Analysis of GitLab Project Using Boa

DESIGN DOCUMENT

sdmay20-43

Advisor:
Simanta Mitra

Client:
Simanta Mitra

Team Member – Roles:
Diego Realpe – Team Lead
Adrian Hamill – Test Engineer
Benjamin Carland – Test Engineer
Megan Miller – Test Engineer | Website Manager
Yi-Hsien Tan – Test Engineer | Website Manager | Organizer

Team Website:
http://sdmay20-43.sd.ece.iastate.edu

Revision	Date
0	9 – 25 - 2019
1	10 – 6 - 2019
2	12 - 6 - 2019
3	12 – 8 – 2019

Executive Summary

Development Standards & Practices Used

- IEEE Standards Association
- P7002 – Data Privacy Process
- P2675 – Standard for Building Reliable and Secure Systems Including Application Build, Package and Deployment
- P982.1 – Standard for Measures of the Software Aspects of Dependability

Summary of Requirements

- Integration of Boa with GitLab.
- Analysis of GitLab project with Boa.
- Generation of analysis report using R with data from GitLab.

Applicable Courses from Iowa State University Curriculum

- SE/ComS309 – Software Production
- SE/ComS319 – Construction of User Interface
- SE339 – Software Architecture and Design
- SE409/509 – Software Requirements
- ComS227 – Introduction to Object Oriented Programming
- ComS252 – Linux Essentials
- ComS362 – Object Oriented Analysis and Design
- ComS342 – Principles of Programming Language
- CprE430 – Introduction to Networks and Protocols

New Skills/Knowledge acquired that was not taught in courses

- Domain Specific Language
- Database queries
- Data Mining
- Data Analysis
- Server and Cluster Handling

Table of Contents

List of figures/tables/symbols/definitions

Index of Figures

Index of Tables

INDEX OF DEFINITIONS

1. Boa – A DSL developed by ISU to data mine software repositories.
2. Data Scrapper – One of the sub-team that specializes in scrapping data from GitLab repositories.
3. Boa Language Expert – One of the sub-team that specializes in understanding and creating Boa queries.
4. Boa expert – The experts who created/are professional in Boa.
5. DSL – Domain Specific Language
6. CLI – Command Line Interface
7. OS – Operating System
8. RID – Requirement ID
9. FR – Functional Requirement
10. NR – Non-functional Requirement
11. DID – Deliverable ID
12. TID – Task ID

1 Introduction

1.1 ACKNOWLEDGEMENT

Special thanks to the boa team experts for providing their professional help throughout our development, as well as our advisor Simanta Mitra for giving us valuable feedbacks, and providing us resources we need.

1.2 PROBLEM AND PROJECT STATEMENT

This project aims to integrate an existing Domain Specific Language (DSL) Boa, developed by a team of experts in Iowa State University, with GitLab. This project can help our client in evaluating the repositories of his classes on GitLab efficiently.

The project consists of three major tasks:

- Integration of Boa with GitLab.
- Analysis of GitLab projects with Boa.
- Generation of analysis report using R with data from GitLab.

The output of our design:

- Command Line Interface (CLI) program that does analysis on GitLab repositories.
- Output analytic reports with R programming language.

1.3 OPERATIONAL ENVIRONMENT

The program should run on both Windows and Linux OS command line. No support for mobile application.

1.4 Requirements

RID	Requirement Name	Description
FR1	Analyze Repositories	The program shall analyze repositories from GitLab.
FR2	Analytic Reports in R	The Program shall display analytic reports with R programming language.
FR3	Report Partitioning	The report shall contain two sections – commitment and code quality analysis.
FR4	Backend Automation	When the program runs, backend work shall be done automatically without manual connection.
NR1	CLI	The program shall be a CLI.
NR2	Ease of Learning	New users shall be able to learn the use all features of this program in less than 10 minutes.
NR3	Ease of Use	Analytic reports shall be displayed with just one click.
NR4	Security and Confidentiality	All information of repositories shall remain confidential.

Table 1 : Requirements

1.5 Intended Users and Uses

This program is intended for:

- SE/ComS309 evaluators/graders.
- Possibly any users who are interested in data mining software repositories, depending if our client wants to publish the program.

