Software Requirements Specification
for
Live Action Role-play facilitating system

Prepared by:

Chris A. Pieterse (u12004333)

University of Pretoria
Software Engineering (COS730)

Ms. Stacey A. Baror

Pretoria, 12. April 2021

Contents

Chapter 1

Introduction

Describe the purpose and scope of the software system- Explain the vision and objectives. State the business need for the application and summarise the scope of the project.

1.1 Purpose

Enhance the immersion of live action role play, and the ease of engagement for new and established participants

1.2 Scope

The facilitation of data transference between implements, users and a centralised information system.

1.3 Vision

Immersive live action role-playing, where participants can free their focus from the “meta-game” and be comfortable that their interaction with other participants are handled consistently and fairly.

1.4 Objectives

- Handle implement data capturing.
- Facilitate inter-implement communication.
- Facilitate implement data transference.
- Synchronizing data with central information system
- Synchronizing data between participants
- Provide information for consumption on centralised information

Chapter 2

User characteristics

2.1 Participant

Purpose
To participate in live action role play and engage with other participants.

Educational level
Rules of engagement

Experience
Any

Expertise
Melee Combat
Ranged Combat

Technical skills
Technologically literate

2.2 Marshal

Purpose
To moderate the state of play and enforce rules of engagement.

Educational level
Formal Rules of engagement

Experience
Journeyman

Expertise
Refereeing
Rules of engagement

Technical skills
Technologically literate

2.3 Game Master

Purpose
To moderate and observe the progression of the story and enrich the context of play.

Educational level
Formal Rules of engagement

Experience
Master

Expertise
Melee Combat
Ranged Combat

Technical skills
Technologically literate

2.4 Spectator

Purpose
To observe the state of play, and support their participant, possibly also use the data gathered to formulate some predictions, or gamble on outcomes.

Educational level
Any

Experience
Any

Expertise
Any

Technical skills
Technologically literate

Chapter 3

Functional requirements

3.1 Use case diagram

3.1.1 Use Case 1: Participant Interaction

U1 : Facilitate interaction between Participant

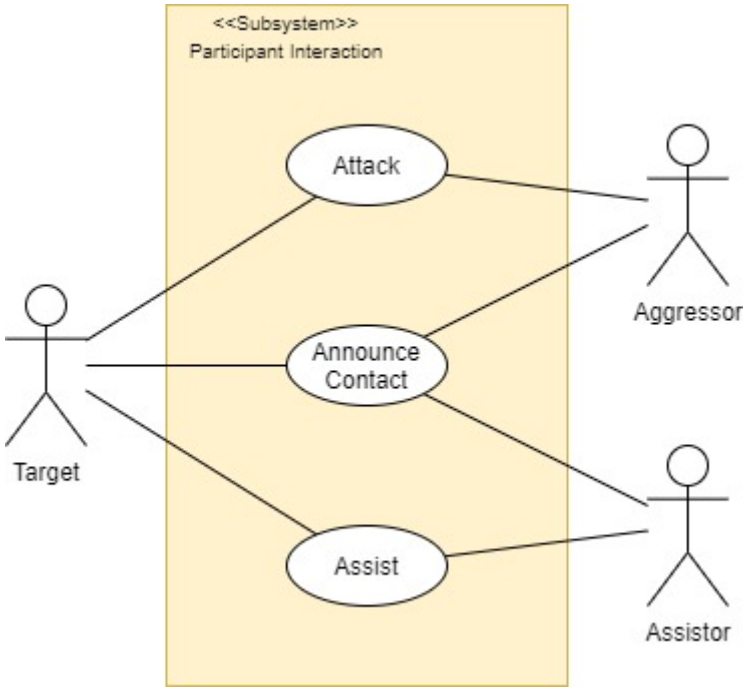

Figure 3.1: Participant Interaction Subsystem

- **R1 :** Attack
  As an Aggressor, a Participant should be able to attack a Target.
  - **R1.1 :** Publish damage
    An Aggressor should publish it's damage value to a Target.
  - **R1.2 :** Handle damage
    A Target should handle incoming damage values from Aggressors, and use defence values to calculate total taken, and health lost.

