ZUCI SYSTEMS

SOFTWARE DESIGN SPECIFICATION TEMPLATE

TABLE OF CONTENTS

1. Introduction

This is an attempt to put together a comprehensive template for the specification of software designs, with guidelines on how to structure the contents of various sections and subsections of the document.

The template has been created using suggestions from various SEI reports and even IEEE documentation standards for software designs and software requirements.

We believe that a completed software design specification document should meet the following criteria:

- It should be able to serve as training material for new project members, giving them them enough information and understanding about the project implementation,.

- It should serve as "objective evidence" that the designers and/or implementers are following through on their commitment to implement the functionality described in the requirements specification.

- It needs to be as detailed as possible, without imposing too much of a burden on the designers and/or implementers and without becoming overly difficult to create or maintain.

1.1. Document Outline

Here is the outline of the proposed template for software design specifications. What follows is just one suggested outline format to use when attempting to present the architecture and design of the entire system as one single document:

- Introduction
- System Overview
- Design Considerations
  - Assumptions and Dependencies
  - General Constraints
  - Goals and Guidelines
  - Development Methods
- Architectural Strategies
  - strategy-1 name or description
  - strategy-2 name or description
- System Architecture
  - component-1 name or description
  - component-2 name or description

- Policies and Tactics
    - policy/tactic-1 name or description
    - policy/tactic-2 name or description
    - (add if needed)

- Detailed System Design
    - module-1 name or description
    - module-2 name or description
    - (add if needed)
- Glossary
- Bibliography

The above outline is only prescriptive. You can follow your own style of numbering as well as sections and subsections as you feel appropriate. For example, some development firms like to place the glossary in the beginning of the document.

The same template is intended for both high-level design and low-level design. The design document used for high-level design is an active document, i.e. it evolves to include low-level design details.

The ordering of the sections in this template correspond to the typical order in which decisions are made during the software design process. While the software design process may not always follow a linear process, this template is useful for the purpose of understanding the design of the system to present them as if they did and also comes in very handy during a formal review. Even if the information is being stored informally, like in a journal, this template can serve as an outline.

1.2. Document Description

Here is the description of the contents of the proposed template for software design specifications:

1.2.1. Introduction

Overview of the entire software design specification document:

- Purpose of this document
- Scope of this document
- Intended audience/ users of the document
- Identify the system/product using any applicable names and/or version numbers.
- References to any other related documents such as:

    - Related and/or companion documents
    - Prerequisite documents
    - Documents which provide background and/or context for this document
    - Documents that could result from this document
- Define any important terms, acronyms, or abbreviations

1.2.2. System Overview

General description of the software system including its functionality and matters related to the overall system and its design (perhaps including a discussion of the basic design approach or organization). This section can be split into subsections as needed.

2. Design Considerations

This section describes many of the issues that need to be considered or resolved before attempting to devise a complete design solution.

2.1. Assumptions and Dependencies

List any assumptions or dependencies regarding the software and its use. These may concern issues like:

- Related software or hardware
- Operating systems
- End-user characteristics
- Possible changes in a functionality

2.2. General Constraints

Describe any higher-level limitations that could impact the design of the system's software. Here’s a (non-exhaustive) list of such constraints:

- Hardware or software environment
- End-user environment
- Availability or volatility of resources
- Standards compliance
- Interoperability requirements
- Interface/protocol requirements

- Data repository and distribution requirements
- Security and regulatory requirements
- Memory and other capacity limitations
- Performance requirements
- Network communications
- Verification and validation requirements (testing)
- Other means of addressing quality goals
- Other requirements described in the requirements specification

2.3. Goals and Guidelines

Describe any goals, guidelines, principles, or priorities that dominate or embody the design of the system's software. Such goals might be:

- The KISS principle (keeping it simple)
- Emphasis on speed versus memory use
- Working, looking, or "feeling" like an existing product

2.4. Development Methods

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of each method.

