Prototyping Walking Robots Using Genetic Algorithms

Kevin Beck

Ryan Romanosky

Kristy Schworn

Joseph Borsodi

California University of Pennsylvania

Instructor Comments/Evaluation

Table of Contents

Abstract

This paper explores the development requirements for creating walking robots using Genetic Algorithms. These algorithms provide unintuitive solutions to problems, and by a system of iteration and refinement, find the most suitable solution to a problem. By testing the designs in a physics simulation, the developed software will systematically revise and reiterate on various parameters in order to find the specifications best suited for the given environment. Utilizing a genetic algorithm allows this system to prototype base level designs of walking robots in any provided environment. The listed requirements in this document outline what is expected of the product as well as what organizational structure and cooperation will be used throughout its development.

Keywords: Genetic Algorithm, Walking Robot, Environment

Introduction

Background

Evolution is the defining force behind innovation in the natural world. All forms of life develop and refine features through evolutionary processes. Through specifically defined mathematical equations this process is extended into computer-driven applications. Putting these methods to use enables a designer to rapidly generate non-trivial solutions to large and complex problem sets. This paper serves as a guideline for how this method of evolutionary design will be applied to the area of walking robots in different and varied environments. Machines such as bomb removal robots or lunar rovers are deployed in areas where the environment is either inhospitable or dangerous. Also, automation is being implemented in areas of manufacturing, previously thought to be uniquely human areas of expertise. This increase in demand requires more and more development techniques for companies developing robotics, and the product described will be one small subset of these techniques. The diverse areas of application lead to a difficult challenge for robot designing engineers as there are numerous variables which can be difficult to factor into the design process. In this area of complication, the genetic algorithm thrives. By rapidly and randomly designing and testing the robots inside the simulated environment, many ideas can be deployed without the cost of building the robot with real materials. This saves significantly on the development time and cost of designing. The remaining pages of this document define the steps of what will be accomplished and various other project related objectives.

Objective

The objective of this project is to create software that will allow a client to request an initial design that can traverse a specific environment following a set of constraints. The specific environment will contain any number of factors and variables that define the difficulties of the terrain the robot is to traverse. The previously mentioned constraints will define the specific restrictions the robot will be designed within. For example, some robots may be required to be under a certain weight, while others will require a length or width restriction. This information will be rigorously sought, and specified by the user of the software. Once the information is received from the client a user can then enter the information into the software that will directly interact with the physics engine to create/test designs. Information will be passed to the physics engine from within the user interface that detail the construction and testing that the robot will undergo. The constraining information will be used to create and test the robots. Through this testing procedure, the robots will receive a score based on their fitness. Fitness will be defined based on the metrics of the testing parameters. One design may desire to test the pull strength of a robot while another may be interested in the top speed of the robot. By allowing the fitness to have different measures the software will be capable of developing many designs with different objectives. The scope of the project will be limited to simple walking robots, but even with this reduced scale, the project will require many man hours to complete.

Team Dynamics

The team developing the project will utilize a mostly democratic approach. However, the team consists of members with varying specializations. When the project shifts into a member's domain of expertise, the group will form a temporary team leader in order to best deploy that member's knowledge. The team will collaborate on each step of the development and will use a democratic approach to the general direction of a solution, but will split off the implementation of modules to individuals or pair programming/development groups. All members of the team will contribute to the planning and writing of the requirements, specifications, and designs documents, and to the execution of the design document in the spring of 2018. The following outlines who each member of the team is and what they will contribute in excess of the core software engineering requirements. Detailed below is the areas of expertise for which each member will lead.

As a Computer Science and Math major at California University, Kevin Beck will contribute in areas for which the need for math applications are the highest. The specifications of the algorithms that are used in the iterative evolutionary portions of the project will be mathematical in nature which will require close analysis. Kevin also has prior experience in the development of projects using simulated physics and will help to implement the physics engine component into the project's design. He will also be the main architect for the design and flow of the software and its elements in the Physics Engine.

Joseph Borsodi is a senior at California University studying for his Bachelor of Science in Computer Science degree. He will be responsible for creating the user interface that interacts with the physics engine. The UI must allow the user to quickly change all the settings including environment settings, robot specifications, and algorithm architecture, as well as be easily navigated without burden. He will then ensure that the data is passed from the front end to the back end to be realized within the physics engine.

