GROUP SOFTWARE DEVELOPMENT

Business Proposal Document

Group A

Muhammad Amirul Haziq Bin Musa – 13033634 – MSc Software Engineering
Andrea Chitty – 15033904 – MSc Software Engineering
Dewey Roskilly – 16037209 – MSc Information Technology
Magda Mosy – 06971609 – MSc Software Engineering
Nicola Hogg – 09027847 – MSc Information Technology

Table of Contents

1 Executive Summary

We are Team A – a small software development team from the University of West England, Bristol. The purpose of the proposal is to analyze, design and build a software system based on X-machine theory.

We are to develop a lightweight, focused and intuitive testing tool that is suitable for ‘thin clients’ that have limited memory.

The tool is to systematically test compiled java code. Systematic testing is a much more in-depth means of debugging software. The aim of systematic tests is to test the software against a broad assortment of inputs in order to find as many errors as possible. [1]

This is vital in certain software development projects where a failed system could potentially cause the loss of human life. Such systems can be found everywhere in our lives from aircraft computer systems to medical machines and equipment.

We feel this lightweight option to exhaustive testing is an over looked gap in the market and hope to develop the Y-Machine to fill it!

2 Background

2.1 Introduction

Testing is an essential part of software development. Whilst new code can appear to be error free, this is very rarely the case, even after close visual inspection. This divide between wishful thinking and reality has long been recognised. Many attempts have been made to address this gap between outcome and intent, from purist mathematical approaches such as functional programming to more pragmatic techniques such as object-oriented programming. None of these methods can by themselves guarantee bug-free operation in a large, practical system.

Even at the coding stage, the rise of agile techniques such as pair programming, in which a developer's coding is monitored by another developer to spot any defects, testing is as valuable today as it has ever been.

In fact, with the rise in complexity of systems, testing has become more and more essential. Whilst an individual component might appear to be error free after unit testing, its behaviour when combined with other components and/or web services may lead to abhorrent idiosyncrasies further on in the execution of the software. Even in unit testing, confidence is only as good as the set of tests. Testing cannot confirm the absence of defects, but it can confirm their presence.

Meyers defines software testing as:

"*The process of executing a program with the intent of finding errors*" [2]

He goes on to say that the common interpretation is that we test to prove there are no errors and that by aiming for the goal of zero errors we are conditioning ourselves psychologically to succeed at that goal and find no errors. We need to turn that on its head and search for errors in order to successfully find them. Software testing is a costly business. It is said that in terms of time software testing can occupy 50% of the development time and over 50% of the development budget [2][3][4]. Indeed for critical systems the effort required for testing can represent a much higher proportion of development time and budget. It is also well acknowledged that errors missed in the early stages of development are far more costly to fix in the later stages or after release of the product. From these facts alone we can determine that testing software requires significant resources including staff and tools.

Effective tools can help to reduce the substantial costs of testing and there is a large variety already on the market. Whilst automated tools can never be relied upon to catch 100% of the errors, they can give a developer an insight into the quality of the code they have written with minimal effort.

2.2 Finite State Machine

A finite state machine is an abstract machine that can only be in a single state at any time. The machine moves from state to state when certain transition conditions are satisfied. The transition conditions will be satisfied by inputs from outside the system.

A simple example of a finite state machine would be the operation of a lift that operates between two floors, ground floor and first floor. The state machine representing the lift will therefore have only two states, ground floor and first floor. The lift can transition from one floor to another by the pressing of a button either inside the lift or externally to the lift.

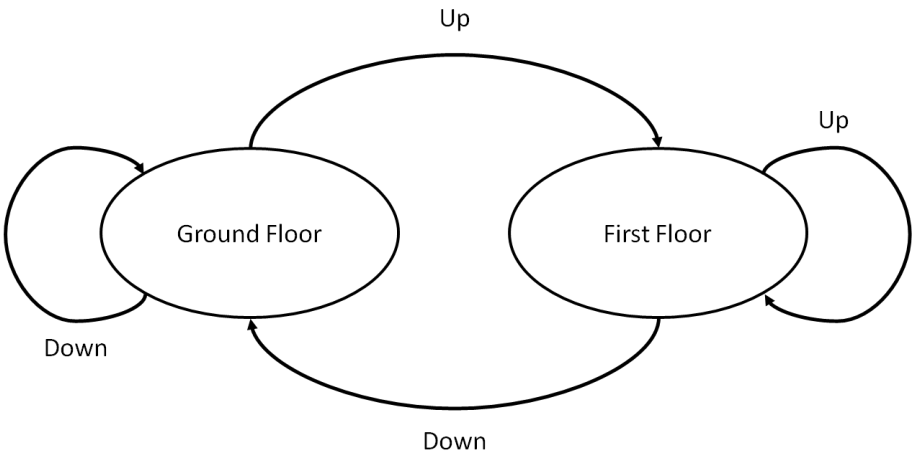

Figure 1: Example of finite state machine

All arcs that leave a single state are mutually exclusive, in other words for a given set of conditions there can only be a single route to leave the state. Also all arcs leaving a single state are collectively exhaustive, so there must be a way to leave a state given a set of conditions. Finite State Machines have been a tool in digital hardware design for many decades, largely because they are a powerful way to avoid unconsidered inputs and state transitions.

The simplicity and predictability of FSM makes them easy for experienced developers to implement. FSM have been around for a long time and their application is well developed. It is also easy to determine how a certain state can be achieved. These characteristics make FSM particularly useful for safety-critical applications.

Many development organizations, however, do not use this technique at all. Like all rigorous specification techniques, it can cramp creativity. In the real world, complex interactive applications also develop as they are written and prototype behaviour is observed. For FSM to become practical in such an environment, it must be easy to back-fit changes into the state machine, which is effectively the design master.

2.3 X-Machine

In 1974 Samuel Eilenberg, an American mathematician and computer scientist developed X-Machine theory. His work went unused until in 1986 Mike Holcombe of Sheffield University started to use X-Machines for specification purposes. It can be said that Mike Holcombe almost single-handedly reintroduced X-Machine theory and expanded it into a scientific and engineering discipline.

Also in 1986, Mike Holcombe was joined by Mike Stannett who had recently completed a PhD in general topology at Sheffield. Stannett began developing non-standard versions of X-Machines and published various papers.

At the time, the Formal Methods Group in the Computer Science Department (also known as "ThR@Sh" [Theoretical Research at Sheffield]), comprised Holcombe and Stannett. Initially, the group focused on establishing the properties of X-Machines and understanding how they could be used to model complex systems.

In simple terms, an X-Machine is a device or system for constructing, destroying or amending entities of type X. For example, a word processor is a machine for processing electronic documents and a calculator is a machine for processing numbers. Whilst the machine manipulates its objects it will move through various states and as it moves from state to state it will commit an action upon the objects.

3 Problem Statement

Our project will aim to use X-Machine Theory to identify coding errors. We will create a desktop-based application that can accept a compiled file of Java code and on execution will present the user with the faults that might exist within that code.

During the project, we will investigate how to derive X-Machine models and to use them as inputs to the desktop application for verifying Java code.

To evaluate the effectiveness of our tool, we will endeavour to find suitable case studies, including sources of compiled code that:

- Contain known errors
- Contain no known errors
- May or may not contain errors

Using the case studies, we will assess the performance of the tool by checking the code for the existence of errors.

During initial evaluation, we will investigate accuracy and precision in detecting known errors in compiled code, followed by confirming that no spurious errors are reported in known-good code.

Finally, we will investigate the performance of our tool against others that are available on the market. The most straightforward element of this, copyright and availability permitting, would be to re-use Java code examples used to demonstrate effectiveness of other tools so that documented results can be compared efficiently. More arbitrary cases would subsequently be addressed.

4 Available Options

4.1 Option 1

The first option is to develop a tool for lazy, systematic unit testing for the Java programming language. “Lazy systematic unit testing is based on the two notions of lazy specification, in which the intended specification of a class stabilizes gradually as the design for the class evolves, and systematic testing, in which the whole state-space of the object is exhaustively tested for conformance to the specification” ("JWalk :: Lazy Systematic Unit Testing") [5].

The tool will operate directly on the Java classes compiled code to explore all method protocols systematically. [6] In layman’s terms the tool will search all the available options and pathways to find any faults in the code. Once the tool has completed the test it will then print out an extensive test report. This tool will lend itself to test driven development as used in the Extreme programming approach to software development [7].

