Assignment 9: Individual Requirements Analysis

By Nick Allegretti

Project Name: Community Health Analytics Open Source Software

Contents-

1. Introduction

1.1 Purpose

This document is intended to briefly specify the requirements and architecture of the software, 'Community Health Analytics Open Source Software', or CHAOSS. This Software Requirements Specification (SRS) will specify an overview of the software product, as well as its system use, system requirements, design constraints, purchased components and a key interface for the software. While CHAOSS is an open source project that is always changing, this document will only represent the design and features of CHAOSS present on March 2nd, 2020.

1.2 Scope

CHAOSS is a Linux Foundation project focused on creating analytics and metrics to help define community health. The CHAOSS Project community is divided into two committees and four working groups. The Metrics Committee of CHAOSS works on establishing implementation-agnostic measuring community activity, contributions and heath. The other committee, the Software Committee produces integrated and open source software for analyzing software development of other open source projects. In addition, the Software Committee is also responsible for defining the standards and models used for other open source projects in specific use cases.

2. Software Product Overview

2.1 System Scope

This section provides the software scope of the CHAOSS project. CHAOSS is divided into three key pieces of software. Those being Augur, Cregit and GrimoireLab. The first, Augur is a Flask web application, Python library and REST server that provides metrics on open source software development project health and sustainability. The next, Cregit is a framework and set of tools that facilitates the analysis and visualization of the evolution of source code stored in git repositories. The final of the three, GrimoireLab is a free set of open source software tools used for gathering data from many development systems, such as git, GitHub, Jira and Bugzilla.

2.2 Feature Overview

This section provides a higher level overview of the three main software components of CHAOSS.

2.2.1 Augur

Enable Comparisons- Allows users to compare their project to other similar existing projects.

Time Metrics- Allows users to see the changes in a project over time and project trajectory using a “Point in time scores” system.

Downloadable Data- All data used for generating the visualizations in Augur can be downloaded as a .cvs or other select data types.

Downloadable Visualizations- All visualizations found in Augur can be exported and downloaded as an .svg file

2.2.2 Cregit

Mouse-Over Token- Stopping your mouse on top of a token will provide a summary of the information of the commit that added the token. This includes the commit ID, the git-author, the git-author-date and the summary log of the commit.

Left Click Token- Left-clicking on a token will open a new window with details of the commit in GitHub. This window will update with new details regarding the commit.

2.2.3 GrimoireLab

Automatic Data Gathering- Automatically gathers incremental data from almost any data source contributing to Open Source development (source code management, issue tracking systems, forums, etc.)

Automatic Data Enrichment- Automatically merges duplicated items and adds additional information about contributor’s affiliation, calculation delays and geographical data.

Data Visualization- Allowing filtering by time range, project, repository and contributor.

3. System Use

3.1 Actor Survey

This sections lists the two type of CHAOSS users, what they use the software for and the features of CHAOSS the user takes advantage of.

Community Manager- These are the users who use existing CHAOSS software to manage the health of open source projects they are working on or contributing to.

System Features:

- Collect and sort project repository information
- Display metrics

Project Front-End Developer- These are the users who will implement CHAOSS to develop their own frontend system to display a repository’s health.

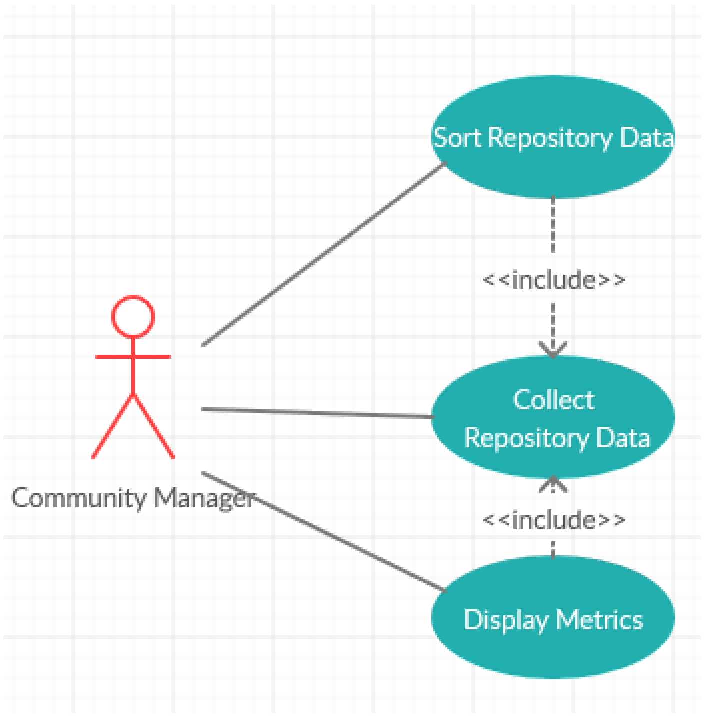
System Features:

- Expose metric data of repository group

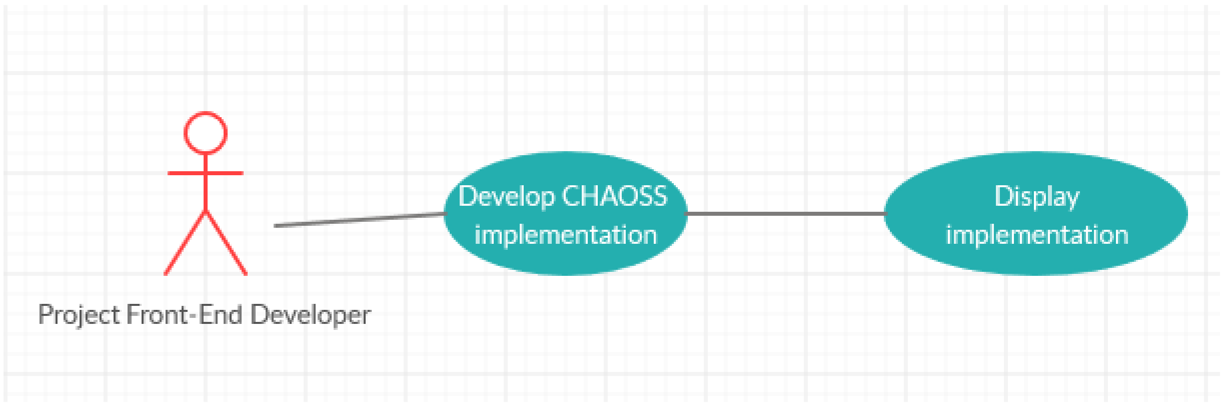
4. System Requirements

4.1 System Use-Cases

Use Case 1: Community Manager



Use Case 2: Project Front-End Developer



4.2 System Functional Specification

- A repository to assess its health
- Server or local machine where CHAOSS can be installed

- Access to git, GitHub or other similar software for team development

4.3 Non-Functional Requirements

- Multiple developers to assess contributions
- Project development history to view changes and health over time

5. Design Constraints

This section covers the constraints the CHAOSS software must fit within.

Constraints:

- CHAOSS should be able to access any software repository
- CHAOSS front-end should be able to run on any browser
- CHAOSS back-end should be able to run on any operating system
- CHAOSS should have separate permission levels for data access
- CHAOSS should be compatible with any major development software such as git, GitHub, Jira and Bugzilla.

6. Purchased Components

This sections lists the purchased components used in CHAOSS.

Components:

- Server Instance to host CHAOSS

7. Interfaces

This section lists the interfaces used in CHAOSS

Interfaces:

- Webpage Interface for frontend view