SE 3XA3: Requirements Document
Genetic Cars

Team 8, Grate
Kelvin Lin (linkk4)
Eric Chaput (chaputem)
Jin Liu (liu456)

October 11, 2016

Contents

1 Project Drivers **1**
- 1.1 The Purpose of the Project . . . 1
- 1.2 The Stakeholders . . . 1
  - 1.2.1 The Client . . . 1
  - 1.2.2 The Customers . . . 1
  - 1.2.3 Other Stakeholders . . . 1
- 1.3 Mandated Constraints . . . 1
- 1.4 Naming Conventions and Terminology . . . 1
- 1.5 Relevant Facts and Assumptions . . . 1

2 Functional Requirements **1**
- 2.1 The Scope of the Work and the Product . . . 1
  - 2.1.1 The Context of the Work . . . 1
  - 2.1.2 Work Partitioning . . . 3
  - 2.1.3 Individual Product Use Cases . . . 3
- 2.2 Functional Requirements . . . 4

3 Non-functional Requirements **14**
- 3.1 Look and Feel Requirements . . . 14
  - 3.1.1 Appearance Requirements . . . 14
  - 3.1.2 Style Requirements . . . 14
- 3.2 Usability and Humanity Requirements . . . 14
  - 3.2.1 Ease of Use Requirements . . . 14
  - 3.2.2 Personalization Requirements . . . 15
  - 3.2.3 Learning Requirements . . . 15
- 3.3 Performance Requirements . . . 15
  - 3.3.1 Speed and Latency Requirements . . . 15
  - 3.3.2 Precision and Reliability Requirements . . . 15
  - 3.3.3 Longevity Requirements . . . 16
- 3.4 Operational and Environmental Requirements . . . 16
  - 3.4.1 Productization Requirements . . . 16
- 3.5 Maintainability and Support Requirements . . . 16
  - 3.5.1 Maintenance Requirements . . . 16
  - 3.5.2 Supportability Requirements . . . 16
  - 3.5.3 Adaptability Requirements . . . 16
- 3.6 Security Requirements . . . 16

3.6.1 Access Requirements . . . 16
3.6.2 Integrity Requirements . . . 16
3.6.3 Privacy Requirements . . . 16
3.7 Cultural Requirements . . . 16
3.8 Legal Requirements . . . 16
3.9 Health and Safety Requirements . . . 16

4 Project Issues **16**
4.1 Open Issues . . . 16
4.2 Off-the-Shelf Solutions . . . 17
4.2.1 Ready-Made Product . . . 17
4.2.2 Reusable Components . . . 17
4.2.3 Products That Can Be Copied . . . 17
4.3 New Problems . . . 17
4.3.1 Effects on the Current Environment . . . 17
4.3.2 Effects on the Installed Systems . . . 18
4.3.3 Potential User Problems . . . 18
4.3.4 Limitations in the Anticipated Implementation Environment that May Inhibit the New Product . . . 18
4.3.5 Follow-up Problems . . . 18
4.4 Tasks . . . 19
4.4.1 Project Planning . . . 19
4.4.2 Planning of the Development Phases . . . 20
4.5 Migration to the New Product . . . 20
4.6 Risks . . . 20
4.7 Costs . . . 20
4.8 User Documentation and Training . . . 21
4.9 Waiting Room . . . 21
4.10 Ideas for Solutions . . . 21

5 Appendix **22**
5.1 List of Figures . . . 22
5.2 Symbolic Parameters . . . 22

List of Tables

1 **Revision History** . . . iii

2 **Work Partitioning Table** 3
3 List of Figures 22

List of Figures

1 The Context Diagram for Grate's Genetic Cars 2
2 The V-Model of Software Development 19

Table 1: **Revision History**

Date	**Version**	**Notes**
October 7, 2016	1.0	Started Functional Requirements
October 10, 2016	1.1	Updated Functional Requirements
October 11, 2016	1.2	Added Context Diagram
October 11, 2016	1.3	Added Work Partitioning Table
October 11, 2016	1.4	Added Off-the-Shelf Solutions

This document describes the requirements for .... The template for the Software Requirements Specification (SRS) is a subset of the Volere template (Robertson and Robertson, 2012). If you make further modifications to the template, you should explicity state what modifications were made.

