CS CAPSTONE DESIGN DOCUMENT
DECEMBER 1, 2018

STRATEGIC PLANNING IN EMERGENCY SERVICES

PREPARED FOR

LEVRUM

CARL NIEDNER

PREPARED BY

GROUP 66
CODE7

ETHAN AHUJA
LAUREN GASTINEAU
DAVID SAHNI
MAXIMILIAN VAN HOUT
TRENT VASQUEZ

Abstract

This document is intended to detail the Emergency Department Optimizer system prototype being developed for Levrum. It will describe the functions of high level structure as well as the components necessary to accomplish them. Additionally, this document will detail design decisions, and the reasoning behind them.

CONTENTS

0.1 Apportioning of work

This section details the work completed by each team member on this design document.

Ethan Ahuja contributed to:

- Helped with Introduction
- Glossary
- System Architecture Intro
- Decomposition Figures
- Image formatting

Lauren Gastineau contributed to:

- Making component design outline
- Introduction purpose, scope, intended audience, conformance
- Architecture design
- Design Rationale
- Design Stakeholders and Their Concerns
- Design Viewpoints Introduction
- Component Design Introduction, Predictor Component, Resiliency Control Component

David Sahni contributed to:

- Purpose
- System Overview
- Simulator Component
- Optimizer Component
- Recommendations and Metrics

Maximilian Van Hout contributed to:

- Outlining document
- Abstract and Introduction

- System Overview
- User Interface Overview
- Thorough rewording of entire document

Trent Vasquez contributed to:

- Viewpoints, Context, Interface, Dependency
- Mockups,
- Helped With Abstract/Introduction
- Component Design, Recommendation Matrix

1 INTRODUCTION

1.1 Scope

This document details the architecture and organization of the Emergency Department Optimizer prototype that is a precursor to software packages to be developed by Levrum. The intent is to enable Levrum to build their final product faster by producing a prototype.

1.2 Purpose

This document will overview the system and detail the stakeholders design concerns. To do this it will display diagrams that detail the overall architecture, then continue on to detail individual components and the interface the clients will use. It will additionally discuss the relevant viewpoints in each section.

1.3 Intended audience

This design document is meant to serve us as developers towards the production of a emergency department multi-objective search system for our client Levrum. This document operates as a guide for the remainder of the year to ensure that we make design decisions that achieve all components of our theoretical final product. This will also make certain our products design satisfies specifications of Levrum, our capstone instructors, and ourselves.

1.4 Conformance

As of November 30th, 2018 this document is in alignment with Levrum and the capstone instructors requirements and criteria for the design of this software. All communications and documents can be made available upon request.

1.5 Glossary

- **API:** - An Application Programming Interface is a ”set of commands, functions, protocols, and objects that programmers can use to create software or interact with an external system” [2].
- **AMGA:** - Archived-based Micro Genetic Algorithm. Adaptive Merging and Growing Algorithm. This enables multi-objective search. Unlike a normal genetic algorithm, this actually keeps multiple models. It also keeps a record of its best recommendations[6].
- **ED:** Also known as an ”ER” or ”Emergency Room” this is the Emergency Department of a hospital that provides immediate treatment for acute illnesses and trauma [4]. The software package we will be producing that helps to increase the performance of such departments.
- **GUI:** - A Graphical User Interface, sometimes referred to as a *gooey*, is the graphical interface that a user interacts with that improves the users experience over that of a command line [3].
- **Historic data:** The anonymous data provided by an ED that is used to predict trends that the ED will face in the future.
- **Metrics:** The output from the simulation that includes an analysis of how well the simulation performed with a specific recommendation. Metric data is used by the Optimizer as feedback on its performance.
- **Monte Carlo:** The Monte Carlo method is a technique for risk analysis in simulations that work ”by building models of possible results by substituting a range of values- a probability distribution- for any factor that has

inherent uncertainty" [1]. The simulator will receive numerous sets of data chosen by the Monte Carlo method at random that will be plotted as a distribution of expected values.

