent is *a software unit that possesses autonomous behavior toward its own service and relationships with other agents.* However, the most common definition in the software agent community is that stated by Wooldridge and Jennings [147]. They state that an agent is a software program that has the following properties:

- **Autonomy:** The agents operate without the direct intervention of humans or others, and have control over their actions and internal states;

- **Social ability:** Agents interact with other agents (and possibly humans) via exerting some form of control over their actions and internal states;

- **Reactivity:** Agents perceive their environment (which may be the physical world), as a user via a graphical user interface, a collection of other agents, the Internet (or perhaps all of these combined).

- **Pro-activeness:** Agents do not only simply respond to their environment, but also exhibit goal-oriented and goal-directed behavior by taking the initiative.

From the viewpoint of AI researchers, an agent is a computer system with the attributes mentioned above. It is either conceptualized or implemented using concepts more usually applied to humans, such as knowledge, belief, intention, and obligation (and sometimes emotion). Some additional attributes of an agent are:

- **Mobility:** The ability to move around an electronic network. The agent is used heavily on the Internet to support Internet applications on the Internet framework

- **Veracity:** The assumption of not communicating false information knowingly.

- **Benevolence:** The assumption of not having conflicting goals.

- **Rationality:** The assumption of acting with a view to realizing its goals, instead of preventing them.

There are different impacts from agent technology design in application domains, which include assistance in the designing of complex distributed systems, which is a source technology in computing systems and a model for complex real world systems. One of the most well known agent applications is video games, which have become a large component of many peoples' lives. The application of agent technology in video games has many aspects. One of the obvious benefits of video games is the elimination of risk to human life that is involved in a real world application. They also make an excellent test for techniques in Artificial Intelligence. However, there are some general attributes and characteristics that distinguish agents [88].

3.3 Agent Typology

Agents have different definitions based on their application. Because of this multiplicity, different types of agents have been introduced. Agents can be classified according to several ideal primary attributes, which each agent should exhibit. We have identified a minimal list of three: Autonomy, Learning and Cooperation [20].

- **Autonomy** the agents can operate on their own without the need for human guidance. Hence, agents have individual internal states and goals, and act in such a manner as to meet their goals on behalf of the user.
- **Cooperation** with other agents is paramount. It is the reason for multiple agents instead of just one. In order to cooperate, agents need to possess a social capability.
- **Learn** for agent systems to be truly smart, they have to learn as they react and/or interact with their external environment [147].

We use these three minimal characteristics in Figure 3.1 to derive four types of agents to include in the typology: Collaborative Agents, Collaborative Learning Agents, Interface Agents and truly Smart Agents or Intelligent Agent [106].

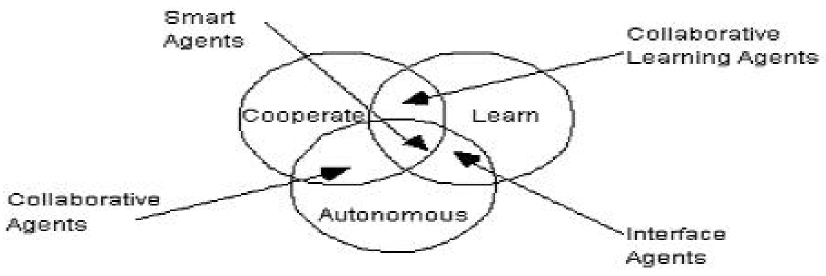


Figure 3.1: A part View of an Agent Typology [106]

In principle, by combining all the previous parameters, we have summarized all the types as shown in Figure 3.2.

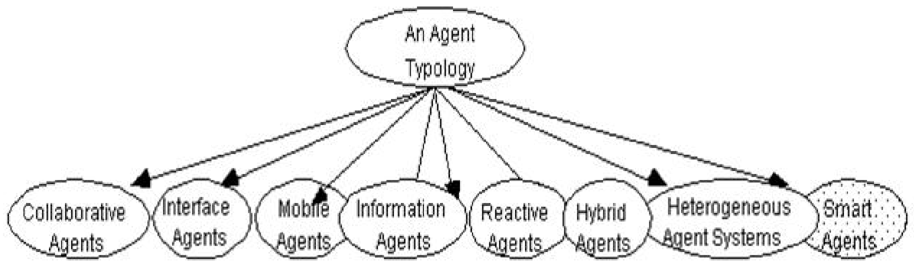


Figure 3.2: A Classification of Software Agents

3.4 Agent Versus Object

Agent Oriented Development (AOD) is an extension of Object Oriented Development (OOD). The word "Development" is sometimes interpreted as "Programming"; in addition, it is frequently interpreted to include the full development process covering requirements specification and design, in addition to the programming itself. Shoham et al [123] have discussed the compression between agent and object, as shown in Table 3.1

Table 3.1: Object Oriented Programming Versus Agent Oriented Programming

	OOP	AOP
Basic unit	Object	Agent
Parameter defining	unconstrained	beliefs, commitments, capabilities, choices,...
Process of computation	message pass and response methods	message pass and response methods
Type of message	unconstrained	inform, request, offer, promise, decline,...
Constraints on methods	none	honesty, consistency

1. **Basic unit**

   The points listed below show differentiation between agents and objects:

- Agents may communicate using an Agent Communication Language (ACL), whereas objects communicate via fixed method interfaces.
- Agents have the quality of volition. That is, using AI techniques, intelligent agents are able to judge their results, and then modify their behavior (and thus their own internal structures) to improve their perceived fitness.
- Objects are abstractions of things like invoices, and agents are abstractions of intelligent beings. They are essentially anthropomorphic. Note that this does not mean that agents are intelligent in the human sense, only that they are modeled after an anthropomorphic architecture, programmed with beliefs, desires, etc.

2. **Parameters defining state of basic unit**

   AOP agents comprise beliefs, commitments, choices, and the like and communicate with each other via a constrained set of speech acts, such as inform, request, promise, decline. The state of the agent is termed its mental state.

3. **Process of computation**

   An object's message may request only one operation, and that operation may only be requested via a message formatted in a very exacting way. An object oriented message broker has the job of matching each message to exactly one method invocation for exactly one object.

   Agent-based communication can also use the OO invocation method. However, the demands that many agent applications place on message content are richer than those commonly used by object technology. While ACL is formal and unambiguous, its format and content varies greatly. In short, an agent message could consist of a character string, whose form can vary while obeying a formal syntax. Meanwhile the conventional OO method must contain parameters whose number and sequence are fixed.

4. **Type of message**

   Agents are commonly regarded as autonomous entities, because they can look out for their own set of internal responsibilities. Furthermore, agents are interactive entities capable of using rich forms of messages. These messages can support method invocation as well as informing agents of particular events, asking something of the agent, or receiving a response to an earlier query. Finally, because agents are autonomous they can initiate interaction and respond to a message in any way they choose. In other words, agents can be thought of as objects that can say "No" as well as "Go".

Due to the interactive and autonomous nature of agents, little or no integration is required to launch an application physically. Objects, on the other hand, are conventionally passive with their methods being invoked under a caller's control. The term autonomy barely applies to an entity whose invocation depends solely on other system components [6].

5. **Constrain on method**

   Usually, object classes are designed to be predictable to facilitate buying and selling reusable components. Agents are commonly designed to determine their behavior based on individual goals and states, as well as the states of ongoing conversations with other agents. While OO implementations can be developed to include nondeterministic behaviour, such thinking is common in agent-based modes.

3.5 Multi Agent System (MAS)

There is no fixed definition of a MAS, only an agreement regarding the most common features like multiple agents acting in one environment. Each agent with specific goals. The communications are between the agents themselves and between agents and the environment. The actions affecting the common environment of all the agents in order to solve problems that are difficult or impossible for an individual agent to solve. MAS is the subfield of AI that aims to provide both principles for the construction of complex systems involving multiple agents and mechanisms for the coordination of independent agents' behaviors. Some domains require MAS. In particular, if there are different people or organizations with different (possibly conflicting) goals and proprietary information, then MAS is necessary to handle their interactions [20]. From an individual agent's perspective, MAS differs from single agent systems, most significantly in that the environment's dynamics can be affected by other agents. In addition to the uncertainty that may be inherent in the domain, other agents intentionally affect the environment in unpredictable ways. Thus, all MAS can be viewed as having dynamic environments [129].

MAS is a promising approach and has the ability to meet new demands. The intelligent agent must have appropriate characteristics (i.e. autonomy, pro-activity, etc.), as these are necessary to meet the requirements of the new application. Unfortunately, there is a disconnection between the advanced technology created by the multi agent community and its application in industrial software. The obstacles to industrial adoption have been the focus of several discussions. Jennings, et al. [75] mentioned two major obstacles to widespread adoption of agent technologies within the industry:

- The lack of complete methodologies and processes to help designers to specify, analyze, and design agent-based applications.
- The lack of industrial strength agent-based toolkits.

An alternative approach to defining industrial strength methodologies that have gained support in the AOSE community is situational method engineering, which promotes flexibility in MAS methods and processes [50].

In a MAS, each agent is aware that it does not possess a global view of the problem and that it cannot solve the problem in isolation, thus it relies on interaction and coordination with others. Nevertheless, it is still programmed to operate autonomously to compete for the satisfaction of its own self-interest, which it believes are benevolent to the overall goals of the group. Fundamental to a MAS environment is the ability for agents to demonstrate social interaction with other agents. As with human social contexts, how agents go about interaction depends on their role and the relationship they have with the target agent [145]. Within the AOSE area, multiple methodologies have been proposed to guide the MAS development process, such as: Gaia, Tropos, INGENIAS, MaSE, MASSIVE, etc. Taking into account several of these methodological proposals we can see that different abstractions or terms have been proposed for the characterization of the MAS; the abstractions most commonly identified are:

- Agents: autonomous and proactive software entities, which achieve their objectives by interacting with each other and are present in a particular environment;
- Actors: an abstraction of autonomous behaviour, internal or external to the system, which has some interest in it and helps define roles.
- Roles: Define the behavior of the agent, and have associated goals and specific tasks to be carried out within the context of the organisation;
- Goals: Define the objectives of both the general system of each actor. Each goal may relate to functional aspects (associated with the services) or functional (associated with quality of service);
- Tasks: A structured set of activities essential to achieve a goal;
- Restrictions: The restrictions allow us to define the desired behavior for both the organization and each agent;
- Interaction between agents: Typically, the agents operate within a context in which they need to cooperate, compete or communicate solely with them to achieve their own goals;
- Interactions with the environment: Agents typically operate in an environment with which they may have to interact (detect and affect) depending on their roles, and their current status;

- Resources: These are specific components of interaction with the environment;

3.6 Existing Agent-based Development Methodologies

The aim of this section is to introduce the AOSE methodologies in general. Those methodologies will be explained in details in Chapter four pages 47.

An agent-oriented software engineering methodology is a business process of developing software, equipped with distinct concepts and modeling tools, in which the key abstraction used in its concepts is that of an agent [18].

Luck et al., [93] surveyed the existing agent-based development methodologies and classified them based on their approach, scope basis, phases they cover, syntax and semantics application area, and type of agency support. The diagram in Figure 5.4 outlines classification of the existing agent-oriented software engineering methodologies.

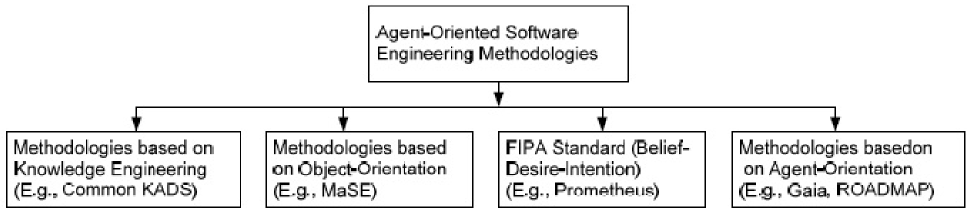

Agent-Oriented Software Engineering Methodologies
Methodologies based on Knowledge Engineering (E.g., Common KADS)
Methodologies based on Object-Orientation (E.g., MaSE)
FIPA Standard (Belief-Desire-Intention) (E.g., Prometheus)
Methodologies basedon on Agent-Orientation (E.g., Gaia, ROADMAP)

Figure 3.3: Categorization of AOSE methodologies [93]

The existing non-agent software development methodology does not possess the capabilities to design agent abstractions or address the software agent abstraction. This is because agent software system has unique abstraction and for the success of its analysis, design and implementation a tailor-made development process is a prerequisite requirement[26]. We have explained the non-agent methods that have been used in games in details in Chapter four pages 43. In Chapter five pages 68, we have explained the problem with current game development methodologies that doesn't deal with agent concept in games.

3.7 Agent Communication

Agents have the ability to communicate with other agents, applications, and humans. This communication includes sending and receiving messages to achieve their goals, or those of the society/system in which they operate. Communication can enable agents to coordinate their actions and behaviors to build more coherent systems. The degree of coordination is the extent to which unnecessary activities can be avoided. Cooperation is coordination among non-antagonistic agents, while negotiation is coordination among competitive agents [145]. The Foundation for Intelligent Physical Agents (FIPA) is an ACL, and is a set of one or more message parameters that include for example sender, receiver, performative role, and content which is expressed in a content language.

3.8 Agent Platforms

Agent platforms are used to provide more realistic modeling of agents in the real world. The platform becomes necessary for agents to communicate with each other using appropriate protocols, to notify agent's presence on a platform, and present common standards for agents to work together. The platforms differ according to their features, abilities, and common standards. There are some tools and environments that support agent developers, which are mostly based on the Java programming language. These include ZEUS toolkit, Java Agent Development (JADE), JACK Intelligent Agents, Java Agent Template (JATLite), Open Agent Architecture (OAA), Foundation for Intelligent Physical Agents-Open ,Source (FIPA-OS), IBM's Agent Building and Learning Environment (ABLE), and Agent Builder [59].

3.9 Agent Modeling Language(AML)

The modeling language for MAS is an important subfield in the research into MAS technology. Researchers and organizations have proposed some methodologies in the development of MAS for the purpose of helping developers to ease the development process and solve the internally complex problems of MAS [122]. In the software engineering domain, Unified Modeling Language(UML) offers a standard way to specify, visualize, modify, construct and document the artifacts of an object oriented software-intensive system that is under development. The most significant motivation driving the development of AML was the extant need for a ready-to-use, comprehensive, versatile and highly expressive modeling language suitable for the development of commercial software solutions based on multi agent technologies [138].

The AgentTool supports the methodology analysis and design phases. Through this tool, it is possible to represent security, authentication and report issue using the roles, Concurrent Tasks and Agents Class Diagrams. This provides the analyst with the responsibility to monitor the project progress and keep the diagrams updated throughout the project phases [125].

3.9.1 AgentTool 3

A useful agent tool, which has been used with MaSE methodology is AgentTool 3 (aT3); this tool is a product of the Multi agent and Cooperative Robotics (MACR) Laboratory at Kansas State University. AgentTool is a Java-based graphical development environment, designed to help users analyze, design, and implement MASs. The initial version of aT3 is an Eclipse plug-in that will give the agent system designer unprecedented flexibility, yet still retain the verification capabilities previously provided [10]. In our practical work we have used AgentTool to create a goal diagram, role diagram, and an agent class diagram. The AgentTool3 web page [10] contains the latest version to download and includes tutorials and examples. We have used the software QSEE, which is UML model software to create a sequence and deployment diagram.

3.10 Agent Service- Oriented Computing

Service-oriented computing (SOC) is a computing paradigm that utilizes services as fundamental elements for developing applications/solutions. SOC has emerged as a powerful paradigm for designing distributed and Internet based systems. The interaction between agent and service can enhance the function and range of the agent; while the dialog between Agent Based Computing (ABC) and SOC can integrate bilateral benefits from both, which are more suitable for building enterprise applications. In general, agent and service are two independent computational concepts. ABC presents unprecedented computational intelligence of flexibility and autonomy when modeling complex software systems. However, ABC faces many challenges from distributed and especially Internet computing[37]. SOC is good at integrating and managing Internet based enterprise, and wrapping legacy applications. The capabilities of ABC and SOC are quite complementary. For the internal qualities of applications, it is the agent and ABC that do well; while service and SOC are more suitable for implementing the software as a service initiative, and application to application architecture. The integration of each can benefit the other in the construction of a more powerful computational system [69]

3.11 Intelligent Agent in Games

The question that remains is "What is an intelligent agent?" The qualities of an intelligent agent fulfil the four properties of agency, as explained in the definition of agent. They also include additional concepts including notions on the Belief Desire Intention model (BDI) of agency as an example of the application of such notions, emotional abilities and the representation of the agent through visual means, perhaps through a cartoon or an animated face [104]. Intelligent agents have been used in a vast range of applications. For this reason, it is of value to identify the application areas of greatest relevance to work on designing agents for interactive entertainment (IE) purposes. For IE agents, the environment is a virtual world in which interaction takes place and becomes ever more complex. A number of attempts have been made to design NPCs for computer games using agent-based techniques. This work identifies a potential intelligent agent architecture for use in IE applications. Agents in IE applications must be able to engage the user (or player) in interesting and, obviously, entertaining interactions. In addition, NPCs must appear to engage other NPCs in interesting social interactions.

3.12 Summary

Agent technology is used in a wide variety of applications, ranging from comparatively small systems to large, complex, and mission critical systems. In this Chapter, we have presented the main concepts linked to agents. Furthermore, we have presented the most popular agent typologies, and the relationship between them. Intelligent agents have been used for a vast range of applications particularly in the context of gaming.

Chapter 4

Critical Review Of Methodologies

4.1 Chapter Overview

In method engineering, the literature uses several terms. The most popular among these are methodology, method, process and model. In our thesis, the terms method and methodology are used synonymously with a process model, while the process is used as an instance of the method or process model. In this chapter, we will begin by describing a Software Engineering methodology that is used in game development methodologies, such as Agile methodology. The next step is to present the AOSE methodology to select one as a hybrid, with an Agile methodology, to create our AOAB methodology. The main aim of this chapter is to present current methodologies, and establish our game development methodology framework, which includes different criteria to select a suitable AOSE methodology as a game development methodology.

4.2 Overview of Software Development Methodologies in Games

Game development has evolved to include large projects employing hundreds of people and development time is now measured in years. Unlike most other software application domains, game development presents unique challenges stemming from multiple disciplines, which contribute to the games. There are many methodologies available in traditional systems and software development. Some of these methodologies include Waterfall, Incremental, Prototyping and Spiral. Each is structured as either linear or iterative, although sometimes a hybrid of both, and is usually used in game development methodology. Most linear manner methodologies are classified as predictive, even if they contain some iteration; although these usually follow sequential phases, such as Waterfall methodology. The prototyping methodology involves breaching the system into small segments, and involves the user in the process. The Spiral methodology combines linear and iterative frameworks. Spiral development breaks the projects down into the number of cycles, all of which then follow a set of increasingly larger steps.

The majority of methodologies selected for use by game developers can be described as predictive, i.e. comprehensively planning a separate task prior to actual development; or adaptive, i.e. using multiple iterations and prototypes to shape a game and its design based on feedback and analysis [70]. In the following section, we focus on the most popular adaptive software development methodology 'Agile', which is mostly used in game development methodology.

4.2.1 Agile Methodology

Agile methodology is based on implementation of documentation with customer collaboration, and has the ability to solve problems and make precise changes quickly. Agile methodology is used as an alternative to traditional methods in game development. It retains essential practices of traditional methodology with a focus on other project dimensions, such as collaboration with the user at all stages of development. Furthermore, it depends on iterative and incremental development, with a very short iteration that helps to provide custom-made solutions.

As the use of Agile developments has increased, a number of different methodologies have surfaced. Some were derived from Agile, others from systems that have been in use, but these were never fully defined or applied to software development. One such method was Scrum [97]. The main characteristics of Agile methodologies were: customer cooperation, simplicity, individuality, interaction, adaptability and being incremental. These characteristics have been brought together to understand an approach to game development based on an Agile methodology [65]. The Agile methodology as mentioned earlier is an iterative and incremental approach. It achieves quality and productivity through iterations. Each iteration or Sprint phase includes a software development team working through a full software development life-cycle, including planning, requirements analysis, designing, coding, unit testing, and acceptance testing, as shown in Figure 4.1, which was adapted from [136] and [99].

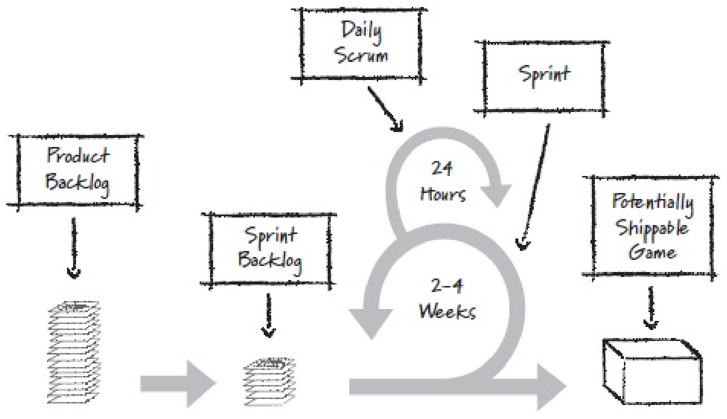


Figure 4.1: Agile Methodology Diagram [136] [99]

The Agile phase approach diagram, which is used by Keith [80] as shown in Figure 4.2, shows that Agile methodology is based on iterations that could trigger new iterations before completion of the previous iteration.

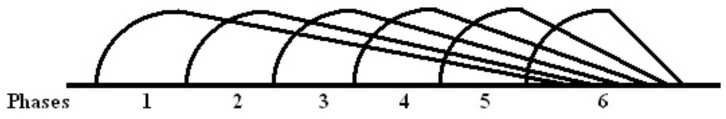


Figure 4.2: Agile Phases Approach [80]

Agile methodology allows use and application through the iterative development framework Scurm [80]. The Scurm game development method is an Agile process that manages game development using iterative and incremental approaches, which are the life of the game project. It works in game development methodology by breaking down the process of creating a game into a series of tasks, named "Sprint". To facilitate work with Sprint, the game developer breaks the games down into groups of related tasks or features that must be recorded in the Product Backlog. As mentioned in Figure 4.1, every two to four weeks at the end of Sprint phase, the whole team meet to discuss the current state of games, and to improve a version of the game with stakeholders, and to select new tasks from the backlog.

According to [80], the Agile game development with Scurm could be labeled an iterative and adaptive model.

4.2.2 Industry Game Design and Development Methodology

Most game development companies or organizations will need to take decisions regarding methodology, applications and hardware and software technology selections. To remain in business an organization must invest in appropriate new technologies. In all of these cases the organization is attempting to understand and balance a number of competing concerns regarding new technology [36]. These concerns include:

- The initial cost of acquiring the new technology;
- The long term effect on quality, time to market, and overall cost of the organization's products and services when using the new technology;
- The impact of introducing the new technology into the organization in terms of training and other necessary support services;
- The relationship of this new technology to the organization's overall future technology plans;
- The attitude and actions of direct competitor organizations with respect to the new technology.

Major software companies are interested in video game development due to the high industry revenues and growing capabilities [13]. Games in this industry area have three main stages; the preproduction stage, when the games are designed and deliverable named GDD. Then, in the production stage, the GDD is used for software design, development and validation. In the post-production stage, games are distributed and monitored following delivery, with the purpose of taking any required corrective action, as well as an analysis of the company's expectations regarding the sales and performance of the game product [117]. By contrast, in some other industries, the game development process has only two stages: preproduction and production. However, these two stages involve several phases, such as an analysis phase and a testing phase. Figure 4.3 is adopted from [94] and illustrates the details of the phases of each stage.

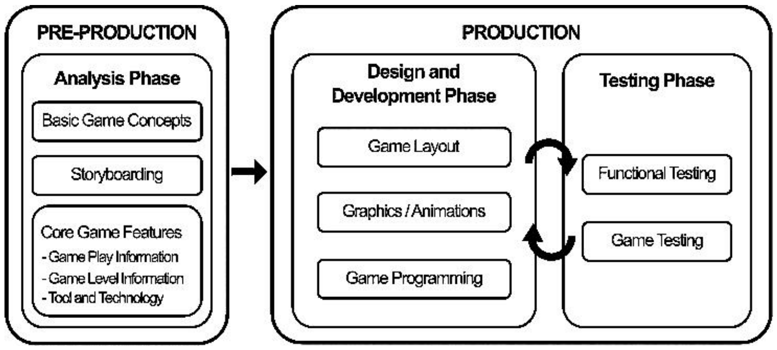


Figure 4.3: Games Development Process [94].

4.2.3 Agent Oriented Software Engineering (AOSE) Methodology

Within recent years, and with the increase in the complexity of projects associated with software engineering, many AOSE methodologies have been proposed for development[18]. At present, the intelligent agent-based system applies to many domains, including robotics, networks, security, traffic control, games and commerce. As part of our work, we focused on the AOSE methodology to apply this to the game domain.

During the last decade, many methodologies for developing agent-based systems have been invented to solve different types of problems. In general, a methodology must contain the guidelines to cover the entire life cycle, and should provide the following: a full life cycle process; a comprehensive set of concepts and models; a full set of techniques (rules, guidelines, protocol); a fully delineated set of deliverables; a modeling language; a set of metrics; quality assurance; coding (and other) standards; reuse advice; and guidelines for project management. The relationships between these components are shown in Figure 4.4 [130].

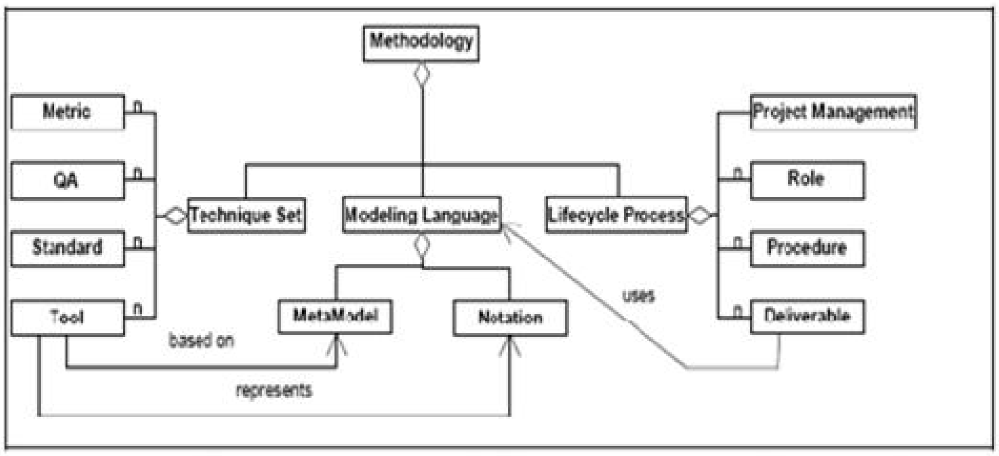


Figure 4.4: Methodology Components and Relationships Between Them [130]

4.3 AOSE Methodology Types

AOSE has been categorised into different major classes based on influencing approaches:

- Methodologies based on knowledge engineering approaches such as MAS-CommonKADS and Tropos, which focus on agent knowledge level concepts.
- Methodologies based on object oriented approaches, such as MESSAGE/UML and Prometheus, which focus on object oriented concepts.
- Methodologies based on existing software engineering methodologies, focusing on state machines and components such as MaSE [132].

While some methodologies cover from the start to the end of the life cycle such as Tropos and MaSE, the six methodologies chosen were: Gaia, MaSE, MESSAGE, MAS-CommonKADS Prometheus, and Tropos. In the next section, we briefly describe each of these.

4.3.1 MaSE Methodology

MaSE stands for Multi Agent Systems Engineering. It is a complete life cycle methodology that helps the developer work with a MAS from start to the end. This means that it describes the process guiding the system developer from the initial system specification to system implementation. The first phase begins with requirement collections and goal capture. The second phase deals with analysis, which includes three steps: capturing goals, applying use case scenarios and refining roles. The design phase has four steps: creating agent classes; constructing conversations; and assembling agent classes and system design. At each step, related models are created. Models in one step produce outputs that then become inputs for the next step, supporting the traceability of models across all components. Furthermore, there is the possibility of free access between components in each phase, as shown in Figure 4.5 [49].

MaSE combines several pre-existing models into a single structured methodology. The majority of the models used within the methodology have therefore already been justified and validated within the realm of agents and MAS. A sequence of guided transformations connects the elements with a strong foundation together into a clear high specification image, detailing how a designer should go about creating MAS [144].

The advantage of MaSE as an agent tool supports analysis and design in each of the seven MaSE steps. In addition, the MaSE methodology operates independently of any particular agent architecture, programming language or communication framework [48]. The goal of MaSE is to guide the system developer from the initial system specification stage to system implementation [51] [49].

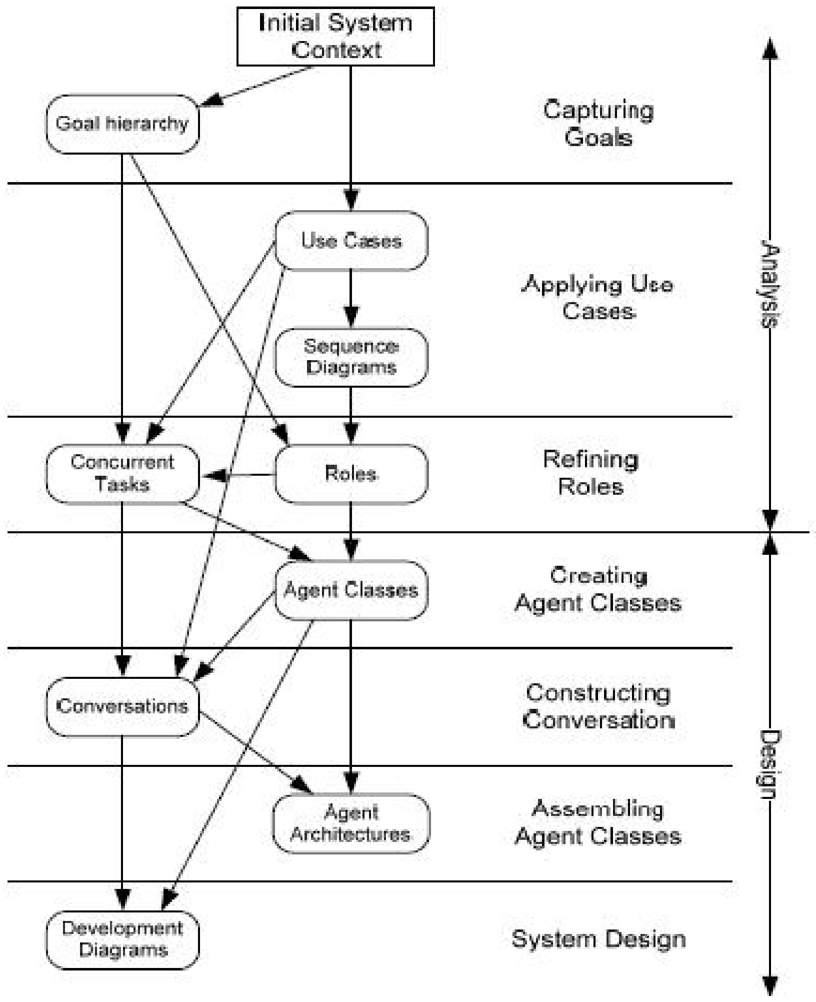


Figure 4.5: MaSE Life cycle [49]

4.3.2 MESSAGE Methodology

MESSAGE stands for Methodology for Engineering Systems of Software Agents, which is not a complete life cycle methodology. The life cycle contains only analysis and design activities and ignores the implementation and testing. The aim of MESSAGE is to extend existing methodologies to allow them to support AOSE, and in particular effects the needs of the telecommunications industry.

In the "Data level", the MESSAGE uses standard UML, while at the "knowledge level" meta-model concepts are used, adding new meta-concepts such agent, goal and task. The meta-model also provides a declarative interpretation of some UML concepts, which are then used to describe the relevant behavior. The Rational Unified Process (RUP) is used as a coarse-grained iterative process model. The lifecycle model of RUP for software development, which provides a generic software engineering project life cycle framework is adapted to MESSAGE [46].

MESSAGE is developed based on a view-oriented approach. The analysis and design phase are shown in Figure 4.6, and contain five different viewpoints: Organization view (OV), Goal/Task view (GTV), Agent/Role view (AV), Interaction view (IV) and Domain view (DV). The different views provide a comprehensive representation, while the design phase is not well-documented and provides flexibility in the sense that the designers are allowed to choose between different design approaches [46]. The MAS organization and architecture driven design approach are considered the target systems as organizations of several agents.

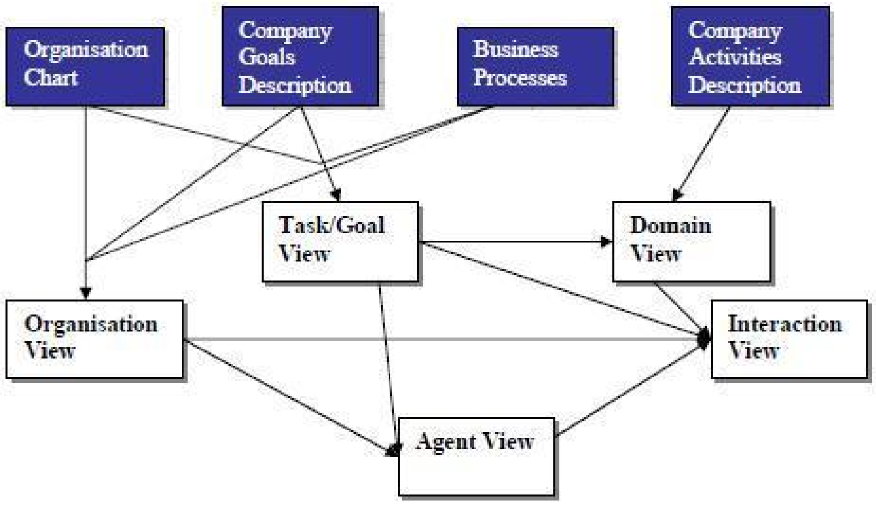


Figure 4.6: The MESSAGE Work Flow [46]

4.3.3 Gaia Methodology

Gaia is an agent oriented analysis and design methodology. It is applicable to a wide range of MAS and is comprehensive. It handles both macro-level (societal) and the micro-level (agent) aspects of systems [146]. The Gaia methodology includes two analysis models and three design models, as shown in Figure 4.7. The analysis phase is based on well-defined concepts, which represent a subset of conceptual needs in agent analysis. The main goal of the design phase is to produce a sequence of steps, an identifiable set of models, and indicate the interrelationships between the models. Gaia guides designers to build agent-based systems, as a process of organizational design.

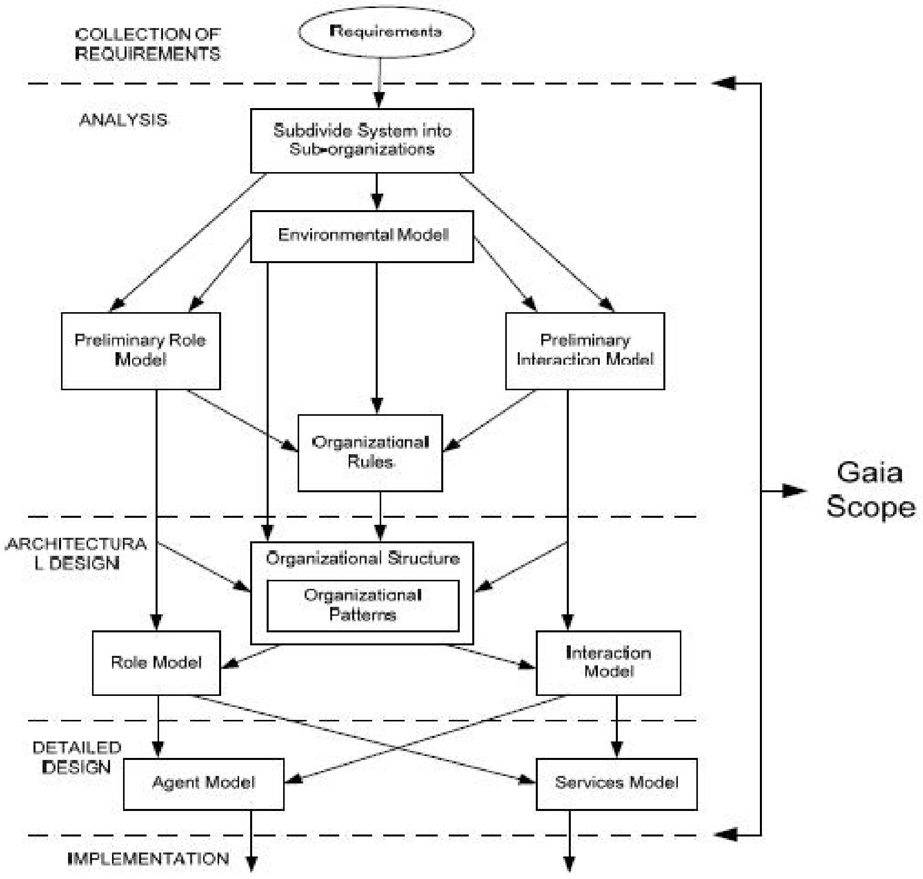


Figure 4.7: Gaia Life cycle [146]

4.3.4 Prometheus Methodology

Prometheus has a complete life cycle methodology for developing intelligent agents, which have evolved out of industrial and pedagogical experiences [109]. The goal when developing Prometheus is pursued by industry practitioners and undergraduate students who do not have a background in agents and who can use to develop intelligent agent systems. Much of the existing methodology does not support an intelligent agent. In the Prometheus methodology, it supports the development of BDI, which renders it useful in many areas. One of the advantages of this methodology is the number of places in which automated tools can be used for consistency checking across the various artifacts of the design process [109]. The Prometheus methodology consists of three phases: the system specification phase, architectural design phase and the detailed design phase as shown in Figure 4.8.

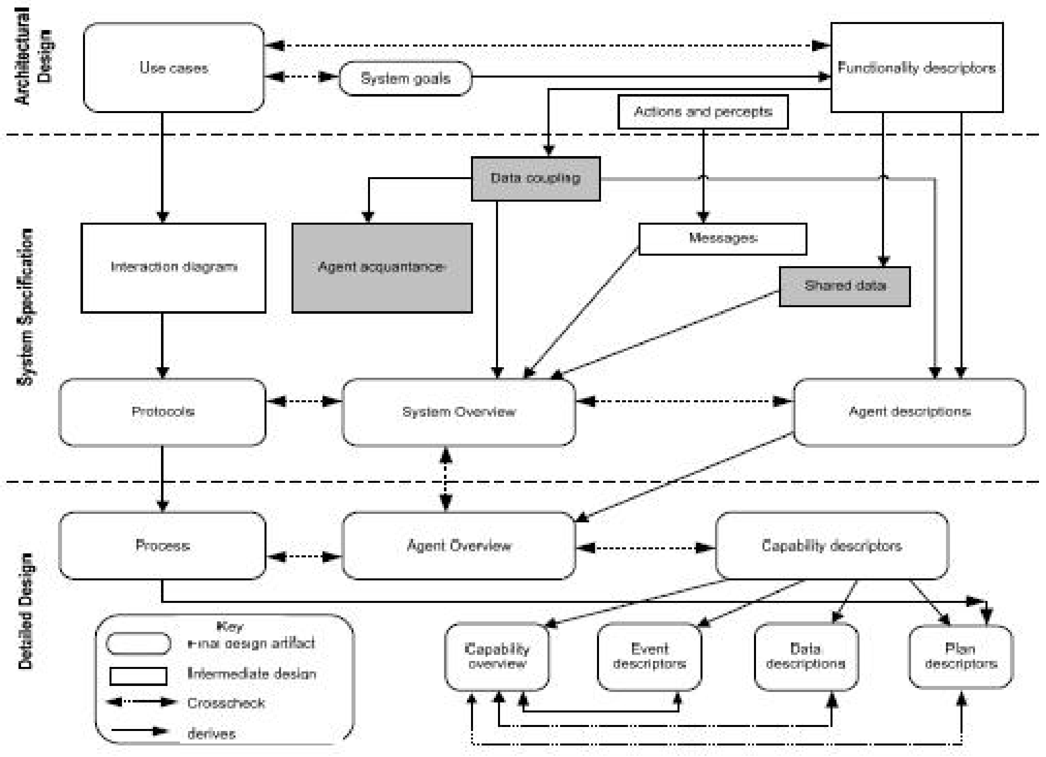


Figure 4.8: Prometheus Life cycle [109]

4.3.5 MAS-CommonKADS Methodology

MAS-CommonKADS extends the knowledge engineering methodology, Common KADS, with techniques to object-oriented and protocol engineering methodologies. MAS-Common KADS covers the software development life cycle of MAS through reusable development models. The software process model combines a risk-driven approach with a component-based approach [71]. MAS-Common KADS has six models for analysis, and three for design, as shown in Figure 4.9. These models are comprehensive and the method lacks a unifying semantic framework and notation.

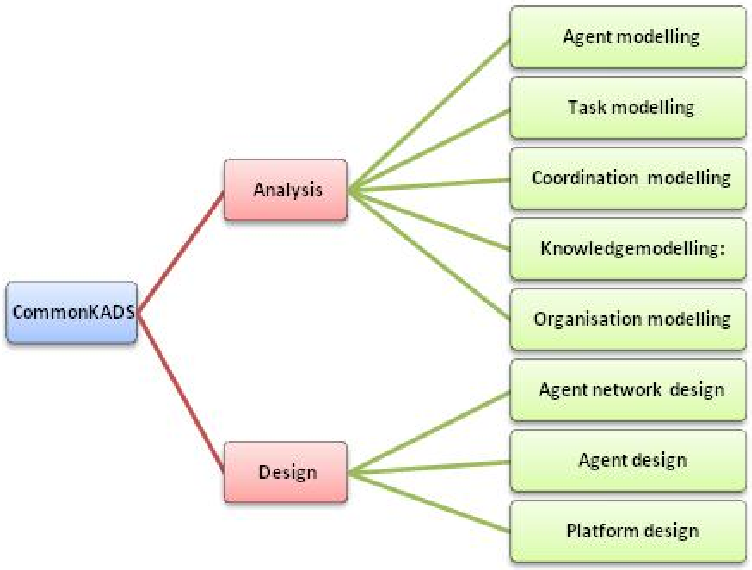


Figure 4.9: MAS-CommonKADS Life cycle

4.3.6 O-MaSE Methodology

O-MaSE denotes an organisation-based multi agent software engineering methodology framework, which integrates a set of concrete technologies aimed at facilitating industrial acceptance through situational method engineering. Specifically, O-MaSE is a customisable agent oriented methodology based on consistent, well-defined concepts supported by plug-ins in an industrial strength development environment. The goal of the O-MaSE methodology framework is to allow method engineers to build custom agent oriented methods using a set of method fragments. To achieve this, O-MaSE is defined in terms of a meta-model. The O-MaSE meta-model defines a set of analysis, design, and implementation concepts, and portrays a set of constraints between them.

O-MaSE depicts the roots of the original MaSE methodology. The O-MaSE methodology framework was based on two meta-models: SPEM 2.0 and the O-MaSE meta-model. The O-MaSE meta-model defines the main concepts and relationships used to define MAS. The O-MaSE meta-model is based on an organisational approach [52]. As shown in Figure 4.10, O-MaSE is composed of five entities:

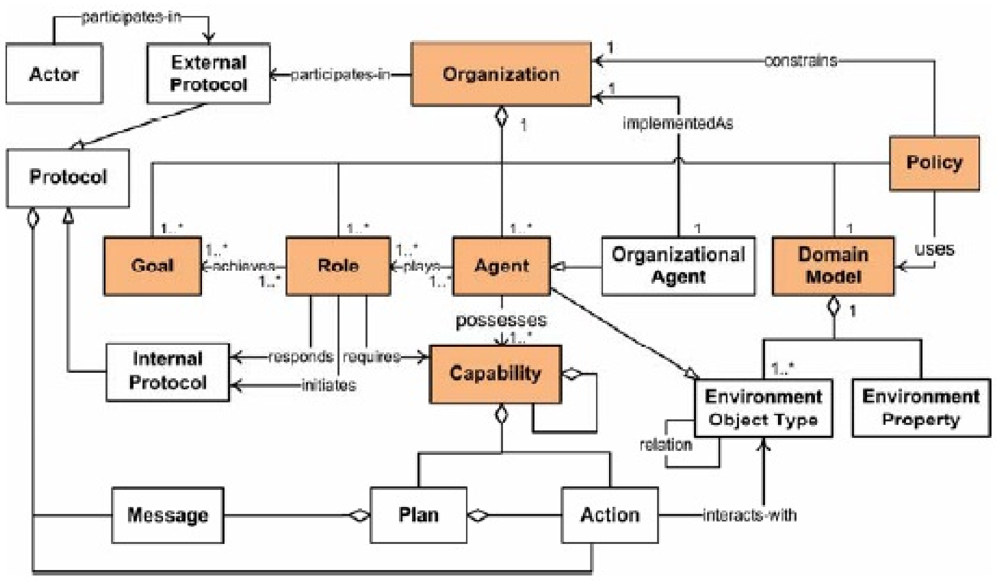


Figure 4.10: O-MaSE Meta-Module [52]

In the agent environment, the goal was defined as a mental attitude representing preferred progressions of particular MAS. O-MaSE uses goals to define the objectives of the organisation. A role defines a position within an organization whose behavior is expected to achieve a particular goal or set of goals. An agent that possesses all the capabilities required to play a role may be assigned that role in the organisation. Capabilities can be defined as

- A set of sub-capabilities

- A set of actions that may interact with the environment
- A plan to use actions in a specific way

Organisational agents (OAs) are organizations that act as agents in a higher-level organization and thereby capture the notion of an organisational hierarchy. The domain model is used to capture the key elements of the environment in which agents operate and finally protocols define interactions between roles or between the organization and external actors.

4.3.7 Tropos Methodology

Tropos is a requirement-driven agent oriented software development methodology, created based on agent-based system development. It focused on early requirement analysis whereby the basic identifier of stakeholders was defined and their intentions identified and analysed. During the late requirement, the system actor was introduced. The system's global architecture was defined in architectural design. In the detailed design, the behavior of each component was defined in detailed. This analysis process described the reasons for developing the software for capture [102]. The software development process of Tropos comprises five phases: early requirements, late requirements, architectural design, detailed design and final implementation, as shown in Figure 4.11 [35]. From the previous figure, the goal of Tropos is divided into two parts: soft goal and hard goal, where the hard goal is related to functional requirements and the soft goal relates to non-functional requirements. Two models represent them at this point in the methodology. First, the actor diagram that describes the stockholders and their relationship in the domain; Second, the goal diagram that presents an analysis of goals and plans with regard to a specific actor with the responsibility of achieving them. One of the goals of the methodology is to reduce the mismatch between the concepts used to describe the operational information systems environment, and the concepts used to describe the architecture and high level design of the systems[89].

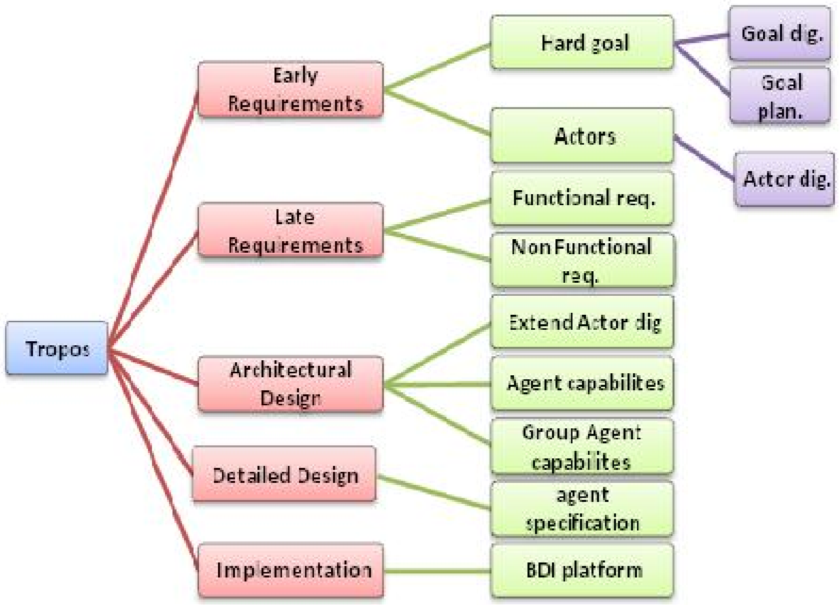


Figure 4.11: Tropos Life cycle

4.4 AOSE Methodologies Evaluation

There are several methodologies in AOSE, each with its own life cycle. However, some of these were precise only for analysis and design, such as Gaia, while others covered the entire life cycle such as Tropos, MaSE and Prometheus. In the evaluation framework literature for AOSE, the majority of researchers [33][46] used qualitative evaluation tools, designed according to the author's viewpoint, or with regards to the questionnaire. Furthermore, the authors of the methodologies presented some of these evaluations, which renders them highly subjective.

The majority of the existing work found in the literature is based on qualitative evaluations using different techniques, such as features based analysis, surveys, case studies and field experiments. Chia et al.[42] consider qualitative evaluation in reference to four main criteria: Concepts and properties; modeling techniques; and process and pragmatics. His evaluation technique was based on a features based analysis. Each criterion contains different attributes, whereby Yes or No have been used to represent the criteria in each of the methodologies.

The criteria with the same definitions have also been used in [19]. Another interesting evaluation of these methodologies is provided by Tran et al. [137] who attempted to cover ten of the most important methodologies by employing different criteria. Thus instead of using Yes or No, as in the case of the previous author, 'H' was used for high, 'L' for low and 'M' for medium. While Saremi et al [119] presented a quantitative evaluation, which evaluated the complexity of diagrams for MESSAGE and Prometheus, and used case study methods to measure the magnitude and diversity and to determine the final complexity. Increases in magnitude and diversity are heightening the complexity of the model.
Additional effort has been performed by Basseda et.al [33] who measured the complexity of the methodology using different attributes. Lower software complexity provides advantages, such as lower development and maintenance time and costs, less functional errors and increased re-usability. Therefore, this approach was commonly used in software metrics research to predict software quality based on complexity metrics. By studying evaluation frameworks proposed to date, it seems that:

- There is no appropriate framework for evaluating methodologies.
- There are frameworks mostly based on feature-based analysis or simple case study methods.
- There are only a small number of frameworks that have been evaluated with quantitative approaches.

Therefore, presenting a proper framework evaluated using both quantitative and qualitative methods and using feature based analysis, surveys and case study methods is anticipated to be highly beneficial.

4.4.1 Game Development Methodology Framework (GDMF)

The framework presented below is based on two criteria: first, some adaptations of existing qualitative frameworks; second, new quantitative evaluation frameworks. Furthermore, the results derived from these particular frameworks are guiding us in the selection of the most suitable methodology for the game development domain. We performed comparisons of well known methodologies, selected based on the following criteria:

a) They were described in detail and had a complete life cycle. The majority of the existing AOSE focuses only on analysis and design; whilst MaSE, Prometheus and Tropos have a full life cycle [132].

b) They are influenced by the software engineering root.

c) They are perceived as significant by the agent community [47].

According to these criteria, we decided to take into account only the MaSE and Tropos. A quantitative evaluation is an important component of the evaluation process, because it is based on fixed results for comparative purposes. The majority of the previous research focused on the qualitative approach, enabling a comparison to be made between methodologies [25]. Some difficulties were encountered during the literature review regarding the quantitative evaluation:

a) The majority of the evaluations such as [119] [38] compared two methodologies, by using a specific case study to determine results.

b) There were no standard attributes used for evaluation.

To evaluate our framework, the following criteria were used:

1. Select the common criteria. Regarding the qualitative evaluation, we decided to first adopt a methodology from [42], because his precise evaluation covered qualitative criteria and used feature based analysis methods and secondly from [137], since his evaluation was based on survey methods. Regarding the quantitative evaluation criteria, the criteria were divided into three sub-criteria: by transferring the existing qualitative attribute values of the quantitative numbers, dealing with meta-model metrics and use case evaluation methods [25].

2. Transfer the qualitative attributes into quantitative values, and then convert these values using a proposed common scale for each metric, as shown in the following:

- Yes-No to 0-1 and the common scale 0-10
- None-Low- Medium- High To 0-1-2-3 and the common scale 0-3-7-10

4.4.2 Converting the Qualitative Results to Quantitative Evaluations

In this section, we adopted [42], [137] evaluation, converting their criteria to numerical results to facilitate comparison between MaSE and Tropos. Initially, we represented the summarized lists provided from [42]. In each of the main four criteria areas: Concept and properties, modeling techniques and process and pragmatics as shown in table 4.1. From the criteria defined by Tran et al. [137], we selected the criteria for the steps measuring usability. However around twenty steps were used to compare Tropos and MaSE, as shown in table 4.2.