1 Project Drivers

1.1 The Purpose of the Project

1.2 The Stakeholders

1.2.1 The Client

1.2.2 The Customers

1.2.3 Other Stakeholders

1.3 Mandated Constraints

1.4 Naming Conventions and Terminology

1.5 Relevant Facts and Assumptions

User characteristics should go under assumptions.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

The following depicts a context diagram for Grate's Genetic Cars:

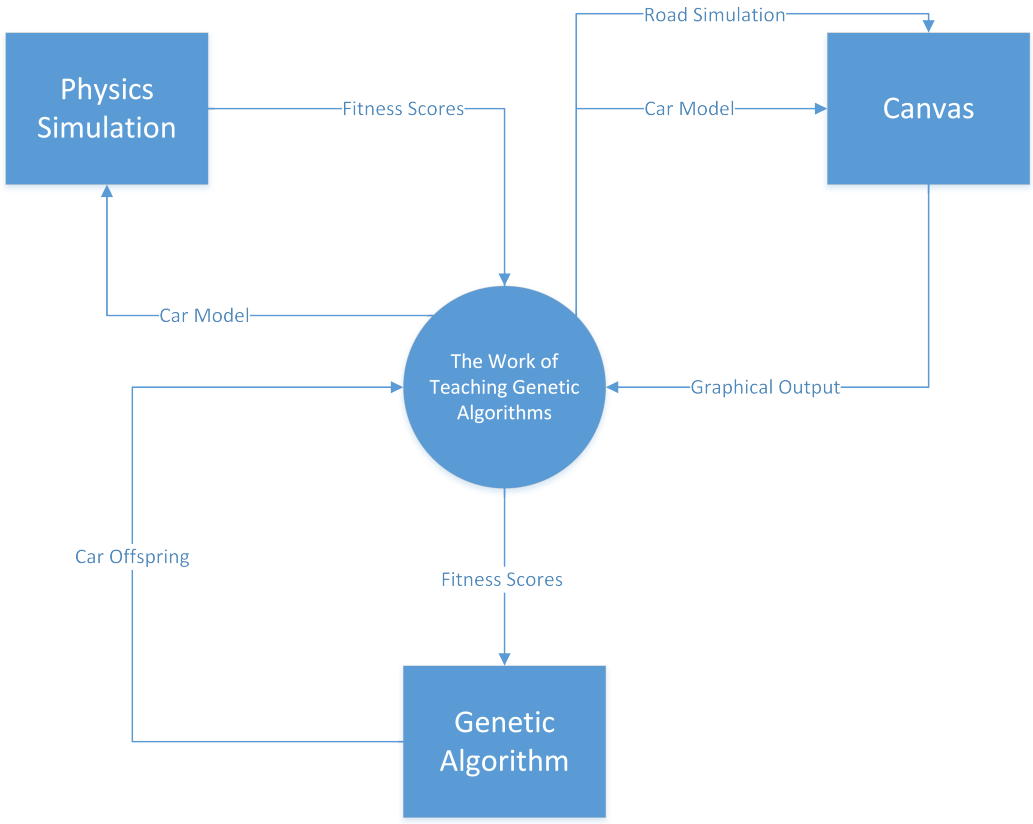

Figure 1: The Context Diagram for Grate's Genetic Cars

2.1.2 Work Partitioning

Event Name	Input and Output	Summary
1. Physics Simulation simulates generation	Car Offspring (in), Fitness Scores (out)	Calculate parents for the next generation
2. Canvas displays car results	Car model (in), Road simulation (in), Graphical Display (out)	Draw car model, Track top cars, Restart at the end of the generation
3. Genetic algorithm generates new generation	Fitness Scores (in), Car Offspring (out)	Selects parents from fitness scores, Cross over genes, Mutate genes

Table 2: **Work Partitioning Table**

2.1.3 Individual Product Use Cases

1. Normal Operation

    - User launches program.
    - The Genetic Algorithm generates a random seed.
    - The random seed is used to generate offspring using the default parameters.
    - The Physics Simulation takes the offspring (Car Model) and performs physics simulations to determine their fitness score.
    - The results are sent to the Canvas, which displays the results graphically.
    - The fitness scores are sent back to the Genetic Algorithm to generate new offspring.