- **Discreet Event Simulation:** A discrete event simulation is a modeling technique where "patterns of events in the problem are recreated so that the timing and resource implications can be examined" [5]. Each event affects the current state of the simulation at a specific time.
- **Model:** A mathematical object that the optimizer uses to generate recommendations. This is equivalent to its 'understanding' the interaction of all of its variables.
- **Multi-objective search:** A kind of optimization problem that is trying to satisfy multiple different criteria. One example of this is balancing patient outcomes and cost.
- **Pareto Optimal:** - A pareto optimal solution is a solution where no alteration in the solution would increase the overall 'fitness' of a given solution. That is to say, you cannot increase the value one of the things you want without decreasing the value of a different thing you want. [7]
- **Optimizer:** The machine learning algorithm that influences the simulation. When analyzing the produced metric data from the previous iteration of the simulation, the Optimizer can change its recommendation to try to get better results.
- **Predictor:** The Predictor is a program that analyzes the historical data from an ER. The program looks for trends in the data that can be used to produce ranges for events.
- **Resiliency:** The degree to which a recommendation can withstand variation. Especially ability to withstand spikes in usage.
- **Recommendations:** Recommendations are the objects that detail how many people to staff, how many machines to use, ect. These are used in final reports produced to the user that contain probability analysis graphs that tell the user what to expect in the future and what the current best plans are for a given trade off.
- **Runtime:** - The time between starting and finishing the program. This is used to define performance requirements QR4, QR5, QR5 is measured in real time, the actual amount of seconds that pass. It not an analysis of the algorithms asymptotic runtime
- **Simulator:** The program that will carry out an accurate representation of an ED to be interacted with by the Optimizer. This will output the metric values. It will be a discrete event simulator.

1.6 References

1 Monte Carlo Simulation: What Is It and How Does It Work? - Palisade. (2018). What is Monte Carlo Simulation?. [online] Available at: http://www.palisade.com/risk/monte_carlo_simulation.asp [Accessed 28 Nov. 2018].

2 Techterms.com. (2018). API (Application Program Interface) Definition. [online] Available at: https://techterms.com/definition/api [Accessed 28 Nov. 2018].

3 Techterms.com. (2018). GUI (Graphical User Interface) Definition. [online] Available at https://techterms.com/definition/gui [Accessed 28 Nov. 2018].

4 Merriam-webster.com. (2018). Definition of EMERGENCY ROOM. [online] Available at: https://www.merriam-webster.com/dictionary/emergency%20room [Accessed 28 Nov. 2018].

5 IGI-Global. (2018). What is Discrete Event Simulation. [online] Available at: https://www.igi-global.com/dictionary/discrete-event-simulation/7878 [Accessed 29 Nov. 2018].

6 Tiwari, Santosh and Koch et al. AMGA: An archive-based genetic algorithm for multi-objective optimization. [journal] Available at: https://www.researchgate.net/publication/220740897_AMGA_An_archive-based_micro_genetic_algorithm_for_multi-objective_optimization [Accessed 29 Nov. 2018].

7 Britannica.com (2018). Pareto-Optimality. [online] Available at: https://www.britannica.com/topic/Pareto-optimality [Accessed 30 Nov. 2018]

2 System Overview

Our Emergency Department Optimizer is a prototype software suite to provide Emergency department decision support for hospital administrators. Hospital administrators have to make trade offs between the cost of additional employees, patient outcomes, and acceptable degrees of risk. This software suite enables this by generating a set of pareto optimal recommendations with good trade offs.

A user must input historical data and sets the constraints for their ED. In order to accomplish this, the user must enter restrictions, such as the maximum cost, the maximum number of pieces of medical equipment, and so on. The precise restrictions we won't know until Levrum assists us with a simulator. The user can achieve this by typing in numeric entry boxes. When the user is finished, they may click 'start' to sets the system into motion. Invisible to the user, variations on this data are used to build a set of best recommendations and a set of models used internally by the optimizer to make its recommendations. While these are being generated, the user is unable to interact with the program and must wait for it to finish. The user may, however, choose to pause the building with another simple GUI button and continue it later. When the simulations have completed, the user may view the resulting recommendations. The software provides this through graphs and tables that detail each recommendation and its trade offs.