Kristy Schworn is a dual major Computer Science and Math and as such will contribute greatly to the investigation and implementation of genetic algorithms and their application to the domain of robotic development. She will explore the various techniques for the combination and mutation of genes. She will also decide how the genetic code of each bot is stored, encoded, and interpreted within the software.

Ryan Romanosky, a Computer Science major, will work alongside Kristy in the accurate and effective implementation of genetics to the design of the robotics. He holds a BS in Biology and will be critical in analyzing how the genetic code of the robots' designs are encoded and transmitted to the next generations. He will provide insight into how the algorithm differs from natural evolution and what features of the algorithm closely mimic natural biology. He will also lead on the development of the individual features of the robots that will be included in the genetic algorithms and the robots' genetic code.

Application Domain

The product resulting from this project will be implemented within a research environment involving robotics design. With modern computers, rapid simulations can enable the creation of unintuitive robots that differ greatly from those designed by engineers. An engineer's design may adhere to certain established patterns, and miss less obvious designs. It is also costly to have a team of humans design a rover from start to finish. Thus, the primary goal of the project is to develop software that will allow a computer to create the initial designs that can successfully traverse terrain without the input of a human designer. After the initial designs have been created, the designs will be further developed in house or passed on to the client to expand on and build fully functional bots. The robotics industry uses robots in numerous conditions and this designing process will allow a user to input constraints or restrictions which will result in the design of a unique robotic solution. Each situation can be independently run, tested, and saved allowing for the rapid development of solutions for various environments. This will allow rapid prototyping of robotic designs in the simulation to be saved and analysed for further development or revision.

Initial Business Model

The system will likely be used by a casual user interested in the process of genetic algorithms or internally at a location where custom designing land traversing robotics is a component of the facility's manufacturing process. The software developed in this project will take specifications on terrain and size constraints and return the best robot design(s). The facility will then go on to add material tests and further develop the design before manufacturing a testing prototype. The software will reduce both the time and costs of the early design phase of the production of the robots through rigorous testing and refinement in a simulated environment utilizing genetic algorithms.

Description of Data Sources and Examples

A company utilizing this software would receive data from a client consisting of the terrain upon which the machine is to be developed, and restrictions on the robot itself. This data would consist of surface frictions, size and type of obstacles, as well as constraints on the size and weight of the robot. Broad examples of this data would be as follows:

- A robot weighing less than 25 pounds, walking across a coarse surface at an incline of 15 degrees
- A stair climbing robot where the stair height is 1.5 feet

- A robot that can consistently climb over a 1-foot tall obstacle while carrying 10 pounds and measuring less than 3 feet in length.

Using this data, the user would enter into the software the specifications as well as details on the algorithm's function and would run as many iterations deemed necessary to find the solution to the given terrain and constraints. The user would then pass this data forward to a design team at which point materials and mechanization will be tested for feasibility. The goal of this software is not to develop completed specifications for a robot, but rather design the initial version of a walking robot in the prescribed situation. Extensive testing and evaluation will be needed to move from the initial design concept to the final physical product.

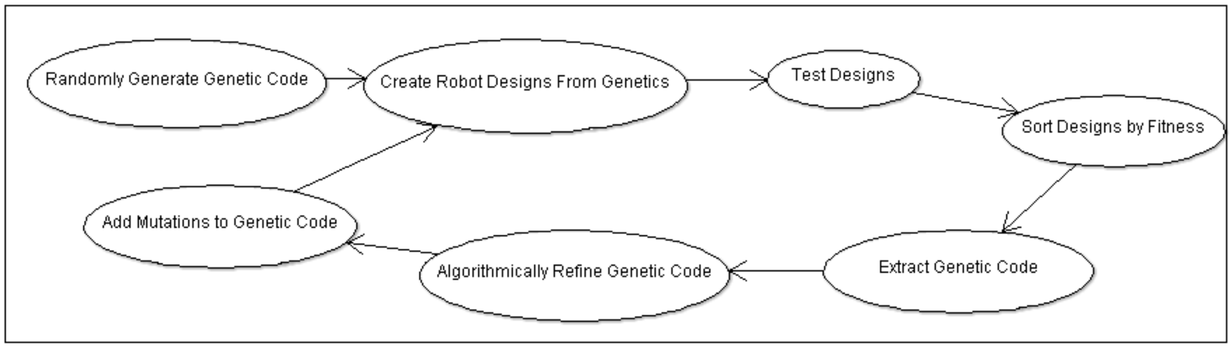
Randomly Generate Genetic Code
Create Robot Designs From Genetics
Test Designs
Sort Designs by Fitness
Extract Genetic Code
Algorithmically Refine Genetic Code
Add Mutations to Genetic Code