2. User Modifies Any Parameter

    - User launches program.

- User modifies fields in the program that pertain to the Genetic Algorithm's attributes.
- The Genetic Algorithm generates offspring based on the user's input.
- The Physics Simulation takes the offspring (Car Model) and performs physics simulations to determine their fitness score.
- The results are sent to the Canvas, which displays the results graphically.
- The fitness scores are sent back to the Genetic Algorithm to generate new offspring.

2.2 Functional Requirements

Requirement #: 1 Requirement Type: Functional
Description: Each car must be composed of at least v vectors.
Rationale: This requirement manages the complexity of the car model, allowing for realistic distribution of traits among members of a population. That is, this prevents large cars from being generated and using an excessive amount of memory.
Originator: Kelvin Lin
Fit Criterion: No car generated within population p shall be composed of more than v vectors.
Supporting Materials: JavaScript
History: Created October 7^{th}, 2016

Requirement #: 2 Requirement Type: Functional
Description: Each car may not have more than *number_of_vertices* wheels.
Rationale: The wheels must be attached to the car via a vertex between two connecting vectors. This requirement ensures that no redundant or unused wheels will be generated.
Originator: Kelvin Lin
Fit Criterion: No car generated within population p shall be composed of more than *number_of_vertices* wheels.
Supporting Materials: JavaScript
History: Created October 7$^{\text{th}}$, 2016

Requirement #: 3 Requirement Type: Functional
Description: The radius of each wheel must be at most r units.
Rationale: This requirement manages the complexity of the car model, allowing for realistic distribution of traits among members of a population. That is, cars with unrealistically sized wheels will not be generated.
Originator: Kelvin Lin
Fit Criterion: No cars generated will have wheels with a radius larger than r.
Supporting Materials: JavaScript
History: Created October 7$^{\text{th}}$, 2016

Requirement #: 4 Requirement Type: Functional
Description: The center of each wheel generated must be attached to a vertex formed by connecting vectors.
Rationale: Wheels cannot be floating on or around the car. This requirement ensures visual coherency by requiring wheels to be attached to the car model. Knowing the center of the wheel will also allow the physics engine to calculate the torque and distance that the car travelled.
Originator: Kelvin Lin
Fit Criterion: Each wheel displayed on the screen is attached to a vertex formed by connecting vectors.
Supporting Materials: JavaScript
History: Created October 7^{th}, 2016