1.6 Assumptions and Limitations

Assumption:

- There are no multiple user running this program at the same time.
- The primary users will be our client/his classes' graders.

Limitation:

- There are some backend functions that must be done manually occasionally.
- Only professionals who understand the program can maintain the program.

1.7 Expected End Product and Deliverables

Expected End Product:

- Backend software for analyzing repositories on GitLab.
- A CLI program.

Expected End Deliverables:

- A functional program uses to analyze repositories on GitLab.
- Able to display analytic reports in R programming language.

DID	Date	Deliverable
D1	12/1/2019	Ideas for commitment and code quality analysis to create Boa queries.
D2	1/20/2020	Conclude what queries to develop.
D3	2/1/2020	Able to access GitLab data with the software created.
D4	2/15/2020	Show working queries (partially).
D5	3/1/2020	Run working queries on GitLab.
D6	4/1/2020	Prototype with R analysis

Table 2 : Deliverables

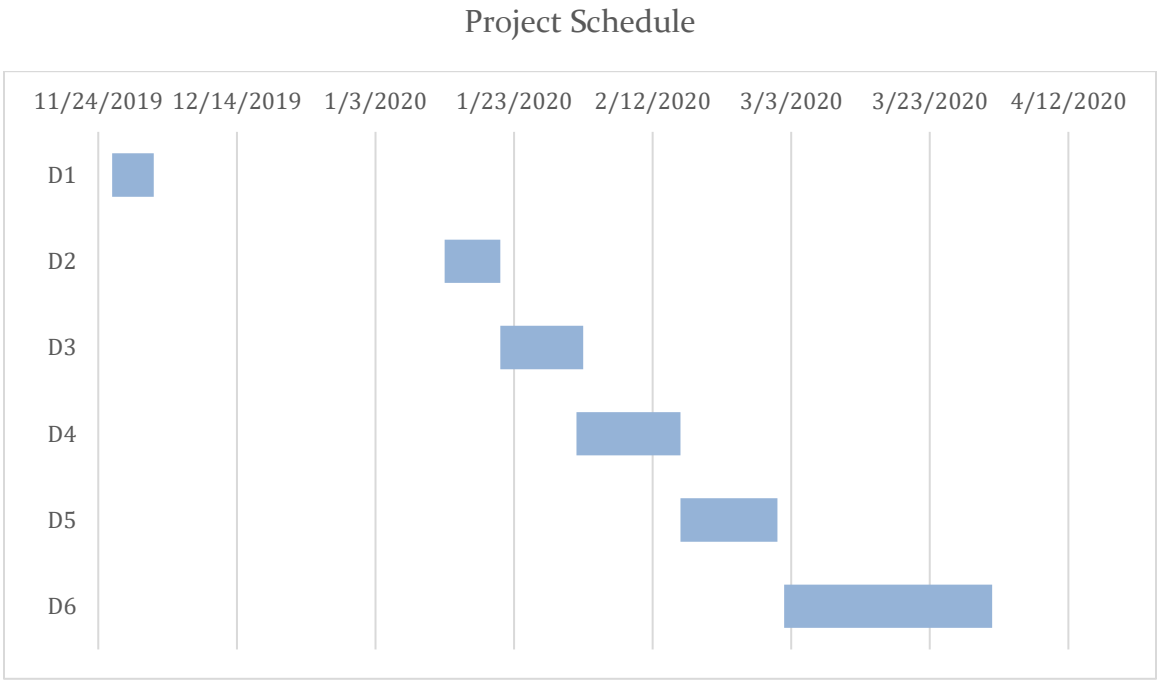

Figure 1 : Gantt Chart for Deliverables

2. Specifications and Analysis

2.1 Proposed Design

On the Boa language side of the problem we have been learning the syntax and semantics of the language. To do this we have been creating short presentations for our advisor to help us learn the language by teaching the language. This is helpful because it both teaches our advisor/client what types of things the language can do and encourages us to do more in-depth research about different things that can be done in Boa. This has involved the Boa team running lots of queries on the Boa repositories to learn about the language and hopefully give us a better idea of what types of queries would be helpful for our purposes.

