Requirements Specification Document

6 December 2018

Team PathLab

Client
Viacheslav Fofanov
Mentor
Isaac Shaffer

Team
A. Turan Naimey
Alexandre Lacy
Chance Nelson
Austin Kelly

Version 1.2

Sponsor Signature ________________ Date ____________

Team Lead Signature ______________ Date ____________

Table of Contents

1. Introduction

In 2011, in Germany, an E. Coli outbreak caused widespread hysteria, and ultimately led to 51 deaths, hundreds of hospitalizations, and billions of dollars in damages from recalled goods. Scientists, using traditional genomic testing methods, determined that the outbreak was caused by Spanish cucumbers. In reality, however, the outbreak was caused by German sprouts. Despite the panic, the proper pathogen was able to be correctly identified and the case was subsequently closed.

The method that these biologists used to identify the E. Coli strain, both initially and subsequently, was Multiplexed Genomics Testing. This method of pathogen identification revolves around identifying a certain segment of suspected pathogen DNA, isolating it, and determining whether or not it really belongs to that pathogen. The most difficult part of this process is isolating the DNA from just the expected pathogen, so as to not be hindered by other DNA in a sample. In order to effectively do this, biologists have developed a method whereby they introduce a compound, called a primer, into their tests. These primers bind on both sides of the interesting section of DNA and allow it to be easily multiplied and subsequently isolated. Ultimately, Multiplexed Genomics Testing is the current method used by biologists worldwide to identify certain genetic segments.

Despite its popularity, this testing method still has several key drawbacks. As seen in the Germany example, these tests are not always accurate. A major part of this inaccuracy is that it is often difficult and time consuming to determine which primers to use in a particular test. Purchasing the incorrect primers might either cause huge delays in the testing process, or result in inaccurate findings. To counteract this, some companies have developed software that helps biologists choose primers more efficiently. However, these solutions are often exclusive, expensive, or generally inaccessible to most labs that are otherwise equipped to perform Multiplexed Genomics Tests.

Our sponsor for this project is the Fofanov Bioinformatics Lab at Northern Arizona University. Under the direction from Dr. Fofanov, the lab's principal investigator, Dr. Furstenau is in the process of creating a command-line program called Primacy, which is able to robustly compare all possible primer choices in a given scenario, inform its user what the best choices are, and let them know if something might go wrong. While this tool will be a boon for the greater scientific community, it still has a few minor issues, which our team, Team PathLab, has been commissioned to address.

This document will outline those issues and our team's plan to address them. It will methodically go through each issue our client wishes for us to address, the specific constraints that they have for our product, and all of the details of what exactly we will include in our finished version. Ultimately, this document will be signed by both our team leader and our client, and will serve as a contract for our implementation in the following semester.

2. Problem Statement

The Fofoanov Lab at NAU are in the process of developing a pipeline called Primacy which utilizes new computational methods to help researchers design better test panels. Primacy will be built as a command line tool with multiple modules. In order to understand the inefficiencies of this command line tool we need to learn how end-user researchers will use this tool to create/design test panels. Figure 1 shows how end-users will be utilizing the Primacy tool.

