SWENG 894 Capstone

Initial Concept of Operations

# Capstone Overview

The Penn State World Campus MSE program includes a culminating Capstone project. The purpose of this document is to introduce a proposed project for further discussion and exploration with the instructor.

This document includes a brief description of my candidate project for my SWENG 894 Capstone. Sections in this document include approach, assumptions, mission statement, solution description, and the required estimated details: domain model, programming language and tool chain.

## Approach

I shall use an Agile approach to achieve a minimum viable product (MVP). I have never truly used Agile before (although I have worked for companies that claim Agile). This is consistent with the capstone project timeline of 14 weeks. I would also like to have the documentation available to continue the project after the conclusion of the MSE program.

**Rationale**: My primary focus is the development of embedded automotive software. Most of my colleagues are not trained in software engineering. (This includes some individuals who are writing software.) I have not used any Agile tools in project management previously. I intend to better understand Agile, evaluate its effectiveness, and influence decision-makers in my company. The tool I intend to build will be immediately useful to my work.

## Foundational Assumptions

* I am well-versed in the problem domain; however, I expect objectives and subsequent decomposition of scenarios will evolve given that some issues are unforeseen.
* All languages and tools identified in this document are subject to change.
* An agile-like approach will be leveraged. Given this is an individual effort, I will personally be managing my effort and time using a Kanban approach for daily management as mapped to the three pseudo-sprints defined for the Capstone program.  
  a) This supports the imperative for innovation and reacting to discoveries.  
  b) This supports the CAPSTONE mandates for traditional/historic artifacts at a timed basis.
* Under normal professional circumstances, tools and languages are selected using a tradeoff analysis based on requirements and scenarios. Given the rapid, simulated approach, this is not possible at the depth requisite a non-academic effort.
* NOTE: To the extent plausible, as determined from my vantage point, this capstone will leverage learnings, examples, and homework assignments from prior classes in this MSE program.
* This document is not a formal SRS (systems requirements document). Detailed functions have not been outlined.
* Future artifacts may include (Use Case Diagram, Context Diagram, and Sequence Diagrams).

# Introduction

My capstone project will be named lzsmgen: lz is an abbreviation for Lititz, a small town in Lancaster County, Pennsylvania, while sm and gen stand for state machine and generator, respectively. While I expect to support at least one additional UML diagram, the state machine diagram is the primary artifact to be supported.

## Mission

The mission of lzsmgen is to provide an easy-to-use tool for generating code from UML diagrams.

## Background and Problem Statement

The current tools for generating code from UML tend to be expensive and do not readily lend themselves to be used within build scripts.

Many embedded systems may be termed reactive systems and can be conveniently modelled with hierarchal finite state machines. The original concept was pioneered by David Harel[[1]](#footnote-1). Most engineers I have encountered can comprehend these diagrams. In contrast, many other UML artifacts are not useful for non-software people.

I used a program called BetterState to generate code from state diagrams prior to 2000. The Mathworks created a similar application called StateFlow and it was bundled with their Matlab/Simulink suite. Soon after, The Mathworks bought BetterState and eventually discontinued it. All that remains of BetterState is a quite limited freeware version called BetterState Lite.

Since then, I have searched for a BetterState replacement. StateFlow is an excellent product, but it requires Matlab and Simulink. I have researched several open-source tools including those in the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Solution | Web | MS Windows | Linux | Notes |
| PlantUML | https://plantuml.com/ | Yes | Yes | Requires separate program to create diagram images. |
| ragel | http://www.colm.net/ | No | Yes | Command line tool |
| UMLet | https://www.umlet.com/ | Yes | Yes |  |
| Umbrello | https://apps.kde.org/umbrello/ | Testing | Yes | Originally only available on Linux. |

None of these satisfied my requirements (listed in the key features section).

## Target Audience

The target audience for this solution are hobbyist software developers and those developers working for organizations where the cost of a more full-featured solution is not justified. Higher level managers of said organizations may also be stakeholders. Indirectly users of generated software will be affected.