- **R2 :** Assist
  As an Assistor, a Participant should be able to assist a Target.
  - **R2.1 :** Publish aid
    An Assistor should publish it's aid value to a Target.
  - **R2.2 :** Handle aid
    A Target should handle incoming aid values from Assistors, and calculate health gained.
- **R3 :** Announce Contact
  As an Target, a Participant should be able to announce contact an Aggressor or Assitor.
  - **R3.1 :** Publish hit acknowledgement
    A Target should publish an acknowledgment of contact to Aggressors.
  - **R3.2 :** Handle hit acknowledgement
    An Aggressor should handle an acknowledgment of contact, and calculate score gained.
  - **R3.3 :** Publish aid acknowledgement
    A Target should publish an acknowledgment of aid to Assistors.
  - **R3.4 :** Handle aid acknowledgement
    An Assistors should handle an acknowledgment of aid, and calculate score gained.

3.1.2 Use Case 2: State of Play Management

U2 : Managing the State of Play

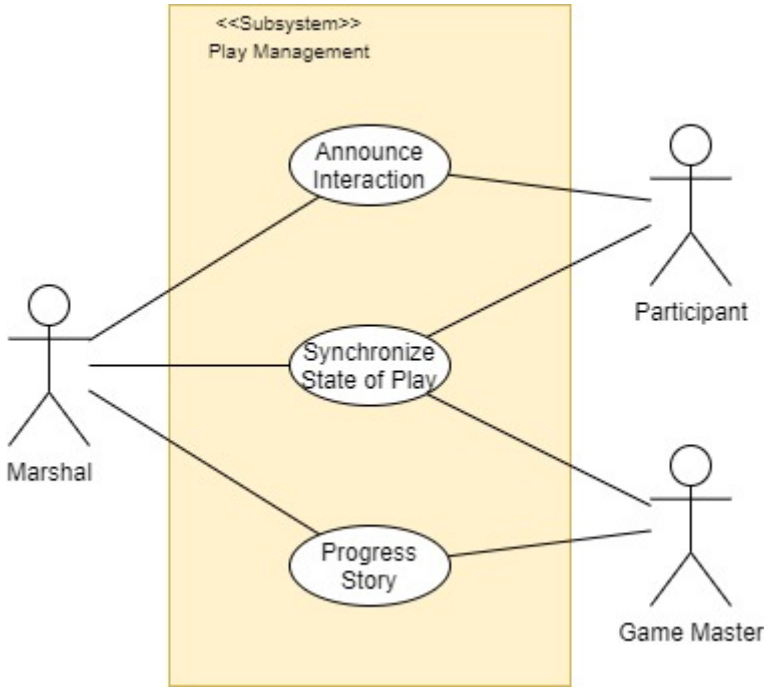

Figure 3.2: Play Management Subsystem

- **R4 :** Announce Interaction
  A Participant should be able to Announce Interaction(s) with other Participants.
  - **R4.1 :** Publish interaction
    A Participant should publish interactions with other Participants to the play log.

  - **R4.2 :** Handle interaction
    A Marshal should handle interaction announcements and consolidate the state of play for the Participants.

- **R5 :** Synchronize State of Play
  A Marshal should be able to Synchronize the State of Play between Participants and the Game Master.
  - **R5.1 :** Publish event state
    A Marshal should publish the state of play to all Participants and the Game Master.
  - **R5.2 :** Handle event state
    A Participant should handle state of play updates and adjust it's state and the context they find themselves in accordingly.
    A Game Master should handle state of play updates and adjust the world state and context accordingly.

- **R6 :** Progress Story
  A Game Master should be able to Progress the Story of the play through the Marshal.
  - **R6.1 :** Publish world event
    A Game Master should publish world events, meant to progress the story, to the play log.
  - **R6.2 :** Handle world event
    A Marshal should handle world event updates and consolidate the state of play for the Participants.