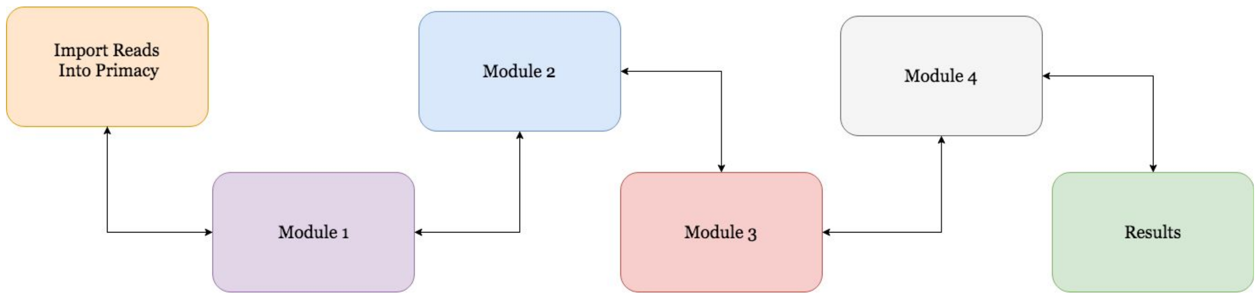


Figure 1: Typical workflow using Primacy

After wet lab processing and retrieval of DNA sequences, a user will input the reads into Primacy. Each module will allow the user to enter and tweak a specific set of inputs. Inputs gathered from each module will create a final result that will be used to design the test panels. The main problem with this command line tool is its accessibility to end-users of different technical expertise. Command line based tools are great for software engineers and tech-savvy users who are already familiar with the interface but most researchers who will be using this tool are experts in their field of work but they might not have the same level of expertise in using different computer interfaces. This is one of the main challenges our team needs to solve. New users often find operating a command line interface more difficult in comparison to traditional Graphical based interfaces, as command line based tools require a higher degree of memorization and familiarity for operation and navigation, and they are more prone to human error (e.g. user misspells a keyword). Error handling can be quite challenging from a developer's perspective when designing command line interface, as the user has a lot of freedom in typing a command.

When using Primacy, users might want to revisit some of the modules to change inputs to fine-tune the final result. This is problematic when using a command line based tool, because a user will need to remember exactly which commands were run with which parameters in place. Collectively, these small issues hold back Primacy from being truly usable by researchers and allowing them to focus on creating/designing test panels instead of worrying about learning new interfaces to operate a program.

In summary our client would like us to address these issues:

- Make the Primacy pipeline easily accessible to users
- Create an interface that is easy to use and intuitive
- Integrate tools to allow researchers to interactively tweak their results
- Allow for easy traversal of the pipeline

3. Solution Vision

Our proposed solution to this problem is to build a Graphical User Interface (GUI). This interface will solve our client's problem by being easy to learn, easy to use, and by providing accurate feedback and statistical visualizations. Some key features of our program will include:

- Able to fully and accurately utilize Primacy
- Easy to learn interface
- Easy to interpret results
- Configurable and consistent design
- User input checking

Our user interface will be able to take input from a user, or in some cases from a file, and check to ensure that those values are proper inputs for the field. Then, upon user request, it will take the data and package it into the format which Primacy requires. After that, it has Primacy run the data, and outputs the results in an easy-to-read manner on the next page. Ultimately, our product will add abstraction and functionality, while only adding marginal runtime to primacy.

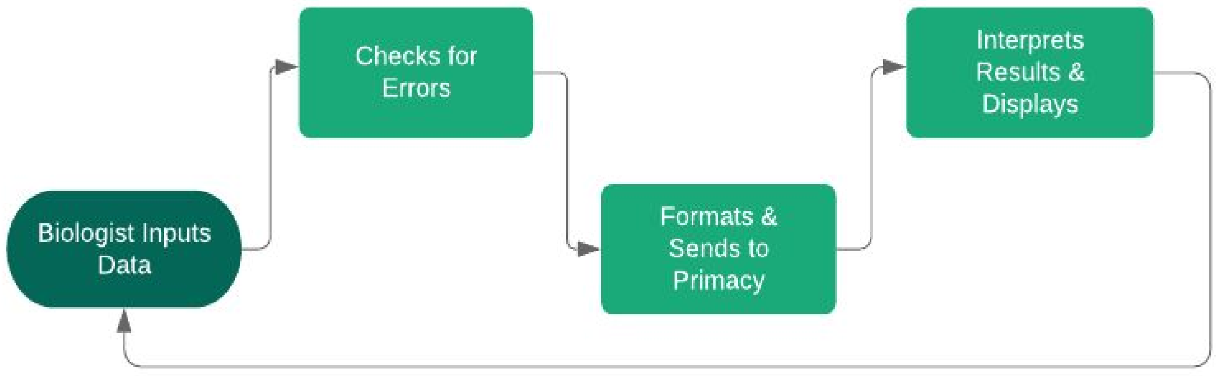


Figure 2: Planned GUI Workflow

Our job, as a team of computer scientists and software engineers, is to take the tool provided to us and morph it into a beautiful, yet simple, system. Currently, the tool exists solely as a Command Line Interface (CLI). Our solution will take this CLI and put an easy to use, user friendly Graphical User Interface on top of it, which will streamline key aspects of the identification process.