3 System Architecture

This section describes the Emergency Department Optimizer software suite system on an abstract level as well as the integration between the major components. The Architectural Design description is accompanied with a diagram outlining the interactions between components at a high level as well as a description of their inputs and outputs. The Decomposition Description section displays the conceptual basis of the major components. The last section is Design Rationale where an explanation of why this solution and its implementation have been selected for the operations of our software suite.

3.1 Architectural Design

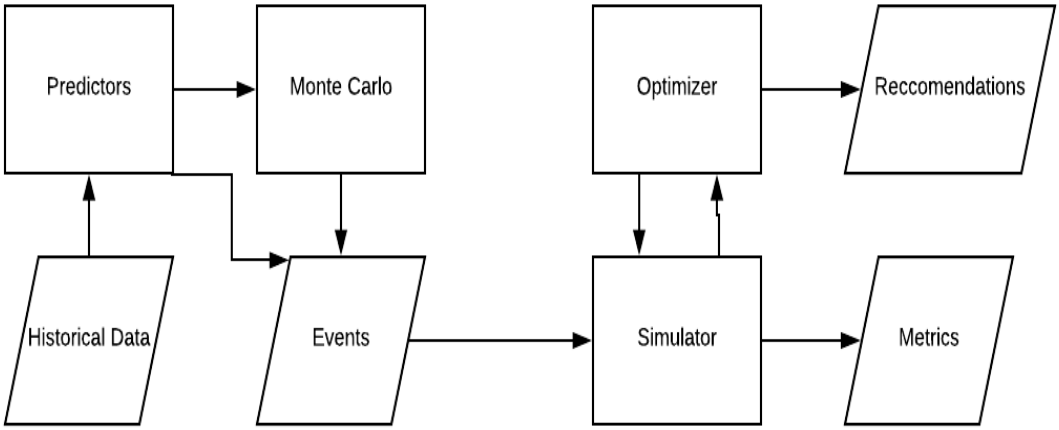

Fig. 1: System Components

Our software has four primary components: the predictor, monte carlo simulator, ED simulator, and optimizer. In addition, we will output ED recommendations and metrics. In the diagram above, items in rectangular boxes are components of our software that we will be developing. Items in parallelograms are data components that are either input by users or output by our software components.

Overall, our software has has very minimal user-interaction. The only required user-based input gets entered into the predictor and simulator. For the predictor, historical data will be entered in a format that is accepted by Levrum's predictor algorithm. For the simulator, constraint data will be entered in a GUI that we will create and is detailed below. The simulator is to be primarily put together by Levrum, but we do have to integrate it with a simulation object that we pass to its API.

Our software flows in the following manner: our predictor outputs events to the monte carlo simulator. This produces variations that enable testing optimizer solutions for resiliency. The simulator creates ED objects based off the events from the predictor and recommendations from the optimizer. Some of these events are variations of future data, others are merely predicted future data. These ED objects are simulated, eventually producing metrics. The optimizer uses

these metrics as feedback to improve its models and thus its recommendations. We also display this metric data to the user when they view the recommendations.

