CENG 491

SOFTWARE REQUIREMENTS SPECIFICATION FOR THE ELERON

SAMET CAN - 1678812
EMRE AKSAN – 1678655
ÇAĞLAR GÜNDOĞDU - 1679000
OKAN HARPUT - 1679042

17.01.2013

Preface

This document contains the software requirements specification for the “The Eleron” software. The document is prepared according to the “830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.”. This Software Requirements Specification provides a complete description of all the software requirements and views of the “The Eleron”.

Version	Date	Sections Changed, Added	Type of Change*	Brief Description
1.0	11.11.2012	-	A	Full document is written
2.0	17.01.2013	2.6.1 3.1.1.2 2.6.1.2 2.6.1.1 2.6.1.7 2.6.1.9 2.6.1.13 2.6.1.14 2.6.2.2 2.6.2.4 2.6.2.5 3.1.1.1 3.1.1.7 3.1.2.6 3.1.2.7 3.1.3.5 3.1.3.3 3.1.4.2 3.1.4.3 3.1.4.5 2.6.1.7 3.1.2.2 3.2.6	M M M D D D D D D D D D D D D D D D D D A A A	M and D sections are according to v1.0 A sections are according to v2.0

* A: Added, M: Modified, D: Deleted

Table of Contents

1 Introduction

1.1 Problem Definition

A lot of humans, animals, vehicles are interacting with each other on a daily basis and these interactions may vary in a large-scale. Often understanding and estimating the behaviors of these agents is a hard task. To understand, estimate the interactions of these agents in a large crowd or even visualization is a common problem due to some reasons such as cost of real demonstrations, security issues, etc. For these very specific cases where visualization of the behaviors of agents is important simulation technology helps us.

While imitating the real world and gathering realistic responses are possible, simulating many agents can be costly and usually ineffective with common simulation techniques. The need for crowd simulation arises when a scene calls for more characters than can be practically animated using conventional systems.

1.2 Purpose

This Software Requirements Specification document supplies a comprehensive description of all the functions and specifications High Level Architecture of Crowd Simulation under the collaboration with ROKETSAN and CEng 490 course. This document will be used as a guideline in the development state of the project. Since this document is a first release, there may be some modifications to adapt changes of requirements and specifications in the system.

The target audience of this document includes all users who are interested in simulation applications and games and prospective software developers associated with the project.

1.3 Scope

The software to be produced is “The Eleron”. This software will be a show case for the software that enables communication between different simulation technologies. The game will include crowd simulation, artificial intelligence and multi-player concept.

In this game, spacecraft, planets and meteors will be supported in 3D. In the game, two or more players will fight against each other as two teams. These teams also include AI controlled units. In pre-game, the player who created the game is expected to specify some predefined rules. These rules refer environment of the game, number of AI agents, difficulty of the game and spacecraft types.

In this version of the document, due to the changes in show case, some requirements and use cases are no longer valid. Therefore, they are extracted.

1.4 Glossary

The following is a list of terms, acronyms and abbreviations.

Term	Abbreviation
RTI	Run-time Infrastructure
HLA	High Level Architecture
AI	Artificial Intelligence
IEEE	The Institute of Electrical and Electronics Engineers
FOM	Federation Object Model

Term	Definitions
AI-Agent	Units in the game controlled by AI server
Bot	Units in the game controlled by AI server
Federate	An HLA compliant simulation entity
Federation	Collection of federates connected via the RTI

1.5 References

IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

1.6 Overview

This document contains two main additional chapters. The first, The Overall Description, describes the process model, pre-research phase, main structure of the system, the interface between main modules and use cases. The next chapter basically includes all requirement specifications about the project, inspected in different components of the project.

2 Overall Description

2.1 Process Model

One of the most important parts of the software engineering is deciding the process model of the project that will be developed, because the process model has significant effects on the overall project. In the Ceng491 course, all the development steps are specified with their deadlines, so it has sequential structure. For this reason, the waterfall model may seem most appropriate model at the first instance. However, there will be minor changes throughout the process, but they will continuously occur. So, using spiral model, which combines incremental development and the waterfall model, can be thought as more convenient model. Requirements or wishes of customer may change during the development process as project team gather with customer, therefore the minor changes becomes inevitable. Also, specifications of the project are not concluded with the exact decision, so it is clear that there will be changes and improvements. All these reasons lead to apply spiral model. Obviously, backtracking earlier stages and making alterations is

impracticable, but these changes are necessary. Some releases and documents will be formed in time as in incremental model, however spiral model provides that features of product will overlap with requirements.

2.2 Team Organization

Team structure of the software team has also significance on the project development as in process model. There is no hierarchy in the team so all decisions can be taken collectively. Since there is no hierarchy, all team members are at the same level, so horizontal communication method is used. Role of each team member decided randomly in most of the time and it is exchangeable if there is demand for that.