The testing tool will be loosely based on JWalk, however we will be stripping back some features such as dynamic analysis [5]. By eliminating some of the functionality the tool will be far more light weight and streamlined. With this in mind we aim to create a simple to use Java testing tool that won’t take up a lot of memory on your hard drive.

Much like how JWalk creates test scripts our tool will use a feedback-directed approach, first discovering the behavior of the prototype, producing reports of its behavior for manual assessment [5].

4.2 Option 2

The second option is to develop an automated testing and static analysis tool for the Java programming language.

The automated testing part of the tool will feature single unit testing. There will be a broad library of common tests for the user to select giving them a great template to start from. The tool then offers the ability to customize these ‘templates’ for a tailored test case to fit your specific code. From here you can save that custom template and recall it again anytime you need. This feature means that the tool is easy to get started with and quickly builds up your very own personalized high coverage test suit. [8][9][10]

This is a big difference from the way JWalk creates test cases. JWalk tools can be used right from the outset, as soon as a prototype class has been compiled. Simon said “The programmer follows a feedback-directed approach, first "exploring" the behavior of the prototype, generating reports of its behavior for manual inspection” [5].

Once testing is complete the tool will print out the test report. This report will provide a complete path for each possible fault detected in the IDE and links it straight to the code. Thus, allowing users to quickly jump to any point that is highlighted in the report. [11]

5 Recommended Option

5.1 Summary

After careful analysis of both option 1 and option 2 the group has decided to proceed and develop option 1 – a tool for lazy, systematic unit testing for the Java programming language. We have decided to remove the complete path analysis functionality because unfortunately it is beyond the scope of this tool. Removing this feature will also allow us to keep with the manifesto of creating a lightweight tool.

Table 1: Comparison between Option 1 and 2

	Option 1	Option 2
Estimated Duration	5 – 6 weeks	8 – 10
Difficulty	Medium	Hard
Programmer preference	100%	0%

5.2 Features

Features are the user tools that help them interact with the software function. Example: If a mobile phone is a function then the volume control, touch screen and power button are features [12].

Thung quoted "*Introducing new features, however, incurs cost. They need time to be developed. Testing and validation efforts are also needed to ensure that the new features work as intended. In addition, introducing new features might also introduce risks as old features that are previously working fine may now stop working due to incompatibilities with the new features. Such cost should be justified properly when resources are dedicated to develop the new features. The justification often comes when the new features are used by and benefit end users. For software libraries, a successful feature is one that is used widely by many client applications*." [13]

1. Select

   The Y-machine will feature a browser to select any compiled Java file the user wishes to test.

2. Choose, edit and save test case

   You can then select a test case from the dropdown menu which can be edited to fit your specific need. There will also be a no template option where by users can create a test case from scratch. All cases can be saved into the library (Original test templates cannot be overwritten however edited versions can be saved. Once the test case has been confirmed the tool will progress to a standby mode ready for the user to initiate testing.

3. Start test

   Once in the preliminary standby mode the user will click the start test button and the tool will run the selected test case.

4. Pause, continue, quit

   While in testing mode the user will have the option to pause, continue or quit testing which will prematurely end the test process and not submit a report.

5. End test

   Once the test has finished the tool returns the preliminary standby mode where by users can either run the test again or exit to Select stage.

6. Print results

   A new window will now pop up showing the test results allowing the user to administer the necessary changes in code.

7. Save results

The result print out will be able to be saved on to the user hard drive.

5.3 Expected Benefits

1. Lightweight

The tool is designed to be ‘light weight’ meaning a program, protocol, device, or anything that is relatively simpler or faster or that has fewer parts than something else. [14] The user can install the tool on their machine taking up as little memory as possible, or atleast allot more than more complicated systems such as JWalk. This makes the Y-Machine great for devices with limited memory such as tablets. These are sometimes refered to as ‘thin clients’.

2. Focused

Y-machine may be lacking in functionality compared to some other testing tools but what it lacks in functionality it makes up for by being a simple, easy to use tool that doesn’t require much training.

3. Templates

One of the great benefits of this tool is that it can be used ‘right out of the box’. The template library allows user to select a broad variety of test case templates and then be tailored to fit the particular code that is requiring said testing. These customised templates can then be saved and recalled to use again at the users discretion.

This feature could perhaps be broadened to allow for an online template achieve which would allow various users to upload their own custom templates with descriptions for others to use. This would increase the template library and could allow for a far greater choice whilst simultaneously permitting users to save time by selecting a more focused template without having to tailor a less specific template.

4. Save Results

Saving the results of the testing is obviously a high priority feature. This will be great for presenting reports on your coding, working in teams where by other developers can see the problems and troubleshooting later in the development stages.

5.4 Business Requirements

The business requirements for this project will be set based on meeting the needs of the user and understanding what they want. The table below lists the non-functional business requirements each to their own ID.

Table 2: Business requirements

Requirement ID	Business Requirement	Comments
BR-01	Project to be completed by 27th April 2017	Following project management plans will mitigate risk and failure of meeting the dateline
BR-02	The system is to be coded as Java application	
BR-03	The system needs to meet the users' needs to satisfactory standards, following the requirements set out in the project plan.	
BR-04	The system needs to accept test (compiled java file)	

6 Initial Requirements

6.1 Functional Requirements

Functional requirements refer to the requirement that the system should do. Typically, the functional requirement will specify the behavior of the system and the function of the product [15].

For this software, there are several main functional requirements that the software must be able to achieve. Firstly, the software must be able to select the file to be tested. The file here refers compiled Java file. The user will be able to select his or her own file to be tested.

1. Select file to be tested

   The user will be able to select any file he or she wishes to test. The file here refers to the compiled java file.

2. Start testing the compiled java file

   By clicking the start testing button, the program will be starting the testing process. The program will test the compile java file by using the X-Machine theory.

3. Pause the testing process

   Pause function allow the user to suspend the testing process. This will not stop the process completely. However, the process will only be suspended. This function is useful, for the users with limited RAM memory to have several running applications on the background.

4. Resume the testing process

   The resume function will allow the user to continue the testing process from the pause state back into testing state. Upon clicking the resume button, the system will continue the testing process.

5. Stop the testing process

   The stop function is a function that will completely stop the entire testing process. This function will not work with the resume button. Because the entire process will be stalled. If the user wishes to testing the same file again, he or she must start the process again. This function is used to stop and end the testing process.

6. Save result of test

   After the program, has successfully finished the testing process of the compile file. The result of the testing process will be save into a text file. The text file will contain all the errors found during the testing process. If the file has already existed, the user will be able to overwrite the file to a current version file.

7. Load the result

   This function would allow the user to load the result of a certain test and displaying it on the program. Therefore, there is no need for the user run the test all over again which would save time and resource.

8. Search or browse the result

   The program searches the result for a search term provided by the user. If there is a match, it will list all the related result with the key phrase.

6.2 Non-functional Requirements

Non-functional requirements cover all the remaining requirement which are not covered in functional requirement. The criteria to judge the operation of the system, the constraints of the system and the behaviours of the system. According to Kotonya and Sommerville non-functional requirements are divided into 3 categories; Product, Process and Externally related requirements [16].

Table 3: Categories and explanation of non-functional requirements

Process	**Product**	**External**
The system should adhere to the ISO standards; ISO 12207 (SDLC) in order to create the system	The system must be user-friendly, which will allow the user to get a grasp on the system without needing to read the manual.	Interoperability: The system must be able to save result into a text file.
	The system must be consistent. The user interface should be the same color, font size and others.	Legal: The system will follow the ethical law in which the system should not be used for wrong purposes.
	The system must be maintainable for future upgrade or when encountering new error and bug.	Economic: The system should not require any money to run or test the compile file.
	The system must be able to test the compile file latest by 5 minutes.	Security: The system must not reveal the confidential information from the compile file to the internet.
	The system must be able to handle larger compile file without worrying the RAM capacity.	
	The program must be runnable from the executable file. No need for user to use NetBeans to run the file.	
	The system must be able to quit without affecting the system and the data is not lost.	
	The system must be compatible with all windows operating system computer.	

	The system must be able to handle the stress test which involve running on low memory and / or low storage.	

7 Project Management

When embarking on a new project there can be unforeseen failure and risks which can be mitigated through the use of project management frameworks or strategies which will also increase the likelihood or level of success. Kerzner said "Project success has traditionally been defined as completing the requirements within the triple constraints of time, cost and scope (or performance)"[17] and there are various project management methodologies which can maintain control and foresee risk when passing the constraints of time, cost and scope.