Figure 1

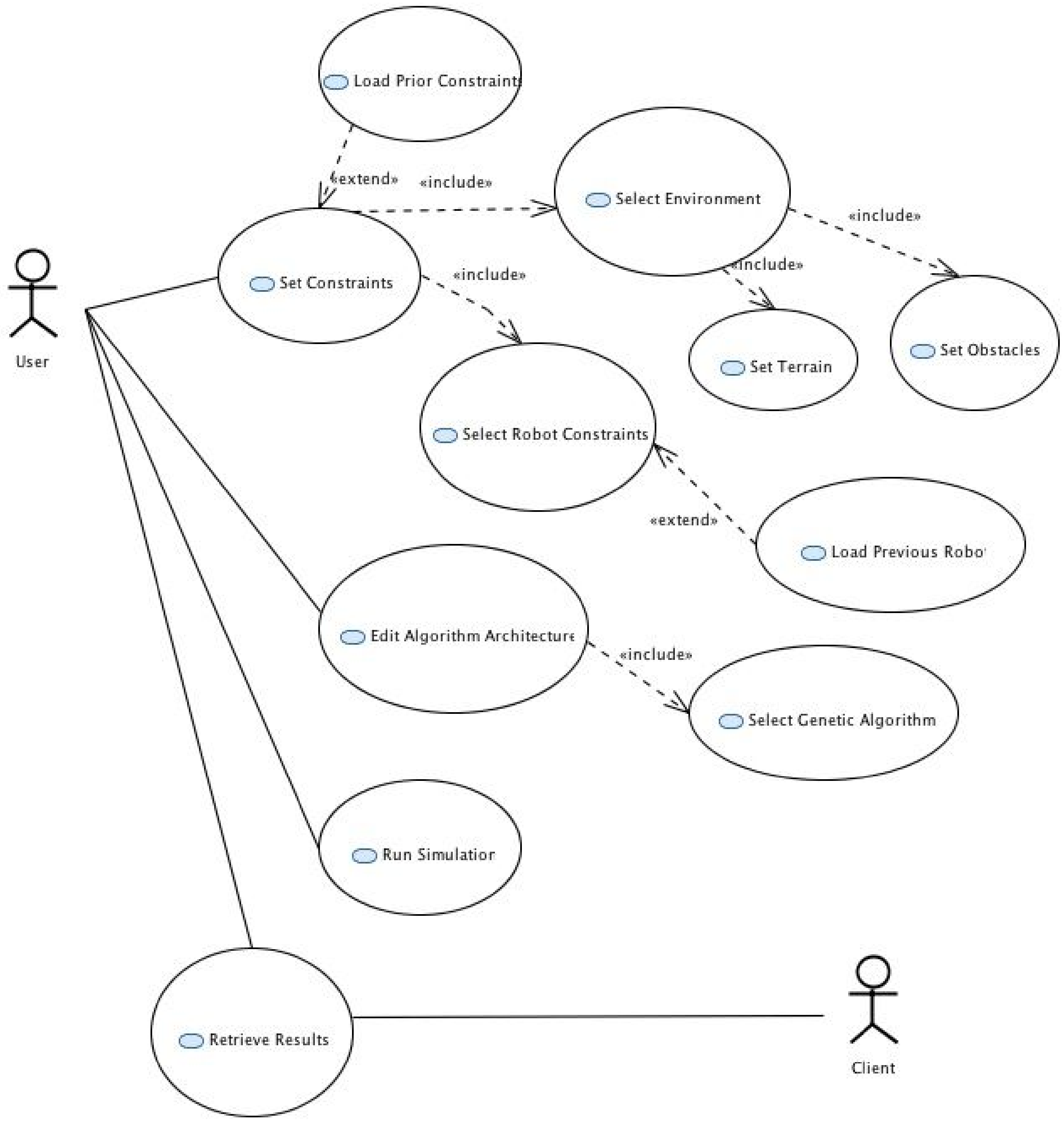
Load Prior Constraints
«extend»
«include»
Select Environment
«include»
«include»
Set Constraints
«include»
User
Set Terrain
Set Obstacles
Select Robot Constraints
«extend»
Load Previous Robo
Edit Algorithm Architecture
«include»
Select Genetic Algorithm
Run Simulation
Retrieve Results
Client