3.2 Decomposition Description

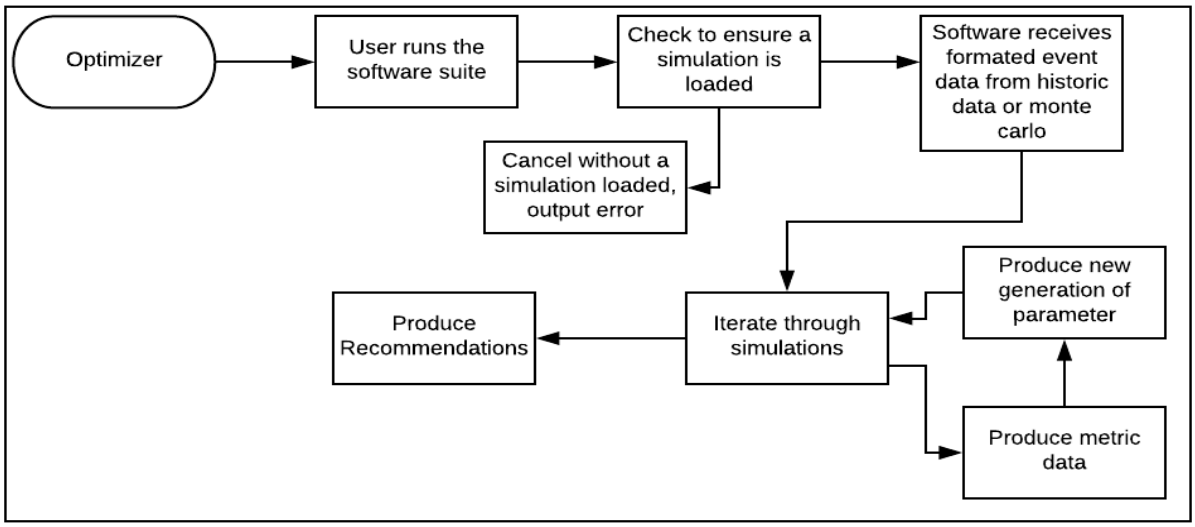

Fig. 2: Optimizer

The focus of our software suite is on the optimization of the users Emergency Department. In order to accomplish this there must be a form of event data that the optimizer makes changes to each generation.

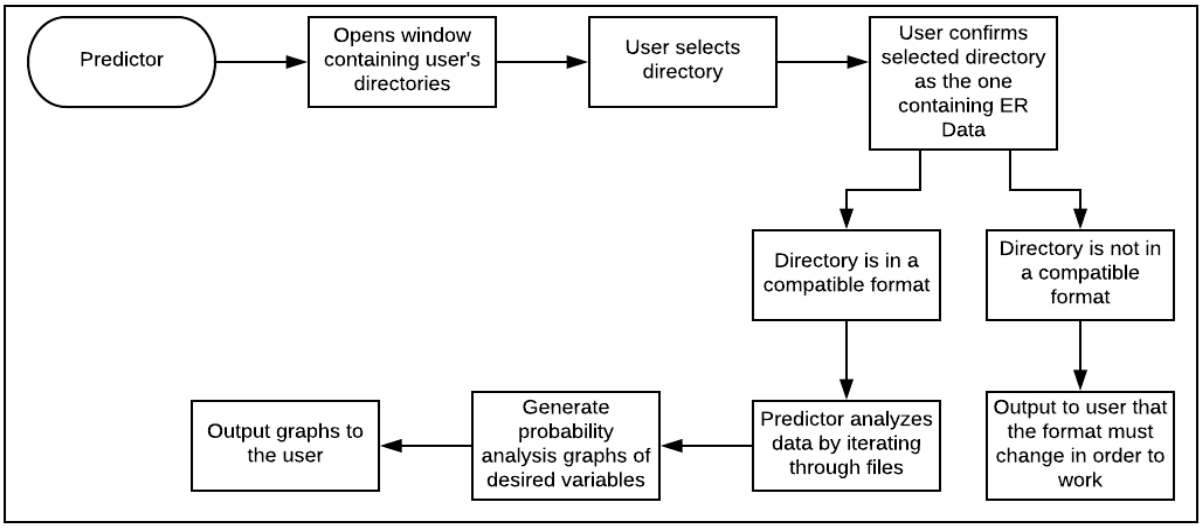

Fig. 3: Predictor

The software suite includes the functionality to interpret historic data and preform a probability analysis based on the trends found. This data is the formatted into events that the optimizer can use for simulations. The user must upload data in a format that the software can read otherwise the program will not accept it.

3.3 Design Rationale

Our software is intended to provide realistic and insightful recommendations enabling users to make better choices on ED arrangement and staffing by knowing both the effects of those choices, and what they sacrifice for choosing one thing versus another. We aim to create a seamless multi-component environment with minimal interaction between our software and the user. The user must only provide historical data and constraints concerning their specific ED in order to get the benefits of our software.

4 Emergency Department Optimizer in Perspective

4.1 Design Stakeholders and Their Concerns

For effective development, the system the needs to be verifiably useful as early as possible. The fundamental problem we expect to run into is not knowing if our system is actually viable and corresponds to reality. To that end each of the systems components have clear inputs and outputs, and can be individually verified as correct. This system needs to be very thoroughly tested before being handed off to our client. Such testing also enables us to iterate on our system to locate and fix problems early.

