DragonQuest - Software Requirement Specification

Jonathan Sharyari Sven Lundgren Kristian Johansson Björn Forsberg

2nd December 2013

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be: Correct Traceable (both forward and backward to prior/-future artifacts) Unambiguous Verifiable (i.e., testable) Prioritized (with respect to importance and/or stability) Complete Consistent Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

1 External Interface Requirements

1.1 User Interfaces

1.1.1 Chat

Description	Interface requirement: There should be a chat window with two fields, one for input and one to display recent messages. Both fields should have support for all languages based on the latin and cyrillic alphabet, that are needed to communicate in those languages.
Inputs	Inputs can be keyboard, mouse and touchpad.
Processing	Pressing the text field with the mouse or the touchpad should change the focus to the text field, in order to allow text input from the keyboard. From this point on, the user can type text, which will be submitted when the user presses the enter key.
Output	The chat display field displays the 5 latest lines of text.
Error handling	If input character doesn't exist that fact should not affect the rest of the provided characters. If data is lost between clients it should be re-sent.

1.2 Hardware Interfaces

1.3 Software Interfaces

1.4 Communications Interfaces

2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

2.1 Game settings

2.1.1 Multiplayer/Single Player Mode

Description The user should be able to choose to play the game in multiplayer format, or single player format.

Inputs The user should either provide an IP-address, as a dot-separated number or a hostname to connect to or alternatively, the user should be able to create a new game.

Processing The client will try to establish a connection to an external server on the host with the supplied IP-address. If this does succeed, the user will enter the existing game.
If the supplied IP-address is localhost, and no server is running on the same computer as the client, a new game will be started. When a new game is started, the user should be able to decide game parameters, most importantly the number of allowed external players and the number of players in total.
Note that the number of total players may be higher than the number of external players, i.e., the remaining players are controlled by the server AI. Creating a single player game then corresponds to the act of not allowing any external players.

Side-effect A game starts.

Error handling If game connection does not succeed, an error message is shown to the user describing the reason for the failure. The most common reasons of failure should be recognized, this includes hostname not being found, game not being found, game full and game has already started.

2.2 Game board actions

2.2.1 Move

Description	The user can move its hero to another room or into a tower.
Inputs	The user should provide a move command to the server, depending on how the client is designed how this is done might vary.
Processing	Except for validation (validation is also done on the server side) the client doesn't do any processing of the input. The input is passed to the server where its dealt with, a move will in normal cases be to any adjacent room (or up or down a staircase) that is allowed. In certain rooms the user has the option to move two or potentially more steps as well but these rules are set by the specific room.
Side-effects	The players turn is over.
Output	The hero in question has been moved to the square the move aimed for.
Error Handling	Any non-valid user inputs are dealt with by the client and a message that the move is invalid is displayed for the user. Invalid data can still reach the server if the data is altered on the way. Therefore the server also validates the data to make sure the move is allowed according to the game state and rules. If the client loses its connection to the server the user will get made aware that the command wasn't executed and that the action should be tried again.

2.2.2 Search room

Description	The user can search a room for items such as chests, hidden doors, treasures etc.
Inputs	The user gives the search room command via the client.
Processing	The room search can start at any point given that it's the player's turn and it stands in a searchable room (not all rooms are). When the search commences the player draws a room search card which will reveal what the player found and further action will take place accordingly.
Side-effect	The card drawn will determine the outcome of the search. See requirements for the specific actions. The players turn is over.
Error Handling	Any non-valid user inputs are dealt with by the client and a message that the move is invalid is displayed for the user. Invalid data can still reach the server if the data is altered on the way. Therefore the server also validates the data to make sure the move is allowed according to the game state and rules. If the client loses its connection to the server the user will get made aware that the command wasn't executed and that the action should be tried again.

2.2.3 Occupying the dragon's lair

Description	A player currently residing in the dragon's lair can decide to stay there for another turn in risk of awakening the dragon and in return taking damage from it.
Inputs	The user chooses to wait when occupying the dragon's lair.
Processing	When the user issues the "stay" command a dragon card is drawn and this will either show that the dragon stay asleep in which case no further action is taken. But the result can also be that the dragon is awaken in which case the hero will receive a random amount of damage and also kicked out of the lair to an adjacent tile.
Output	Either nothing (dragon stays asleep) or damage suffered and movement of the player's hero.
Error handling	If the player loses connection and cant reconnect in time an AI will take over and play the turn for him/her.

2.3 Battling

2.3.1 Battle

Description	A player can battle an opponent, controlled either by another player or the AI.
Inputs	The player and opponent both choose a card from their respective battle hands. The player uses a mouse or touchpad to do so.
Processing	The winner of the battle is determined by comparing the two cards. The participant with the highest total attack value wins the round, and the opponent loses one as many hit points as the number of cards the winner of the round played. There is a possibility of deathblow and counter attack, according to their respective requirements.
Output	The player and opponent are visually informed of the outcome of the battle round.

2.3.2 Ambush

Description	A player can be the victim of an ambush by an opponent when entering a new room.
Inputs	The player may choose whether to fight the opponent or flee from the battle. The choice is made using a mouse or touchpad.
Processing	If the player chooses to fight, a battle between the player and opponent commences. If the player chooses to flee, the player and opponent compares their respective power cards. If the player's power card has a higher attack value, the player successfully flees. Otherwise, the player takes damage ??? and a battle between the player and opponent commences.
Output	If the player chooses to flee, the player is informed visually of the outcome.
Error handling	See Requirement 2.4.1.

