Kai Tindall

k.tindall-2018@hull.ac.uk

Abstract

This document will outline the timelines and procedures used in the project to try to ensure success.

Software Development Plan

Security in Internet of Things Devices

Table of Contents

[1.1 Revision History 2](#_Toc57125705)

[1.2 Document References 2](#_Toc57125706)

[1.3 Acronyms 2](#_Toc57125707)

[1.4 Glossary 2](#_Toc57125708)

[2 Software Development Activities 3](#_Toc57125709)

[2.1 Requirements Engineering 3](#_Toc57125710)

[2.2 Software Design and Implementation 3](#_Toc57125711)

[2.3 Software Verification and Validation 3](#_Toc57125712)

[2.4 Software Maintenance 3](#_Toc57125713)

[3 Software development tools 4](#_Toc57125714)

[3.1 Workstation 4](#_Toc57125715)

[3.2 Requirements management and documentation 4](#_Toc57125716)

[3.3 Software Design 4](#_Toc57125717)

[3.4 Coding and automated tests 4](#_Toc57125718)

[3.5 Configuration management 4](#_Toc57125719)

[3.6 Development rules and standards 4](#_Toc57125720)

[4 Reviews 4](#_Toc57125721)

[4.1 Minor Reviews 4](#_Toc57125722)

[4.2 Major Reviews 5](#_Toc57125723)

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Date** | **Author** | **Comment** |
| 1.0 | 24/11/2020 | Kai Tindall | Initial draft of the document |
| 1.1 | 25/11/2020 | Kai Tindall | Fixed missing references. |
|  |  |  |  |

## Document References

|  |  |  |  |
| --- | --- | --- | --- |
| **Identification Number** | **Description** | **Version** | **Reference Number** |
| SITD-0001 | The process initiation document (PID). Outlines the project in broad terms. | 1.1 | 1 |
|  |  |  |  |
|  |  |  |  |

## Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| PID | Process Initiation Document |
| UML | Unified Modelling Language |
| SRS | Software Requirements Specification |
| API | Application Programming Interface |
| JRE | Java Runtime Environment |
| VM | Virtual Machine |

## Glossary

|  |  |
| --- | --- |
| **Word** | **Definition** |
| Pseudocode | Code that is easy to understand but is not actually written in a real language. Normally written to convey the semantics of a design rather than any syntax. |
|  |  |
|  |  |

# Software Development Activities

The project will use a list of activities defined by (Elgabry, 2017).

## Requirements Engineering

This is the process of creating a list of statements that outline what is expected of the project. These include statements that relate to goals (primary, secondary, and “nice to haves”) and legislation that legally must be met.

There will be a “Software Requirements Specification Document” produced by the project that will outline all of these and will act as the main document that any testing will reference to be testing against. This document will be validated against section 2 of the PID [1].

## Software Design and Implementation

The design activity is where the project will be modelled in a combination of pseudocode and UML that describe the software that will fulfil the requirements specified in the SRS. There will be absolutely no implementation details within the design as it is not good practise. The design and implementation of the software should be separate. This activity will culminate in the creation of a software design document.

As a specific activity within the software design activity for this project will be the design of the API that will be used to expose the communications between the client and the command and control server.

Implementation will aim to take the design that has been produced and turn it into working C++ code. If the implementation is true to the design and the design fulfils the requirements, then by the time the design has been fully implemented the software should reflect the requirements. This activity will produce the C++ code.

## Software Verification and Validation

Verification is the process of checking that the software meets the specification laid out in the SRS. The project will aim to achieve this by using a combination of automated and manual testing. Automated unit and integration testing will be employed as a way of constantly checking code is working as it is intended to be; however, the main verification will be completed by running through manual system testing that will be defined in a software testing document.

Validation is the process of checking the project against the needs of the stakeholders and the goals of the project. This will be done continuously by analysing the codebase at each minor review for diversion from the project goals.

## Software Maintenance

Software maintenance is about the changing of requirements as well as fixing bugs. The former shall be handled with an agile project management style that will allow for requirements to change midway through the project without causing too many problems. Each major review will also assess if any requirements have changed and make the proper arrangements if it is so.

The process of raising and tracking bugs will take place through GitHub issues. This will allow for a central place where the bugs will be documented. Through using this the project members will be able to quickly see if there are any bugs that need to be dealt with.

# Software development tools

## Workstation

The workstation for the project will be a windows PC that has VirtualBox installed with a virtual machine using a Debian 10 (buster) image created.

## Requirements management and documentation

Requirements will be stored in a word document in a tabular form, split into sections that relate to different parts of the solution. Documentation will also be stored mainly in word files however other formats may be used if appropriate. All documentation will be stored in a documentation folder in the main Git repository.

## Software Design

The tool that will be used for software design is draw.io, a free online tool that specifies in drawing flowcharts, UML diagrams, and other standards of diagrams. This tool has been chosen because it is free, and due to it being online it can be accessed from both the host Windows machine and the development Debian VM.

## Coding and automated tests

Code will be written in the Debian VM as it resembles the target system so will make development smoother. Automated unit and integration testing will be completed by using the GoogleTest framework.

Code will be compiled using g++, the gnu C++ compiler.

## Configuration management

The project will use Git in combination with GitHub for configuration management. All team members can competently use Git and it is widely used in industry. GitHub also acts as a remote repository in the cloud so even if all of the development machines become corrupt the code will still be available in the cloud.

## Development rules and standards

All C++ code written will adhere to C++17 (ISO/IEC 14882:2017) as this is the latest stable C++ standard at the time of writing. This will be enforced by ensuring that the compiler is set to compile to the correct standard, in g++ this is done by adding “-std=c++17” as a flag.

# Reviews

The project will hold regular minor and major reviews. These will assess where the project is in relation to the Gantt says it should be. It will then state a reason for any difference and create some corrective actions to be made.

Templates will be created to speed up the review process.

Reviews will be stored in their corresponding folders in the Git repository.

## Minor Reviews

Minor reviews will take place weekly on a Friday and focus on project timescale and unit testing. The timescale review will be to compare the progress the project has made to the Gantt chart, and specify any corrective actions.

The minor review will also assess the testing effectiveness that has been conducted, taking a look at produced unit tests and making sure they’re working correctly. It will also conduct a code coverage report and ensure a result of at least 60% coverage.

## Major Reviews

Major reviews are to be conducted every four weeks. These reviews will act to assess timescale on a bigger scale and review the Gantt chart to see if a revised version should be created to work against.

The major review will also assess the direction the project is taking and whether the project is drifting out of scope or whether the scope of the project has or needs to change. This may be due to timescale issues or stakeholder’s values changing.

The major review will be conducted after each fourth minor review.

Elgabry, O., 2017. Software Engineering — Software Process Activities (Part 3) [WWW Document]. Medium. URL https://medium.com/omarelgabrys-blog/software-engineering-process-activities-part-3-ca1ef6818fd6 (accessed 11.23.20).