3.1.3 Use Case 3: Community Engagement

U3 : Facilitate Engagement with Community

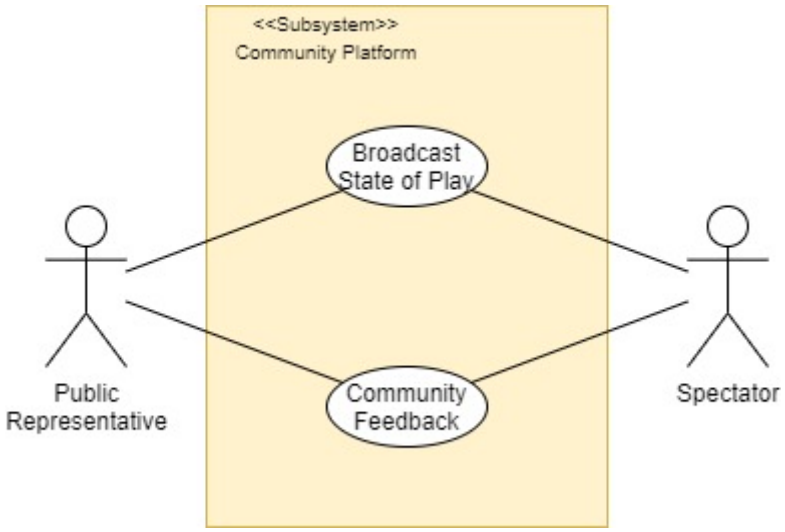

Figure 3.3: Community Platform Subsystem

- **R7 :** Broadcast State of play
  A Public Representative should be able to broadcast the World/Play State for Spectator consumption.
  - **R1.1 :** Publish world state
    A Public Representative should publish the world state periodically, or on request.
  - **R1.2 :** Handle world state
    A Spectator should handle world state updates and take it into consideration when providing feedback.

- **R8 :** Community feedback
  A Spectator should be able to provide Feedback for community consideration.
  - **R2.1 :** Publish feedback
    A Spectator should publish feedback on the world state.
  - **R2.2 :** Handle feedback
    A Public Representative should handle the feedback and take it into consideration when generating world events.

3.2 Traceability matrix

Requirements	Participant Interaction (U1)	Play Management (U2)	Community Platform (U3)
R1	R1.1, R1.2	-	-
R2	R2.1, R2.2	-	-
R3	R3.1, R3.2, R3.3, R3.4	-	-
R4	-	R4.1, R4.2	-
R5	-	R5.1, R5.2	-
R6	-	R6.1, R6.2	-
R7	-	-	R7.1, R7.2
R8	-	-	R8.1, R8.2

Chapter 4

Domain models

4.1 Domain model class diagrams

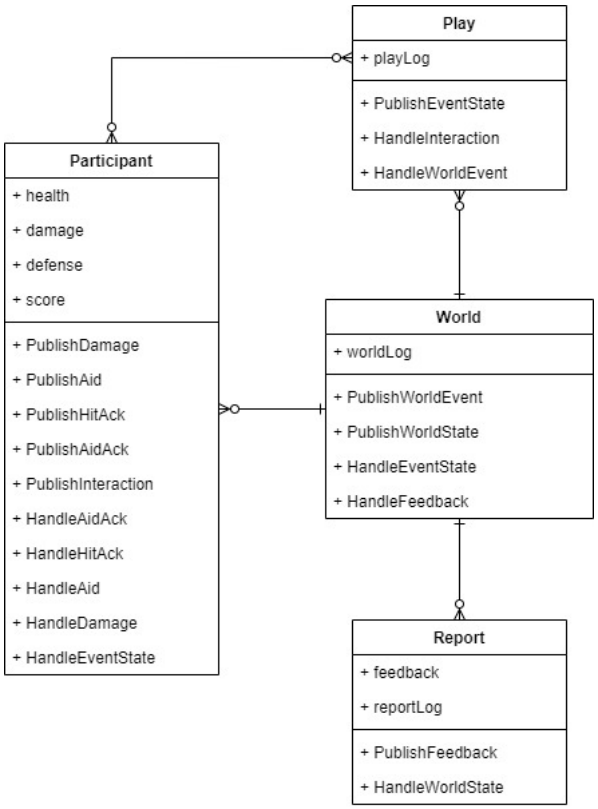

Figure 4.1: Complete Domain Model Class Diagram

Participant
The Participant, as it appears in Figure 4.1. is composed from the properties:

- **health** - Used to determine whether a participant is alive or deceased, and by extension whether they can interact with the play.
- **damage** - Used to determine the value of damage a participant publishes to targeted participant.
- **defence** - Used to determine how much health a participant loses from incoming damage.