First of all we should consider what defines a project, to enable a better understanding of the importance of project management. There are various ideas of what a project is and no single definition but Pinto [18] collects various definitions of a project to create a summary of a project being unique each time, with specific objectives, goals and specifications, with clear start and end dates whilst being restricted by parameters of cost, time and quality. Projects are also often, customer focused.

Barker & Cole specified a quick project health check, to ensure before proceeding that anything unclear or of risk can be addressed and also allow project management methodologies to be followed through successfully [19].

"1. Are the project's objectives clear and measurable?

2. Has anyone documented what the project needs to deliver - and have your customers signed up to this?

3. At first sight, do existing commitments to deliverables, timescales and resources look realistic?

4. If work is already well under way, is there a clear audit trail of significant decisions taken and underpinning assumptions?

5. Is your team working together productively and does everyone know what they're expected to deliver?" [19]

When considering the business case for a project, it's important for managers and the team to remain honest with one another and to themselves if any issues arise or they feel the project's benefits no longer exceed the project's cost/effort. Kerzner lists examples of when the business case causes project failure, including a faulty business case from the beginning, a change of stake holders - making the previous requirements and business case potentially invalid, new technologies or competitors beating your project and dominating your potential market etc. [17]. Many unforeseen instances can occur, out of the control of the project and its team, therefore it's important to compare and discuss the available project management methodologies to find those best suited for this type of project.

8 Software Development Life Cycle

8.1 Waterfall Model

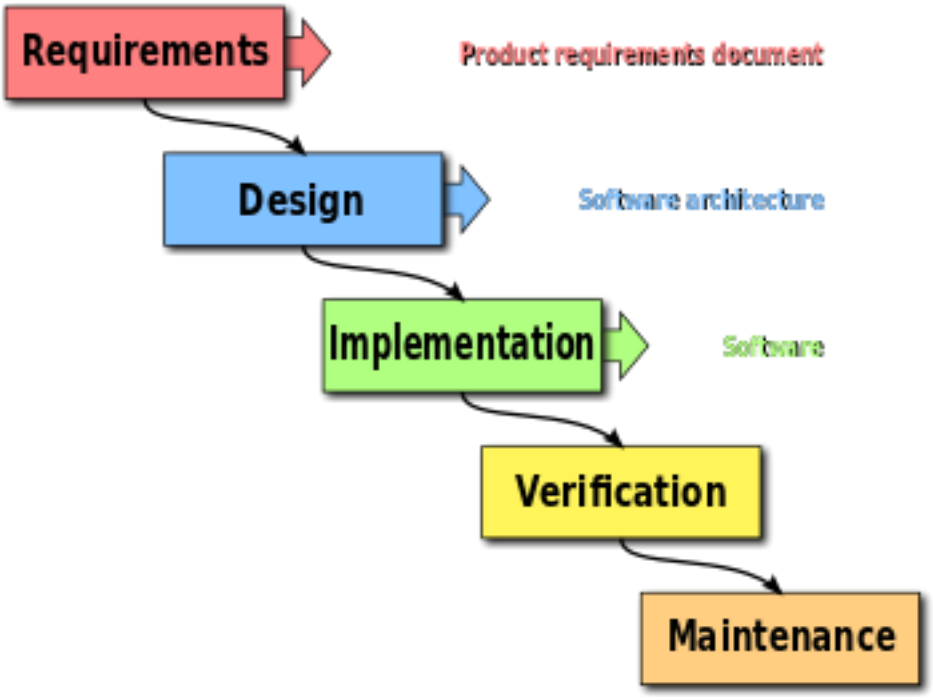

Figure 2: Waterfall Model (tutorialspoint,n.d.)

The Waterfall Model is a sequential software development methodology. Whilst the sequential element keeps tasks simple and ordered, by only moving on to the next task once the current one is completed for example, establishing requirements will be the first step and only once all requirements are established can the project move on to the next step of design. Whilst this model is clear and direct, it also comes with high risk due to the lack of testing which is encouraged throughout the model, in particular user-experience testing.

The Waterfall Model is better suited to small/short projects as the risk with this model is too high for a longer project, if the project is larger when using the Waterfall Model it is likely the project will be broken into iterations, with each iteration containing another project plan. Based on the non-functional business requirement of BR-01, the deadline for the project completion being April 27th 2017, this project will be too long to successfully complete with risks mitigated wherever possible, therefore another model will be more suitable.

8.2 V Model

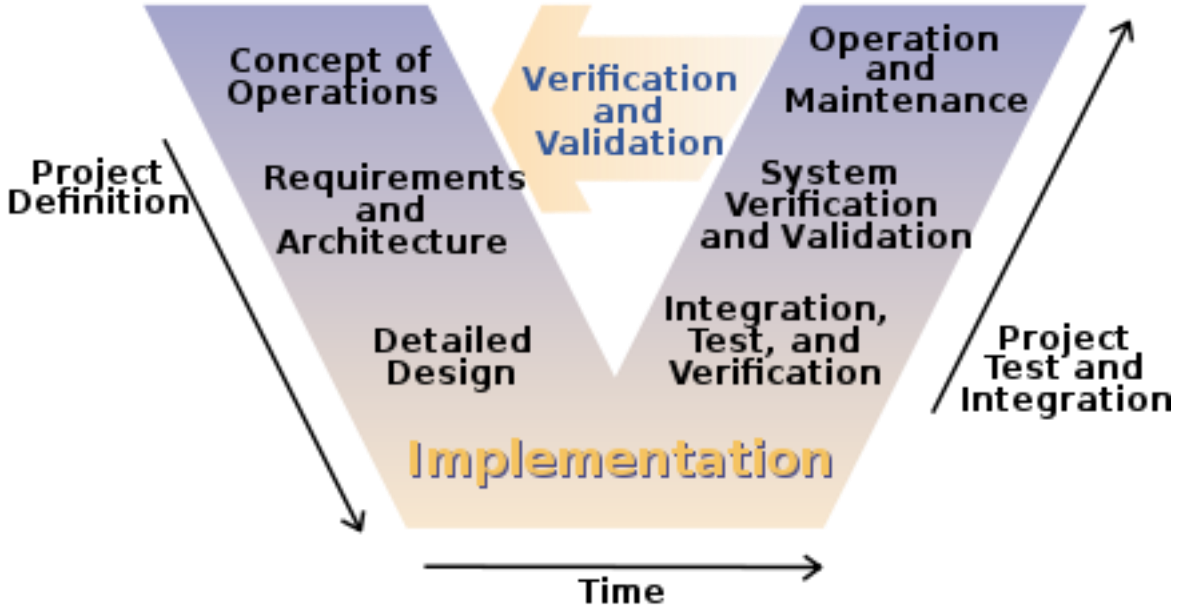

Figure 3: V Model (tutorialspoint, n.d.)

Similarly to the Waterfall Model, the V model has a sequential feel to it. However, unlike the Waterfall Model, the V model is designed for testing and building, the processes of the project in the V model are broken into iterations, parallel to one another, allowing for testing and validity alongside production, ensuring deliverables can be met.

For example, in the V-model, the upper left is the starting point of the V-model, this may state 'Business Needs', which encourages a set of requirements to be made alongside user stories. These can then be tested again with the user, in the parallel section, then validate and verify the business needs, acting as a confirmation to proceed to the next step. As the V-model progresses, each step will reflect the previous. The V-model may be better suited for this project, and certain develops from the Waterfall Model to ensure more verification and validation.

8.3 Agile Model

The Agile method is one of the most flexible software development lifecycle methodologies, but isn't necessarily suitable for every project. Agile typically suits projects where the end user may not necessarily know what they want, in the case of this project, we are seeking a game in the market of X-machines and looking for a way to implement this idea, therefore we haven't got a strict set of requirements from a client, whilst there will certainly be non-functional requirements, we still have scope on how we approach this project, making the Agile method a potentially good option.

Agile best suits projects with a constant variable but can cater for changing priorities, and works best with a cross-functional team.

The overall structure of Agile is to organise the project into a story, produce a backlog based on the story and create a set of tasks from the backlog. The story is the use-case, in this case the user may want to test their changing code for successes and failures, knowing this use-case we can form a set of requirements (the software must do this etc.) which forms the backlog, and then perform tasks to achieve the backlog, to achieve the story.