On the Gitlab implementation side of the problem they have replicated the framework on a machine and are currently working on json scrapping. There is some difficulty with due to the lack of understanding of this from the current graduate students working on boa, so this is something the team has been working on for a week or so. This has been the primary implementation that we planned on, particularly because it's the simplest way. The json scrapping is important here because we want the data to hold any and all the GitLab data possible. This would allow the maximum manipulation of the repositories for our purposes.

In terms of IEEE standards quality of code is the primary concern here. We want to ensure that there are no unforeseen errors in the queries or in the implementation that would cause problems for students or teachers.

2.2 Design Analysis

The team set up a basic implementation of the Boa framework for GitLab, which was built referencing the current implementation of Boa for GitHub. This was a very effective strategy so far; it was well thought out and very realistic. As said in the previous section there has been issues with scrapping, but this is a very recent development and is being handled.

In the future the Boa Expert team will be required to make more direct contribution to the project when the focus switches from learning to creating. So far, the Boa Expert team hasn't had any significant issues.

2.3 Development Process

The team has been operating in Agile methodology. An iterative plan has been laid out where we meet with our client/advisor and get feedback on works accomplished weekly, as well as providing feedback within the team. The team then works on their weekly duties and meets back the next week unless there is reason to meet sooner.

2.4 Design Plan

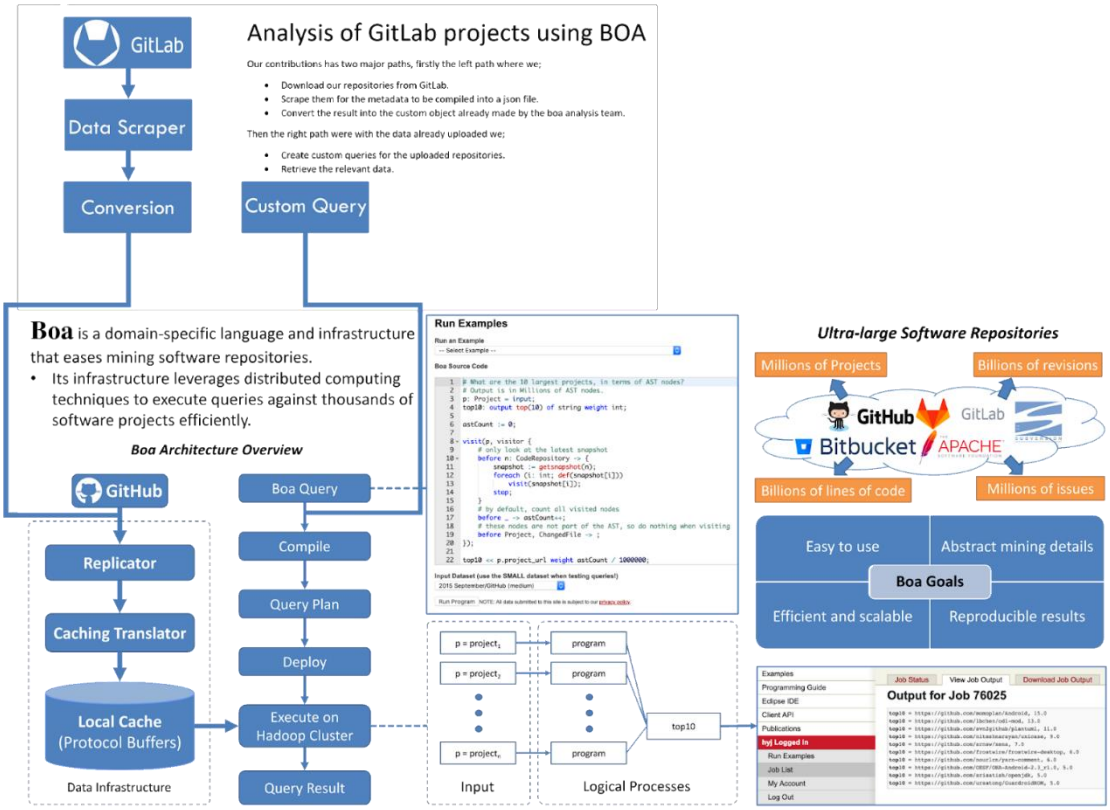

Figure 2 : Conceptual Visual Plan

3. Statement of Work

3.1 Previous Work And Literature

As previously mentioned, Boa is developed by a team of experts in Iowa State University. Being able to review their sample queries, data structure and such are definitely a big help in this project. So far, the program on the Boa website (boa.cs.iastate.edu) does most of the job we anticipate such as getting data from the software repositories, running the Boa queries on the targeted software repositories, displaying output data, and our main purpose is to make the Boa program work on GitLab.

