Software Design Specification (SDS) Template

In the template, the parts in *italic* are parts that you are supposed to expand on. The parts in ***bold and italics*** are explanatory comments and are provided just for your understanding of the document.

Complete and tailor the document by expanding the relevant parts and removing explanatory comments as you go along. If you decide to omit a section, you might keep the header and insert a comment saying why you omitted the data.

(Team Name)

(Team Logo)

Software Design Specification

Document

Version: (n) **Date: (mm/dd/yyyy)**

Table of Contents

1 Introduction

The following subsections of the Software Design Specifications (SDS) document should provide an overview of the entire SDS. The thing to keep in mind as you write this document is that you are telling how the system should do what it is supposed to do, so that the system can be implemented.

1.1 System Overview

Provide a brief high-level description of system structure, functionality, interactions with external systems, system issues, etc. If you wrote a good overview for the project plan and requirements document, then this is a simple paste job.

1.2 Definitions, Acronyms, and Abbreviations

Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SDS.

1.3 References

List any references or related materials here. For instance, if the design required interfacing with an X10 hardware devices, then you would want a reference the X10 specification.

In this section:

(1) *Provide a complete list of all documents referenced elsewhere in the SDS*
(2) *Identify the document by title, report number (if applicable), date, and publishing organization*
(3) *Specify the sources from which the references can be obtained*

This information can be provided by reference to an appendix or to another document.

1.4 Design Map

Define all major sections of this document and provide a brief summary of each.

2 Design Considerations

These subsections describe issues that need to be addressed or resolved prior to or while completing the design as well as issues that may influence the design process.

2.1 Assumptions

Describe any assumption, background, or dependencies of the software, its use, the operational environment, or significant project issues. These are things that you are assuming to be true and that directly affect the design.

2.2 Constraints

Describe any constraints on the system that have a significant impact on the design of the system. (e.g., technology constraints, performance requirements, end user characteristics) These are things the customer has told you that directly influence the design (e.g., the DB must be an open-source, freely available DBMS).

2.3 System Environment

Describe the hardware and software that the system must operate in and interact with.

2.4 Design Methodology

Summarize the approach that will be used to create and evolve the design for this system. This is not a rehash of your project lifecycle or change-management plan. This is for stating whether you will use object-oriented design, formal specifications, or other specific methodologies. Most people will use some object-oriented technique with UML.

3 Architectural (High-level) Design

The architecture provides the top-level design view of a system and provides a basis for more detailed design work. These subsections describe the top-level components of the system you are building and their relationships. For an OO implementation in Java, for example, our components could become packages (or set of packages, depending on the level of granularity considered and the size of the system).

In defining your architectural design, you can follow one of the organizational styles seen in class (shared data repository, shared services and servers, and abstract machine/layered) or pick a different one if none of those is appropriate for your system.

3.1 Overview

This section provides a high level overview of the structural and functional decomposition of the system. The section should list the different components and concisely discuss the major responsibilities and roles such components must play.

3.2 Rationale

This section discusses why you are using the architecture you have chosen.

3.3 Conceptual (or Logical) View

This section should provide and describe a diagram that shows the various components and how they are connected. The conceptual view shows the logical/functional components of the system, where each component represents a cluster of related functionality. For UML, this

would typically be a component diagram or a package diagram.

3.4 Other Views

High-level designs are most effective if they attempt to model groups of system elements from a number of different views. Beside the Conceptual/Logical view, examples of additional viewpoints are:

(a) ***Process View: this represents the runtime view of the system. The components are threads, processes, or distributed components. In UML, this would typically be a process interaction diagram.***

(b) ***Physical View: this view is for distributed systems. The components are physical processors that have parts of the system running on them. For UML, this would be a deployment diagram.***

(c) ***Module View: this view is for project management and code organization. The components are typically files or directories. This view shows how the directory structure of the build and development environment will be designed.***

(d) ***Security View: this view typically focuses on the components that cooperate to provide security features of the system. It is often a subset of the Conceptual view.***

(e) **…**

Note that it is not necessary to document all these views. For many smaller applications, the conceptual view is all that is necessary. Document those views that will help you design and implement the system and create a subsection for each one of them.

4 Low Level Design

This section provides the low-level design for each of the system components identified in the previous section. For each component, you should provide a subsection that shows its internal structure. In the case of an OO design, this internal structure would typically be expressed as an UML class diagram that represents the static class structure for the component. For smaller systems, you may have a single UML class diagram that each component description refers to.

4.1 Component 1

4.2 Component 2

4.n Component n

As discussed above, these subsections should provide and discuss detailed diagrams of each software module. For at least some of the components, you should provide diagrams that show a dynamic view of the component internals (i.e., that show the dynamic interaction between classes). In the case of an OO design, UML state or interaction diagrams can be

used to this end.

5 User Interface Design

These subsections discuss the user interface design.

5.1 Application Control

This section details the common behavior that all screens will have. Common look and feel details, such as menus, popup menus, toolbars, status bars, title bars, drag and drop, and mouse behavior should be described here.

5.2 Screen 1

5.3 Screen 2

5.n Screen n

These sections illustrate all major user-interface screens and describe the behavior and state changes that the user will experience. A screen transition diagram or table can optionally be created to illustrate the flow of control through the various screens.

Note that these sections may not show actual screenshots (in case you have not completed the implementation yet). In these cases, they can be drawings or mockups created using some rapid GUI-building tool.