The backlog production is organised into a sprint, often a series of mini projects completed by the cross-functional team, lasting a minimum of one week and a maximum of four weeks. The cross-functional team makes up the scrum, who work together on that particular sprint. Next the team moves onto sprint planning, in which the backlog is validated by the end-users for the first part of the day, in the latter part, the scrum prioritise the backlog into most likely to be achieved. Agile requires regular scrum meetings, which ensures clear communication is made about where the project is at and seek help with any challenges.

Agile then takes the team onto the sprint review, where the end product is demonstrated to the users and the achieved backlog is displayed. Finally, a sprint retrospective occurs, in which the successes and failures of the sprint are discussed, forming a new backlog of any requirements not achieved in the previous sprint.

Agile also uses a visual method to monitor progress, through a Burndown chart, which shows the Effort (number of hours), the number of days and the number of tasks in comparison to one another, which allows clear communication on the progress of the project. Due to the length of this project, and having a cross-functional team of five, Agile appears to have the highest risk mitigation most suitable organisation for the complexity of the project.

8.4 Conclusion

Based on the analysis of several software development lifecycle methodologies, Agile will be the most appropriate, catering for a cross functional team and allowing a new sprint to be made upon completion of each task from the user story. It will allow for good communication amongst the team to discuss any issues which arose in any sprint to bear in mind for the next or even allow for early reconsideration of any changes to the project if we equally become concerned of a risk. Alongside this for extra assurance and planning, a Gantt chart can be used to monitor milestones with time remaining, and the v-model could also be referred to from time to time to allow for testing alongside production, again to mitigate any risks and ensure end user requirements are met.

9 Solution Outline

9.1 Use Case Diagram

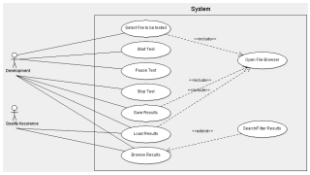

Figure 4: Use Case of the expected system

9.2 Use Case Specification

Table 4: Use Case Specification Number 1

Use Case	**Reference:** UC-01
Name	Select File To Be Tested
Actor	Development
Description	The user will be asked to select a file of compiled Java that they wish to test for errors
Normal Course	1. User clicks “Select File” 2. System opens a file browser 3. User browses file system for the file they wish to test 4. User selects file 5. System checks that the file is a valid file 6. System opens file for testing 7. System provides feedback of success
Alternate Course	4. 4.1 User presses cancel instead of selecting file 4.2 System closes file browser 5. 5.1 System determines that the file is not a properly formatted compiled Java file 5.2 System provides feedback advising an incorrectly formatted file 5.3 System returns to file browser window for user to select again
Pre-Conditions	• None
Post-Conditions	• File is opened for testing
Assumptions	• None

Table 5: Use Case Specification Number 2

Use Case	**Reference:** UC-02
Name	Start Testing
Actor	Development
Description	The system starts the testing cycle
Normal Course	1. User clicks “Start Test” 2. System checks to see if a test is already under way but is paused 3. System starts testing file 4. When System reaches end of testing provides feedback to the user and displays results
Alternate Course	2. 2.1 System determines that a test is under way but is paused 2.2 System continues current test from the point it was paused

	3. 3.1 3.1.1 User clicks “Pause Testing” 3.1.2 System Pauses Testing 3.1.3 System Provides feedback that testing is paused 3.2 3.2.1 User clicks “Stop Testing” 3.2.2 System halts the test 3.2.3 System provides feedback that the test has been stopped 3.2.4 System displays results
Pre-Conditions	• File has been selected for testing
Post-Conditions	• Test results for testing that has been completed are displayed
Assumptions	• None

Table 6: Use Case Specification Number 3

Use Case	**Reference:** UC-03
Name	Pause Testing
Actor	Development
Description	The system pauses the testing cycle
Normal Course	1. User clicks “Pause Test” 2. System checks to see if a test is already under way but is paused 3. System pauses testing file 4. System provides feedback that testing is paused
Alternate Course	2. 2.1 System determines that a test is under way but is paused 2.2 System provides feedback that the testing is already paused and to click “Start Test” to restart testing
Pre-Conditions	• Test is already underway
Post-Conditions	• Test is paused
Assumptions	• None

Table 7: Use Case Specification Number 4

Use Case	**Reference:** UC-04
Name	Stop Testing
Actor	Development
Description	The system stops the testing cycle
Normal Course	1. User clicks “Stop Test” 2. System checks to see if a test is already under way

	3. System opens a dialog box “Are you sure?” 4. User clicks OK 5. System stops the testing 6. System displays results of tests completed
Alternate Course	2. 2.1 System determines that no test is under way 2.2 System provides feedback that no testing is under way 4. 4.1 User clicks “Cancel” 4.2 System continues testing
Pre-Conditions	• Testing has been started
Post-Conditions	• Test results for testing that has been completed are displayed
Assumptions	• None

Table 8: Use Case Specification Number 5

Use Case	**Reference:** UC-05
Name	Save Results
Actor	Development
Description	The System saves the results of a test cycle
Normal Course	1. User clicks “Save Results” 2. System checks to see if results are available 3. System opens file browser 4. User browses file system 5. User selects location and types file name 6. System saves results and provides feedback
Alternate Course	6. 6.1 System determines that a file already exists with that file name 6.2 System displays dialog “File already exists, overwrite?” 6.2.1 6.2.1.1 User selects “Overwrite” 6.2.1.2 System overwrites existing file 6.2.2 6.2.2.1 User selects “Cancel” 6.2.2.2 System returns to file browser 6.2.2.3 User enters a new file name
Pre-Conditions	• Testing has been completed
Post-Conditions	• Test results for testing that has been completed are saved
Assumptions	• None

Table 9: Use Case Specification Number 6

Use Case	**Reference:** UC-06
Name	Load Results
Actor	Development & Quality Assurance
Description	The system opens a file of saved results
Normal Course	1. User clicks “Open results” 2. System opens file browser 3. User browses file system 4. User selects file to open 5. System checks if file is properly formatted 6. System opens file and displays results
Alternate Course	5. 5.1 System determines that file selected is not properly formatted 5.2 System provides feedback and returns to file browser

Pre-Conditions	• User has saved data previously
Post-Conditions	• Test results that are loaded are displayed
Assumptions	• None

Table 10: Use Case Specification Number 7

Use Case	**Reference:** UC-07
Name	Search results
Actor	Development & Quality Assurance
Description	The system searches results for a search term provided by the user
Normal Course	1. User enters a phrase to search for in search box 2. User clicks “Search” 3. System searches for test results containing phrase 4. System displays only the test results containing phrase 5. User clears search box and clicks search to clear search 6. System displays all test results
Alternate Course	4. 4.1 System determines there are no test results containing search phrase 4.2 System provides feedback that there are no results with that search phrase 4.3 System displays all test results
Pre-Conditions	• Results are available to search
Post-Conditions	• Test results that match search phrase are displayed
Assumptions	• None

9.3 Use Cases / User Stories

A user story is part of an Agile methodology approach, it is the starting of communication between the team working on the product and the product owner, with the aim of meeting requirements from an end-user perspective.

A user story establishes how the user will interact with the software in order to complete the story, this requires following two structures of Role, Goal and Benefit, alongside the acronym, INVEST (Independent, Negotiable, Valuable, Estimable, Small and Testable.)

When considering Role, Goal and Benefit, we must take the attitude of the user, understand their goal and which benefit they hope to achieve. The following table can display this.

Epic	Stories	Tasks
As a user I want to have automated testing of Java to detect errors, vulnerabilities and unexecutable code	As a user I want code coverage to detect unexecuted code	Add font colour for executed code
		Add font colour for unexecuted code
	As a user I want to import existing Java files	Add an import option/button
		Add a browse files button/menu
	As a user I want to have tests automatically generated	Create code for software to automatically generate tests where needed, and send feedback to the user
	As a user I want to receive code coverage to know which part of code has been/ is being covered	Create a visual response in the interface to communicate to the user which code is being covered
	As a user I want unit tests generated for different methods	Code the software to recognise particular methods and conduct the relevant tests
	As a user I want to be able to add my own generated tests for scenarios the software may not be aware of	Alert the user of a method or test element it does not recognise which requires testing
		Allow user to create and run their own tests
	As a user I want to run tests in isolation of one another	Ensure functionality of system allows for isolated testing

Figure 5: Agile Stories and Task

The above user story shows the epic, with the individual stories the user may wish to achieve, with the tasks required to complete them. The running benefit of the stories relates back to the epic, a user being able to have automated testing of their Java code to detect errors, vulnerabilities and un-executable code, having automated testing of this from a software saves time and effort for the developers/team work on the project as they can focus their effort in other areas where time and efficiency is made from the benefits of the stories.