For Levrum, our systems components are substitutable, highly specific in use, and thoroughly commented in order to more readily build upon our prototype. Our software will be developed in a programming language that is understood by both our client and our development team. This is because Levrum intends to continue this project as an addition to their software product line.

Our software will be easy to use and our recommendations will be organized and clear for our users. This is so that our users can effectively use our software and understand our metrics.

4.2 Design Viewpoints

This section details design viewpoints for our multi-objective search optimizer software. The relevant viewpoints are context, interface, dependency, algorithm, structure, and interaction viewpoints.

4.2.1 Context Viewpoint

It should be remembered that the the Emergency Department Optimizer is a proof of concept and not a finished product. This prototype status means that while the intended users of the final version of the software are members of hospital staff and administration, this prototype is fundamentally being built to ease Levrums development process in making that final version. Due to an intended audience with limited programming knowledge, the software should have a very simple exterior.

4.2.2 Interface Viewpoint

When a person intends to use the software, input of emergency department data should be easy and user friendly. While more technical knowledge would help the user, the software should come with a default tuning that encompasses what the average emergency department would need. As well as the default option, users should have the ability to modify values and weights inside of the predictor/simulator/recommendation to overall modify the output of the system. In addition, the output should be presented in a well formatted way for hospital staff to understand.

4.2.3 Dependency Viewpoint

The Emergency Department Optimizer will take preexisting hospital data and transform it through the stream of our components to the final recommendations output. Historical hospital data will be piped into the predictor/monte carlo simulator where it will output a set of data for the simulator. The simulator will take the variable data presented and output the results. With these results, the optimizer will attempt to to make a new set of variables, then pass them to the simulator again. After a set amount of attempts, the data will be passed to a recommendation engine for output to the user.

4.2.4 Algorithm Viewpoint

The algorithmic viewpoint is mainly applicable in the context of the optimizer, where we will implement a multi-objective search algorithm to optimize simulation output. The objectives that the algorithm will search for will depend on domain research that we will undergo with Levrum, however some preliminary ideas include: time before seeing a doctor (by acuity level), the average bed occupancy, effective staffing levels, and average patient length of stay. There are many possible objectives here, and further research and testing will be required to create a final set. The search algorithm will use an implementation of the archive based Micro Genetic Algorithm (AMGA), which is a genetic algorithm that uses a small sample size in combination with a historical archive of previous solutions to provide a solution set. AMGA uses a small population set on each generation, and as a result requires less time performing simulation than other options. The algorithm performs analysis on the archive of previous solutions to create a new population, and this is advantageous as the simulator is the main bottleneck for the system. Levrum uses an implementation of this algorithm, and will assist us in both implementing it and testing our implementation. The future event data predictor will use simple univariate time series prediction or a more complex neural network approach. This decision will be dependent on time and we will work closely with Levrum to determine which approach we will take.

The psuedocode for AMGA is as follows [6]

```
Begin
Generate Initial Population
Evaluate Initial Population
Update the archive (using the initial population)
repeat:
    Create the parent population from the archive
    Create the mating pool from the parent population
    Create off-spring population from the mating pool
    Evaluate the off-spring population
    Update the archive (using the off-spring population)
until(termination)
Report desired number of solutions from archive
End
```

5 Component Design

This section describes the various pieces that compose our multi-objective search algorithm. For each component, the dependency element, algorithm element, and design rationale will be discussed.

5.1 Predictor

5.1.1 Dependency Element

The predictor is the beginning component of our software. The predictor takes in historical ED data that will be input in a form decided upon by Levrum. Historical data includes ED arrival time, patient treatment time, acuity level, and staffing amounts. The predictor takes ED data and makes predictions about future these future ED variables. The predictor connects to the monte carlo simulator.

5.1.2 Algorithm Element

This is the machine learning algorithm that generates predicted ED data from historical ED data. The algorithm will identify trends in historical data in order to predict future ED data. The algorithm we will utilize for this component will be an in-house prediction algorithm developed and provided by our client, Levrum.

5.1.3 Design Rationale

From a developer viewpoint, utilizing Levrums prediction algorithm that was developed for general future prediction allows this component to be more quickly developed.