2.3 Research

2.3.1 Literature Research and Analysis

To be able to choose suitable tools that will be used in the project, detailed research made on the Internet.

2.3.1.1 RTI

In this project HLA architecture will be used. It enables communication of several systems with each other. The main operation made on run-time infrastructure which is underlying technology of HLA architecture. We have searched several RTI software on the internet. First one was MAK High Performance RTI and it has detailed documentation including information about how to make communication with UDK. At first instance, it seems nice chose, but MAK High Performance RTI is commercial program, therefore we start to search non-commercial RTI software such as CERTI and Portico. They were both seem like suitable choices, but by considering majority of example applications and available documentation about the software, we decided to use Portico as RTI in this project.

2.3.1.2 Graphic and Game Engines

Unity: Unity is the advised game engine for the beginners. It has huge community, so the opportunity for taking support is easier, also it can work on almost all operating system. It is possible to import Google Earth terrains into unity3d. In our search we could not discover any project that connects unity and HLA.

UDK : UDK is freeware game engine and it is easiest game engine in the issues of finding support after unit3d. It uses Javascript and unreal script. It has unreal Kismet software which makes possible to design level of game without even changing one line in the code. In the video quality, UDK is better than other open source game engines.

Irrlicht: Irrlicht, written in C++, is open source graphic rendering engine. It can work on almost every platform which is basically Windows, MacOS, Linux, iPhone, XBOX, Symbian OS, etc. It uses zlib license in which one can make any changes in the source code and use that new code for commercial purposes without informing or reporting these changes to Irrlicht developers. It supports much more rendering API than other 3D graphic engines (DirectX 8, DirectX9, OpenGL, etc.). The documentation and community of Irrlicht engine is excellent. There are lots of tutorials and examples in their website and those examples start with

HelloWorld to advanced examples. But, Irrlicht is not a game engine and for that reason in needs external libraries like sound and physic engines.

2.3.2 Meeting with ROKETSAN

The scenario decision made after meeting with ROKETSAN according to their desires. Also the tools to be used evaluated at the meeting and their pros and cons specified, and as a result tools will be used are decided.

2.4 System Overview

2.4.1 High Level Architecture

A high-level architecture (HLA) is a general purpose architecture for distributed computer simulation systems. Using HLA, computer simulations can interact (that is, to communicate data, and to synchronize actions) with other computer simulations regardless of the computing platforms. The interaction between simulations is managed by a Run-Time Infrastructure (RTI).

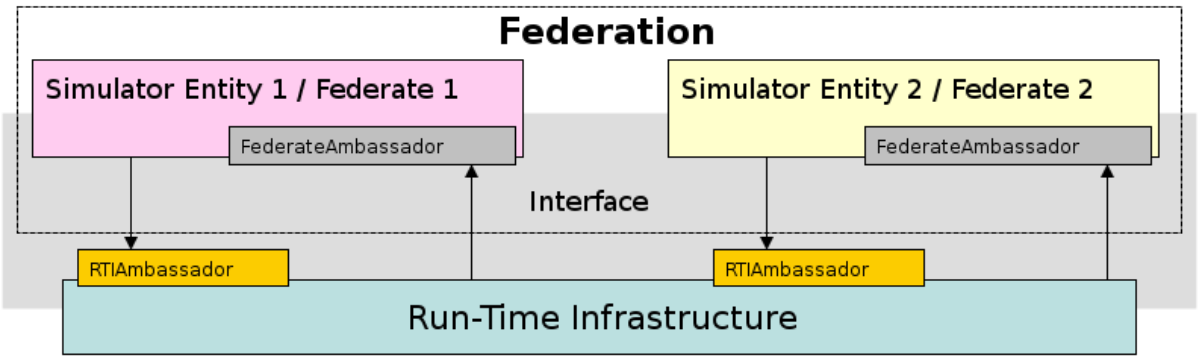


Figure 1: HLA-RTI Overview

Our system is basically constructed on a HLA-RTI middleware which is required to implement HLA. Over the RTI, game engines and AI engine will be able to communicate. Each game engine and the AI engine corresponds to a federate. According to requests from game engines, new state is calculated by AI server and sent back to the game engines.