Table 4.1: Comparison of MaSE and Tropos [42].

Criteria	Sub-Criteria	Tropos	MaSE	Comment
Concept and Properties	Autonomy	10	10	
	Mental Mechanism	10*	10	achieve goals and soft goals
	Reactivity	10	10	
	Pro-activeness.	10	10	
	Adaptation	10	0*	Needs to add iteration
	Concurrency	10	10	

Table 4.1: Comparison of MaSE and Tropos [42]

Criteria	Sub-Criteria	Tropos	MaSE	Comment
	Agent interaction	10	0*	Needs to add agent interaction
	Collaboration	10	10	
	Teamwork	10	10	
	Agent oriented	10	10	
Modeling techniques	Expressiveness	10	10	
	Modularity	10	0	
	Refinement	10	10	
	Traceability	10	10	
	Accessibility	10	10	
Process	Life cycle Coverage	10	10	
	Architecture Design	10	10	
	Implementation Tools	10	10	
	Deployment.	0	10	
	Management	0	0*	Needs to add management
	Requirement capture	10	10	
Pragmatics	Tools Available	0	10	
	Modeling Suitability	0	10	
	Domain Applicability	10	10	
	Scalability	10	10	

Table 4.2: Comparison Regarding Steps and Usability of Tropos and MaSE Adopted From [137]

Steps	Tropos	MaSE
Identify system goal	10	10
Identify system tasks/behaviors	10	10
Specify use case scenario	10	0
Identify roles	10	0
Identify agent classes	10	10
Model domain conceptualisation	0	0
Specify acquaintance between agent classes	10	7
Define interaction protocol	10	10
Define content of exchange message	7	7
Specify agent architecture	7	0
Define agent mental	0	7
Define agent behavior interface	0	0
Specify system architecture	0	0
Specify organizational structure	0	10
Model MAS environment	0	10
Specify agent environment interaction mechanism	0	0
Specify agent inheritance	0	0
Instantiate agent classes	10	0
Specify instance agent deployment	10	0

4.4.3 Evaluation of the Methodologies by Meta-Model Metrics

This section of our framework addresses meta-model diagrams, using meta-modeling techniques to define the abstract syntax of MAS modeling languages (MLs), which is a common practice today. We also use a set of metrics to measure the meta-models. These metrics help to quantify two features of the language: specificity and availability.

1. The availability metric as shown in equation 4.1 measures how appropriate an ML is for modeling a particular problem domain. A higher value is better in resolving the domain problem.

   The ncmm indicates the number of necessary meta-model elements; nc indicates the number of necessary elements; mc indicates the number of missing concepts.

$$Availability = nccm \div (nccm + mc) \tag{4.1}$$

2. The specificity metric, as shown in equation 4.2, measures the percentage of the modeling concepts used for modeling a particular problem domain. If the value of this metric is low, this means many ML concepts are not being used for modeling the problem domain [95].

$$Specificity = nccm \div cmm \tag{4.2}$$

The term cmm represents the number of all the concepts in the meta-model. Table 4.3 was created by [95] and uses four use cases, with six methodologies included Tropos and MaSE to find the availability and specificity for the purposes of comparison.

Table 4.3: Availability and Specificity to Compare Between Tropos and MaSE.

	Tropos		MaSE	
Case study	Availability	Specificity	Availability	Specificity
Cinema	75	75	77.7	60.9
Request example	91.7	45.8	100.0	52.2
Delphi	72.2	66.782.4	82.4	60.9
Crisis Management	63.6	58.3	77.7	60.9
Total Average	75.8	61.5	84.5	58.7

Therefore, we could conclude that MaSE was considered better than Tropos with regard to the final results. Actually, MaSE obtained a higher percentage of availability than Tropos; but Tropos had a higher percentage in terms of specificity. The availability is a more important parameter than specificity in game development because game development implements modules based on priority.

4.4.4 Evaluation of the Methodologies using Diagrams

The majority of the AOSE methodologies was delivered in phase diagrams or tables, in particular during the analysis and design phase, i.e. UML diagrams and agent diagrams. An important point to consider in this evaluation is that we worked within an abstract level of methodology, which resulted in difficulties finding artifacts to be qualified. The important points raised in this evaluation are based on a case study, and the results depend on the case study itself [32]. Therefore, in some case studies, MaSE may obtain better results than those associated with Tropos and vice versa.

According to results found in Basseda et al [32], which compared diagrams, and the complexity of the dependency of modules for three AOSE. MaSE shows greater complexity in terms of dependency modules than MESSAGE and Prometheus. This means that MaSE has more dependencies between its models. Thus, using MaSE requires more time and effort.

4.4.5 Critical Analysis

The proposed framework was applied to MaSE and Tropos, as two case studies to demonstrate how the framework could be used to evaluate the possible methodologies. There are differences in the results of the comparison. Both [131] and [47] evaluated MaSE as better than Tropos. Moreover, when we made the suggested change to MaSE, this increased its efficacy MaSE. In [39], the author compared two popular reference works [47]and [131] using a profile analysis, which is a multivariate statistical method. The majority of the results were found to be similar to those in our own evaluation and results.

We calculated the means for Tropos and MaSE from table 4.1, and found the means of Tropos =0.84 and obtained the same results as with MaSE, in this case we also applied the suggested enhancement to MaSE, as shown in the comment column of Table 4.1, the means of MaSE after enhancement are =0.96. The means for MaSE is=1.7894 as shown in Table 4.2, which was greater than Tropos=1.2631.

As we observed from Table 4.3, MaSE obtained a higher percentage of availability than Tropos; but Tropos had a higher percentage in terms of specificity. Availability is a more important parameter than specificity in game development because game development implements modules based on priority. When we calculated the total percentage for both specificity and availability, MaSE obtained 71.6 and Tropos obtained 68.65. Thus, MaSE was considered better than Tropos with regard to the final results of that metric.

4.5 Summary

Game development has evolved to deliver large projects employing hundreds of people over development times measured in years. Unlike most other software application domains, game development presents unique challenges which stem from the multiple disciplines that contribute to gaming. The relationship between games and AOSE is clear, given that software agents and intelligent agents are used as virtual players or actors in computer games and simulations. The development process is very close to the process of game development[66]. From the previous measurements, we found that MaSE used frameworks that obtained higher scores than Tropos in the majority of measurements, using a feature based analysis, survey and case study evaluation methods. Furthermore, we observed the following weaknesses with most AOSE methodologies:

- All AOSE methodologies lacked industrial strength tools and standards. In addition, they did not seem to cover team work, since project management and time planning were not considered in the AOSE methodologies found in the literature.
- There was a weakness at the implementation phase.
- There were no standard metrics that could be used in each phase to evaluate the phase output, the complete system or the most effective methodology for an application.

The next chapter will present the proposed game development methodology based on a critical review of previously conducted work.

Chapter 5

Agent Oriented Agile Based Game Development Methodology (AOAB)

5.1 Chapter Overview

The suggested methodology AOAB combines Agile methodology to meet the dynamic requirements of the customer with MaSE. MaSE is a rapidly developing area of research, designed to support the development of complex and distributed systems in open and dynamic environments, with the use of intelligent component. Game development methodology works better using an iterative methodology, because this permits fast preparation of features, enabling discovery and adding fun to games. Through the process of research, a number of development models have been used. The AOAB methodology focuses on two archetypical development models, the predictive and the adaptive models.

It is important to have a formal understanding of the game development process, and of how we can create a formal game development methodology that will be generic to many game genres. Ideally, the type of hybrid development methodology approach, which we already defined in AOAB is recommended for use by independent game developers. It possesses a mix of characteristics that can be positioned somewhere between those of a predictive or an adaptive approach to a generic methodology that is useful for game projects. In this chapter, we have begun to analyse and detect problems with current game development methodologies. The archetypical development methodologies are presented in detail. The remainder of this chapter focuses on presenting AOAB with detailed descriptions of each AOAB phase. At the end of this chapter, we present a critical analysis of AOAB methodology.

5.2 Problems With Current Game Development Methodologies

A vast majority of the problems facing the entertainment industry and its development are deeply rooted in the production methodology employed. Teams of approximately 100 people are still using methodologies that were developed at a time when ten people were considered an excessively large team[97]. Specific features of game development have been found to prevent the success of great games. The major problems that arise are in the areas of project management. The use of methodology focuses on game development, and takes into account the project management concept to help avoid management problems.

Games and software engineering comprise important aspects that can be mutually beneficial to understanding, as in the main they share the same methodology and problems. A game contains a confluence of interesting properties, such as emergence, real time interaction and challenging components, which create a new field of study [91].

Software engineering has greatly assisted the entertainment industry in its resolution of problems. The unique aspects of gaming, which are not available in traditional software development include the requirement for the game to be ‘fun‘, which cannot be assessed by any metric, and is purely subjective.

After a survey of the current game development methodology problems, we will highlight the main problems found in the literature:

Schedule Problems: According to Flynt [62], a key reason for a project being delivered behind schedule is that no target has been established. Likewise, problems may occur when an estimated deadline does not include the time needed for communication, and lacks documentation or emergent requirements that may alter the system architecture and thereby cause serious problems. Furthermore, delay can be caused by a multidisciplinary approach; as it is essential to include input from different teams, as delays may occur. A task involves a series of risks that imply underestimates, resulting in cumulative schedule delays. Flynt et al [62] report that developers recurrently fail in their estimates, due to a lack of historical data to assist them in determining a realistic time frame to complete a task [111].

Crunch Time Problems: In the entertainment industry crunch time is a term that is typically used for the period of work when overload may occur; this is usually, in the final weeks before the validation phase or the deadline for project delivery. During this period, the developer may work in excess of 12 hours a day and take between 6 and 7 days to complete unfinished tasks. In the entertainment industry, crunch time is a fact of life [111].

Scope and Feature Creep Problems: Feature creep is a term used in the entertainment industry when a new functionality is added during the development phase. This increases the project scope and alters the schedule timing [111]. Any new functionality should be evaluated carefully. Any unmanaged feature creep can lead to increased error, possible defects and increased likelihood of failure. However, some feature creep is unavoidable, since it adds fun to the game [78]. Flynt et al [62], report the biggest reason for game project imperfection is the failure to accurately establish project scope.

Risk management helps a project manager to understand the changes to a plan and identify the potential costs in terms of time and money. The project scope will never be a true reflection of required effort, due to the iterative and exploratory nature of game development; however, it can be an effective guide when predicting success, such as when discussing milestones, time lines, and budgets[78].

Technology Problems: All games are technology dependent. Technological components generate risks to game projects that can require greater effort and a higher investment of time. According to Gershenfeld et al [64], the risks associated with technology risks are generally higher when a team works on a new platform, because of two factors; the first being that the developer has not worked with the technology before. The second being that related hardware frequently contains problems.

Documentation Problems: Lack of documentation is a common source of additional difficulties. Documentation can be valuable in reducing feature creep. Having a finite amount of documentation is useful when game developers work on difficult projects, as this helps them to obtain a good estimate of project scope and schedule. Usually, Game Design Document(GDD) generates considerable uncertainty around a game's goal and solution requirements [87].

Collaboration and Team Management Problems: One of the main problems when creating games is communication between teams. The teams in games include people with distinct profiles, such as developers, plastic artists, musicians, scriptwriters and designers. Different teams need to collaborate and explain their work and instructions to others.

Training Problems: One of the biggest problems with Agile game development especially, and in game development methodology generally, is new employee training.

Linear Process Problems: Game development is not a linear process [62]. Iteration informs the lifeblood of game development. Game developers use the Waterfall methodology with enhancements, adding iteration to the methodology.

Petrillo et al[111] present in Figure 5.1 the histogram of occurrence of problems in a decreasing sequence. From the previous study, we can observe that the most traditional software problems are the same as the game development problems.

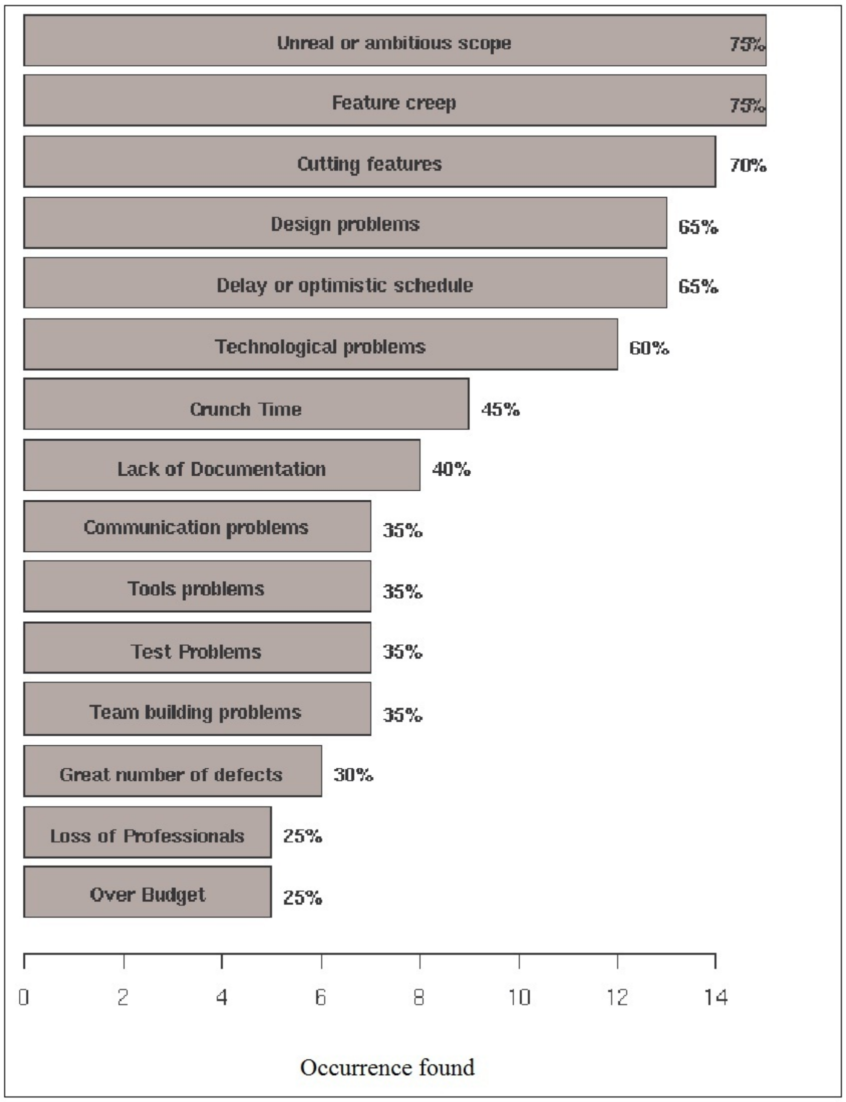


Figure 5.1: Occurrence of Problems in Current Game Development Methodology [111].

5.3 Archetypical Development Methodologies

There are many methodologies offered within traditional systems and software development scenarios. As mentioned above, some of the better known methodologies include Waterfall, Incremental and Spiral. Each of these are structured as either linear or iterative, and are sometimes hybrid. Most linear methodologies are classified as predictive, even where they contain iteration; they typically follow sequential phases, such as Waterfall methodology. Prototyping involves dividing the system into small segments to involve the user in the process.

Spiral methodology combines linear and iterative frameworks. Spiral development breaks down projects down into a number of cycles, all of which follow a set of increasingly larger steps. The majority of methodologies employed by game developers are predictive, employing comprehensive planning as separate tasks prior to actual development. Some of these methodologies are also described as adaptive, because multiple iterations and prototypes are used to shape the game [70]. The question here is

What criteria should be applied to choose between predictive and adaptive models in game development methodology?

In AOAB, we will not select between the types, because we need to draw on both concepts for game creation. For this reason AOAB is a hybrid methodology, combining the best features from predictive and adaptive models. The second important question is

How can components from a variety of game design and development models be integrated into standard development guidelines?

In reality, it is a challenging task to create generic game development methodology to cover the most important game requirements. Furthermore, there should be standard game development guidelines. In this chapter, we have created an AOAB methodology that provides all the answers to these questions and requirements.

5.3.1 Predictive Model

In general predictive models are preferable when we have clear goals and the customer's requirements are clear and complete and the specific structure of the game must be withheld at all costs, allowing for a definite vision of the final product to be established long before it takes a playable form as shown in Figure 5.2. We will take the AOSE methodology as an example of predictive methodology

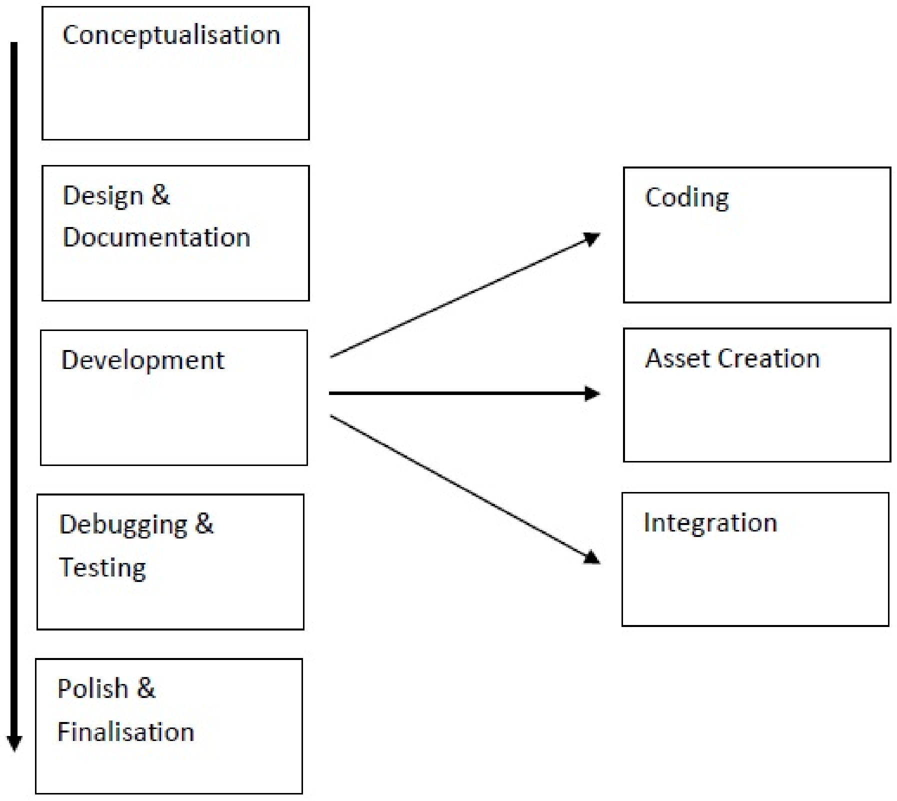


Figure 5.2: Predictive Development Methodology[70].

5.3.2 Adaptive Model

Adaptive models are encouraging the change in customer requirements and customers are being allowed to add new goals or new requirements even in the late stages of games development, and thus these must not affect the game plan. Furthermore the customer is usually invited to give a direct response on the development process, regarding the lessons learned, as shown in Figure 5.3. We will take Agile methodology as an example of an adaptive methodology.

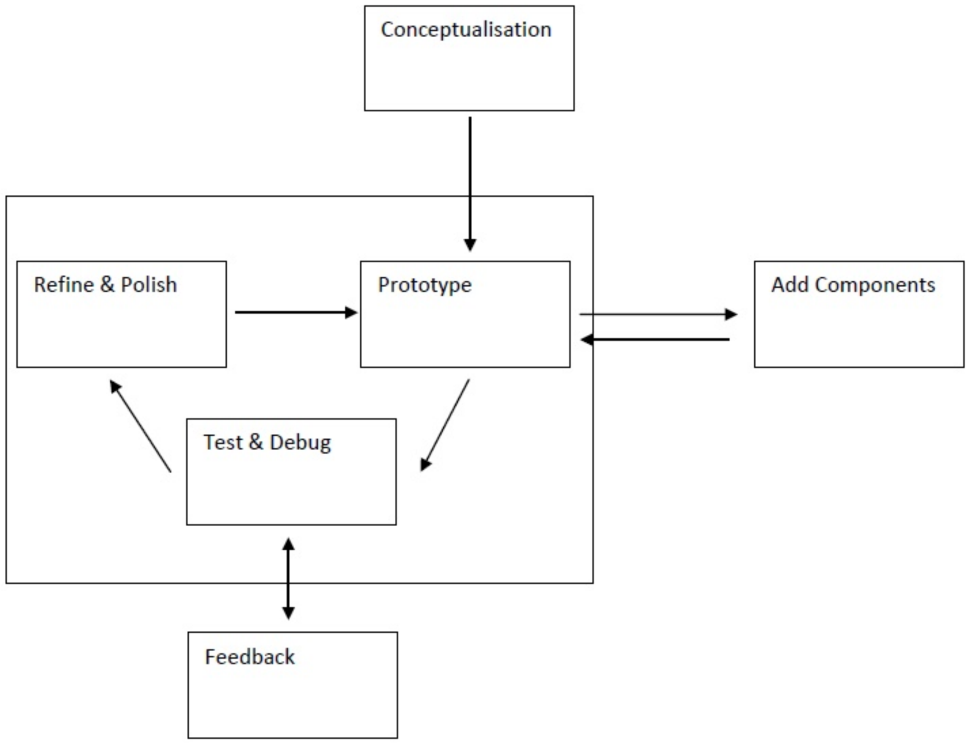


Figure 5.3: Adaptive Development Methodology[70].

5.3.3 Predictive vs. Adaptive Methodologies

This section presents the difference between predictive and adaptive development methodologies to reach a common understanding of predictive and adaptive characteristics. The adaptive models are based on iteration with follows a bottom up approach. Adaptive focuses on are developing more than simply documentation. The predictive models are based on a linear process which follow a top bottom approach. Predictive focus on documentation and expected changes in requirements only effect the initial stages of development. Unlike the adaptive model which anticipates a change in the customer's requirements in any stage of development.

The other difference between adaptive and predictive relates to testing and debugging. The adaptive model deals with the integration of testing and debugging in the development process and iteratively in prototype building. However, the predictive testing and debugging is done mostly at the end of the development process and tests all the system components. Ideally, the type of hybrid development methodology recommended for use by independent game developers would be likely to possess a mix of characteristics [70]. Table 5.1 from [70] presents a comparison of adaptive and predictive characteristics.

Table 5.1: Comparison of adaptive - predictive characteristics [70].

predictive	Adaptive
Linear	Iterative
Pre-planned	Planned
Focused on Documentation	Minimal Documentation
A broad definition of the game early in development	Game features are developed, then later synthesised
Restriction of changes to the initial concept	Refinement and adaption of the initial concept
Testing and Debugging discrete from content development	Integration of testing and debugging throughout the development process
Sequential creation of final game components from scratch	Game development are prototypes; these are then built upon and improved iteratively

The final difference between adaptive and predictive is the planning concept. The important question here is *"How much planning is enough"* . Adler et. He asserts that having a planned approach and project schedule before development begins can minimize the high risks of software development. Both of adaptive and predictive approaches are using planning concepts but in different ways. Predictive approach is based on pre-panned approach. This is because;

predictive is usually used when we have well understood of customer requirements.
While in adaptive approach, we also deals with planned approach over the software development process [17]. Keith argues that as much new knowledge is generated during the design process, constant iterations are needed in order to incorporate this knowledge into the final product. He stresses that working prototypes, playable demonstrations and constant adaption are integral to producing a fun, coherent game, rather than comprehensive design documents and pre-planning [80]. Aside from being iterative by being structured around a series of task iterations and revisions, Keith's Agile game development model with Scrum could be labeled as adaptive. In this sense, although a final game is planned from the beginning of the development cycle, the developers constantly adapt the game's features (and thus final form) throughout the development cycle, in response to feedback from prototyping and demonstrations. This differs from other more archetypically predictive development models [80].
Overall, predictive models would be preferable when there is a pre-defined customer expectation or specific structure the game must withhold at all costs, allowing for a definite vision of the final product to be established long before it takes a playable form. Adaptive models encourage change and thus will not usually allow for all aspects of a game to be planned in unison, seeking to allow a game's final project to be a direct response to its development process and the lessons learnt within [70].
However, one can see that such models are rarely purely adaptive or predictive, often incorporating elements primarily from one archetype but possessing a few from the other. And these concepts are appearing clearly in AOAB methodology because it is fully hybrid methodology. It is recommended that independent teams should at least partially conceptualize and plan the core features of their game before development begins. The game's scope should realistically be considered in terms of the team's abilities and first and foremost focus on delivering a finished, playable game.

5.4 AOAB Goal and Objective

We have listed some goal oriented criteria to enhance the development process and to minimise errors. As a result, consistency through the development will increase. These criteria are:

- AOAB methodology should be easy to use and easy to learn. The use of clear and uncomplicated notation and clear phases of the methodology will make AOAB easy to use by both experts and new game developers such as Indie developers.
- AOAB must have complete steps and clear documentation to guide the developer through initial requirements, moving through design and evaluation to reach the final game release.
- AOAB is utilising the best practices from AOSE, Agile methodology, Software engineering methodology and other strategies and techniques.
- The characteristics of game development are unique and require more research. Therefore, we are leaving some provision for future work.
- GDD in AOAB should be easy to use and filled by Indie and expert game developers.

5.5 Agent Oriented Agile Based Game Development Methodology (AOAB)

In AOAB, we use systematic and formal development processes in game design. It is important for game design novices to structure work around a preconceived pattern, taking time to plan and consider the design and overall structure of the game before undertaking the task of actually coding and building it.

Formal development models can facilitate learning by providing an evenly rounded development process, such as by allowing time for analysis, planning, development (coding) and evaluation. They are also used to guide uninitiated designers so that they do not jump straight to coding and omit the overarching lessons that the other stages can teach about game design. [70].

Agile methodology is usually used to address dynamic changes and requirement specifications by the customer, and to deal with customer involvement in the development phases. For flexibility in adding new requirements prior to game release, which does not add extreme cost to the project, Agile game development methodology will be adapted to AOAB as an adaptive model.

AOSE provides such intelligence through agents. The agent may perform tasks individually. In complex and distributed system, Agents can be used to monitor the interactions among components and interact in the same manner as humans. MaSE will be adapted to our suggested game development methodology as a predictive model, which depends on linear processes and has good GDDs [100]. AOAB methodology consists of many development phases. Figure 5.5 illustrates the suggested game development methodology, showing how it functions as a hybrid methodology using both adaptive and predictive models.

Management is important in the game industry. Poor management can negatively affect the best of teams. While complexity in the game world, and the number of teams increases, good communication in a company is essential for success. Agile methodology achieves quality and productivity through iteration and communication with customers, as part of the methodology deals with management. Agile methodology typically depends on daily Scurm meeting to assure good communication, but on many occasions there is no need to discuss this daily because it is only a waste of time for teams. In AOAB, we have suggested that a meeting is not necessary to save time on a daily basis. This should only be done when important issues arise from multidisciplinary teams such as artists, musicians, developers and clients. Furthermore, a group may have a sub-group, such as the AI team or a textures team. An example would be a unit composed of two programmers, a texture artist and an animator. Combining groups does seem to enhance communication across disciplines. Bringing diverse groups together can enhance understanding and communication between teams [78]. AOAB methodology attempts to provide an adaptive and predictive development methodology. Sometimes a combination of more than one model may be the most suitable option[99]. The MaSE life cycle from Figure 4.5 are used in the Sprint phase which is the main part of Agile methodology from Figure 4.1. Both Agile and MaSE methodology has been explained in details in Chapter four. The Figure 5.4 illustrate how we will hybrid the two methodologies.

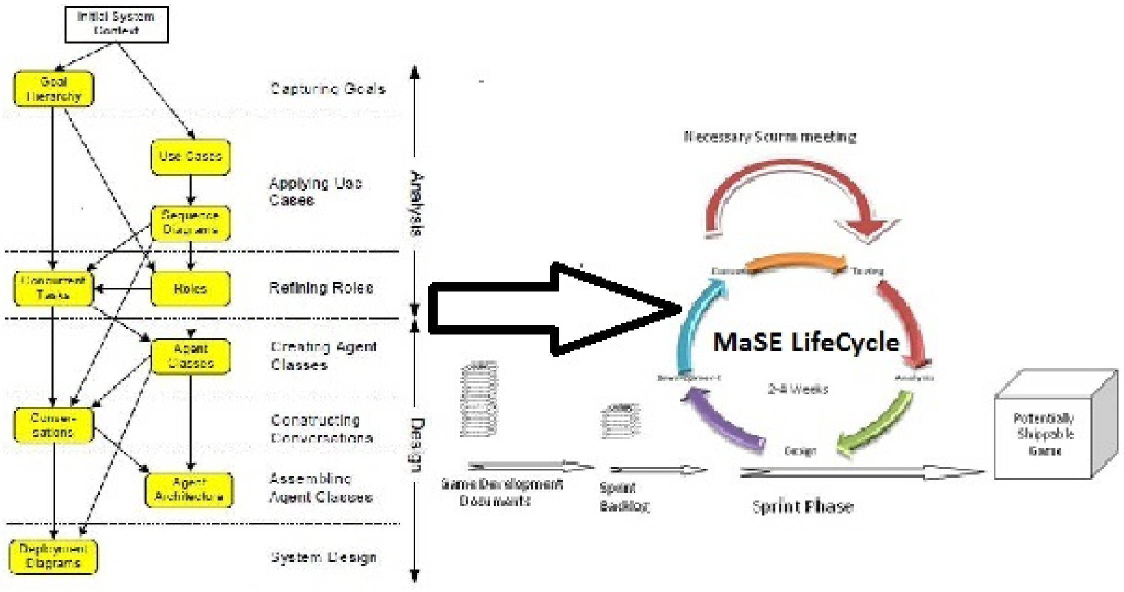


Figure 5.4: AOAB Designed From Hybrid Agile and MaSE Methodology.

While Figure 5.5 illustrate the final AOAB Methodology. Each iteration includes analysis, design, implementation, testing and evaluation of MaSE. The reason to adapt MaSE in the core of Agile, is that in complex systems and distributed systems such as games, it is difficult to trace a single point of control, since objects are distributed [99].

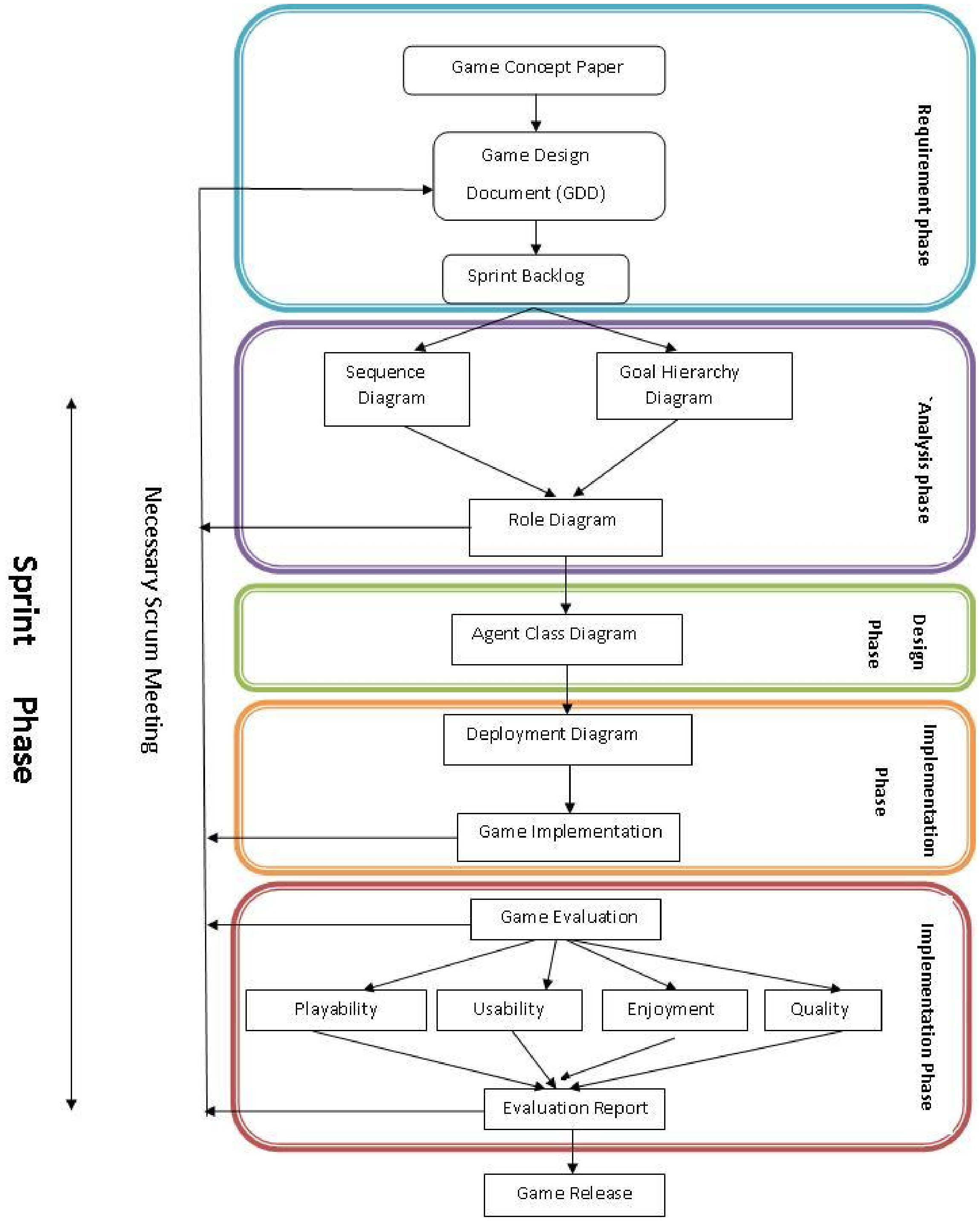


Figure 5.5: Agent Oriented Agile Based Development Methodology

5.6 AOAB Features

AOAB has many features, the main focus of which is to cover industry requirements. The two key features are a clearly defined duration, and the team collaboration policy. The following two sections will explain these features in detail.

5.6.1 Game Development Duration

AOAB is an iterative methodology that focuses on delivering features. Planning, documentation and development activities are repeated in every iteration, and the number of iterations is based on the scope of the project. The aim of iteration is to deliver many working versions of a game after a short iteration period. The feedback obtained from customers in relation to each iteration will help to solve many problems at an early stage and, as a result, the game development time will be reduced. The AOAB has the ability to begin working on new features prior to completing a current feature. In which case, the duration of the game development period will be reduced, as no time is wasted on waiting. In the planning stage, the customer and developer will typically cooperate to select new features, which are then added to the Sprint Backlog to discuss whether they are high priority. The main effort in using AOSE alone is mainly expended in the preparation of the documentation, as shown in Figure 5.6 [99]. AOAB reduces documentation by creating GDDs and dividing these in Sprint.

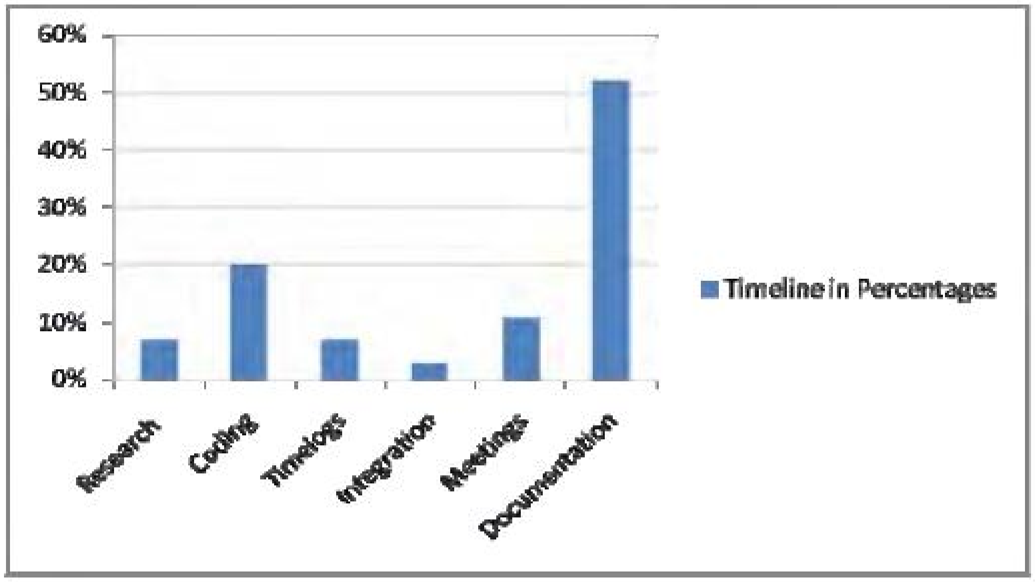


Figure 5.6: Duration for AOSE [99].

Game documentation is important, and is required for the analysis and design phases, as those details are needed to maintain games or to create new versions of a game. The overheads incurred through communicating with large teams, and the cost of longer development efforts, have led to a demand for certainty from the stakeholders. Large detailed design documents attempt to create that certainty [80].

5.6.2 Team Collaboration

A game development team is different to a traditional software development team. The primary distinction between the two is that for games, the development groups consist of people with different fields of expertise. In the initial stage of game development, a scriptwriter is required and presents documents, usually known as a “Concept Paper”. [27]. The concept paper will be converted to a GDD by a game designer, and the GDD will guide the game development process.

Game companies usually employ a group of programmers including, for example, an engine programmer, graphics programmer, AI programmer, sound programmer and tool programmer. Furthermore, the game company might collaborate with a musician and sound effects technician. In addition, the artwork for the game will be created by 2D or 3D graphic artists. Prior to game's release, the game company will hire testers to play and evaluate the games, find bugs and to suggest solutions to problems or recommend changes to game play [115] [127]. There are advantages to multidisciplinary teams, for example, they create an environment that encourages creativity; however, there is also a need to ensure cooperation between teams.

The main objective of game creation is to ensure players are satisfied and enjoy the experience of playing the game. Interaction between the customers and developers leads to greater customer satisfaction, and an understanding of dynamic changes in customer requirements. When using Agile, the communication between the developer and customer is typically on a daily basis, whereas in AOAB, communication between the customer and developer takes place on a 'needs only' basis, as it is not easy to facilitate a daily meeting with all teams.

5.7 AOAB Methodology Life Cycle Description

5.7.1 Requirement Phase

The first step of the requirement phase is preparing the GDD based on concept paper, which is divided to Sprint Backlog. From the Sprint Backlog, it is possible to estimate the number of iterations that the Sprint phase will cover.

Game Design Document (GDD)

In this phase, prior to creating the GDD, the developer must determine the objective of the game, identify the 'fun factor' and decide on the types of people who will play game. The creation of a GDD is an important step to accomplish in the first phase of developing the game, as it is responsible for determining the scope of the project, and thus also affects the development and testing phases of the game. A poor GDD may result in feature creep and consequently result in delays and possibly missed milestones. [65].

There is no standard way to build a GDD, although it requires a comprehensive description of all aspects of the game. It must also describe the objects and characters of the game, their effects, how they interact, and their role and behavior in the game. The GDD will go through many changes and will also have additional requirements. The risks of any changes should also be evaluated, as and whether deadlines can still be met. Later in the requirement phase, GDD will be translated into a Sprint Backlog, for small games, as it may be optional when translating any requirements directly as a Product Backlog. This can save time, but may also increase the risk of feature creep, or lead to a game that is not adequately entertaining [23] [65].

If the GDD is designed carefully, the project manager will be able to plan the iteration of Sprint Backlog so that the game is playable at the conclusion of each iteration. This has several benefits; first, testers can check the game for errors in a playable state that mimics what the end user would encounter. Beginning a playable game as early as possible helps the team to see the potential of the end product, and is therefore beneficial in game publication prior to final release. Included in the Appendix A.1 is a GDD template for GDD that is easy to use and follow. The main aim when dealing with a GDD template is to provide a standard, generic GDD that can be used for different game genres, as is the goal of AOAB. The Appendix A.2 also includes another version of the GDD, following participant workshop feedback. Participants suggested creating a GDD that is easy to fill in, particularly for Indies developer. This part of the work will be presented in detail in Chapter Eight.

Sprint Backlog

During the Sprint planning stage, all teams are tasked with dividing the large GDD, if it is too large, into a Product Backlog. However, if the GDD is not too large, this task is not required. The team invested a significant amount of time in analysing the requirements and consequences of supporting network game play. At the end of the Sprint, they expressed a desire to replace this feature, as implementing it in a sufficiently balanced quality would threaten the overall quality of the game [121].

At the end of each Sprint, the team is able to update the Sprint Backlog and drive the necessary activities for the next Sprint. For the final presentation, the team were expected to successfully deliver a fully playable, feature complete prototype. They could easily pinpoint problematic phases or crucial moments, and were eager to analyse their progress. In conclusion, they perceived Scrum as very supportive throughout all phases of the project, with a single exception [121]. At this stage, once the GDD is complete, the game idea, in other words what the game is really about, the project scope and what actually needs to be done should become clear to the developers and customers. The GDD should then be translated into the Sprint Backlog.

In each iteration of the game lifecycle, the most important backlog should be addressed first and divided into smaller tasks. AOAB uses the same techniques as Agile, as suggested by Keith [80] and shown in Figure 4.2. This means that at the end of a Sprint phase, some work can still be under development. A new iteration can begin using a new Sprint Backlog. The goal is to achieve a continuous flow of content creation.

Sprint Phase

A development team usually treats the Sprint phase as a mini waterfall, with small design documents written at the start and working code not coming together until completion. This is superior to the traditional Waterfall model, although coaching the team to keep them communicating with each other and to ensure that the construction is ongoing, rather than writing documents, will further improve performance. One of the main differences between the Scrum and Waterfall methods is the idea that the product is kept in a state of near-completion in every Sprint, and that the features added in every Sprint have a level of completeness that improves the value of the final game. The goal is to prove the value of the feature in every Sprint. Design, coding, debugging, testing and assets are all taken into consideration [81].

In AOAB the MaSE methodology is used to cover the sub-phases of a Sprint, such as analysis, design, implementation and evaluation. The purpose of working this way is to show customers the value of a feature every two to four weeks, to show how it improves Sprint by Sprint and, at the same time, acquire documentation that will be useful in the evaluation or creation of a new version of the game [22].

5.7.2 Analysis Phase

The Analysis phase includes three steps: capturing goals, applying use cases, and refining roles [48]. In AOAB, the first part of the analysis, capturing goals, is already defined and added to the GDD. Using the GDD, use cases can be identified, and the initial set of roles can be refined and extended.

Model Goal

The objective of the model goal is to transfer the GDD from a system requirement to a set of standard goals for the game. The goal model is used in many agent-oriented methodologies. There are two steps to capturing goals: identifying and structuring the goals. The analyst identifies goals by analysing whatever requirements are available [51]. In a goal hierarchy diagram, goals are organised by importance. The model goal is based on AND/OR decomposition, which converts the overall goal of the game into a set of sub-goals. The AND is defined by whether all sub-goals must be achieved in order to achieve the parent goal. While the OR is defined by whether the sub-goal represents an alternative way to achieve the parent goal. However, the purpose of the goal hierarchy diagram used in MaSE is to identify the main system level goals, not individual agent goals. The goal model creates a high level specification regarding what the system should do.

Applying Use Cases

The use cases step is important in translating goals into roles and associated tasks. Use cases are drawn from the system requirements and are narrative descriptions of a sequence of events that define the desired system behavior [51]. To help determine the actual communications required in MAS, the use cases are converted into sequence diagrams. MaSE sequence diagrams are similar to standard UML Sequence Diagrams, except they are used to depict sequences of events between roles in order to explain the communication that must take place. The roles identified in this step form the initial set of roles used to fully define the system roles in the next step, and the events identified are also used later to help define tasks and required conversations.

Role and Task Model

The model roles' task identifies all the roles in the organization as well as their interactions with each other and external actors. The result of the role models task is a role model. The goal of role modeling is to assign each sub-goal of the organization goal model to a specific role. Roles are identified from the use cases as well as the system goals. All system goals are accounted for by associating each goal with a specific role that is eventually performed by at least one agent in the final design. Each goal is usually mapped to a single role; however, there are many situations in which it is useful to combine multiple goals in a single role, for convenience or efficiency. Role definitions are captured in a standard Role Model [51]. Once roles have been identified, detailed tasks that define how a role accomplishes its goals are defined and assigned to specific roles. A set of concurrent tasks provides a high level description of what a role must entail to satisfy its goals; including how it should interact with other roles. This step is documented in an expanded role model.

5.7.3 Design Phase

In the design phase, analysis models are transformed into constructs that are useful for game creation. The design phase has three steps: creating agent classes, constructing conversations and assembling agent classes. In AOAB, the focus is mainly on creating an agent class diagram, which is the most important part of the design phase. Constructing conversations and assembling agent classes are nominated due to a need to focus on the diagrams that will be helpful either in later stages of game creation, or after the game has been released.

Model Agent Class

The model agent class identifies the types of agents that may participate in the organisation. Agent classes can be designated specific roles, or they may be defined in terms of capabilities, which implicitly define the types of roles to be played. An agent class is a template for a type of agent in the system; each agent class identifies the capabilities that it possesses, or the roles it can play, or both [50]. As with goals and roles, a one to one mapping between roles and agent classes can be defined; however, it is possible to combine multiple roles in a single agent class, or map a single role to multiple agent classes. Since agents inherit the communication paths between roles, any paths between two roles become conversations between their respective classes.

5.7.4 Implementation Phase

Code generation is achieved during the implementation phase. The game is implemented in an incremental way and is the result of a Sprint. A base can be implemented in one or more programming language. The purpose of the generate code task is to take all of the design models created during the development phase and convert them into a code that correctly implements the models. There are numerous approaches to code generation, depending on the chosen runtime platform and implementation language [50].

Deployment Diagram

The diagram relating to the implementation phase is a deployment diagram. Deployment diagrams define the configuration of the actual system to be implemented; furthermore, they define the overall system architecture to show the number, type and location of agents within a system.

5.7.5 Evaluation Phase

Game development companies have strong competition, and the gaming experience has become an important factor in differentiating similar kinds of game titles. If the gaming experience is not optimal, players can easily switch to another game. The gaming experience can be evaluated once a prototype is implemented and ready for beta testing. At this point, the correction of any problems will be too expensive, or the project schedule will not allow any delays, for marketing reasons. As a result, there is a need for an evaluation method that can identify these problems prior to the commencement of beta testing, thus providing adequate time for corrections to take place [86].

The evaluation process, known as formative evaluation, is conducted to detect problems [107]. These are usually identified at an early stage of the development processes of the game and can be used to improve and enhance the game at each iteration before it is ready for release.

In game development methodology, the evaluation phase is carried out by two types of evaluator, expert evaluators and real users. Expert evaluators has are individuals who have both knowledge and experience of conducting evaluations using various expert evaluation methods. Meanwhile, real users are a group of target users, for whom the game is being developed. These users will be the respondents in an evaluation using different user evaluation methods. Furthermore, the process will be repeated and the results of the current iteration compared with results of previous iterations, prior to game release. In order to address the evaluation problem, different protocol sets are proposed, based on different points of view.

The first step in the creation of an evaluation protocol s set requires an understanding of how the protocol s are developed and identifying the criteria previously used by authors.
In 1990, Nielsen and Mack offered a protocol evaluation, which was used to evaluate the user interface of software productivity [105]. These protocol s are useful, in the development phase, to obtain design guidelines. Several authors have subsequently noted that games require protocol s of their own [61] [85] [108]. Usability protocol s address issues concerning playability; playability, unlike usability, does not have a standard definition. For this reason, several authors provide a definition and protocol set in relation to playability [86][53][85] [103]. Federoff's thesis [61] presents a protocol s model that could be considered the first specific protocol s model, due its structure and design methods. Federoff presents 40 protocol sets, divided into the following sub-criteria: the game interface, game mechanics and gameplay.
Desurvire [53] presents Protocol for Evaluating Playability (HEP) based on Federoff's sub-criteria, using game play and game mechanics, and adding usability and game story as part of the 43 protocol sets. Likewise, Korhonen and Koivisto [85] present playability protocol that focus on mobility games, using Desurvire game play and usability and adding the sub-criteria mobility, which consists of 29 protocol. However, Schaffer [120] suggests protocol based on the researcher's expertise in HCI fields, so divided protocol into five categories: general, graphical user interface, game play, control mapping and level design, with 21 protocol. Finally, Pinelle [113] introduces ten usability protocol designed for multi player games.

Game Evaluation Protocol Framework (GEPF)

Special attention is required to specifying the methods that evaluate games. Game evaluation is still ongoing and the protocol sets are quite different, but do include some common issues [86]. Multiple protocol sets are available, and it is important to select carefully the variables to be measured, as well as the correct methods for collecting the data [30]. Some protocol are proposed that have not been validated, whilst others are targeted towards specific game genres, or do not cover all of the evaluation needs. This leads to an important question, *"What aspects of games can be evaluated?"*

It can be a challenging task to define the protocol that are able to capture the essential aspects of game evaluation. Therefore, it is important to select protocol sets of a high level suite to different game genres without losing power to guide evaluators during the evaluation phase. At the same time, all evaluation aspects should be covered in order to obtain a measurement that is suitable for enhancing the next iteration of game development methodology. Furthermore, the correct methods for collecting and comparing the results must also be used.

This work aims to achieve clarity, generality and usefulness of protocol sets in order to evaluate the majority of the game genre. The suggested framework for evaluating games has standard requirements for facilitating the work in game development companies. Rather than using different evaluation sets for each game type, the protocol sets used in this research cover all important aspects of high level game protocol. The evaluation task inspects the game for any problems arising at any time. The purpose of the protocol sets is to guide the evaluation phase and remind the evaluators to pay attention to certain important aspects, which will be explained in detail. The initial high level protocol framework for games that will be used in AOAB will be presented.

For the above reasons, the proposed evaluation sets contain 100 protocol organised into four categories; first, game playability (42 protocol), which deals with three main issues: game story, game play, and game mechanics, as shown in Table B.1. Second, game usability (26 protocol), which deals with two major issues: game interface and game control, as shown in Table B.2. Third, game quality (12 protocol), made up of three key concepts: game functionality, game efficiency and game adaptability, as shown in Table B.3. Finally, game enjoyment (20 protocol), which covers enjoyment elements not contained in previous protocol sets, as shown in Table B.4. Meanwhile, the user evaluation methods will cover similar protocol sets to those used by expert evaluators, except game mechanics, which deals with techniques [21]. User evaluation is typically carried out using a questionnaire, interview or scenario as shown in Figure 5.7.

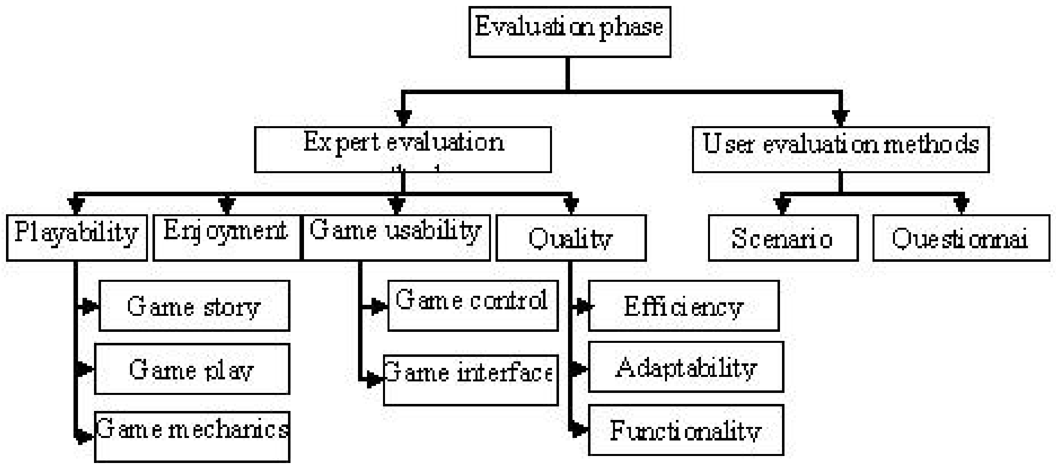


Figure 5.7: Game Evaluation Criteria

The Game Evaluation Process

An initial step in the game evaluation process is the explanation of the overall process. Figure 5.8 illustrates the overall process, which must be followed in each iteration of the game development methodology. The results of each iteration are compared with the results from the previous iteration to ensure that all encountered problems are solved. In each of the evaluated games, a significant amount of criteria are measured, and correlations made between them. Those criteria are broken down into a number of more measurable factors, defined on the basis of current gaming literature [60]. The data from the overall correlation highlights users' views, expert evaluations and comments to develop a prototype, and identifies specific elements that need to be enhanced. In the tables, two columns are added that are deemed useful for obtaining accrued results for statistical correlation. In column number 5, a score for each protocol set is given, from 1 to 5, 1 being worst, 5 being best, in order to compare every iteration with the previous iteration. Furthermore, in column 6, priorities from 1 to 3 are stated, 1 being the highest priority, 3 being the lowest priority, which is useful for ensuring the importunity of protocol , depending on the game genre and the evaluators' points of view.