A lot of our reference comes from the previous Boa program developed [1], and there are some advantages and shortcomings to it:

Advantages:

- A great reference as a high quality, well maintained DSL.
- We can personally meet with the creators and the experts themselves for consultation.

- The program is still active and being maintained.

Shortcomings:

- Since it's a DSL, there are no detail documentations about it.
- There are limitations when it comes to using program created by others.

3.2 Technology Considerations

Technology	Strength	Weakness
Boa DSL	Its highly readable, very efficient and specifically designed for data mining.	High learning curve, lots of limitation as there are no external libraries.
bash	Powerful language and native on most machine we use for our development.	Its hard to debug and the syntax is very particular.
Java	A comfortable language for all members, good for working locally with Boa DSL.	A lot more to code than languages like python/bash
R	Simple and powerful analysis generator. High manipulability.	Its designed for data set that's way larger than what we are working with.

Table 3 : Technologies

3.3 Task Decomposition

Our requirements and deliverables will be broken down into several tasks:

- Able to access metadata of GitLab repositories like GitHub provides.
    - Pull all GitLab repositories to a server.
    - Make copies of the repositories to GitHub.
    - Push the metadata from the server to each of the respective repository to GitHub.

  *This allow us to generate metadata for GitLab repositories with APIs from GitHub.
- Boa queries for analyzing software repositories.
    - Brainstorm ideas for boa queries.
    - Ideas should be separated into two parts: code quality and commitment.
    - Meet with our client to decide which ideas should be taken.
    - Develop those ideas into actual boa queries.
    - Test those queries on the pre-existing Boa program.
- R Data Analysis.
    - Run R analysis on results generated by the pre-existing Boa program.
    - Make sure the analysis report is styled correctly to fit our client's taste.
- Testing and prototyping our program.
    - Make sure metadata from GitHub can be read with our program.
    - Test queries on our own GitLab repositories data set.
    - Export results run by our program to R for generation of analysis report.

TID	Description	Dependent TID(s)
T1	Pull all GitLab repositories to a server	N/A
T2	Make copies of the repositories to GitHub	T1
T3	Push the metadata from the server to each of the respective repository on GitHub.	T2
T4	Brainstorming ideas for boa queries.	N/A
T5	Separating ideas into two parts.	T4
T6	Picking ideas for development.	T5
T7	Develop ideas into boa queries.	T6
T8	Test queries on pre-existing Boa program.	T7
T9	Run R analysis on results generated by T8.	T8
T10	Style analysis report.	T9
T11	Make sure metadata from GitHub can be read with our program.	T3
T12	Test queries on our own GitLab repositories data set.	T3, T11
T13	Export results run by our program to R for generation of analysis report.	T3, T9, T11, T12

Table 4 : Task Decomposition

3.4 Possible Risks And Risk Management

Risk Type	Possible Error	Mitigation Plan
Cost	N/A	N/A
Material	N/A	N/A
Equipment	Break down of MacBook used for data scrapping tasks.	Borrow/look for replacement for a new MacBook.
Knowledge Aspect	Since each of our member has different experience, depending on the technologies chosen for development, team experience will vary. High learning curve of Boa DSL might cause some hard stuck on our progress.	Constantly sharing resources to help each other on different aspect. Work closely with our advisor and the Boa experts and seek for help if things went wrong.

Table 5 : Risk Management

3.5 Project Proposed Milestones and Evaluation Criteria

Below shows our milestone planned for the second half of the year:

- Milestone 1: Make an actionable plan for gathering data
- Milestone 2: Create questions that can be answered by our data
- Milestone 3: Execute the plan for gathering metadata
- Milestone 4: Show working queries (partially).
- Milestone 5: Run working queries on GitLab.
- Milestone 6: Polish and create a finished product for the data gathering and query creation

Unit testing is unfeasible for backend pipeline. Data processing can be tested by setting up sample repositories with edge cases and small amounts of data. After, verify it metadata was integrally carried over.