## Key Features

The lzsmgen application should:

* Allow the user to quickly draw hierarchal state machines and at least one other type of UML diagram.
* Allow diagram to be exported into formats for use in web pages or other presentations.
* Include a command line interface to allow the tool to be operated without user interaction in a build script.
* Store user diagrams on computer storage media.
* Support both C/C++ code generation. (The tool should not be limited to generating only object-oriented languages.)
* Include option for generating MISRA-C compliant code.
* Be operable on MS Windows and Linux.
* Be consistent with other desktop applications. (Intuitive)

## Additional Considerations

The effort should use open-source software components and standard file formats where possible for speed of development.

# Domain Model

I expect to use some variation of the Model-View-Controller design pattern for elements of this project.

A diagram of a diagram

Description automatically generated

## Languages

The following list of languages is representational and may change based on discoveries during early prototyping and innovation. Supporting libraries are not selected in their totality.

|  |  |  |
| --- | --- | --- |
| Purpose | Language | Rationale |
| Core/GUI | GNU C++ | * Common language between operating systems * High level of experience with * No requirement for runtime/scripting language * Available GUI libraries |
| Configuration | Lua | * Small footprint * Easy to embed |
| Build Scripting | Lua/Lake | * Common with configuration * Portability * Lake scripts can use GNU C++ and Visual C++ with little to no change |
| Infrastructure as Code | XML/JSON | * Large number of parsing libraries |

## Toolchain

This listing is intended to give a sense of the tools being used though it is not an exhaustive listing given the likelihood of new tools being necessary and leveraged.

|  |  |  |
| --- | --- | --- |
| Type | Tool | Rationale |
| Integrated  Development  Environment (IDE) | Visual Studio Code | * Experience and comfortable working with * Support for integration with GitHub * Broad community of users enabling better online Q&A * Extensive marketplace of extension packs for languages, tools, and other integrations |
| Project Management | Jira | First time use |
| Source Code Repository | Github | * Industry leading * Free and hosted as a service * Used before * Aligns to commercial ventures |
| Compiler | GNU C++/Visual C++ | * GNU C++ is industry standard * Visual C++ is faster on MS Windows |
| Defect Tracking | Jira | First time use |
| Unit Test Framework | GoogleTest | * Linux and MS Windows * Respected developer (Google) |

Team Contract

# Team Member

|  |  |  |
| --- | --- | --- |
| Name | E-mail | Phone |
| James Esterby | [Jde95@psu.edu](mailto:Jde95@psu.edu) | (717) 201-0361 |

# Team Procedures

Not applicable

# Participation

I expect to spend at least 20 hours/week (the equivalent of a part-time job) on the project during the semester. Concurrently, I shall be working a full-time job (40+ hours/week) developing embedded software for an agricultural machine called a Windrower. There may be opportunity to check the output of the tool on this application, however, company intellectual property must not be compromised.

Work breakdown table:

|  |  |
| --- | --- |
| Activity | Time |
| Sprint planning | 5% (Agile daily meeting) |
| Requirements | 30% |
| Design | 25% |
| Implementation/Debugging | 10% |
| Testing | 30% |

“Without requirements and design, programming is the art of adding bugs to an empty text file.” –Louis Srygley (<https://www.defprogramming.com/quotes-by/louis-srygley/>)

# Roadblock Handling

## Set-up Issues

Most of the tools I plan to use are already installed on my machine. Problems with Jira shall be resolved by consulting the documentation first, followed by questioning colleagues and/or the course instructor. If necessary, I will use an alternate project management tool.

## Other Issues

Other issues shall be handled on a case-by-case basis. I do have resources at work as well as participating students. Note that PSU Academic Integrity will be carefully observed.

# Team Consequences

Not applicable

# Personal Accountability Statements

During the course of the project, I shall adhere to the PSU Code of Conduct (<https://studentaffairs.psu.edu/student-accountability/code-procedures/student-code-conduct>) and the standards described by PSU Academic Integrity documentation (<https://ed.psu.edu/current-students/academic-integrity/ai-form>)

1. Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming, 8, 231–274. [↑](#footnote-ref-1)