DorcSlayer

Functional Model Document

Team Christine
Christine Laurendeau

Submitted to:
Dr. Christine Laurendeau
COMP 3004 - Object-Oriented Software Engineering
School of Computer Science
Carleton University

September 5, 2019

NOTE: This document is partial only! It is not complete. It is meant solely as an example of the *format* that is expected for some portions of assignments and/or deliverables. It does NOT include all the details that are expected of work submitted for credit. For more information on what is expected, please refer to the assignment and/or deliverable description, as well as the corresponding discussion posts in *cuLearn*.

1. Functional Requirements

F-01	GamePlayer user must be able to load a new game
F-02	GamePlayer user must be able to load a saved game
F-03	GamePlayer user must be able to save the current game
F-04	GamePlayer user must be able to control the actions of the player character, a Tortoise by default
F-04-01	GamePlayer user must be able to view the player character's inventory
F-04-02	GamePlayer user must be able to move the player character to an adjacent position on the map
F-04-03	GamePlayer user must be able to have the player character pick up an item and place it in inventory
F-04-04	GamePlayer user must be able to have the player character eat a food ration from his inventory
F-04-05	GamePlayer user must be able to have the player character wield a weapon from his inventory
F-04-06	GamePlayer user must be able to have the player character hit another character
F-04-07	GamePlayer user must be able to have the player character rescue the Dragon
F-05	GameAdministrator user must be able to modify the game parameters
F-05-01	GameAdministrator user must be able to modify the dorc generation frequency
F-05-02	GameAdministrator user must be able to modify the dorc velocity
	... and maybe more ...

2. Non-Functional Requirements

Usability	
NF-01	GamePlayer user must be able to view the updated game map in real-time
NF-02	Shortcuts should be provided for the user to enter game commands, on platforms where this is possible
NF-03	Save operations should be confirmed by the user
NF-04	All error messages should be descriptive and suggest appropriate solutions
Reliability	
NF-05	A game in play should be backed up every 30 seconds or after every move, whichever comes first
NF-06	If the system fails during a game in play, the user should be prompted for a restore from the last backup upon the next startup
Performance	
NF-07	The game map should be refreshed within 2 seconds of the user entering a game command, 95% of the time
Supportability	
NF-08	The system should be extensible to any GUI platform with minimal modification
NF-09	The system should be extensible to a multi-player version
Implementation	
NF-10	The system must work on the SCS Linux network
NF-11	The system must be written in C++
Packaging	
NF-12	The system must be available as an executable for download
	... and many more ...

