# SCOPE

This document outlines the specifications of the Goo State Machine Application.

# SUMMARY

The “Goo State Machine Application” (afterwards known as the “App”) creates an “add-in” for Microsoft Power-point (PP) to implement a state machine framework that allows state machine code to be generated automatically from the state diagram.

# REQUIREMENTS

The following sections outline the requirements for the App.

## GENERAL

### Language/Forma*t*

#### The App will be written in C#

#### The App will be written as a Microsoft PP “Add-In” using Visual Studio Community Edition.

### Workflow

#### The user will define the states and transitions using standard PP tools.

## POWER-POINT IMPLEMENTATION

### Format

#### State machines will be limited to a single slide and the slide page size determines the working area.

#### A presentation may have as many slides as needed. “Non-FSM” slides may be included.

### State Machine Objects

#### All state machine objects will be created via the custom ribbon. This allows “standard” versions of all the usual objects to be used in power point.

#### Custom data for state machine objects should be stored in a Dictionary using the name as the key.

### Slide

#### Individual FSM shall be implemented on a single Slide.

### States

#### FSM States shall be defined using standard PP rounded rectangles.

#### State names shall be assigned to the text of the rounded rectangle.

#### States shall have a default color and boundary when first created.

#### States color and boundary may be edited by normal PP means.

### Transitions

#### FSM Transitions shall be defined using standard PP arrows or curved connectors.

#### If not currently visible, FSM Transitions shall display the transition name when hovered over.

#### If visible, FSM Transitions shall display a textbox with the transition name.

## USER INTERFACE

### Custom Ribbon (FSM Interface)

#### A custom ribbon will be used as a user interface for operations.

#### Ribbon commands will operate on the current active slide.

#### Ribbon commands shall be divided into groups:

##### A group (***PRESENTATION***) shall be provided for the overall presentation.

##### A group (***SLIDE***) shall be provided for individual state machines (applies to current slide).

##### A group (***FSM***) shall be provided for FSM actions (applies to current slide).

#### The following commands shall be available in the ***PRESENTATION*** group:

##### Create New FSM

##### Generate Code (All Slides)

#### The following commands shall be available in the ***SLIDE*** group:

##### Edit FSM Parameters

##### Generate Code (Current Slide)

#### The following commands shall be available in the ***FSM*** group:

##### Create New State

##### Create New Transition

##### Edit State Machine Code

### Context-Sensitive “Right-Click”

#### The user shall be provided with a context sensitive menu of options by “right-clicking” the mouse.

#### Available options shall be dependent on what it is currently selected (if any). If no items are selected, the options shall be for the ***SLIDE*** operations.

## FINITE STATE MACHINE

#### A separate class shall be used to hold all parameters required by the finite state machine.

### Finite State Machine

#### Parameter: Name

#### Parameter: Library Type

#### Parameter: Description

#### Parameter: Common State Method Signature (if used)

#### Parameter: Enable Common State Methods

### State

#### Parameter: Name

#### Parameter: Description

#### Parameter: Code

#### Parameter: Transition Entry Code

#### Parameter: Transition Exit Code

### Transition

#### Parameter: Name

#### Parameter: Description

#### Parameter: List of “From” States

#### Parameter: List of “To” States

### State Code

### Code Editing Dialog Box

#### Code editing will be done in two phases: 1) Simple text box editor, 2) Intelli-sense editor to provide auto-complete and object reference to objects appropriate to the state machine. This will require some research into things like Rosyln and Omnisharp.

## CODE GENERATION

### Format

#### “Template” files shall be made to simplify the code generation process.

#### Templates shall dictate the overall format of the generated code file with “tags” at strategic locations to allow an easy “search-and-replace” method of generating the file.

#### Auto-generated files shall include a message to let user know they are over-written.

#### Auto-generated files shall include a date/time on the generation.

#### A method of synchronization between the generated file and the source file shall be available to allow differences to be detected.

### FSM Library

#### The underlying state machine shall be independent of the library used to generate the code.

#### All library specific FSM code (Appcelerate, or whatever state machine library is used) shall be part of the “code-generating” class