Requirement #: 5 Requirement Type: Functional
Description: The mass of each car must not be less than *min_weight*.
Rationale: In order to have realistic physical simulations of car models, the mass of the car must have a lower limit. The lower limit will guarantee the simulations to work as expected.
Originator: Kelvin Lin
Fit Criterion: The mass of any car models generated are greater than *min_weight*.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 6 Requirement Type: Functional
Description: The mass of each car must not exceed *max_weight*.
Rationale: In order to have realistic physical simulations of car models, the mass of each car must have an upper limit. An upper limit reduces the possibility of type incompatibility with certain APIs. Additionally, it ensures that the mass of each car is encoded using a known number of bits.
Originator: Kelvin Lin
Fit Criterion: The mass of any car models generated do not exceed *max_weight*.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 7 Requirement Type: Functional
Description: The program shall display each generation of cars traversing the road.
Rationale: The purpose of this program is to show its users the effects of genetic algorithms in an interesting and engaging manner. If the program did not display each generation of cars traversing the road, then the program would fail in accomplishing its original objective.
Originator: Kelvin Lin
Fit Criterion: Each generation of cars can be seen traversing a road on the medium of output.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 8 Requirement Type: Functional
Description: The program shall display the fitness of the top n cars.
Rationale: The ability to compare the performance of cars during each generation is useful for observing the effects of genetic algorithms because it shows the users the improvement and regression of the car's performance over time.
Originator: Kelvin Lin
Fit Criterion: A medium of output exists to provide the fitness of the car on the medium of display.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 9 Requirement Type: Functional
Description: The program shall allow the user to enter a random seed to generate cars from in lieu of a randomly generated seed.
Rationale: The ability to enter a random seed allows the results of cars to be compared and to be run on multiple computers: results are not lost as a result of restarting the application.
Originator: Kelvin Lin
Fit Criterion: The user can input a random seed into the program through an input device, and the random seed is used to dictate the random behaviours of the program.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 10 Requirement Type: Functional
Description: The user shall be allowed to modify the mutation rate, *mutation_rate*.
Rationale: Allowing the users to modify the mutation rate allows the program to fulfil its objective by showing the users how the mutation rate can impact the performance of the cars.
Originator: Kelvin Lin
Fit Criterion: The user can input a mutation rate into the program through an input device, and the mutation rate is used to produce offspring in the program.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 11 Requirement Type: Functional
Description: The user shall be allowed to change the number of cars per generation s in lieu of the default value.
Rationale: Allowing the user to change the number of cars per generation s will allow the user to see how the size of a generation affects the genetic algorithm.
Originator: Kelvin Lin
Fit Criterion: s is equal to the user's input for every generation produced by the program.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 12 Requirement Type: Functional
Description: The road generated must be the same across all generations.
Rationale: Using the same road for each generation allows for comparability of performance between each generation. That is, since every car will traverse the same course, their fitness and performance can be compared.
Originator: Kelvin Lin
Fit Criterion: The road for all simulations is the same.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 13 Requirement Type: Functional
Description: The product must generate at least s car samples per generation.
Rationale: GAs improve by having a large number of samples (representing members in a population) intermix traits. This requirement allows the GA to work by guaranteeing that a sufficient sample will be present at all times.
Originator: Kelvin Lin
Fit Criterion: Given a user generated input, s, the program should generate s cars for each generation.
Supporting Materials: JavaScript
History: Created October 7^{th}, 2016

Requirement #: 14 Requirement Type: Functional
Description: The number of cars per generation s shall not exceed *max_cars_per_gen*.
Rationale: Having a maximum number of cars per generation prevents memory overflow from generating too many cars per generation.
Originator: Kelvin Lin
Fit Criterion: The number of cars generated per generation does not exceed *max_cars_per_gen*.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 15 Requirement Type: Functional
Description: The program shall use the top t cars to generate offsprings.
Rationale: The number of cars allowed to reproduce needs to be specified; otherwise, no improvement can be made in car performance over the generations.
Originator: Kelvin Lin
Fit Criterion: The parent cars of the offspring are within the top t cars.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 16 Requirement Type: Functional
Description: The top t cars shall not exceed t_max.
Rationale: This restriction prevents t from exceeding s or take on an unreasonable value. It ensures that the program can always run by setting an upper limit to the number of cars that can reproduce in a given generation.
Originator: Kelvin Lin
Fit Criterion: The number of cars to choose from during reproduction does not exceed t_max.
Supporting Materials: JavaScript
History: Created October 10th, 2016

Requirement #: 17 Requirement Type: Functional
Description: The top t cars shall not be less than t_min.
Rationale: This requirement ensures that there will be a sufficient number of cars to produce offspring in the subsequent generations.
Originator: Kelvin Lin
Fit Criterion: In each generation, there are at least t_min parents to generate offspring.
Supporting Materials: JavaScript
History: Created October 10th, 2016

Requirement #: 18 Requirement Type: Functional
Description: A car that stalls for more than *max_secs* shall be deemed non-moving.
Rationale: A time limit needs to be imposed on the simulations in order to prevent the cars from running indefinitely without making progress.
Originator: Kelvin Lin
Fit Criterion: All cars that say in the same spot for *max_secs* are marked as non-moving and the simulation for that car is stopped.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 19 Requirement Type: Functional
Description: The fitness of a car shall not be calculated until a car is deemed to be non-moving.
Rationale: The fitness of a car is determined by distance it moves during the simulation, and the simulation runs while the car is moving. Therefore, the fitness of a car cannot be determined until the car is non-moving.
Originator: Kelvin Lin
Fit Criterion: After a car is deemed non-moving, it's fitness value can be assessed.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