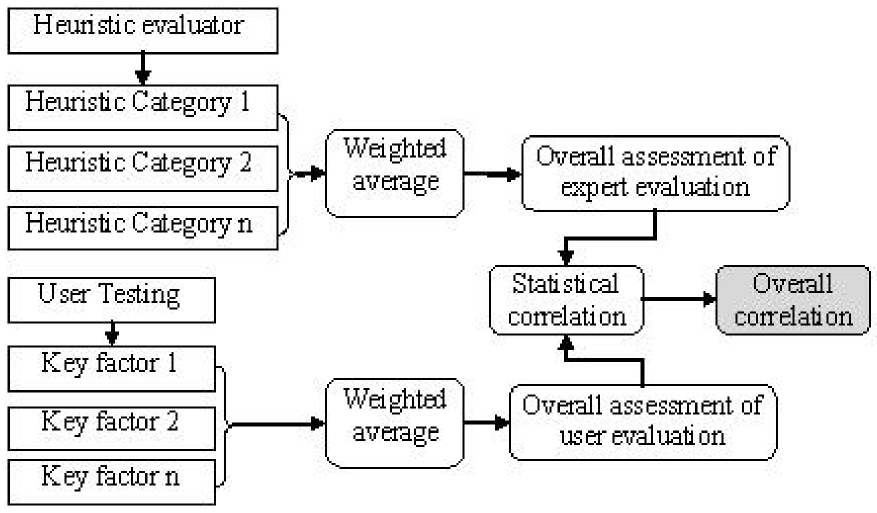


Figure 5.8: Game Evaluation Process

Expert Evaluation Methods

The game developers' perception of expert evaluation methods is interesting, as it can measure different aspects that help in the evaluation and testing of games. The use of expert evaluation criteria allows many game issues to be identified and evaluated in-depth [134]. The protocol sets used in this research are general purpose, which means they are applicable to evaluating the majority of game genres, covering playability, quality, mobility, enjoyment and usability. These criteria are selected because they are common to all games; they cover most aspects needed in the evaluation of games and are easy to use.

The appendix B includes all required tables for each evaluation criteria and sub criteria, with full descriptions.

Playability

What is playability and what are the criteria that affect playability? This section will answer the above questions. Games have good playability if they are easy to use, are challenging and fun. According to the protocol set, playability is a combination of game play, game story and game mechanics, as shown in Table B.1. Game usability is related to playability, and is an important concept covering game control and the game interface. For the above reason, usability is treated as a main criteria, and not as a part of playability.

- Game play: This is the set of problems and challenges a user must face to win a game [53]. When evaluating game play, the evaluators must have some game design experience, they should understand the goal and know the target players [85].
- Game story: This usually includes all plot and character development [53].
- Game mechanics: These are tested by Quality Assurance (QA) personnel in game companies to ensure that no broken games get shipped [61]. This involves the programming which provides the structure through which units interact with the environment [53].

Game Usability

Game usability covers aspects of game control and game interface, through which the player interacts with the game, as shown in Table B.2. Good usability of a game ensures that the player will have a fun and enjoyable session [85]. The majority of the existing game usability protocol sets are based on the ten usability protocol detailed by Molich [105], which are used to perform protocol evaluations of software engineering and websites. The game interface should allow the player to control the game fluently, and should display all necessary information regarding the game status and any possible actions [85].

Game Quality

One of important measurement throughout the game development life cycle is quality. The quality evaluation process begins with a careful planning phase, which includes the purpose of the evaluation, the timing of the evaluation and the people involved in conducting the evaluation process [58]. The quality protocol was used to assess not only the final versions of the games, but also quality throughout the game development life cycle, which enables developers to prevent the majority of game failures.

In this research, the Quality Evaluation Framework (QEF) form is adopted [58]. The QEF is divided into three criteria, and each criterion aggregates a set of factors. A factor is a component that represents the system performance from a particular point of view. The dimensions of the criterion of this research in the quality space are: efficiency, adaptability and functionality, as shown in Table B.3.

- Efficiency: This measures the system's ability to present different views of its content with minimum effort.
- Adaptability: This measures the effect of the extended scenario and system contents and presents different instructional design theories and different learning environment on a common platform
- Functionality: This reflects the characteristics of the games related to its operational aspects.

Enjoyment

Research in psychology and neuroscience most often uses the term "pleasure" to describe agreeable reactions to experiences in general. In computer game, they used the terms *fun factor, entertainment, enjoyment and engagement* to measure the player satisfactions. More specific notion is that since video games are designed with the primary purpose of entertainment, and since they can demonstrably motivate users to engage with them with unparalleled intensity and duration, game elements should be able to make more enjoyable and engaging as well [54].

Player enjoyment is an important goal for all games; if players do not enjoy the game then they will not play again [134]. A piece of research conducted by Sweetser focuses on game enjoyment [134] in relation to game flow. Game flow is a model for evaluating player enjoyment of games, and consists of eight elements, which are as follows: concentration, challenge, player skills, control, clear goal, feedback, immersion and social interaction [21]. In Table B.4, duplicate protocol that are included earlier in the table are identified, such as control, which falls under usability. Whereas, when clear goals are set, the challenge is covered by playability. Finally, feedback is also covered in usability.

5.7.6 Testing Evaluation Protocol sets

The game industry is broad and has continued to grow dramatically over time. The Telecommunication Regulatory Authority has organised a competition between undergraduate university students in the Sultanate of Oman, a country in the Middle East. The evaluation protocol of this research use a weighted-average of scores and priority of values obtained from protocol sets to help identify the best game in the competition.

Competition Overview

The competition is structured to cover different areas, such as audio, posters, games and short films. Appendix C contains the competition poster detailing the competition area and prizes. The participating students are required to submit creative work that must contain the competition logo, the objective of the game section of the competition being to design a game that reflects the theme of "my address is Omani". The competition poster has been sent to all Oman universities and colleges, and the competition itself dictates a period of four months during which to submit the work. The participants must be university students, who can submit their work individually or as a group. For the game competition section, the

Telecommunication Regulatory Authority asked for permission to use the evaluation method detailed above, and for the researcher to head up the game community judges, made up of five members. Three of these members have an academic background and two an industry background. The prizes are subject to a final evaluation score; first prize is approximately 3,000 US Dollars, second prize is approximately 2,500 US Dollars, third prize is around 2,000 US Dollars and there is an encroaching prize of approximately 1000 US Dollars.

Results of the Industry Competition

The judges received a great deal of usable gaming software through being involved in the competition, some of which was not nominated due to not embodying the competition concept. The protocol sets detailed previously were used to evaluate the games submitted, and each game received a score out of 100 based on the evaluation criteria. With regard to the games having been created by undergraduate students, it was noted that as such they could be classified as having a 'simple' game design and are in that sense unlike commercial games. For this reason, it was decided that the judging committee would reduce the number of evaluation criteria required in order to fit more appropriately the competition concept and the quality of the students' work.

5.8 Critical Analysis of AOAB Methodology

For AOAB, it is suggested that daily meetings are not necessary, to save time. Meetings should only occur on a daily basis when important issues arise from within the multidisciplinary teams, such as artists, musicians, developers and clients. These groups may contain a sub-group, such as an AI team or a textures team, as AOAB creates functional combinations of specialties. Regarding the problem of project scope, AOAB is used for prototype techniques and the GDD to cover the important aspect of games. Iteration in AOAB enables the designer to evolve features and reduce the amount of feature creep in the game.

Regarding project management and team organisation, AOAB offers rigorous processes underlying feedback on a project and communication between teams. In relation to project scope and feature creep, many situations in the entertainment industry reflect multiple features discovered during game development, which are relevant to game success.

AOAB is not a linear, but an iterative process. Thus, if an interesting feature is discovered, it must be analysed in terms of its risk and, if viable, it should be added to the project schedule [111]. It is noted that the cost of changes in traditional software increased in the late of the project times to solve any problems and a late change to the system will incur extra time and cost. Whereas, in the case of Agile, these will also increase, but typically at the end of the project, as the nature of Agile is such that it can accept additional requirements and updates at later stages of the project [135]. Keith [80] suggests that at the end of the Sprint, some work may still be under development. The goal is to achieve a continuous flow of content creation, as shown in Figure 4.2, a core concept of AOAB.

When AOAB uses Agile concepts, it improves on the quality and efficiency of large, complex game projects. Furthermore, this strengthens the communication between the developer and the end user. The data from the evaluation protocol and the comments on the developed prototype identify elements that need to be enhanced in the next methodology iteration. In order to obtain quantitative results, an extra two columns were added, one scoring the protocol and others to set priorities for the protocol based on the game genre and the evaluators' points of view.

As the protocol for this research use a critical review of games to identify problems and to develop a set of design requirements for the games, it is argued that this general methodology is a new approach that can be used by researchers and designers to understand design issues in most game genres. The research aims to clarify the different aspects of protocol sets and their usefulness in the game evaluation phase. Most of the protocol sets can be used in the early development phase; in which case, solving problems will not incur additional costs or time, and will enhance the games.

5.9 Summary

The AOAB proposed in this research solves most of the previous problems in game development by considering suitability for researchers and professionals in the industry. Adaptive models encourage change and thus do not usually allow for all aspects of a game to be planned in unison, which allows a game's final project to be a direct response to its development process and the lessons learned in that process [70].

The use of Agile concepts by AOAB leads to the improvement of the quality and efficiency of large, complex game projects. Furthermore, it strengthens the communication between the developer and the end user. Furthermore, Using MaSE in AOAB ensures the consistency of games and facilitates complete and easy documentation that will be understandable for different teams, useful for when a new generation of a game is required. The iterative nature of AOAB is important for game development as although perfect game scope will never be achieved, the main goal is to develop a solid scope that will help to guide the project to its conclusion and achieve its goals.

Several studies have suggested that one of the benefits of using protocol evaluation is that it helps designers to identify important classes of problems that are not always found through user testing [114][74]. In this chapter, the life cycle of AOAB is discussed, with detail descriptions. The next chapter of the research will focus on selecting a suitable evaluation method to evaluate AOAB methodology.

Chapter 6

Evaluation Methods Selection For AOAB

6.1 Chapter Overview

Evaluations are always difficult. However, one of the most difficult items to evaluate is a methodology [84]. The goal of the evaluation of the AOAB methodology is first, to develop a methodology by drawing on different evaluation methods, and second, to validate the methodology through feedback from professional individuals. The AOAB Evaluation Framework (AEF) will provide a comprehensive evaluation tool for many criteria. The hierarchical evaluation criterion enhances the usability of the framework and provides enough detail to evaluate different game development methodologies.

A critical analysis is presented alongside a clear description and justification of the best evaluation method for evaluating AOAB methodology. This will demonstrate that the procedures for collecting data were carefully and systematically planned, allowing the reader to assess the quality of the data collection procedure. Furthermore, it documents the research methods by providing a framework for the evaluation, which could easily be used in another methodology evaluation. The AEF will focus on three main evaluation methods; first, a survey will facilitate comparisons between different game development methodologies, and a workshop will be conducted. These workshops will help the research to receive feedback from expert individuals who will complete a questionnaire. Second, is an academic experiment. Third, an industry case study of a game development company will be presented. At the end of this chapter, the AOAB evaluation framework procedure and evaluation time scales are presented.

6.2 Purpose of Evaluation

Before beginning an evaluation, certain decisions must be made: *"what evaluation method should be used?" And "what is the purpose of AOAB evaluation?"* Among the evaluation methods community [31] [73] [82], there is agreement that the first and most important step of any evaluation process is to identify its purpose, as no rational comparison is possible without defining the purpose of the exercise. Depending on the purpose, the method of carrying out an evaluation and the results may vary significantly. As the development of game methodologies is still in the early stages, the aims and purpose of evaluating AOAB are:

- Better understand the nature of AOAB methodologies, including their philosophies, objectives, features and so on.
- Identify their strengths and weaknesses as well as the commonalities and differences in order to perform classifications and to improve future game development methodologies.
- Conduct an evaluation with the aim of adopting a new methodology for the existing development process
- The selected methodology must be best suited to the game development needs and must require no significant changes to the current practice process.

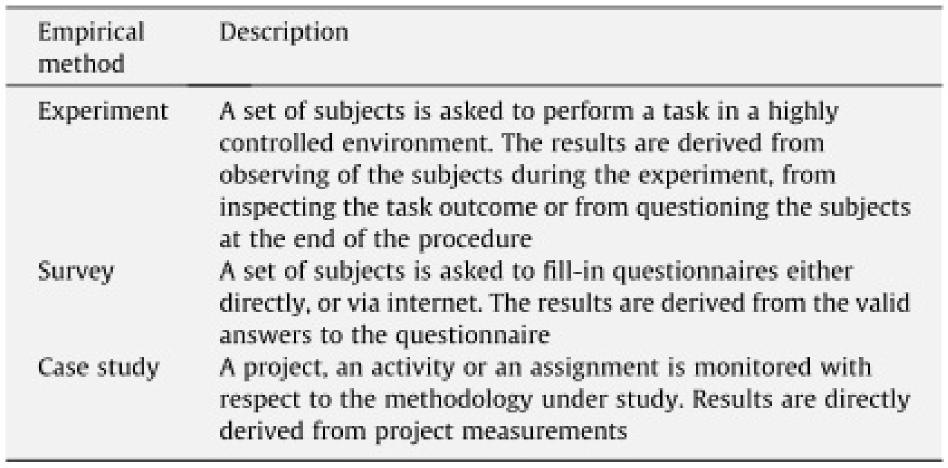
One important fact that must be understood is that different methodologies are appropriate for different situations; thus, a methodology should be selected by considering various different issues. These influencing factors might include the context of the problem being addressed, the domain, the organization and its culture. However, it is expected that evaluation will also assist in practical matters such as identifying the domains of applicability for each evaluated methodology [46].

6.3 Evaluation Context

The evaluation framework is context-dependent, which means that AOAB is not expected to be the best in all circumstances. It is possible that one of the game developers might identify AOAB as superior while another developer comes to a different conclusion. Hence, differences in the results of the evaluation may be due to the properties of the game developer that performs the evaluation, and not the methodologies themselves. For this reason our evaluation framework will be as general as possible, and will try to cover the most important criterion to help obtain results. Furthermore, AEF deals with game developers and designers with different backgrounds, such as academic experts, industry experts and Indies people.

6.4 AOAB Evaluation Framework Process

The AOAB Evaluation Framework (AEF) is a multi-step method to define each part of the framework using sets of criteria. Each criterion set defines the characteristics and main elements of the AOAB. AEF requires improvement to remove inconsistencies, conflicts, overlap and the addition of criteria that cover more than one aspect. In the case of a particularly complex system or game *" making a choice from the apparently very wide range of methods and tools available can itself be a complex and costly process"* [90] it is necessary to define a systematic evaluation framework in order to identify a suitable evaluation method. According to [143], there are three main methods that used to evaluate new techniques, methodologies and tools, which are: surveys, case studies and experiments. A short definition and description of each of these is provided in Figure 6.1.



Empirical method	Description
Experiment	A set of subjects is asked to perform a task in a highly controlled environment. The results are derived from observing of the subjects during the experiment, from inspecting the task outcome or from questioning the subjects at the end of the procedure
Survey	A set of subjects is asked to fill-in questionnaires either directly, or via internet. The results are derived from the valid answers to the questionnaire
Case study	A project, an activity or an assignment is monitored with respect to the methodology under study. Results are directly derived from project measurements

Figure 6.1: Evaluation Methods [143].

From [28], the Figure 6.2 describes a comparison between the evaluation methods employed in game engineering. The domains of the comparison are selected as follows: (a) Game software engineering is a subcategory of conventional software engineering. (b) traditional software engineering is a mature scientific area. (c) Agile software development is selected because it is a young domain and consequently complements traditional software engineering, with respect to area maturity.

The results of the study suggest that case studies are most frequently employed in Agile software development research. In addition, surveys are most frequently employed in game development research and finally, experiments are more frequently conducted in software engineering research. The AEF procedure will include surveys, case studies and experiments, as shown in Figure 6.3.

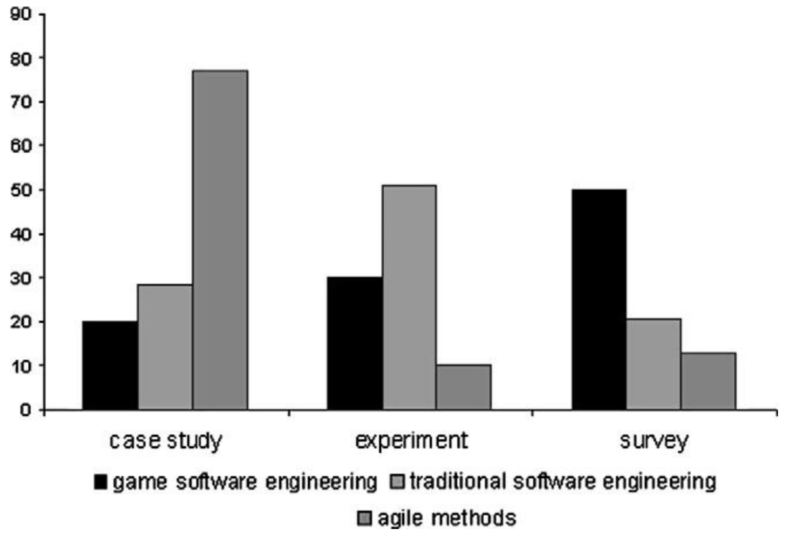


Figure 6.2: Comparison Between Evaluation Methods [28].

Most popular software engineering methodologies are created to be generic and suitable for many kinds of projects. The evaluation methods for the AEF are divided into two types: qualitative and quantitative evaluation.

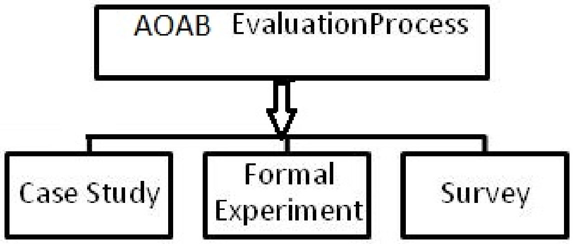


Figure 6.3: AOAB Evaluation Process

6.4.1 Survey Approach

This approach does not involve the practical use of evaluation. The survey can be used in many ways; sometimes, an organization or individual with experience is asked to use the methodology and then provide information about the methodology. This information can then be analysed using standard statistical techniques [82].

This technique is used in the present AOAB evaluation by conducting workshops in the academic and industrial sectors. This part of the research will be described in detail in the chapter Eight and Nine. Another form of evaluation survey is to compare the new methodology with similar existing methodologies. Methodology evaluation is a complex issue that is subject to various different points of view. The results of AOAB have already been compared in terms of development time, project management and the benefit of adapting two methodologies to create new hybrid methodology. In the evaluation methodology process, the following methodologies are selected to be compared with AOAB methodology:

1. DeLoach et al. [50] present an enhanced version of MaSE, called an Organisation-based Multi Agent Software Engineering (O-MaSE) methodology, to address a lack of industrial strength methods and tools to support multi agent development. Management and deployment issues are initially covered in O-MaSE, but not in a way that involves the customer in the process or accepts any change in development time, as used with

AOAB. Furthermore, the testing and evaluation phase is not included in the O-MaSE life cycle. The O-MaSE methodology framework is based on two meta-models: SPEM 2.0 and the O-MaSE meta-model. The SPEM meta-model defines methodology-related concepts, while the O-MaSE meta-model describes product related concepts [50].

2. Chella et al [41] present a hybrid methodology named PASSIAgile, to be used in robotics, the main idea of which is similar to AOAB, a hybrid between AOSE and Agile methodologies. Chella uses PASSI for the named code generation phase. This phase is largely supported by (Agent Factory) to automatically compile agent structure, patterns reuse and code generation [40]. For AOAB, the reason for selecting MaSE is its ability to define the agent goal in the initial phase. Furthermore, MaSE is able to add a new goal to an agent in any phase.

3. Jamont et al.[72] present a DIAMOND multi agent methodology that focuses on the hardware and software requirements of the system. The implementation phase consists of partitioning the system hardware and software parts to produce the code and the hardware synthesis. On the other hand, AOAB focuses only on the software requirements part of game creation, although both use an iterative process.

According to [84], DESMET is a method for evaluating software engineering methods, and tools are mentioned the advantage and disadvantage of survey as follows:

Advantages of Survey

- They make use of existing experiences (i.e. existing data).
- They can confirm that an effect generalises to many projects/organisations.
- They make use of standard statistical analysis techniques.
- They require less time and effort than the formal experimental approach.

Disadvantages of Survey

- They rely on different projects/organizations keeping comparable data.
- They only confirm association, not causality.
- They can be biased due to differences between those who respond and those who do not respond.
- The difficulty of finding the right people to ask to participate in the survey, particular if the evaluated methodologies are not popular

Several tasks are associated with this survey, such as choosing the type of survey, for example a web-based survey or personal interview, building the survey documentation, such as a questionnaire, and identifying people to participate in the survey. Finally, the evaluators will run the survey and collect and analyze the responses according to the survey design [46]. In Chapters Eight and Nine, the workshop process will be covered in detail, from building the questionnaire, inviting participants, collecting data and, finally, analyzing the participant feedback.

6.4.2 Formal Experiment Approach

A formal real world experiment involves asking real companies to perform a task or play a game using the new methodology. The results will be analyzed using standard statistical techniques. For the AOAB methodology, a formal experiment has been conducted with undergraduate students completing a game design and implementation assignment. This part of the work will be covered in detail in Chapters Eight and Nine. Formal experiments are appropriate for exploring relationships; this approach is likely to produce the most reliable results, as it seems to reduce the influence of single assessor differences. It is, however, the most costly and time consuming approach [46].

6.4.3 Case Study Approach

The case study is similar to an experiment, but the level of control is lower in the sense that they are mostly observation-based studies [143][83]. Case studies is easier for a game developer organization to perform as there is no replication; the only limitation relates to the confidence that case study will allow an assessment of the true effect of the methodology. For AOAB, a case study approach is used to create a test drive game. The industry case study required approximately three months of work with an interested company who collaborated on the research free of charge as they are interested in improving their work and knowledge. The number of participants involved in the case study was four individuals, including the researcher. This part of the evaluation will be covered in detail in Chapters Seven and Nine, in the critical analysis of the research results.
According to [84], DESMET is an appropriate method for evaluating software engineering methods and tools are mentioned the advantage and disadvantage of case study as follows:

Advantages of Case Studies

- They can be incorporated into normal software development activities.
- If they are performed on real projects, they are already "scaled-up" to life size.
- Provide a practical evaluation performed by an actual user of the methodology.
- They allow the researcher to determine whether or not expected effects apply in specific organizational and cultural circumstances.

Disadvantages of Case Studies

- With little or no replication, they may yield inaccurate results.
- There is no guarantee that similar results will be found in other projects.

- There are few agreed standards or procedures for undertaking case studies. Different disciplines have different approaches and often use the term to mean different things.

6.5 AOAB Evaluation Framework Procedure

The evaluation framework procedure is the way in which the evaluation is organised. A well developed structure for the evaluation framework allows the results to be considered in terms of what AOAB methodology adds to existing methodologies. Most evaluation frameworks can be quantitative, qualitative or a hybrid evaluation. The suggested evaluation framework for this research consists of three main steps, which are as follows:

1. **Define the parameter needs for the evaluation**

   In this section the main criteria must be identified, which are necessary for the evaluation. Those criteria are based on the following three roots: game, Agile and agent root.

2. **Define the methods used for the evaluation framework**

   The evaluation aims to explain how well AOAB fits the needs and culture of an organisation. The most popular evaluation methods are: Case Studies, Survey, Formal Experiment, Feature Analysis and Bench Marking. Generally, the evaluation framework applied in this research will use survey, case study and formal experiment, which will be explained in detail in Chapters Seven, Eight and Nine.

3. **Critical analysis of results and suggestions**

   This part of the evaluation framework will critically analyse the results of the previous steps and make suitable suggestion regarding AOAB. The critical analysis of the AOAB methodology will be discussed at length in Chapter Nine.

6.6 AOAB Evaluation Time Scales

As there are many different evaluation methods, the methods are ranked in order of the likely time scale required to perform an evaluation. The relative time scales are divided into the following types:

First: Long, for projects over three months. The evaluation methods addressing long term projects take the form of academic experiments, as the assignment preparation and submission process takes 14 weeks.

Second: Medium, several months. Industry case studies are generally used.

Third: Short, several weeks; the workshops take just one day to present, and several weeks of preparation [84].

It is important to ensure that the choice of evaluation method conforms to external time scales constraints. AEF covers all of the discussed timescale types.

6.7 Summary

The evaluation of the methodology is a tool that helps facilitate a better understanding of the steps required to carry out a quality evaluation. An overview of the evaluation methodology plan is presented in this chapter, based on the AOAB evaluation criteria. The goal of the evaluation is to reach decisions and make the necessary enhancements to the AOAB methodology. This can be achieved and documented with confidence when a systematic, well-planned evaluation process is applied. The degree of confidence in the results obtained through a quality evaluation process clearly justifies the effort involved in performing the process systematically.

Chapter 7

Evaluation by Applying AOAB Methodology to Industry

7.1 Chapter Overview

This chapter evaluates the AOAB methodology in the industry sector. A game company name"**3D Design**" with medium size projects is selected for use in the study. This game company principally provides multimedia and gaming for medium size projects in the area of television and learning games. The industry case study covers the whole life cycle of AOAB methodology, from requirement specification to game release. First, the concept paper and GDD is defined, then the analysis and design phases are addressed. In those phases, the QSEE software is used to create the required AUML diagrams. In the implementation phase, the 3Duninty game engine is used for game implementation. The game is a serious game, which is based on imitating a driving test exam. This chapter includes a discussion of how AOAB can be evaluated according to AEF. At the end of this chapter, a critical analysis is presented, as well as an evaluation of the results of current work.

"3D design" is a game development company showing a strong interest in adapting AOAB to implement a serious game, using Agile methodology for the game creation. A serious game genre is selected for the game idea of this research; the game is mainly designed to help people who want to obtain a driver's license.

7.2 Serious Game

Serious games are useful for teaching peoples how to interact with each other and with their environment. The best serious games are simulations that have the appearance of a game, but whose events or processes are real. Usually they include business domains or military operations; many popular entertainment games are based on business and military operations, but with simpler rules [66].

Mike Zyda [151] provides a definition of a 'serious' game, which 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."*

The relation between serious games and the AOAB proposed in this research is that the agent can play the roles of adversary and collaborator in a serious game. The serious games can entertain, but its primary goal is to educate, investigate or advertise. In particular, this research is oriented to a category of serious games called game-learning, the main objective of which is training [66]. Many commercial applications, including IBM INNov8 a [8] have built different games. The important question is *what methodology underlying their development is used*. As mentioned by [66], the processes and methodologies followed are based on old and rigid processes and methodologies that do not consider the development of a serious game as anything remarkable.

Games have been used for educational purposes for many years [141]. Games can be integrated into higher education in three main ways. First, traditional exercises can be replaced with games motivating students to put extra effort into exercises, giving course staff the opportunity to monitor how students complete the exercises in real-time [124] [63]. Second, games can be used within a traditional classroom lecture environment to improve the participation and motivation of the students through knowledge-based multi player games, played by the students and the teacher [140][139]. Third, game development projects can be used in computer science (CS) or software engineering (SE) courses to acquire specific CS or SE skills [57] [148].

7.3 Drive Test Game

The main aim of using an experimental study for "Drive Test game" is to demonstrate how the actors, scenes, context and game environment could simulate, specified, design and development using the AOAB. The agent needs to be informed about player characteristics and use them in the game play control, and the simulation must feature engaging comments that will motivate the player to play and learn [79].

The driving test game is an educational game with a graphic format. The player controls the character and this character interacts with different environments and situations and must make decisions. The game is made up of several scenarios, or locations, based on the Oman learn driving centre website [12], where the driver must perform certain actions correctly to pass the three parts of the driving exam.

7.4 Requirement Specification

The first step of the game creation is to write the requirements scripts. This script is usually called a concept paper [27]. The game designer will convert the information from the concept paper to a GDD, which will serve as a guide throughout the development process. The last steps of the requirement specification are to create a Sprint Backlog based on the GDD, which should provide an estimation of the number of iterations needed prior to game release.

7.4.1 Game Concept Paper

The driving test game is a serious game. The goal of the game is to simulate the environment of the driving test that must be passed to obtain a driving license for an automatic or manual car. The environment of the game will be split over three areas, based on information from [12], which contains basic information required to know and understand how to drive safely, particularly for the road test, drum test or parking test and, finally, the traffic sign test.
The current web page [12] includes most of the scenarios that a driver could face. The proposed game will be more useful and understandable than any PDF file or web page. Furthermore, the language of the web page is in English only, which, in the Sultanate of Oman, many people do not understand well. In the game proposed by this research, both the Arabic and English languages will be used.
The first environment of the game is the traffic sign test. Usually, the test presents 5 to 8 signs. The game will ask for 5 signs at random, with an option to view all signs and their meanings. The second environment is the parking test or drum test, which deals with how

to park a car between two rows of drums, or between two cars. When the driver parks in a handicapped parking space, or touches a drum or another car, the game will highlight the incorrect parking. The third environment is the road environment, where the driver is sitting with an examiner and following their orders on a real road. This is the hardest part of the driving test, as the examiners sometimes give the driver incorrect orders to follow. The real test takes approximately 15 minutes and the same time limitations will apply in the game. The driver will sometimes need to refuse the examiner if wrong orders are given, for example if the examiner asks the driver to park in handicapped parking; if the driver accepts this order then they will fail the test.

7.4.2 Game Design Document (GDD)

For most designers, GDD is a fun and interesting activity, as they are able to apply the vision that was presented in the concept paper. The complete GDD is not an easy piece of work [34]. A poorly elaborated GDD can lead to a need for reworking and loss of investment in game development phases. Therefore, this research will analyse several available GDDs found in existing literature, comparing the findings to propose an improved general GDD placed alongside a commercial GDD. Most authors agree that there is no established structure for a GDD, as there are significant differences between games. However, there is a set of common elements of game design [117]. These common elements are used to create a general GDD as a template that is easy to use and can be applied to different game genres. Appendix A.1 includes a template with an explanation of GDD elements. Et. Jesse Schell [76] suggests creating more than one document to serve all necessary purposes. Schell defines six groups, which need to remember and communicate different things, as shown in Figure 7.1.

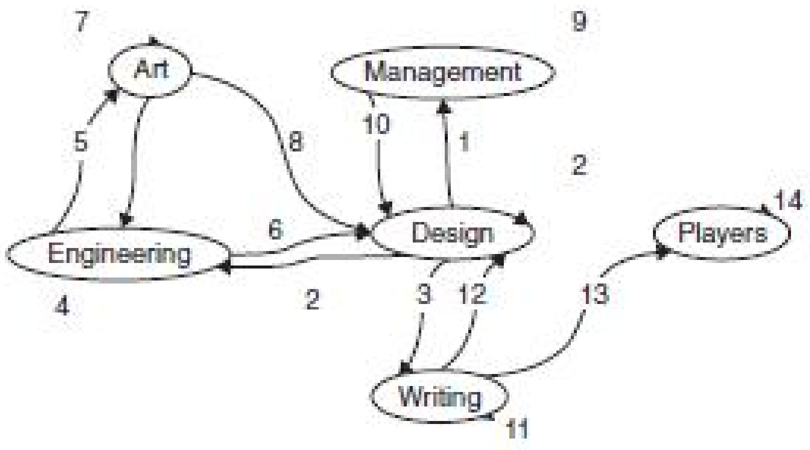


Figure 7.1: Game Design Document Elements [76].

The GDD template is constituted of one document divided into multiple subsections. Appendix D.1 includes the GDD template completed to cover the requirements of the driving test game.

7.4.3 Sprint Backlog

In this research, a full GDD is created. The final step in the requirement specification of the AOAB is to translate the GDD into a Sprint Backlog. The workload was distributed among the team during the Sprint planning meeting. In this meeting, it was decided to divide the work into four iterations, where each iteration would cover one environment of the game. In the first Sprint, a full prototype will be created, as well as the functions of the traffic sign test environment.

Table 7.1: Time Plan of Our Game

Phase	Activity	Deliverables	Week
Requirement phase	Idea creation, Sprint Backlog, Capture goal	GDD, Sprint iteration	1- 3

Table 7.1: Time Plan of Our Game

Phase	Activity	Deliverables	Week
Analysis phase (Sprint 1)	Use Case, Sequence Diagram, Define tasks and role	Apply Use Case and role diagram	4
Design phase (Sprint 1)	Agent classes, Agent Architecture	Creating Agent class diagram	5
Implementation phase (Sprint 1)	Deployment Diagram	Create prototype 1	6
Analysis phase (Sprint 2)	use case, Sequence Diagram, Define tasks and role	Apply use case and role diagram	7
Design phase (Sprint 2)	Agent classes, Agent Architecture	Creating Agent class diagram	8
Implementation phase (Sprint 2)	Deployment Diagram	Create prototype 2	9
Analysis phase (Sprint 3)	use case, Sequence Diagram, Define tasks and role	Apply use case and role diagram	10
Design phase (Sprint 3)	Agent classes, Agent Architecture	Creating Agent class diagram	11
Implementation phase (Sprint 3)	Deployment Diagram	Create prototype 3	12
Testing and evaluation phase (Sprint 4)	Integrate game iterations, user evaluation, expert evaluation	Code review, final game release	13-15

In the second Sprint, a full prototype, as well as the functions, of the drum or parking test environment will be created. In the third Sprint, the full prototype and functions of the road test environment will be completed. In the fourth Sprint, all of the environments must be integrated, and the final game version tested and evaluated prior to game release. Table 7.1 shows the time schedule for the game following the Sprint planning meeting. Appendix D.2 shows all the required diagrams for the analysis phase, Appendix D.3 contains those for the design phase, and Appendix D.4 covers the implementation phase and is divided into subsections that cover all the required figures for implementation, in detail, as shown in Appendices D.4.1, D.4.2, D.4.3, D.4.4 and D.4.5 .

7.5 Analysis Phase

7.5.1 Model Goal (Goal Hierarchy Diagram)

The analysis phase is a very fruitful period; it depends mainly on what is collected in the requirement specification phase. The first step in the analysis phase is capturing goals, which usually depends on the requirement specifications phase and transforms into a structured set of system goals, depicted using a goal hierarchy diagram for the drive test game. In the goal hierarchy diagram, the main goal is defined, which is less likely to change than detailed steps and activities. The goals are organized by importance. Sub-goals are assigned to specific parent goals, and state what must done to accomplish the parent goal. Briefly, the goal model for this work is divided into four. Appendix D.2 illustrates all of the analysis phase diagrams. The first model, as shown in Figure 7.2, is the main goal model, which is made up of three sub-goals. Each sub-goal is presented in a separate diagram. The work plan is divided into three main Sprints, as shown previously in Table 7.1. Sprint four is usually integrates the three prototypes produced after each iteration.

Sprint 1

Figure D.2 shows the goal hierarchy for the first part of the driving test. The first part is typically the traffic sign test. The main goal is to pass the multiple-choice test, whereby the player must read and identify different traffic signs.

Sprint 2

Figure D.3 represents the goal model for the second part of the driving test game, which deals with the drum test. In order to pass the drum or parking test, the player must select the gear type needed for the test, which could be manual or automatic. Furthermore, the player should achieve the goal of controlling the car, and be able to drive forwards and backwards.

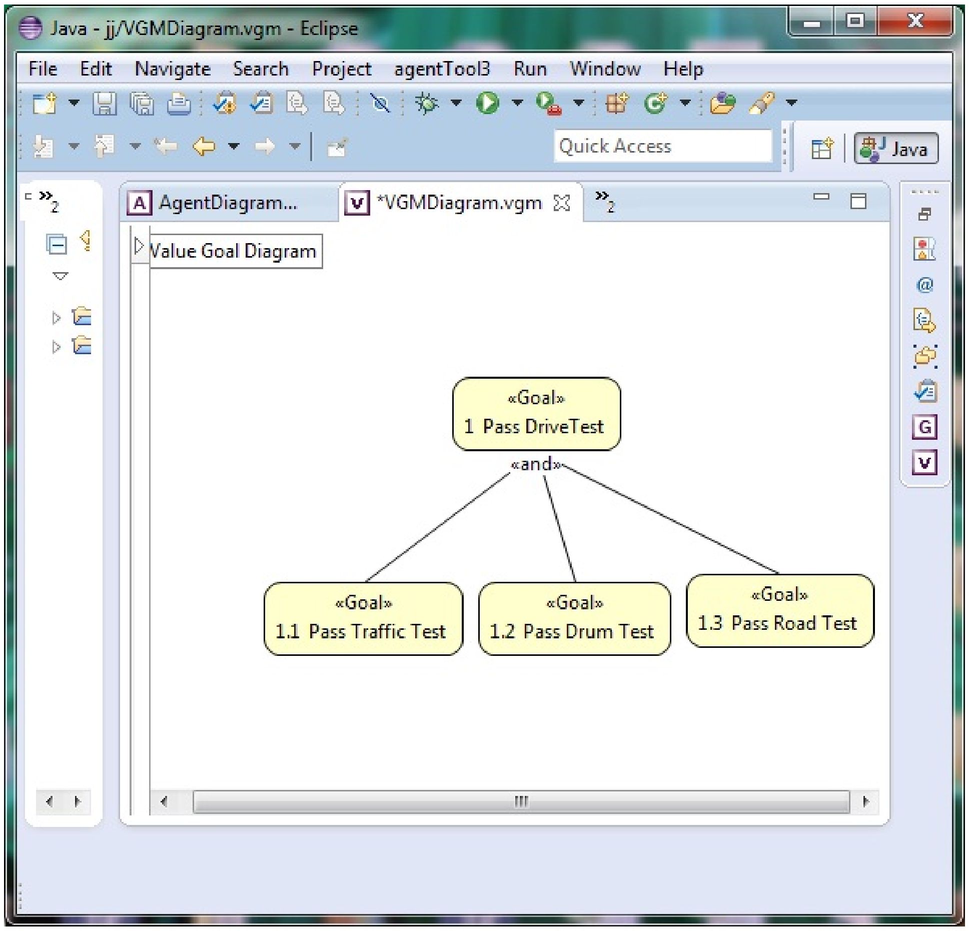

Java - jj/VGMDiagram.vgm - Eclipse
File Edit Navigate Search Project agentTool3 Run Window Help
Quick Access
Java
AgentDiagram...
*VGMDiagram.vgm
Value Goal Diagram
«Goal» 1 Pass DriveTest
«and»
«Goal» 1.1 Pass Traffic Test
«Goal» 1.2 Pass Drum Test
«Goal» 1.3 Pass Road Test

Figure 7.2: Main Goal Hierarchy

Sprint 3

Figure D.4 shows the goal model of the third part of the driving test game, the road test. This is the hardest part of the real world test, because the goal is full control over the car, and the ability to analyse the examiner's orders. The player should ignore incorrect orders and follow only correct ones. Furthermore, the player should have knowledge regarding road rules and instructions.

7.5.2 Apply Use Case and Sequence Diagram

The use case diagram determines the actual communications required within games. A sequence diagram is used to depict a sequence of events across multiple roles and defines the minimum communication that must take place between roles. The sequence diagram shows the events that occur when an agent plays the game. Each goal is typically mapped to single role, with associated tasks. To create the sequence diagram, the QSEE software is used. Four sequence diagrams are created; the main sequence diagram, Figure 7.3 illustrates the main part of the work, which is divided into three sub-sequence diagrams covering three Sprints. Appendix D.2 contains the three sub-sequence diagrams.

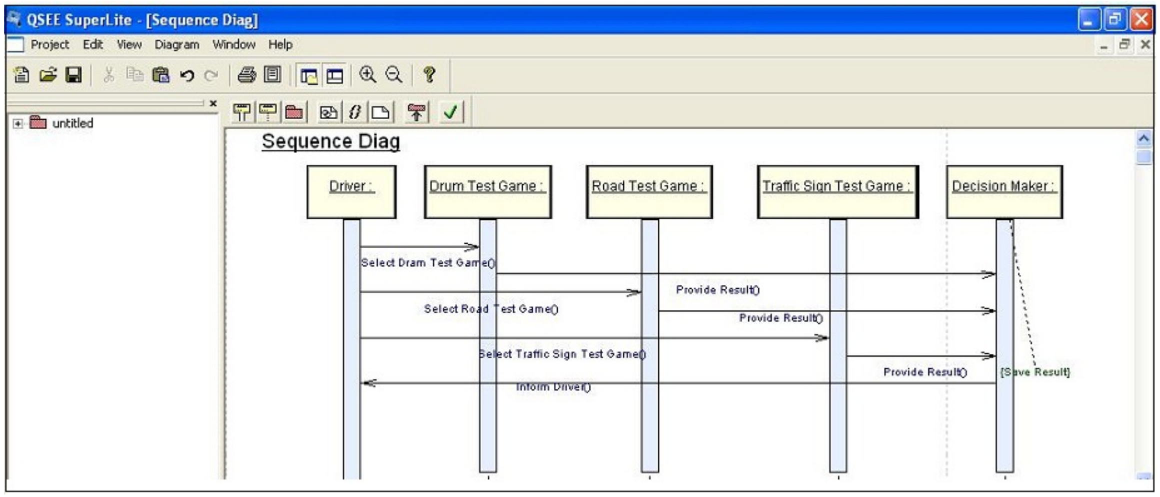


Figure 7.3: Main Sequence Diagram

Sprint 1

Figure D.6 shows the sequence diagram of the first Sprint, which covers the steps of the traffic sign test from the start of the test until informing the driver of the final result.

Sprint 2

Figure D.7 represents the sequence diagram of the second Sprint, which covers the drum or parking test steps from the start of the test until informing the driver of the result.

Sprint 3

Figure D.8 shows the sequence diagram of the third Sprint, covering the steps of the road test from starting the test to informing the driver of the result. In Figure D.7 and Figure D.8, new roles are added that deal with car control, to be sure that the driver has the ability to drive and control the car. Furthermore, another additional option is to select the gear type, either automatic or manual. This is because, in many countries, including Oman, there are two types of driving license, one for automatic cars only, the other for both automatic and manual cars.

7.5.3 Role and Task Model

The third activity in the analysis phase is the role and task model. The developer must ensure that all the necessary roles are identified, and develop tasks that define role behavior and communication. The same classification is used to divide the work into three Sprints. Figure 7.4 shows the main role diagram.

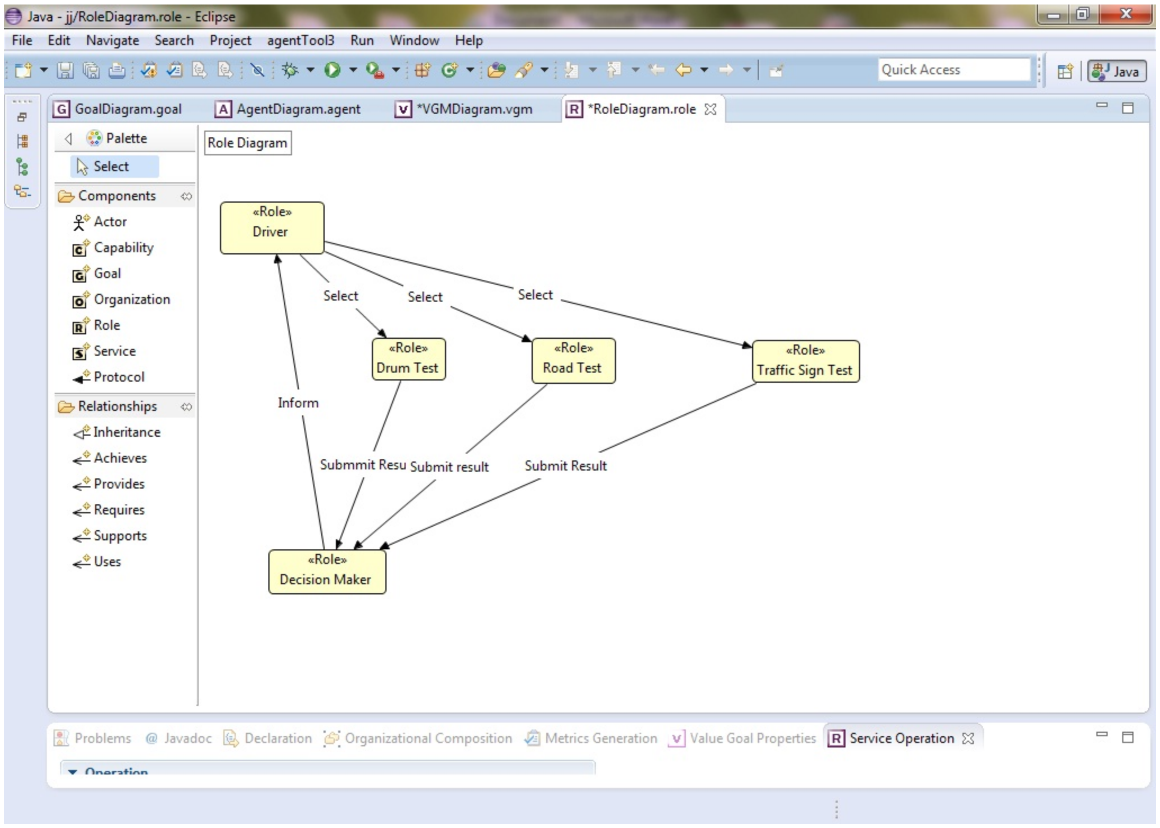


Figure 7.4: Main Role Diagram

Sprint 1

Figure D.10 relates to the traffic sign test; it has more detail, showing the result, giving a score mark and adding a database that contains most of the traffic signs with their description and usage.

Sprint 2

Figure D.11 covers the drum or park test. In this part, the driver must drive a car, control the car, select a parking spot and, finally, park the car successfully. The job of the examiner is to check the parking steps and decide whether the driver will pass or fail the test.

Sprint 3

Figure D.12 covers the road test. This part is the hardest for the driver in the real life test, and also in the game, as the examiner will give driver both right and wrong orders. In this case, the driver should analyse the orders according to their experience, and then follow the correct ones, ignoring the wrong orders. At the end of the test, the examiner will inform the driver of the final result.

7.6 Design Phase

7.6.1 Creating Agent class

The agent class diagram plays a specific role. In each agent diagram, the capability of the agent is defined. In this research, three agent diagrams are provided, one for each of the three Sprint phases. Each Sprint will cover one game level. Appendix D.3 contains the three sub-sequence diagrams.

Sprint 1

Figure 7.5 represents the first level of the game, which is the traffic sign test. For this stage, three agents are created: an examiner, player and decision maker.

Sprint 2

Figure D.14 shows the second level of the game, the drum test. In this part of the research, the previous agents from Sprint one are used. Additional agents, such as car control and car parking, are also created.

Sprint 3

Figure D.15 shows the third level of the game, the road test. Again, the same agents from previous sprints are used, with some additional roles.

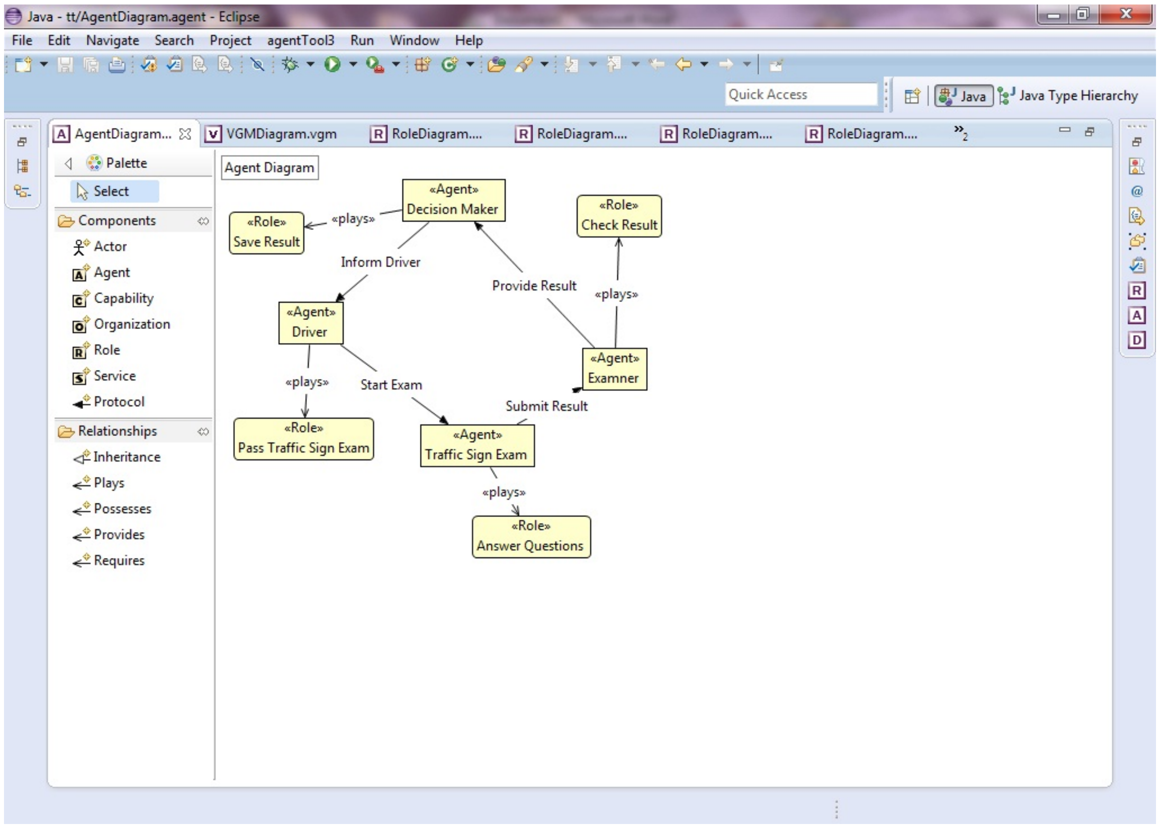


Figure 7.5: Agent Diagram (Sprint 1)

7.7 Implementation Phase

7.7.1 Deployment Diagram

In the implementation phase, a deployment diagram is used as shown in Appendix D.4.1. Figure D.16 illustrates the deployment diagram for this work, which contains a drum or park examiner, a traffic sign examiner, a road examiner; the driver will select one part at a time, it is not necessary to sequence.

7.7.2 Implementation Section

This section will discuss the actual implementation, using 3Dunity as a game engine.

Appendix D.4 relates to the implementation section and has been divided into subsections according to the game layout in Appendix D.4.2, the communication messages that appear to the player in Appendix D.4.3, the Java Script code used in the game in Appendix D.4.4 and finally, a screen shot of the sequence of the game in Appendix D.4.5.
The next sections will explain the implementation section, according to two directions: The first direction is divided into three categories: game layout and graphics, the animation added to the game and, finally, the programming language and Java code used in the game. The second direction shows how the actual game was implemented based on the previous design, which was divided into four Sprint phases.

Game Layout and Graphic

The initial step is to import the car model to the 3DUnity working area, as shown in Figure D.17 which will appear in 3Dunity as in Figure D.18
The car model is an important part of the game design. In this work, some traffic sign images are imported to form part of the traffic sign test, as shown in Figure D.19. Furthermore, in Figure D.20 some sample traffic signs are included with their Arabic and English descriptions. As the game has different levels, different images are used for the scenery and background, seen in Figures D.21, D.22, D.23 and D.24 in Appendix D.4.2.
Multiple messaging scenes are used to communicate with the player and explain how to pass the driving test, which can be seen in the following Figures D.25, D.26, D.27, D.28, D.29 and D.30 in Appendix D.4.3. Figures D.31, D.32 and D.33 illustrate the results of certain actions, if they are correct or if they are wrong, and also provides the results of the test.

