Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The SDD shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. You should be aware of this, before writing your reports.

One template is prepared for both Initial Design Report and Detailed Design Report. In the initial design, the overall system architecture and data architecture is defined. In the detailed design report, more detailed data structures are defined and algorithms are developed for the defined architecture. So you can omit the Detailed Design section in your Initial Design Report. But still you should give the design of the subsystems/modules of the system in the Initial Design Report. More detailed and complex sub components can be described in the Detailed Design Report.

In the following template, we didn't state which diagrams you should give. However, we expect you to draw UML diagrams like use case, class, data flow, component, sequence, activity diagrams, etc. Draw all types of necessary diagrams which are suitable for your project (for example; if you are following an object-oriented approach, draw the class diagrams) and which are enough to represent the whole design and architecture of your system. And obviously, these diagrams can be less detailed in the Initial Design Report than the Detailed Design Report.

Software Design Report

1. Introduction

Provide an overview of the entire design document.

1.1. Problem Definition

Give a detailed problem definition.

1.2. Purpose

Describe the purpose of this SDD and its intended audience.

1.3. Scope

Describe the scope of this document

1.4. Overview

Summarize the contents of this document

1.5. Definitions, Acronyms and Abbreviations

Define any important terms, acronyms, or abbreviations

1.6. References

Provide references for any other pertinent documents

2. System Overview

Provide a general description of the software system including its functionality and matters related to the overall system and its design. Briefly explain the goals, objectives and benefits of your project. This will provide the basis for the brief description of your product.

3. Design Considerations

Special design issues which need to be addressed or resolved before attempting to devise a complete design solution are noted here.

3.1. Design Assumptions, Dependencies and Constraints

Describe any assumptions or dependencies regarding the software and its use (For example; related software or hardware, end-user characteristics, etc). Describe any global limitations or constraints that have a significant impact on the design of the system's software (For example; hardware or software environment, time constraints, security constraints, standards compliance, availability or volatility of resources, performance constraints, etc.).

3.2. Design Goals and Guidelines

Describe any goals, guidelines, or priorities which dominate or embody the design of the system's software (For example; the KISS principle, emphasis on speed versus memory use, Don't Repeat Yourself principle, portability, or usability, etc.). For each such goal or guideline, unless it is implicitly obvious, describe the desirability.

4. Data Design

4.1. Data Description

Explain how the information domain of your system is transformed into data structures. (These data structures can be files that are created for temporary use, or data structures which are passing among components of the software, or data structures which are available to major portions of the architecture, etc..) Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items. Describe the database(s) created as a part of the applications and provide enough detail to create the database.

4.2. Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 5.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

5. System Architecture

A description of the program architecture is presented here.

5.1. Architectural Design

Describe the system structure chosen for the application. A pictorial representation, using a UML component diagram, of the architecture is presented (show the major subsystems and data repositories and their interconnections).

Develop a modular system structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems. Identify each high level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. Try not

to go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together.

If there are any diagrams, models, flowcharts, documented scenarios or use-cases of the system behavior and/or structure, they may be included here or if they are too complex include them in the Detailed System Design section. Diagrams that describe a particular component or subsystem should be included within the particular subsection that describes that component or subsystem.

5.2. Description of Components

Provide a decomposition of the subsystems in the architectural design. Depending on your design, you can give a functional description or an object-oriented description. For a functional description, put top level data flow diagram (DFD) and structural decomposition diagrams. For an OO description, put subsystem model, object diagrams, generalization hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), interface specifications, and sequence diagrams here. If diagrams are too complex, put them under the *Detailed Design* section. State the responsibilities of the components. State the the interacting components. Explain how each component works; describe the algorithms used(the description for components can be organized like below).

You can explain the component in subsections. 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 (but remember to leave the deep details for the Detailed Design *section). If you feel the component is a very particular one, you don't have to explain it in this section, and if the component is very detailed then you can explain the details in Detailed Design section; in that case state that further details can be found in Detailed Design section.Note that; in this section the subsystems or main modules of the main system should definetely be described here. According to various circumtances, the sub components or the details of these modules can be described in the Detailed Design section.*

5.2.1. Component n

Describe the software component. *(According to the detail level of the component you can either explain the expected information about the component in the below subsections or in the Detail Design section)*

5.2.1.1. Processing narrative for component n

Present a processing narrative for component n. It should describe the responsibilities of the component.

5.2.1.2. Component n interface description

Describe of the input and output interfaces for the component.

5.2.1.3. Component n processing detail

Present an algorithmic description for the component.

5.2.1.4. Dynamic behavior component n

Prensent a description of the interaction of the classes. Present a sequence diagram for each use case the component realizes.

5.2.2. Component n+1

Describe the software component.

5.2.2.1. Processing narrative for component n+1

Present a processing narrative for component n. It should describe the responsibilities of the component.

5.2.2.2. Component n+1 interface description

Describe of the input and output interfaces for the component.

5.2.2.3. Component n+1 processing detail

Present an algorithmic description for the component.

5.2.2.4. Dynamic Behavior component n+1

Prensent a description of the interaction of the classes. Present a sequence diagram for each use case the component realizes.

5.3. Design Rationale

Provide some sort of rationale for choosing this particular decomposition of the system including critical issues and trade/offs that were considered. Discuss other alternative designs that were considered, and why they were rejected.

6. User Interface Design

6.1. Overview of User Interface

Describe the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

6.2. Screen Images

Display screenshots showing the interface from the user's perspective.

6.3. Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

7. Detailed Design

It contains the internal details of each design entity/component. These details include attribute descriptions for identification, processing and data. It contains all the details that will be needed by the programmers for implementation. Short English-like descriptions can be used to describe the algorithms utilized. Data structure details should also be given.

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:

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

Definition: The specific purpose and semantic meaning of the component. This may need to refer back to the requirements specification.

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? For some components, this may need to refer back to the requirements specification.

Constraints: Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, postconditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

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

Uses/Interactions: A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclasses, and metaclasses.

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. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

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

Interface/Exports: The set of services (resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. .... For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Include necessary diagrams like class diagrams, sequence diagrams, etc.

8. Libraries and Tools

9. Time Planning (Gannt Chart)

9.1. Term 1 Gannt Chart

9.2. Term 2 Gannt Chart

10. Conclusion