This also follows the invest acronym:

Table 11: INVEST Acronyms [20]

INVEST	
I - Independent	Each of the user stories are independent from one another, whilst they may still compliment one another.
N - Negotiable	Each of the user stories are negotiable, they could be adapted if need be, or may be able to be reconsidered for their essentiality in the backlog.
V - Valuable	Each of the stories can be valuable to the end user, they all offer a feature an end user can benefit from.
E - Estimable	It is possible to estimate how long each story task will take to complete and therefore can be implemented into an Agile strategy or Gantt chart.
S - Small	Each story is small, ensuring it won't exceed a sensible time to complete and each iteration can be monitored efficiently.

9.4 Wireframes / Screen Concepts

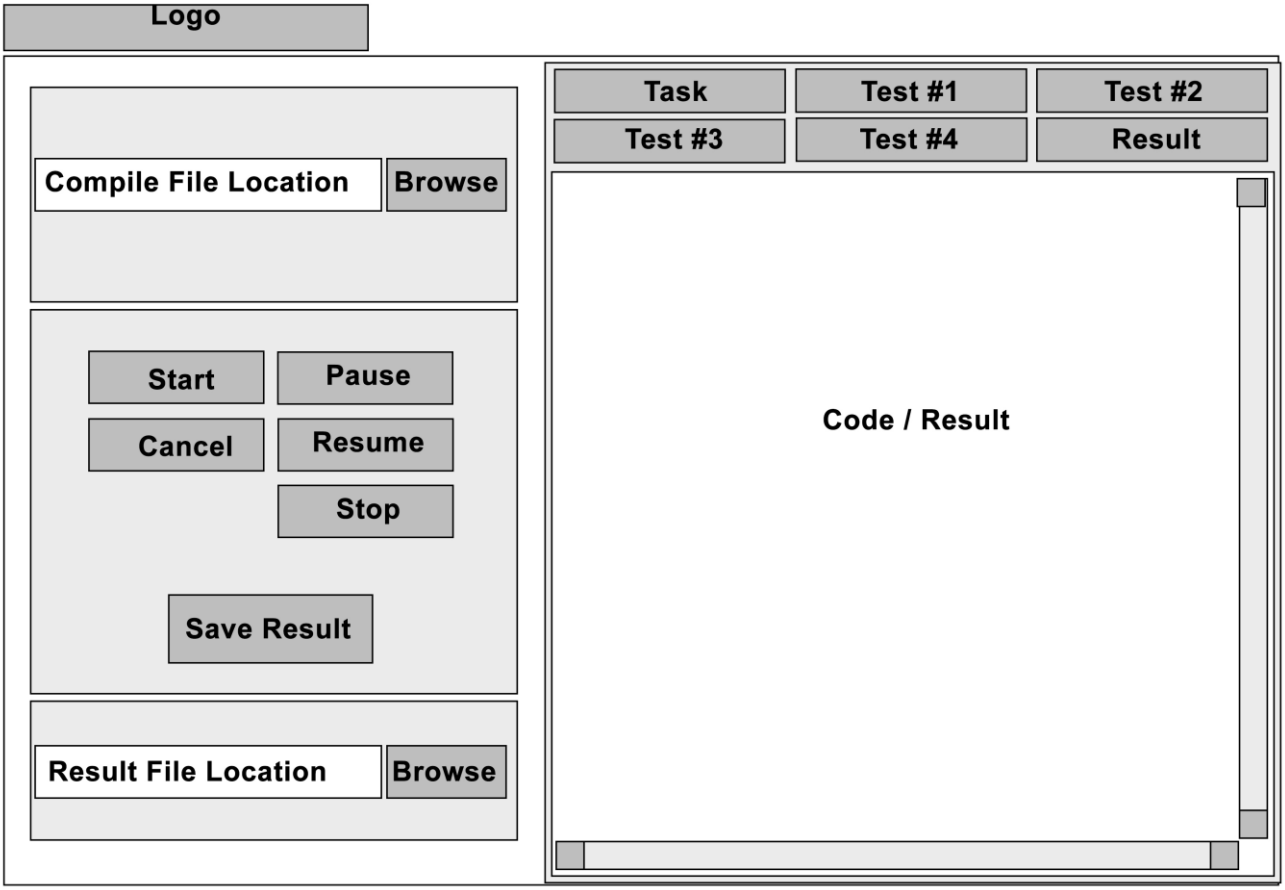

Figure 6: Expected UI result of the system

10 Implementation Approach

10.1 Project Execution

10.1.1 Tools to develop program

Development Tools

There are many softwares that allows the user to download and develop their application on. In this project, it has stated that the software will be developed in Java. So to make it easier, the team has decided to compare the best Java software development platform. For example of the most common ones for a Java application would be the Eclipse IDE and Netbeans IDE.

Eclipse is an integrated development environment (IDE) is that is widely used in Java programming language [21]. Making its primary used to develop Java based applications. However, it may also be used to develop application in other programming language such as C and C++. Eclipse also contains extensible plugin which allows the user to customize the environment based on the needs and requirements.

Netbeans is another software development platform written mostly in Java. Like, Eclipse Netbeans too allow the developer to code in other languages other than Java. Netbeans is also a cross-platform which allows the user to run in on Windows, Mac OSX and Linux.

There are several factors that the team has decided to go for the Netbeans software application rather than Eclipse. The first reason would be the experience with each development software. In our group, majority of the team members have tried and programmed on the Netbeans. Therefore, the overall user interface and functions are quite familiar with the team members. Since, we would take a longer investment of time to get familiar with Eclipse.

Secondly, Netbeans have a wider variety of built in plugins over Eclipse. For example, the SWING GUI features, MySQL and etc. Due to this reason, we have a variety of plugins to include in the develop software. With wider varieties, it will allow us to develop the software with many options and such which option is the best suited in our scenario. The plugins does not only apply for Java, there is also plugin which aids the design of UML diagram.

Thirdly, the debugger and the profiler in Netbeans. The Netbeans debugger us to place breakpoints in the source code and using this we could take snapshots and monitor the execution process as it occurs [22]. The IDE includes a visual debugger to let us take GUI snapshot which could be use and in our implementation phase. While the profiler provides assistance for optimizing the develop software (Y-Machine) speeds and memory usage.

UML Tools

UML or unified modeling language is a modelling language used in software engineering field that provides the standard way to visualize the design and interactions of a system. There a many types of UML diagrams such as the use case diagram, class diagram, sequence diagram and many more. Each of these diagrams has their own intended purpose.

The use case diagram commonly used to display the interaction between the system and the users. Within the use case diagram, there are many uses cases which will help identify the different types of users of the system. There are also feature of use like the include and extend functions.[23] The include help distinguish and categorize the common behaviour of one use with another while the extends is a choice or option, the actors has the option to choose which interaction he/she wishes to go for.

A class diagram is a static structure diagram that describes the structure of a system by showing the system classes. Within each classes, there will be attributes (names), operations (methods or functions) and the relationship amongst the object. The relationship could be zero or many or 1 to 1 and many more. [24]

These are some of the diagrams, that will be developed throughout the project. The diagrams will be used to describe and aid the development stage of the software. There are several UML tools that can be used to design all these diagram such as the Visual Paradigm, ArgoUML and IBM Rhapsody. Since the diagram will be use in order to develop the system. Each of these UML tools have their own advantages over the other.

For example, in IBM Rhapsody the user is also able to code the functions of the system without actually going into the software development tools. However, IBM Rhapsody are only available in the campus, making it harder for us to use. It is the best option for us however, due the reason that it is only available in the campus is restricting our resource. Also, IBM Rhapsody is not free to use it is expensive to buy. The next best tools, would be the ArgoUML,which allows us to design the UML and its is also Java compatible which is an advantage for us. Another reason would be it is open source, therefore it is free to use in our personal computer.

Project Management Tools

Project management is application of processes, method, knowledge, skills and experience to achieve the project objectives. A project is usually deemed to be a success if it achieves the requirements and the objectives according to the criteria, within the budget and time. Which can also be classified as the resources [25].