2.3.3 Death blow

Description	A player will do a death blow to an opponent thus doing extra damage to the opponent during that combat turn if the played attack card is matching any card in the combat stack and the value of the card is higher than that of the opponent.
Inputs	The attack card played by the player in question
Processing	The type of the attack card played is compared to the type of the cards in the combat stack. If the the type matches and the value of the player's card is higher the player may take all cards of that type from the combat stack and put it into the opponents damage stack. The battle turn is than resolved as usual.
Output	The output is the changed amount of damage the opponent takes as a result of the successful death blow
Error handling	If a player disconnects and does not re-establish connection within the specified round time, AI will take over until the player returns.

2.3.4 Counter Attack

Description	A player may counter attack after the initial battle cards have been played.
Inputs	Provided that the player has a counter attack card that matches the type of the opponents chosen attack a counter attack can be made by providing one or more CA cards.
Processing	The CA cards that are added to combat stack will increase the total attack value of the player in question. After these cards have been played the battle round continues as usual.
Output	The attack value of the counter attacking player will be increased accordingly.
Error handling	If a player disconnects and does not re-establish connection within the specified round time, AI will take over until the player returns.

2.4 Connections

2.4.1 Connection loss/stall

Description	The client should be able to reconnect to the game server without losing the game state.
Inputs	N/A
Processing	If the client reconnects within the servers specified turn time the game continues. If the client is unable to reconnect one of the following happens: • If player not in battle: The players turn is forfeit and the player loses 1 HP. • If player in battle: AI takes over until the player returns.

3 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95

3.1 Performance

3.2 Game experience

3.2.1 Competition

The game contains elements that allow for competativeness between players.
`Priority: 3`

3.2.2 Game community

The should exist a community within the game to keep the players as engaged as possible. This includes the abilities to keep track on other players (friends), the ability to change the game environment (on the client side) to the largest extent possible, and the ability to create and share user generated content within the community.

Priority: 4

3.2.3 Medievalness

The game should have provide a medieval experience, including music, graphics and language.
Priority: 3

3.2.4 Board game correlation

The game should deviate as little as possible from the standard board game, unless otherwise specified in this document.
Priority: 2

3.3 Reliability

3.3.1 Responsiveness

The game should stall as little as possible in order to improve the game experience. The game should also minimize stalls caused by users.
Priority: 1

3.3.2 Stability

The game shall not respond unexpectedly to erronous user input.
Priority: 1

3.4 Availability

3.4.1 Learning curve

It should be possible to learn the game adequately within 20 minutes of game play. No tutorial should be needed.
Priority: 3

3.4.2 Gameplay development

The game should include elements of gameplay that allow the user to develop as a player. Also, the characters in game should evolve in some extent based on in-game experiences, in order to keep the game as versatile as possible.
Priority: 3

3.5 Security

3.5.1 Validation

The server should validate the client input to make sure they are legal according to the game rules. The client needs to validate server data user input.
Priority: 1

3.5.2 Privacy

Any collected informations should be handled in accordance with Personuppgiftslagen (PUL), the swedish personal data law.
Priority: 1

3.6 Maintainability

3.6.1 Maintenance

The code should be easy to maintain and extend.
`Priority:` 1

3.6.2 Reusability

The code base of the project should be as little intertwined as possible with the specific game rules of DungeonQuest. This should allow for code reuse, when developing other board games in the future.
`Priority:` 1

3.7 Portability

3.7.1 Mobile devices

The game should support the most common mobile devices, such as the Windows mobile, iOS and Android platforms. It shall be possible to create a lower resolution client, in order to support any phone that fulfills the minimum requirements stated in 5.2.1.
`Priority:` 2

3.7.2 Multi-platform

The game should be written in Java, in order to function on any platform supporting the java virtual machine. As iOS currently does not support the machine, a objective-C version of the *client* should be made available. An iOS compatible version of the server will not be prioritized before the initial game release.
`Priority:` 2

4 Inverse Requirements

State any *useful* inverse requirements.

5 Design Constraints

Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

5.1 Process

5.1.1 Iterative development process

The game development should follow an iterative design process. Each iteration shall be two weeks long, and a snapshot of the development shall be handed in for review to the customer at the end of each iteration.

5.1.2 Deadlines

The game must be ready for deployement by christmas 2013, to exploit the christmas rush.

5.2 Product

5.2.1 Low hardware requirements

The hardware requirements of the game should be kept as low as possible. The game should run on any computer with at least 20mb RAM and 300mhz processor and a graphics card with 3d accelleration. (We're just making this up. It would be possible to have this kind of constraint but we don't know what numbers are reasonable).

5.2.2 Data transfer

The game should rely on low amounts of data transfer, minimizing both the cost for the users to play the game, and the demands on connectivity. This is essential for mobile device support.

5.2.3 Multi-client

The game should have a server-client architecture. It should be possible for users to design 3rd party clients, but not 3rd party servers for the game. The server should ensure that all clients follow the game rules, also see 3.5.1.

6 Other Requirements

Catchall section for any additional requirements.