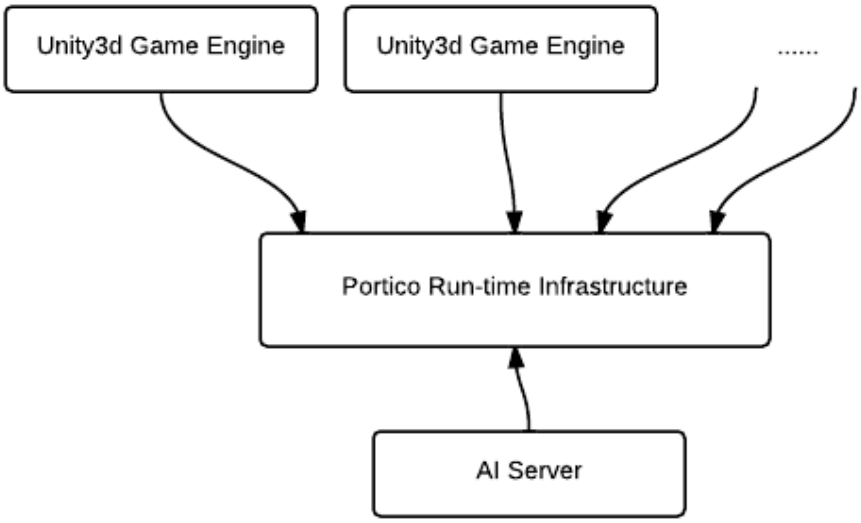


Figure 2: System Overview

2.4.1.1 Portico Run-Time Infrastructure

Among other alternatives such as MAK, Certi, etc. Portico is decided to use as RTI engine. Portico is an open source RTI software and supports implementation in Java language. It has also a large community and good documentation. Portico uses IEEE 1516 interface.

2.4.1.2 Unity3D Game Engine

Unity3D provides environment for modelling, graphics rendering, animation, sound and physics engine. There are at least two game engines for different players. Each player controls a group of planes from their camera viewpoint. For each player each game engine continuously interacts with RTI to update simulation state. For each action/event request such as change direction, fire, etc. the request and state of the simulation in game engines is informed to AI server over RTI. The response is new state and it is sent to game engines to simulate the scene.

2.4.1.3 AI Server

In the game, there are AI controlled planes that follows instructions of the players. The bots' responses to the events/actions are determined by the AI server. With the parameters sent from game engines, for each AI agent, possible moves are generated and sent back to the game engines. The bots' interactions can be defined by the players, which affect response of the AI server.

2.5 Constraints

2.5.1 Project Schedule

Since this is a course project, the project has to follow certain deadlines. During the limited time, basic knowledge about HLA architecture, RTI dynamics, Unity3D game engine, 3D modeling and realizing AI agents must be gained. The project documents like SRS, SDD or STD must strictly obey the deadlines. Mean time, the implementation of the project and after the implementation, the testing phase should be handled. The project must be delivered at the end of second semester, June 2012.

2.5.2 Interface between External Systems

Since the project is originally a simulation link project that gathers different external systems, a generalized bridge between the external systems must be implemented. HLA-RTI providing an interface between different systems should be generalized that will be able to run with different federations (FOM objects) of same external systems.

2.5.3 Execution Speed for Synchronization

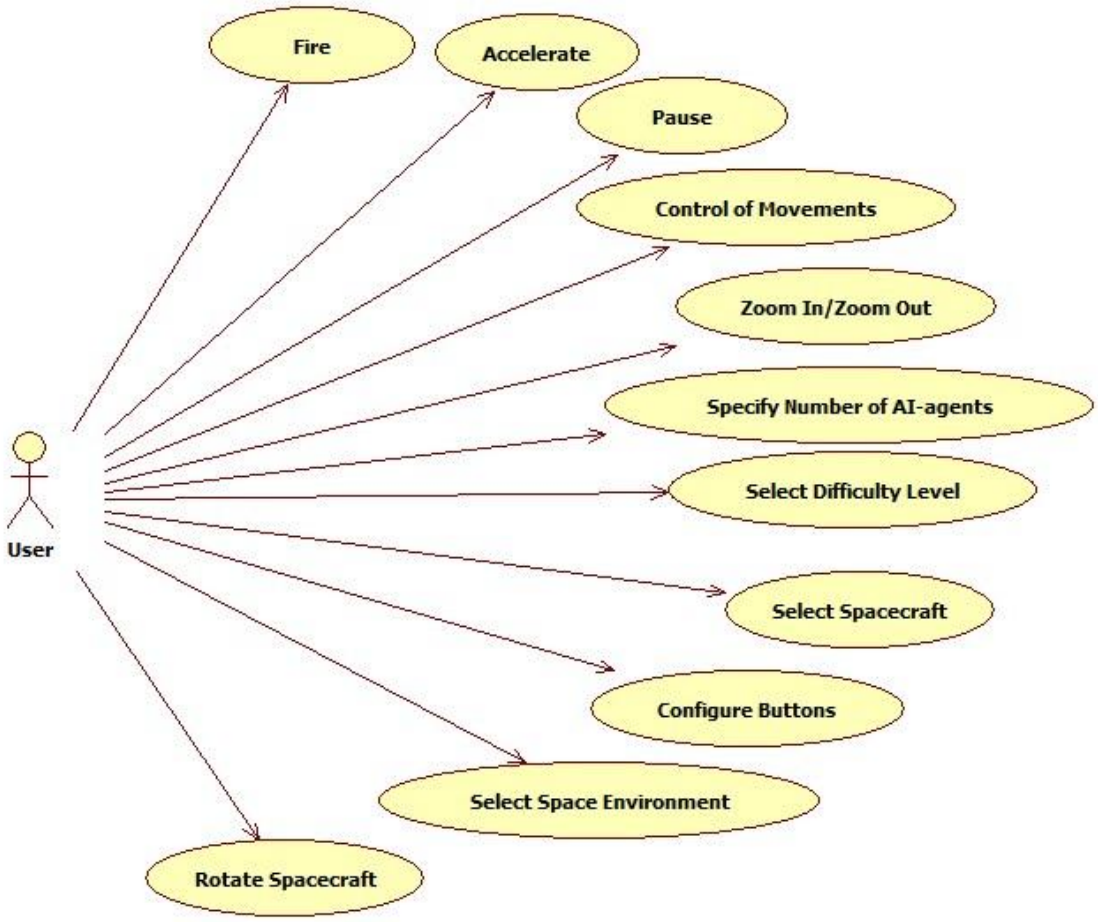
The system gathers number of different systems and there must be synchronization between them. There will be at least two players and their states will be updated real-timely. The server must get state and event information from different game engines and calculates new states of the players and the AI server must also update AI-based agents' status in the mean time. The server should bear up against all these challenging interactions.