Figure 2

Initial Requirements

Functional

<u>**User Interface**</u>

The software developed will use a menu-driven environment that will direct the user to four main control areas. These four distinct areas of the program will be Environment Variables (EV), Robotics Specifications (RS), Algorithm Architecture (AA), and Program Settings. Environment Variables consists of all of the factors related to the terrain in which the robot will operate. Algorithm Architecture details the tools and specifications that the algorithm will use to develop each generation of robots. Robotics specifications will be restrictions on what features will be allowed to develop and the boundaries in which the randomization of each feature will adhere. The Program Settings will relate to the layout and visual/auditory settings within the software itself. Each of these categories is explained in more detail in the following paragraphs.

<u>**Terrain Settings**</u>

The user of the software will be given a type of terrain in which the robot is to be developed. Within the developed software, the user will enter in these specifications to best mimic the terrain. The parameters to be manipulated will consist of gravity, surface friction, obstacles, and incline. Other environmental variables will be obstacles that can be included in the simulation. These will consist of large and small rocks or boxes. These obstacles can be constant, or can be scaled over distance to create an increasingly difficult terrain for the bot to advance through.

Algorithm Architecture

Algorithm Architecture will manage all factors that indicate how the genetics of each iteration of bots is passed forward to the next. The architecture sector of the program will allow the user to change and manipulate how the program decides what defines success through the changing of a fitness variable, as well as change the specific type of genetic algorithm used. Through this tweaking and refinement, the user will be able to change the results of the simulated design by managing factors related to the algorithm itself.

Robot Specifications and Constraints

The factors related to the actual robot design will be handled through the Robotics Specification portion of the software. This section will relate to the specific limitations that the robotics will function. These will consist of size and weight restrictions as well as what features the robots can utilize within the simulation. The Robots will be given legs that function in a broad way, but the specifications of how long and strong the legs are will be randomized. The function of the leg lifting, moving forward, lowering, and moving back will be pre-coded into the leg component, the structure and strength of the leg component will then be randomized and manipulated, and attached to the body segments of the bot. The other factors that can be implemented onto the bot would be the ability to pull a weight with an indicated friction, or to carry an object of certain size and shape. Not all the factors will be included in each simulation, but are necessary as most walking robots will have a function that will include more than walking such as the mentioned carrying, dragging, etc.

Menu Settings

The program itself will have its own variables to alter and affect and the user will be able to change these factors through the Program Settings area of the software. This portion of the product will allow the user to save and load different previously run simulations as well as change the appearance and sounds related to the project. This will include brightness and volume as well as color schemes.

Running the Simulations

Once the user enters all the data requested by the client they will then be able to begin the simulations. As soon as the user hits the "Run" button the data entered will be transformed into a set of constraints the simulation must follow. From there onwards the program itself will randomly design walking robots following these guidelines. Once a design is generated it will get its own chance to move throughout the environment and make it as far as it can. The further the bot moves from the starting line to the goal the more its fitness rating will go up until it either completes the task or can no longer move forward. The data for this specific bot will be recorded then a new design will be created and tested in the environment. This will be repeated a number of times, each time keeping track of each bot and its final fitness, which will represent Generation 1.

