**FSM Generator**

**Software Architecture Specification**

# Contributors

Chris Loos

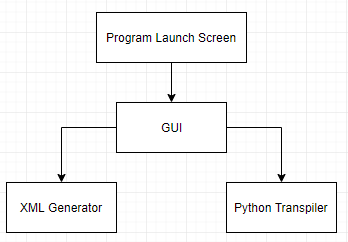
Justin Butler

# Proposed Software Architecture

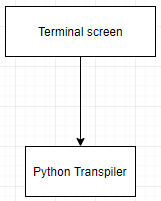
# Subsystem Decomposition

The system involves both a GUI and a terminal interface. The GUI allows the user to generate a finite state machine (FSM) graphically and optionally export it as a transpiled python file. The terminal accepts a pre-made FSM file to generate a transpiled python file.

## GUI Interface



## Terminal



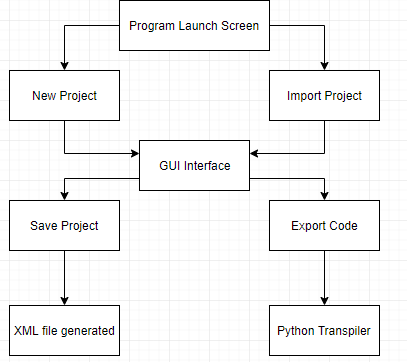
# Hardware / Software Mapping

Through the use of the Java programming language, the program should operate independent of machine through the Java virtual machine. The minimum requirement to run the Java virtual machine is 98 MB of free disk space and a Pentium 166 MHz processor with 64 MB of physical RAM. Python was also chosen as the initial only export due to its portability as a language.

# Persistent Data Management

Data will be stored locally to the user’s machine. Generators within the GUI will create an XML export of the design state at the time of saving or a FSM file based on the diagram created within the GUI. The XML file will contain all state nodes, links, accept states, and eventually functions to be reloaded with the import function. The FSM file is created using the same data, but in a pre-determined format that can be processed into a graphical JPG representation for printing or easier viewing without launching the program.

# Global Software Control



# Boundary Conditions

The program will run locally on the user’s machine and requires no additional access or authentication. The only major possibility for error is a corrupted XML file which the program can no longer import. Any additional language support will require additional methods and will need to be placed by an experienced Java programmer.