5.2 Simulator

5.2.1 Dependency Element

The simulator will be dependent on the ED event data that is generated by the predictive and resiliency pieces. This data will be in the form of events that specify ED arrivals and acuity levels. The specific data format will be decided by Levrum depending on simulator constraints. The data format that the simulator uses will be the prevailing format for all ED event data passed into the predictive or resiliency portions as the simulator is the most restricted component. Hospital administrators and the target users must also input a model of their hospital, with information regarding total beds, staffing availability, and department specific constraints. This object model of our simulation will be built in coordination with Levrum and Emergency Department physicians.

5.2.2 GUI Element

In order to maintain a simple GUI there will be no visual representation of the simulation. Instead the user will be shown a notification declaring that the simulation is running, as well as a current run time and an estimated time of completion. The users will interface with a simple data entry GUI to create an object model of their emergency department to be used in the simulator. This GUI should be no more complex than an HTML form or something similar using a C++/Python GUI library with minimal complexity.

5.2.3 Design Rationale

The simulator is an instrumental part of the software suite. Without the simulator the software could not function therefore it is vital to keep the simulator simple and functional. Removing the visual elements of the simulation will help to minimize runtime without compromising user benefit.

5.3 Resiliency Control

5.3.1 Dependency Element

The resiliency control component of our software requires predictive ED data, output from the predictor, in order to stress test the forecasted data. This component will simulate and test the predictive data with a range of possible outcomes, most significantly extreme values. As a result, resilient predictive ED data will be output to the simulator.

5.3.2 Algorithm Element

This component will utilize a monte carlo algorithm. This is a resource-conscious algorithm that will return probability-based data. Using it, we can ensure that our predicted ED data is realistic and resilient to outliers.

5.3.3 Design Rationale

Monte carlo algorithms are ideal for us as developers because it is a resource-restricted algorithm. The results have a margin of error to account for, but the system will not use more than the anticipated amount of time or resources.

5.4 Optimizer

5.4.1 Dependency Element

The optimizer is dependent on the simulator output and model configuration. It must be able to evaluate the simulation result to assign a fitness to each model for the genetic algorithm. Once assigning fitness values, the algorithm will need to reconfigure the simulation configuration to test a new solution. This will involve editing the configuration file that the simulator takes in. In addition, the optimizer will execute many simulations to test many solutions, and therefore is dependent on its ability to start the simulator as well.

5.4.2 Algorithm Element

This is the multi-objective genetic search algorithm that makes changes to the simulation based on the metric data produced from the previous generations of simulation. The algorithm we will utilize for this component is AMGA. This algorithm uses an archive of previous solutions to compare against current solutions for the creation of the next generation, and as a result requires less simulation iterations. This algorithm has been used by Levrum for other multi-objective search problems, and will be provided by Levrum.

5.4.3 Design Rationale

From a developer viewpoint, AMGA is ideal for our optimizer because of its use of an archive of previous solutions. While this requires more memory to be available, it reduces the number of simulations that need to be run. Since the simulator is the running time bottleneck of the system, this property of AMGA is quite desirable. AMGA also emphasizes non-dominating procedures when updating its archive to maintain diversity, which is advantageous in our large search space.

5.5 Recommendations/Metrics

5.5.1 Dependency Element

The recommendations that are output to the user at the end of optimization are dependent on the optimizer and simulation. The optimizer will have a set of best solutions that the metrics component will display to the user. AMGA also maintains a set of useful statistics such as: diversity, hypervolume, and delineation [6]. These statistics will likely not be useful to end users, however they will be crucial in testing the algorithms implementation.

5.5.2 GUI Element

While the software is intended as a proof of concept, we are still planning a basic output gui for the user. This will include a basic set of graphs and tables helping demonstrate the metrics defined by the optimizer. In addition, the user will have basic functionality to modify how this data is presented.

5.5.3 Design Rationale

With the output from the optimizer, the goal is to use python libraries for developing visuals of the data. The rationale for python is that the language and accompanying libraries provides simplistic and easy to use tools for building basic GUIs.

6 User Interface Design

6.1 Overview of User Interface

The user interface only needs to enable the user to do three things: input historical data, detail constraints, and view ED recommendations. Everything else is in the 'backend' and so invisible to the user. Viewing recommendations includes past simulations so that the user does not have to run the simulation all over to review their recommendations if they should close the program.