- **score** - Used to determine whether a participant succeeding in the play.

Participants must belong to a world, but does not necessarily need to participate in play(s).

Play
The Play, as it appears in Figure 4.1. is composed from the properties:

- **play log** - Used to determine the state of play, and the interactions between participants, the play environment, and the greater world.

A Play must belong to a world, and contain at least one participant.

World
The World, as it appears in Figure 4.1. is composed from the properties:

- **world log** - Used to determine the state of the world, and the events that contribute to the world context.

A World is required by all other domain models.

Report
The Report, as it appears in Figure 4.1. is composed from the properties:

- **feedback** - Used to determine community opinion about the state of the world.
- **report log** - the state of the world, reduced to fulfill the interests of the report.

A Report must belong to a world.

4.2 System Sequence diagrams

Participant Interaction
Participant interaction, as depicted in Figure 4.2.

- **(a) Aggressor sequence** - The Aggressor participant interacts with a Target participant, and notifies the system about the interaction.
  - Publishes damage to Target
  - Handles contact acknowledgement from Target
  - Publishes interaction to system
  - Handles play state update from system
- **(b) Assistor sequence** - The Assistor participant interacts with a Target participant, and notifies the system about the interaction.
  - Publishes aid to Target
  - Handles aid acknowledgement from Target
  - Publishes interaction to system
  - Handles play state update from system
- **(c) Target sequence** - The Target participant interacts with a Aggressor/Assistor participant, and notifies the system about the interaction.

- Publishes contact acknowledgement to Aggressor
- Handles damage from Aggressor
- Publishes aid acknowledgement to Assistor
- Handles aid from Assistor
- Publishes interaction to system
- Handles play state update from system

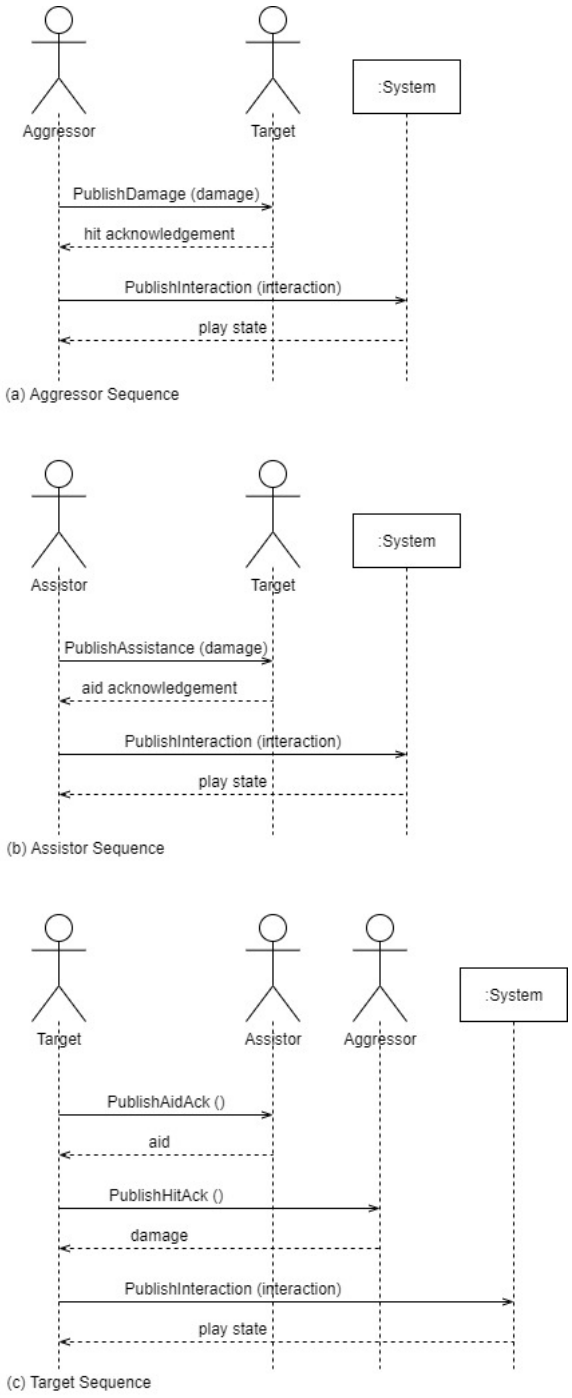

Figure 4.2: Participant Interaction System Sequence Diagram (a) Aggressor sequence (b) Assistor sequence (c) Target sequence

Play Management
Play management, as depicted in Figure 4.3.