The Boa program itself will probably also use manual testing; however, we will use the GitHub implementation of Boa for most of the testing of our queries.

Have blind tests with repositories with high- and low-quality code and determining if our analytics can distinguish them apart.

3.6 Project Tracking Procedures

The team will have a weekly meeting to keep track of each member's work and progress. We will also be using Slack and Google Planner to track overall progress of our project. With the assistant of these tools, we will ensure our completion of each milestone, and eventually the project itself by May 2020.

3.7 Expected Results and Validation

The desire outcome of our project would be a CLI program that can analyze all software repositories on an account (our client's) on GitHub/GitLab. The analysis result will then be translated into graphical analysis report with R programming language. User should be able to choose specific analysis (commitment/code quality, and which query to run) and get specific result. All the backend translation/analysis should be done automatically and hidden from the user.

When the prototype is present, the team will run multiple user acceptance test (mainly with our intended users and our client) before releasing to ensure our program quality and satisfaction.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Our project timeline will be shown on the Gantt Chart below according to tasks and deliverables.

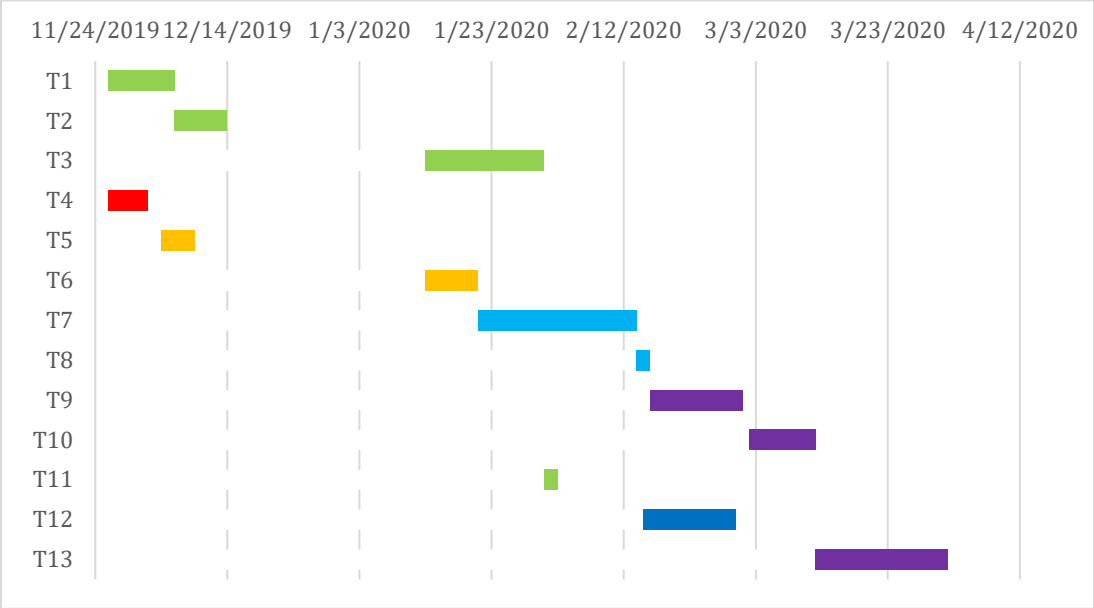

Figure 3 : Gantt Chart for Project Timeline

All tasks mentioned in part 3.3 are covered in the Gantt Chart above with duration (day) and are sorted by deliverables.

- Deliverable 1: Red
- Deliverable 2: Orange
- Deliverable 3: Green
- Deliverable 4: Light Blue
- Deliverable 5: Dark Blue
- Deliverable 6: Purple

Below includes a table to show each task's start date, end date and duration needed:

TID	Start Date	End Date	Duration (day)
T1	11/26/2019	12/5/2019	10
T2	12/6/2019	12/13/2019	8
T3	1/13/2020	1/31/2020	18
T4	11/26/2019	12/1/2019	6
T5	12/4/2019	12/8/2019	5
T6	1/13/2020	1/12/2020	8
T7	1/21/2020	2/13/2020	24
T8	2/14/2020	2/15/2020	2
T9	2/16/2020	3/1/2020	14
T10	3/2/2020	3/11/2020	10
T11	1/31/2020	2/1/2020	2
T12	2/15/2020	3/1/2020	14
T13	3/12/2020	4/1/2020	20

Table 6 : Task Duration Estimation

The team has divided all the tasks into 6 deliverables. We have accomplished deliverable 1 and planning to work on half of deliverable 2 before the winter break. The team agrees on this planning as most of them are achievable, and since our prototype is expected to be delivered on the 1st of April 2020, we believe there are plenty of time to spare if improvements/changes are needed.