2.5.4 3D Modeling

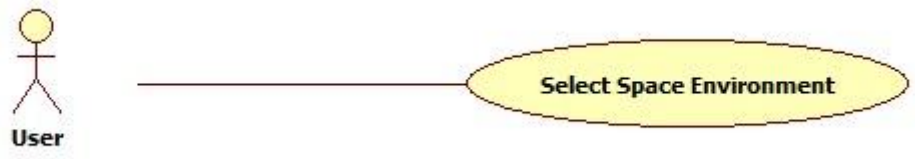
Modeling of the all objects is a tedious work. Due to time limit and lack of experience to create 3D models, ready-to-use models may be used for some objects.

2.6 Product Function

2.6.1 User Use Cases

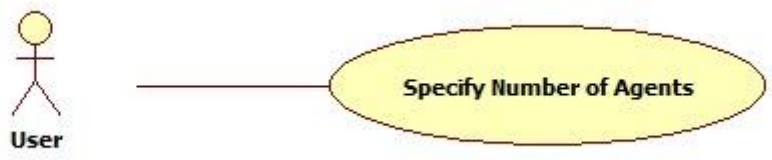


2.6.1.1 Use Case UC01: Environment Determination



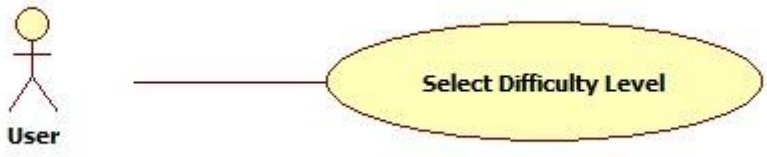
Brief Description	The User adjusts the simulation environment.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already opened environment selection screen. 1. Possible space environment selection options shown on the screen, user can select one of them just by one click.
Xref	Section 3.1.1.2, Environment Determination

2.6.1.2 Use Case UC02: Agent Number Specification



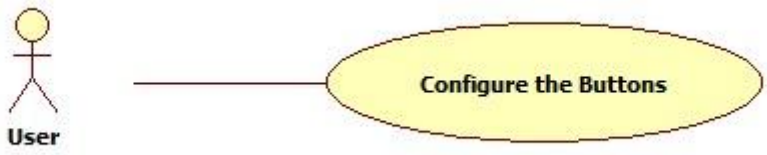
Brief Description	The User specifies number of agents.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already opened agent number specification screen. 1. The user enters the desired number of agents from the keyboard. 2. In the simulation, there would be exactly specified number of agents.
Xref	Section 3.1.1.3, Agent Number Specification

2.6.1.3 Use Case UC03: Difficulty level selection



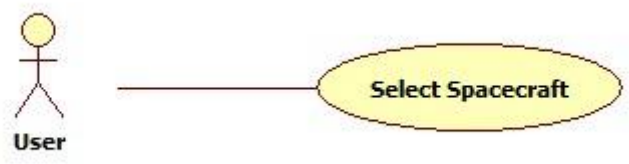
Brief Description	The User selects the difficulty level.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already opened difficulty level selection screen. 1. The user selects one of the difficulty level which are low, medium and high 2. The game difficulty is set according to the user choice.
Xref	Section 3.1.1.4, Difficulty level selection