3. Architectural Strategies

Describe any design decisions that affect the overall organization of the system and its higher-level structures, providing insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and how any design goals or priorities were balanced off. Such decisions might concern things like:

- Use of a particular type of product (programming language, database, library, etc.)
- Reuse of existing software components to implement various parts/features of the system
- Future plans for extending or enhancing the software
- User interface paradigms (or system input and output models)
- Hardware and/or software interface paradigms
- Error detection and recovery

- Memory management policies
- External databases and/or data storage management and persistence
- Distributed data or control over a network
- Generalized approaches to control
- Concurrency and synchronization
- Communication mechanisms
- Management of other resources

Each significant strategy employed should probably be discussed separately in this document, or if it is vital then in a separate document. It’s a good practice that when describing a design decision you also discuss any other significant alternatives that were considered, and your reasons for rejecting them.

4. System Architecture

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves and provide a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

Begin with the major responsibilities that the software must undertake and the various roles that the system must perform. Describe how the system was broken down into its components/subsystems (identifying each top-level component/subsystem and the roles/responsibilities assigned to it). Describe how the higher-level components collaborate with each other in order to achieve the required results, and the rationale behind these choices.

If there are any diagrams, models, flowcharts, documented scenarios or use-cases of the system behavior and/or structure, they may be included here.

Note:
This section (and its subsections) really applies only to newly developed (or yet-to-be developed) portions of the system. Pre-existing parts that are modified or enhanced need to be described only to the extent that is necessary for the reader to gain a sufficient understanding.

4.1. Subsystem Architecture

If a particular component merits a more detailed discussion than what was presented in the System Architecture section, provide that in a subsection of the

System Architecture section. If necessary, describe how the component was further divided into subcomponents, and the relationships and interactions between the subcomponents.

If any subcomponents are also deemed to merit further discussion, then describe them in a separate subsection of this section (and in a similar fashion). Proceed to go into as many levels/subsections of discussion as needed in order for the reader to gain a high-level understanding of the entire system or subsystem.

5. Policies and Tactics

Describe any design policies and/or tactics that would not significantly affect the overall organization of the system and its high-level structures, but which nonetheless affect the details of the interface and/or implementation of various aspects of the system. These could include considerations such as:

- Choice of specific product to use (compiler, interpreter, database, library, etc.)
- Engineering trade-offs
- Coding guidelines and conventions
- Protocol of one or more subsystems, modules, or subroutines
- Choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality
- Plans for ensuring requirements traceability
- Plans for testing the software
- Plans for maintaining the software
- Interfaces for end-users, software, hardware, and communications
- Hierarchical organization of the source code into its physical components (files and directories).
- How to build and/or generate the system's deliverables

Each particular policy or set of tactics employed should probably be discussed in its own subsection. Make sure that when describing a design decision you must also discuss any other significant alternatives that were considered, and your reasons for rejecting them.

In this section, it may become difficult to decide whether a particular policy or set of tactics should be discussed here, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. Use your own judgment to decide this.

6. Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Other lower-level components and subcomponents may need to be described as well. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

6.1. Classification

The kind of component, such as a subsystem, module, class, package, function, file, etc.

6.2. Definition

Specific purpose and semantic meaning of the component.

6.3. Responsibilities

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? And so on.

6.4. Constraints

Any relevant assumptions, limitations, or constraints for this component, such as constraints on timing, storage, or component state, etc.

6.5. Composition

A description of the use and meaning of the subcomponents that are a part of this component.

6.6. Uses/Interactions

A description of this component’s collaboration with other components. What other components is this entity used by? What other components does this entity use This concerns the method of interaction as well as the interaction itself..

6.7. Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library.

6.8. Processing

A description of how this component goes about performing the duties necessary to fulfill its responsibilities. This should encompass a description of algorithms used; changes of state; relevant time or space complexity; concurrency; methods of creation, initialization, and cleanup; and handling of exceptional conditions.

6.9. Interface/Exports

The set of services, such as resources, data, types, constants, subroutines, and exceptions, that this component provides. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. and software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code. It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.

6.10. Detailed Subsystem Design

Provide a detailed description of this software component. Complex diagrams showing the details of component structure, behavior, or information/control flow may be included in the subsection devoted to that particular component. The description should cover any applicable software component attributes.

7. Glossary

An ordered list of defined terms used throughout the document.

8. Bibliography

A list of referenced and/or related publications.

GET IN TOUCH

: sales@zucisystems.com

: US: +1 (331) 903-5007
India: +91 (44) 49525020

: Chennai, Tamil Nadu
Chicago, Illinois

www.zucisystems.com