COMP-361 Project
Requirements Models

(15% of final grade)

The goal of this milestone is to provide a complete requirements specification for the medieval warfare game.

1 Use Cases (Use Case Model)

We want you to provide a use case model that describes the interactions between the system and the environment that take place when interacting with your medieval warfare game. The model must cover all functionality described in the informal requirements document handout, including connecting the players, setting up the game, taking turns, sending messages among players and ending the game. To summarize your model, create a use case diagram that shows all involved actors (with multiplicities), associated to the use cases in which they participate. Don't forget to depict dependencies between use cases, if any.

Note: you don't need to create individual use cases for every kind of game move, since probably the interactions with the system for different kinds of game moves are similar. However, we want to make sure that you are aware of all possible kinds of moves, and therefore we want to minimally see at some point an enumeration of the kind of moves that a user can communicate to the system.

How is this section going to be graded? We will check that:

- The use case diagram uses correct UML
- The fields of each use case template have been filled out correctly.
- The system boundary is clearly described, i.e., each basic interaction step describes a single interaction between an actor and the *system.*
- That your main success scenario (and in the extensions) contain all mandatory interaction steps between the environment and the system to achieve the use case goal.

2 Architectural Decisions

We would like you to describe your distributed architeture. For sure you need some kind of "server" that allows players to find opponents and that keeps track of player statistics. This server can be very general, i.e., a "game lobby service" it allows matchmaking and statistics for any kind of game. You might be even able to use some already provided service for that, e.g., the Apple Gamecenter. On the other hand, the server can also be very specific, i.e., it knows about the medieval warfare game and rules, and is involved in every game move that is played.

In this section we want to know how many different executables you are planning to build. For example, in a typical client/server architecture you would develop two different executables. In a game that runs in a peer-to-peer style you would use a generic game lobby and then design only one executable. At this point it might also make sense to mention where (i.e. on the client or the server, or on one peer or both) some of the game functionality is going to reside. Interesting functionality to consider includes move verification (i.e. compliance with the game rules), and saving/loading functionality.

3 Requirements Models

For each application/executable that you described in the architecture section above, you will need to submit an Environment Model, a Concept Model, an Operation Model and a Protocol Model.

3.1 Structural Requirements (Environment Model and Concept Model)

The Environment Models must contain all input messages necessary to trigger the game functionality described in the game handout and elaborated in the use cases. The Environment Model must also contain all output messages that the system can produce according to the game handout and the use cases. In addition, the Environment Model must contain the messages that are exchanged between the client and the server, if any. The actors in the environment model should represent the devices/other systems that your system is going to interact with. In other words, the "player" should not show up as an actor in the environment model anymore, since your software can not directly interact with a human, but replaced with a GUI actor.

The Concept Models needs to contain all conceptual game state that needs to be stored in order to play the game. If you have a client/server architecture, some concepts will be part of both systems, some might only be part of one.

How is this section going to be graded? We will check that:

- Both the Environment Model and the Concept Model use correct UML syntax.
- For the Environment Model: every system functionality can be triggered by some input message.
- For the Environment Model: every interaction between the environment and the system described by the use cases is realized by some input or output message.
- For the Environment Model: the multiplicities of actors and interactions are specified correctly.
- For the Concept Model: it must contain the conceptual state of the game, as well as the players. It should also contains all conceptual state needed to provide the desired functionality described by the game handout and the use cases when an input message is received.
- For the Concept Model: all association multiplicities are specified correctly, and each association end has a role name.

Tool: You can use any modelling tool or drawing tool to create the environment and concept models. Note that you can use the TouchRAM tool to create the Concept Model, if you are not afraid of using a tool that might crash from time to time :) http://www.cs.mcgill.ca/~joerg/SEL/TouchRAM.html. TouchRAM has some limitations when you try to use it to define a concept model:

- TouchRAM cannot show the system class that contains all other classes. So it is ok to omit it.
- TouchRAM can not visualize actors as "stickmen", please use as a naming convention the postfix "Actor", i.e., ScreenActor, or ClientActor.
- TouchRAM does not support association classes. If you want to define an association class, use as a naming convention the postfix "AC".

If you choose to use TouchRAM, please upload the .core and .ram file to myCourses.

3.2 Behavioural Requirements (Operation Model and Protocol Model)

In the Operation Models, please state for each identified system operation in the environment model what concepts of the concept model are affected by the system operation and what output messages are (potentially) sent. Describe the state change as a post condition formulated in English text. The Protocol Models take the form of use case maps and should describe in what sequence the systems expect to process the input messages and produce outputs.

How is this section going to be graded? We will check that:

- For each operation: the described state changes and output messages sent do indeed correspond to the desired functionality.
- The set of provided operations, when executed in the order specified by the use cases, achieve the use case goal.
- The ordering that the protocol model specifies for system operations corresponds to the ordering described in the use cases.

Tool: If you use jUCMNav to create your protocol models, please upload the corresponding .jucm file to myCourses.

Hand-in Date

Please submit your models (one printed copy per group, upload pdf and models on myCourses) by November 21st 2014.