Viewing of recommendations is the most visually intensive part of this application. After the user has run a simulation, they can review the recommendations via our recommendations GUI. The user can from here can view recommendations that maximize patient outcomes, minimize cost, etc. The trade offs between each of the other values are clearly visible from the metrics GUI in order for the user to make informed choices.

6.2 Screen Images

6.2.1 Simulator Configuration Setup

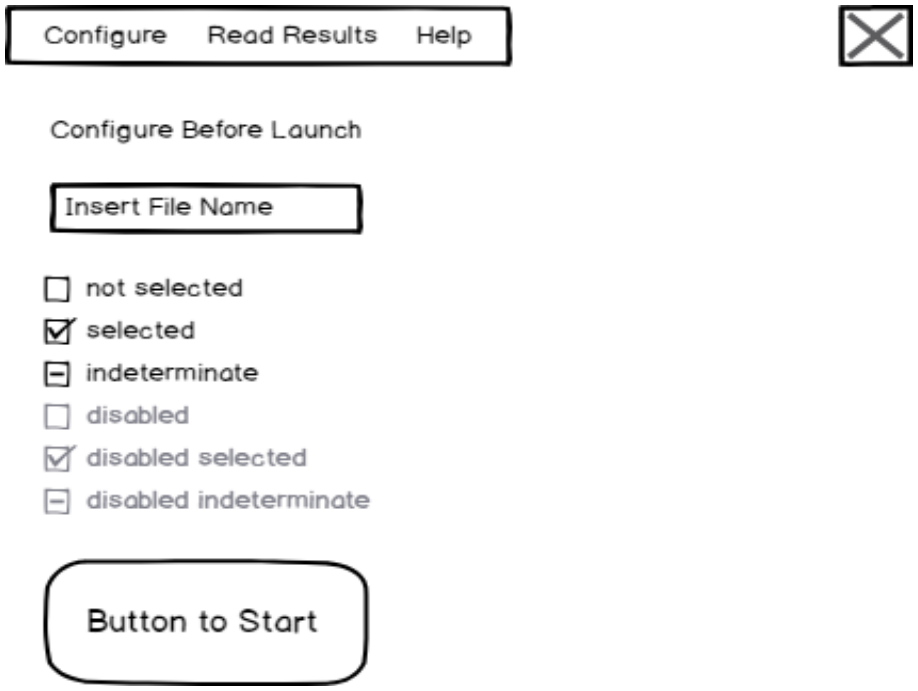

Fig. 4: Configuration GUI

6.2.2 Recommendations/Metrics GUI

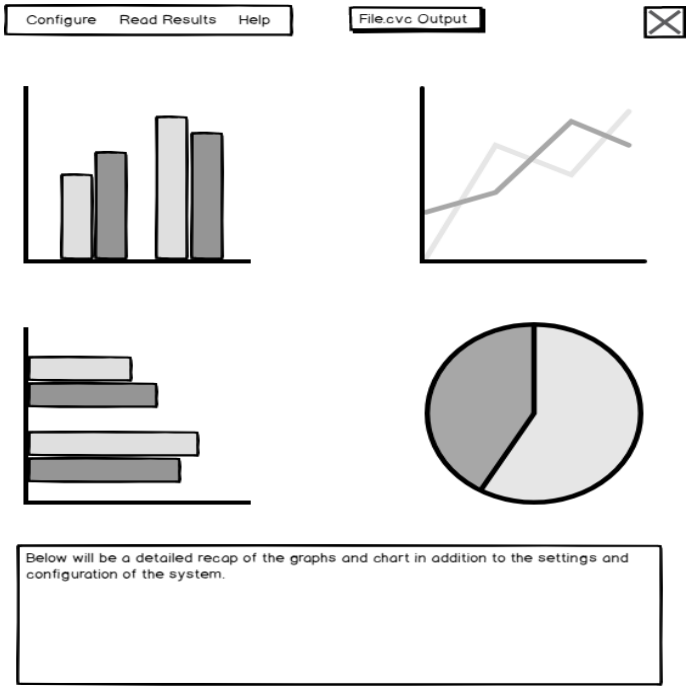

Fig. 5: Recommendation GUI