CS4320 Software Engineering

Group 6

Sprint 1

Augur Libyear Metric

Ben Hudson, Aidan Henry, Andrew Gillis

15th April, 2021

Our Server: http://20.69.127.173:5000/api/unstable/repos

Our Fork: https://github.com/ajhenry00/augur

1. Introduction

a. Brief Project Overview
   i. Choass/Augur is an open source project that measures the “healthiness” of Open source projects. Additional information can be found on the github page. The objective of the Augur Dependency Metric is to measure the age of open source repositories dependencies in “Libyears” and create an API endpoint in which another team can implement a UI. Libyears represent the summation of the years behind the current version for each dependency. This metric is inspired by and will be using the nasirhjafri/libyear repository.

b. The Prompt from Customer
   i. After discussing with Professor Goggins he suggested the task of creating a risk metric regarding dependency age would be a useful contribution to the augur project. Goggins then recommended to use the libyears repository as a baseline for this metric. He also said making a worker with libyear and creating an api endpoint and maybe a visualization would be sufficient for the scope of this project.

c. Team Roles
   i. Types
      1. Programmer
         a. Contributes code to the Product
      2. Documenter
         a. Documents code about the Product
      3. Organizer
         a. Organizing team meetings and holding people accountable for attending meetings and work performance.
      4. Communicator
         a. Communicates the question the group has to the professor/TA/helpful students and returns to the group with answers. This prevents flooding the chat accidently with parallel questions.
      5. Requirements Czar
         a. Knows the fine details of a Sprint of assignment as to verify the completion and correctness of assignments/sprints.
   ii. Assignments
      1. Andrew Gillis
         a. Programmer
         b. Documenter
         c. Organizer
      2. Aidan Henry

    a. Programmer
    b. Documenter
    c. Communicator
3. Ben Hudson
    a. Programmer
    b. Documenter
    c. Requirements Czar

2. Software product overview

a. The Augur Libyear Metric will be made by creating the Libyear_Worker which gets the dependencies manifest of any given github repository if permissions allow. Then will use the Libyear project to calculate libyears of the Open Source Project to be used as a metric. Finally an API endpoint will be created to be able to access this data easily by the group of people implementing the UI.

3. System Use, including an actor survey

a. System Use
    i. When Requested by the UI (Going to a specific repository statistics page) the Libyear_Worker will being calculating the libyear of the given repository ID on a different thread and callback when a Libyear result has been found or when it is unable to determine a viable libyear. The final return value is either a floating point number or an error.
b. Actors
    i. User (User Interface Handler)
    ii. Libyear_Worker
    iii. Libyear Library

4. System Requirements

System Functional Specification and Non-functional Requirements

L

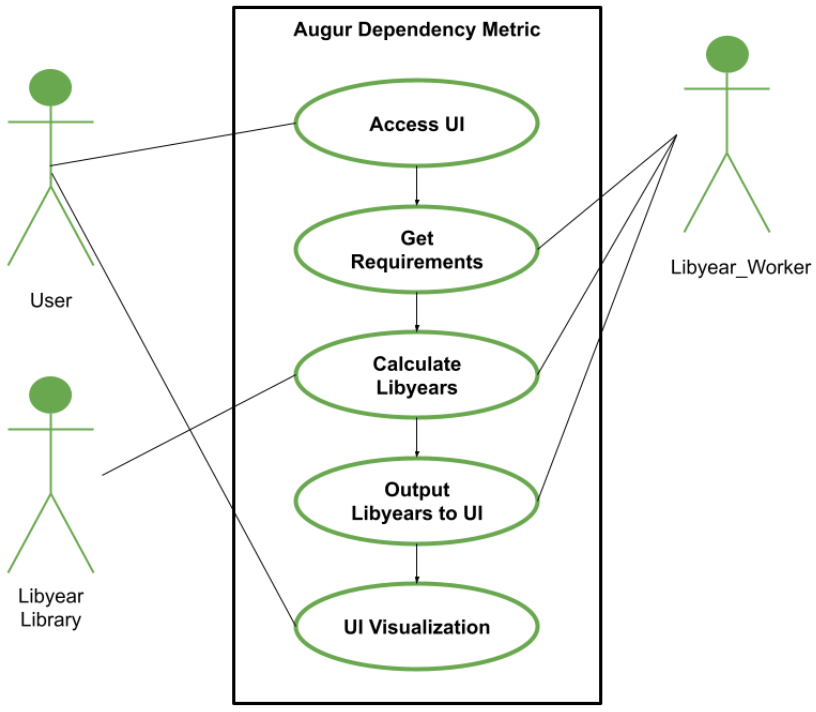
#	Users of Software System	Activity	Relevant Data	Constraints	Description	Priority 1-10 10 is critical 0 is Not Completed by this team
1	User (UI)	Show change in Libyears Overtime Visualization	Floating Point Libyear Value	None	Shows the Libyear of an OSP using a line graph. The x is time and y is libyears.	0
2	User (UI)	Show the current Libyear of OSP Visualization	Floating Point Libyear Value	None	Shows the current Libyear of an OSP	0