Game Animation

To achieve a perceptible representation of the dynamical movement of a car, the car control Java script is added to the car model to provide movement for the car. Two box Colliders are used for the car, as well as a wheel Collider to the four car wheels. A simple way to check collisions using unity is by adding a Rigidbody component. Rigidbodies are physically simulated objects that can be used as marks. A way of using Colliders is to mark them as a trigger. For this research, it is useful for triggering a specific event in game. Furthermore, a second camera is added to follow the car's movements. As a first step, another camera is created and imported using a package named Script to use the Smooth follow Java file, and link it with our car. To achieve a better viewing experience for the player, the distance between the camera and the car is minimised.

Game Programming

As described earlier, 3DUnity has the ability to deal with Java Script, C sharp and Boo programming languages within the same project. Appendix D.4.4 illustrates the Java Script code. Java Script, as shown in Figure D.34, is used to move from one scene to another within the 3DUnity project. Each time the game moves between scenes, the scene name appears in the last line of Figure D.34. Figure D.35 is a Java script imported from the scripts package. A Smooth follow Java file allows a second camera to follow the car and enable a clear view for the player. The Java script in Figure D.36 shows a case in which the player selects the correct option on traffic sign test; his/her final score will increase by one. In the case of an incorrect selection, the score will not increase, as seen in Figure D.37. Figure D.38 illustrates the final score for the player after answering all questions in the traffic sign test.
The Java script is imported from Scripts with name car control to add control to the car, as shown in Figure D.39. Each of the previous Java Scripts must be linked with a scene or a component of a scene in order to work perfectly. Usually, if the component details are selected, the name of the linked Java file can be found. The next section will discuss the actual steps for implementing the game, based on the previous analysis and design section. The sequence of the game appears as presented in Appendix D.4.5

7.7.3 Implementation of Sprint 1

Sprint 1 is principally focused on implementing the traffic sign part of the game, and five Figures D.44, D.45, D.46, D.47, D.48 will appear to the player. If the player selects the right answer, their score will increase by one, otherwise the score will not increase. After answering all questions, the player is shown Figure D.49, the final score. The player then has two options, return to main menu, or start the next level.

7.7.4 Implementation of Sprint 2

Sprint 2 is related to how to park a car, and covers three types of parking. Initially, Figure D.50 explains the rules of this part of the game. The player is instructed to park the car within the red lines, as shown in Figure D.51 Once the player parks the car successfully, Figure D.52 will appear. The player has the option to progress to the next level or return to the main menu. In the second part of the game, Figure D.53 explains the rules to the player. The player is instructed to park the car within the red lines between two cars, as shown in Figure D.54 Once the player has parked the car successfully, Figure D.55 will appear. The player has the option to progress to the next level or return to the main menu. In the third part of the game, Figure D.56 explains the rules. The player is told to avoid parking the car under the 'no parking' sign, as shown in Figure D.57 If the player parks the car in the prohibited area, Figure D.58 will be shown. The player has the option to proceed to the next level or return to the main menu.

7.7.5 Implementation of Sprint 3

This section is under construction; Figure D.59 explains the rules to the player. At this stage, the player can drive freely, as shown in Figure D.60. The driving test game is under construction with a game development company. There is a plan to simulate Muscat Road, simulate car locations if Oman Government funding can be secured for this project. Many of EU and UAE countries use simulation driving games to help drivers get a license.

7.7.6 Implementation of Sprint 4

Sprint 4 is used to integrate all parts of the game into a complete game. The main screen, as shown in Figure D.40, us divided into five sections. Figure D.41 explains to the player the rules of this level of the game. The player has the option to take the test in either Arabic or English, seen in Figure D.42. There was a problem, initially, with the selection of Arabic language, as it is not supported by 3DUnity. It is possible to write in Arabic, but not in the standard way; 3DUnity does not mix the letters, as is seen in Figure D.43. Finally, there is a section added into the game that explains to the player the goals of the game and how to act in a real world test, as shown in Figure D.61.

7.8 Critical Review

The case study evaluation provides a complete test and evaluation of AOAB in the industry section. The first step in this case study was to find a games company interested in collaborating on the project. This was a big challenge, because in Middle Eastern country companies are interested in the gaming field from a commercial perspective. The company contributing to this research, however, showed an interest in enhancing their work. This is an interesting experiment from the perspective of teamwork and knowledge sharing. The team is small and the communication excellent. The company were happy to create a detailed GDD that saved a lot of time at the implementation stage. Furthermore, the analysis and design diagrams gave team members a clear view of what needed to be done, how it should be done, and by whom. This company is now in communication with the government, making plans to simulate parts of the exam in the official test drive. Finally, it is clear that AOAB enhanced the progress of the game in relation to the final game release. AOAB provides powerful documentation, which is useful for the game evaluation and the creation of new versions of the game.

7.9 Summary

This case study is mainly designed to evaluate the proposed AOAB in the industry sector. The company involved were glad to have access to our AOAB as it allowed consideration of the management section. The schedule, budget and user satisfaction are the main goals of any commercial game company, and AOAB deals with all of these important points in a clear and systematic way. Most games companies avoid dealing with complex methodology; they typically use clear steps and easy to follow phases, which appear clearly in the AOAB methodology. Furthermore, one of the main and critical problems in the industry sector is feature creep; AOAB provides a solution to this problem by including the customer in each game iteration, as well as complete analysis and design for games, covering all game requirements and needs.

Chapter 8

Evaluation AOAB Methodology by User Perspective

8.1 Chapter Overview

This chapter will discuss and evaluate AOAB from a user perspective by using AOAB in an academic experiment and conducting workshops to explain and evaluate AOAB with experts in the games field. Regarding the academic sector, a group of undergraduate mobile computing students have been selected to complete a game creation assignment using AOAB methodology. Other students are using Agile methodology as a course descriptive.

Regarding workshops, two workshops will be conducted, the first in a Middle Eastern country, Jordan, with participants who are industry experts. The second will be conducted in the UK, Lincoln, with academic expert participants. In order to obtain direct feedback on AOAB methodology from the different participants, they will be asked to fill in a questionnaire after the workshop. The main goal of this chapter is to present AOAB methodology to different types of game developers. According to their feedback, any weaknesses and problems with AOAB, such as creating a new GDD template, can be addressed according to the developer feedback. The new version of the GDD template is easier to use and more suitable for different game developers.

8.2 Academic Game Design Experiment

This part of the work describes an academic experiment regarding how AOAB methodology could be used to create a game for a “Further Programming for Mobile Devices and Handheld Devices” course. In this course, students have to construct, design and implement game using Agile methodology and Android software. For this experiment, the methodology for a selection of students will be AOAB instead of Agile methodology. The results of this experiment will be based on feedback from the course staff, the students’ project reports and a mandatory course evaluation.

8.2.1 Course of Mobile Computing

The Further Programming for Mobile Devices and Handheld Devices course is an undergraduate course offered to Mobile Computing (MC) students as a mandatory course, and to CS, Information System (IS) and Computing Science (CIS) as an optional course, at Gulf College, Oman, which is affiliated with Staffordshire University, in the UK. The course is taught every semester for level 6 semester 2, the last semester before students graduate; between 15 and 20 students attend each semester. This study focuses on game development, which is part of a project introduced in the course to teach students mobile computing skills. Many students are happy to create their own game, and the course allows students to create their own games as part of the assessment method. One of the main objectives of the mobile computing course specification is to teach students about mobile technologies, game user interfaces and game or software development methodologies. For students to pass "Further Programming for Mobile Devices and Handheld Devices", they must understand how to develop a mobile game, publish the game and how to approach game development methodology. The course contents, requirements, assignment contents and how the marking criteria are used are explained in the Appendix E.

8.2.2 How the Course Changed

This section will outline the changes made to the course in order to integrate AOAB methodology within the course contents. Those changes are the following:

1. **Changes to the Syllabus and Course Contents**

   It is difficult to change the syllabus of the course due to the affiliation with Staffordshire University. Therefore, rather than change the syllabus, there is an additional option to select either Agile or AOAB methodology for the game development section. The opportunity to use the Agile methodology was not removed.

2. **Changes the Assignment**

   The course staff preferred not to make any changes to the assignment requirements and evaluation criteria. The only change to the assignment is the methodology selection section, and how the methodologies are used in game creation.

3. **Changes of the Staff and Schedule**

   The change in staffing will be covered by two staff members; one of whom will deal with students in group A, and the researcher, as an invited teacher, will deal with students in group B. Group A will implement their games using Agile methodology, while group B will use AOAB methodology. The main changes made to the course schedule for group B students are:

   - An extra bounce mark for group B that covers assignment methods as an extra features section.
   - Adding a two hours introductory lecture to explain AOAB and AOSE methodologies.
   - Adding an extra two lectures as technical support to give students in group B an introduction to using the AUML modeling language to create diagrams, such as the role diagram, agent class diagram and so on.
   - The students have the option to register for the course as either group A or group B.

8.3 Workshops Conducted

AOAB is evaluated by conducting two workshops. Appendix F, includes full details relating to those workshops, such as the description, workshop materials and questionnaire template. Appendix F.7 provides some of the positive and negative feedback received from the participants.

8.3.1 Workshop Aims and Outlines

One of the priorities of the workshops is to present the concept and ability of using AOAB methodology for game creation. Furthermore, the feedback from the expert game participants will help to identify and resolve any lack of confidence or misunderstanding.
The general outline of each workshop is as follows:

1. At the end of the workshop, the participants will complete the questionnaire shown in Appendix F.3.

2. AOAB is introduced and demonstrated in detail.

3. The participants use AOAB in the case study. Furthermore, they work as a group and learn how to distribute the work as a team.

4. The participants gain knowledge of AUML diagrams, as shown in Appendix F.5 by using Agenttool3 software.

5. The participants gain knowledge of UML diagrams, as shown in Appendix F.6 by using QSEE software.

6. The participants share their knowledge. In a roundtable discussion, the facilitator insists that each group should be mix of academic, industry and Indie people, as seen in the workshop pictures in Appendix F.4.

7. The participants gain knowledge of how to create and build a GDD by using a GDD template, as shown in Appendix A.

8.3.2 Workshop Activities'

Two workshops were conducted, both following the same schedule. Each workshop was divided into three parts, as follows:

1. First part, theory is explained, followed by a 15 minute break.

2. Second part, begin to explain the AOAB life cycle in detail. The white board is used to show students how to create AUML and UML diagrams. In this part, a hard copy of QSEE, AgentTool and GDD templates are distributed. Finally, before the session ends, it is explained to the participants that the case study is about how to pass a driving test exam. The driving test exam has three parts. For the AOAB concept, this requires three iterations. Before the completion of this stage, the participants are divided into four groups; the last group is instructed to complete the GDD in a professional way, and each group is responsible for part of the game. Each participant is given a number to ensure a good mix of groups, where each group should contain individuals from industry, Indie and academia. Participants are then given a 15 minute break.

3. Third part, The participants begin creating their game, according to AOAB phases. At this time, each group is visited individually to discuss different issues. At the end of this stage, a general note of each group is taken and the questionnaire is distributed; participants are asked to write their feedback and suggestions individually.

8.3.3 Workshop Evaluation Process

The end of the workshop is an activity based on the participants' feedback, where participants work as a team. According to the participants, they received good knowledge of AOAB and how to work as a team. At the end of the workshop, the participants were asked to complete the questionnaire in Appendix F.3. Comments included:

The workshop was very formative about new techniques and methods for the game and for project management.

The strong point was about an agent that has its own rules and goals to determine the final or correct solution.

AOAB has many diagrams that can explain the game idea better than plain text.

AOAB makes the work flow faster and game updates are easier for the developer.

Helps a lot in the case of maintenance and improvement of development

Drawing on the questionnaires collected following the workshops, the next chapter will present both a qualitative and quantitative evaluation. A quantitative evaluation aims to establish different measurements, such as participant age, experience, gender and preferred game genres. Many kinds of metrics are depended on for measurements from different roots, as shown in Figure **??**. Quantitative evaluation is not as flexible as qualitative evaluation. Qualitative evaluation, in this research, aims to explain how well AOAB fits the needs and culture of industry organizations and game researchers. The qualitative evaluation identifies the expected benefit of the methodology, and feedback helps to deduct any limitations to the work. The next chapter will analyse and discuss in more detail the outcome of both workshops.

8.4 Critical Review

This chapter focuses on the user perspective and feedback, in two specific ways: First, by dealing with undergraduate students creating games for an assignment. When the course was introduced to the students at the beginning of the semester, they were told that this was the first time AOAB methodology was being used, and students were informed that they would receive extra lectures in order to explain the AOSE concept, a new trend in software engineering. Most of the students were interested in the new concept, particularly as it dealt with games. Game creation is a popular form of assignment among students; AOAB provides them with clear steps that are easy to follow, and allows them the time to add extra features, as they will not face huge problems in the implantation stage, unlike group A. It was observed that the performance of group B was a little higher than group A. These issues are considered in the final game release and the final marking evaluation. The documentation produced by group B is good quality, consistent and sufficiently describes the system. It is also noted that this documentation saves time in the coding and evaluation phase, on which students from group B spent less time than group A. This is fewer problems were identified that needed solving than were by group A. Every effort was made to obtain accurate results that facilitate the work and provide a comparative view. Different types of data were collected and will be analysed in details in Chapter Nine.

Second, by dealing with experts from different backgrounds, who were invited to both workshops. Each workshop took place in a different country and participants with different interests were invited to both. For the first workshop, most participants are from the industry sector and Indie, The second workshop was held in parallel with the GAMEON conference, and therefore most of those participants are from the academic sector. In each workshop, the facilitator was present and participated in the group discussion.

One of the pieces of negative feedback regarding the workshop itself, received from participants in the first workshop, is that the workshop would be better if it were carried out over two days, in order to get an extra explanation of the Agent concept and to provide more time to work on and finalise the case study. On the other hand, participants of the second workshop asked the facilitator to present the workshop as a conference session, as they already had knowledge of Agile and AOAE methodologies. The other conflict between users' points of view relates to GDD. The expert developers prefer a complete and detailed GDD, as presented in Appendix A.1, while Indie and small teams prefer a small and concise GDD, as presented in Appendix A.2.

8.5 Summary

This chapter describes how AOAB is evaluated based on AEF, which is presented in the previous chapter. The structure of this chapter consists of two parts: academic and workshop evaluation experiments. The next chapter will provide a critical analysis of those experiments, in full detail. One challenge discovered in this chapter is the need to conduct more than one workshop. Only expert people from academia, industry and Indies were invited, in order to make the experiment more professional. The first workshop was conducted in a Middle Eastern country, and the second in the UK country in order that the experiment be international and to get different feedback. By contrast, finding a suitable academic module was not a challenge, as this could be selected from current modules.

Chapter 9

Critical Analysis of AOAB Methodology

9.1 Chapter Overview

This chapter comprises the critical analysis and the findings of the previous evaluations of AOAB. The evaluation method covers both industry and academic perspectives, and takes into consideration the participant questionnaires from the workshops. The first part is based on the results of the industry case study, and the second part is based on the results of the academic experiment. The third part is based on the results of both workshops, and the final part, deals with the final AOAB critical analysis and the general evaluation of results, discussing the strength and weaknesses of AOAB methodology.

9.2 Part One : Industry Results

In the games industry, the creation and development process is not an easy task. The general sense from the industry case study is that the game idea is the core element of game creation, whereas the design and implementation of the idea makes a game successful in terms of marketing. Otherwise, the game will face a lot of problems and will fail to be fun for the player to use again.

It was observed that the game engine considerably increased the productivity of the game design, and 3DUnity was adequate for the project's development needs. It is easy to create executable files for desktop computer platforms; furthermore, 3DUnity does not require much effort to work with multiple platforms.

One of the research aims was to use the AOAB methodology in the industry sector and increase the quality of the game prior to final release, but there are still some limitations highlighted by the industry case study. It was required, after each iteration, to integrate the work, but this can sometimes create a problem if certain steps are not performed in the correct way. In some cases, there were problems regarding diagrams in new game iteration.

A positive of AOAB is that it is possible to estimate the time for game creation accurately, and users are happy with the management of AOAB, which involves customers and end users in each iteration to solve problems early and get feedback to enhance the game prior to release. The final release of the game was obtained after four iterations, and each iteration provided a running prototype. For reasons explained earlier, industry staff consider AOAB to be a good methodology for multilevel games.

9.3 Part Two: Academic Results

This section presents experiences and results derived from running the course. The results were collected from course staff interviews and notes, the final course evaluation, the project reports submitted by students and feedback during and at the end of the course. The students created a "Catch a coin" game and this showed the game mechanics worked effectively for both groups. Group A used Agile methodology and group B used AOAB methodology for their game creation.

1. Staff Experience

   Two academic staff members were involved in the experiment: the module leader, and the researcher, who was an invited teacher for some lectures explaining AOAB methodology concepts and requirements. The result was that five students opted to work in group B and nine students opted for group A, as shown in Figure 9.1.

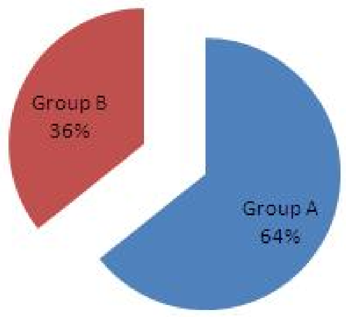


Figure 9.1: Register Students Percentage in Groups

Figure 9.2 shows a comparison between the two groups in relation to final assignment results. The final exam results were not taken into consideration. The results of the two groups are similar, with no significance difference between them.

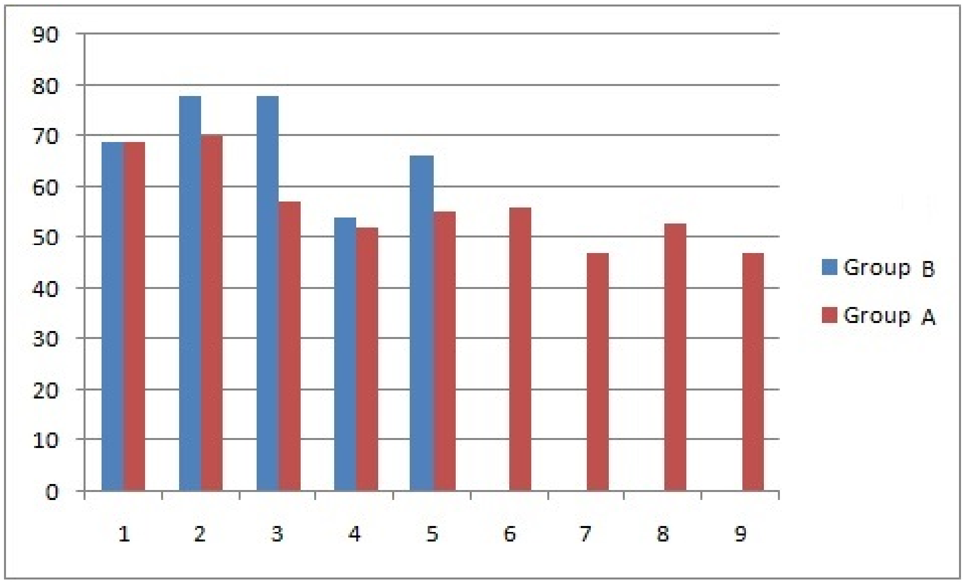


Figure 9.2: Students Assignment Results

2. Students Evaluations

   In the first week of the semester, students were given the option to choose between the groups. Group B were given extra lectures, as explained previously, in order to explain the AOAB methodology in detail. Students in group A already had a background in Agile methodology from a previous course, titled "User Centre System Development". The students from group B mentioned that they spent more time working on games than group A. Group B added extra features, such as sounds and saved scores; they were able to do this because they completed the analysis and design of their work, which positively affected the quality of the implementation phase.

   Figure 9.3, produced during the experiment, shows how group A and group B divided their time prior to game release. The abbreviations used in Figure 9.3 are listed below :

- Requirement Specification (RS)
- Game Design Document (GDD)
- Goal Hierarchy (GH)
- Sequence Diagram (SD)
- Role Model (RM)
- Agent Class Diagram (ACD)
- Class Diagram (CD)
- Deployment Diagram (DD)
- Evaluation Phase (EP)
- Game Release (GR)

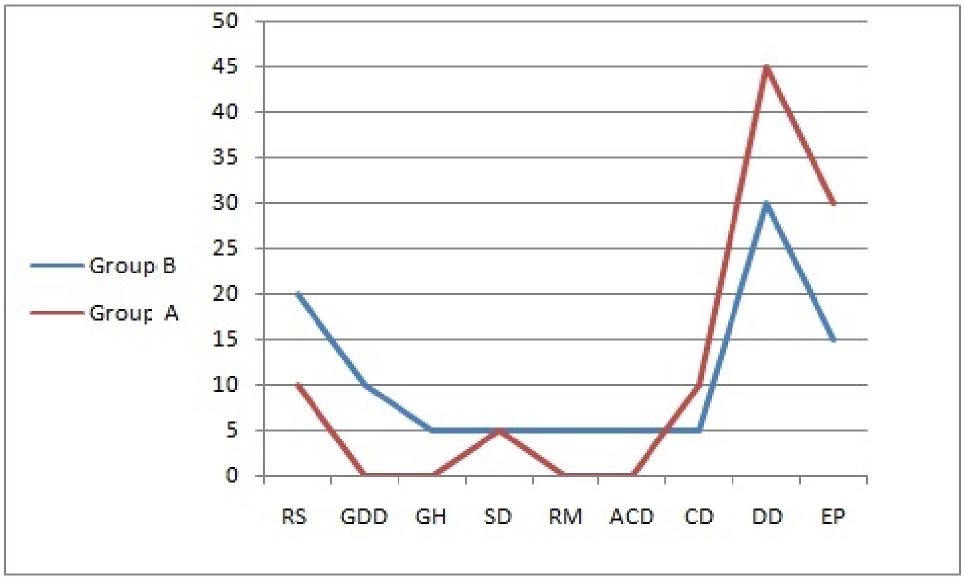


Figure 9.3: Students Assignment Percentage Time Spent on Different Phases

It can be seen that a few Agile or AOAB phases and steps took no time because they are not supported. In this case, a zero percentage is reported.

Some activities took more time in group A than in group B, and vice versa. This is clearly seen in the requirement phase, where group B students needed more time to complete this phase. On the other hand, in the implementation phase, group A needed more time, because Agile is based on documentation over coding. Furthermore, the evaluation phase took group B less time to complete, with fewer problems, whereas group A needed more time to solve the many problems that appeared in the final stage of their work. It is worth noting that group A spent most of their time coding, which is the nature of Agile, where the focus is more on coding and evaluation than design.

3. Feedback from staff and students after the course

   At the end of course, all students from both groups completed a course evaluation, a standard form for modules, in order to get official feedback from students at the end of each course. The students in group B were happy with AOAB, as the time line is broken down perfectly for them, whereas group A mention that they were busy at the end of the semester solving problems, more so than group B. On the other hand, group B students mention that it was challenging to work with AOSE, which was a new idea for them, and the course schedule is heavy in theoretical presentations in the first part of the semester. Group A students did not have a similar problem with Agile methodology, as they had already used it in another course.

   The staff member in group A was happy with the experiment and attended all of the extra classes made available to the students, and observed that the students felt challenged because they knew that the results of both groups would be compared. This had the positively effect of increasing students' interest and motivation in relation to the course assignment. The overall lessons for both students and staff were a mixture of positive and negative observations.

General comments offered by the research are to advise the module leader to communicate with Staffordshire University to change from individual submissions to group submissions, as Agile mostly uses team work.

For future work, the research team plan to examine AOAB methodology as implemented in final year projects, which is loaded as three courses, where the student spends one academic year finalising the project, and should cover all methodological phases perfectly.

9.4 Part Three: Workshops Results

Two workshops were conducted. Table 9.1 contains brief details for each workshop; further details are included in Appendix F such as workshop details in Appendix F.1 and the workshop invitation in Appendix F.2.

Table 9.1: Workshop Details.

	No. of Participant	Location	County	Duration	Date
Workshop 1	25	GameLab	Jordan	6h	26/07/2014
Workshop 2	8	University of Lincoln	UK	4.5h	11/09/2014

9.4.1 Workshop 1

The first workshop was held in GameLab. GameLab is supported by the government of Jordan, and is designed to meet the needs of the developer and companies involved in game design and development. The game laboratory can be used for gaming business meetings and workshop activities [4]. The workshop was 6 hours in duration and was attended by academic, industry and Indie participants. 'Indie' people are independent developers who may choose to work with one or more game publisher, or to self-publish their titles. Most Indie participants were aged between 15-22, and some of them were already dealing with the game company. In each workshop, emphasis was placed on allowing participants to gain experience of using the AOAB methodology in their game development and design.

Initially, the game development methodology was explained, and then the knowledge of AOAB methodology transferred through using a case study to create a game.

9.4.2 Workshop 2

The second workshop was held at Lincoln University, in the UK, in parallel with the GAMEON conference, which is organized by EUROSIS conferences. The workshop took place on the third day of the conference and eight participants attended the workshop. The participants expected to complete the workshop within one conference session of approximately two hours. For this reason, the facilitator reduced the time required to deliver the material. The participants of this workshop had good knowledge of AOSE methodology, Agile and current game development methodology.

9.4.3 Workshops Evaluation

To complete the experiment perfectly, it is necessary to analyze the data and the feedback from the participants. The iteration process of AOAB is composed of a clear number of steps, and customers are heavily involved in each iteration. This is one of the main strengths highlighted by participants in the questionnaire. The reaction of Indie participants was surprising, most of whom were undergraduate students; they much prefer systematic game creation as they need to submit strong documentation for their game idea to international games companies.

There was limited experience of dealing with agent concepts in general, as agents are supposed to be autonomous and proactive and should achieve the goals of the game without any supervision. By contrast, this lack of agent understandability was not evident in the second workshop, where most of the participants came from the academic sector.

Agile methodology is used in AOAB; most participants stated that they are already using Agile and thus have a strong background. They are pleased to see improvements to Agile, particularly in regards to documentation. It can be observed that AOAB is well suited to team work, medium or large game size and different developer types. The participants appreciate efforts to create a template with standard requirements of a GDD. Furthermore, participants mention that replacing daily Agile meetings with only necessary meetings is more suitable for the real industry sector.

The documentation produced with AOAB methodology is high quality, consistent and sufficiently describes the game. One of the issues that emerged from observing the workshops was that participants prefer to work individually; following a roundtable discussion, the team work is able to work in an organized and helpful way, and ideas and information can be shared easily by brainstorming ideas for the game.

Many of the participants felt positively about the sequence diagram, which describes the sequence of the game using an easy and understandable figure. Participants observe that iterative design is a very effective method of game design and provides an easy way to integrate all game levels. The participants of both workshops state that AOAB is not suitable all of the time, for instance for a small team or a small sized game. Participants would prefer the GDD template to be redesigned and structured according to team and game size. On this point, lengthy documentation is not preferred by programmers; expert programmers mention that if documentation is good, and not necessarily large, there will be fewer missing parts or errors cropping up during game creation.

For this reason, and because the methodology aims to be generic and cover different perspectives and points of view, the GDD has been updated to create a second version, as shown in Appendix A.2. The new version is designed to be just like an electronic form designed for small size games or teams; it covers the main requirements and features to be included in game design. Furthermore, in each GDD section, there is an extra optional section to include any additional extra details the designer may need.

9.4.4 Qualitative and Quantitative Questionnaire Analysis

This section comprises an analyses of the results of the workshop questionnaire, containing both qualitative and quantitative analyses of data. The first part will discuss the direct results of the questionnaire, and will mainly focus on quantitative results. The second part will deal with interviews and feedback from interviews, and will mainly focus on qualitative results. A total of 33 workshop participants completed the questionnaire, which was divided into three parts. The first section is designed to obtain general information about the participant. The second section is for the game developer, and the third section is designed for a normal player as shown in Appendix F.3. Most of the participants are game developers, as participants were mainly sourced through invitation. For this reason, we will not take the third section of the questionnaire into consideration.

Some of the data gathered through the questionnaire is subject to a frequency count, which means similar answers are counted to identify their frequency of occurrence, as shown in Table 9.2. From Table 9.2, it can be seen that most participants are under 40 years of age. Typically, younger game developers care more about implementation than creating professional work. It is also noted that there are more males in attendance than females, as shown in Figure 9.4 .

Table 9.2: Participants' age distribution.

Age	Total
16-23 yrs	12
24-30 yrs	9
31-40 yrs	11
More than 40 yrs	1
Total	33

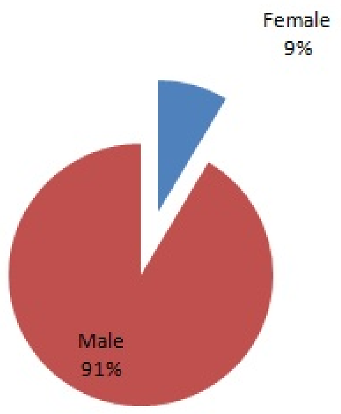


Figure 9.4: Attendance Classified by Gender Category

Figure 9.5 shows the background of participants, which is divided into three categories: industry, academic and Indies.

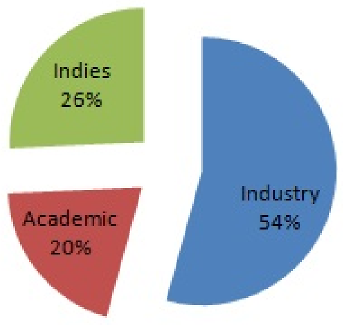


Figure 9.5: Attendance Classified by Background Category

The previous results are based on a quantitative analysis of data obtained from the workshop questionnaire. Table F.7 shows the qualitative feedback regarding the workshops. Each answer to the questionnaire questions is divided into positive and negative viewpoints. Not all comments are included in the table; the total number includes the final results. Some of the participants did not complete the questionnaire; in those cases, a column is added for 'no comment'.

9.5 AOAB Evaluation Results

The evaluation of methodology is a complex issue, from several points of view [41]. When the experiment was set up, the expectation was to receive either positive or negative feedback. The AOAB process is composed of a number of phases, and various steps within each phase. These vary depending on game design, and provide both qualitative and quantitative evaluation results. Regarding the qualitative element, the focus is on definition, using quantitative metrics where possible. Regarding the qualitative element, efforts are focused on providing a discrete value, to enhance the comprehensibility of the results.
An important question is, what are the characteristics that are needed to provide a generic game development methodology? AOAB provides the following characteristics:

1. **Flexibility:** Game design cannot always follow the same structure and standardization. AOAB provided general steps and offers the possibility to add any required extra steps in the main Sprint phase.

2. **Standardized:** Standard material and procedures needed for game design are provided. Furthermore, a GDD template is given that facilitates data collection for the game developer.

3. **Comparative:** AOAB is comparative, particularly in the evaluation phase, where expert and user evaluations are compared to solve all possible problems prior to game release.

4. **Triangulated:** A mixed method approach is taken, using a predictive and adaptive approach in AOAB.

5. **Multilevel:** Industry requirements are met by paying attention to team, management and customer satisfaction. Furthermore, academic requirements are met by using standard models and diagrams to provide a complete and comprehensive document.

6. **Expandable:** As explained earlier, AOAB provides developers with the approachability to add an extra step, requirements, data measurable in the AOAB methodology.

7. **Multi Purpose:** AOAB is currently designed to be a game development methodology. Future work might use AOAB in many distributed systems, or in MAS applications.

8. **Coverage the life cycle:** AOAB coverage of the development process in relation to software development life cycle. Furthermore, all steps of the development activities are understandable.

9.5.1 AOAB Strengths and Limitations

From the AEF, it is deduced that no evaluation methods are consistently superior. The most appropriate evaluation method depends on the different circumstances. The following points summarize the observed strengths and limitations, based on AEF results:

Strength Points

1. A high level model is provided with generic concepts that can be used to create different game genres.

2. Each phase and sub-phase of AOAB is described, to make it easy to use. Furthermore, AOAB is used in the academic and industry sector, and feedback and comments have been obtained from workshop participants.

3. AOAB pays much attention to the evaluation phase of the game design. At this phase, any problems are identified and solved, prior to game release.

4. AOAB provides guidelines for the developer, beginning with the early game requirements through to analysis, design implementation and evaluation, in order to build successful games.

5. AOAB has the ability to dynamically create agents to achieve the goal of the game. Furthermore, the developer is able to add any extra steps to facilitate the game creation.

6. Using AOAB, there is no restriction regarding the agent type or game genre, which means the developer can use AOAB in many cases.

7. AOAB provides a generic template for GDD, which is a good starting point in game creation, to make it easy and standard.

In practice, and observed in many years of experience working on methodologies, there is no general methodology that fits all game design requirements. However, it is possible to address the general interests of designers, players, programmers and other stakeholders, to make the methodology standard and generic, and usable by all developers.

Limitation Points

AOAB provides a significant advantage in creating games; however, it is not without weaknesses:

1. The main weakness identified in the workshop feedback include that AOAB is not suitable for small game sizes, or for small teams.

2. The developers must have knowledge regarding the agent in general in order to receive the benefit of the methodology.

3. The AOAB uses agent diagrams that some developers have not dealt with before. This can either be a strength, as it provides a new experience, or weakness, as users must learn and download additional software.

9.6 Summary

The overall experience of running AOAB game development methodology in different sectors was very positive. In all of the experiments, an increasing interest was noted, and positive feedback was received from game developers. Following the evaluation of AOAB, the main positive point is that it is a standard methodology that covers most requirements of game design as well as industry sector requirements, in an understandable and systematic way.

The main negative point is that it is not suitable for small games or for individual work. However, this is not a major issue, as most games are implemented through teamwork.

Chapter 10

Conclusion and Future Work

Several game development methodologies have been proposed by previous researchers. Unlike existing methodologies, AOAB formulates the requirement of game design and translates this into a general game development methodology suitable for different game genres. In this research, we have answered the following questions to conclude our work:

What I have done?

A new game development methodology has been created for both academic and industry sectors. AOAB has the ability to function in generic game genres. AOAB methodology based on adaptive and predictive approach.

Why New Methodology?

Game development has evolved to incorporate large multidisciplinary projects employing hundreds of people and a development time measured in years. Furthermore, a major issue negatively affecting the game development industry is that many companies adopt existing enterprise system development methodologies that do not fully match the requirements of gaming systems.

Why I have done it?

Most famous commercial games have more than one version, and it is for this reason that AOAB is presented. The current GDM are designed to solve specific problems and mostly suitable for specific game genres. Current GDM focuses on design and implementation phase and finally, to close the gap between in the existing GDM.

What I have found and what are the implications of these?

It is not easy to find a " common solution for common problem in games" AOAB facilitate the game creation by following sequential and standard steps. Actually, AOAB make easier to incorporate extra team members in short notice and make communication between developers easier. Furthermore, documentation more understandable, easy to follow and covering all the game elements.

AOAB is proposed as a hybrid methodology combining an adaptive and predictive approach.

This combination is generic and systematic, and generates complete and consistent documentation and implementation for games. Chapter Five of this thesis details the full life cycle of and guidelines for AOAB. The research includes a full textural example describing each phase in details. This is supported by unique sets of diagrams relating to the MaSE methodology that facilitates the work. In Chapters Six, Seven and Eight, the work is demonstrated and evaluated using a case study, experiment and workshops. Throughout the research period, every effort has been made to create a systematic, generic and standard methodology to offer an easy starting point for game developers across different sectors. As AOAB is configured based on MaSE and Agile methodology, it is easy to integrate with another type of application, such as robotics or a distributed system.

10.1 Thesis Contribution

This section will consider the original contributions discussed in Chapter One, and address them individually.

- Create a novel game development methodology which is a hybrid of adaptive and predictive approaches.
- Evaluate AOAB using different evaluation methods to arrive at a general and standard methodology.
- AOAB should be suitable for industrial and academic use.
- AOAB must cover the full game creation life cycle and pay greater attention to the evaluation phase of the methodology.
- Create a generic GDD template that is suitable for different game genres and for all game developers.

The primary aim of this research is to fill the game development methodology gap between the industry and academic sectors. The research outcome is a set of steps, processes and diagrams to guide game developers and designers throughout the game design process. This thesis enhances current game development methodology by providing a generic methodology that meets most game requirements. Furthermore, a GDD template is provided that covers the mandatory and majority of requirements for different game genres. A heavier focus is on the evaluation phase, as this is an important part of game creation and helps to solve any problems early in the process, before final game release. Furthermore, the game development life cycle provides all required Agent-UML (AUML) diagrams in order to facilitate the work, as well as easy documentation, which is needed for game updates or the creation of new versions of games.

The agent structure of AOAB can behave in an autonomous mode, and the game goal for each agent is easily defined. This research represents a new trend in the field, which opens the door to integrating and implementing more agent features within the games field. The proposed AOAB game development methodology is evaluated in many ways, in order to determine its strength and weaknesses. The results provide a possible outline for future work.

10.2 Future Work

In regards to future work, we have planned to do the following:

1. An extended version of AOAB is planned, to be used not only for game creation, but for the creation of different distributed systems that deals with agents.

2. To work in detail with agents and game metrics, which is a new field and requires more research. Metrics could provide a quantitative measurement, which is important to achieve accurate results, as is mentioned in the evaluation phase of AOAB.

3. Further research regarding game evaluation should also be carried out, as currently there is a need to compare games to identify games with the highest fun factor, as well

as many other issues. Most famous commercial games have more than one version, and it is for this reason that AOAB is presented.

4. Further research regarding re-engineering concepts, which could be implemented in the games field.

Appendix A

Game Design Document Template

A.1 GDD Template First Version

1. **General Information:** This area should present information about the game, the person who authored the document and for what company.

    - Game name
    - Game Genre: Describe the Genre for example: Role-play, Adventure, Strategy, Simulator...etc.
    - Version Number :
    - Created Date
    - Last Update
    - Organization Name
    - Game Team Members
        - Job
        - Information
        - Contact

2. **Engineering**

- Hardware Requirements
- Software Requirements
- Network Requirements
- Game Platform: need to decide first the development language to decide then the development environment and game engine such as Java, C++
- Game Engine: such as XNA, Untity3, SDk

3. **Design**

- Story overview (Gameplay)
  - Combat: If there is combat or even conflict, how is this specifically modeled ?
  - Game Levels: This is where information pertaining to level design and visuals of the level design goes.
  - Score
  - Power Up
  - Replay and Saving
- Functional Requirements (Game Mechanics)
  - Game Rules: What are the rules to the game, both implicit and explicit. Think of it as a simulation of a world, how do all the pieces interact? This actually can be a very large section.
  - Game Movements
- Game Level Overview(s)

4. **Game World (Environment)**

- General Look and Feel of World
- Area 1

  - General Description
  - Physical Characteristics
  - Levels that use area
  - Connections to other areas

- Area 2
  - General Description
  - Physical Characteristics
  - Levels that use area
  - Connections to other areas

- Area N
  - General Description
  - Physical Characteristics
  - Levels that use area
  - Connections to other areas

5. **Interface**

- Sound
  - Music
  - Sound Effected in Game

- Graphics
  - Sprites: consist of anything which can move
  - Tiles: Make up the rest of the graph such as font type and size of players in the game
  - Plot

- Menu

- Camera: Describe the way the camera will work and then go into details
- Screen Flow: A graphical description of how each screen is related to every other and a description of the purpose of each screen.
- Game Control: How does the player control the game? What is the specific commend.
- Help System

6. Management

- Game Budget
- Organization Size
- Production Size
  - Number of Locations
  - Number of Levels
  - Number of Non-Player Character
  - Number of Weapons
  - etc.
- Project Schedule
- Risk Analysis
- Test Plan

7. Writing

- Script
- Tutorial and Manual

8. player (Character)

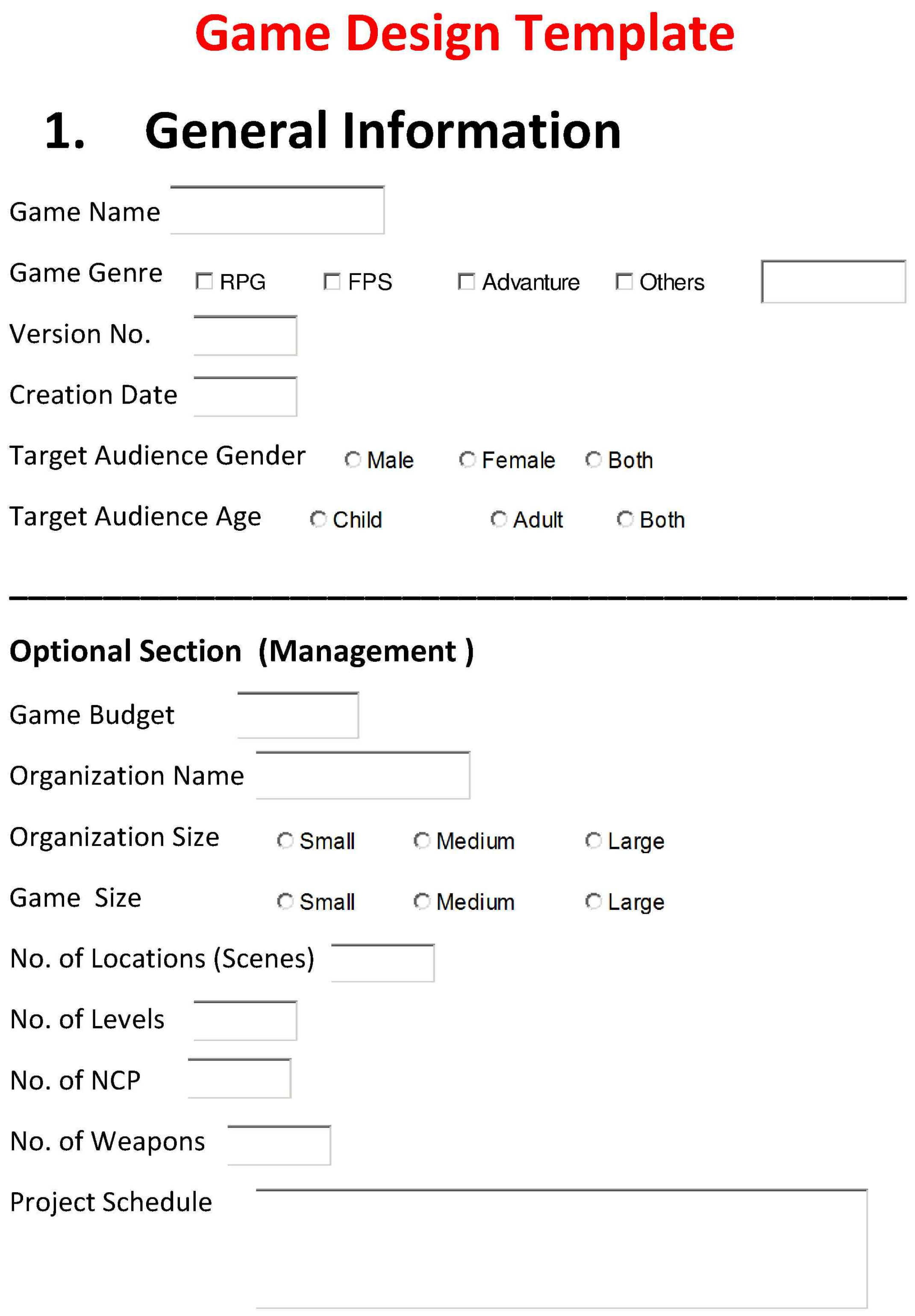
- Number of Player: The Number players that can play the game at once

- Player Details (Target Audience):. Each character should include the back story, personality, appearance, animations, abilities, relevant to the story and relationship to other characters.
  - Genres
  - Age
  - Qualification
- Non-Player Character

9. Artificial Intelligence (AI) : This is where visuals and written description(s) of the antagonistic element's behaviors.

- Opponent and Enemy AI: The active opponent that plays against the game player and therefore requires strategic decision making
- Non-combat and Friendly Characters
- Support AI: Player and Collision Detection, Path finding

A.2 GDD Template Second Version



Game Design Template

1. General Information

Game Name

Game Genre ☐ RPG ☐ FPS ☐ Advanture ☐ Others

Version No.

Creation Date

Target Audience Gender ○ Male ○ Female ○ Both

Target Audience Age ○ Child ○ Adult ○ Both

Optional Section (Management)

Game Budget

Organization Name

Organization Size ○ Small ○ Medium ○ Large

Game Size ○ Small ○ Medium ○ Large

No. of Locations (Scenes)

No. of Levels

No. of NCP

No. of Weapons

Project Schedule

2. Engineering Requirements

Hardware Requirement

Software Requirement

Network Requirement

Game Engine ☐ XNA ☐ 3Dunity ☐ SDK ☐ Others

Game Platform ☐ Java ☐ C# ☐ C++ ☐ Others

Optional Section (Extra Requirements)

3. Game Design

Story Overview (GamePlay)

Game Level Design

Combat

[✓ Game Movement] ☐ Game Score ☐ Save and Replay

Game Rules (Game Mechanics)

Optional Section (Game World)

Look and Feel World

Area Description

Character Description

Game Art

4. Game Interface

Sound	☐ Music	☐ Sound Effect	☐ Audio

Graphic Sprites

Tiles

Plot

☐ Menu ☐ Camera ☐ Extra Camera ☐ Help System

Optional Section (Extra Details)

Screen Flow and Description

Test Plan

5. Game Artificial Intelligence

Optional Section

- ☐ Non-Combat Character
- ☐ Opponent AI
- ☐ Friendly Character

Support AI

Extra AI Techniques

Appendix B

Game Evaluation Criteria Sets

Table B.1: Playability Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
Playability	1	Game Play	The game has varying activities and pacing during game play.		
	2		The game provides clear goals or supports player-created goals.		
	3		The game provides consistency between the game elements and the overarching setting and story to suspend disbelief.		
	4		There is an interesting and absorbing tutorial that mimics game play.		
	5		The game is fun for the player and enjoyable to replay.		
	6		Game play should be balanced with multiple ways to win.		

Table B.1: Playability Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	7		Player is taught skills early that you expect the players to use later, or right before the new skill is needed.		
	8		Players discover the story as part of game play and holds interest.		
	9		The games should change strategy for same failure of player.		
	10		The game should give rewards that immerse the player more deeply in the game by increasing their capabilities (power up), and expanding their ability to customize.		
	11		There are variable levels of difficulty and an unexpected outcome.		
	12		There are multiple goals on each level.		
	13		Players are able to save games in different states and resume them later.		
	14		The game gives hints, , but not too many.		
	15		Game can be played multiple times using different paths through the game.		
	16		Challenges are positive experiences rather than negative ones.		
	17		The player sees the progress in the game and can compare the results.		
	18		The player is in control.		
	18		There are no repetitive or boring tasks.		

Table B.1: Playability Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	20		The game supports different playing styles.		
	21		It allows players to build content.		
	22		There must not be any single optimal winning strategy.		
	23	Game Story	Player understands and interest in the story line as a single consistent vision.		
	24		The Player spends time thinking about possible story outcomes.		
	25		The Player feels as though the world is going on whether their character is there or not.		
	26		The Player has a sense of control over their character and is able to use tactics and strategies.		
	27		Player experiences fairness of outcomes.		
	28		Player is interested in the characters because (1) they are like me; (2) they are interesting to me, (3) the characters develop as action occurs.		
	29		Take other person into account.		
	30		Games don't waste the player time.		

Table B.1: Playability Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	31	Game mechanics	Game should react in a consistent, challenging, and exciting way to the player's actions (e.g., appropriate music with the action).		
	32		Make effects of the Artificial Intelligence (AI) clearly visible to the player by ensuring they are consistent with the player's reasonable expectations of the AI actor.		
	33		A player should always be able to identify their score/status and goal in the game.		
	34		Mechanics/controller actions have consistently mapped and learnable responses.		
	35		Controls should be intuitive, and mapped in a natural way; they should be customizable and default to industry standard settings		
	36		Player should be given controls that are basic enough to learn quickly yet expandable for advanced options.		
	37		Camera views match the action.		
	38		Player teaches skills that will be needed later in the game.		
	39		There are predictable and consistent responses to a user's action.		

Table B.1: Playability Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	40		Responses to user's actions are timely, allowing for successful interaction.		
	41		Feedback should be given immediately to display user control		
	42		Get the player involved quickly and easily		

Table B.2: Usability Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
Usability	1	User Interface	It use sound to provide meaningful feedback or stir a particular emotion.		
	2		Players do not need to use a manual to play game.		
	3		The interface should be as non-intrusive to the Player as possible.		
	4		Controls are customizable.		
	5		Menu layers are minimized, or can be minimized.		
	6		Screen layout is efficient and visually pleasing.		
	7		Device UI and game UI are used for their own purposes.		

Table B.2: Usability evaluation sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	8		The player understands the terminology.		
	9		Control keys are consistent and follow standard conventions		
	10		It provides users with information on game status.		
	11		It provides instructions, training, and help.		
	12		It follows the trends set by the gaming community to shorten the learning curve		
	13	Game Control	Player's should perceive a sense of control and impact onto the game world.		
	14		The game should be easy to learn and hard to master.		
	15		It provides immediate feedback for user actions.		
	16		The player can easily turn the game off and on, and be able to save games in different states.		
	17		The player should experience the menu as a part of the game and should contain clear help		
	18		Upon initially turning on the game, the player has enough information to get started.		
	19		There are means for error prevention and recovery.		
	20		Game controls are convenient and flexible.		
	21		The player cannot make irreversible errors.		

Table B.2: Usability evaluation sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	22		The player does not have to memorize things unnecessarily.		
	23		It allows users to customize video and audio settings, difficulty and game speed.		
	24		It provides predictable and reasonable behavior for computer controlled units.		
	25		It provides intuitive and customizable input mappings.		
	26		It provides controls that are easy to manage, and that have an appropriate level of sensitivity and responsiveness.		

Table B.3: Quality Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
Quality	1	Adaptability	The game is easily integrated with other environments.		
	2		The game includes an evaluation system, during the development process.		
	3		The game allows for new techniques and better learning.		

Table B.3: Quality Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	4		The game allows for activities that keep the curiosity and the interest of the player in the content.		
	5		The game allows players to take decisions.		
	6	Efficiency	Is there no extra information?		
	7		The game has a good program structure that allows easy access to content and activities.		
	8		The speed of communication between the program and the user is adequate.		
	9		The program execution is efficient and with no operational errors.		
	10		The system is developed with originality.		
	11	Functionality	The information are well structured and does it adequately distinguish the objectives, context, results, multimedia resources.		
	12		The game checks all the alert message.		

Table B.4: Enjoyment Evaluations Sets

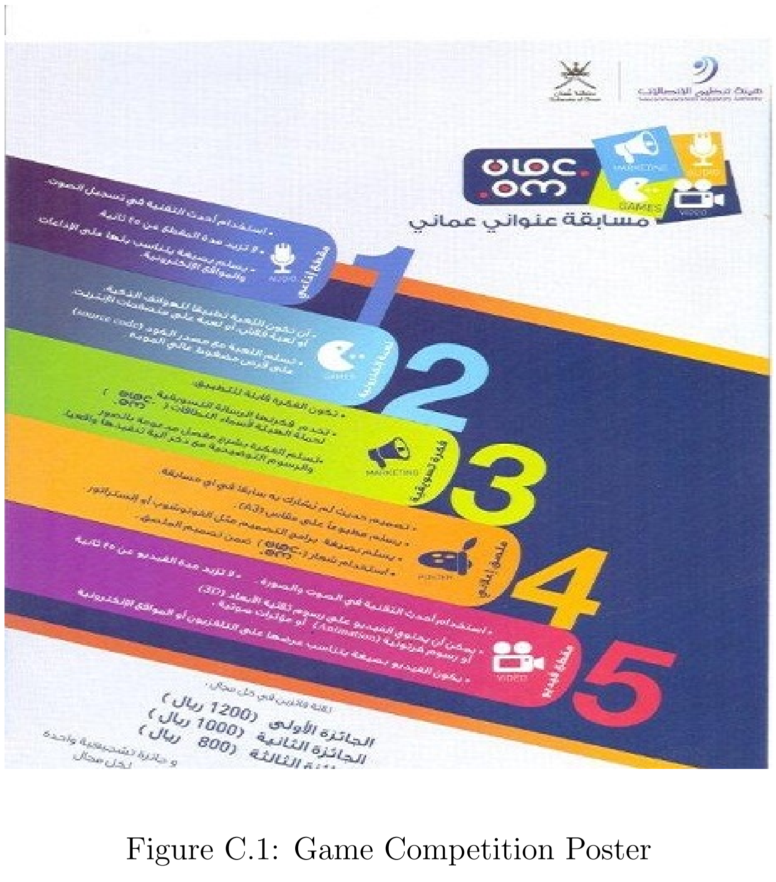
Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
Enjoyment	1		Games should provide a lot of stimuli from different sources.		
	2		Games must provide stimuli that are worth attending to.		
	3		Games should quickly grab the player's attention and maintain their focus throughout the game.		
	4		The player shouldn't be burdened with tasks that don't feel important.		
	5		Games should have a high workload, while still being appropriate for the player's perceptual, cognitive and memory limits.		
	6		Players should not be distracted from tasks that they want / need to concentrate on.		
	7		Challenges in games must match the player's skill level.		
	8		Games should provide new challenges at an appropriate pace.		
	9		Learning the game should not be boring, it should be part of the fun.		
	10		Games should include online help so the player doesn't need to exit the game.		
	11		Overriding goals should be clear and presented early.		
	12		Intermediate goals should be clear and presented at appropriate times.		