2.6.1.4 Use Case UC04: Button Configuration



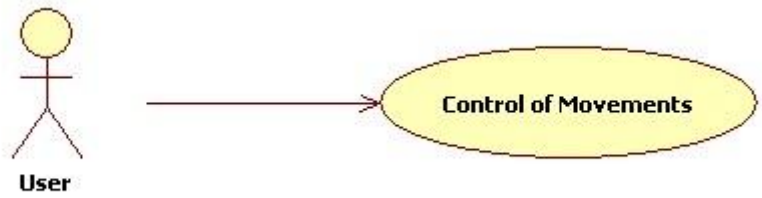
Brief Description	The User configures the buttons.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already opened button configuration screen. 1. The user selects the functionality which one keyboard button will be assigned. 2. The user press one of the buttons on the keyboard to specify the button for functionality selected at step 1.
Xref	Section 3.1.1.5, Button Configuration

2.6.1.5 Use Case UC05: Spacecraft Selection



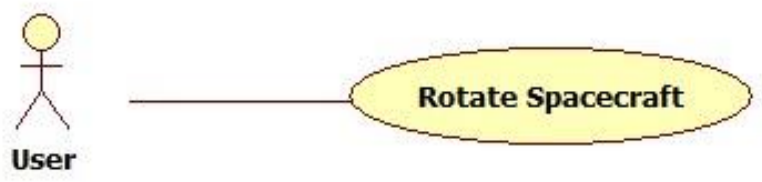
Brief Description	The User selects aircraft type or model.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already opened aircraft selection screen. 1. The user selects the spacecraft type or model according to his / her desires with just one mouse click
Xref	Section 3.1.1.6, Spacecraft Selection

2.6.1.6 Use Case UC06: Control of Movements



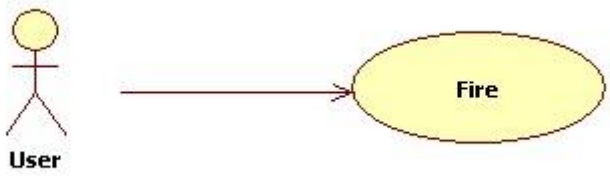
Brief Description	The User directs the selected character direction.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already started the game(simulation) 1. During the simulation, the User presses the “w”,”a”,”s” and ”d” buttons of keyboard. 2. Depending on pressed button, the selected character’s direction change accordingly.
Xref	Section 3.1.2.1,Control of Movements

2.6.1.7 Use Case UC07: Rotation about direction axes



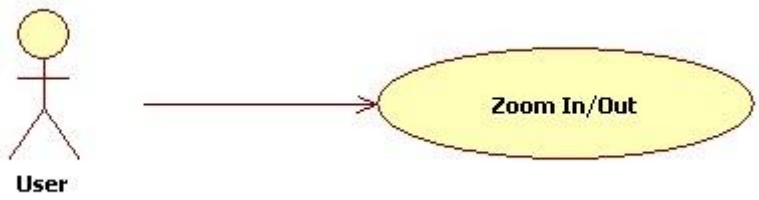
Brief Description	The User directs the selected character rotation.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already started the game(simulation) 1. During the simulation, the User presses the “q” and “e” buttons of keyboard. 2. Depending on pressed button, the selected character’s rotation change according to direction axes.
Xref	Section 3.1.2.2,Rotation about direction axes

2.6.1.8 Use Case UC08: Fire



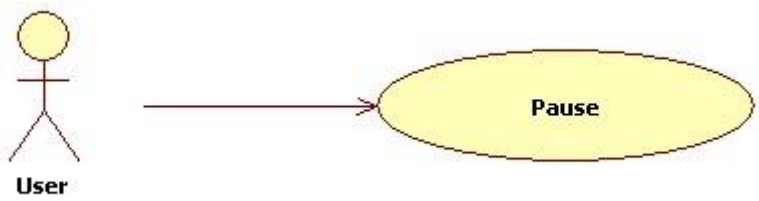
Brief Description	The User fires to the targets.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already started the game(simulation) 1. During the simulation, the User moves mouse. 2. The user takes aim at targets. 3. Then he/she clicks one of the mouse buttons. 4. Depending on clicked button, the selected character uses primary or secondary weapon and fires.
Xref	Section 3.1.2.3,Fire

