System and Software Design Description(SSDD):
Incorporating Architectural Views and Detailed Design Criteria
For

Swords and Sorcery

Keith Drew, ...

May 5, 2014

Swords and Sorcery
Table of Contents

1 Introduction

This is the System and Software Design Document for Swords and Sorcery. This is one of five documents that describe the computer adaptation of the Swords and Sorcery board game developed by the Software Engineering class at the University of Idaho in Spring 2014.

1.1 Document Purpose, Context, and Intended Audience

1.1.1 Document Purpose

The purpose of this document is to describe the system and software design of Swords and Sorcery.

1.1.2 Document Context

This document is written as part of a larger document that describes the Swords and Sorcery project developed by the CS383 students at University of Idaho, in Spring 2014. This document only describes the system and software design of the project, which is only a subset of the project.

1.1.3 Intended Audience

This document is intended to be read by Dr. Clinton Jeffery and members of the class, as well as any interested members of the University of Idaho Computer Science department. This document is not intended to be distributed publicly in any way.

1.2 Software Purpose, Context, and Intended Audience

1.2.1 System and Software Purpose

The purpose of the Swords and Sorcery system and software is to provide a computer adaptation of the complex board game of the same name. The system is designed to provide multiplayer functionality over the internet, and to simplify the complex rules of the original Swords and Sorcery.

1.2.2 System and Software Context

The context of this project is, again, restrained to the classroom, as it is an educational project, not intended for distribution. However, the source code for the project, as well as many resources, are available publicly on github.com.

1.2.3 Intended Users of System and Software

The intended users of the Swords and Sorcery system are...

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
AD	Architectural Description: Ä collection of products to document an architecture.¨ISO/IEC 42010:2007
Alpha Test	Limited release(s) to selected, outside testers.
Architectural View	Ä representation of a whole system from the perspective of a related set of concerns.¨ISO/IEC 42010:2007
Architecture	T̈he fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.¨ISO/IEC 42010:2007
Beta Test	Limited release(s) to cooperating customers wanting early access to developing systems.
Design Entity	Än element (component) of a design that is structurally and functionally distinct from other elements and that is separately named and referenced.¨IEEE STD 1016-1998
Design View	Ä subset of design entity attribute information that is specifically suited to the needs of a software project activity.¨IEEE STD 1016-1998
GUI	Graphical User Interface - What the user sees and interacts with - also called the HUD.
HUD	Heads Up Display - What the user sees, with respect to interface - also called the GUI.
IP	Internet Protocol - Typically refers to an IP Address.
S&S	Swords and Sorcery
SSDD	System and Software Design Document
SSRS	System and Software Requirements Specification
System	Ä collection of components organized to accomplish a specific function or set of functions.¨ISO/IEC 42010:2007
System Stakeholder	Än individual, team, or organization (or classes thereof) with interests in, or concerns, relative to, a system.¨ISO/IEC 42010:2007

1.4 Document References

1. CSDS, textitSystem and Software Requirements Specification Template, Version 1.0, July 31, 2008, Center for Secure and Dependable Systems, University of Idaho, Moscow, ID, 83844.

2. ISO/IEC/IEEE, *IEEE Std 1471-2000 Systems and software engineering – Recommended practice for architectural description of software intensive systems,* First edition 2007-07-15, International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC), Case postale 56, CH-1211 Geneve 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

3. IEEE, *IEEE Std 1016-1998 Recommended Practice for Software Design Descriptions*, 1998-09-23, The Institute of Electrical and Electronics Engineers, Inc., (IEEE)

445 Hoes Lane, Piscataway, NJ 08854, USA.

4. 3) ISO/IEC/IEEE, *IEEE Std. 15288-2008 Systems and Software Engineering – System life cycle processes,*Second edition 2008-02-01, International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC), Case postale 56, CH-1211 Geneve 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

5. ISO/IEC/IEEE, IEEE Std. 12207-2008, *Systems and software engineering – Software life cycle processes*, Second edition 2008-02-01, International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC), Case postale 56, CH-1211 Geneve 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

1.5 Overview of Document

Section 2 of this document describes the concerns and constraints of the system and software, with respect to environmental constraints, system requirement constraints, and user characteristic constraints. Section 2 also describes stakeholder concerns.

Section 3 of this document describes the System and Software architecture of Swords and Sorcery, from different points of view, namely the user's point of view and the developer's point of view.

Section 4 of this document describes the finer details of the software design, listing software and system components that are crucial to the operation of the Swords and Sorcery game.

Section 5 of this document describes the requirements traceability of the Swords and Sorcery project, highlighting how our original project requirements have been met, modified, and implemented.

1.6 Document Restrictions

This document is for LIMITED USE ONLY to UI CS personnel working on the project.