Table B.4: Enjoyment Evaluation Sets

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	13		Players should receive immediate feedback on their actions.		
	14		Players should become less aware of their surroundings.		
	15		Players should become less self-aware and less worried about everyday life or self.		
	16		Players should feel emotionally involved in the game.		
	17		Players should feel viscerally involved in the game.		
	18		Games should support competition and cooperation between players.		
	19		Games should support social interaction between players (chat etc.).		
	20		Games should support social communities inside and outside the game.		

Appendix C

Game Competition Poster



عمان.om
MARKETING
AUDIO
GAMES
VIDEO
مسابقة عنواني عماني
1
2
3
4
5
الجائزة الأولى (1200 ريال)
الجائزة الثانية (1000 ريال)
الجائزة الثالثة (800 ريال)

Figure C.1: Game Competition Poster

Appendix D

AOAB in Industry Section

D.1 GDD For Drive Test Game.

1. **General Information:**
    - Game name : Drive Test Game
    - Game Genre: Serious Game
    - Version Number : 1
    - Created Date: 22/07/2014
    - Last Update
    - Organization Name: 3D Design
    - Game Team Members
        - Name: Rula Al-Azawi
        - Job: Game Developer, Methodology Developer
        - Information
        - Contact
        - Name: Salim Al-Hajre
        - Job: Project Manager

  - Information
  - Contact
  - Name: Aflah AlBusaidi
  - Job : Game Developer, Programmer
  - Information
  - Contact
  - Name: Namariq AlRawahi
  - Job: Graphic Designer
  - Information
  - Contact

2. **Engineering**

   - Hardware Requirements: Processor above Intel Pentium 3800 MHZ, HD 850 MB, Memory more than 512 MB, Graphic card 2GB
   - Software Requirements: C sharp, JDK, 3Dunity, word, graphic s/w, sound s/w
   - Network Requirements: No need
   - Game Platform: C sharp and Java script
   - Game Engine: 3Dunity

3. **Design**

   - Story overview (Gameplay): The player interactive simulation game which simulates the car drive test. In order to progress in the game, the player must ensure that he/she could pass all the game levels. The player can move about freely in the game world and jump to any level as player wish. The game is organized in levels that correspond to the different stages/roles in the driving test. The player must at all times ensure that understand and follow the driving rules.

The player interacts with other agents through the road test with an examiner who will check that the player is understood and follow all the rules.

- Combat: No need
- Game Levels: This is where information pertaining to level design and visuals of the level design goes.
- Score: Score will use in sign test part with the other tests the player has only two options either pass or fail in test
- Power Up: No need
- Replay and Saving: No need

• Game Level Overview(s): The game will contains three levels. First, for the sign test. Second, for parking test and third for road test.

• Functional Requirements (Game Mechanics)

  - Game Rules:
    * When the player in park test touches the drum boarder, the player will fail in the test.
    * When the player in a road test touches the road boarder, the player will fail in the test.
    * When the examiner at the road test ask player to stop in nonstop area and the player follows the order and stop, players will fail in the test.
    * When the player drive in the right side of the road and examiner asks to make a U turn, the player must refuse this order; otherwise the player will fail in the test.
    * If there is a sign of no entry to the street, the player must refuse to enter.
    * The player must not drive faster than the speed limit sign on the street.
    * The player must not park in the Handicapped parking area even if the examiner wants.

        * When the examiner in road test ask player to stop in bicycle area and the player follow the order and stop, player will fail in test.
        * When the examiner in road test ask player to make left turn and the player didn't use the flash sign left, player will fail in test.
        * When the examiner in road test ask player to make right turn and the player didn't use the flash sign right, player will fail in test.
        * When the player in sign test get score three or greater, player will pass in test.
        * When the player in sign test get score less than 3, player will fail in test.
    - Game Movements: the player has the right to move between levels freely

4. **Game World (Environment)**

    - General Look and Feel of World
    - Area for sign test
        - General Description: This area was designed to help the player in two languages. The player could select either Arabic or English language for test. Before test start, the player will read a description about how to answer questions. After finalizing the exam, the player will inform with the final score.
        - Physical Characteristics
        - Levels that use area: First level
        - Connections to other areas: connection is optional when a player passes this area or not she/he could go to park test or road test.
    - Area for parking test
        - General Description: The player must park the car successfully in the parking area without touching the boarders. The player should also have the ability

to park the car between two cars. Finally, if the player park the car in not allowed park area, she/he will fail in the test.

  - Physical Characteristics
  - Levels that use area: Second level
  - Connections to other areas: Connection is optional when player pass this area or not could go to sign test or road test

- Area for road test
  - General Description: The player must drive the car in the real street and follow the Wright order from the examiner and ignore the wrong order from examiner. This part of the game also should provide two language speaking for examiner either Arabic or English to be selected by player before start the test.
  - Physical Characteristics
  - Levels that use area: Third level
  - Connections to other areas: connection is optional when player pass this area or not could go to park test or sign test

5. **Interface**

- Sound: We will need the following sound: driving sound, crash sound, sound for wrong selection and sound for right selection
  - Music: Is optional in park level and in road level
  - Sound Effected in Game: The examiner will speak in two language, Arabic and English when gave instruction to the driver.
- Graphics
  - Sprites: The car need to move and controlled by player
  - Tiles: Need to enter a different type of road sign.

  - Plot

- Menu: The menu will contain five options. first three option for each level and the player could select any level to start not sequentially, fourth option for about the system and last option for game exit
- Camera: We could show the driver different view of the road and we need to use two cameras for forward and backward drive.
- Screen Flow :The sign level will divide into sub-screen related to the number of signs we will add it. The park level will divide into sub-screen which simulated from the real world. The road level will simulate the Muscat city road
- Game Control :The payer controls the car by Stearns, gear front, gear back, speed push, break push, left flash sign, right flash sign, speed measure.
- Help System: Explains to player how to play the game and mention that the player need to follow only the right order from the examiner.

6. Management

- Game Budget: Will depend on requirements
- Organization Size: Medium size from 3-10 persons
- Production Size: Medium size
  - Number of Locations: Three locations
  - Number of Levels: Three levels
  - Number of Non-Player Character: Two non-player character (car and examiner)
  - Number of Weapons: No need
- Project Schedule: Around five months. First month for perpetrations and one month for each level and last month for testing

- Risk Analysis
- Test Plan: We will follow the evaluation phase of AOAB.

7. **Writing**

    - Script
    - Tutorial and Manual

8. **player (Character)**

    - Number of Player: Only one player can play the game at once
    - Player Details (Target Audience): Each character should include the back story, personality, appearance, animations, abilities, relevant to the story and relationship to other characters.
        - Genres: Both male and female
        - Age: Above 18 years
        - Qualification: Adult can speak Arabic or English and completed a driving course.
    - Non-Player Character: We have two non-player character car and examiner.

9. **Artificial Intelligence (AI):** This is where visuals and written description(s) of the antagonistic element's behaviors.

    - The player must have the ability of decision making by ignoring the wrong order from the examiner and how the driver could drive in different circumstances.

D.2 Analysis Phase Diagrams

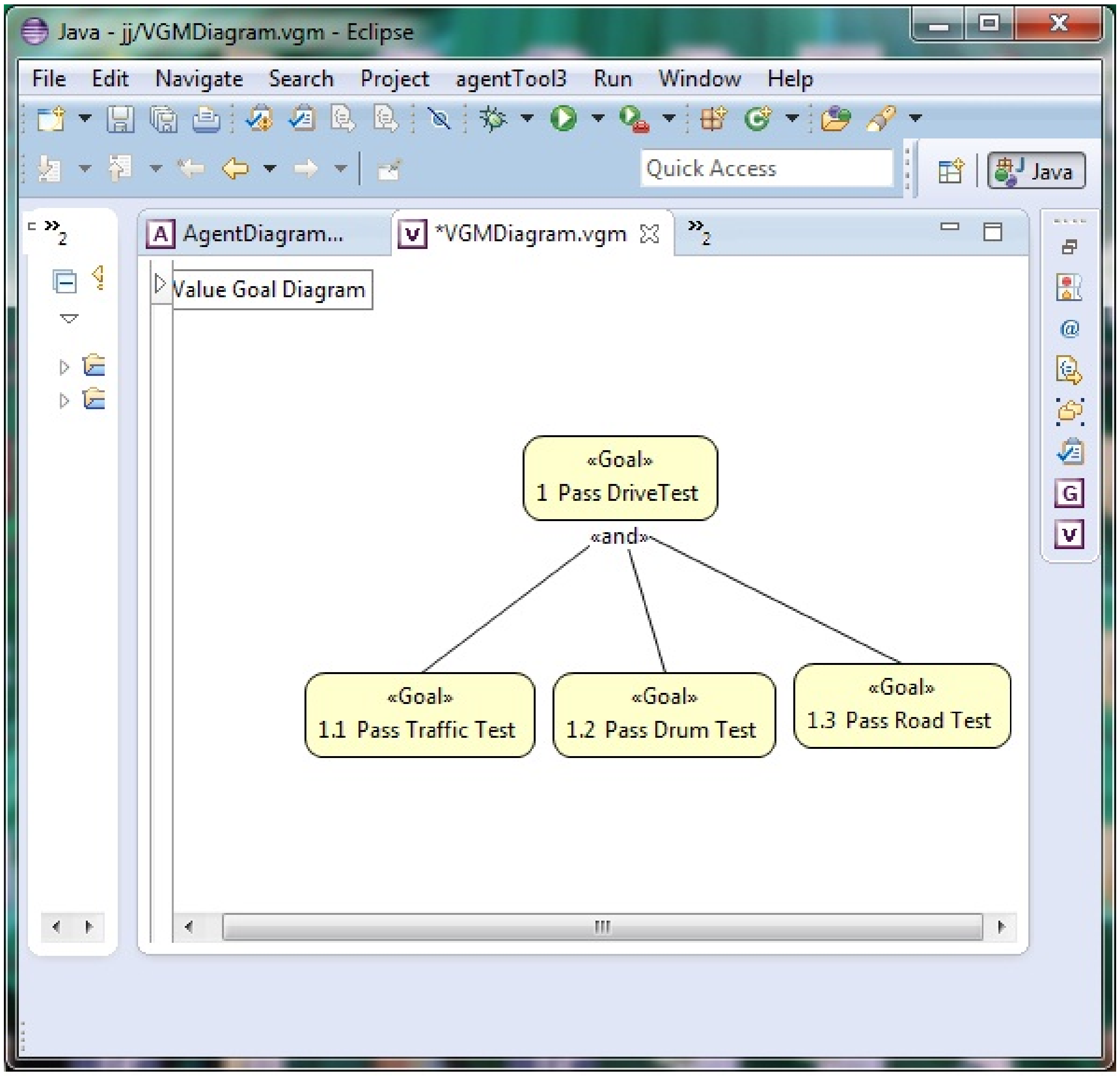


Figure D.1: Main Goal Hierarchy

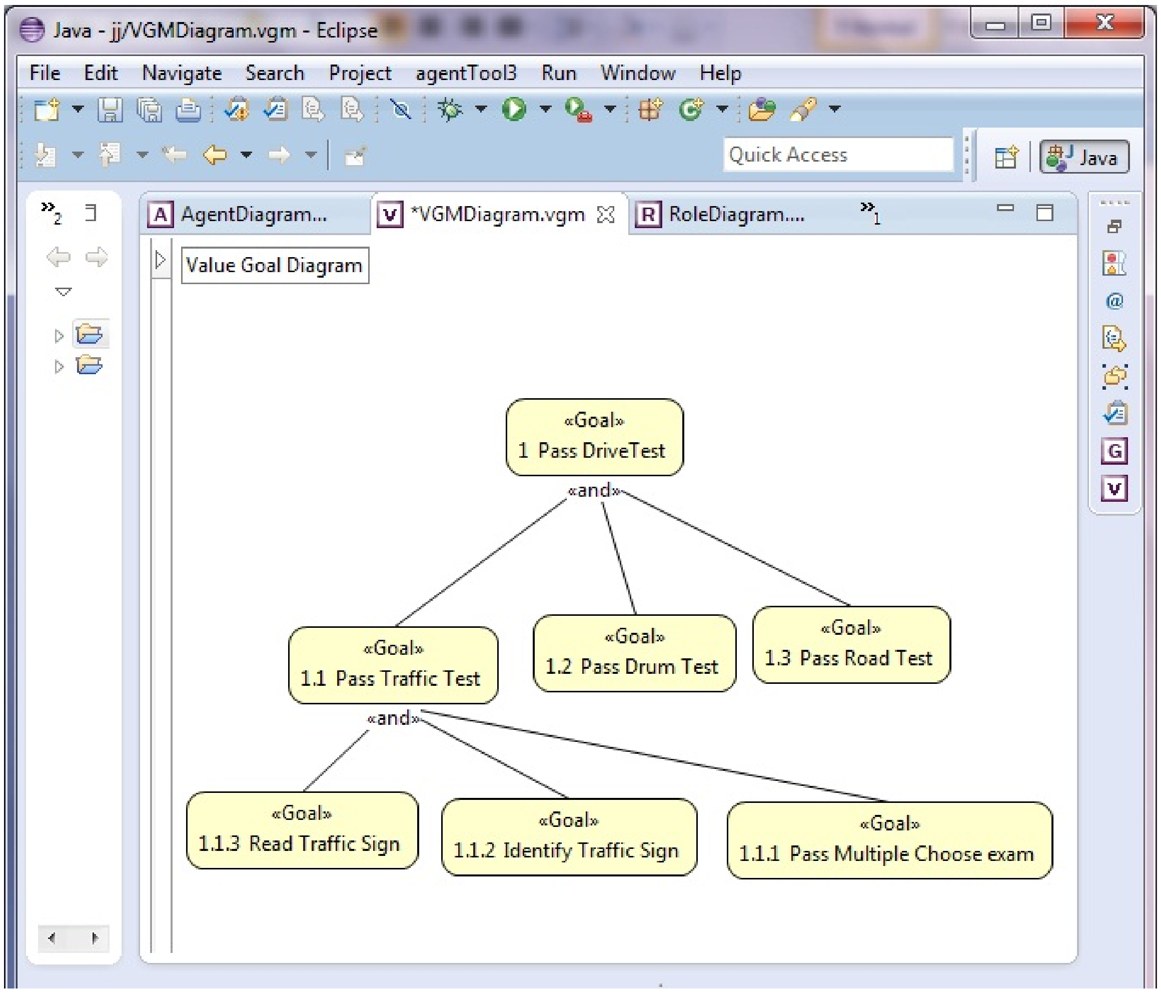

Java - jj/VGMDiagram.vgm - Eclipse
File Edit Navigate Search Project agentTool3 Run Window Help
Quick Access
Java
AgentDiagram...
*VGMDiagram.vgm
RoleDiagram....
Value Goal Diagram
«Goal» 1 Pass DriveTest
«and»
«Goal» 1.1 Pass Traffic Test
«Goal» 1.2 Pass Drum Test
«Goal» 1.3 Pass Road Test
«and»
«Goal» 1.1.3 Read Traffic Sign
«Goal» 1.1.2 Identify Traffic Sign
«Goal» 1.1.1 Pass Multiple Choose exam

Figure D.2: Goal Hierarchy (Sprint 1)

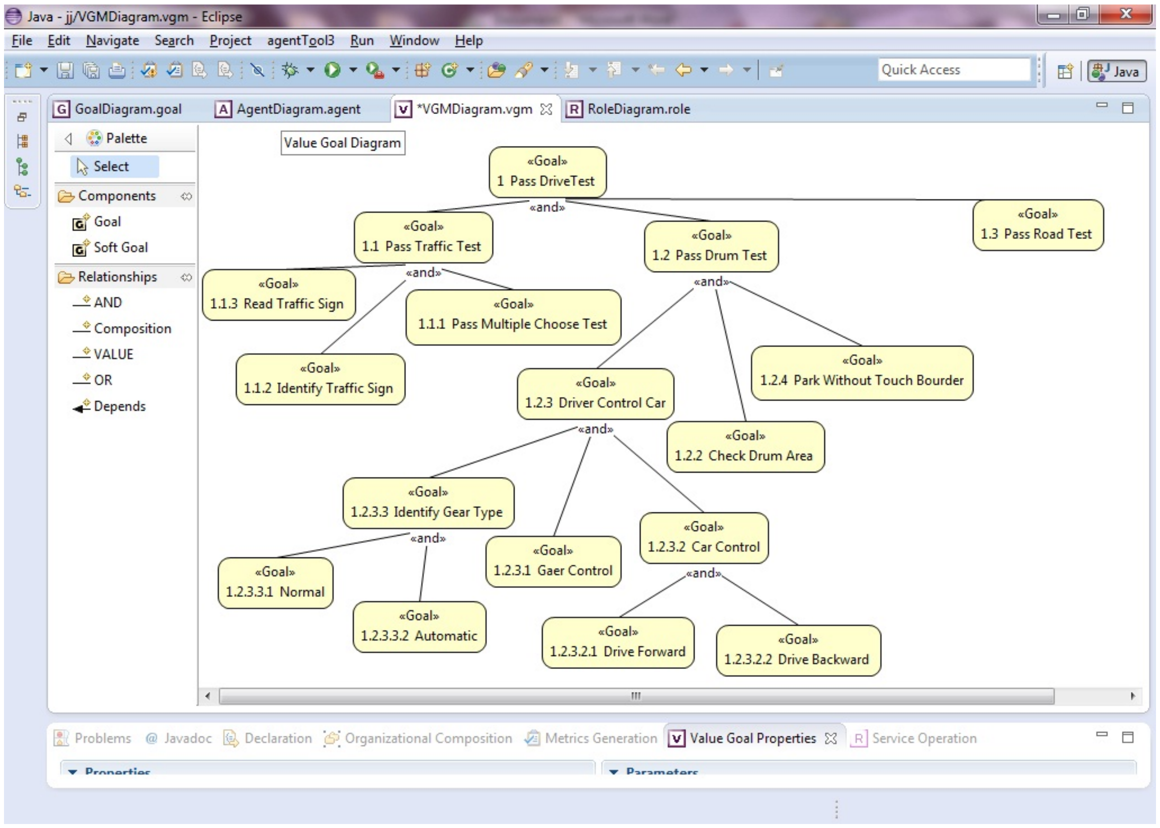

Java - jj/VGMDiagram.vgm - Eclipse
File Edit Navigate Search Project agentTool3 Run Window Help
Quick Access
Java
GoalDiagram.goal
AgentDiagram.agent
*VGMDiagram.vgm
RoleDiagram.role
Palette
Select
Components
Goal
Soft Goal
Relationships
AND
Composition
VALUE
OR
Depends
Value Goal Diagram
«Goal» 1 Pass DriveTest
«and»
«Goal» 1.1 Pass Traffic Test
«Goal» 1.2 Pass Drum Test
«Goal» 1.3 Pass Road Test
«Goal» 1.1.3 Read Traffic Sign
«and»
«Goal» 1.1.1 Pass Multiple Choose Test
«Goal» 1.1.2 Identify Traffic Sign
«and»
«Goal» 1.2.4 Park Without Touch Bourder
«Goal» 1.2.3 Driver Control Car
«and»
«Goal» 1.2.2 Check Drum Area
«Goal» 1.2.3.3 Identify Gear Type
«and»
«Goal» 1.2.3.1 Gaer Control
«Goal» 1.2.3.2 Car Control
«and»
«Goal» 1.2.3.3.1 Normal
«Goal» 1.2.3.3.2 Automatic
«Goal» 1.2.3.2.1 Drive Forward
«Goal» 1.2.3.2.2 Drive Backward
Problems
Javadoc
Declaration
Organizational Composition
Metrics Generation
Value Goal Properties
Service Operation
Properties
Parameters

Figure D.3: Goal Hierarchy (Sprint 2)

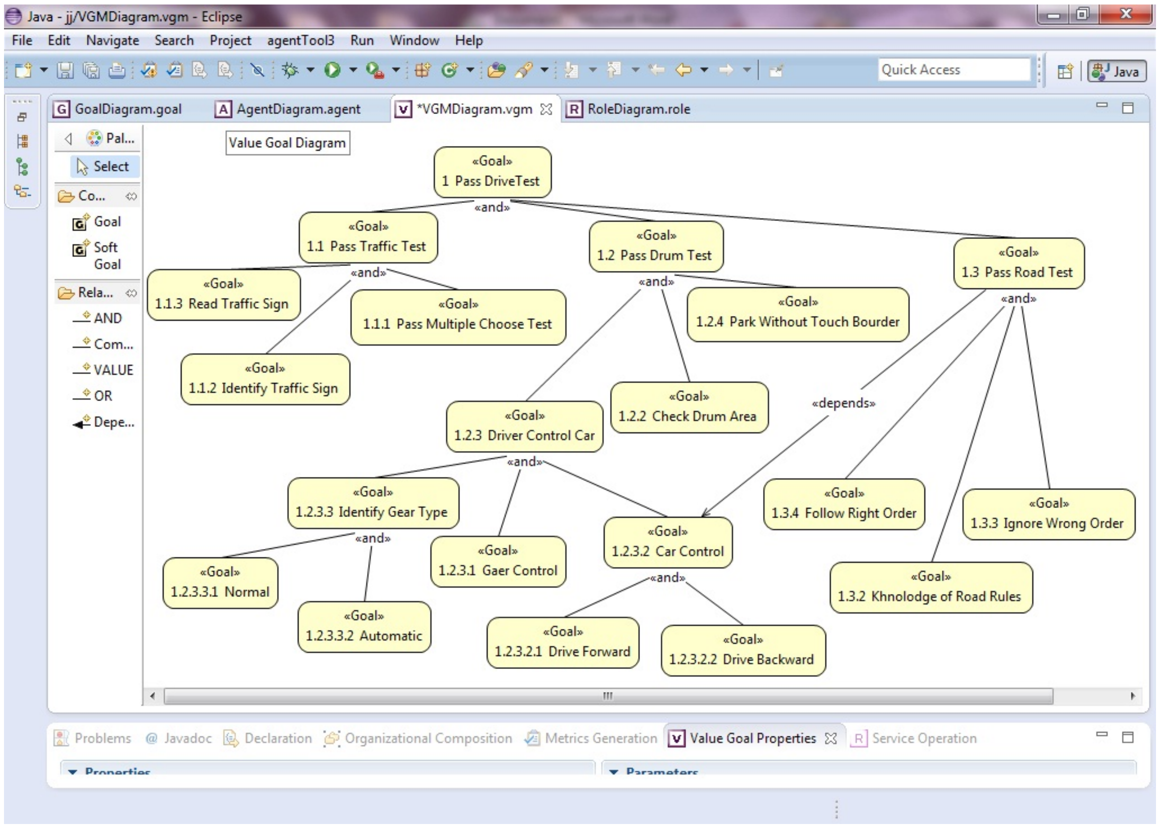

Value Goal Diagram
«Goal» 1 Pass DriveTest
«and»
«Goal» 1.1 Pass Traffic Test
«Goal» 1.2 Pass Drum Test
«Goal» 1.3 Pass Road Test
«Goal» 1.1.3 Read Traffic Sign
«Goal» 1.1.1 Pass Multiple Choose Test
«Goal» 1.1.2 Identify Traffic Sign
«Goal» 1.2.4 Park Without Touch Bourder
«Goal» 1.2.2 Check Drum Area
«Goal» 1.2.3 Driver Control Car
«depends»
«Goal» 1.2.3.3 Identify Gear Type
«Goal» 1.2.3.1 Gaer Control
«Goal» 1.2.3.2 Car Control
«Goal» 1.3.4 Follow Right Order
«Goal» 1.3.3 Ignore Wrong Order
«Goal» 1.3.2 Khnolodge of Road Rules
«Goal» 1.2.3.3.1 Normal
«Goal» 1.2.3.3.2 Automatic
«Goal» 1.2.3.2.1 Drive Forward
«Goal» 1.2.3.2.2 Drive Backward

Figure D.4: Goal Hierarchy (Sprint 3)

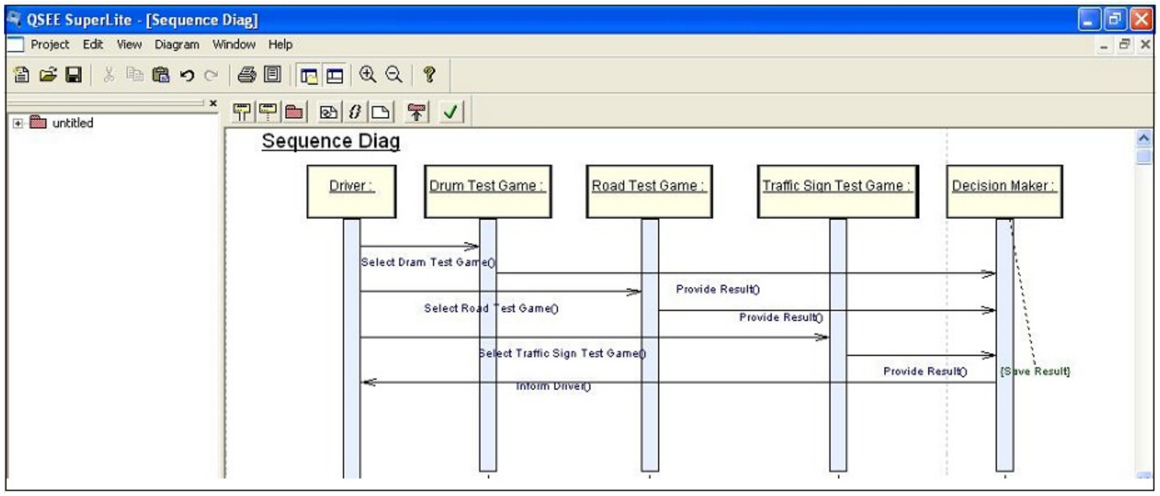


Figure D.5: Main Sequence Diagram

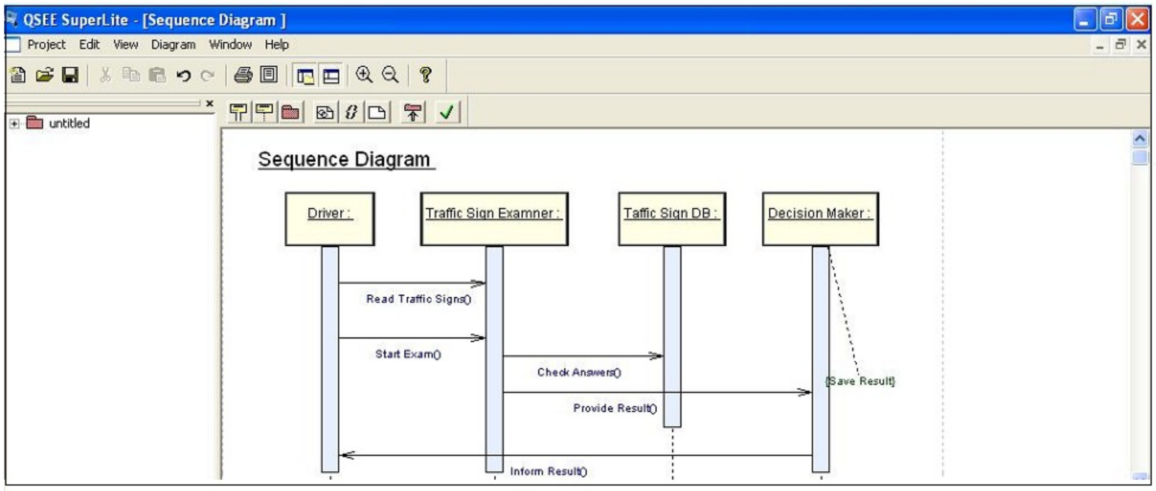


Figure D.6: Sequence Diagram (Sprint 1)

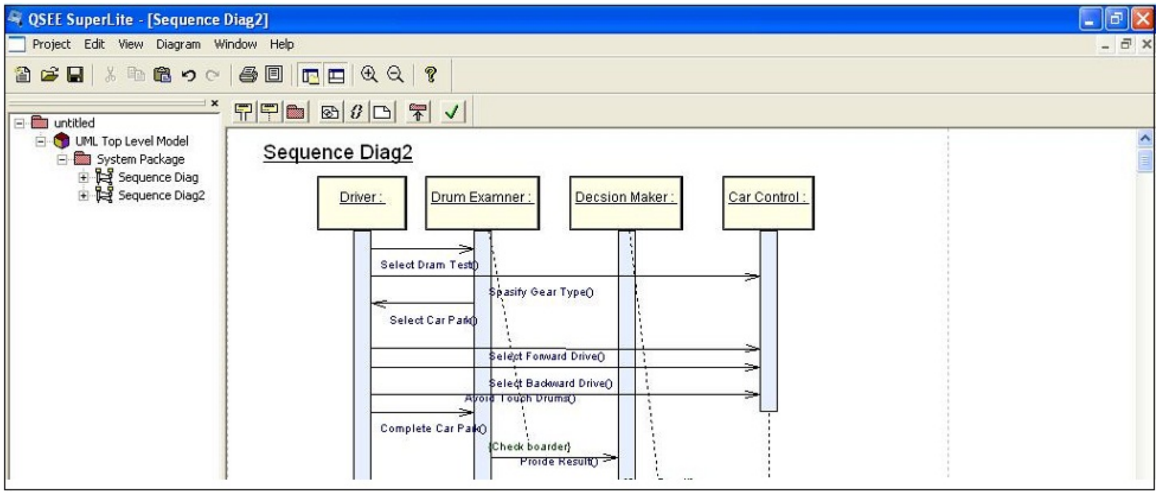


Figure D.7: Sequence Diagram (Sprint 2)

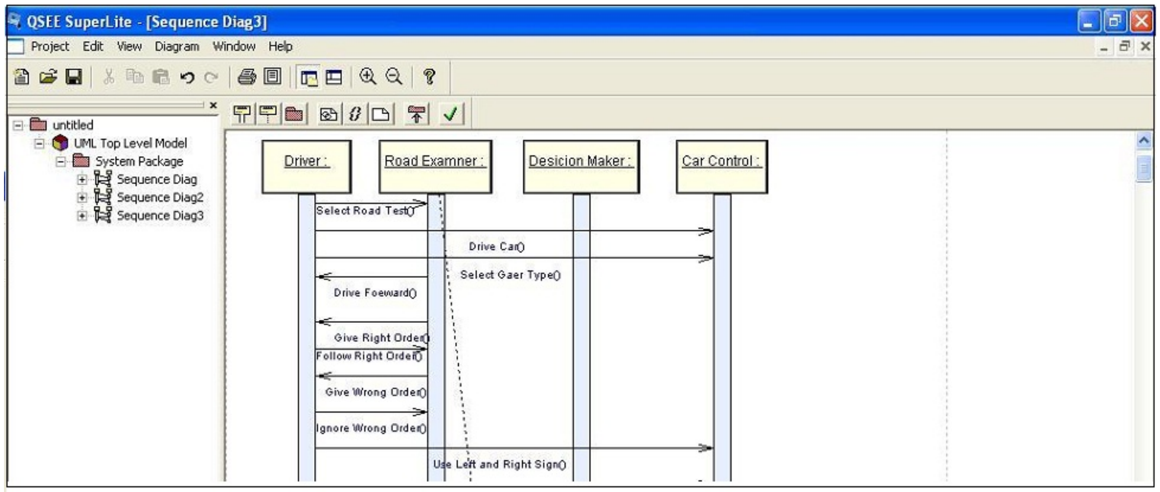


Figure D.8: Sequence Diagram (Sprint 3)

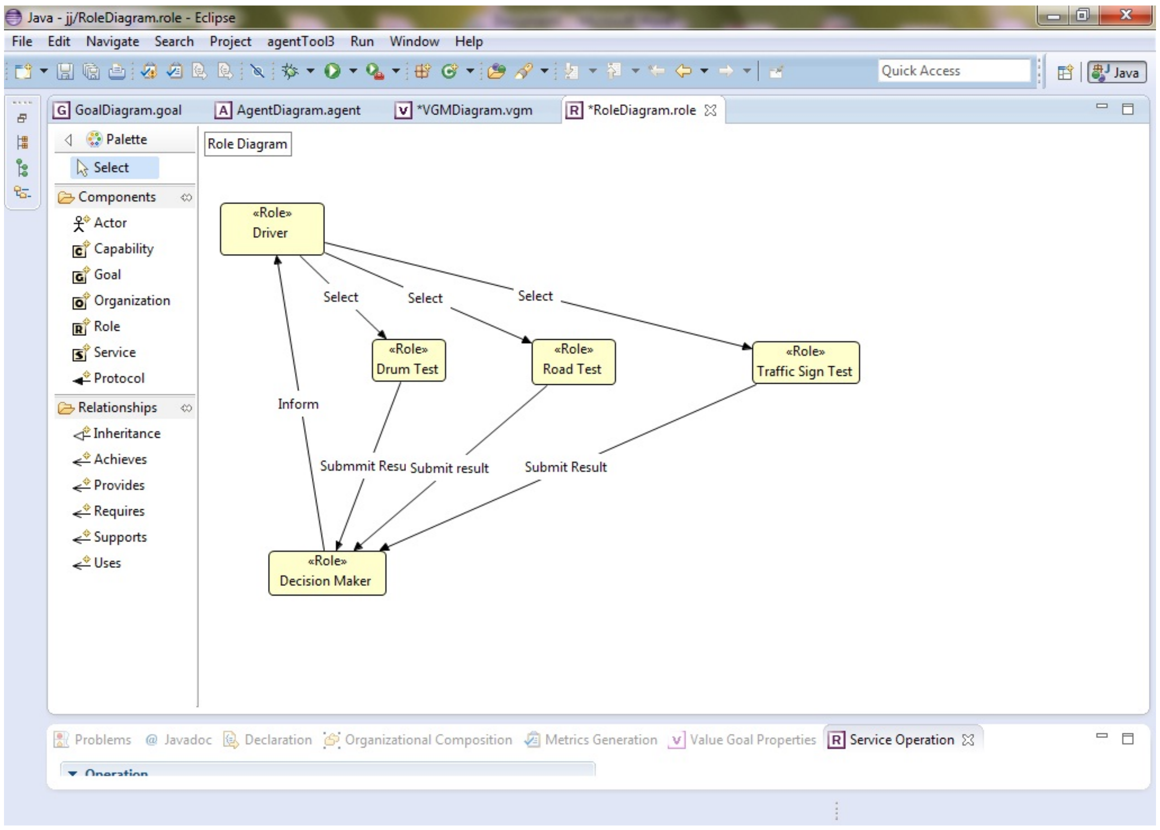


Figure D.9: Main Role Diagram

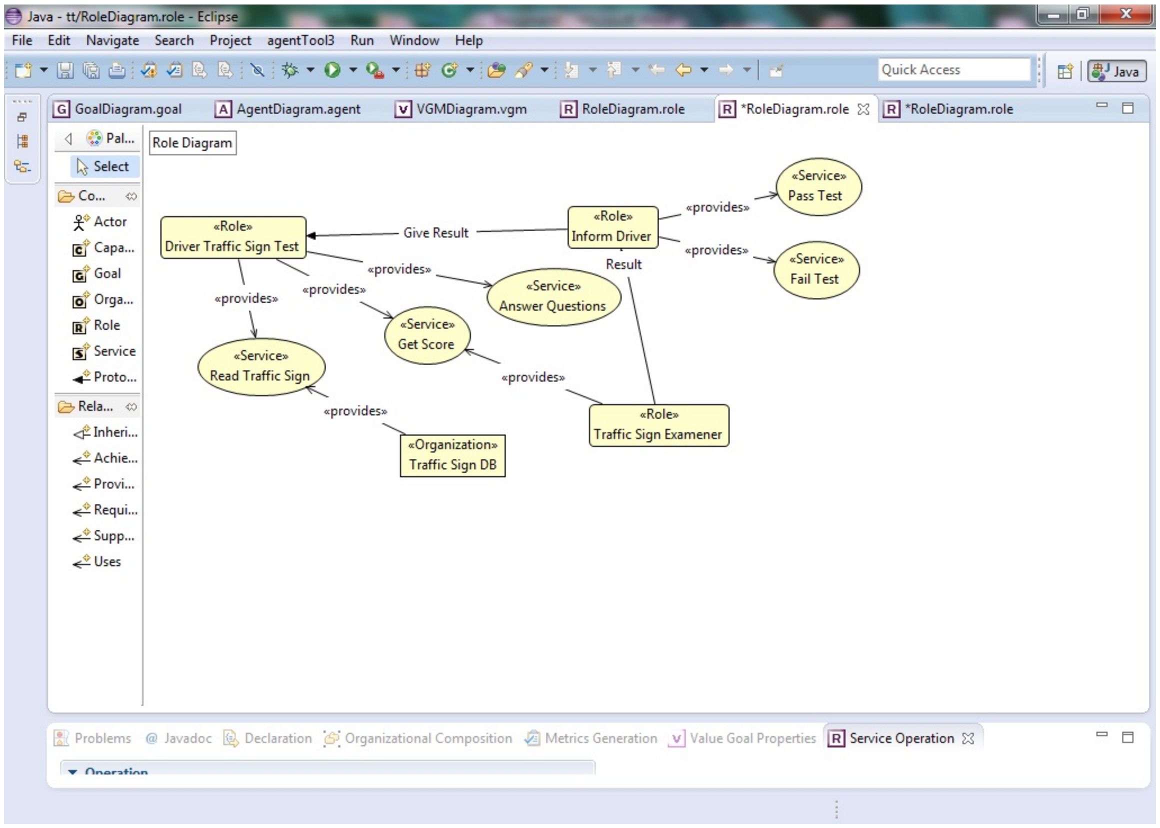


Figure D.10: Role Diagram (Sprint 1)

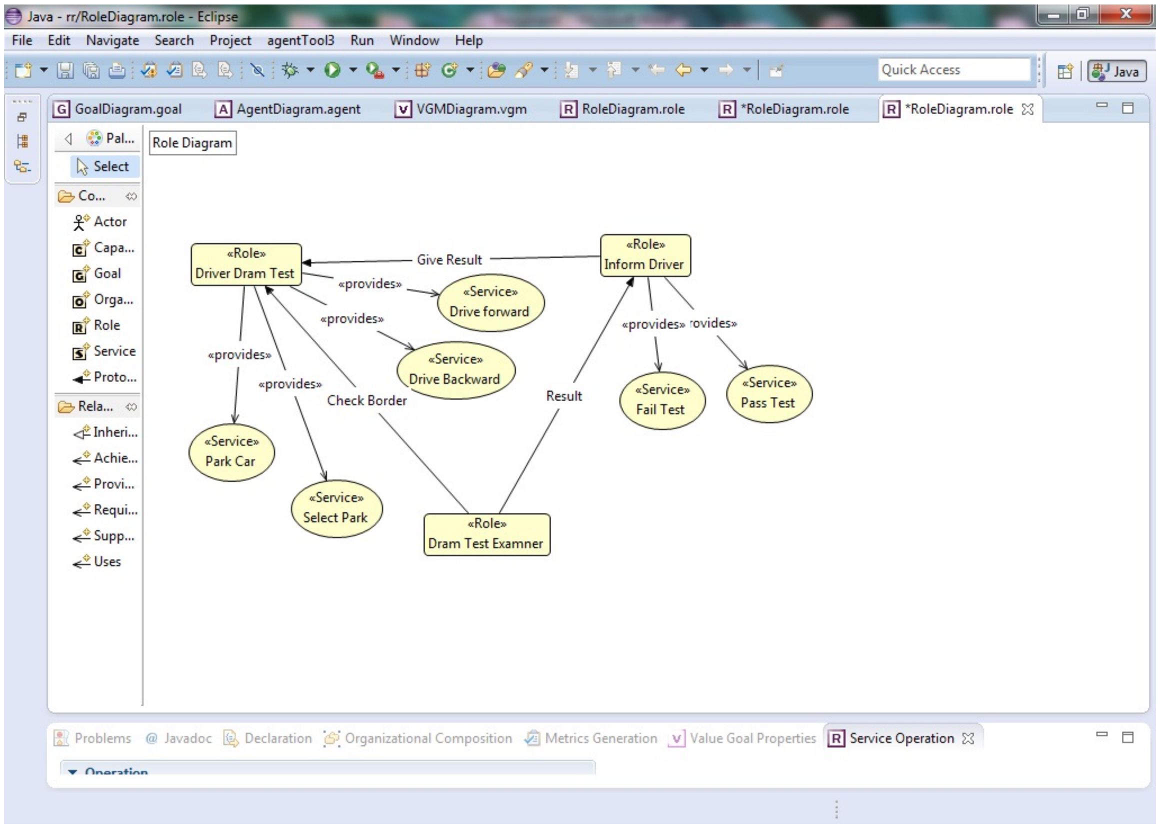


Figure D.11: Role Diagram (Sprint 2)

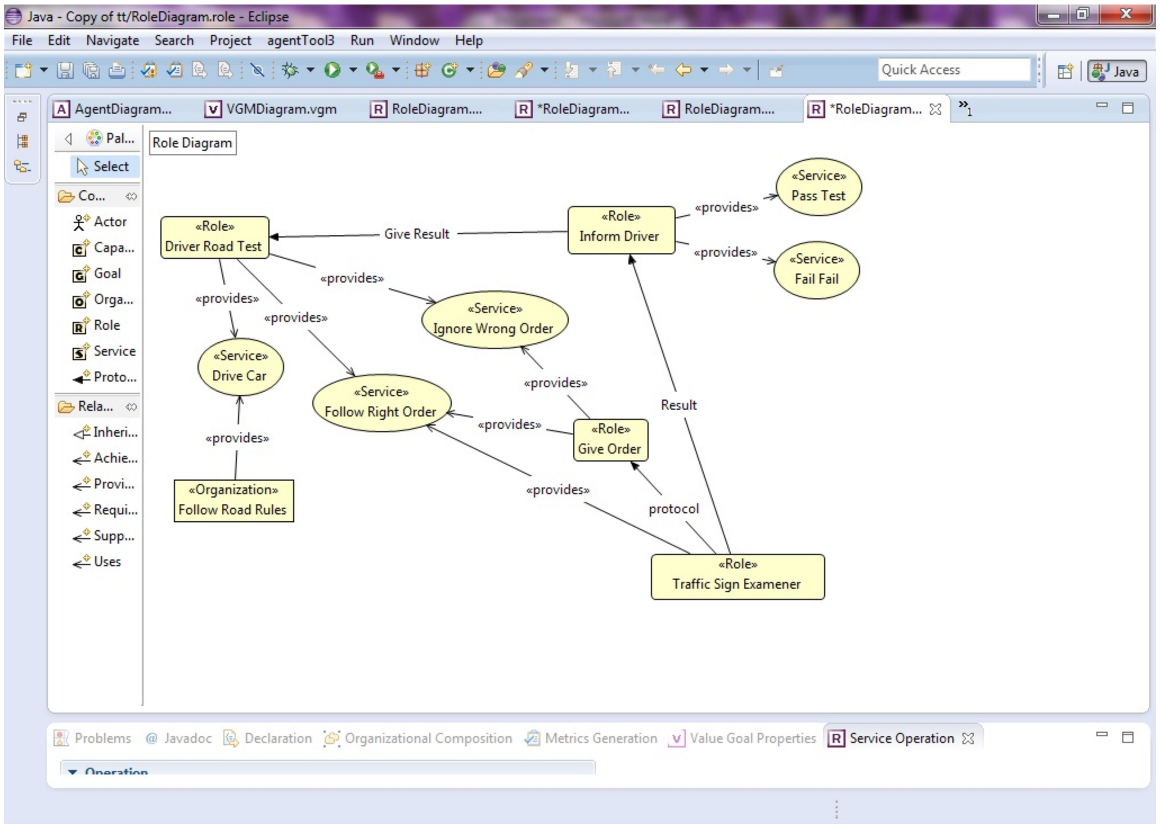


Figure D.12: Role Diagram (Sprint 3)

D.3 Design Phase Diagrams

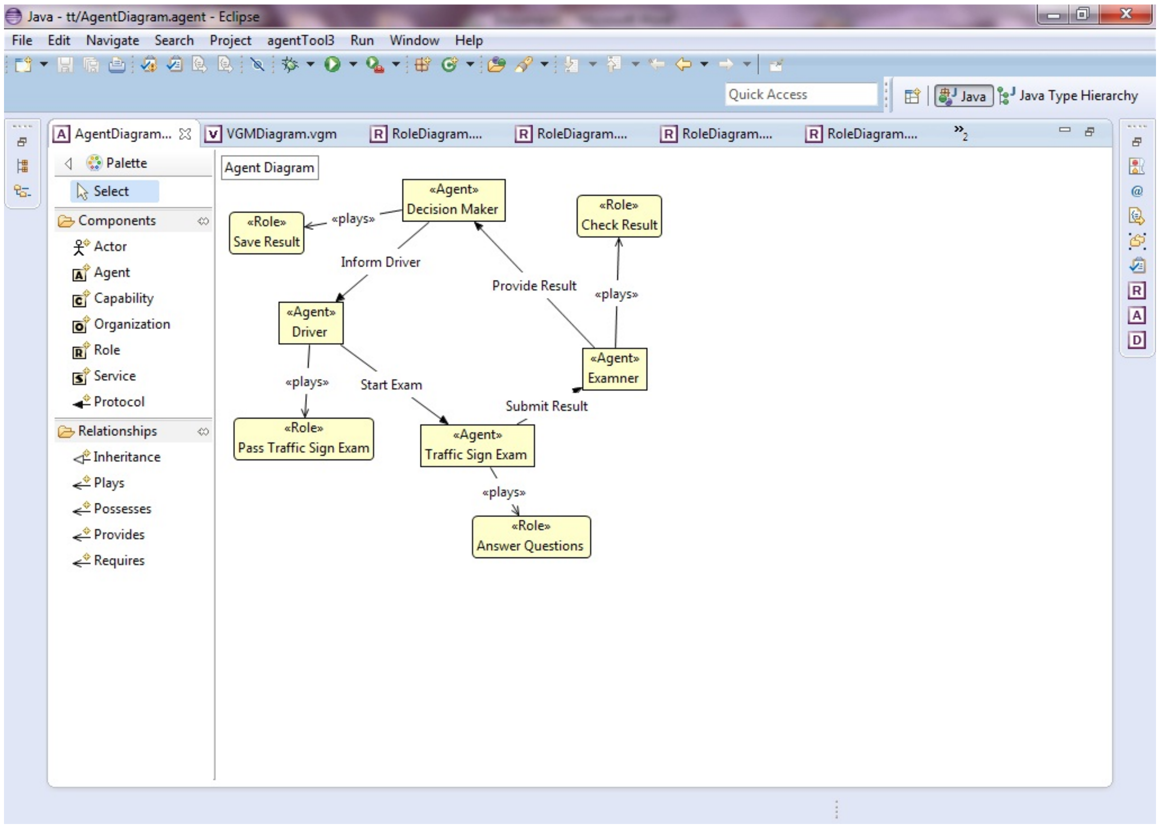


Figure D.13: Agent Diagram (Sprint 1)

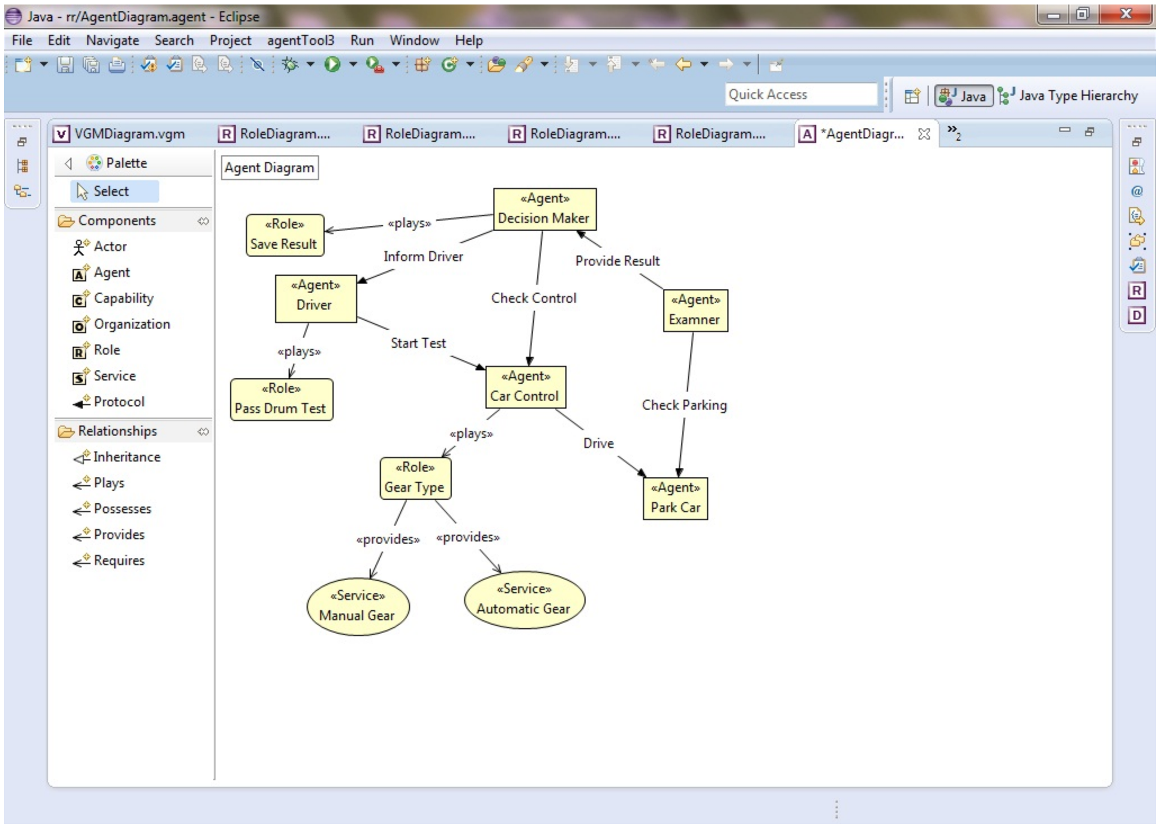


Figure D.14: Agent Diagram (Sprint 2)

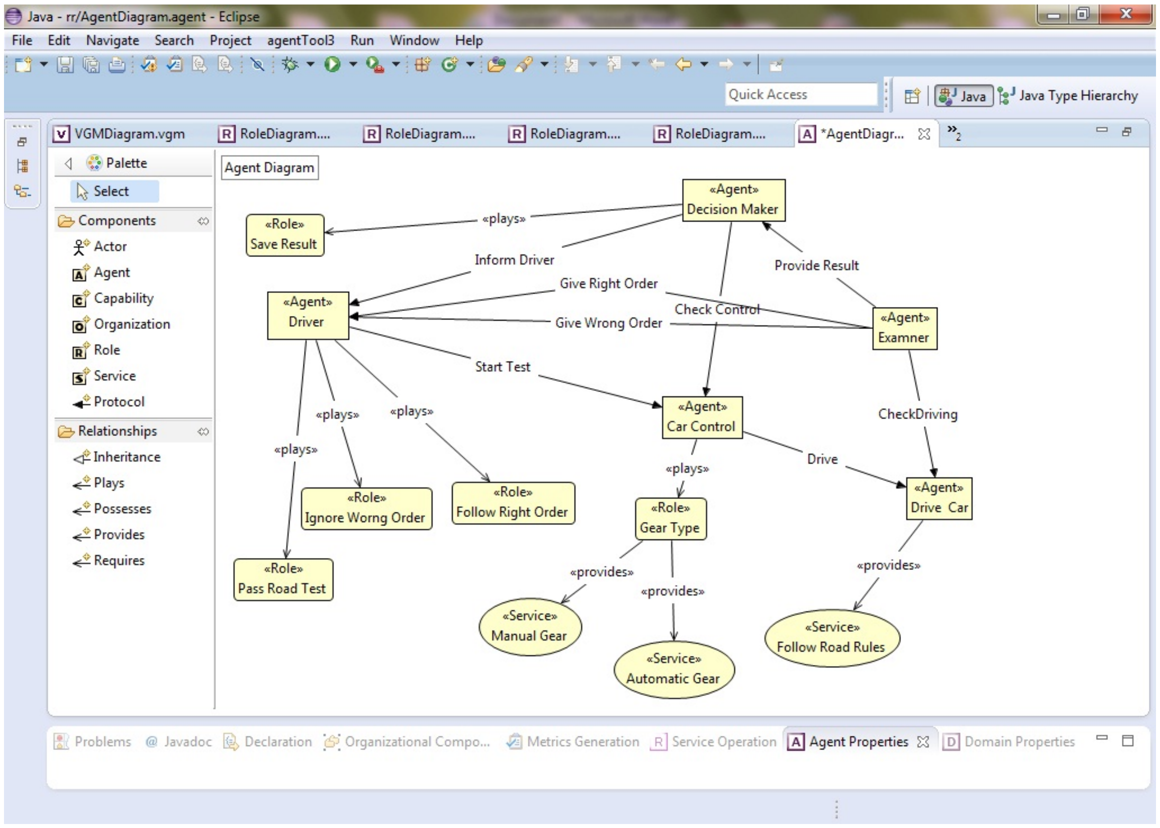


Figure D.15: Agent Diagram (Sprint 3)