2.6.1.9 Use Case UC09: Zoom In/Out



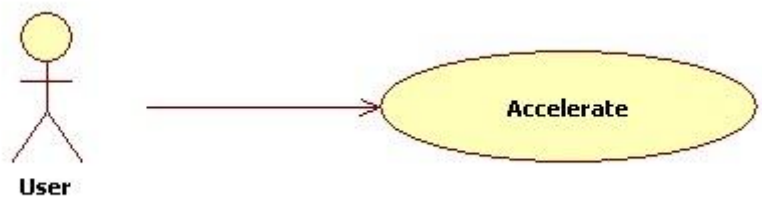
Brief Description	The User zooms in/out the view.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already started the game(simulation) 1. During the simulation, the User rolls mousewheel. 2. Depending on which dimension of mousewheel is rolled, the view is zoomed in/out.
Xref	Section 3.1.2.4,Zoom In/Out

2.6.1.10 Use Case UC10: Pause



Brief Description	The User pauses game(simulation).
Initial Step-By-Step Description	Before this use case can be initiated, the User has already started the game(simulation) 1. During the simulation, the User presses “Esc” button of keyboard. 2. The game is paused.
Xref	Section 3.1.2.5,Pause

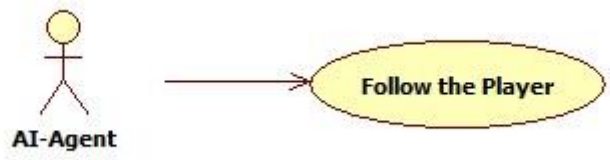
2.6.1.11 Use Case UC11: Accelerate



Brief Description	The User accelerates the speed of selected character.
Initial Step-By-Step Description	Before this use case can be initiated, the User has already started the game(simulation) 1. During the simulation, the User keeps press “space” button of keyboard. 2. The speed increases unless it is equal to the max speed.
Xref	Section 3.1.2.8,Accelerate

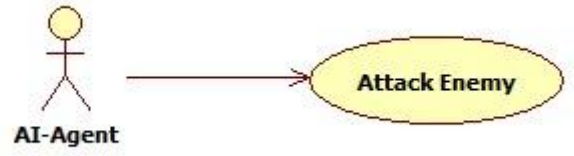
2.6.2 AI Use Cases

2.6.2.1 Use Case AI01: Follow the Player



Brief Description	The bots will be able to move according to the player.
Initial Step-By-Step Description	If the AI-agent is not taken any command rather than following to player: 3. If the player changes its velocity, direction or altitude, the AI server is informed with these changes. 4. The AI server computes new state for bots according to player parameters and pays attention to keep formation, distance between units same. 5. The AL agent follows the user.
Xref	Section 3.1.4.1,Follow the Player

2.6.2.2 Use Case AI02: Attack Enemy



Brief Description	The bots will be able to attack to the enemy if an enemy unit is in range.
Initial Step-By-Step Description	1. The AI controlled unit encounters with an enemy unit. 2. The AI controlled unit fires to the enemy unit if the enemy is in attack range.
Xref	Section 3.1.4.4,Attack enemy

3 Requirement Specifications

There are basically two types of requirements namely functional and non-functional. Each requirement has a priority value.

- **Essential:** Implies that the software will not be acceptable unless these requirements are provided in an agreed manner.
- **Conditional:** Implies that these are requirements that would enhance the software product, but would not make it unacceptable if they are absent. Only if enough time and sources are available, these requirements are fulfilled.

3.1 Functional Requirements

3.1.1 Pre-Game Requirements

3.1.1.1 Environment Determination

Description of the requirement	The system shall provide the User with the functionality that the User can make selections for the space environment
Priority	Conditional
Inputs	Mouse click
Source	Mouse input
Outputs	Desired environmental conditions are selected.
Pre-condition	The user already opened the environment screen.
Action	The user made decision for space environment just by clicking the desired selection.
Xref	UC02

3.1.1.2 Agent Number Specification

Description of the requirement	The system shall provide the User with the functionality that the User can specify the number of agents.
Priority	Conditional
Inputs	Keyboard press
Source	Keyboard input
Outputs	Number of agents is specified.
Pre-condition	The user already opened agent number selection screen.
Action	The user presses one of the number buttons on the keyboard and set the agent number to the given number.
Xref	UC03

3.1.1.3 Difficulty level selection

Description of the requirement	The system shall provide the User with the functionality that the User can select the difficulty level.
Priority	Conditional
Inputs	Mouse click
Source	Keyboard input
Outputs	Look for alternative difficulty levels and select one of them.
Pre-condition	The User clicked
Action	There are three difficulty levels which are low, medium and high. User can select one of them just by clicking.
Xref	UC04

3.1.1.4 Button Configuration

Description of the requirement	The system shall provide the User with the functionality that the User can assign any functionality of the game to any button. The formations can only be assigned to 1, 2 or 3 button on the keyboard.
Priority	Conditional
Inputs	Keyboard press
Source	Keyboard input
Outputs	According to the user choices, buttons for movement, buttons for guns, buttons for formation and other game functionalities assigned to some buttons.
Pre-condition	The User press one of the keyboard buttons.
Action	Flight formation of aircrafts can be only attained to 1, 2 or 3 buttons. Movement, camera, gun and other functionalities can be assigned to remaining buttons. Firstly user selects the functionality, and then presses the button he/she prefers.
Xref	UC05