3	Libyear_Worker	Get Libyear	Github ID, Requirements manifest	Passes Pytest cases.	Gets the Libyear of a given github ID by finding its requirements manifest and calculating its libyears	10
4	Libyear_Worker	Get Libyears over time	Github ID, Requirements manifest, Historical Data of OSP	Passes Pytest Cases.	Gets the Libyear of a given github ID by finding its requirements manifest and calculating its libyears. Now calculate that for every version of the OSP.	5

One Use Case

- Title
  - Calculate Libyear
- Description
  - Given a requested github ID, the user will receive the calculated libyears based on the summation of versions behind each of the project's dependencies.
- Triggers
  - When the user visits an Open Sourced Project's stats page on Augur this triggers the Libyear calculation to occur so it can be displayed on the page as a metric.
- Actors
  - User (User Interface Handler)
  - Libyear_Worker
  - Libyear Library
- Preconditions
  - Searched Github repository still exists (Has been created and is not deleted!)
  - Github repository allows the dependency metrics to be viewed from outside sources.
  - Github repository has a requirements manifest.
- Main Success Scenario (Goals)
  - User received calculated output of libyears of a given github repository as a floating point number.

- Alternate Success Scenarios
    - None
- Failed End Condition
    - Github repository not found
    - Lack of permissions for viewing dependencies/does not have requirements manifest.
    - Versions from the respected dependencies are not in appropriate form leading to miscalculation of libyears.(“Not Detectable Somewhat”)
- Extensions
    - None
- Steps of Execution (Requirements)
    - User accesses UI of stats of Open Source Project
    - Libyear_Worker is assigned and starts task on different thread
    - Libyear_Worker gets Requirements Manifest of github project or throws a permissions error.
    - Libyear_Worker calls Libyear Library function to calculate Libyears
    - Libyear Library returns a floating point number of the Libyears or an Error if no requirements manifest/other errors.
    - Libyear_Worker now has the Libyear number or an error and calls the UI callback to send the libyear number for the respected Open Sourced Project.
    - UI Displays Libyears for a given Repository in its stats page or NA if an error occurred.
- Diagram
    -



- Dependent Use Cases
    - None

5. Design Constraints

a. Create Pytest Cases for the Activities
b. Can NOT massively alter the architecture of the Augur project
    i. Examples of massive alterations
        1. Changing Database tables
            a. Make new ones instead
        2. Changing Homekeeper and Its Workers code (Not including our worker)
            a. JUST DONT Touch them
        3. etc

6. Purchased Components

a. Azure to host the Augur's Frontend and Backend
    i. This was used to run the actual augur server. We were able to use our student status to get $100 dollars in credit.

7. Interfaces

The interface for this should be a augur api endpoint so that it can be integrated by some other team into the UI