Sarah McLaughlin

Fall 2019 CS 4320 – Software Engineering

4 November 2019

Assignment 8

Individual Requirements Analysis for Semester Project

Software Requirements Specification (SRS)

Table of Contents

Introduction

The purpose of this Software Requirements Specification is to uncover the issues that surround open source projects. Most of the major concerns will be surrounding the issue of storage, real-time operating time and the safety of projects. With more contributors and repositories comes more risk. Risk for security and the ability to establish guidelines. These guidelines can include licensing, peer review, software sharing and security and operating systems.

Software Product Overview

Open sources must provide a specific type of software that makes code and repositories, specifically who created what, added what and when, available members or possibly only specific member(s). Members of repositories not only want their code source changes to be seen by fellow members in order to peer review and see the specific changes that were made to the code.

A commonly used software that is used for open source project is GNU/ Linux operating system. GNU is made up of “free” software programs that allows developers, and other users, to share their project and work with one another.

Similar to GNU, the Apache Software Foundation, also used in within Augur, is yet another software used for open source projects and is free and open-source software (FOSS).

Another software used by Augur is Comcast, which is also used for open source projects and includes helpful built in projects. In addition, Comcast also includes open-source standards groups including Cloud Foundry Foundation, Linux, Apache Foundation and many others.

Each software mentioned here provide specific licensing that is necessary for developers to collaborate and share their work with one another and to do so freely. In addition, many of these open source softwares mentioned also allow for a development of backend collaboration and data work.

System Use

The system usage for an open-source project would include being able to trace code, enhancements, changes and modifications made to the code or project itself. Any programmer or specific programmers, depending on the security of the project, may want to make changes or alterations to fellow members code in order to correctly implement, or to implement completely new code or applications to the project itself.

Actor Survey

For the first actor survey, the owner of the source code and open-source project itself would benefit from these open source projects. Rather than submitting and adding multiple code/software applications by any member, open source projects not only allows for peer review but also guidelines before making any changes. There may exist hundreds of branches or repositories that are gone through before the master, or default branch is ever affected by any code. These checks and reviews allows for multiple “drafts” or error checks before making and major changes.

In this actor survey, the members or developers themselves also come into play. As each one works individual on their own portion or sub-repository, their code is eventually compared and reviewed by other members. This can be instrumental and extremely helpful to work in teams and to be able to find solutions that may not have been found on their own. In addition, the communication that is included with many of the open source softwares mentioned allow for more eyes and more brains. If a developer were to run into any issues or need help, they could receive answers from fellow contributors.

Of course the final actor survey would include the customer. Being able to work both individually and as a group allows for more flexibility and agility, but most importantly for the customer, is the speed and cost-effectiveness. Without having to pay for and provide multiple software licenses, access fees and whatever else may be needed, open source platforms allow for each developer to gain access to whatever is used within their project. In addition, more eyes and brains allow for better talent overall and less error that the end user or customer may run into.

System Requirements and Use Cases

System Functional Specifications

The most important system requirements for any project, open-source or not, would be storage space. Working either locally or not, storage is of course important as you would not want to lose any work that has been done within the open-source project, or even your portion either.

Another important, but also related functional requirement would be the data and the storage of the data. It is important that the data of each member, group and repository is stored. This is important when returning and looking back at changes that have been made to the source code. Just as important, failure when committing or pushing work by members to a repository or important to keep track of so that lost work can be found again and retrieved. Within whatever database or data source that is used, it is important to also include the data types, data sources, and whatever else may be stored in the database.

In addition, the ability to test and accurately test the project is very important. You would want each member to have the shared capability to test, whether it be on a domain, interface, or software so that their changes can be visually seen.

Non-Functional Requirements

Important non-functional requirements for an open-source project would be to have a fluid and real-time operating system. For example, it would be terrible to make changes to a part of the project, only to find out it had already been altered.

In addition, another important non-functional requirement would be an ease of communication between collaborators with a similar goal. You would not want to work on a single open-source project with members that have different goals or do not understand the goal in the same way that another member might. Not only would you want fluid communication between the collaborators, but also a fluid communication and shared goal with the customer themselves. It would be terrible to have a customer change a portion of their goal or what they want to see and not be able to communicate that with the developers and do so clearly.

Design Constraints

A design constraint that I would think many developers would run into would be the type of design architecture or software that is used. Depending on the number of platforms that could be used, in order to follow strict guidelines of open-source projects, there would likely only be limited platforms that could and would be used. Using a limited number of platforms for these designs would also mean there would be a limited amount of design capabilities that could be used.

Another constraint would involve the agreement on the design itself or the design goals. Although a customer may give a general idea of what their looking for, it would be up to the group of developers in this case to make those ideas come to life. The agreement on what the design should be may be difficult to reach.

A major design constraint would be the performance of the design itself. Although it may be easy to code what you want your project to do, to incorporate the design with the functionality of the code is difficult as well.

Most major design platforms are also extremely expensive. This may also be a major constraint if the collaborators are working within certain cost constraints themselves. In addition to the fees that are used when designing, there may be additional fees that may need to be added, for example if they were to add more buttons or pages and the baseline sort of platform does not already provide that.

Finally, it would be extremely important that the design and UI that is tested on and created is dependable. It would be awful the platform may test or show one functionality, but it is not portrayed that way anywhere else. It would be important for the project to invest in a dependable design platform that can be trusted and reused and tested over and over again.

Purchased Components

Any fees or costs that may go into the collaborators hardware, software, architectural design/UI design, database and storage, or whatever else may need to go into the open-source project will not be cheap. Although, it may be funded by the customer, there is not unlimited funds to go around. With the sharing of more users and collaborators can include more fees depending on how many different hardware or software platforms, for example, need to given to each collaborator.

Interfaces

The interface of the project may be the most important aspect of the open-source project, since it is what most end users and customers will be seeing and using. It is important to have a clear module flow and communication between each page. Although it may be something multiple people have worked on and edited, it is also important that the interface does not seem to be created by different people. A sort of trend or theme in the interface is important, and just as important, is that it matches the ultimate goal or user criteria that needs to be met.

The interface, although what may include the least amount of work and coding, is what each end user will be viewing and it is very important that it is not only visually appealing but of course, also functional and reasonable for each user to use. It would not make sense to use bright neon colors for example for a business page, or to use images of blood and gore on a school website. These things are very important when taking into consideration the customer and the audience that will ultimately be viewing and using the interface.