3.1.1.5 Spacecraft Selection

Description of the requirement	The system shall provide the User with the functionality that the User can choose the aircrafts wished to be used.
Priority	Conditional
Inputs	Mouse click
Source	Mouse input
Outputs	According to the user choices, aircrafts in the simulation are selected.
Pre-condition	The User press one of the keyboard buttons.
Action	In the spacecraft selection screen, there are several spacecraft types and models, user can select one of them to be able to use in simulation or select them just for agents. The selection procedure made with mouse click.
Xref	UC06

3.1.2 Game Control Requirements

3.1.2.1 Control of Movements

Description of the requirement	The system shall provide the User with the functionality that the User can direct the selected character dynamically.
Priority	Essential
Inputs	Keyboard press
Source	Keyboard input
Outputs	Changing the directions and positions of the selected character
Pre-condition	The User press “w”, “a”, ”s” and ”d” buttons of keyboard.
Action	When the User pressed on one of these buttons, the KeyboardListener receives the keyboard event. • If “w” button is pressed, character changes its direction to up. • If “s” button is pressed, character changes its direction to down. • If “d” button is pressed, character changes its direction to right. • If “a” button is pressed, character changes its direction to left.
Xref	UC08

3.1.2.2 Rotation about direction axes

Description of the requirement	The system shall provide the User with the functionality that the User can direct the selected character rotation dynamically.
Priority	Essential
Inputs	Keyboard press
Source	Keyboard input
Outputs	Changing the directions and positions of the selected character
Pre-condition	The User press “q” and ”e buttons of keyboard.
Action	When the User pressed on one of these buttons, the KeyboardListener receives the keyboard event. • If “q” button is pressed, character changes its rotation to left. • If “e” button is pressed, character changes its rotation to right.
Xref	UC09

3.1.2.3 Fire to the Target

Description of the requirement	The system shall provide the User with the functionality that the User can fire to the targets.
Priority	Essential
Inputs	Mouse click
Source	Mouse input
Outputs	The selected character fires to the targets.
Pre-condition	The User clicks the mouse.
Action	When the User clicked mouse, the MouseListener receives the mouse event. The selected character fires to the target which had been taken aim before. The selection of weapons which will be used for firing depends on which mouse button is clicked. • If left mouse button is clicked, primary weapon is fired. • If right mouse button is clicked, secondary weapon is fired.
Xref	UC10

3.1.2.4 Zoom In/Out the View

Description of the requirement	The system shall provide the User with the functionality that the User can zoom in/out the view.
Priority	Essential
Inputs	Mouse wheel roll
Source	Mouse input
Outputs	The view of game is zoomed in/out.
Pre-condition	The User rolls the mouse wheel.
Action	When the User rolled mouse wheel, the MouseListener receives the mouse event. Depending on the rolls direction, the view of game is zoomed in or out. • If wheel is rolled in forward direction, the view is zoomed in. • If wheel is rolled in backward direction, the view is zoomed out.
Xref	UC11

3.1.2.5 Pause

Description of the requirement	The system shall provide the User with the functionality that the User can pause the game at any time.
Priority	Essential
Inputs	Keyboard press
Source	Keyboard input
Outputs	The game stops and selection menu is shown.
Pre-condition	The User press “ESC” button of keyboard.
Action	When the User pressed this button, the KeyboardListener receives the keyboard event. The game is stop until the “ESC” button is pressed again.
Xref	UC12

3.1.2.6 Accelerate

Description of the requirement	The system shall provide the User with the functionality that the User can accelerate the selected character.
Priority	Essential
Inputs	Keyboard press
Source	Keyboard input
Outputs	The speed of selected character increases.
Pre-condition	The User press “space” button of keyboard.
Action	When the User pressed this button, the KeyboardListener receives the keyboard event. The character has the default speed. When “space” button is not pressed; • If the current speed of character is equal to the default speed, the speed does not change. • If the current speed of character is greater than the default speed, the speed decreases up to the default speed. When the User keeps pressing “space” button; • If the current speed of character is equal to the max speed, the speed does not change. • If the current speed of character is smaller than the max speed, the speed increases up to the max speed. PS: default and max speed is specified later.
Xref	UC15