One of the project management tools would be the Gantt chart. The Gantt chart is a specialized bar chart which illustrate the overview and the schedule of all the task to indicate the work elements and the dependencies of the tasks or project [25]. These elements comprise the work breakdown structure (WBS) of the project. There are many advantage of doing a Gantt chart. Firstly, it help the team to organize and divide the task easily. The tasks can also be assigned to one or members in order to track the progress without affecting one or another. Secondly, it help the team to set a realistic time frame. This can help us to get things in perspective properly.

There are many options of Gantt chart tools in the web. Some example would be Microsoft project and "TeamGantt". The functionality of these 2 application is almost identical. For example, both allow the user to create Gantt chart with ease, simple user interface allow feature like start to finish and many others [26]. The group has decided to go for the "TeamGantt" because of this reason. Unlike the other Gantt chart tools, TeamGantt is an online tools, which makes it easier to use on multiple computers without having to worry on the installation of software or even the operating system. It also allows multiple user to view and edit the gantt chart at the same time. Lastly, TeamGantt provide a fast customer feedback with their customer, therefore we can message the employee regarding some issues. The employee would then try to help us solve the problem regard the GanttChart planning.

Design Tools

In the later section of this proposal, we would need to create the wireframes or mock user interface for the system. To design this wireframe, one would need to use a photo editing and manipulation software. There are built in tools, like in Windows paint.

Advantages of using paint, it is simple to use and yet functional for editing image. Beside that, the processing time or time taken to save the file is short. Due to this reason the output of the file is small. The disadvantages of using paint, it is seems unprofessional to edit image by using a proprietary software. Also, there are certain part that part are lacking, for example the usage of layer properties. In paint, we are not able to lay the image on top of another without affecting the entire canvas.

Advantage of using Photoshop, there are lot more features such layer arrangement. For example, we can easily change the layout only without affecting the other component. Secondly, to edit or touch up certain part is easy, there is no need to create the picture from scratch. Disadvantages of using Photoshop, the time taken to learn the software is longer and the user interface is fairly complex compared to Paint due to the extra features.

The tools that we have decided to use would be Photoshop. Despite the disadvantages mentioned about the time take to learn and getting familiar with the software. One of our member, has been using Photoshop for awhile. Therefore, she has the knowledge to operate and design the wireframes for the system without needing the extra time to learn and this extra time, we could use the time (resource) on the implementation.

10.1.2 Comparison with existing tools

First of all, we need to define what is testing, and why developers do testing in programming, and what tools developers use to help them in testing their work?

“The process of operating a system or component under specified condition, observing or recording the results, and making an evaluation of some aspect of the system or component.” (IEEE Standard 610.12-1990, “IEEE Standard Glossary of Software Engineering Terminology”) [27]

The purpose of testing is to show whether the software works or it does not work, or, to detect errors, bugs and to check that the functionality of an application does met with requirements. So, when a human is not used for testing a particular module, then it means that we are using a tool for testing and when we use any tool for testing a module (software) then it is known as automated testing. Testing tools facilitate a software tester in different conditions where testing becomes critical.

There are many testing tools that could be used in testing, but in here, we focus on some of the well-known and used testing tools as follow: -

1. JUnit

JUnit is one of unit testing frameworks which is known as xUnit [28]. JUnit is an open source framework, which is used for writing and running tests for the Java programming language. It helps automate the process of code testing. JUnit is easy to use and provides features to perform a single unit test as well as grouping a series of unit tests together into a single suite. With unit testing we divide code up into its component parts(units). Each unit test tells a story of how that part of the program should do in a specific scenario.

To test java unit using JUnit, the tester must first write a test program. To establish a connection between the java unit to be tested and the Junit framework. Actual test use the assertEquals method, where an assertion contains the pass/fail result of executing the called unit with test case values. For example,

The assertion assertEquals (true, VaidDate.validDate (29, 2, 2000));

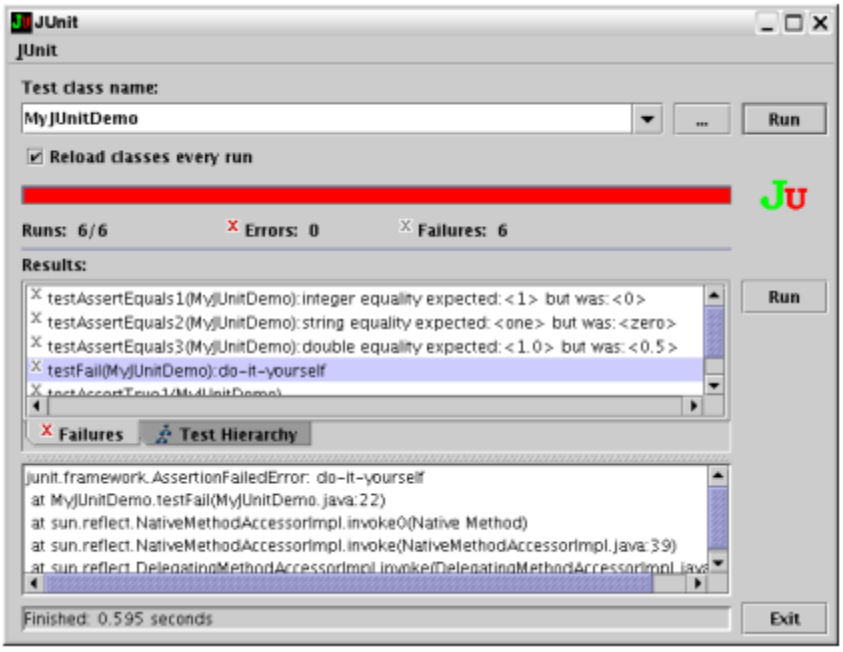

Figure 7: JUnit UI (JUnit, n.d.)

2. JWalk

JWalk is a unit testing toolkit for lazy, systematic unit testing [29]. It is a free to use testing tool, under a specific terms by its provider. It was developed to classify the systematic unit testing needs, and it's adopting the context of agile methods. JWalk operates directly on the Java class's compiled code. JWalk tool can be used straightforward after the prototype class was compiled. The programmer follows a feedback-directed random test generation approach, first examining the prototype behavior to generate reports of that prototype behavior, for the process of inspection manually. If the behavior varies from the program expectations, so tester will decide whether to make changes to the code or just leave it as it is. Then, with a limited user-interaction and with a dynamic analysis, the programmer will validate the test class. Eventually, the programmer should confirm or discard the key test results. Currently, there are two versions of the JWalk tool, one as a command-line utility program, which generate test reports on standard output, and the other one as an API that can be integrated with third-party programs, using Java event-model to communicate between JWalk and the third-party user interface.

Features of JWalk

The advantage of JWalk comparing to other testing tools, is that the tool can be directed to systematically explore all method protocols, and it can print a long test report. It provide static analysis, and perform automated testing according to a state-based model of the class under testing. It explore and validate all interleaved methods to a given path depth. JWalk automatically conclude the test cases that the developer needs to provide. The tool can automatically, build and display these test cases, which will benefit the programmer of saving his time and effort. When the test class is extended or modified by other subclasses, JWalk generate new required test cases for all novel interleaved combinations of local and inherited methods. JWalk provides the full power of specification-based conformance testing for programmers who don't want to write formal specifications! The following is a graphical example of JWalk tool:-