Processing Generation 1

Once Generation 1 has finished running the simulations, the acquired data will be processed using a Genetic Algorithm (selected and specified in the User Interface). The algorithm will take the design of each robot in the form of a genetic code its fitness level achieved and create a new set of designs to be tested as generation 2. The algorithm will introduce mutations into the group through randomized parameters. By limiting or refining the randomness in the new iteration of robots, the algorithm allows the designs that have performed well in the previous generations to potentially achieve a higher fitness level. Through each generation, the fitness level will be tracked and the algorithm will continue to refine the results to favor the designs who perform most aptly.

Arriving at Generation X

Eventually the program will arrive at Generation X either through running a set number of times or the user determines the current generation is working well enough to end simulations. Allowing the user to end simulations will also in return allow them to set the total number of generations to be run fairly high and then they can end whenever they feel it is appropriate such as the current generation is meeting the client's criteria. The data for Generation X will be stored and then easily accessible through the main menu's UI. The user will be able to go see the results for the test and get a copy of them to provide for the client.

Nonfunctional Requirements

Running Time

The program's running time will vary from test to test, as each design needs its own allotted time to attempt to traverse the environment. However, the program will have certain features intended to save as much time as possible. For example, if a bot's fitness is decreasing or becomes constant that test can be prematurely ended to move on to the next one. This will save a noticeable amount of time over the creation and testing of many generations.

Integration into Physics Engine

It has been decided that in order to increase usability of the program the User Interface will be integrated into the physics engine. The merging of UI and physics engine in return should create less windows for the user, speed up time to enter and retrieve data, as well as decrease the initial start time between each simulation.

The software developed ideally will result in the creation of the most favorable robot for the designated terrain. A series of trials will run and, with the use of the genetic algorithm, multiple functional robots will begin to form. The algorithm will use crossover and mutation to develop several different machines. Through a process modeled after natural selection, the optimal machine will be created. Unity will be used as the physics engine for this project. This engine can be used for any platform required.

Recoverability

After each generation completes its simulation the results will be saved within the results tab marked as partial-incomplete. In the event the program is abruptly ended, power outage for example, the current simulation time will not be wasted and can be recovered. Initial plans are to include functionality that will allow the user to continue running partial-incomplete tests.

Scalability

The product will be engineered in a way that will allow greater expansion into the field of genetic design. In the future the product could be revised and refined to include more distinct features, better algorithms, and more terrain types to provide a better development environment. The condition in which the completed product will be at the end of this project, will be a state of modular base level design. Adding to the product in the future should be a simple and obvious task for any needed scaling or adjustments. There could be issues scaling the product to large numbers of simulations or allowing multiple computers access to the same simulation at a time is not within the scope of the current project.

Security and Maintenance

The long-term success of this software relies on security and maintenance. Maintenance for this software will be handled in a timely manner in order to keep the software reliable. The team will use both perfective and adaptive maintenance to ensure the software is kept up to date and functions properly. Security will not be the main concern as no part of the project will be connected to the internet. All information revolving around the project will only be accessible to the designers. Designs will not be vulnerable as they are basic archetypes, which removes any fear of losing or having leaked any proprietary information or intellectual property.

Performance

There will not be many restrictions on the type of machine that can run the simulations. Up-to-date, modern day machines should face no issues regarding the functionality of the simulations. While outdated technology will not be suitable for this project, it is not seen as a major restriction due to the team using technology that is fully capable of running the program. Since the program is being designed for the testing environment at a professional/large scale level, it is assumed that a standard mid-range computer will be able to run the program.

Testing / Revisions (evidence of)

Throughout the development of this project, the focus of the software has shifted to various entities before arriving to the current form. The original plan for the product was to design cars to drive utilizing some form of AI. Thorough exploration of this topic revealed substantial pitfalls and this idea was left for another time. The following iteration involved a concept of an AI competing in a video game which also led to an extremely involved product with an exceptionally broad scope. The final concept is the current structure outlined in this paper. Through iteration and revision, the application of the software has been scaled to a position where the goals are both attainable and productive. While a project involving high-level machine learning, or deep learning, would have been interesting, the scale of such a project is immense. By scaling the project to a single complex algorithm, the software is much more applicable to a single application: walking robots.

Throughout the development of the idea and the writing of this paper, many revisions were made. The scope of the project is directly proportional to the number of variables presented in the simulation. As the development will occur in a limited allotted time, the number of variables has decreased in order to reach the established goals in a meaningful manner:

1. Robot locomotion reduced from driving, flipping, and walking to walking due to time restrictions.
2. Robot shapes limited to cubes and spheres for initial version.