On top of that, the team is prepared to adjust the schedule as needed if there are requirement changes from our client, or any unforeseen issue that will lead to an extension on our working duration. Since our team is divided into two sub-teams, we are able to work on two different aspect of the project at the same time. This allow us to have more room of flexibility when it comes to time constraint.

4.2 Feasibility Assessment

This is a very wide end project, as our client does not limit us on our creativity and imagination. The requirements of the project are clear, but we have plenty of freedom to design and enhance our program. At the very minimum, the final product will consist of a CLI frontend that takes command of queries, a backend analysis on software repositories of GitLab, and generation of analysis report in R.

The overall requirements of this program are very feasible and practical, as they are in the simplest form possible to get the work done. Since our client's basic requirement is to have the simplest working software, this allows future additional implementation depending on our schedule. The project timeline will help us keep track on our progress, and the team believes we will most likely be ahead of schedule.

Foreseen Challenges:

- Data Scrapping: At this point, the team decided that it is more feasible to do the 'migration' of GitLab to GitHub to obtain metadata of all the targeted software repositories. It is too time consuming and difficult to replicate GitHub's feature on GitLab.
- Boa DSL: Boa is a high learning curve DSL, even with good understanding on its syntax, it is challenging to translate our ideas into actual queries. Without detail documentations, the Boa Language Expert sub-team will have to meet up with the Boa experts to learn more about the language instead of self-studying.
- Maintenance: The client would like to use the program for a long period of time. As mentioned in 1.6 limitation, the backend part of the program is hard to maintain if the person has no knowledge about it.

4.3 Personnel Effort Requirements

Below shows our explanation of estimation of effort on each of the task. Estimated time will not be included in this table; please refer Table 4.1 for estimation of time for each task.

TID	Description	Estimation of Effort
T1	Pull all GitLab repositories to a server	This is the very first requirement to get our backend work running. This allow us to manually access the repositories without connecting to our client's class repository.
T2	Make copies of the repositories to GitHub	Since we decided to 'migrate' the repositories from GitLab to GitHub, this should only take time more than being difficult.
T3	Push the metadata from the server to each of the respective repository on GitHub.	Similar to T2, this will take much more time and creating a script to do so could prove to be challenging.
T4	Brainstorming ideas for boa queries.	This task will require us to come up with as many ideas as possible for our client to hand pick some of the best. Although its deliverable is on the 1st of December, this task should last for the entire development if we come up with better ideas.
T5	Separating ideas into two parts.	This should be one of the easiest tasks in our task list, it also helps us to determine if a query idea is good by evaluating if it is sortable into either commitment or code quality. It is not a good idea if it is too vague to be in either category.
T6	Picking ideas for development.	Handpicking ideas for development should not take up much time as well as we only need a handful of queries to test our program. Similar to T5, this task should last for the entire development in case there are better ideas to translate into queries.
T7	Develop ideas into boa queries.	This task is estimated to be the most challenging and time-consuming task as none of the team member is familiar with Boa DSL, and as mentioned, it has a high learning curve. This will be the task that takes up the most development time.
T8	Test queries on pre-existing Boa program.	Most part of the testing should also be done in T7 during development. This task is created for the final testing to make sure the queries we created work before delivering.
T9	Run R analysis on results generated by T8.	Since none of our team member has any experience with R, this could potentially be a time-consuming task. Our client ensured us it is a fairly easy language to pick up on, so we estimated about two weeks' time to learn and apply R on generating analysis report on our query results.