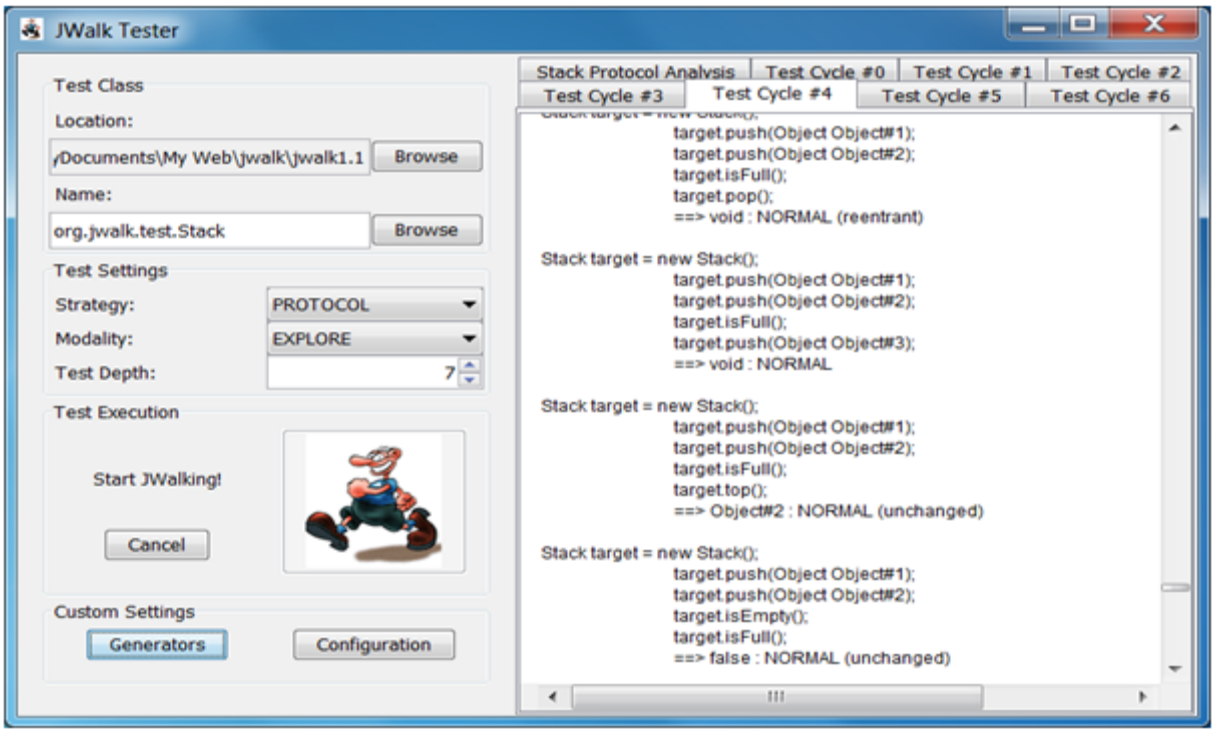

Figure 8: JWalk UI (JWalk, n.d.)

Table 12: Comparison between JUnit and JWalk

JUnit	JWalk
The JUnit tester has to devise suitable tests by hand, which is hard and error-prone.	The JWalk tools propose all the significant test cases systematically.
Time-consuming and tedious – if test cases are executed by human resources, it is very slow and tedious.	The tools construct and present these test cases automatically, saving the programmer time and effort.
The JUnit tool merely automates the repeated execution of the tests.	The JWalk tools generate new tests after a class's behaviour has changed.
The saved regression tests cannot exercise new behaviour introduced later.	The JWalk tools predict the test outcomes for many more test-cases.
Provides annotations to identify test methods.	JWalk tools take the effort of thinking up the right test cases away from the tester.
Less investment in human resources − Test cases are executed using automation tools, so less number of testers are required in automation testing.	

3. Arquillian

Arquillian is an integration testing framework for Java EE. It allows java developers easily to create automated integration, functional and acceptance tests. Instead of managing a runtime in the test, Arquillian brings the test to the runtime. Arquillian provides a component model for integration tests, which includes dependency injection and container life cycle management.

It is also capable of deploying archive into containers and execute tests in the containers and capture results and create reports. Arquillian integrates with familiar testing frameworks such as JUnit4, TestNG5. It allows tests to be launched using existing IDE, and because of its modular design it is capable of running Ant and Maven test plugins.

Features of Arquillian

It manages the lifecycle of the container (start/stop), and bundle the test class with dependent classes and resources into a deployable archive. It picks up where unit tests leave off, targeting the integration of application code inside a real runtime environment. It captures results and failures. With Arquillian, developer writes a basic test case and annotates it with declarative behavior that says, "run with Arquillian."

Arquillian, zero reliance upon a formal build; can be run or debugged from IDEs like Eclipse, IDEA, NetBeans. It supports remote and embedded containers: JBoss AS, GlassFish, Jetty, Tomcat, OpenEJB, OSGi and more on the way.

4. JTest

JTest is an automated Java software testing and static analysis tool made by Parasoft. It focuses on practices for validating Java code and applications and it seamlessly integrates with Parasoft SOAtest to enable end-to-end functional and load testing of today's complex, distributed applications and transactions. JTest also provides a complete path for each potential defect in the IDE and cross-links it to the code, enabling users to quickly jump to any point in the highlighted analysis path. JTest includes functionality for Unit test-case generation and execution, static code analysis, data flow static analysis, and metrics analysis, regression testing, run-time error detection.

Features of JTest

JTest allows developer to peer code review process automation and run-time error detection for e.g.: Race conditions, exceptions, resource and memory leaks, security attack vulnerabilities.

It automatically generate complete tests, including test drivers and test cases for individual functions, and use them for initial validation of the code’s functional behavior. Test helps development teams produce better code, test it more efficiently, and consistently monitor progress toward quality goals.

5. The Grinder

The Grinder' is a Java load testing framework that was designed to make sure it was easy to run a distributed test's using many load injector machines. It can run tests in parallel on multiple machines, allowing developer to check how your application would behave under heavy load.

It can be used to test anything with a Java API, which includes SOAP and REST services, HTTP web servers, and application servers such as CORBA, RMI, JMS, and EJBs. The Grinder 3 allows any code (Java, Jython, or Clojure) code to be encapsulated as a test. Java libraries available for an enormous variety of systems and protocols, and they can all be exercised using The Grinder. The testing scripts in Grinder are built in the powerful scripting language Jython which makes scripting flexible and uses the HTTPClient library.

Features of The Grinder

Grinder provides a console that can be used as an IDE. The GUI console for The Grinder allows you to have multiple load injectors to be monitored and controlled and Automatic management of client connections and cookies, SSL, Proxy aware and Connection throttling.Tests are monitored and controlled via the graphical console that comes in four languages English, German, French and Spanish.The Grinder 3 allows any code (Java, Jython, or Clojure) code to be encapsulated as a test.

6. TestNG Tool

TestNG is a testing framework designed for the Java programming language and inspired by JUnit and NUnit.TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc. It is supported, out-of-the-box or via plug-ins, by each of the three major Java IDEs - Eclipse, IntelliJ IDEA, and NetBeans. It also introduced some new functionality that make it more powerful and easier to use, such as: Annotations, Running tests in big thread pools with various policies available, code testing in a multi thread safe, flexible test configurations, data-driven testing support for parameters, and more.

Features of TestNG

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use, such as annotations, test that developer code is multithread safe, and Flexible test configuration. TestNG support for data-driven testing (with @DataProvider), and parameters. It also embeds BeanShell for further flexibility. It's default JDK functions for runtime and logging (no dependencies).It allows developers to run their tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...).TestNG has a more elegant way of handling parameterized tests with the data-provider concept. Using TestNG framework allows us to generate test reports in both HTML and XML formats. Using ANT with TestNG, we can generate primitive Testing reports as well. Also in TestNG there is no constraint like you have to declare @BeforeClass and @AfterClass, which is present in JUnit.