D.4 Implementation Phase

D.4.1 Deployment Diagram

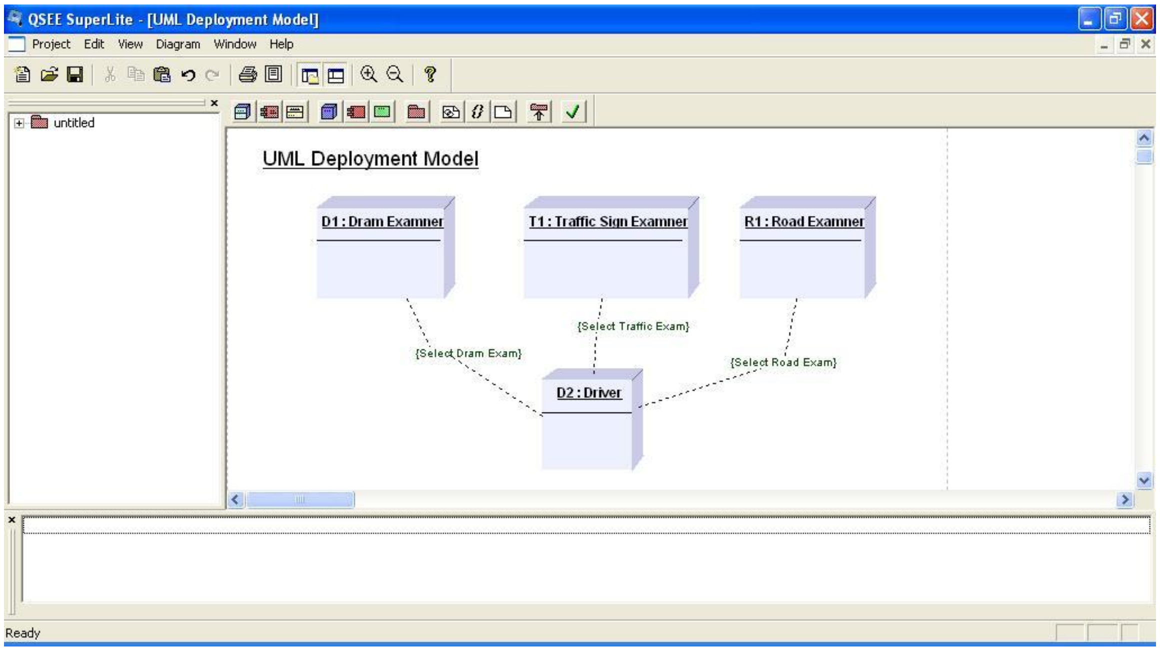


Figure D.16: Deployment Diagram

D.4.2 Game Layout and Graphics

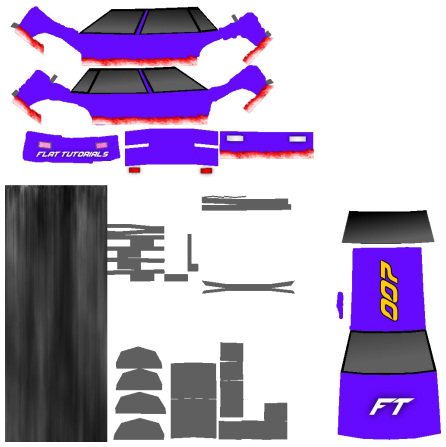
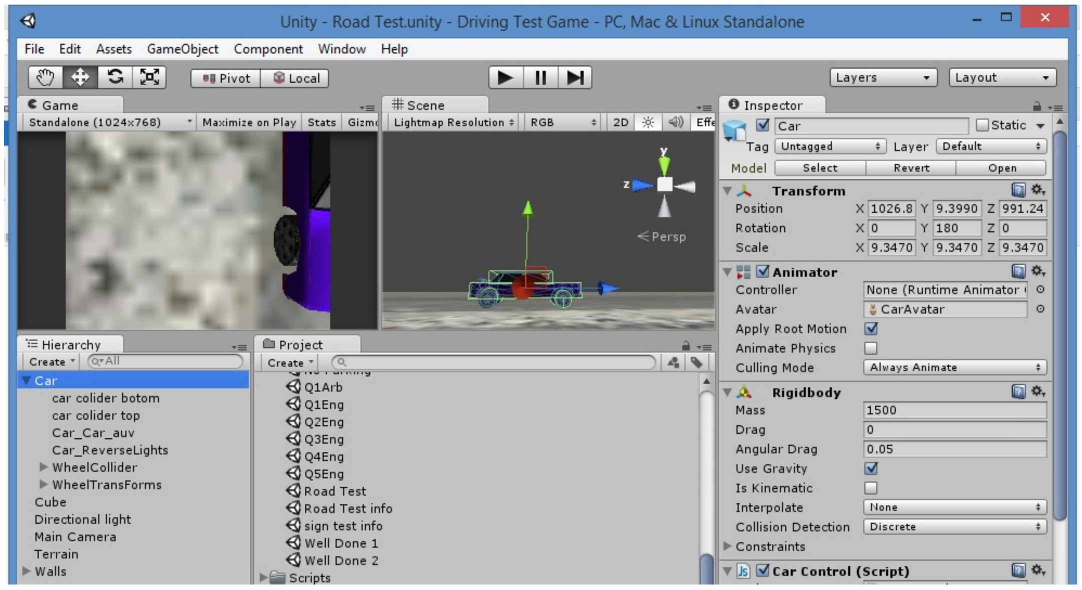


Figure D.17: Car Model



Unity - Road Test.unity - Driving Test Game - PC, Mac & Linux Standalone

Standalone (1024x768)

```
Car
  car colider botom
  car colider top
  Car_Car_auv
  Car_ReverseLights
  ▶ WheelCollider
  ▶ WheelTransForms
Cube
Directional light
Main Camera
Terrain
▶ Walls
```

```
Q1Arb
Q1Eng
Q2Eng
Q3Eng
Q4Eng
Q5Eng
Road Test
Road Test info
sign test info
Well Done 1
Well Done 2
▶ Scripts
```

```
Car                                   Static
Tag Untagged        Layer Default
Transform
Position    X 1026.8   Y 9.3990   Z 991.24
Rotation    X 0        Y 180      Z 0
Scale       X 9.3470   Y 9.3470   Z 9.3470
Animator
Controller          None (Runtime Animator
Avatar              CarAvatar
Apply Root Motion   ✓
Animate Physics
Culling Mode        Always Animate
Rigidbody
Mass                1500
Drag                0
Angular Drag        0.05
Use Gravity         ✓
Is Kinematic
Interpolate         None
Collision Detection Discrete
▶ Constraints
Car Control (Script)
```

Figure D.18: Unity Car Model

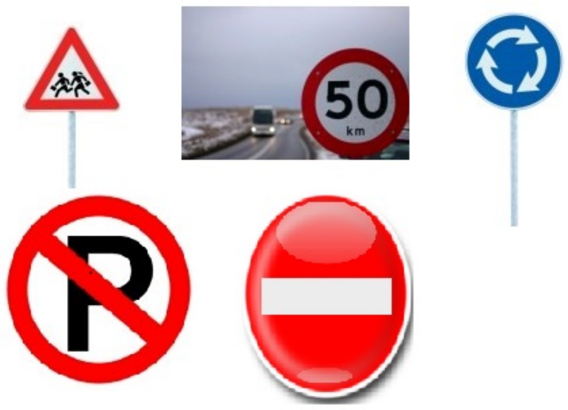


Figure D.19: Traffic Sign Images

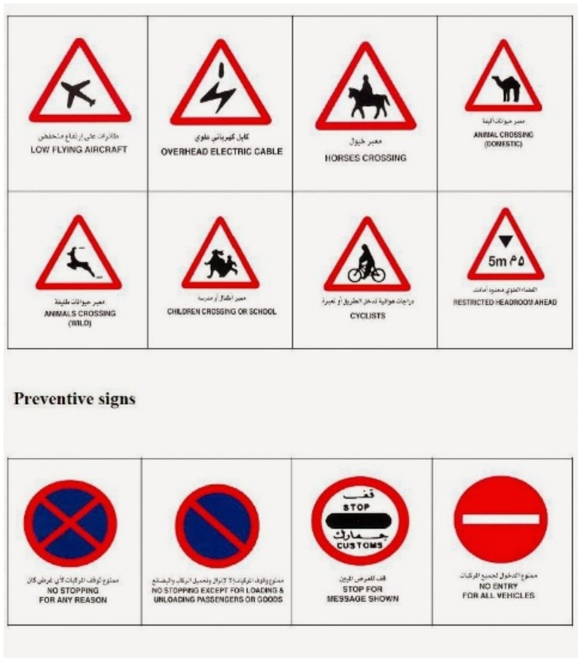

LOW FLYING AIRCRAFT
OVERHEAD ELECTRIC CABLE
HORSES CROSSING
ANIMAL CROSSING (DOMESTIC)
ANIMALS CROSSING (WILD)
CHILDREN CROSSING OR SCHOOL
CYCLISTS
5m
RESTRICTED HEADROOM AHEAD
Preventive signs
NO STOPPING FOR ANY REASON
NO STOPPING EXCEPT FOR LOADING & UNLOADING PASSENGERS OR GOODS
STOP
CUSTOMS
STOP FOR MESSAGE SHOWN
NO ENTRY FOR ALL VEHICLES

Figure D.20: Traffic Sign Sample

Figure D.21: Background Sample 1

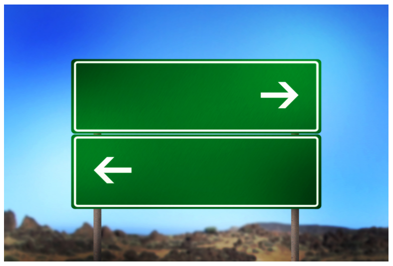


Figure D.22: Background Sample 2

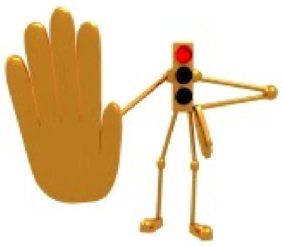


Figure D.23: Background Sample 3

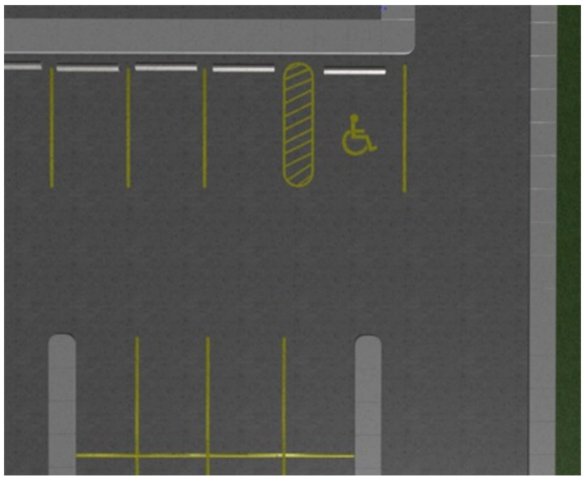
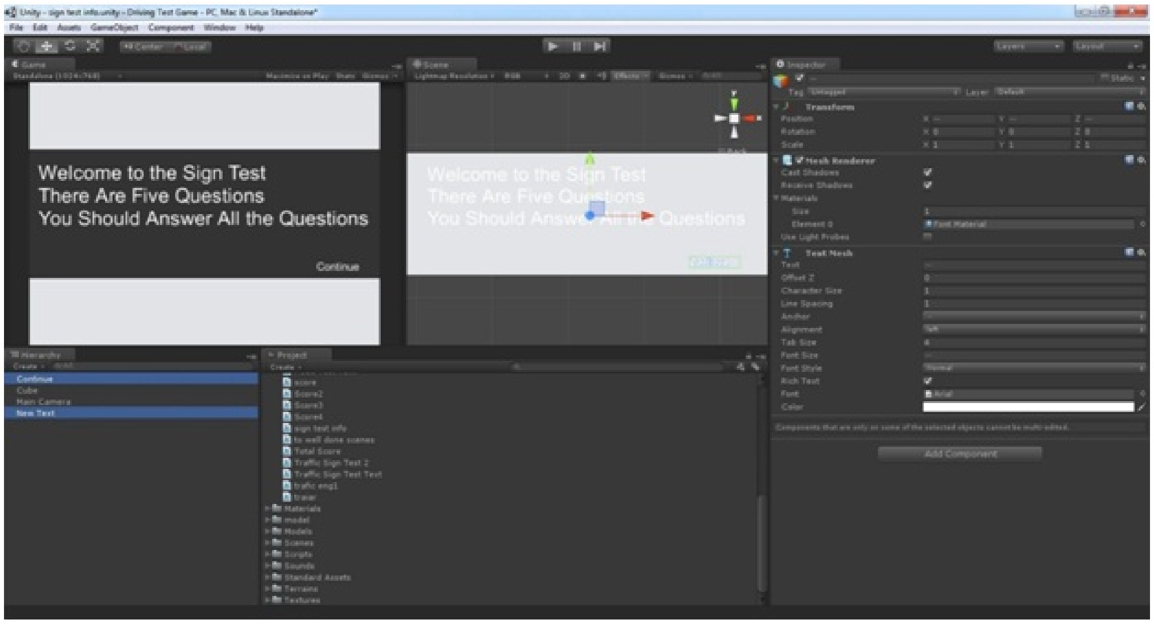


Figure D.24: Background Sample 4

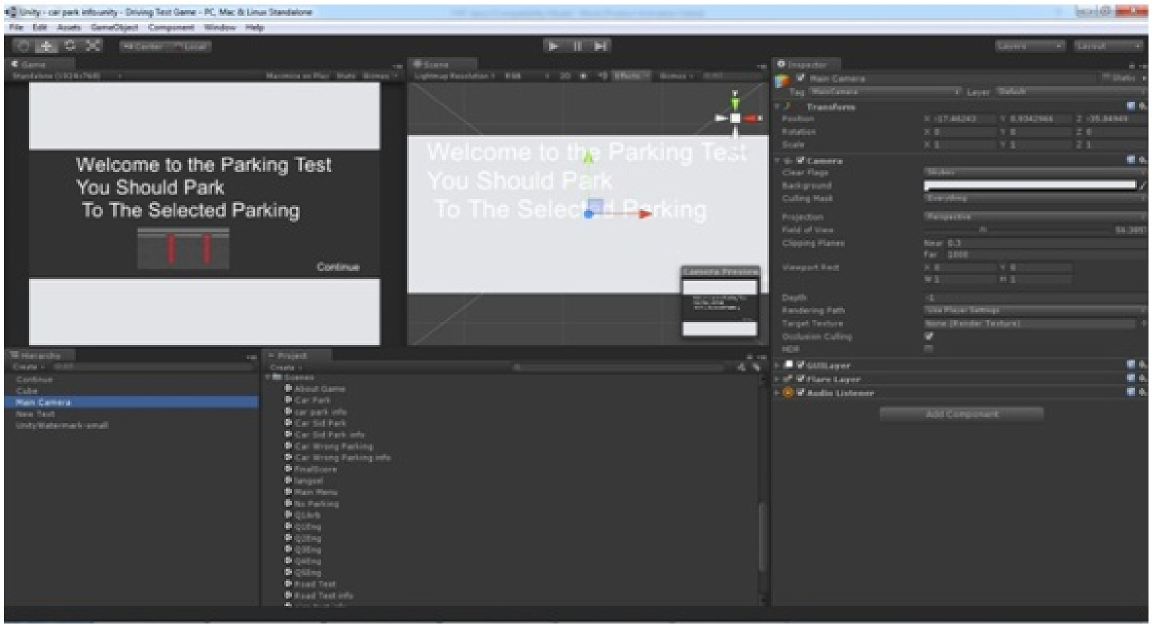
D.4.3 Communication Messages of the Drive Test Game


Unity - sign test info.unity - Driving Test Game - PC, Mac & Linux Standalone*

Welcome to the Sign Test
There Are Five Questions
You Should Answer All the Questions

Continue

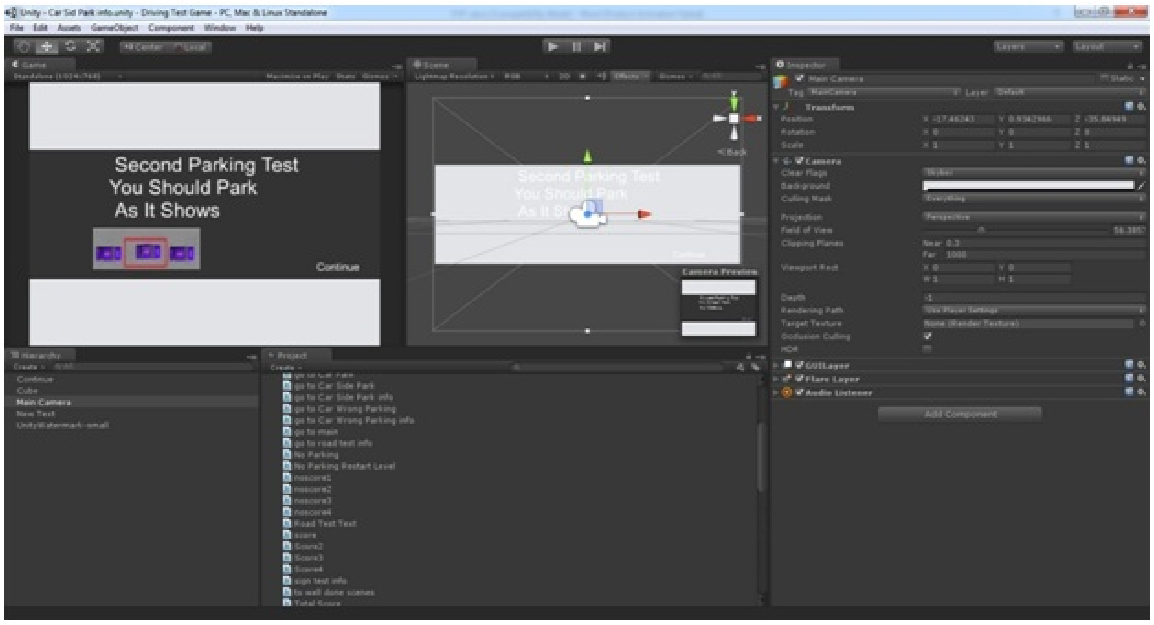
Figure D.25: Communication Message 1


Unity - car park info.unity - Driving Test Game - PC, Mac & Linux Standalone

Welcome to the Parking Test
You Should Park
To The Selected Parking

Continue

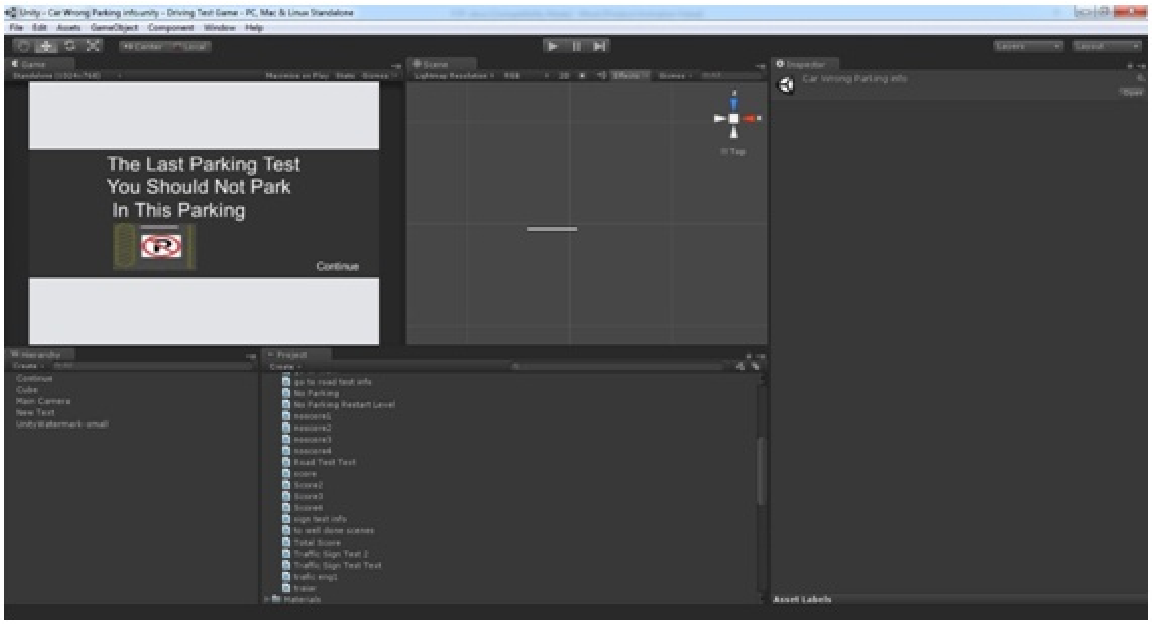
Figure D.26: Communication Message 2



Second Parking Test
You Should Park
As It Shows

Continue
Unity - Car Sid Park info.unity - Driving Test Game - PC, Mac & Linux Standalone
File Edit Assets GameObject Component Window Help
Game
Scene
Inspector
Hierarchy
Project
Main Camera
Transform
Camera
GUILayer
Flare Layer
Audio Listener
Add Component
Camera Preview

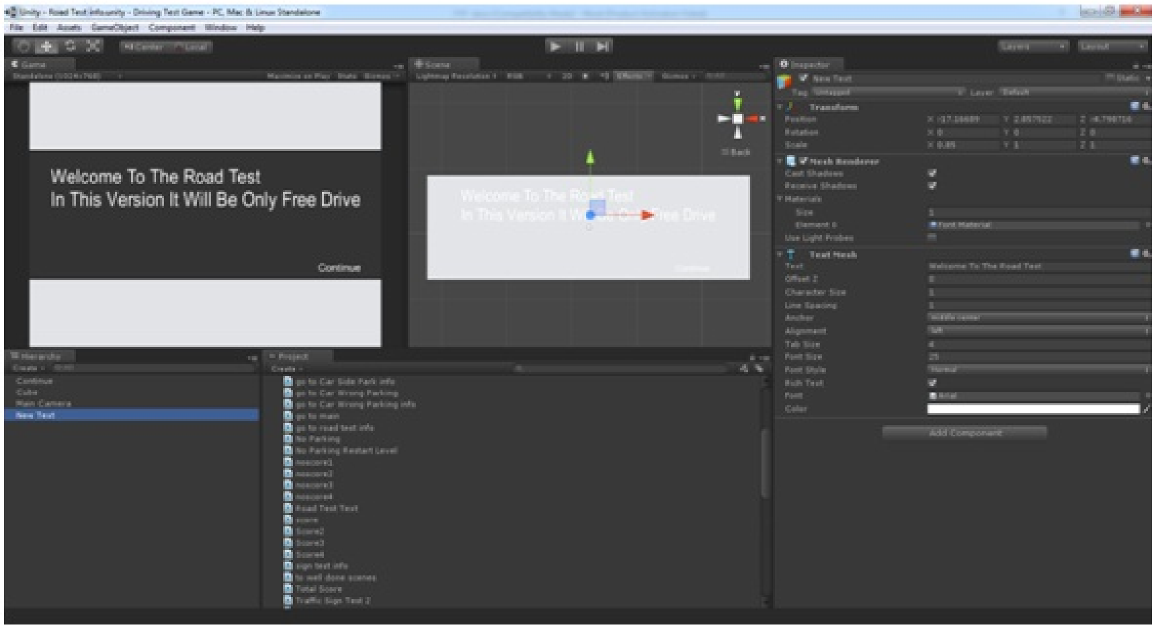
Figure D.27: Communication Message 3



The Last Parking Test
You Should Not Park
In This Parking

Continue
Unity - Car Wrong Parking info.unity - Driving Test Game - PC, Mac & Linux Standalone
File Edit Assets GameObject Component Window Help
Game
Scene
Inspector
Car Wrong Parking info
Hierarchy
Project
Asset Labels

Figure D.28: Communication Message 4

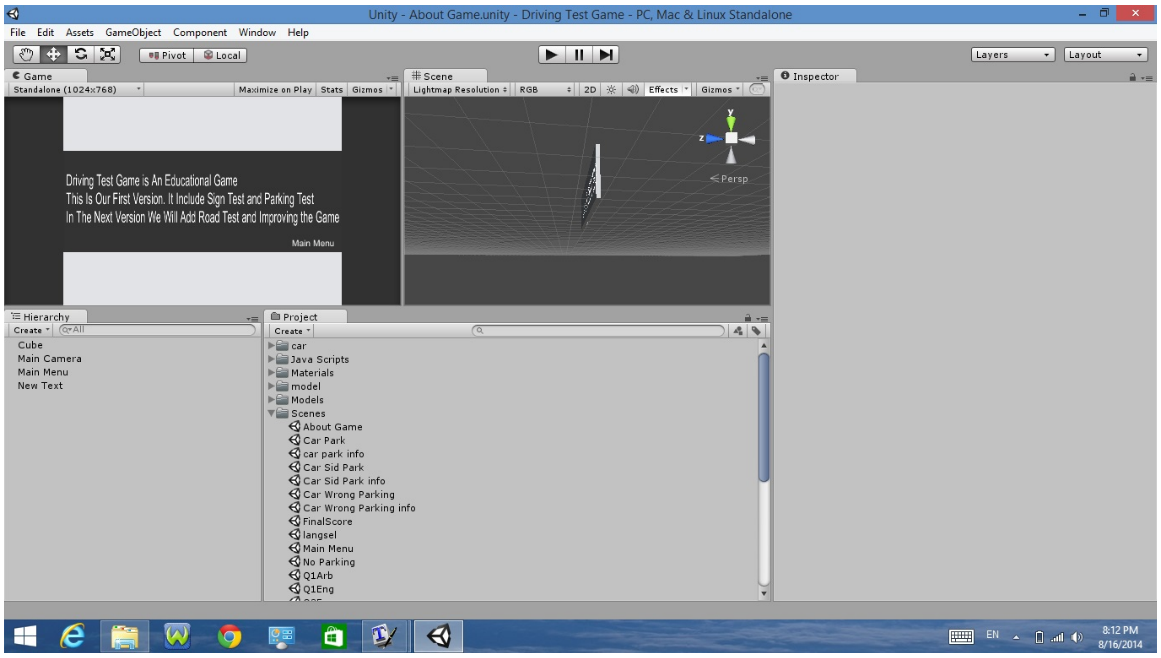


Welcome To The Road Test
In This Version It Will Be Only Free Drive

Continue

Text Mesh
Text: Welcome To The Road Test
Offset Z: 0
Character Size: 1
Line Spacing: 1
Anchor: middle center
Alignment: left
Tab Size: 4
Font Size: 25
Font Style: Normal
Rich Text: ✓
Font: Arial

Figure D.29: Communication Message 5



Driving Test Game is An Educational Game
This Is Our First Version. It Include Sign Test and Parking Test
In The Next Version We Will Add Road Test and Improving the Game

Main Menu

Figure D.30: Communication Message 6

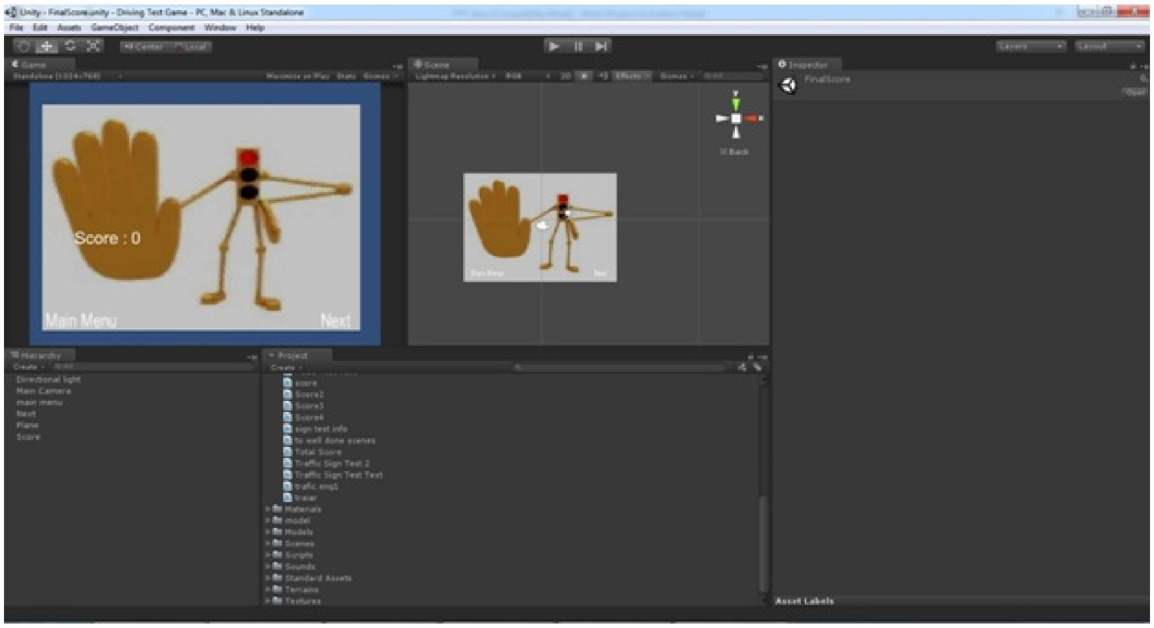


Figure D.31: Final Score

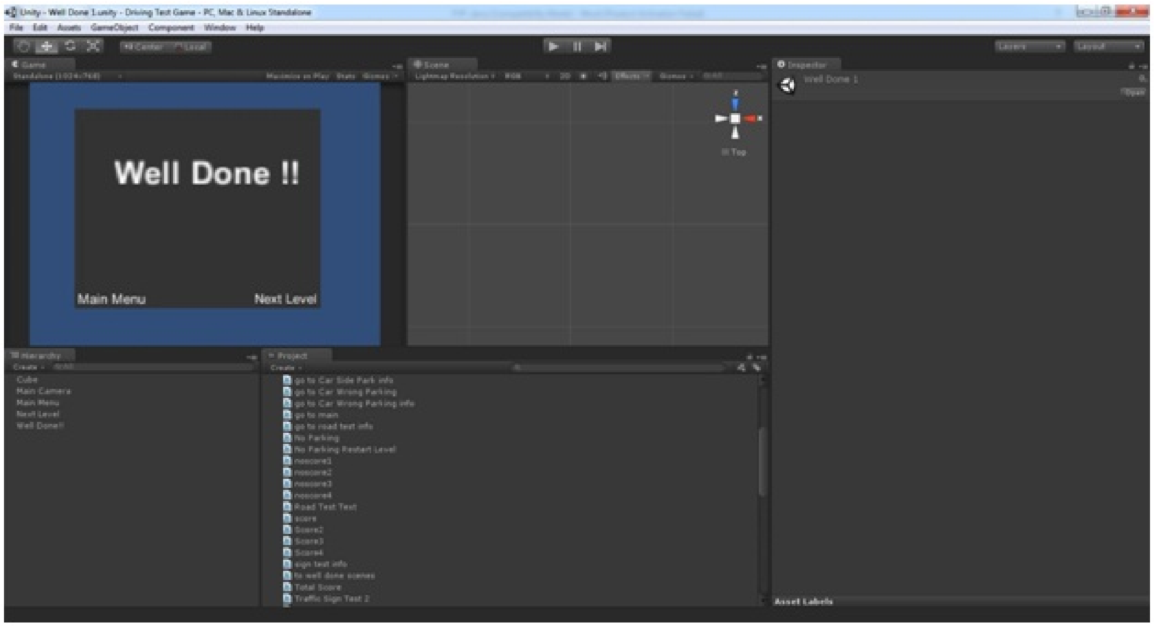
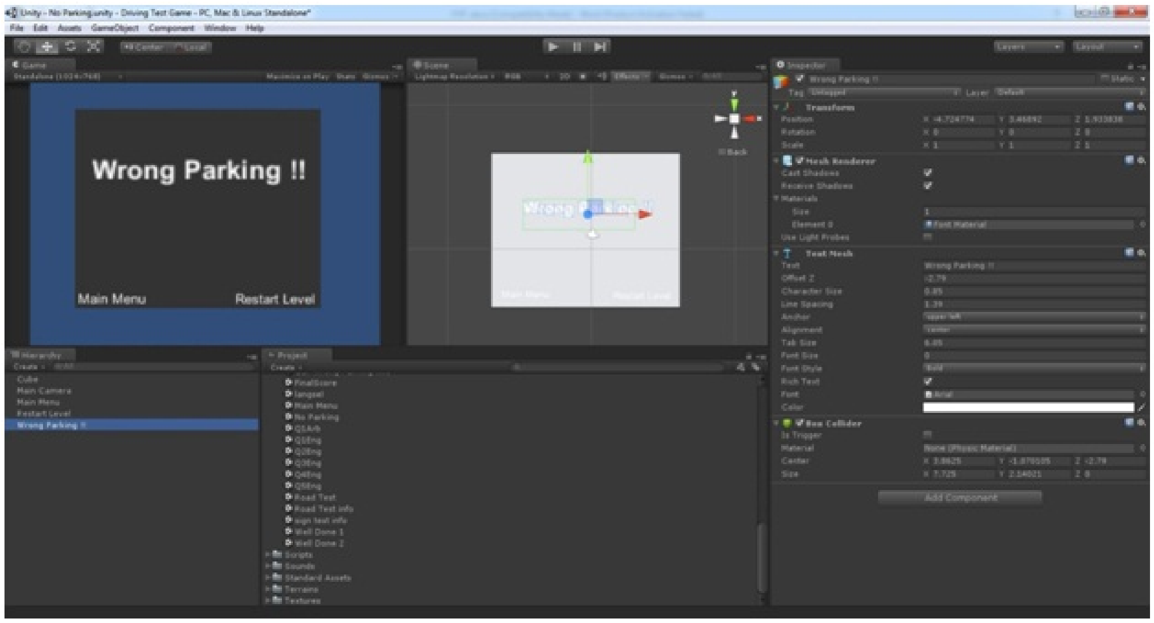


Figure D.32: Well Done Parking


Unity - No Parking.unity - Driving Test Game - PC, Mac & Linux Standalone*
File Edit Assets GameObject Component Window Help
Game
Scene
Inspector
Hierarchy
Project
Wrong Parking !!
Main Menu
Restart Level
Add Component

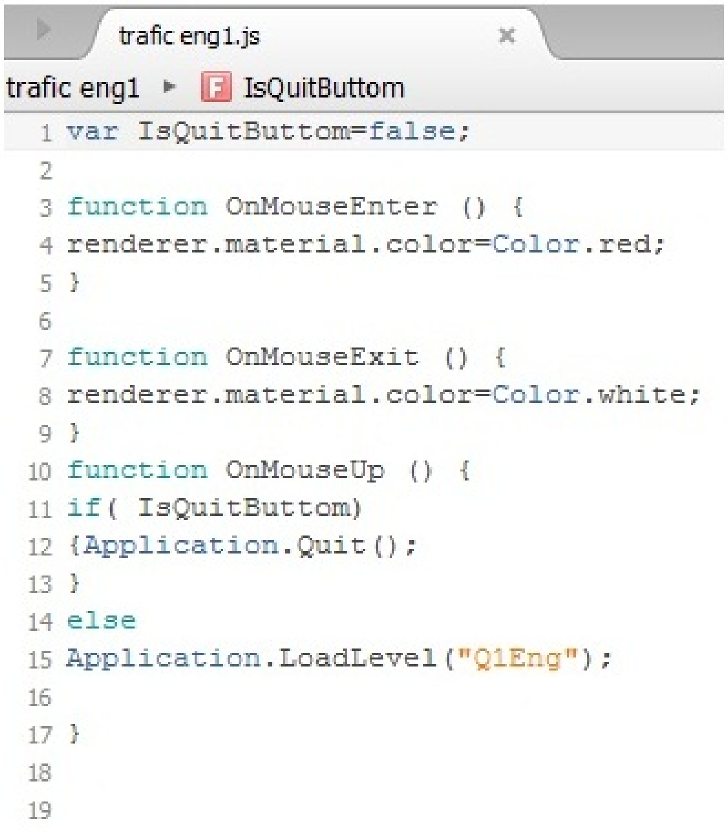
Hierarchy: Cube, Main Camera, Main Menu, Restart Level, Wrong Parking !!

Project: FinalScore, langsel, Main Menu, No Parking, Q1A/b, Q1Eng, Q2Eng, Q3Eng, Q4Eng, Q5Eng, Road Test, Road Test info, sign test info, Well Done 1, Well Done 2, Scripts, Sounds, Standard Assets, Terrains, Textures

```
Wrong Parking !!                     Static
Tag  Untagged          Layer  Default
Transform
  Position   X -4.724774   Y 3.46692     Z 1.933838
  Rotation   X 0           Y 0           Z 0
  Scale      X 1           Y 1           Z 1
Mesh Renderer
  Cast Shadows       ✓
  Receive Shadows    ✓
  Materials
    Size             1
    Element 0        Font Material
  Use Light Probes
Text Mesh
  Text               Wrong Parking !!
  Offset Z           -2.79
  Character Size     0.85
  Line Spacing       1.39
  Anchor             upper left
  Alignment          center
  Tab Size           6.05
  Font Size          0
  Font Style         Bold
  Rich Text          ✓
  Font               Arial
  Color
Box Collider
  Is Trigger
  Material   None (Physic Material)
  Center     X 3.8625      Y -1.070105   Z -2.79
  Size       X 7.725       Y 2.14021     Z 0
```

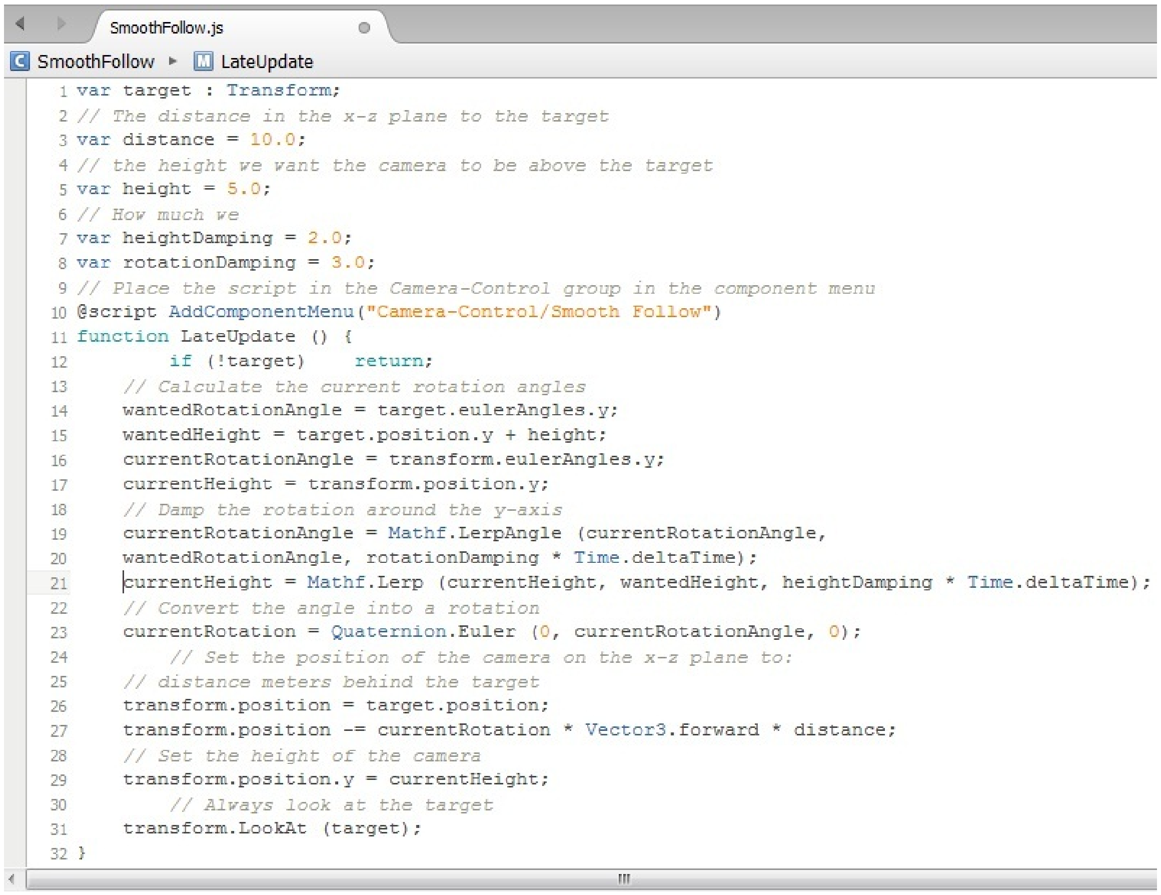
Figure D.33: Wrong Parking

D.4.4 Java Script Code of the Drive Test Game



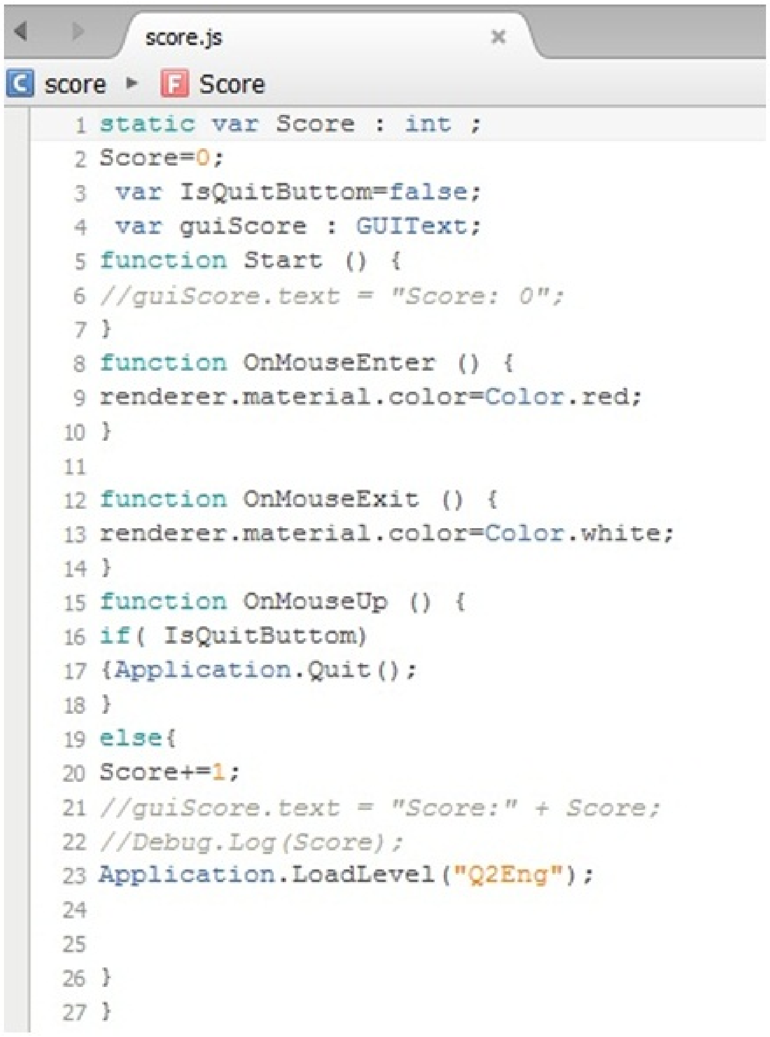
```
var IsQuitButtom=false;

function OnMouseEnter () {
renderer.material.color=Color.red;
}

function OnMouseExit () {
renderer.material.color=Color.white;
}
function OnMouseUp () {
if( IsQuitButtom)
{Application.Quit();
}
else
Application.LoadLevel("Q1Eng");

}
```

Figure D.34: Java Script Code for movement to Scene "Q1Eng"



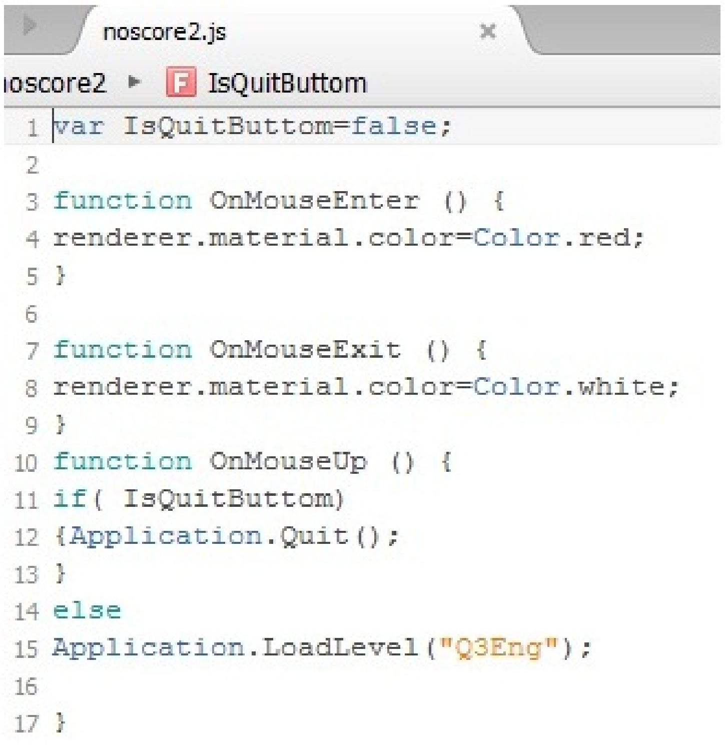
```
var target : Transform;
// The distance in the x-z plane to the target
var distance = 10.0;
// the height we want the camera to be above the target
var height = 5.0;
// How much we
var heightDamping = 2.0;
var rotationDamping = 3.0;
// Place the script in the Camera-Control group in the component menu
@script AddComponentMenu("Camera-Control/Smooth Follow")
function LateUpdate () {
        if (!target)    return;
    // Calculate the current rotation angles
    wantedRotationAngle = target.eulerAngles.y;
    wantedHeight = target.position.y + height;
    currentRotationAngle = transform.eulerAngles.y;
    currentHeight = transform.position.y;
    // Damp the rotation around the y-axis
    currentRotationAngle = Mathf.LerpAngle (currentRotationAngle,
    wantedRotationAngle, rotationDamping * Time.deltaTime);
    currentHeight = Mathf.Lerp (currentHeight, wantedHeight, heightDamping * Time.deltaTime);
    // Convert the angle into a rotation
    currentRotation = Quaternion.Euler (0, currentRotationAngle, 0);
        // Set the position of the camera on the x-z plane to:
    // distance meters behind the target
    transform.position = target.position;
    transform.position -= currentRotation * Vector3.forward * distance;
    // Set the height of the camera
    transform.position.y = currentHeight;
        // Always look at the target
    transform.LookAt (target);
}
```

Figure D.35: Java Script Code for Second Camera Move Follow



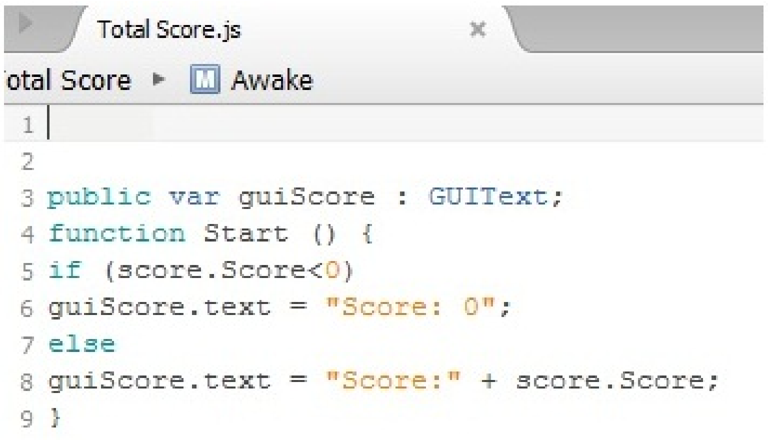
```
static var Score : int ;
Score=0;
 var IsQuitButtom=false;
 var guiScore : GUIText;
function Start () {
//guiScore.text = "Score: 0";
}
function OnMouseEnter () {
renderer.material.color=Color.red;
}

function OnMouseExit () {
renderer.material.color=Color.white;
}
function OnMouseUp () {
if( IsQuitButtom)
{Application.Quit();
}
else{
Score+=1;
//guiScore.text = "Score:" + Score;
//Debug.Log(Score);
Application.LoadLevel("Q2Eng");


}
}
```

Figure D.36: Java Script Code for right Selection of Traffic Sign Test



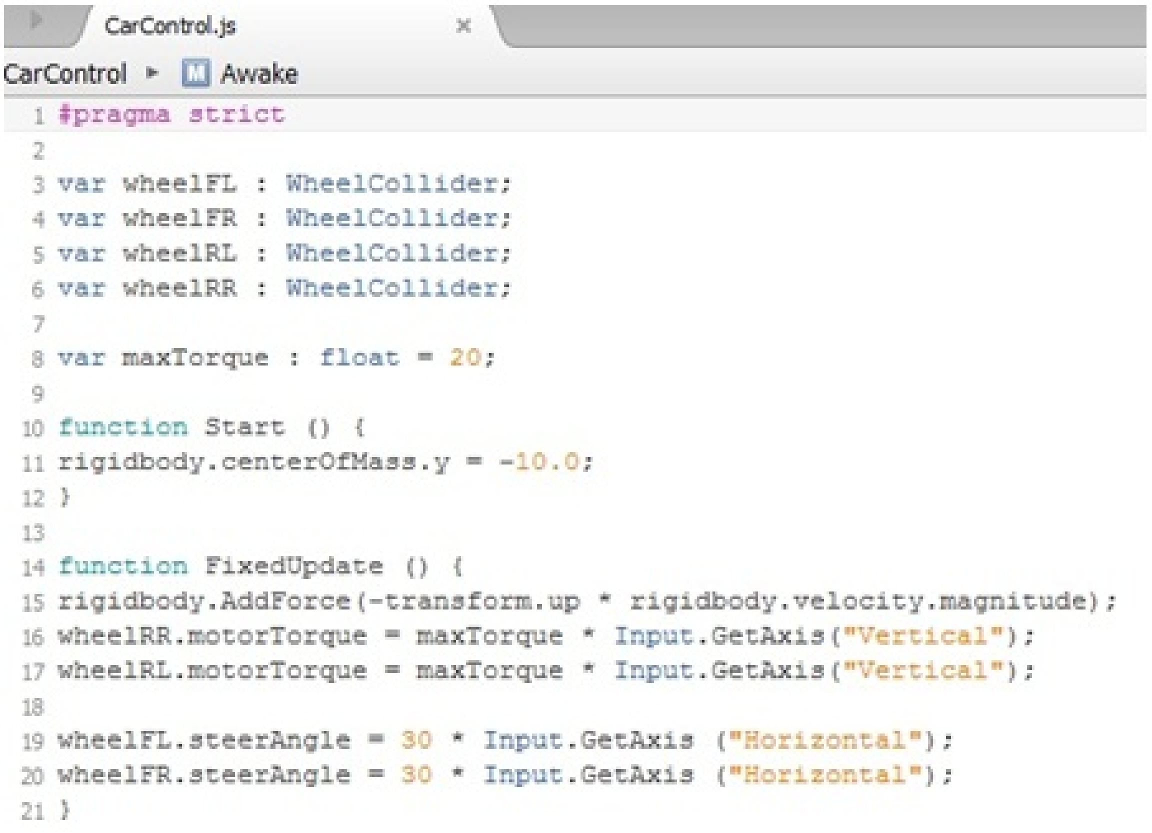
```
var IsQuitButtom=false;

function OnMouseEnter () {
renderer.material.color=Color.red;
}

function OnMouseExit () {
renderer.material.color=Color.white;
}
function OnMouseUp () {
if( IsQuitButtom)
{Application.Quit();
}
else
Application.LoadLevel("Q3Eng");

}
```