Requirement #: 20 Requirement Type: Functional
Description: The user shall be able to specify t in lieu of the default value.
Rationale: This will allow users to see the effect of changing the selectivity of the genetic algorithm.
Originator: Kelvin Lin
Fit Criterion: In each generation, t cars are chosen to generate offspring.
Supporting Materials: JavaScript
History: Created October 10^{th}, 2016

3 Non-functional Requirements

3.1 Look and Feel Requirements

As discussed in section 1.2 of this document, the users of this product include students and others interested in learning about genetic algorithms. With this in mind, the Genetic Cars project must be accessible to those without a background in mathematics or computer science. This accessibility begins with the look and feel of the project. The Genetic Cars project should appear aesthetically pleasing while still presenting its functions in as clean a manner as possible.

3.1.1 Appearance Requirements

The product shall be attractive to a student audience, with an emphasis on secondary and post-secondary students. A sampling of representative users shall, without prompting or enticement, be able to comprehend and use the product within sixty seconds of their first encounter with it. This same sampling shall also rate the appearance of the product on a scale from 1 to 10, and this rating shall be used to evaluate and refine the product's appearance. All licensing shall also be clear for the user to observe upon use of the product.

3.1.2 Style Requirements

The product shall appear inviting and educational and professional. After their first encounter with the product, a majority of representative users shall, without enticement, agree that they feel they would want to utilize the product and that they would learn about Genetic Algorithms by using the product. Representative users should also feel that they can trust the product.

3.2 Usability and Humanity Requirements

3.2.1 Ease of Use Requirements

The product shall be easy for anybody over the age of 6 to use. The product shall not expect the user to remember anything about the product given multiple uses. The product shall make the user want to use it and to show

the product to their friends/family/etc.. The product shall be used by people with no training or education except for a basic knowledge of the English language and the most very basic functions of a computer, such as how to navigate to a web-site and how to enter inputs when prompted to do so. A representative sample of users shall be able to successfully complete a given set of tasks with the product within a specified period of time to be determined at the time of the sample. The representative sample shall also show a willingness to show the product to others.

3.2.2 Personalization Requirements

The product shall allow the user to make simple adjustments to the product to allow for a variable length and amount of trials depending on user input.

3.2.3 Learning Requirements

The product shall be easy for an intended user of the product to learn. The product shall be able to be used by these users with no training before use. A representative sample of users shall be able to successfully complete a given set of tasks with the product within a specified period of time to be determined at the time of the sample.

3.3 Performance Requirements

3.3.1 Speed and Latency Requirements

The response time of the product shall be fast enough to avoid a loss of interest by the user following an input, which shall be a period of time no longer then five seconds. The initialization of the product shall be no longer then one minute.

3.3.2 Precision and Reliability Requirements

The product shall always converge towards a more optimal car. The product shall achieve 99 percent uptime. The product display shall be accurate to two decimal places.

3.3.3 Longevity Requirements

The product shall be easy to update and upgrade following its initial public release.

3.4 Operational and Environmental Requirements

3.4.1 Productization Requirements

3.5 Maintainability and Support Requirements

3.5.1 Maintenance Requirements

3.5.2 Supportability Requirements

3.5.3 Adaptability Requirements

3.6 Security Requirements

3.6.1 Access Requirements

3.6.2 Integrity Requirements

3.6.3 Privacy Requirements

3.7 Cultural Requirements

3.8 Legal Requirements

3.9 Health and Safety Requirements

This section is not in the original Volere template, but health and safety are issues that should be considered for every engineering project.

4 Project Issues

4.1 Open Issues

Not applicable for this project.