2 Constraints and Concerns

2.1 Constraints

Swords and Sorcery requires the Java Runtime Environment 8 to run. S&S also requires the JDK 8.0 and Netbeans 8.0 or better to develop. The game will run on Windows, Mac, and Linux, provided those systems have the mentioned software packages (JRE to play, JDK/Netbeans to develop). To play the game, a network connection is required, as well as the IP Address of the game server. As S&S is a multiplayer game, one client is required for each player. Typically, this should be done using multiple computers, however, multiple clients can be run on the same computer.

2.2 Stakeholder Concerns

There are no financial stakeholders, however, Dr. Clinton Jeffery can be considered a stakeholder, as well as each class member. As a class member, our concerns are providing a quality product, while Dr. Jeffery's concerns may include using good design principles, as this is a Software Engineering course. Other concerns include design requirements such as portability,

3 System and Software Architecture

3.1 Developer's Architectural View

3.1.1 Developer's View Identification

There are multiple developer views within the scope of the S&S project for CS383. The views represent each subteam, and there are three subteams - HUD Team, Rules Team, and Networking Team. The descriptions of each sub-view follow:

HUD Team View The HUD Team view includes all software related to the HUD/GUI and handles how the user(s) interact with the S&S game.

Rules Team View The Rules Team view includes all software implementations of the S&S rule set, some of which overlaps with other views. Typically, Rules Team is in charge of implementing internal logic and data structures.

Networking Team View The Networking Team view includes all network communication related to the S&S game. This includes the client/server model, the communication protocols, the server setup and more.

3.1.2 Developer's View Representation and Description

3.1.3 Developer's Architectural Rationale

it breaks without text so often it seems

3.2 User's Architectural View

3.2.1 User's View Identification

3.2.2 User's View Representation and Description

3.3 Blank's Architectural View

3.3.1 Blank's View Identification

3.3.2 Blank's View Representation and Description

3.4 Consistency of Architectural Views

3.4.1 Detail of Inconsistencies Between Architectural Views

3.4.2 Consistency Analysis and Inconsistency Mitigations

4 Software Detailed Design

4.1 Developer's Viewpoint Detailed Software Design

4.2 Component Dictionary

Name	Type/Range	Purpose	Dependencies	Subordinates
Movement Calculator	Utility	Determine Legal Moves	UnitPool, MainMap	Retreat/Move

4.3 Component Detailed Design

4.3.1 Detailed Design for Component: Movement Calculator

Purpose The movement calculator is a static java class that handles most forms of movement.

Input The movement calculator takes two inputs to generate a list of moves: the unit moving, and the hex object they are beginning from. To calculate a retreat, the movement calculator takes as input the unit retreating, the hex they are retreating from, and the number of hexes they are required to retreat.

Output The movement calculator produces two main outputs: A hashmap of moves that a unit can reach (within the rules of movement specified by the board game) during a given movement phase, paired with their remaining movement cost after moving to a key hex in the hashmap, or an arraylist of moves that a unit can move to while retreating, during the combat phase.

Process The movement calculator uses recursion to examine the neighbors of the provided hex location. From each neighbor, it evaluates their neighbors, and so on. In both cases (movement/retreat) the recursion is terminated by reaching a lower bound (0) on the limiting value for their movement. For a moving unit, this is their given movement allowance per turn. For a retreating unit, this is the number generated from the combat results table that indicates how many hexes a unit must retreat. For each step of recursion, decisions are made within control flow that are designed to model the rules of the original S&S board game. These factors

include, but are not limited to, hex terrain types, hex edge types, geographical obstacles, and enemy occupation.

Design Constraints and Performance Requirements The design was constrained by two factors - code complexity and time. By designing the movement calculator to use recursion, the complexity of the component was greatly reduced. However, due to the many factors involved in movement, the design is still complex. Also, the moves available to a unit need to be calculated quickly. However, recursion is not very fast. Thankfully, Colin Clifford added some optimization code to the calculator, which has greatly increased performance with respect to time.

4.4 Data Dictionary

Name	Type/Range	Defined By...	Referenced By...	Modified By...
UnitPool	HashMap	UnitPool.java	Movement Calculator,...	HUDController

5 Requirements Traceability

5.1 Movement

Requirements Description Our requirement for movement was that a unit would be selected from the GUI and the GUI would highlight all available moves for the given unit. The player could then select the desired location for movement and the unit would move there.

Implementation Description Our implementation of movement uses recursion to generate a list of available moves that are highlighted in the GUI. The moves are then displayed as highlighted hexes. When the controlling player then right-clicks the desired hex (within the highlighted set), the unit moves to the indicated hex.

Differences There is no difference between our requirement for movement and our implementation of movement.

6 Appendix A