Figure D.37: Java Script Code for Wrong Selection of Traffic Sign Test



```


public var guiScore : GUIText;
function Start () {
if (score.Score<0)
guiScore.text = "Score: 0";
else
guiScore.text = "Score:" + score.Score;
}
```

Figure D.38: Java Script Code for Final Score of Traffic Sign Test



```
#pragma strict

var wheelFL : WheelCollider;
var wheelFR : WheelCollider;
var wheelRL : WheelCollider;
var wheelRR : WheelCollider;

var maxTorque : float = 20;

function Start () {
rigidbody.centerOfMass.y = -10.0;
}

function FixedUpdate () {
rigidbody.AddForce(-transform.up * rigidbody.velocity.magnitude);
wheelRR.motorTorque = maxTorque * Input.GetAxis("Vertical");
wheelRL.motorTorque = maxTorque * Input.GetAxis("Vertical");

wheelFL.steerAngle = 30 * Input.GetAxis ("Horizontal");
wheelFR.steerAngle = 30 * Input.GetAxis ("Horizontal");
}
```

Figure D.39: Java Script Code for Car Control

D.4.5 Screen Shot of the Drive Test Game

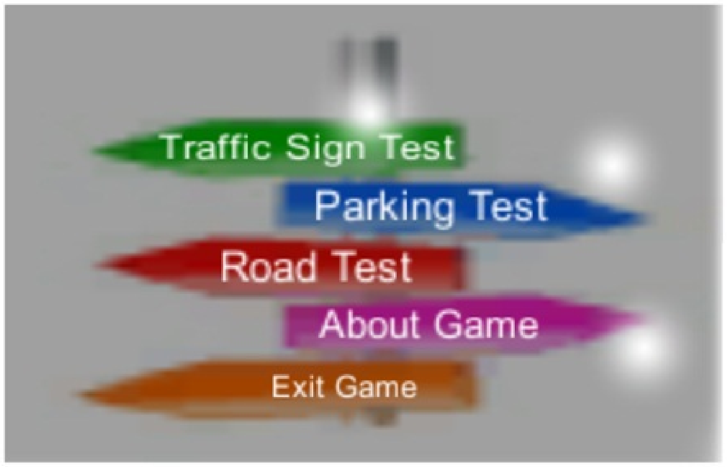


Figure D.40: Main Screen of Drive Test Game

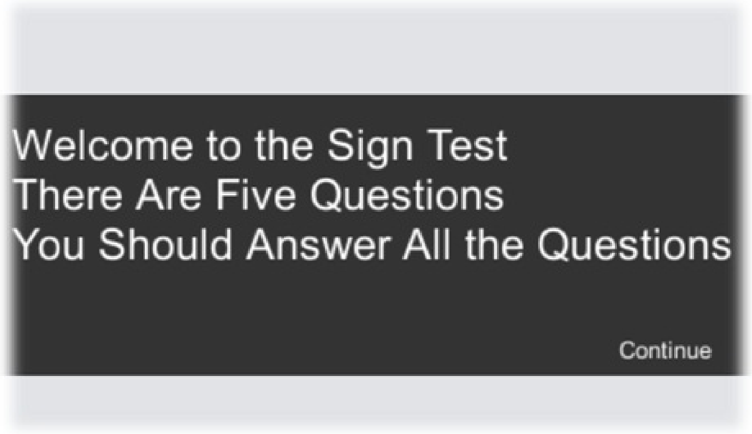

Welcome to the Sign Test
There Are Five Questions
You Should Answer All the Questions

Figure D.41: Description of Traffic Sign Test

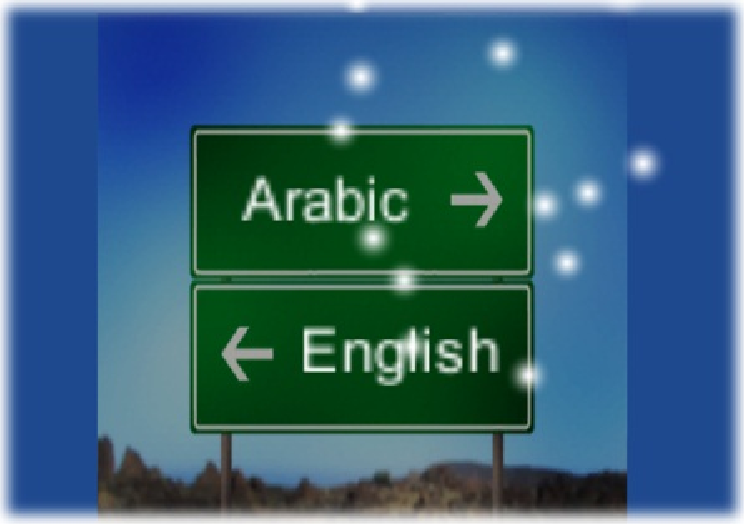


Figure D.42: Game Language Selection

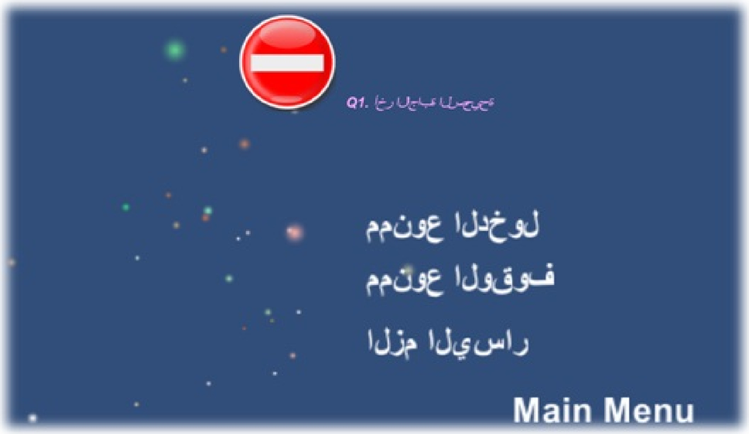
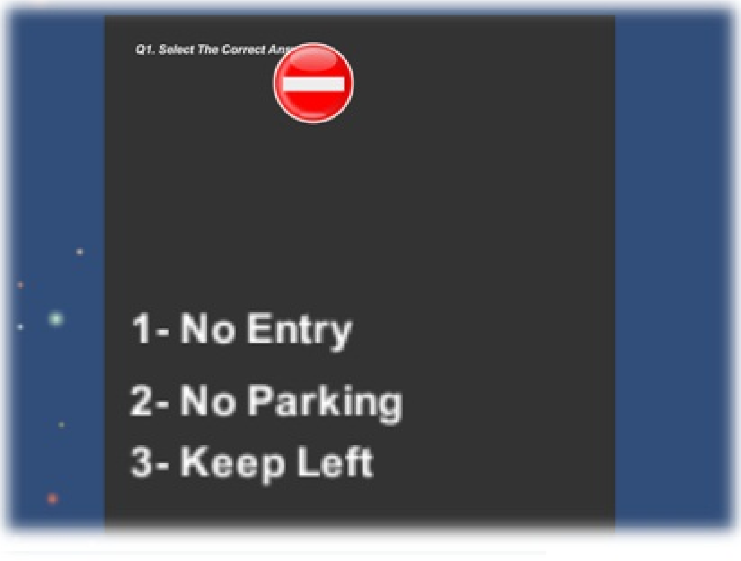


Figure D.43: Arabic Language Test



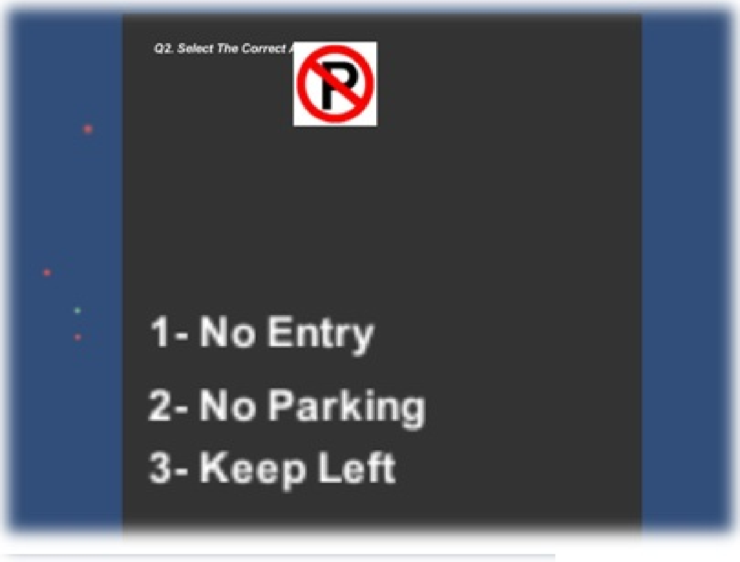
Q1. Select The Correct An

1- No Entry

2- No Parking

3- Keep Left

Figure D.44: First Question in Traffic Sign Test



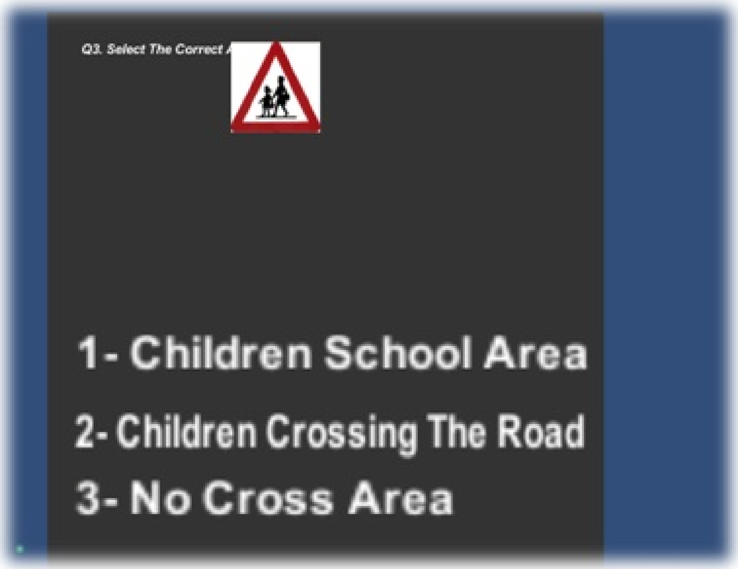
Q2. Select The Correct

1- No Entry

2- No Parking

3- Keep Left

Figure D.45: Second Question in Traffic Sign Test

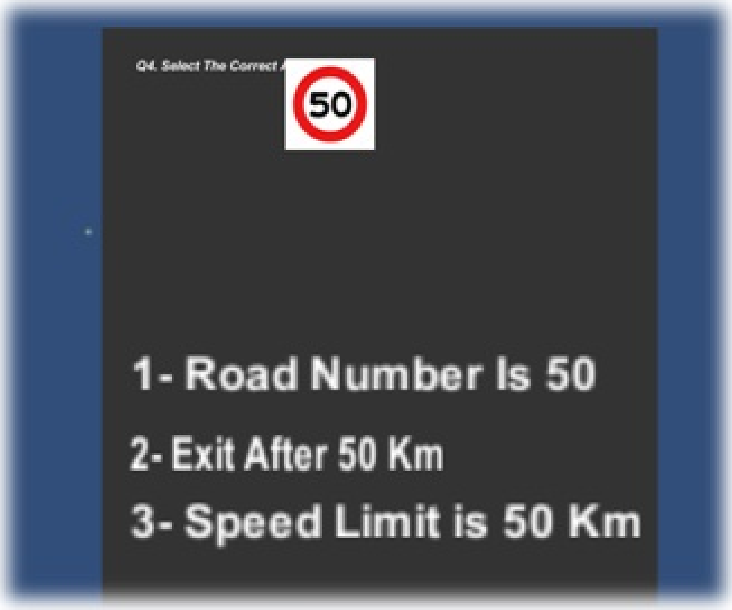

Q3. Select The Correct A

1- Children School Area

2- Children Crossing The Road

3- No Cross Area

Figure D.46: Third Question in Traffic Sign Test



Q4. Select The Correct A

1- Road Number Is 50

2- Exit After 50 Km

3- Speed Limit is 50 Km

Figure D.47: Forth Question in Traffic Sign Test

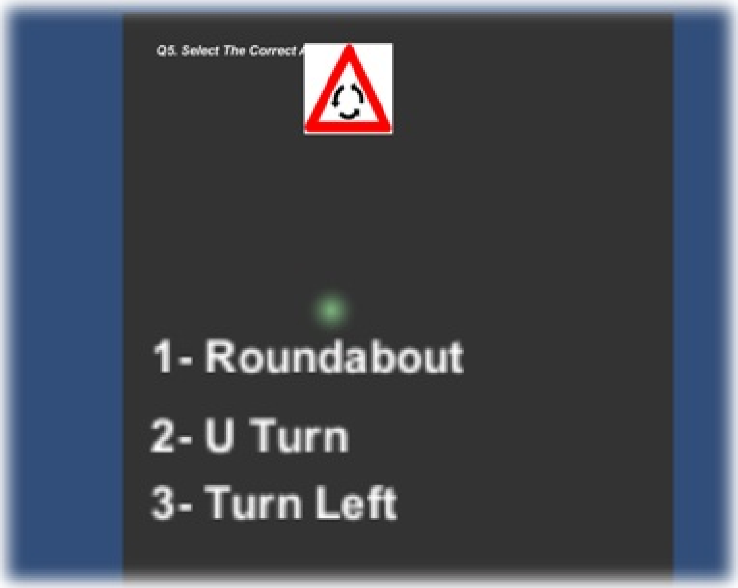


Figure D.48: Fifth Question in Traffic Sign Test

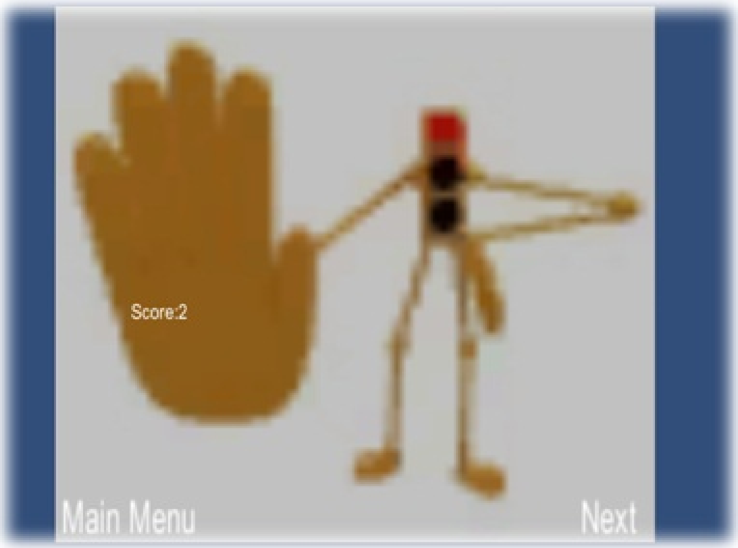


Figure D.49: Traffic Sign Test Final Score

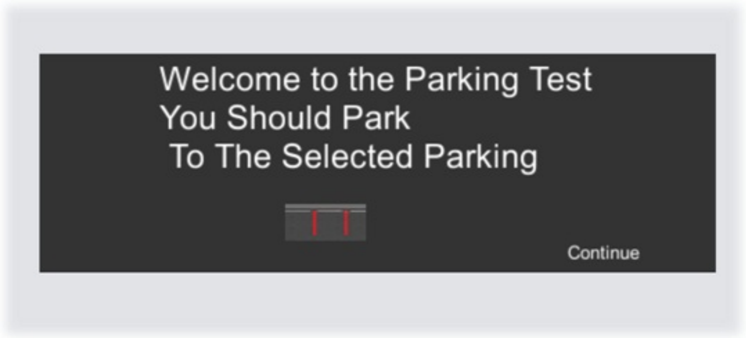

Welcome to the Parking Test
You Should Park
To The Selected Parking

Figure D.50: Description of First Part of Parking Game

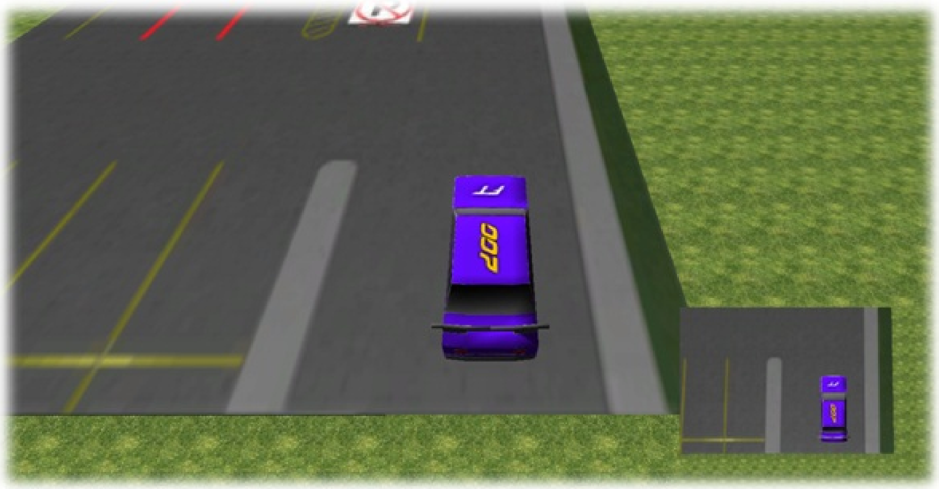


Figure D.51: First Part of Parking Game

Figure D.52: Result of First Part of Parking Game

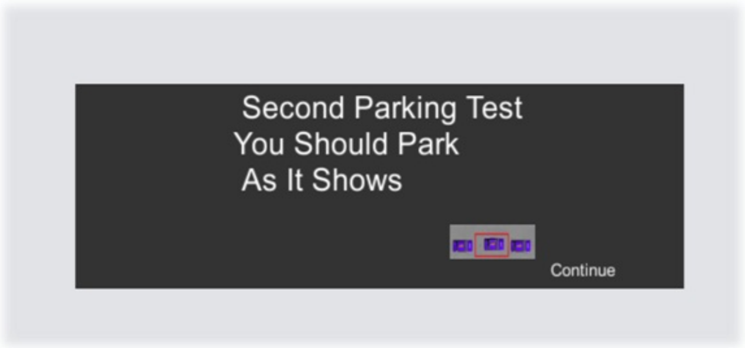

Second Parking Test
You Should Park
As It Shows

Figure D.53: Description of Second Part of Parking Game

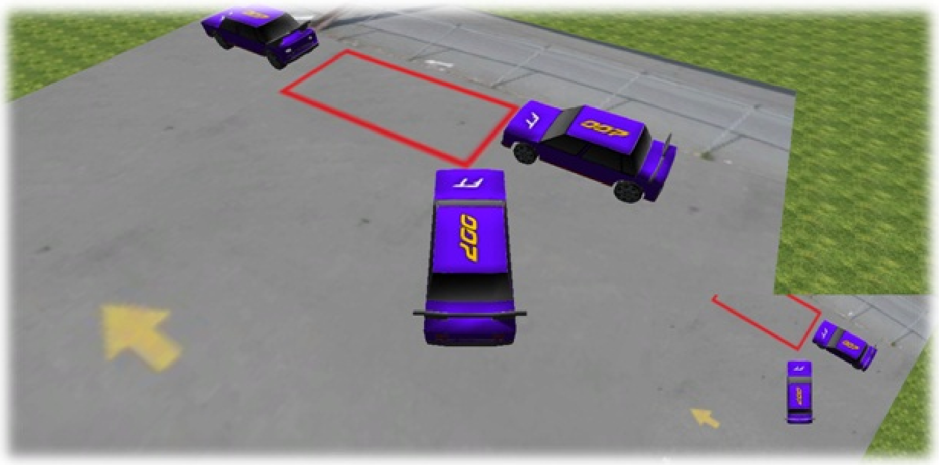
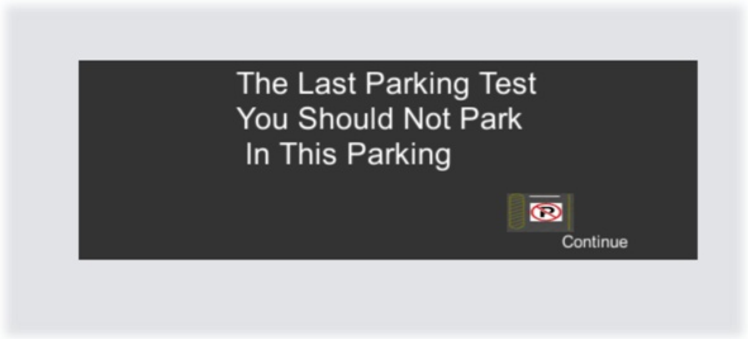


Figure D.54: Second Part of Parking Game

Figure D.55: Result of Second Part of Parking Game



The Last Parking Test
You Should Not Park
In This Parking

Figure D.56: Description of Third Part of Parking Game

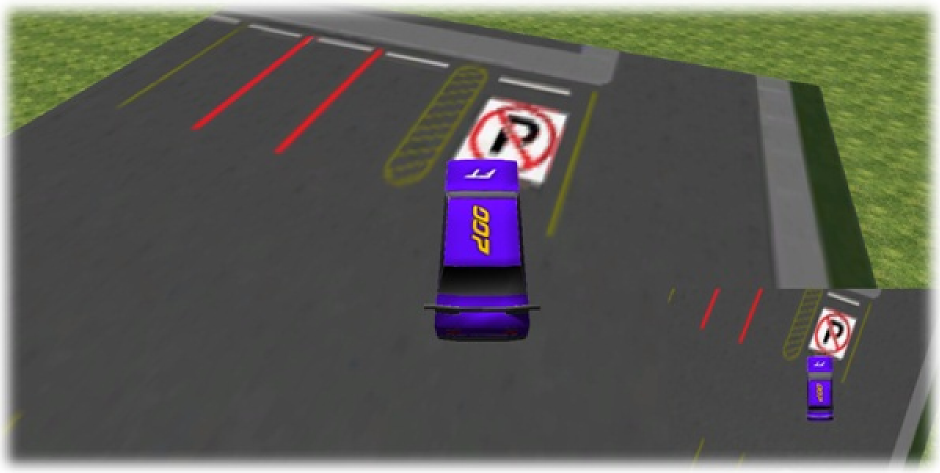
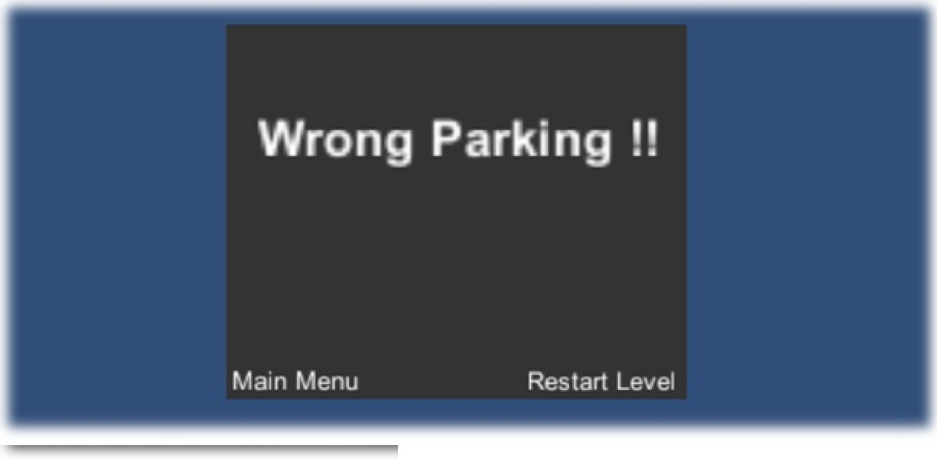


Figure D.57: Third Part of Parking Game



Wrong Parking !!

Figure D.58: Result of Third Part of Parking Game

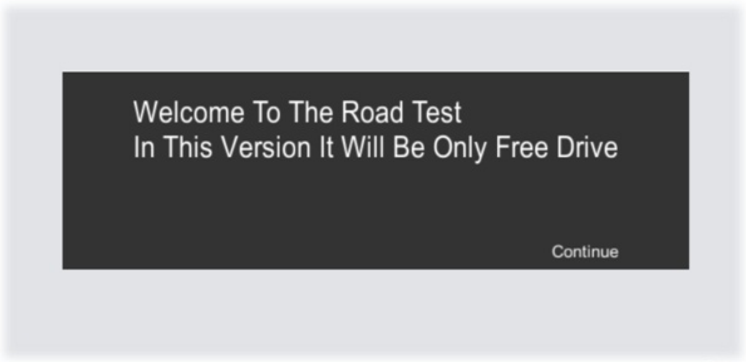

Welcome To The Road Test
In This Version It Will Be Only Free Drive

Figure D.59: Description of Road Test Game

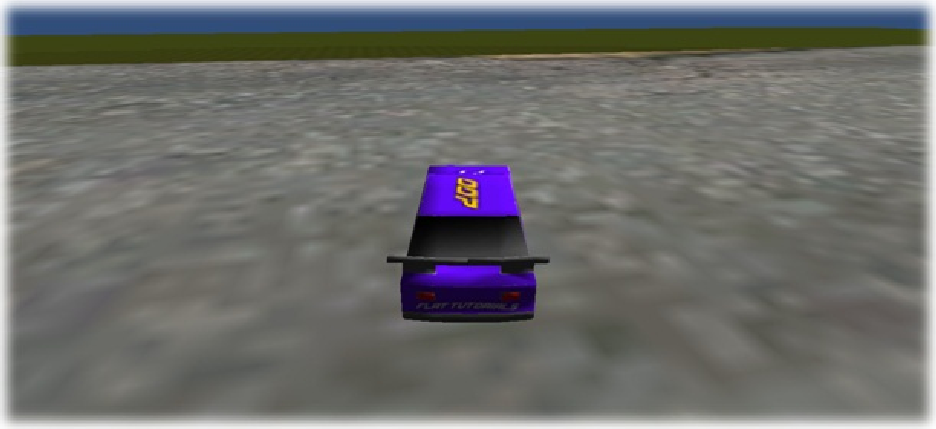


Figure D.60: Road Test Game

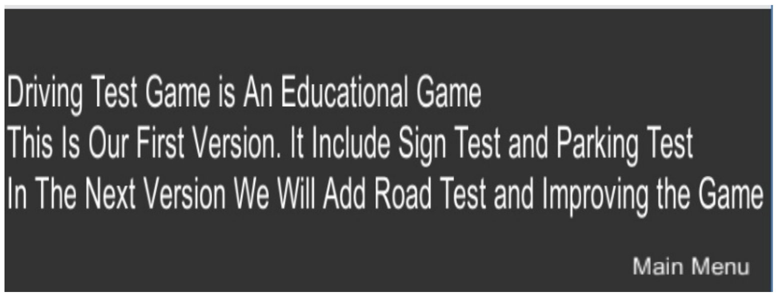

Driving Test Game is An Educational Game
This Is Our First Version. It Include Sign Test and Parking Test
In The Next Version We Will Add Road Test and Improving the Game

Figure D.61: About Game

Appendix E

Mobile Course Details

- **Assessment Details**

  50 percentage examination 2 Hours assessing.

  50 percentage assignment with a practical component assessing.

- **Learning Strategies**

  One lecture with two hours per week. The course will be covered with 14 weeks per semester.

  One practice session with one hour per week using application toolkits for Android software development.

- **Indicative Content**

  This course includes: Games Programming for Mobile Devices, Optimization, Network communication (e.g. Bluetooth and Wi-Fi), Databases and persistence in mobile systems, SMS programming, Maps and location-based services, and mobile security.

- **Prospectus Information**

  This module will build on the previous game and portable device programming experience, enabling you to consider more advanced issues and techniques when creating games for deployment on cellular phones, PDAs and handheld gaming platforms.

- **Resources**

  Suitable development environment such as ADT, NetBeans or similar IDE with J2ME development capabilities and device emulators Visual Studio Professional with mobile development environment PDAs or portable computer devices.

- **Special Admissions Requirements**

  Previous study of programming for mobile or equivalent.

- **Text Book**

  Creating mobile games: using Java ME platform to put the fun into your mobile device and cell phone, Hamer, Carol, Apress 2007, ISBN 9781590598801

- **Assignment Details**

  The data from course evaluation is based on the student's responses to the final course assignment evaluation. The feedback from the hard copy of the assignment submission by students. We usually submit to the students the marking criteria as shown in TableE.1 that they need to follow it to create their own game.

Table E.1: Marking Criteria for the Further Programming for Mobile Devices and Handheld Devices Course.

Criteria	Mark
Design of the Layout	5
Resource and Strings	5
The character used for the animation	10
Collision Detection	10
Game loop	5
Sound implementation	5
Alert Dialog and Toast	5
Switching between the Activities	5
Coding Style and Comments	5
Clear and concise documentation report, properly referenced where appropriate	15
Extra Features	5
Application Deployment	10
Demonstration Six Questions (Each question carries 2.5marks)	15

– **Game Scenario**

The following scenario is what we have submitted to the students to create their own game.

"You will be building a single user "Catch a coin" game for delivery to an Android Phone. A possible scenario for catch a coin is given below, but this is just for guidance. Do not stick rigidly to this idea."

- **Game Idea**

  The player will jump for joy (and coins) playing this game. The player will use the touch screen to move Toto (the game character) around to collect coins. Watch out for knives (or any other obstacle) that get in the way. Tap the screen to make Toto jump over obstacles and to save his life. If Toto gets hit by the obstacles, his health (life) will be drained. In case Toto's health is drained, stop the game and display how many coins are collected by Toto.

- **From a Programmer's View.**

  Toto The game character, can be created with any sprite object.

  Coins You can use any kind of coin, same or different to be collected by Toto.

  ObstaclesYou can use any bitmap to show the obstacles (it may be a knife, dagger, a stone or any other object). **Background** The background of the game is your choice totally.

  You should take care of the following points in your application.

  * The touch playing capability should be implemented in the game (i.e. The game character would be moving around the screen using the touch panel).
  * The coins collected by the user should be displayed in the form of Score or Points on the screen
  * The player should be able to turn on or turn off the game sounds.
  * Any sound should be played whenever the user jumps, collects the coins and gets hit by the obstacles.
  * The game should be played in full screen mode.

  The game application requires the following:

  * accelerometer
  * touch panel
  * music and video library

Consider the advance concepts and techniques that you can adopt in the development of the game application mentioned above. Find out the variety of mobile platforms on which you can deploy this application and discuss in the report how you will deploy your application available on different mobile platforms.

Appendix F

AOAB Workshops

F.1 Workshop Details

This section includes the full details of the workshop that is provided to the participants before they decide to attend our workshop.

F.1.1 What is this workshop about?

This workshop about to introduce a new game development methodology and provide a practical skills in order to ensure the quality and usability of AOAB methodology for both academic and industry sectors. A basic knowledge and some experience with Agile and AOSE methodology is provided.

F.1.2 What is the objective of this workshop?

This one day training will introduce you to the fundamental principles of the game development methodology and practices of Agent Oriented Agile Base methodology (AOAB). Attendees will participate in many hands-on activities that will help them practice the theory they learn, compare and evaluate. In this workshop, the attendees will learn the principles and values of AOAB and will explore in detail the AOAB life cycle. During the one day session, the attendees will share his years of experience in game development methodologies.

F.1.3 Who should participate in this workshop?

This workshop is designed for people who are responsible for specifying, acquiring, developing, evaluating, supporting and/or managing games, for example:

- Game Development Team Leaders.
- Game Development Academic staff.
- Game Development student.
- Game Developer.
- Game Designer.
- Game Programmers.
- Project Managers of Games in industry area.
- Expert game development evaluator.
- Any person interested in the future of game development trends.

F.1.4 What is the organizer background?

This is our first workshop on the game development methodology. Over the last few years, we have worked on game development methodologies. This is due to the subject of my PhD thesis. Rula Khalid is a senior lecturer in software engineering and games. She is in last year PhD at De-Montfort University, UK. She has many publications on the game development methodology and game evaluation. She is currently an award leader of computing department in gulf college, Oman affiliated with Staffordshire University-UK.

F.1.5 What is the workshop goal?

The goal of the workshop contains the following:

- Provides a framework for researchers and practitioners in the field of game development methodology.
- The thematic focus on the feedback of the researcher and participate regarding the new game development methodology which is based on Agent Oriented Software Engineering methodology (AOSE) and Agile methodology as an important design parameter of our methodology.
- Plan for the future works in game development methodology.
- Assess how professional designers and developments as well as academics and end users are using the methodology to create and get the professional game in the real world.

At the end of workshop, you will learn how to:

- Manage, design, test and develop any game genres.
- Model requirements and applications using AgentUML models.
- Monitor progress with backlogs.
- Increase quality with user and expert evaluation of the first game prototype.
- Maximize team productivity and communication.
- Become an advocate in your organization of practical methods to improve game project performance.
- Identify causes of game development problems and drive game design and development performance improvements.

F.1.6 Our full day workshop agenda

Part 1

- Provide an overview of the current game development methodologies and elements.
- Identify the challenges and the academic analysis of these issues.
- Explore the space of difference approach and learn game development principals.
- Explaining the adaptive methodology vs. traditional methodologies.
- Encouraging the team to be adaptive .
- Creating a safe environment in which the team can explore novel solutions.
- Running collaboration games to identify and solve problems.

Part 2

- Developing responses to typical scenarios.
- We present in general the concept of Agile development methodology.
- We present in general the concept of AOSE methodology.
- Compare the difference between an adaptive approach such as Agile methodology and predictive approach such as MaSE methodology.
- We present the results of recent and ongoing original work.
- Identifying features for development in an iteration.
- Determining ideal iteration length.
- Identifying development tasks in the Sprint Backlog.
- Revising team behavior on the basis of lessons learned.

- Identifying Game Design Document (GDD).
- Documenting nonfunctional and system requirements.
- Increasing communication with stand-up meetings, task boards and regular reflection.
- Maximizing team productivity.
- Identify the Agent UML modules such as goal hierarchy diagrams, Agent class diagram, Role model diagram, use case diagram and deployment diagram.

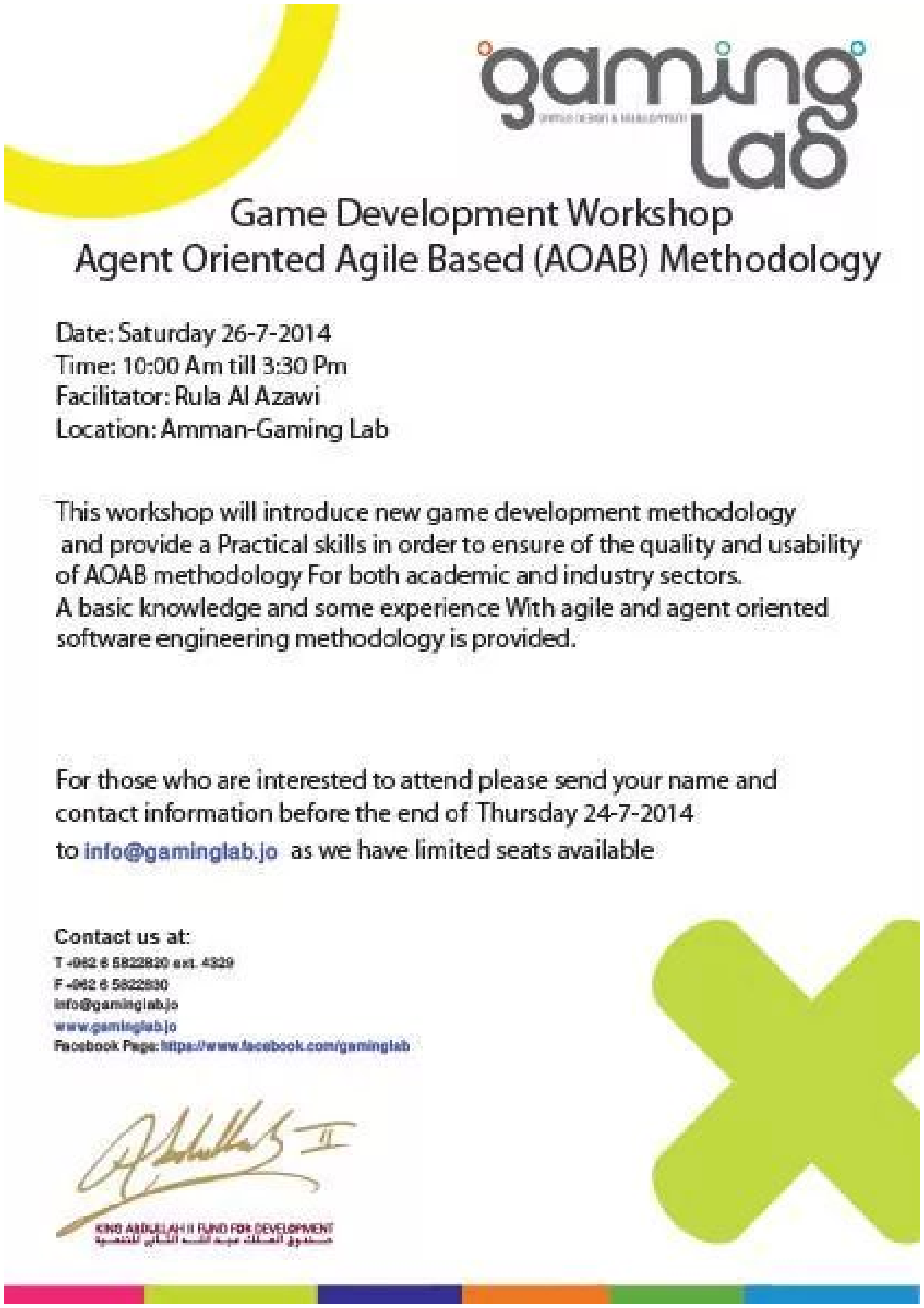
Part 3

- The workshop will conclude with a round-table discussion, questionnaire and will highlight open research question and identify new ways in which game development methodology can deals and understand these issues.
- Work on small group to design games based on different methodology.
- Measuring work completed with backlogs.
- Identifying best practices for team productivity.
- Writing user stories.
- Prioritizing and estimating work.
- Optimizing Agile teams.
- Creating a Product Backlog as a list of requirements and technical issues.
- Iterating development through cycles.
- Capturing user needs as stories.
- Using capacity-based planning to plan progress.

- Measuring estimated effort with story points.
- Utilize UML diagrams.

F.2 Workshops Invitation

The first workshop invitation is in the Figure F.1. The second workshop invitation exists on the main page of the GAMEON conference [5].



gaming lab

Game Development Workshop
Agent Oriented Agile Based (AOAB) Methodology

Date: Saturday 26-7-2014
Time: 10:00 Am till 3:30 Pm
Facilitator: Rula Al Azawi
Location: Amman-Gaming Lab

This workshop will introduce new game development methodology and provide a Practical skills in order to ensure of the quality and usability of AOAB methodology For both academic and industry sectors.
A basic knowledge and some experience With agile and agent oriented software engineering methodology is provided.

For those who are interested to attend please send your name and contact information before the end of Thursday 24-7-2014 to info@gaminglab.jo as we have limited seats available

Contact us at:
T +962 6 5822820 ext. 4329
F +962 6 5822830
info@gaminglab.jo
www.gaminglab.jo
Facebook Page: https://www.facebook.com/gaminglab

[illegible]

KING ABDULLAH II FUND FOR DEVELOPMENT
صندوق الملك عبدالله الثاني للتنمية

Figure F.1: First Workshop Invitation

F.3 Workshop Attendees Questionnaire

This questionnaire is for academic purposes only. It forms part of my PhD. All responses are strictly confidential and for research purposes only.

Please you need to start with part one which includes general information then select either part two or part three depending on your background and experience.

- **Part One: General Information**

1. **Name (optional):**
2. **Occupation:**
3. **Education:**
4. **Age:**
5. **Gender:**
6. **You have strong relation with**

   A. Industry Sector B. Academy Sector C. Both of them.

7. **Are you satisfy of the workshop content? Please provide some comments**

- **Part Two: Expert Game Designer and Developer**

1. **What kind of development methodology you prefer with games**

A. Adaptive B. Predictive C. Hybrid.

2. **What is your suggestion to enhance the AOAB Methodology**

3. **What is the weak point you have found in AOAB Methodology**

4. **What is the strength point you have found in AOAB Methodology**

5. **Does the current game development methodology enough with your work, if not explain why**

6. **What are the attention wait you give to select game development methodology**

7. **What are the size of game you have been working on**

   A. Small B. Medium C. Large.

- ## Part Three: General Game User

1. **Do you care about stories in games?**

   A. Yes B. No

2. **What is your favorite game? (Or general favorite games)**

3. **Do you think games can be educational?**

   A. Yes B. No

4. **Do you think it enhances any learning skills by playing games?**

5. **Do you prefer learning through games or learning through exercises and books?**

   A. Through games B. Through books C. Through exercises D. Others

6. **Why do you play games?**

7. **What games would you like to see more of in the future?**

8. **Gaming technology in your house is?**

   A. PC B. Xbox C. Game-boy D. Laptop E. others

9. **Approximately what is the average time you play games each week?**

   A. Three hours B. Four hours C. More than D. Less than

10. **What type of games do you like? (genre of the game)**

11. **How do you select games to play?**

    A. From friend B. From Website C. From Internet friend D. Others

12. **Why do you like some games? What are some of the features that makes you like it?**

F.4 GameLab and Lincoln Workshop Pictures

Figure F.2: Facilitator Explains AOAB Methodology

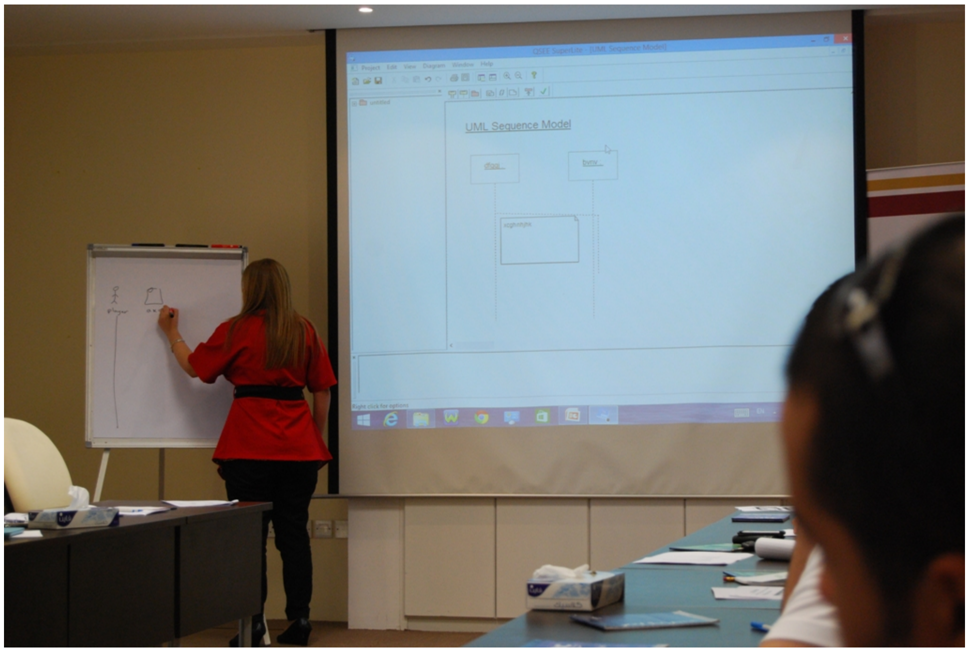


Figure F.3: Facilitator Explains AOAB Diagrams

Figure F.4: Audience in the Workshop

Figure F.5: Round table Discussion

Figure F.6: GameLab Entrance

Figure F.7: Laboratory in GameLab

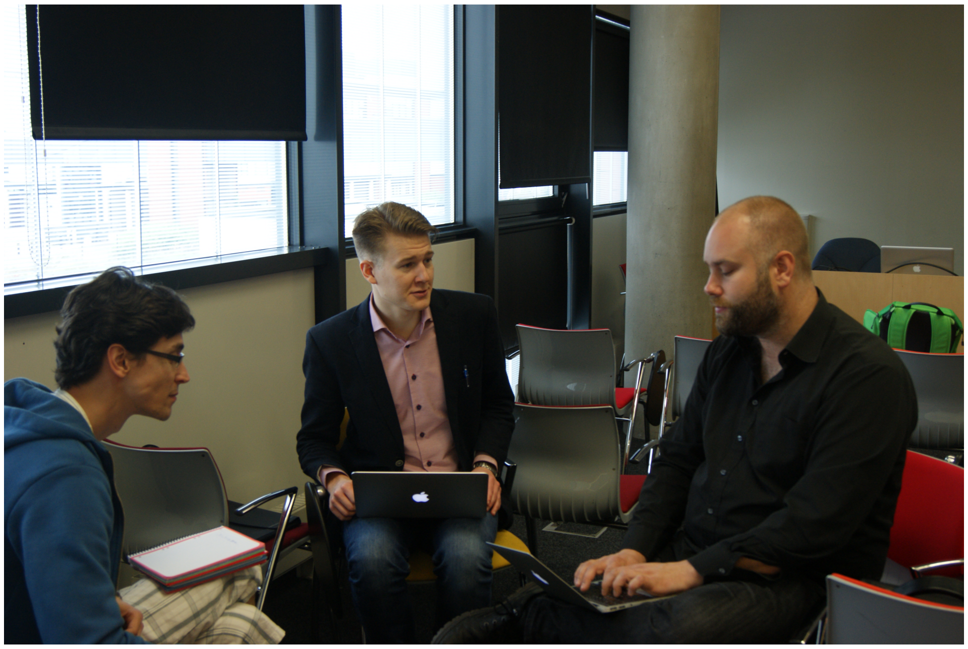


Figure F.8: Audience in the Workshop

Figure F.9: Audience in the Workshop

F.5 AgentTool3

Agent Tool3 software downloaded from the following Link: [10]

To install AgentTool III, you must have the following software installed:

1. Java Runtime Environment 1.5.0 or higher

2. Eclipse 3.4.2, or above

Step-1: Open Eclipse and select New Project:

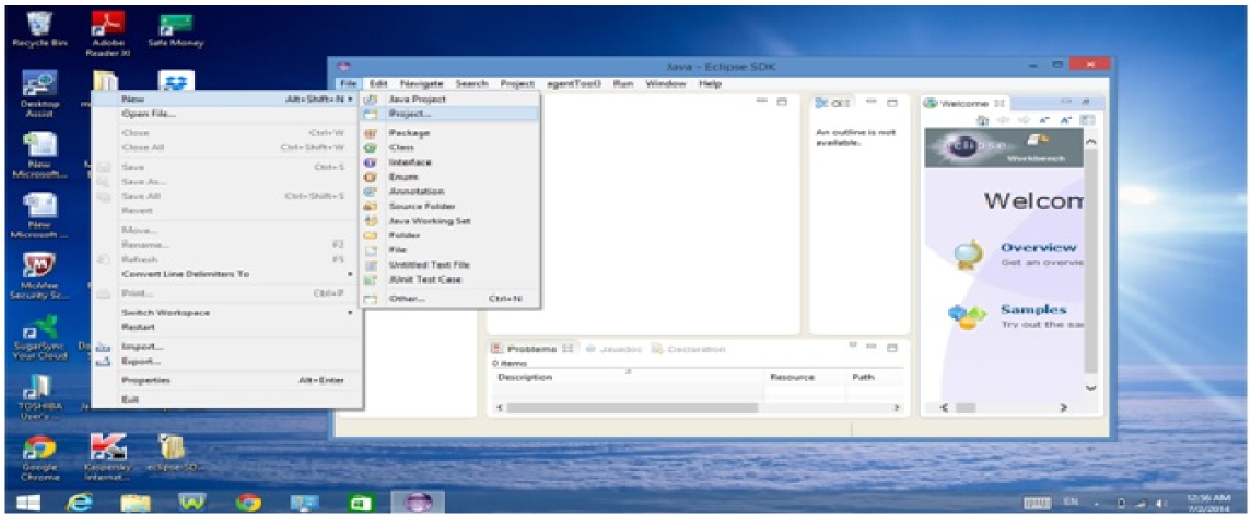


Figure F.10: Step 1

Step-2: Press next , button.

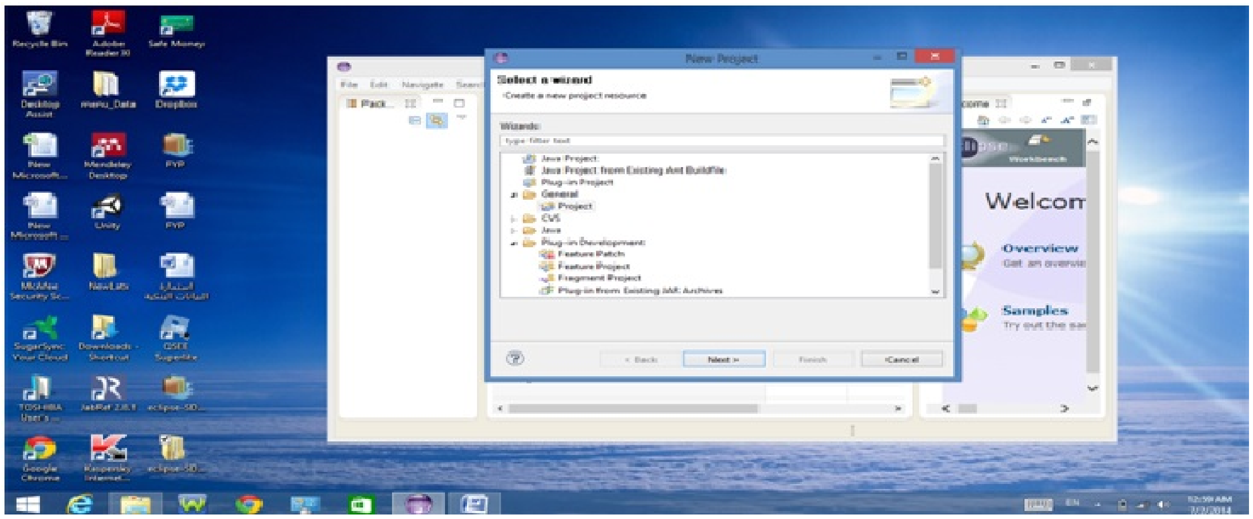
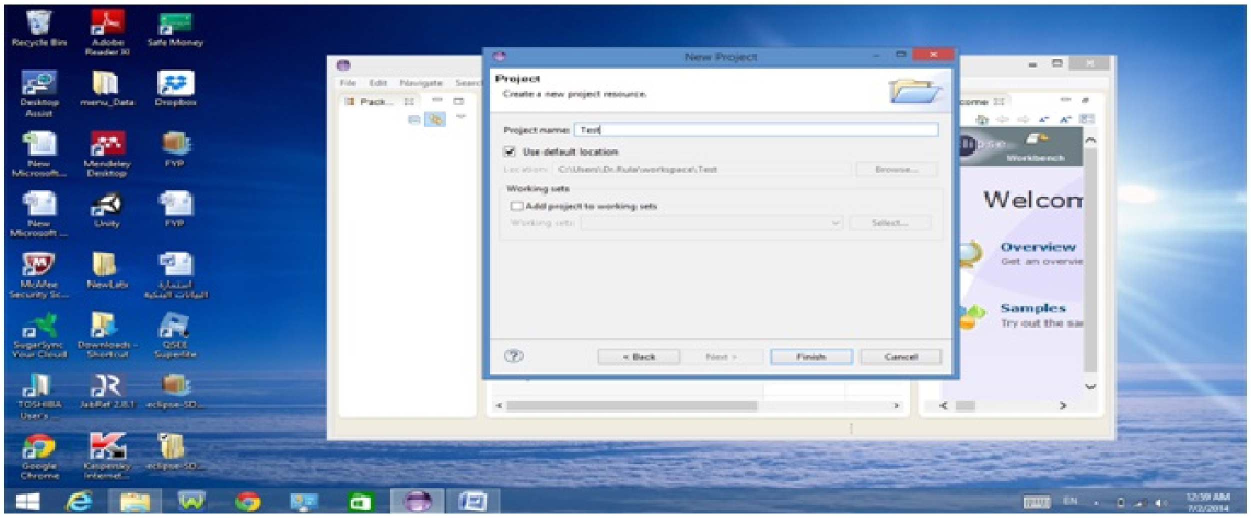


Figure F.11: Step 2

Step-3: Print project name then press finish , button.



