State-Machine Configurator

Supplementary Specification

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 18.03.2020 | 1.0 | <details> | Sas Cosmin Andrei |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 4

3. Design Constraints 4

Supplementary Specification

# Introduction

[The introduction of the **Supplementary Specification** provides an overview of the entire document.

The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:

Legal and regulatory requirements, including application standards.

Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.

Other requirements such as operating systems and environments, compatibility requirements, and design constraints.]

The platform should not have any legal constraints, being a simple IDE.

The code generated is not supposed to be the most optimal one because that would require human logic (Maybe an AI could help with that). The reliability and usability is given by the nature of the task the platform will do.

It will on any OS that is compatible with Python.

# Non-functional Requirements

*[Define system quality attributes in terms of scenarios according to the following template:*

* *Quality attribute definition*
* *Source of stimulus: the entity (human or another system) that generated the stimulus or event*
* *Stimulus: a condition that determines a reaction of the system*
* *Environment: the current condition of the system when the stimulus arrives*
* *Artifact: is a component that reacts to the stimulus. It may be the whole system or some pieces of it*
* *Response: the activity determined by the arrival of the stimulus*
* *Response measure: the quantifiable indication of the response*
* *Tactics*

*]*

## Availability

I see the platform as a free Development tool.

## Performance

The performance of the platform should be good, the only thing that could drag it down would be the animation of the graphical interface but I think that could be solved using the graphic card.

Talking about the performance of the generated code, it will be a “case”-based C Code. It’s performance will not be the greatest.

## Security

-

## Testability

I plan on implementing a function that will allow the user to visualize the flow into the state-machine. That could be one option of testing it. The other option is just running the generated code.

## Usability

I could say this is the strongest feature of the project. It gives the possibility of developing C code to everyone without even knowing C.

# Design Constraints

The application will be made in Python 2.7.

I will use the following modules: - pygame ( for the graphical representation )

- tkinter ( for data related operations )

- [numpy ( for number processing )]