3.1.3 In Game Notification Requirement

3.1.3.1 Player Health Bar

Description of the requirement	The system shall provide the user with the functionality that the User can see remaining health value of the planes his and his opponents according to damage they takes.
Priority	Essential
Inputs	Damage taken by aircraft
Source	Current state information coming from server
Outputs	Displaying current health values of aircraft
Pre-condition	Starting game
Action	When the User starts game all aircrafts have full health bar. After combat starts when an aircraft is hit or crashes, its health value is decreased according to amount of damage it takes.
Xref	None

3.1.3.2 Game logs

Description of the requirement	The system shall provide the user with the functionality that the User can see the destroyed aircraft and the name of the destroyer of that aircraft and the location where it happens.
Priority	Essential
Inputs	Destruction of an aircraft
Source	Information coming from server
Outputs	Displaying name of the destroyed aircraft and its destroyer
Pre-condition	Destroying an aircraft in combat
Action	When an aircraft is destroyed in combat the names of the destroyed player and destroyer player and the coordinates of battle will be displayed in game screen.
Xref	None

3.1.3.3 Speed

Description of the requirement	The system shall provide the User with the functionality that he/she can see current speed of his/her plane.
Priority	Essential
Inputs	Player movements
Source	Current state information coming from server
Outputs	Displaying current speed value of plane
Pre-condition	Starting game
Action	When the User starts game all planes have 0 speeds. After takeoff speed increases. When user slows the plane speed decreases. These changes calculated according to environment conditions, the pressing duration of movement buttons and the damage that plane takes until that moment.
Xref	None

3.1.3.4 Map Information

Description of the requirement	The system shall provide the User with the functionality that he/she can see current locations of all spacecrafts in the game.
Priority	Essential
Inputs	Coordinates of all spacecrafts and space
Source	Coordinates information coming from server
Outputs	Displaying all spacecrafts and battles on map
Pre-condition	Starting game
Action	When the User starts game the in left corner of GUI map will be opened automatically. In this map teammates, opponents and battles will be shown according to these specifications: 1. Team mates will be represented by green arrow. 2. Opponents will be represented by red cross. 3. When a battle begins exclamation point of battle location. 4. Skull for a destroyed plane coordinates.
Xref	None

3.1.4 AI Requirements

AI bots will basically have two properties. They have to follow the human player in their team. They have to response to the movements of the player and avoid any collision with other units during any maneuver. When the player interacts with enemies, with their team leader, the ai agents will attack or retreat.

3.1.4.1 Follow the Player

Description of the requirement	The AI controlled agents, in the game, shall follow the player otherwise the player gives different orders.
Priority	Essential
Inputs	Player instructions, tactics
Source	Current state information coming from server
Outputs	Bots make their move according to the real player in their team.
Pre-condition	The User does not provide any order that makes the bots move in different manner.
Action	When the user changes its direction, velocity or altitude, apply these changes to bots' flight parameters. During movement, current formation, distance between planes shall be kept same.
Xref	AI01

3.1.4.2 Attack Enemy

Description of the requirement	When the enemy is in attack range, the AI units shall fire to the enemy.
Priority	Essential
Inputs	None
Source	Current state information coming from server
Outputs	Bots attack to the enemy.
Pre-condition	An enemy unit must be in attack range.
Action	During simulation, when an AI-agent encounters with an enemy unit, it should attack to the enemy. If the enemy gets out of range, according to the strategy (i.e. defensive, aggressive), it will keep chasing to get in range again or keep it's moving.
Xref	AI03

3.2 Non-Functional Requirements

3.2.1 Usability

This project is a warfare game; users will be inexperienced when they start to play. Users will be interact with simulation after it is started and with the help of user friendly GUI and changeable control settings they will adapt the environment easily. To help users to adapt game environment all graphic models shall be as clear as possible.

3.2.2 Extensibility

The Eleron shall be open to changes that necessary additions will be able to add and unnecessary parts will be removed. These changes will be applied to make game play more playable and less complicated.

3.2.3 Reliability

The Eleron shall be played without any problems or bugs. All components will work synchronously that will not cause any performance issues. The reactions of bots shall always be consistent.

3.2.4 Performance

All implementations will be efficient. It's RTI, game engine and audio engine implementation will have high performance that will not cause any performance concerns. Two users shall play without any performance concerns.

3.2.5 Documentation

For project users, manual will be provided. In this manual the information about installation, configuration and running game stages will be given to the users. Moreover, for beginners at game start some tooltips will be available to show basic movements.

3.2.6 Reusable Connection Component

The communication component between different technologies shall be generic. It should be easily reusable in other projects. To enable this, an API may be implemented. By providing basic send and receive methods the data transfer shall be handled.

4 Conclusion

In this report, software team's perspective to online team collaboration platform game. Game scenario is specified with some details. Interaction between the user and system is defined. Scheduling timeline and task sharing between software team is arranged. After some research, technologies to be used understood by the team and team members realized their capabilities too with the help of this document.