Project name: Test

Location: C:\Users\Dr.Rula\workspace\Test

Figure F.12: Step 3

Step-4: Go to file and select new then select other (Ctrl+ N).

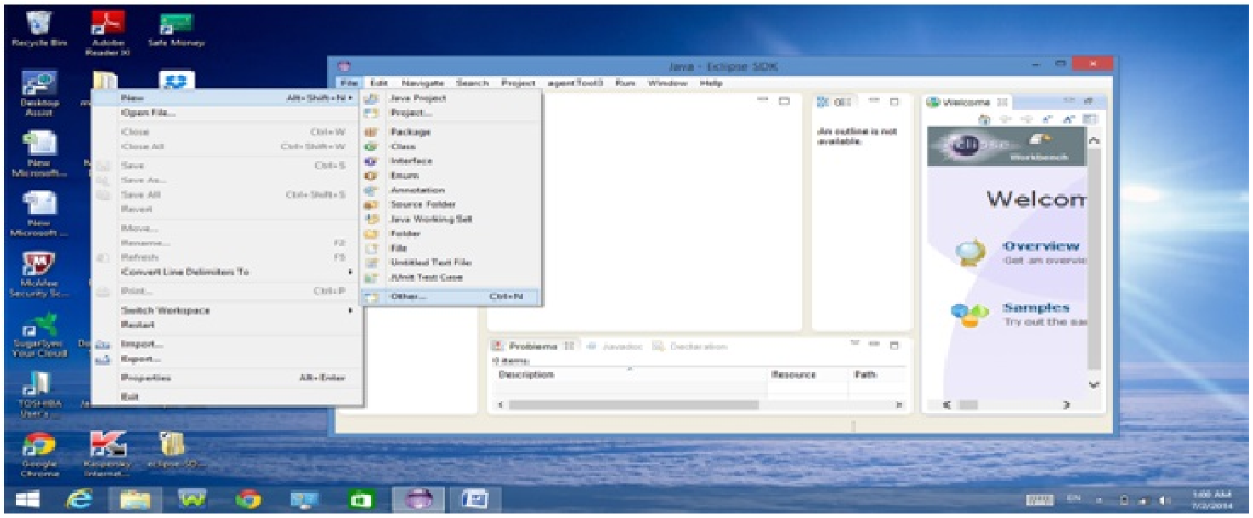


Figure F.13: Step 4

Step-5: Select the required agent diagram then press next , button.

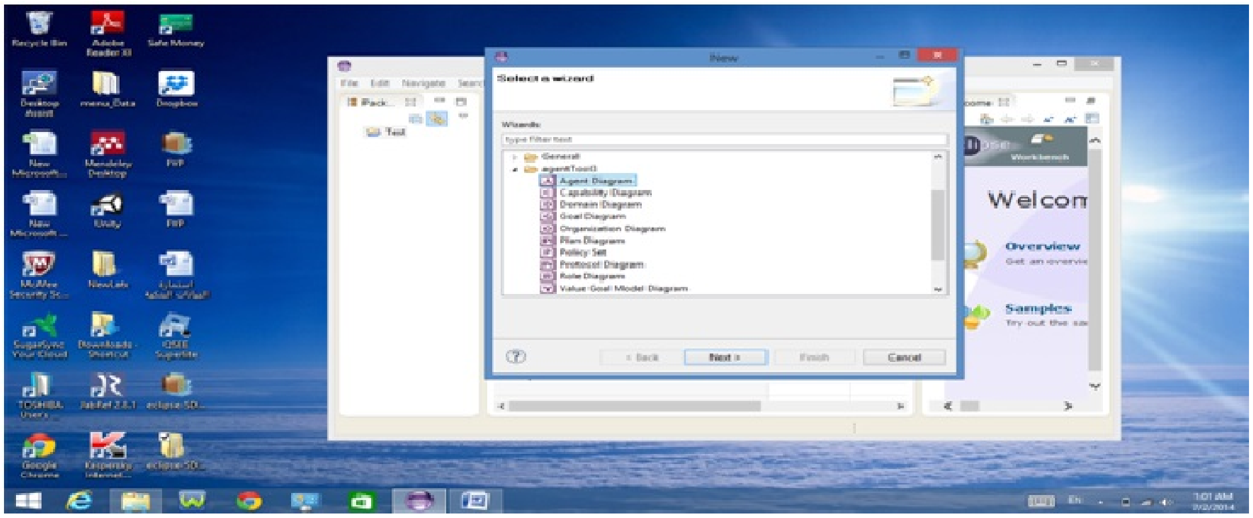
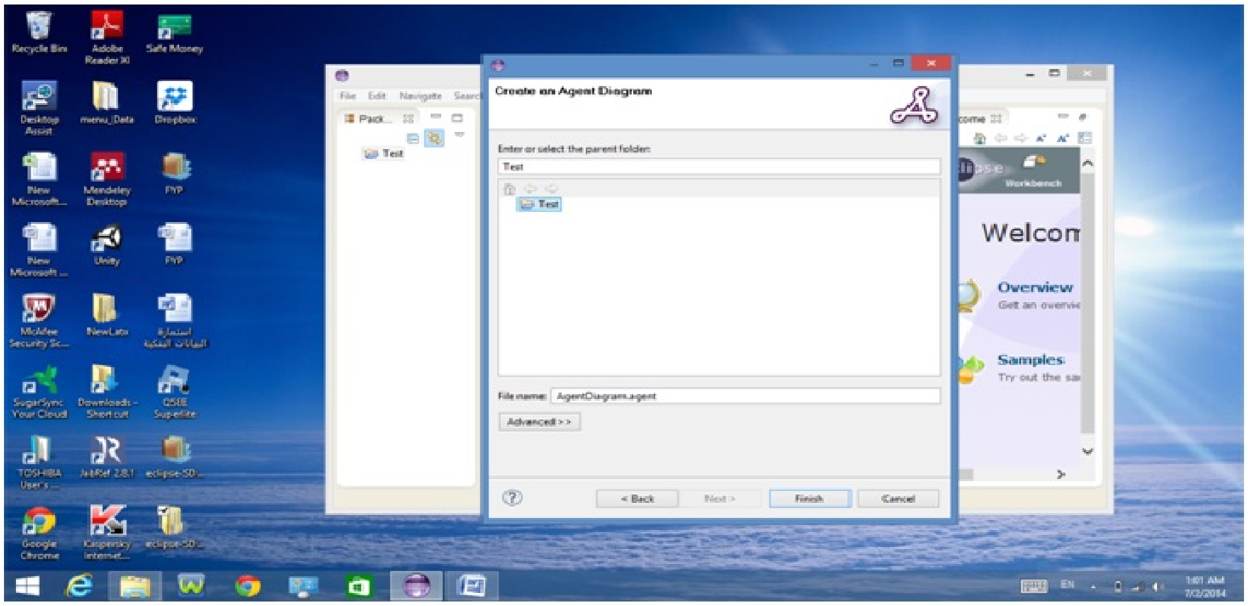


Figure F.14: Step 5

Step-6: Select the project name that you have created before, then press finish.



Enter or select the parent folder: Test

File name: AgentDiagram.agent

Figure F.15: Step 6

Step-7: You have all the required components. Drug and Drop and change the names.

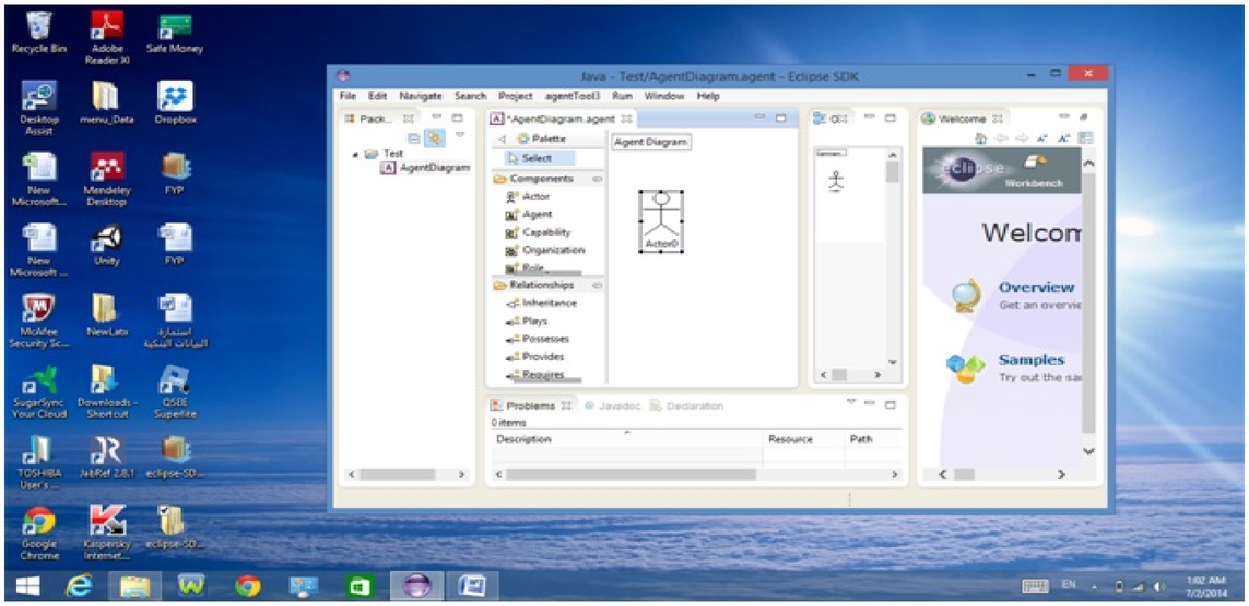
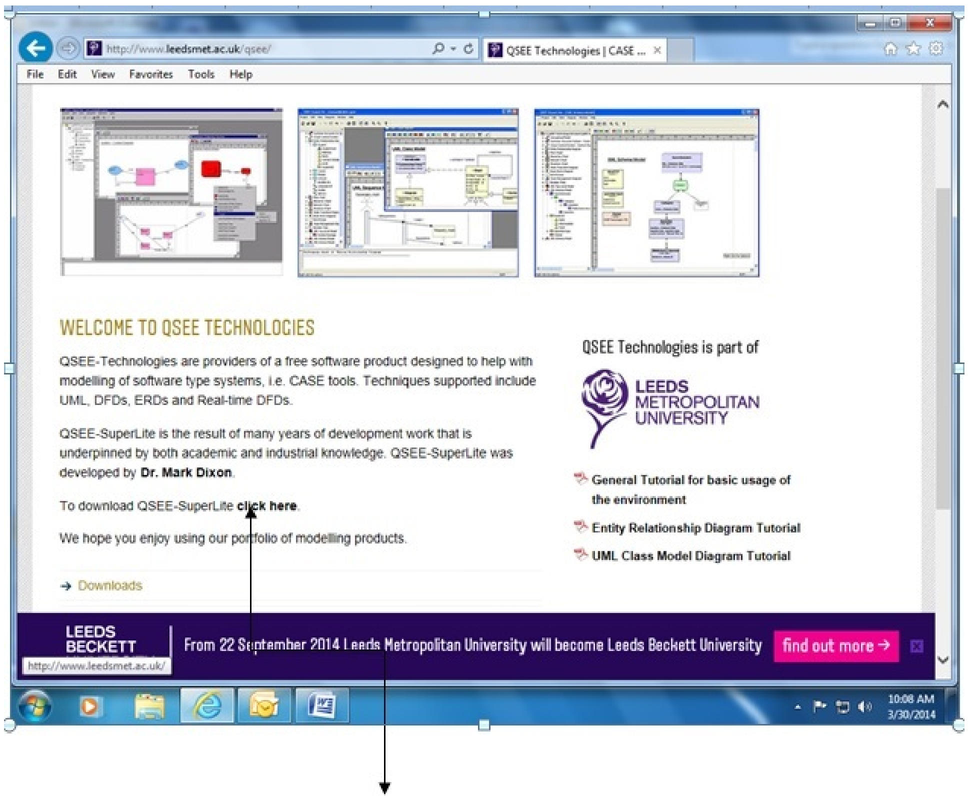


Figure F.16: Step 7

F.6 QSEE Technology

QSEE Superlit software downloaded from the following link [11] and as shown in Figure F.18 to draw the UML Models.


http://www.leedsmet.ac.uk/qsee/
QSEE Technologies | CASE ...
File Edit View Favorites Tools Help
LEEDS METROPOLITAN UNIVERSITY
Downloads
LEEDS BECKETT
find out more →
http://www.leedsmet.ac.uk/
10:08 AM
3/30/2014

WELCOME TO QSEE TECHNOLOGIES

QSEE-Technologies are providers of a free software product designed to help with modelling of software type systems, i.e. CASE tools. Techniques supported include UML, DFDs, ERDs and Real-time DFDs.

QSEE-SuperLite is the result of many years of development work that is underpinned by both academic and industrial knowledge. QSEE-SuperLite was developed by **Dr. Mark Dixon**.

To download QSEE-SuperLite **click here**.

We hope you enjoy using our portfolio of modelling products.

QSEE Technologies is part of

General Tutorial for basic usage of the environment

Entity Relationship Diagram Tutorial

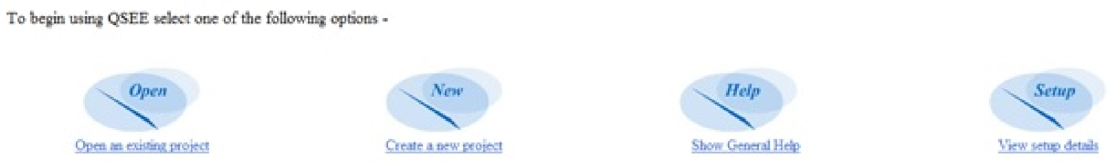
UML Class Model Diagram Tutorial

From 22 September 2014 Leeds Metropolitan University will become Leeds Beckett University

Click Here → Click **Run** button → **Save** button.

Figure F.17: Install QSEE

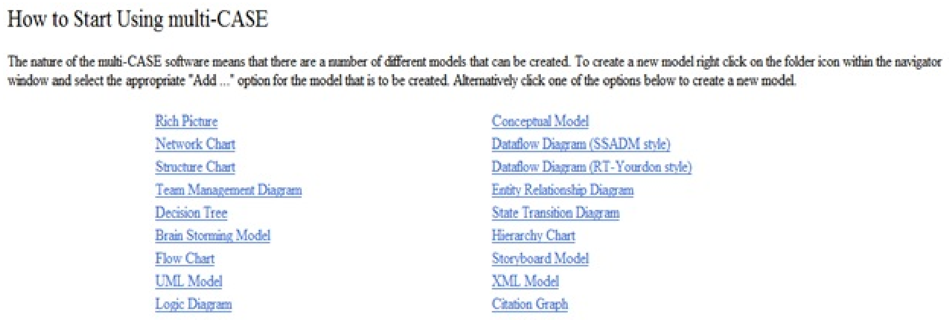
Step-1: Open New Project: Click New or Create New Project. It shows following screen.



To begin using QSEE select one of the following options -

Figure F.18: Step 1

Step-2: Select UML Model



How to Start Using multi-CASE

The nature of the multi-CASE software means that there are a number of different models that can be created. To create a new model right click on the folder icon within the navigator window and select the appropriate "Add ..." option for the model that is to be created. Alternatively click one of the options below to create a new model.

Rich Picture	Conceptual Model
Network Chart	Dataflow Diagram (SSADM style)
Structure Chart	Dataflow Diagram (RT-Yourdon style)
Team Management Diagram	Entity Relationship Diagram
Decision Tree	State Transition Diagram
Brain Storming Model	Hierarchy Chart
Flow Chart	Storyboard Model
UML Model	XML Model
Logic Diagram	Citation Graph

Figure F.19: Step 2

Step-3: Adding the Entity. Right click on the diagram area and select the UML diagram option

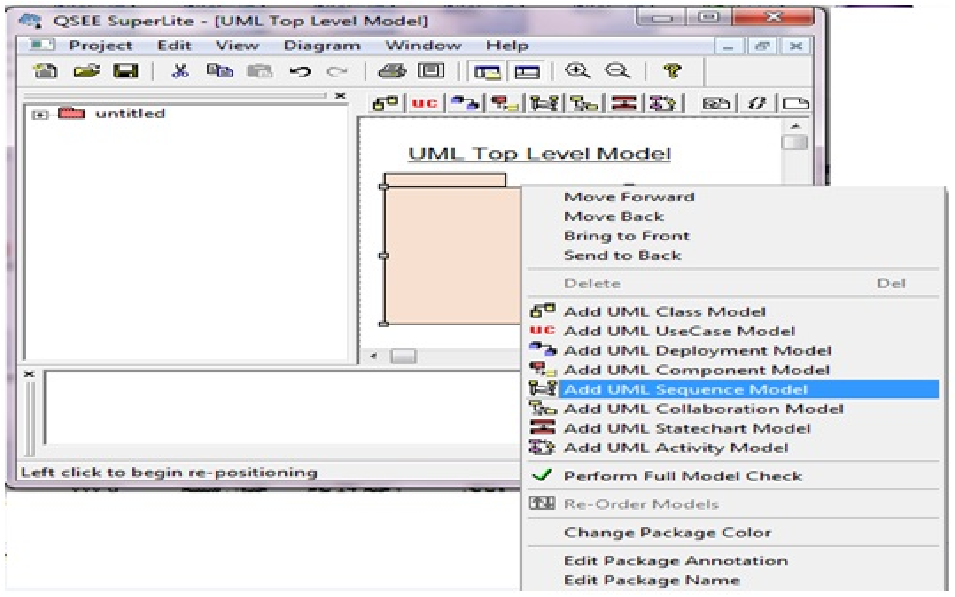


Figure F.20: Step 3

Step-4: Type name of diagram you prefer and press the OK , button on the form.

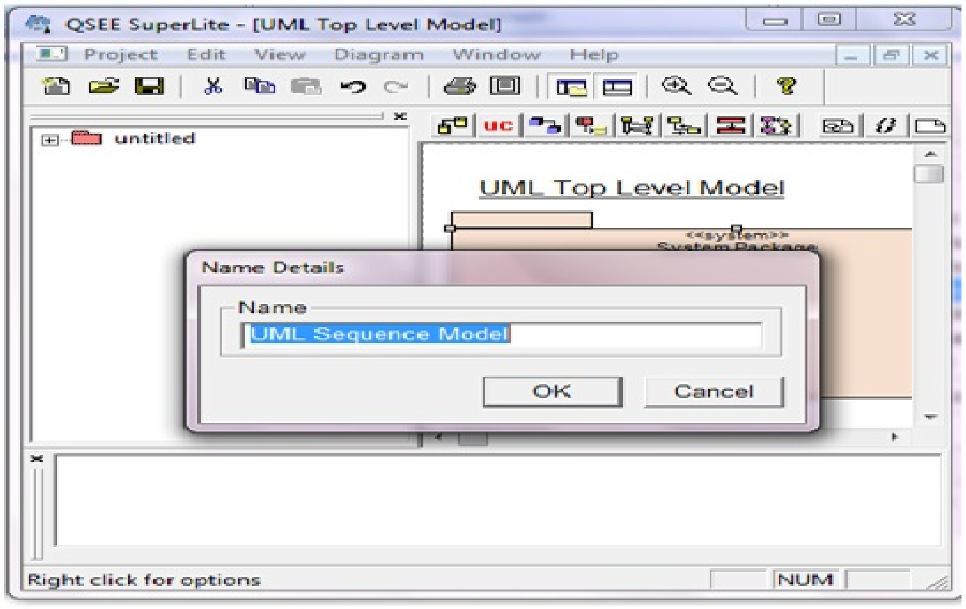


Figure F.21: Step 4

Step-5: Adding the Component. Right click on the entity and add Component.

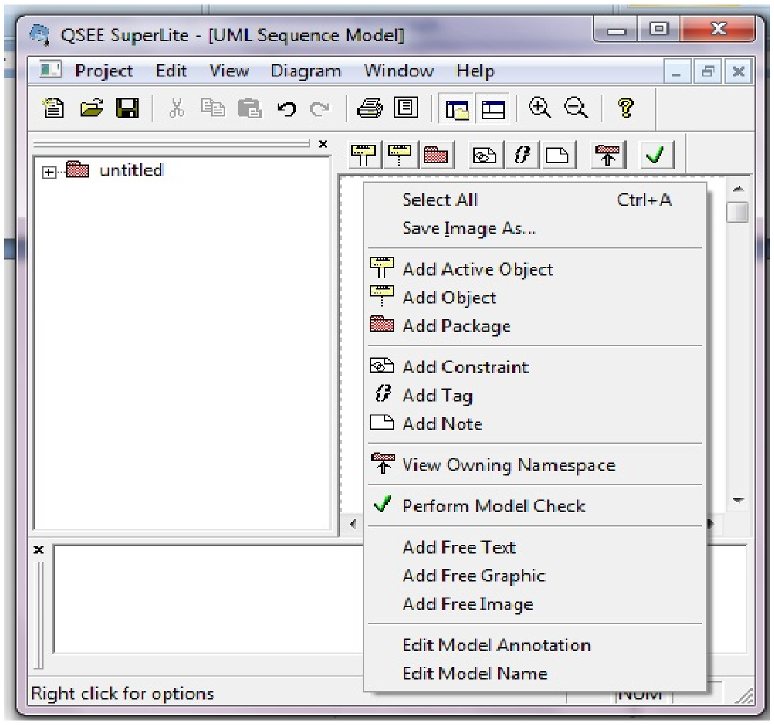


Figure F.22: Step 5

Step-6. Enter the details into the dialogue and press OK.

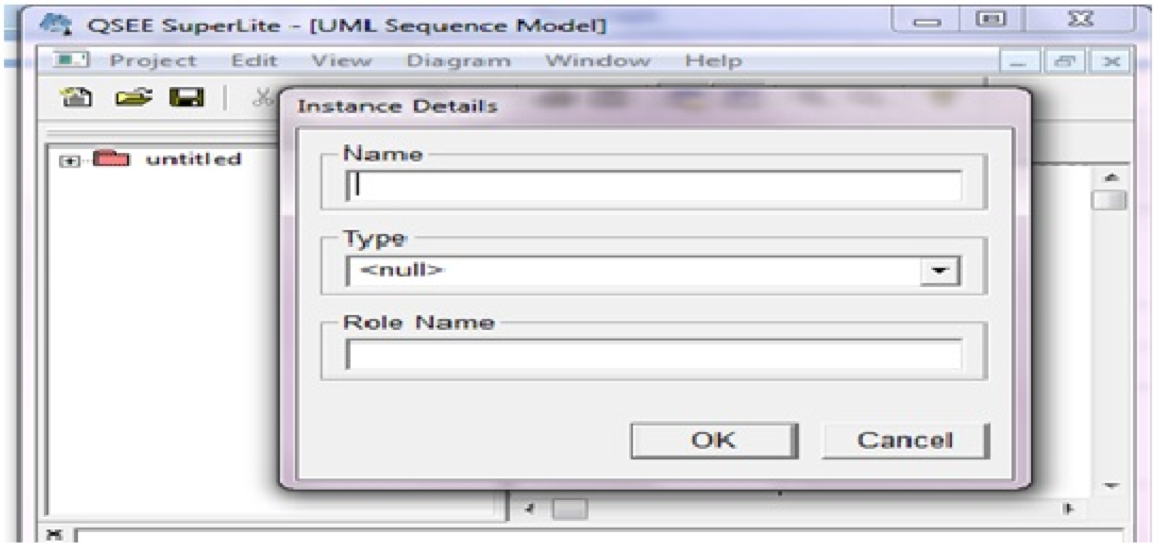


Figure F.23: Step 6

F.7 Participants Positive and Negative Feedback

Q1.Are you satisfy of the workshop content? Please provide some comments		
Positive Comment	Negative Comment	No Comment
1. The content of the workshop is very useful 2.It is easily to be understandable by developer 3.It is informative and explored some interesting ideas 4.AOAB has clear steps and phases to follow	1.It cannot cover some genre of games 2.It is not good for small games	
Total=22	Total=3	Total=8
Q2. What is your suggestion to enhance the AOAB Methodology		
Positive Comment	Negative Comment	No Comment
1. It is easy and applicable 2. Can add intelligent to game 3.Need more explanation for agent diagrams 4.Provide clear life cycle 5.Facilitate the team work	1.Redesign the GDD 2.Need to deal more with business concept	
Total=11	Total=2	Total=20
Q3. What is the weakness and strength point you have found in AOAB Methodology		
Positive Comment	Negative Comment	No Comment
1.It helps to be organized and will reduce the no. of errors 2. It helps to have full idea of game aspect before start implementation 3. It is deal with agent easily 4.Create complete documentation 5.Minimize the error in implementation phase 6.Very useful when create new version of game	1.Create large documentation 2.Include many diagrams 3. Take more time in create diagram 4.Developer prefer programming code rather than modeling diagrams	
Total=19	Total=4	Total=10
Q4.Does the current game development methodology enough with your work, if no explain why		
Positive Comment	Negative Comment	No Comment
1.Yes, we deal with adaptive methodology 2.Enough with small game 3..Yes, when I work individually	1.Not enough with large game	
Total=13	Total=2	Total=18

Bibliography

[1] *Atomic Object LLC. http://spin.atomicobject.com/2012/12/29/game-programming-boo-unity-engine-part-1/; Last Access: March2012.*

[2] *Blender orgnazation. http://download.blender.org/documentation/pdf/; Last Access: October 2013.*

[3] *Create 3D Gasmes. http://create3dgames.wordpress.com/2012/07/03/unity-vs-udk/; LastAccess: July2014.*

[4] *GameLab; http:gaminglab.jo/jo/about-2/; Last Access:Feb2014.*

[5] *GAMEON'2014; http://www.eurosis.org/cms/index.php?q=node/2844; Last Access: July2014.*

[6] *http://conkerjo.wordpress.com/category/xna/; Last Access: March 2013.*

[7] *http://rpgmaker.net/engines/gm/; Last Access: April2013.*

[8] *IBM Innov8 : http: //www-01.ibm.com/software/solutions/soa/innov8; Last Access: April 2013.*

[9] *Intent Media Ltd; http://www.develop-online.net/tools-and-tech/the-top-10-game-engines-no-4-unity-3d/0116475 ;Last Access: Feb2013.*

[10] *Kansas State University; http://agenttool.cis.ksu.edu/; Last Access: Jan2012.*

[11] *Leed Metropolitan University : http://www.leedsmet.ac.uk/qsee/; Last Access:Feb2014.*

[12] *Oman Driving License; http://omandrivinglicense.blogspot.com/2013/10/blog-post.html?m=1; Last Access: Feb2013.*

[13] *Video Game Industry Statics,http://www.esrb.org; Last Access:March 2013.*

[14] *Wikipedia. http://en.wikipedia.org/wiki/Game; Last Access: March 2012.*

[15] *wikipedia. http://en.wikipedia.org/wiki/MicrosoftXNA; Last Access: Feb2013.*

[16] C Abt. *Serious games*. University Press of America, 1987.

[17] P. S. Adler. The evolving object of software development. In *Organization*, volume 12, 2005.

[18] Z Akbari. A Survey Of Agent-oriented Software Engineering Paradigm: Towards Its Industrial Acceptance. *Journal of Computer Engineering Research*, 1:14–28, 2010.

[19] Zohreh Akbari and Ahmad Faraahi. Evaluation Framework for Agent-Oriented Methodologies. In *Proceedings of World Academy of Science (WCSET)*, volume 35, pages 419–424, Paris, France, 2008.

[20] Rula Al-Azawi and Aladdin Ayesh. Comparing Agent -Oriented Programming Versus Object-Oriented Programming. In *ICIT 2013 The 6th International Conference on Information Technology*, pages 24–29, Jordan, 8-10 May 2013. IEEE Jordan Chapter.

[21] Rula Al-Azawi, Aladdin Ayesh, and Mohaned Al-Obaidi. Generic evaluation framework for games development methodology. In *Third International Conference on Communications and Information Technology ICCIT 2013*, pages 55–60, Lebanon, 19-21 June 2013. IEEE Computer Society.

[22] Rula Al-Azawi, Aladdin Ayesh, and Mohaned Al-Obaidi. Towards agent-based agile approach for game development methodology. In *WCCAIS'2014 World Congress On Computer Applications and Information Systems,*, Hammamet, Tunisia, January 2014. IEEE Computer Society.

[23] Rula Al-Azawi, Aladdin Ayesh, Ian Kenny, and Khalfan AL-Masrur. Multi agent software engineering (mase) and agile methodology for game development. In *14th Middle Eastern Simulation and Modelling Multiconference, MESM 2014 - 4th GAMEON-ARABIA Conference, GAMEON-ARABIA 2014., In The 14th Middle Eastern Simulation and Modelling Multiconference (MESM) - The 4th GAMEON-ARABIA Conference.*, pages 116–122, Muscat, Oman, 2014.

[24] Rula Al-Azawi, Aladdin Ayesh, Ian Kenny, and Khalfan AL-Masruria. Analysis of using intelligent technique in games. In *3rd GAMEON-ARABIA'2012 Conference*, pages 83–87, Oman,Muscat, December 2012. EUROSIS.

[25] Rula Al-Azawi, Aladdin Ayesh, Ian Kenny, and Khalfan AL-Masruria. Towards an aose: Game development methodology. In *Distributed Computing and Artificial Intelligence, 10th International Conference. Advances in Intelligent and Soft-Computing series of Springer*, volume 217, pages 493–501, Selemenca, Spain, May 2013.

[26] EF Al-Hashel. A Novel Development Methodology for Cooperative, Distributed Multi-agent Systems. Master's thesis, Faculty of information science and engineering, University of Canberra, Australia, 2010.

[27] Apostolos Ampatzoglou and Alexander Chatzigeorgiou. Evaluation Of Object-oriented Design Patterns In Game Development. *Information and Software Technology*, 49(5):445–454, May 2007.

[28] Apostolos Ampatzoglou and Ioannis Stamelos. Software Engineering Research For Computer Games: A Systematic Review. *Information and Software Technology*, 52(9):888–901, 2010.

[29] EF Anderson. Playing Smart-artificial Intelligence In Computer Games. *Proceedings of CON03 Conference on Game Development, ZFX - 3D Entertainment*, 2003.

[30] Gustavo Andrade, Geber Ramalho, AS Gomes, and Vincent Corruble. Dynamic Game Balancing: An Evaluation Of User Satisfaction. In *the 2nd Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE06).*, pages 3–8. AAAI Press, 2006.

[31] D. Avison and G. Fitzgerald. *Information Systems Development: Methodologies, Techniques and Tools.* McGraw-Hill, New York, 2nd edition edition, 1995.

[32] R. Basseda, F. Taghiyareh, T. Alinaghi, C. Ghoroghi, and A. Moallem. A Framework For Estimation Of Complexity In Agent Oriented Methodologies. In *Conference on Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International*, pages 645–652, May 2009.

[33] Reza Basseda and Tannaz Alinaghi. A Dependency Based Framework For The Evaluation Of Agent Oriented Methodologies. In *IEEE International Conference on System of Systems Engineering, SoSE 2009.*, 2009.

[34] E Bethke. *Game Development And Production*. Wordware Publishing, Inc., 2003.

[35] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, may 2004.

[36] W. Brown Alan and C. Wallnau Kurt. A Framework for Systematic Evaluation of Software Technologies. *IEEE Software*, 15213(September), 1996.

[37] Longbing Cao, Chengqi Z, and Ni Jiarui. Agent services-oriented architectural design of open complex agent systems. In *Intelligent Agent Technology, IEEE/WIC/ACM International Conference on. IEEE,*, 2005.

[38] L Cernuzzi. On The Evaluation Of Agent Oriented Modeling Methods. In *the OOPSLA (2002) Workshop on Agent-Oriented Methodologies.*, Seattle, 2002.

[39] Luca Cernuzzi. Profile Based Comparative Analysis For Aose Methodologies Evaluation. *Proceedings of the 2008 ACM symposium*, pages 60–65, 2008.

[40] A. Chella, M. Cossentino, L. Sabatucci, and V. Seidita. Agile PASSI: An Agile Process for Designing Agents. *International journal of computer systems science & engineering. special issue on software*, (i):1–6, 2004.

[41] Antonio Chella, Viale Scienze, Massimo Cossentino, Luca Sabatucci, Valeria Seidita, Alte Prestazioni, and Consiglio Nazionale. From Passi To Agile Passi : Tailoring A Design Process To Meet New Needs.

[42] Chia-En Lin, Krishna M. Kavi and Frederick T. Sheldon and Thomas E. Potok. A Methodology To Evaluate Agent Oriented Software Engineering Techniques. In *40th Annual Hawaii International Conference on System Sciences, HICSS 2007*, pages 1–20, Island of Hawaii,USA, 2007. IEEE Computer Society.

[43] Mark Claypool and Kajal Claypool. Latency Can Kill : Precision and Deadline in Online Games. pages 215–222, 2010.

[44] G. Costikyan. I have no words & i must design. In *In Interactive Fantasy.*, number 2, 1994.

[45] Chris Crawford. *Chris Crawford on game design.* New Riders, 2003.

[46] Khanh Dam. *Evaluating And Comparing Agent-oriented Software Engineering Methodologies*. PhD thesis, School of Computer Science and Information Technology, RMIT University, Australia., 2003.

[47] Michael Dam, Khanh and Winikoff. Comparing Agent-oriented Methodologies. In *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems (AAMAS). Lecture Notes in Computer Science; Springer*, volume 3030, pages 78–93, Melbourn, Australia., 2004. Springer Berlin / Heidelberg.

[48] S DeLoach. Analysis and Design using MaSE and agentTool. In *In Proceedings of 12 Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001), Miami University, Oxford, Ohio*, pages 1–7, 2001.

[49] S. DeLoach. Multiagent Systems Engineering Of Organization-based Multiagent Systems. *ACM SIGSOFT Software Engineering Notes*, 30(4), July 2005.

[50] S DeLoach and J Garcia-Ojeda. O-mase: A Customisable Approach To Designing And Building Complex, Adaptive Multi-agent Systems. *International Journal of Agent-Oriented Software Engineering*, 4(3):244–280, 2010.

[51] S. DeLoach, E. Matson, and Y. Li. Applying Agent Oriented Software Engineering To Cooperative Robotics. *Proceedings of the 15th International FLAIRS Conference(FLAIRS 2002). Pensacola, Florida*, 2002.

[52] S. DeLoach and J Valenzuela. An agent-environment interaction model. *Agent-Oriented Software Engineering VII. Springer Berlin Heidelberg*, 4405:1–18, 2007.

[53] H. Desurvire, M. Caplan, and J. Toth. Using Heuristics to Evaluate the Playability of Games. In *CHI 2004 Late Breaking Results Paper*, pages 1509–1512, Vienna, Austria, 24-29 April 2004.

[54] Sebastian Deterding, Dan Dixon, R Khaled, and L Nacke. From game design elements to gamefulness: defining gamification. *Proceedings of the 15th ...*, 2011.

[55] Damien Djaouti, Julian Alvarez, Jean-Pierre Jessel, Gilles Methel, and Pierre Molinier. A Gameplay Definition through Videogame Classification. *International Journal of Computer Games Technology*, 2008:1–7, 2008.

[56] S. Easterbrook, J. Singer, and M. Storey. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering. Springer.*, pages 285–311., London, 2008.

[57] M. S. El-Nasr and B. K. Smith. Learning through game modding. *Computers in Entertainment*, 4(1):45– 64, 2006.

[58] P Escudeiro and Nuno Escudeiro. Evaluation of Serious Games in Mobile Platforms with QEF: QEF (Quantitative Evaluation Framework). In *2012 Seventh IEEE International Conference on Wireless, Mobile and Ubiquitous Technology in Education*, pages 268–271, Takamatsu, Kagawa, Japan, 27-30 March 2012.

[59] M. Fasli. *Agent Technology for E-commerce*. Wiley & Sons, 2007.

[60] A. Febretti and F. Garzotto. Usability, Playability, And Long-term Engagement In Computer Games. In *CHI 2009*, pages 4063–4068, Boston, MA, USA, April 4-9 2009. ACM.

[61] M. Federoff. Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games. Master of science thesis, Indiana University, 2002.

[62] J. Flynt and O. Salem. *Software Engineering for Game Developers*. Course Technology Ptr, November 2005.

[63] B. A. Foss and T. I. Eikaas. Game play in engineering education concept and experimental results. *International Journal of Engineering Education*, 22(5):1043–1052, 2006.

[64] A. Gershenfeld, M. Loparco, and C. Barajas. *Game Plan: The Insiders Guide To Breaking In And Succeeding In The Computer And Video Game Business.* St. Martins Griffin Press,New York, 2003.

[65] A Godoy and EF Barbosa. Game-Scrum: An Approach to Agile Game Development. In *Proceedings of SBGames 2010 Computing*, pages 292–295, November 8th-10th 2010.

[66] Gomez, A. and Gonzalez, J. and Ramos, D. and Vazquez, L. Modeling Serious Games Using Aose Methodologies. In *11th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 53–58, 2011.

[67] Paul Guyot and Shinichi Honiden. Agent-Based Participatory Simulations : Merging Multi-Agent Systems and Role-Playing Games. *The Journal of Artificial Societies and Social Simulation*, 9(4):8–20, 2006.

[68] A. Hevner, S. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004.

[69] Michael N. Huhns and Munindar P. Singh. Service-oriented computing: Key concepts and principles. In *Internet Computing, IEEE 9.1*, pages 75–81, 2005.

[70] LA Hunt. Predictive And Adaptive Game Development A Practical Application Of Development Models To The Independent Video Game Industry. Master's thesis, School of Communications and Arts, 2011.

[71] C Iglesias and Mercedes Garijo. Analysis And Design Of Multiagent Systems Using Mascommonkads. *Intelligent Agents IV Agent*, 1365:313–327, 1998.

[72] JP Jamont and Michel Occello. Designing Embedded Collective Systems: The Diamond Multiagent Method. *19th IEEE International Conference on Tools with Artificial Intelligence*, 2:91–94, 2007.

[73] Nimal Jayaratna. *Understanding and Evaluating Methodologies: NIMSAD a Systematic Framework.* McGraw-Hill, New York, 2nd edition edition, 1994.

[74] R. Jeffries and J. Miller. User interface evaluation in the real world: A comparison of four techniques. In *Conference on Human Factors in Computing Systems CHI 91*, pages 119–124., New Orleans, LA, April 1991. ACM Press.

[75] Sycara-K. Jennings, N.R. and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[76] Jesse Schell. *The Art of Game Design:A Book of Lenses*. Morgan Kaufmann, 2008.

[77] Daniel Johnson and Janet Wiles. Computer Games With Intelligence . *Australian Journal of Intelligent Information Processing Systems*, 7:61–68, 2001.

[78] C Kanode and M Haddad. Software Engineering Challenges in Game Development. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 260–265. IEEE Computer Society, 2009.

[79] Effie Karouzaki and Anthony Savidis. A Framework for Adaptive Game Presenters with Emotions and Social Comments. *International Journal of Computer Games Technology*, 2012:1–18, 2012.

[80] C Keith. *Agile Game Development With Scrum*. Addison-Wesley Signature Series, 2010.

[81] C Keith and TY AGO. Scrum Rising. *Agile Development could save your studio. Game Developer*, 14:22–26, 2007.

[82] B. Kitchenham, S Linkman, and D. Law. Desmet: A methodology for evaluating software engineering methods and tools. *Computing and Control Engineering Journal*, 8.3:120–126., 1997.

[83] B. Kitchenham, L. Pickard, and S. Pfleeger. Case studies for method and tool evaluation. *Software Magazine*, 4(12):52–62, 1995.

[84] Barbara Kitchenham. Desmet: A method for evaluating software engineering methods and tools. Technical Report TR96-09, University of Keele, U.K, August 1996.

[85] H. Korhonen and E. Koivisto. Playability Heuristics For Mobile Games. In *MobileHCI*, number 9-16, page 9, Helsinki, Finland, September 12-15 2006. ACM Press.

[86] H. Korhonen, J. Paavilainen, and H. Saarenpaa. Expert Review Method In Game Evaluations: Comparison Of Two Playability Heuristic Sets. In *MindTrek 2009*, pages 74–81, Tampere, Finland, September 30-October 2 2009. ACM.

[87] R Kortmann and Casper Harteveld. Agile Game Development: Lessons Learned From Software Engineering. In *Proceedings of the 40th conference of the international simulation and gaming association*, 2009.

[88] J. Laird and M. Van. Human-level ai's killer application: Interactive computer games. *The AI Magazine*, 22:15–25, 2000.

[89] Alexei Lapouchnian. Modeling Mental States in Requirements Engineering : An Agent-Oriented Framework Based on i * and CASL. Master's thesis, York University, Toronto, Canada, July 2004.

[90] D. Law and T. Naem. Desmet: Determining and evaluation methodology for software methods and tools. In *the Science Conference on CASE - Current Practice, Future Prospects*, Cambridge, England, March 1992.

[91] Chris Lewis and Jim Whitehead. The Whats And The Whys Of Games And Software Engineering. In *Proceedings of the 1st International Workshop on Games and Software Engineering, GAS '11*, pages 1–4, New York, New York, USA, May 2011. ACM Press.

[92] By Craig A Lindley. Game Taxonomies : A High Level Framework for Game Analysis and Design. *Framework*, pages 1–10, 2003.

[93] Ashri R. DInverno M Luck, M. *Agent-Based Software Development.* Artec House Inc, London, 2004.

[94] Daniel . Macedo, Formico Rodrigues, and Maria A. Experiences With Rapid Mobile Game Development Using Unity Engine. *Computers in Entertainment*, 9(3):1–12, November 2011.

[95] I. Magarifto, J. Jorge, and R. Sanz. An evaluation framework for MAS modeling languages based on metamodel metrics. In *Agent-Oriented Software Engineering IX, 9th International Workshop, AOSE*, volume 5386, pages 101–115, Estoril, Portugal, May 12-13, 2008 2009. Lecture Notes in Computer Science-Springer.

[96] Kevin. Maroney. My entire waking life. *The games journal*, 5, 2001.

[97] R McGuire. Paper Burns: Game Design With Agile Methodologies. *Gamasutra: The Art and Business of Making Games.*, pages 1–7, 2006.

[98] Qinghai Miao, Fenghua Zhu, Yisheng Lv, Changjian Cheng, Cheng Chen, and Xiaogang Qiu. A Game-Engine-Based Platform for Modeling and Computing Artificial Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):343–353, June 2011.

[99] M.Nachamai, M.Senthil, and V.Tapaska. Enacted Software Development Process Based On Agile And Agent. *International Journal of Engineering Science and Technology (IJEST)*, 3(11):8019–8029, November 2011.

[100] A. Molesini and V. Andrea, O. and. Mirko. Environment in agent-oriented software engineering methodologies. *Multiagent and Grid Systems - Engineering Environments in Multiagent Systems*, 5:37–57, January 2009.

[101] B. Morse and P. Field. Qualitative research methods for health professionals. In *SAGE Publications London*, 1995.

[102] Haralambos Mouratidis. Secure Tropos: An Agent Oriented Software Engineering Methodology for the Development of Health and Social Care Information Systems. *International Journal of Computer Science and Security*, 3(3):241–271, 2009.

[103] Lennart Nacke. From playability to a hierarchical game usability model. In *Conference on Future Play - FuturePlay '09*, pages 10–11, Vancouver, BC, Canada, 2009. ACM Press.

[104] Brian Mac Namee and P Cunningham. A Proposal for an Agent Architecture for Proactive Persistent Non Player Characters. In *the 12 th Irish Conference on AI and Cognitive Science*, pages 221–232, 2001.

[105] J. Nielsen and R. Mack. Heuristic Evaluation Of User Interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, number April, pages 249–256. Empowering people. ACM, 1990.

[106] H.S. Nwana. Software agents: An overview. *Intelligent Systems Research , Advanced Applications & Technology Dep. , Cambridge university U.K. Knowledge Engineering Review*, 11:1–40, Sept 1996.

[107] HM Omar, Roslina Ibrahim, and Azizah Jaafar. Methodology To Evaluate Interface Of Educational Computer Game. In *2011 International Conference on Pattern Analysis and Intelligent Robotics*, number 28-29 June, pages 228–232, Putrajaya, Malaysia, 2011. IEEE.

[108] Janne Paavilainen. Critical Review On Video Game Evaluation Heuristics: Social Games Perspective. In *Future Play 2010*, pages 56–65. ACM Press, 2010.

[109] L Padgham. Prometheus: A Methodology For Developing Intelligent Agents. *Agent-oriented software engineering III*, 2003.

[110] Panagiotis Petridis, Ian Dunwell, David Panzoli, Sylvester Arnab, Aristidis Protopsaltis, Maurice Hendrix, and Sara Freitas. Game Engines Selection Framework for High-Fidelity Serious Applications. *International Journal of Interactive Worlds*, 2012:1–19, June 2013.

[111] Fábio Petrillo and Marcelo Pimenta. Houston, We Have A Problem...: A Survey Of Actual Problems In Computer Games Development. In *In SAC08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 707–711. ACM, 2008.

[112] Fabio Petrillo and Marcelo Pimenta. Is Agility Out There?: Agile Practices In Game Development. *SIGDOC '10 Proceedings of the 28th ACM International Conference on Design of Communication*, pages 9–15, 2010.

[113] D. Pinelle, and Stach T. Wong, N., and Gutwin. Usability Heuristics For Networked Multiplayer Games. In *Proc. of the ACM 2009 International Conference on Supporting Group Work*, pages 169–178, Sanibel Island, Florida, USA., May 10-13 2009.

[114] David Pinelle, Union Street, and Goodwin Hall. Heuristic Evaluation for Games : Usability Principles for Video Game Design. In *SIGCHI Conference on Human Factors in Computing Systems. ACM*, pages 1453–1462, 2008.

[115] G. Pleva. Game programming and the myth in a childs play. *Journal of Computing Sciences in Colleges*, 2:125–136, 2004.

[116] Marcb Ponsen and P. Spronck. Improving Adaptive Game AI With Evolutionary Learning. *Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*, pages 389–396 University of Wolverhampton., 2004.

[117] MG Salazar and HA Mitre. Proposal Of Game Design Document From Software Engineering Requirements Perspective. In *The 17th International Conference on Computer Games*, pages 81–85. IEEE, 2012.

[118] E. Salen and K. Zimmerman. *Rules of Play: Game Design Fundamentals.* MIT Press, 2003.

[119] Amin Saremi and Mostafa Esmaeili. Evaluation Complexity Problem In Agent Based Software Development Methodology. *Industrial and Information*, (August):577 –584, 2007.

[120] N. Schaffer. Heuristics for usability in games. Technical report, Rensselaer Polytechnic Institute, White Paper., April 2007.

[121] Jonas Schild, Robert Walter, and M Masuch. ABC-sprints: Adapting Scrum To Academic Game Development Courses. In *the Foundations of Digital Games*, pages 187–194, 2010.

[122] Tong ShaoPeng and Zhang Jun. A Research on Multi Agent Modeling Language. *Procedia Engineering*, 15:1842–1847, January 2011.

[123] Y Shoham. Agent- Oriented Programming. *Artificial intelligence*, 60.1:51–92, 1993.

[124] G. Sindre, L. Natvig, and M. Jahre. Experimental validation of the learning effect for a pedagogical game on computer fundamentals. *IEEE Transactions on Education*, 52(1):10–18, 2009.

[125] R Sousa, A. da Cunha, L Martins, R. Cysneiros, and L. Werneck. Evaluating MaSE Methodology in the Requirements Identification. In *the 33nd Annual IEEE Software Engineering Workshop, IEEE Computer Society Press, Skovde*, pages 136–143, October 2010.

[126] Pieter Spronck, Marc Ponsen, and I Sprinkhuizen-Kuyper. Adaptive Game AI With Dynamic Scripting. *Machine Learning*, 63(3):1–42, 2006.

[127] P. Stacey and J. Nandhakumar. Managing projects in a games factory: Temporality and practices. In *38th Hawaii Interna- tional Conference on System Sciences*, pages 1–10, 2005.

[128] Janusz A. Starzy k and Pawel Raif. Cognitive agent and its implementation in the blender game engine environment. In *Computational Intelligence for Human-like Intelligence (CIHLI), 2013 IEEE Symposium on. IEEE*, 2013.

[129] Peter Stone, Park Ave, and Florham Park. Multiagent Systems : A Survey from a Machine Learning Perspective. *Robotics*, pages 1–57, 2000.

[130] Arnon Sturm. A Framework For Evaluating Agent-oriented Methodologies. *Agent-Oriented Information Systems*, (1):60–67, 2004.

[131] Arnon Sturm and O Shehory. A Framework For Evaluating Agent-oriented Methodologies. In *Agent-Oriented Information Systems, 5th Int. Bi-Conference Workshop, AOIS 2003. Lecture Notes in Computer Science 3030, Springer-Verlag.*, pages 94–109, 2004.

[132] Jan Sudeikat and Lars Braubach. Evaluation Of Agent Oriented Software Methodologies Examination Of The Gap Between Modeling And Platform. In *Agent-Oriented Software Engineering, Lecture Notes in Computer Science*, volume 3382, pages 126–141, Berlin, Germany, 2005. Springer Verlag.

[133] P Sweetser. Current AI in Games: A review. Technical report, School of ITEE, University of Queensland, 2002.

[134] Penelope Sweetser and Peta Wyeth. Gameflow: A Model For Evaluating Player Enjoyment In Games. *Computers in Entertainment (CIE)*, 3:1–24, 2005.

[135] Victor Szalvay. An Introduction to Agile Software Development. *Danube Technologies, Inc., Bellevue*, (November):1–9, 2004.

[136] H. Takeuchi and I. Nonaka. The new product development game. In *Harvard Business Review*, pages 137–146., January- February 1986.

[137] QNN Tran and Graham C. Low. Comparison Of Ten Agent-oriented Methodologies. *Agent-oriented methodologies*, pages 341–367, 2005.

[138] I Trencansky and R Cervenka. Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS. *Informatica*, 29:391–400, 2005.

[139] A. Wang, T. Fsdahl, and O. Morch-Storstein. An evaluation of a mobile game concept for lectures,. In *21st Conference on Software Engineering Education and Training,(CSEET 08)*, number 197-204, 2008.

[140] A. Wang, O March-Storstein, and T Fsdahl. Lecture quiz mobile game concept for lectures,. In *11th IASTED International Conference on Software Engineering and Application (SEA 07)*, November 2007.

[141] Alf Wang and Bian Wu. Using Game Development to Teach Software Architecture. *International Journal of Computer Games Technology*, 4:1–12, 2011.

[142] Charles Weddle. Artificial Intelligence and Computer Games. *Dissertation, Computer Science, Florida State University.*, pages 1–8, 2008.

[143] C. Wohlin. *Experimentation in Software Engineering*. Springer, Boston/Dordrecht/London, 2012.

[144] Mark Wood and Scott Deloach. An Overview of the Multiagent Systems Engineering Methodology. *Computer*, 1957(January):207–221, 2001.

[145] M. Wooldridge. *An Introduction to MultiAgent Systems.* Wiley, 2002.

[146] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. A Methodology For Agent-oriented Analysis And Design. *Proceedings of the third annual conference on Autonomous Agents - AGENTS '99*, pages 69–76, 1999.

[147] Michael Wooldridge, Chester Street, M Manchester, Nicholas R Jennings, Mile End Road, and E London. Intelligent Agents : Theory and Practice. In *Electronic Engineering*, number January, pages 1–62, 1995.

[148] B. Wu and A. I. Wang. An evaluation of using a game development framework in higher education. In *22nd Conference on Software Engineering Education and Training, (CSEET 09)*, pages 41–44, Hyderabad, India, February 2009.

[149] Bian Wu and Alf Inge Wang. A Guideline for Game Development-Based Learning: A Literature Review. *International Journal of Computer Games Technology*, 2012:1–20, 2012.

[150] B Yue. The State Of The Art In Game Ai Standardisation. *the international conference on Game research and development*, 223:41–46, 2006.

[151] M. Zyda. From visual simulation to virtual reality to game. In *Computer*, volume 38, pages 25–32, 2005.