4. Project Requirements

After several meetings with our client and mentor, we have acquired the necessary requirements for our project, and we all feel confident that we can complete them. In order to assemble an effective solution for the problems outlined in the beginning of this document, a detailed list of requirements shall be created. This creates an effective 'jumping off' point to begin work on prototypes, and to ensure our team has a complete understanding of exactly what the problem is. As Team PathLab, we have followed the standard procedure for requirements acquisition, with a set of central 'domain' requirements, and many functional, performance, and environmental requirements that are offshoots of the domain requirements.

4.1. Domain Requirements

Domain level requirements outline our most important requirements in such away that both technically and non-technically skilled readers can understand the core of our program at a glance. As such, they not only play an important role in the readability of this document, but they also provide an abstracted scope that guides our more detailed requirements. In order to create this list of domain requirements, the group met several times with the clients. These meetings contained discussion on what the tool does, how it should do it, and many other constraints. The product of this process is as follows:

1. **Biologist Usability:** One of the main audiences that Primacy is built for is the biologist community. The major challenge Primacy has in reaching this group is that many biologists would find using a command line interface to be cumbersome, with other approaches being more efficient. To supplement this, our team's primary goal is to construct a GUI that is *usable* by biologists.

2. **Pipeline Traversal:** Primacy is built with a pipeline style for data flow. This means that the tool is subdivided into several steps, which all need to be completed in order. Rarely, however, are pipeline tools utilized once. On many occasions, it is required to go back one or several steps to change arguments, alter the data set, etc. Due to this, the tool must be capable of both going forward *and backward* through the pipeline.

3. **Maintainability/Expandability:** Due to the temporary nature of our team's involvement in the construction and maintenance of the project, the final product must have a readable codebase, and have the capability for future expansion.

4. **Input Validation:** Due to foundational concerns, Primacy does not have the ability to effectively check all input arguments for sanity. This can create certain problems with data inputs, such as corrupted FASTA databases, unrealistic temperature constraints, etc. One of our team's major requirements is validating this data *before* introduction to the pipeline.
5. **Cost Effective:** Primer identification is currently only involves the use of proprietary software or the purchase of expensive equipment. The primacy pipeline and our abstraction in our GUI package will allow users to accomplish the same tasks free of charge. This will drastically increase the size of the demographic for Primacy, and has the potential to spread around the world.

4.2. Functional Requirements

Functional requirements are the basic standards of which we base our development around. They are the elements that contribute to the bare functionality of our product, and reflect what our project will do. We have laid out that Primacy will contain the following elements:

1. **Tab Traversal**
    - Primacy will divide the steps of the processes into a series of tabs for each large step where calculations are made.

- A loading bar will be used when switch takes more than a few seconds.
- Within each tab, flexbox containers will spread out each section within the larger process, containing input fields, descriptions and images as needed.

2. **Interaction with CLI pipeline**
    - We will make an API to directly correlate functions in the pipeline with our Electron front-end environment.
    - States will be saved for each major step by managing JSON files at each point in the process.

3. **User Guidance**
    - The software will give feedback on the input provided and provide guidance for the user for sections that are difficult to understand. For example, the current tab will use highlighting to make it inherently clear to users that they are on that tab.

4. **Error Prevention**
    - Primacy will mark input as good, risky or invalid depending on the level of severity of the risk involved
    - Good will be marked in green and be completely acceptable
    - Risky will be marked in yellow and is considered bad practice, or potentially dangerous, but still will be accepted
    - Invalid will be marked in red and will not be allowed, as it would cause a serious issue.