4.2 Off-the-Shelf Solutions

4.2.1 Ready-Made Product

Similar solutions to Grate's Genetic Cars already exists, notably BoxCar2D (boxcar2d.com/) and Rednuht's Genetic Cars (rednuht.org/genetic_cars_2/). Both products demonstrate the effect of genetic algorithms through evolving cars by selecting for the longest distance travel. They differ in that BoxCar2D displays one car at a time, whereas Rednuht's Genetic Cars displays all of the cars in one generation at once. They both allow users to adjust parameters of the genetic algorithm in order to observe the effects of the algorithm. They both show the performance of the cars over time.

4.2.2 Reusable Components

The Box2D API and the D3 API can both be reused in Grate's Genetic Cars. The Box2D API provides functionality such as Vectors and Polygons that can be used to model the car. The D3 API provides visualization functionality that can be used to visualize the performance of cars over time. These APIs provide functionality external to the core functionality of Grate's Genetic Cars, which will make them valuable assets as Grate will not have to reimplement these APIs.

4.2.3 Products That Can Be Copied

Rednuht's Genetic Cars is licensed under the author's custom license which grants others the right to reuse his code as part of their solution. Furthermore, both BoxCar2D and Rednuht's Genetic Cars have released their algorithms in some form: BoxCar2D through visuals and text, and Rednuht's Genetic Cars through source code. Grate's Genetic Cars can draw inspiration from these sources in implementing a new innovative solution to teaching genetic algorithms.

4.3 New Problems

4.3.1 Effects on the Current Environment

Not applicable for this project.

4.3.2 Effects on the Installed Systems

Not applicable for this project.

4.3.3 Potential User Problems

Users may experience fatigue from prolong use of the product. Symptoms of fatigue can include eyestrain, dizziness, nausea, muscle pain, or general discomfort. To prevent fatigue, users should take periodic breaks when using the program for prolonged periods of time. However, fatigue should not pose a significant risk for this project, as the target audience for Grate's Genetic Cars is older teenagers and adults who have prior experience using computers and are knowledgeable about the health concerns associated with prolong use of computers.

Users with epilepsy or prior history of seizures may also experience seizures or blackouts while using the product. Symptoms of seizures may include convulsions, eye or muscle twitching, loss of awareness, altered vision, involuntary movements, and disorientation. However, this risk is not significant as the risk of having a seizure from light flashes or patterns is low (about 1 in 4000), and the likelihood of experiencing a seizure can be reduced through simple steps. Users can reduce the risk of experience seizures while using the product if they sit or stand away from the screen, use the smallest screen possible, use the product in a well lit room, take frequent breaks, and refrain from using the product if tired.

4.3.4 Limitations in the Anticipated Implementation Environment that May Inhibit the New Product

Not applicable for this project.

4.3.5 Follow-up Problems

The implementation environment may become depreciated before the completion of the project, and the platforms the product is built for may no longer support the product after completion of implementation. This is a significant risk as modern programming languages and operating platforms are constantly evolving; however, this risk can be mitigated through writing maintainable code that can be converted to new standards as they arise.

4.4 Tasks

4.4.1 Project Planning

The V-Model of Software Development will be used in the development of this project. The V-Model to be followed is depicted below:

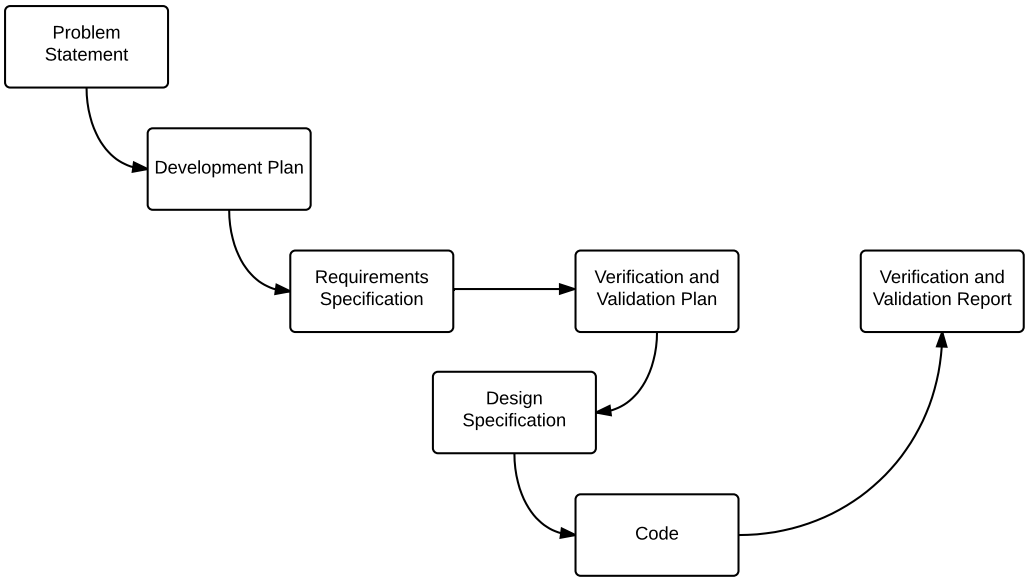

Figure 2: The V-Model of Software Development

Grate is dedicated to writing high quality code through the application of strong software engineering principles. In order to meet that standard, the project will begin separated into 3 main stages. First, an extensive design and documentation phase before any part of the system is implemented. Once the project has been designed, then coding can commence. At the conclusion of coding, there will be an extensive testing and validation phase in order to build confidence in the functionality of the final product. The total time anticipated for this project is 4 months.

The design phase contains 6 deliverables: the problem statement, the development plan, the requirements specifications, the verification and validation plan, and the design specification. Creating the problem statement and the development plan should both take about 1 week. Creating the re-

quirements specification, the verification and validation plan, and the design specification will all take about 2 weeks.

The coding phase involves implementing the project following the design specifications. The implementation of the project will take about 3 weeks.

Verification and Validation involves testing the project following the Verification and Validation Plan. The verification and validation of the project will take about 2 weeks.

Note that, despite the arrows in the diagram, this is an iterative process. This means that processes can happen more than once during the development process, with each iteration making improvements upon the previous iteration. Steps can also be skipped within iterations, or be performed out of order if deemed necessary.

4.4.2 Planning of the Development Phases

4.5 Migration to the New Product

Not applicable for this project.

4.6 Risks

The Box2D API poses the most signicant risk for the car model. The Box2D API denes the car entity in terms that can be used with many physics equations, which is important for calculating the tness function of the car. In the event that the Box2D API proves to be infeasible for Team 8, alternate arrangements will have to be made in order to complete the project: the team will resort to using basic kinematics equations to calculate the tness function instead of using the API. A possible drawback to this approach would be that the members of Team 8 are generally unfamiliar with Newtonian mechanics, so external assistance would be required.

4.7 Costs

There will be no cost at all since all the software (Latex editor, code complier etc.) and web-hosting are free.

4.8 User Documentation and Training

The user documents will be simple and efficient for our project since this project will not ask the user to do many things. The main responsibility for training documentation is letting user familiar with the start button, reset button and output table.

4.9 Waiting Room

Audio effect is expected to be add to this project.

4.10 Ideas for Solutions

Good structure and design for this project.

References

James Robertson and Suzanne Robertson. *Volere Requirements Specification Template*. Atlantic Systems Guild Limited, 16 edition, 2012.

5 Appendix

5.1 List of Figures

5.2 Symbolic Parameters

Symbol	Definition
s	The number of samples in a generation
v	The number of vectors in a car
$number\_of\_vertices$	The number of vertices formed by connecting vectors in a car model
r	The radius of a wheel
min_weight	The minimum mass of a car
max_weight	The maximum mass of a car
max_secs	The maximum amount of time a car is allowed to stall in one spot
n	The number of car statistics to display
$mutation\_rate$	The rate at which genes mutate
$max\_cars\_per\_gen$	The maximum number of cars in a given generation
t	The number of parents in a generation
t_max	The maximum number of parents
t_min	The minimum number of parents

Table 3: List of Figures