T10	Style analysis report.	Styling should be an important task as we want our client to be satisfied with the look of the analysis report for easier understanding and better grading.
T11	Make sure metadata from GitHub can be read with our program.	Similar to T8, this task is created for the final testing to make sure the metadata from GitHub (migrated from GitLab) can be read with our program before delivering.
T12	Test queries on our own GitLab repositories data set.	This testing task allow us to officially put the work from two sub-teams together and make sure there are no errors. About two weeks' time are spared in case of any unforeseen circumstance.
T13	Export results run by our program to R for generation of analysis report.	The last task will allow us to run end-to-end testing from running queries on our own GitLab repositories data set to generating analysis report with the results. We will also start prototyping our program.

Table 7 : Task Effort Estimation

4.4 Other Resource Requirements

Physical Resources:

- The only physical resource the team need is a working MacBook to develop the backend part of the program.

Software Resources:

- The pre-existing Boa program on the Boa website will be our main way to learn about Boa DSL before we are able to have our own data set. The Boa Language Expert sub-team will be using IDE such as Eclipse for some local testing as well.

Knowledge Resources:

- Internet sources and assistance from our client, as well as the Boa experts.

4.5 Financial Requirements

The team has no financial requirements to meet on this project.

5. Testing and Implementation

5.1 Interface Specifications

The software interfaces we will need to test our software are minimal. A series of a few GitLab repositories with dummy code of good and bad quality and various amounts of commits that we wish to set up so we can test the efficiency of the suite of queries we have created and determine of they were able to provide useful insights to each of them.

5.2 Hardware and software

- CS309 Old Repositories: These will be used to perform load testing on our system, filled with code from past 309 semesters
- Dummy Repositories: These will have mocked code with edge cases and various types of errors.
- Mac Mini: This computer will be our test server where the first prototype will be tested.
- Boa Website: Functional website from the Boa labs where isolated queries will be compiled and tested.

5.3 Functional Testing

- Unit: We are going to make each of the queries we develop to go through unit testing by running them in isolation on the Boa website which has an online compiler of the Boa language. This to test the correct functionality of each before integrating them in the query suite
- Integration: We are going to use some dummy repos developed to observe if the transition between the backend pipeline to the analysis piece of the system transitions the information accurately
- System: Our final tests for the system will be performed close to the end of Spring semester, using the repositories from CS309 of the current semester. If successful we should see analysis and insights that correspond to those student's final grades.

5.4 Non-Functional Testing

- Load: Using access to the large amounts of code found in the archives of CS309 we will perform a load test to determine that the final product is fully able to handle the volume of an average 309 class for our client.
- Compatibility: Unfortunately, the overarching project that we are using can only handle MacOS. However, we will test compatibilities on both the Client's machine and our Mac Mini (which have different versions of MacOS) to ensure that the transition between one and the other has no bugs or issues.
- Usability: At the end we will allow Dr. Mitra to have a period of test with the final product and address any usability features he would like changed or added.

5.5 Process

The process for testing each of our methods will follow a semi manual testing. Since the technology we are using is very niche, there are no real ways of mocking it. Our overall plan is to perform blind experiments with our data to test the reproducibility of our insights. By feeding the pipeline

unknown quality of code and obtaining our results, we can check the veracity of it by actually reading from the project afterwards and if the assessments of the system seem accurate