4.3. Performance (non-functional) requirements

Non-functional requirements are requirements which form the basis for judging the performance of the team's final product, in practice. Therefore, it is just as crucial to possess accurate non-functional requirements as it is to have the same for functional requirements, because functional requirements to not necessarily reflect the *feel* of the final product. The following are the non-functional requirements that have been formulated, based off of meetings with the client:

1. **UI Quality:** As a crucial facet of a successful user interface is its aesthetic quality, the final product must reflect a professionally designed and implemented user interface as possible in the time allowed for implementation.

2. **Interface Usability:** If aesthetic quality represents 50% of a user interface's chance of success, usability is the other half. The tool must be specifically tuned

and designed in such a manner that the target audience must be capable of fully utilizing the tool within an hour. To assist the user in the endeavour, tooltips, notifications, and a wiki and install guide must be available for the user to read and research.

3. **Interface Performance:** Another peripheral facet of a quality user interface is its responsiveness to changes in inputs. Due to the tool's requirement to have adaptive validation for all user inputs, it is imperative that the tool can validate these inputs in as short of a timespan as possible. After discussions with the client, it has been decided that small inputs, such as range checking, temperatures, and others, should be almost instantaneous, while large inputs, particularly fasta databases, can take a maximum time frame of 10 minutes to validate.

4.4. Environmental Requirements

Environmental requirements deal with the domain requirements and their interactions with the systems the final product will run on. They are as follows:

1. **Cross-Platform:** In order to maintain usability, the final product must be runnable and produce the same outcomes on Mac and Windows systems, with an extended goal of being runnable on Linux.

2. **Packaging with the Pipeline:** Installation of the Primacy tool with the GUI must be easy, and be done in a single action. Therefore, the final product shall come prepackaged with Primacy.

3. **Pipeline Communication:** The client has outlined that the medium of communication with Primacy will be JSON string objects. The final product must be able to effectively parse and send information in this manner.

5. Potential Risks

While we believe the best decisions have been made for this project, there are still several potential risks that need to be addressed. Our technical requirements are a low risk, but domain and performance requirements present a medium risk.

5.1. Potential Inefficiency

- While we can say with certainty that the programming for this task can be completed, we can not guarantee outstanding efficiency. JavaScript is a slower language than some of its lower-level counterparts. We will be handling large JSON files with many calculations. The time between tabs for extremely large data sets will likely take several minutes or longer if our program is not optimized properly.
- If we run into a serious efficiency issue, we have decided to move any inefficient algorithms to the back end using C as a backup plan.
- Progress bars will also clearly indicate the sections that will load slowly, to help aid with transparency to the user.

5.2. Extensibility

- Another issue that could arise is from our extensibility requirement. While we plan to follow general conventions while programming, the possibility still exists that people wishing to extend our functionality in the future might encounter difficulty.

Overall, we have thought carefully about our design choices and we will be mindful of the potential risks going forward.

6. Project Plan

During the development our project, we wanted to make sure that we had a generalized plan for the future ahead to ensure all the requirements, requested features, and standards for this project are met. We started this semester by completing the team startup tasks to get our team up to speed with the project. Initial interviews conducted with our client helped us understand the scope of the project and requirements better. We started by building our team website, followed by a general discussion about requirements with our client, and team meetings to decide the right technologies. We then moved on to write our feasibility analysis document and also developed a simple demo to showcase our chosen technologies working in tandem to our client. We are currently in the process of working on our first technical prototype which will utilize ElectronJS, Flexbox and ChartJS.

Next semester we are planning to start with the API implementation and then move to work on backend operations including error checking and input validation. We plan to work simultaneously on backend operations and front-end features to efficiently use our time. Once we have most of our features completed we will move to basic testing and refinement of our software. This will allow us to detect and fix any bugs. During the final phase of our development we will be user testing to ensure all requirements have been met and users are able to use the software seamlessly. See our schedule attached in Appendix 1 for more timeline details.