- **(a) Marshal sequence** - The Marshal consolidates the participant interactions and world events into a synchronised, consistent play state.
  - Publishes state of play to system, for distribution to Participants and the Game master
  - Handles interaction announcements from the system, and consolidates them to the play log
  - Handles world events from the system, and consolidates them to the play log
- **(b) Game master sequence** - The Game master consolidates the play states into the world state, and steers the play state with world events.
  - Publishes a world event to the system, to steer state of play
  - Handles play state updates from the system, and consolidates them into the world log

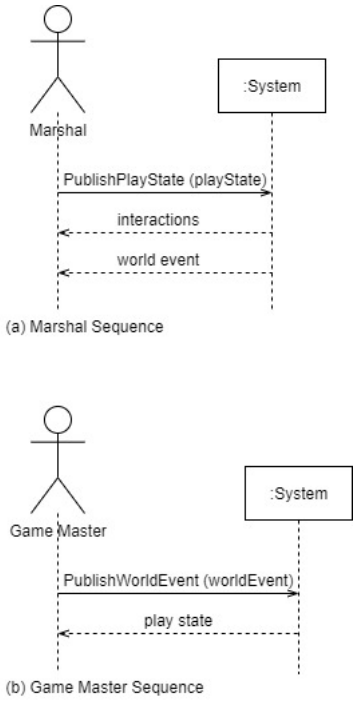

Figure 4.3: Play Management Sequence Diagram (a) Marshal sequence (b) Game master sequence

Community Engagement
Community engagement, as depicted in Figure 4.4.

- **Community engagement sequence** - The Spectator consumes world states and provides feedback on the community's opinion of the state of the world.
  - Publishes feedback to the system
  - Handles world state updates, and determines the opinion of the community for progress of world state

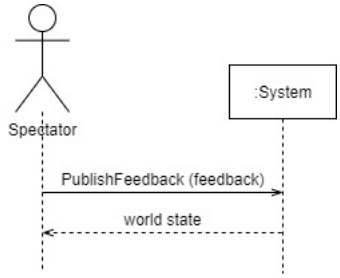

Figure 4.4: Community Engagement Sequence Diagram

Chapter 5

Non-functional requirements

5.1 Quality requirements

- **Performance**
  The system will be distributed over:
  - Embedded systems
  - Mobile systems
  - cloud hosted systems

  and should perform real-time computations, therefore it should be optimised on each respective system.

- **Reliability**
  The system will be operating in real time, and the course of play, and user experience will be greatly impacted by reliability issues.

- **Security**
  The system should make use of basic authentication, but no personal information about the users will be persisted, thus removing the risk of personal data leaks.

- **Maintainability**
  The system should comply to modern best practices, and sufficient domain abstraction, to improve comprehension by industry professionals.

- **Useability**
  The system should facilitate the play, and as such user interaction with the system should be implicit through user actions.

- **Flexibility**
  The system should be open for OEM's to extend the functionality of the system in new and intuitive ways that will enhance the user experience.

5.2 Specify and quantify

- **Performance**
  - Embedded systems: Handle simultaneous interactions.
  - Mobile systems: No more than a 10% footprint on mobile battery.
  - cloud hosted systems: Support up to 50 simultaneous users.
- **Reliability**
  In the case that any of the systems should fail, the rest of the systems should be unaffected.
- **Security**
  The system should make use of basic authentication. The system should not persist personal information about the users.
- **Maintainability**
  - S.O.L.I.D. Principal compliant software architecture.
  - TDD implementation approach.
- **Useability**
  The system should limit user goal fulfillment steps to a maximum of 3.