Software Design Document (SDD) Template

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. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, i.e. the detailed design stage, more detailed data structures are defined and algorithms are developed for the defined architecture.

This template is an annotated outline for a software design document adapted from the IEEE Recommended Practice for Software Design Descriptions. The IEEE Recommended Practice for Software Design Descriptions have been reduced in order to simplify this assignment while still retaining the main components and providing a general idea of a project definition report. For your own information, please refer to [I EE E St d 1016­ 1998](http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee-SDD.pdf) 1 for the full IEEE Recommended Practice for Software Design Descriptions.

1 http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee­SDD.pdf

TLD(traffic light decider)

Software Design Document

Name (s): Igor krol, Denis Shapira

Lab Section:

Workstation:

Date: (31/3/2020)

**TABLE OF CONTENTS**

**1. INTRODUCTION 2**

1.1 Purpose 2

1.2 Scope 2

1.3 Overview 2

1.4 Reference Material 2

1.5 Definitions and Acronyms 2

**2. SYSTEM OVERVIEW 2**

**3. SYSTEM ARCHITECTURE 2**

3.1 Architectural Design 2

3.2 Decomposition Description 3

3.3 Design Rationale 3

**4. DATA DESIGN 3**

4.1 Data Description 3

4.2 Data Dictionary 3

**5. COMPONENT DESIGN 3**

**6. HUMAN INTERFACE DESIGN 4**

6.1 Overview of User Interface 4

6.2 Screen Images 4

6.3 Screen Objects and Actions 4

**7. REQUIREMENTS MATRIX 4**

**8. APPENDICES 4**

**1. INTRODUCTION**

**1.1 Purpose**

This document contains the complete design description of the *TLD*. This includes the architectural features of the system down through details of what operations each code module will perform and the database layout. It also shows how the use cases detailed in the SRS will be implemented in the system using this design.

The primary audiences of this document are the software developers.

**1.2 Scope**

There are three parts of this system:

* The *data collection and adaptation* gathers information from traffic and adapts it to the algorithm.
* The *algorithm(TLD)* based on the data decides on the lights on each traffic-light for a period of time(cycle).
* The *results adaptation* displays the cycle on the screen with the data from the algorithm.

**1.3 Overview**

Provide an overview of this document and its organization.

**1.4 Reference Material**

*This section is optional.*

List any documents, if any, which were used as sources of information for the test plan.

**1.5 Definitions and Acronyms**

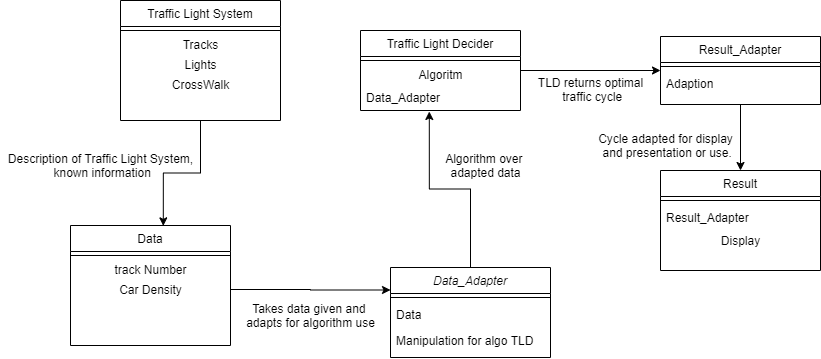
|  |  |
| --- | --- |
| Term | Definition |
| TLD | Traffic light decider |
| Cycle | Results over a period of time |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**2. SYSTEM OVERVIEW**

Give a general description of the functionality, context and design of your project. Provide any background information if necessary.

**3. SYSTEM ARCHITECTURE**

**3.1 Architectural Design**



**3.2 Decomposition Description**

Traffic light system: data which describes the module of the traffic with information

Which doesn’t change.

Data: the information that the system will gather to decide the next cycle.

Data\_Adapter: manipulates the data from Data to better suite TLD.

TLD: the algorithm that calculates the next cycle.

Result\_adapter: manipulates the data from TLD to better suite Result.

Result: displays the information it receives on the screen.

**3.3 Design Rationale**

When deciding on the architectural design we pondered in which way we wanted the software to be divided, After deciding on 3 part division we tried to think on the implementation of each part and realized that we needed to adapt and manipulate the data from each part so we could improve a part without changing the whole software. Or if someone had a better algorithm for example he could switch that part and he will have to change only the adapters.

**4. DATA DESIGN**

**4.1 Data Description**

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

**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 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

**5. COMPONENT DESIGN**

In this section, we take a closer look at what each component does in a more systematic way. If

you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.

**6. HUMAN 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. These can be hand­ drawn or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

**6.3 Screen Objects and Actions**

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

**7. REQUIREMENTS MATRIX**

Provide a cross­reference that traces components and data structures to the requirements in your

SRS document.

Use a tabular format to show which system components satisfy each of the functional requirements from the SRS. Refer to the functional requirements by the numbers/codes that you gave them in the SRS.

**8. APPENDICES**

*This section is optional.*

Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document.