SWENG 894 Capstone

Software Requirements Specification

# Introduction

This document is intended to describe the requirements of the “lzsmgen” application. Its development will follow Agile conventions, so the document will evolve over the course of the project. Items in *non-emphasized* *italics* are not to be considered as part of the requirements, but nevertheless include useful information that may be added to other documents, such as the user manual, API documents, etc.

## Purpose: Mission statement

The purpose of the application is to provide an easy-to-use tool for generating code from UML state chart diagrams.

*At some point, I would like to add support for additional UML diagrams, but this is probably beyond the scope of the project.*

## Scope

The program is a desktop application that consists of one (or two) executables and additional supporting shared libraries. In one mode of operation the application runs with user intervention, while in the other mode, the user interacts with the application via a GUI.

If the GUI is sufficiently decoupled, there may be the possibility of controlling the application via a web interface.

## Product overview

### Product perspective

In the current conception of the product, it resides as a standard application on a Microsoft Windows or Linux operating system. It is opened and responds to mouse and keyboard inputs in the same manner as other applications on those systems. Under certain conditions the product shall not open a window, but simply run to conclusion without operator interaction. The application reads and interprets certain files and may write files to computer storage media.

### Product functions

The product is intended to facilitate code design, explanation of system operation, and software builds. For code design, while it will not eliminate coding, the user will more readily be able to visualize the resulting system’s operation. In a like manner, the visualization may be shared with stakeholders, using exported images. Lastly, the diagram becomes the “code” for building, where the diagram is “compiled” into code and integrated with the rest of the user’s software product.

### User characteristics

A typical user might be a hobbyist programmer, interested in creating reactive systems for embedded systems (such as Raspberry Pi or Arduino). Professional software developers may also find the product useful.

### Limitations

In order to use the program, the user will need to have some expertise in the target language.

## Definitions

|  |  |
| --- | --- |
| Term | Definition |
| Reactive system | A system which continues to run indefinitely, responding to various events like button clicks, switch operations, and other condition changes. |
| Target language | Language that the system will generate. |

# References

1. Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming, 8, 231–274.
2. <https://www.omg.org/spec/UML>

# Specific Requirements

## External interfaces

### Deployment

The system shall reside on a user’s PC on storage media such as a hard drive or USB stick. *Non-functional*  
*requirement*

### File access

The system shall only read or write files permitted by the parent operating system. *Non-functional requirement – security? Do we need to restrict access to files external to the user’s PC?*

## Functions

### The system shall provide for operation without user interaction.

### If the user does not elect to initiate the system with user interaction, a GUI shall be presented.

*The GUI will probably include an edit area, a menu, and a component pallet.*

#### The program GUI shall enable the user to load, save, and create diagram files.

#### The program GUI must enable the user to place diagram components.

*TODO: The component could automatically be resized according to the component characteristics, or the user could be permitted to change the size manually.*

#### The program GUI shall provide a means of editing component characteristics.

There are two types of characteristics, those that will affect the generated code and cosmetic characteristics (like color and dimensions).

#### The program GUI shall enable the user to select a target language.

#### The program GUI shall provide a means of user customization.

#### The program GUI must provide a means of program termination.

### The system shall be able to export an image file based on the currently loaded diagram.

### The system shall be able to generate a code file in the selected language from a currently loaded diagram.

### The system shall indicate if it cannot generate code from the currently loaded diagram.

### The system shall permit the user to reference external modules in generated code.

Generated code is not expected to run alone. For example, C/C++ uses a preprocessor directive (#include) to enable modules to use external libraries and functions while Python uses the “import” directive.

### The system shall allow the user to configure the exports of generated code.

The system shall generate a default interface. This requirement allows the user to adjust that.

### The system shall have a help function for all user entry possibilities and diagram components.

### The system shall warn the user if the user attempts to close the program which has a currently loaded diagram with unsaved changes.

### Program configuration settings shall be saved between program runs.

## Usability requirements

TBD.

## Performance requirements

### Generated code characteristics

The generated code shall implement a table-based state machine. *Should the user have the option of a switch-case machine?* It will also include a standard event/message pump. For C/C++, one source and one header file shall be generated.

### Speed requirements

Not applicable. *Performance may be governed by the characteristics of the system in which the program*  
*is installed. This may need to be revisited.*

## Logical database requirements

Not applicable. *This section may be removed.*

## Design constraints

## Software system attributes

The system shall be operable on Linux or Microsoft Windows. *Non-functional requirement.*

## Supporting information

# Verification

*This section will eventually mirror section 3.*

# Appendices

## Assumptions and dependencies

## Acronyms and abbreviations

|  |  |
| --- | --- |
| Term | Meaning |
| API | Application programming interface |
| FSM | Finite state machine |
| GUI | Graphical user interface |
| PC | Personal computer |
| TBD | To be determined |
| UML | Unified Modelling Language |