3. Use cases

3.1 Use case diagram for the full DorcSlayer system

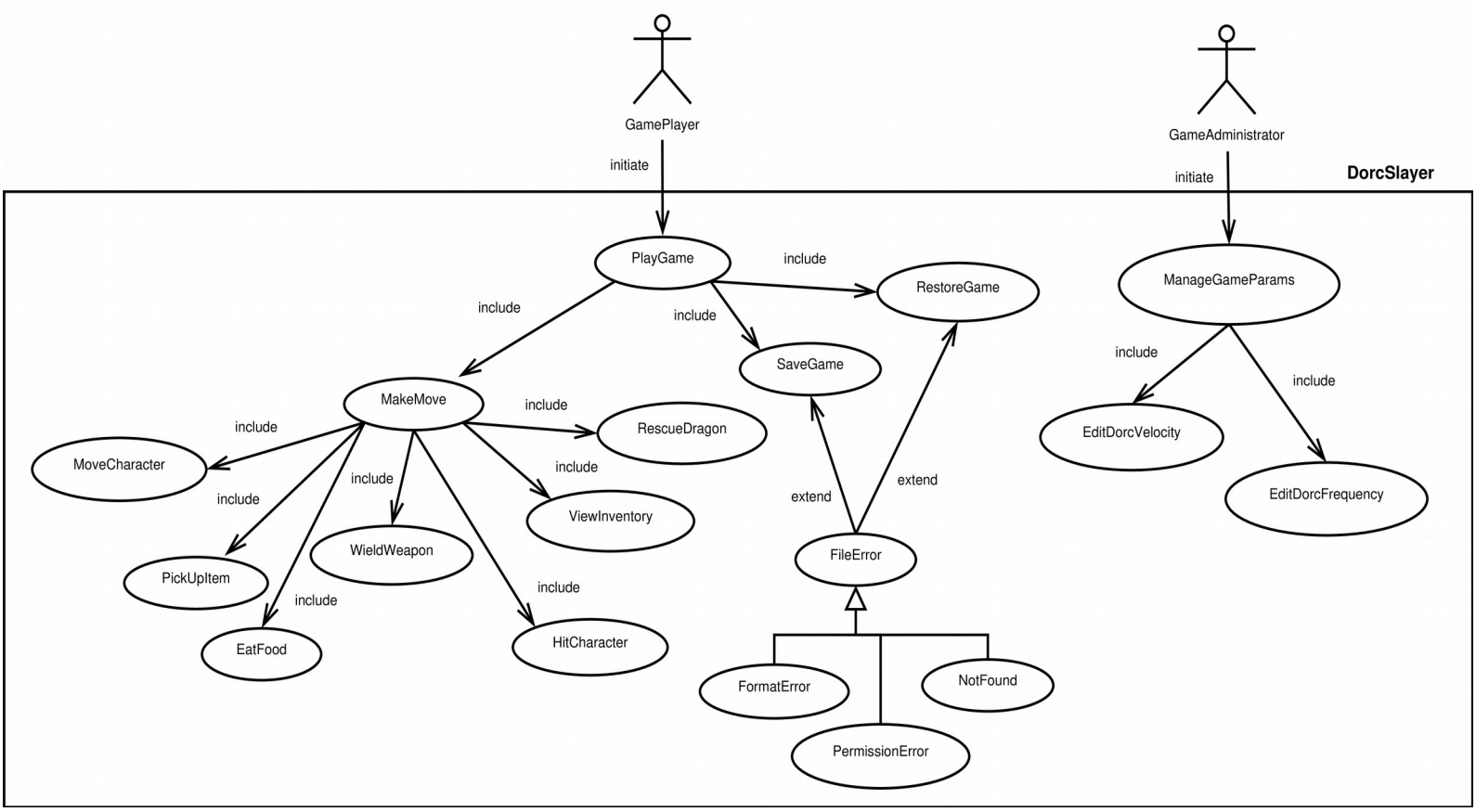

Figure 1 -- UML Use Case Diagram

3.2 Use case table descriptions

This section does **NOT** cover all the use cases. It is included here for example purposes only.

Use case Id	UC-01
Name	**PlayGame**
Participating actors	Initiated by GamePlayer
Flow of events	1. The GamePlayer makes a sequence of moves, as handled by the **included** use case **MakeMove**, or enters a game management command, as handled by the **included** use cases **SaveGame** and **RestoreGame**. 2. The system executes the specified command.
Entry conditions	
Exit conditions	The game map is updated after each move.
Quality requirements	
Traceability	F-01, F-02, F-03, F-04, NF-01

Use case Id	UC-03
Name	**SaveGame**
Participating actors	Initiated by GamePlayer
Flow of events	1. The GamePlayer selects a directory where the game will be saved. 2. The GamePlayer selects or enters a file name for the game to be saved. 3. The system stores the current game under the given name, in the specified directory.
Entry conditions	The GamePlayer must currently be playing a game.
Exit conditions	The game is stored in the specified file and directory.
Quality requirements	Games should only be stored in directories where the GamePlayer has read and write permissions.
Traceability	F-03

Use case Id	UC-06
Name	**MakeMove**
Participating actors	Initiated by GamePlayer
Flow of events	1. The GamePlayer specifies a game command, which is handled by the corresponding **included** use case (**MoveCharacter**, **PickUpItem**, **EatFood**, **WieldWeapon**, **HitCharacter**, **RescueDragon**, or **ViewInventory**). 2. The system performs the specified game command. 3. The system performs the same command on the Hare non-player character, as well as on the Dragon, after it is rescued. 4. The system determines if a new dorc should be generated. If so, a new dorc is positioned on the game map. 5. The system displays an updated game map, based on the position of the player character. If the player character is in a cave, then the cave map is displayed. If the player character is outside a cave, the surrounding area is displayed.
Entry conditions	The GamePlayer must currently be playing a game.
Exit conditions	The game map has been updated after the move.
Quality requirements	• A special "wait" icon will be displayed while the game command is being processed. • The game map will be updated and refreshed within 2 seconds. • Any problem encountered in processing the game command will pop up a message for the GamePlayer.
Traceability	F-04, NF-04, NF-07

Use case Id	UC-08
Name	**PickUpItem**
Participating actors	Initiated by GamePlayer
Flow of events	1. The GamePlayer specifies a command for the player character to pick up the item at his current position on the game map. 2. The system removes the item from the current position. 3. The system adds the item to the player character's inventory
Entry conditions	• The GamePlayer must currently be playing a game. • The item must be in the position occupied by the player character.
Exit conditions	The item is included in the player character's inventory.
Quality requirements	
Traceability	F-04-03

Use case Id	UC-14
Name	**FileError**
Participating actors	GamePlayer
Flow of events	1. The system notifies the GamePlayer that an error has occurred with accessing a selected game file.
Entry conditions	• A file processing operation has failed. • **Extends** use cases **SaveGame** and **RestoreGame**.
Exit conditions	The requested operation (save or restore) is aborted. Any processing that was completed is rolled back.
Quality requirements	
Traceability	NF-04