Though the number of variables has contracted, the option to *increase* our variables remains open in the event that our progress exceeds our expectations. "Feature creep" is only a detriment to a product if it adds delays for an unneeded, or unnecessary, addition.

The User Interface is the main way for the user to interact with the program, so it must be easy to use and understand. Therefore, the UI is to be integrated into the physics engine to help increase ease of use for the user. Other side effects of the integration include faster loading times and less windows to deal with.

Appendix: Technical Glossary

Environment: In the terms of this project the environment will be the generated terrain the robot traverses.

Fitness: The measure by which a robot is scored within the genetic algorithm. Depending on the application, this could be the distance or speed the robot covers or any number of other measures decided by the user of the software.

Gene: A subset of a genome that describes specific traits of a being. In this project the beings represented are robotic entities.

Generation: A description of what iteration the current population represents.

1. Generation 1: The first iteration of robot designs.
2. Generation X: The final iteration of robot designs.

Genetic Algorithm: An algorithm in which some form of evolution is applied to a genetic code leading to iterative generations.

Genome: In the domain of this project the genome is the entire list of values pertaining to the specification of a robot's construction

Instantiation: The act of creating an object within the simulation defined by genes. This can be a single component such as a leg or the entire entity defined by the complete genome.

Mutation: The alteration, addition, or removal of a gene from a genome. This occurs between different generations of robots.

Physics Engine: An environment in which physics and collisions are solved and simulated given certain parameters.

Population: The entire list of currently active robots within the simulation.

Species: A loose description of what potential groupings can be made from a population of robots. These groupings are identifiable through specifics gene similarities. For example, a species could be the subset of the population which has a single pair of legs, or that has a certain number of body segments.

Unity3D: Unity is a specific physics engine capable to provide a platform on which the software will be constructed.

Walking Robot: In this domain, a walking robot is represented by any number of robotic designs, all of which utilize at least 1 pair of legs. These robots have various applications and the term is intentionally vague to include all potential applications of this project.

Appendix: Team Details

Kevin Beck:

Kevin managed the separation of tasks for the requirements document. The tasks were parsed and subdivided into sections that were then delivered to each member of the team. As the leader for the requirements document, Kevin created the general framework. This framework enabled the team to develop the paper quickly and fluidly. By dividing the tasks into subsections, each portion of the paper to different authors, each thought could be iterated on by each member of the team. Kevin created the UML diagrams and wrote the Initial Business Model, and Initial Requirements portions of the paper. After the recombination of the document, the final revision was done cooperatively by the group to ensure validity and correctness.

Kristy Schworn:

Kristy handled several portions of the nonfunctional requirements. Details on matters regarding security and maintenance were covered as well as applying biology to the idea of the genetic algorithm's functionality. Terms used both in biology and the genetic algorithm were brought together to describe the usefulness of this algorithm when simulating natural selection. Kristy also took care of revision of the document section by section. Grammatical and spelling corrections were made throughout the making of the paper. Once the requirements document was complete, the team came together to do a final error check to the entire document.

Joseph Borsodi:

Joseph Borsodi was responsible for expanding upon the initial requirements to provide a clear description of what the program will do once the user enters the data provided by the client and beings running simulations. Joe created the table of contents and was also responsible to create portions of the nonfunctional requirements and testing/revisions sections that involve the User Interface and its ease of use aspects. Joe was apart of the final revision and error checking of the project.

Ryan Romanosky:

Ryan created the glossary containing all the key terms throughout the paper. He also created the abstract, outlining the main idea of the project using the major keywords involved in this document. Ryan aided in the creation of the UML diagrams and worked on the overall organization of the document as well as creating the cover page and header. Ryan joined the rest of the group in the final revision of the document.

Appendix: Workflow Authentication

The above document is valid and correctly portrays the workflow and production of the group.

Kevin Beck___________________________ Date_____________________

Kristy Schworn________________________ Date_____________________

Ryan Romanosky_______________________ Date______________________

Joseph Borsodi_________________________ Date______________________