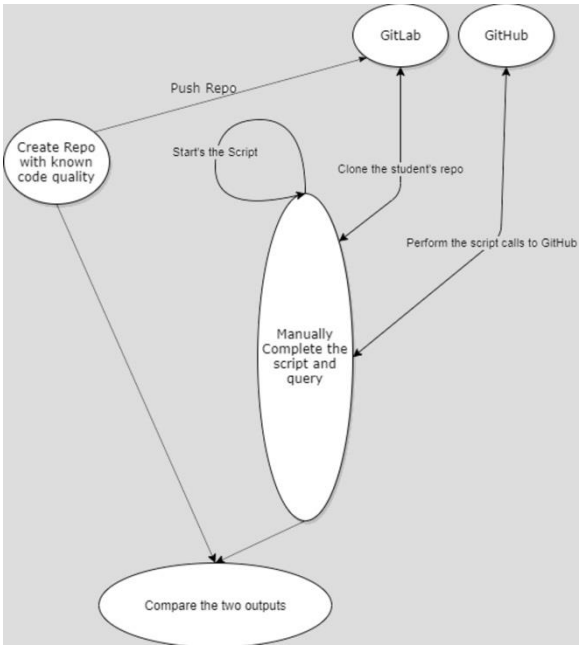

Figure 4 : System Testing Flow

5.6 RESULTS

Backend Results

The pipeline backend has yielded some results already regarding the cloning, scraping and automation of the data. Our first solution for this problem looked to pull directly from the GitLab repo using a government program designed for this. However, after conclusions that the program was not suitable and failed to provide the particular pieces of metadata we required we moved onto a different one. This solution allowed us to make clones of the repositories in GitHub with all data intact (which is one of our requirements) and from there we can use the Boa Labs code which does work on GitHub and allow the pipeline to query a GitHub API to construct the metadata JSON files necessary for Boa to execute against. A diagram of the breakdown of the system can be seen to the left where the process is detailed.

This system still needs to be tested against the edge cases of a repository, due to be completed in the next semester, but based on the trials and errors we underwent to bring a prototype of the scaping script and process we have gained tons of knowledge on how Boa operates and also has given us a time to partner with the experts that use Boa, allowing us to have a quick place where to go for extra resources or when we get stuck in a problem.

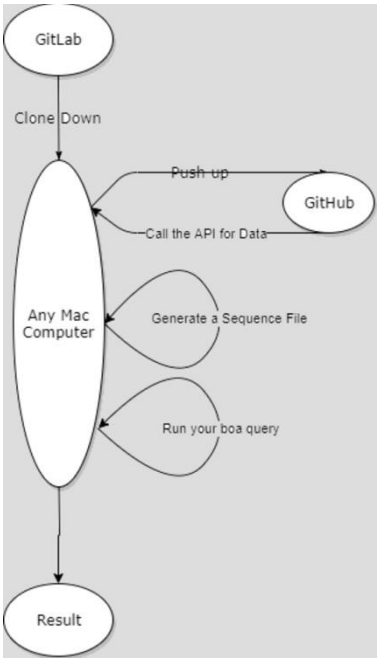

Figure 5 : Data Pipeline Process

Boa Analysis

The members on this side of the project have had a lot of great ideas and research into how to use some of the tools and functions that the Boa libraries offer to make a tool that talks about code quality on each of the repos. In the future as we go into the development and writing of these queries, the directions we take to assess the quality of a project will spread out and multiple aspects of the metrics we are given will be compared.

As mentioned before, Boa is a complex language to write high level analysis tools for and we expect a lot of trial and error in this section as lots of ideas will be tried and discarded while we push the limits of what Boa can and cannot do.

Final results on this area will be a suite of queries that can be toggled on or off that analyze different parameters of the code, we will also attempt to check the lines of code themselves too for flawed design patterns and hopefully catch system errors like stack overflowing or unused variables that could be slowing or hurting the performance of the analyzed project itself.

6. Closing Material

6.1 Conclusion

This semester our client challenged us to help him obtain the best analysis of his student's projects by using a very powerful but still obscure tool. Give him a product to improve his teaching. We have designed a plan that both incorporates his specifications and advice, the tool he requested and the knowledge of those who came before us.

So far in our project we have accomplished a solid understanding of our tools and have a clear view of how to approach development from here on out. This after many hours of research into them and practice. The back end, the data gathering of the project also has it's first working prototype after a couple of exploratory attempts that unfortunately didn't work. We decided to opt out of our previous solution aimed at working directly with GitLab, and instead take advantage of the already existing GitHub API to design the pipeline.

Our plan of action for the next phase of development is refining the data pipeline we have up to now. Seek to make it as automated as possible to aid our client, avoiding malfunctions of it for lack of knowledge of the system. Our Boa experts will meanwhile work on the analysis part of the project, drawing from their experiences and research from this semester, develop a suite of queries that tells important metrics and facts about the code it is given. These will need to be calibrated to properly tell the "health" of a project.

Lastly, we will perform tests with the sample code we have been provided from past semesters and when we are sure of the functionality and readiness of the program, we will bring our client Dr. Mitra to use the final version himself using his students from the current semester and get the last reviews regarding usability and load testing.

6.2 References

[1] H. Rajan, T. N. Nguyen, R. Dyer, H. A. Nguyen, *boa – Mining Ultra-Large-Scale Software Repositories*. Accessed on: Dec. 2, 2019. [Online]. Available: www.boa.cs.iastate.edu