7. Conclusion

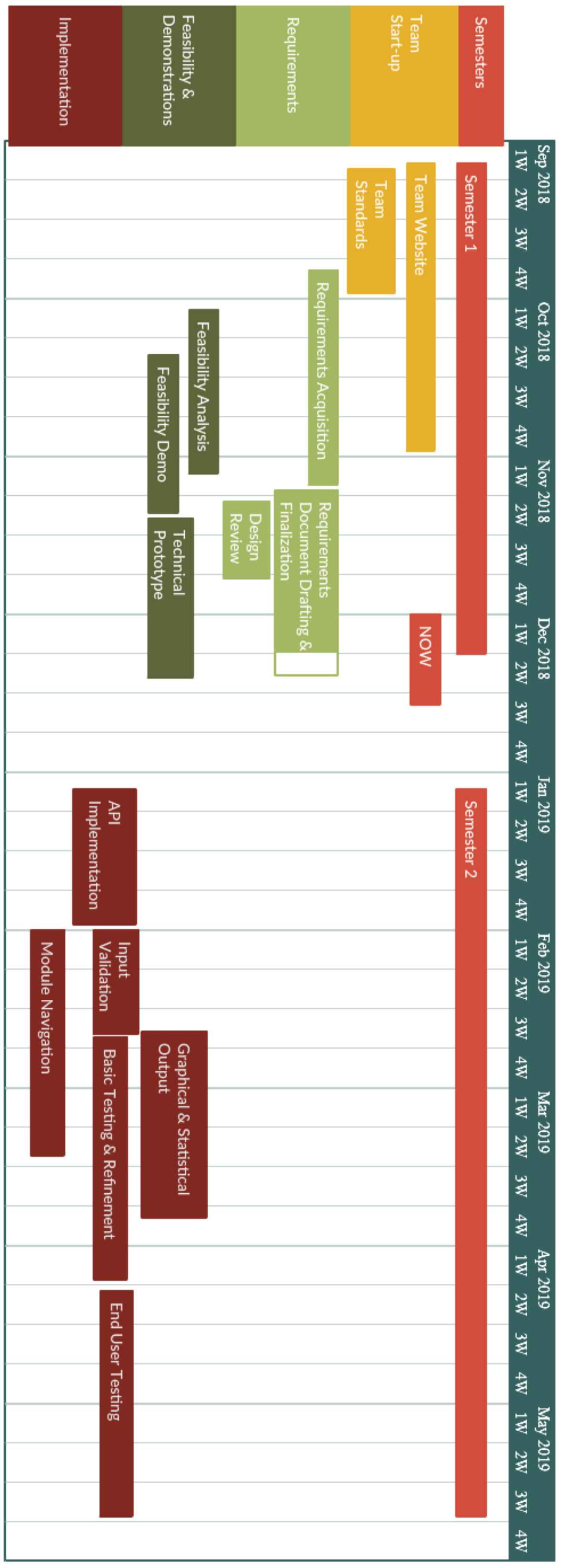
In this project our aim is to create a reliable, easy to use user interface. This user interface will abstract the command line program provided to use by the Fofanov Bioinformatics lab. Abstraction will allow for biologists who already know the command line to make fewer mistakes, as well as allow newer biologists to learn the tool more easily.

In order to make this user interface possible, we plan to use Electron, a JavaScript based platform. This platform is expected to meet all of our various levels of requirements, as outlined in detail in this document. Some levels of risk are involved by making this choice, such as processing time, but we have mitigating factors and contingency plans in place to counter these risks.

The goal of this document is to solidify the exact requirements that we foresee for this project, and unify them with the expectations of our client. As a requirements document, this paper will serve as a foundation and guideline for our prototype and full implementation of the software.

By creating a beautiful, easy to use product, we hope to innovate how the field of biology sees primer identification. By making Primacy accessible, we not only enhance the experience of those currently in the field, but we will be able to prevent the dissuasion of fresh talent by the high technical requirements posed by a CLI. With this project, we hope to remove the tedium that is a distraction to the field of comparative genomics.

8. Appendix



Appendix 1:
Gantt chart showing project development timeline