10.2 Project Plan

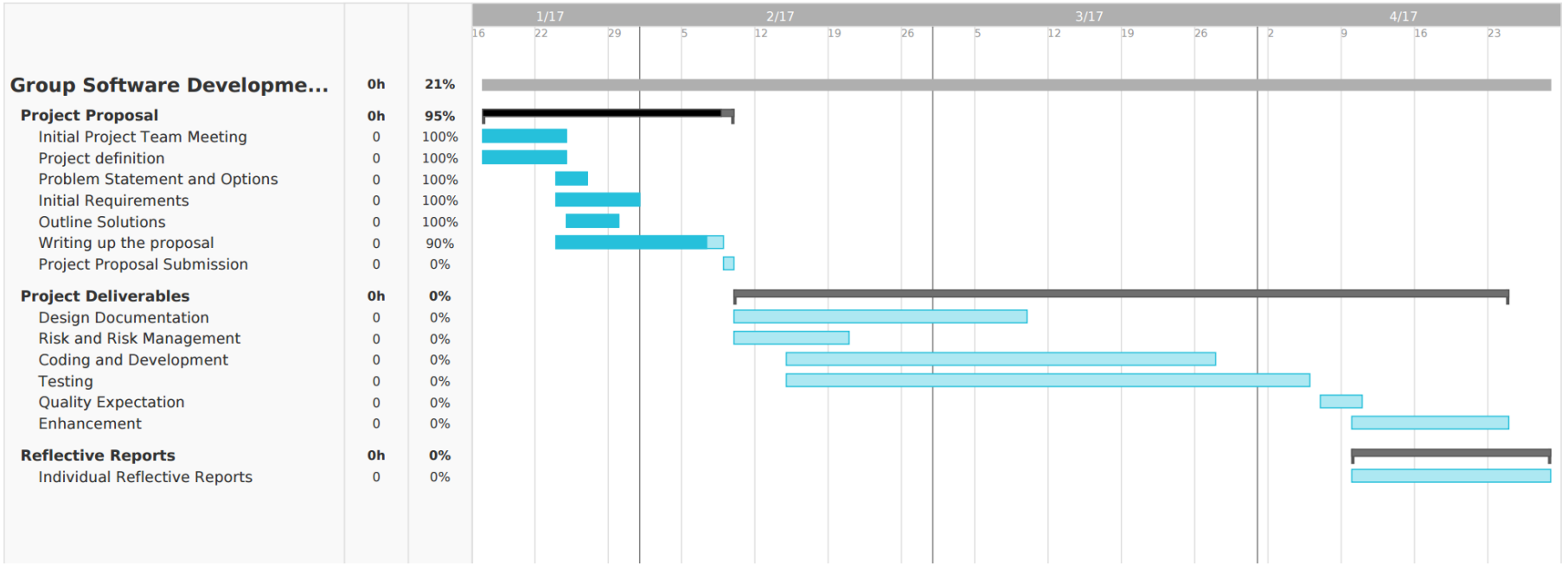

Figure 9: Gantt Chart for the group tasks

10.3 Quality Expectations

The system will need to produced following a set of requirements and planning stages to ensure the expectations are met. Progress will be documented, assessing the development processes to the criteria set by the client. As the system is for a client, it is important to refer back to BR-01, with the project completion due on the 27th of April 2017, for the project to meet its standard it's important the deadline is met with the work to a satisfactory standard.

To ensure quality of the system, testing will be implemented throughout production along with testing of the final product, to create a report of any bugs/errors discovered and how they were fixed. This will be useful for any future developers working on the project in case similar errors occur in future modifications of the system.

10.4 Project Risk

10.4.1 Introduction

Risk can come from both external and internal sources. The importance of Risk management, is that, without it, a project cannot possibly define its objectives for the future, and also, trying to avoid or minimizing any risks might affect the final product of the project, and to identify, assess and control uncertainties in terms of threats and opportunities.

10.4.2 Identified risk

So, in this project, the new testing tool may fail to work probably, or maybe not work at all, or even, not to be completed to be a recognised testing tool. Because of so many reasons as follow: -

Skills Limitation.

There is a main risk of skills limitation in our team, as most of the team members have little or no programming experience. The impact is huge and has the strong potential risk to affect achieving the project main goal of producing the testing machine tool, which indeed needs at least 2 to 3 good programmers to achieve it in a good time and at best quality.

Other responsibilities of team members·

As all of team members got other study areas, and different other work to be accomplished alongside with this assignment, this have also a great potential risk of affecting the project goals to be met.

Limited time.

We think that the time we have to produce the testing machine is not enough. As mentioned above, the programming skills of most of team members are limited and could be not enough to finish that tool in the desired time. And also, other responsibilities of team members might cause a big delay in finishing the tool in the needed time.

Limitations of technologies knowledge.

As we need to produce a new software testing tool in corresponding to the x-Machine principals with java programming language. And as mentioned before that the team programming skills and knowledge are not strong enough, so, we not sure, which technologies that might be best, to use for producing the software testing tool.

10.4.3 Risk control plans

So, we did agree that some of the team members to focus on improving their programming knowledge in java by learning it more and, do lots of practice on it. Also, we might ask for help from other expertise of different groups, just as guidance! And the rest of our team members, to focus on finishing the other research areas related to project. Time planed between team members to different specific tasks and person's roles. More research on best technologies to be used will take place

11 Conclusion

In conclusion, we have completed the business proposal qualifying our system for consideration for the project and funding. We have shown our technical knowledge and demonstrated how our Java testing tool will be the gap in the market that developers would find valuable.

The Y-Machine is a niche testing tool for systematically testing vital systems with limited memory.

We have factored in time, cost, scope, effort and team's various expertise into the project plan and have concluded that we will be able to deliver a quality product on time and on budget.

We now leave you with the Projects manifesto:

"*The 'Y-Machine' is a lightweight, focused, and intuitive tool to allow systematic testing on systems that have limited memory."*

12 Appendix

Table 13: Tasks break down

Tasks/Names	Muhammad Amirul Haziq Musa	Andrea Chitty	Dewey Roskilly	Magda Mosy	Nicola Hogg
Executive Summary			✔		
Background		✔			
Introduction		✔			
Finite State Machine		✔			
X-Machine		✔			
Problem Statement			✔		
Available Options			✔		
Option 1			✔		
Option 2			✔		
Recommended Option			✔		
Summary			✔		
Features			✔		

Expected Benefits			✔		
Business Requirements					✔
Initial Requirements	✔				
Functional Requirements	✔				
Non-functional Requirements	✔				
Project Management	✔				✔
Software Development Life Cycle					✔
Waterfall Model					✔
V Model					✔
Agile Model	✔				✔
Conclusion	✔				✔
Solution Outline		✔			✔
Use Case Diagram		✔			
Use Case Specification		✔			
Use Cases / User Stories		✔			
Wireframes / Screen Concepts	✔				✔

Implementation Approach	✔			✔	
Project Execution	✔			✔	
Tools to develop program	✔				
Comparison with existing tools				✔	
Project Plan	✔				
Quality Expectations				✔	
Project Risk				✔	
Introduction				✔	
Identified risk				✔	
Risk control plans				✔	
Conclusion					✔
References	✔				
Compiling the document	✔				
Research on Finite State Machine	✔	✔	✔	✔	✔
Research on X-Machine	✔	✔	✔	✔	✔
Research on JWalk	✔		✔	✔	✔

Research on existing tools				✔	✔
Doing the presentation slides	✔	✔	✔	✔	✔

13 References

[1] Radatz, J. (1990) *IEEE Standard Glossary of Software Engineering Terminology* [online]. Available from: IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990), IEEE Computer Soc., Dec. 10, 1990. Terminology standard.

[2] Kushwaha, D. and Misra, A. (2008) *Software test effort estimation. ACM SIGSOFT Software Engineering Notes* [online]. 33 (3), pp.1-5.

[3] Myers, G.J., Badgett, T. and Sandler, C. (2012) *The Art of Software Testing* [online]. 3rd ed. Hoboken, N.J: Wiley.

[4] Tahbildar, H., Borbora, P. and Khataniar, G.P. (2013) *TEACHING AUTOMATED TEST DATA GENERATION TOOLS FOR C, C++ , AND JAVA PROGRAMS. International Journal of Computer Science & Information Technology* [online]. 5 (1), pp.181-195.

[5] Simons, AJH. "The JWalk Home Page". The JWalk Home Page. N.p., 2013. Web. 1 Feb. 2017.

[6] Simons, Anthony J. H. JWalk: A Tool For Lazy, Systematic Testing Of Java Classes By Design Introspection And User Interaction©. 2017. Web. 1 Feb. 2017.

[7] Hutagalung, W. "Extreme Programming". http://www.umsl.edu/. N.p., 2006. Web. 5 Feb. 2017.

[8] Jtest - Parasoft's Automated Java Testing Tool". Parasoft. N.p., 2017. Web. 5 Feb. 2017.

[9] Bell, Jason. "JDJ Product Review — Parasoft Jtest 8.0". http://java.sys-con.com/. N.p., 2006. Web. 5 Feb. 2017.

[10] "Chapter 7. Testing Using Junit". Drjava.org. N.p., 2017. Web. 6 Feb. 2017.

[11] "Integrated Error-Detection Techniques: Find More Bugs In Java Applications". cdn2.hubspot.net. N.p., 2017. Web. 5 Feb. 2017.

[12]Functions And Features". Inclusive.com. N.p., 2017. Web. 6 Feb. 2017.

[13] F. Thung, D. Lo and L. Jiang, "Diffusion of Software Features: An Exploratory Study," *2012 19th Asia-Pacific Software Engineering Conference*, Hong Kong, 2012, pp. 368-373. doi: 10.1109/APSEC.2012.139
keywords: {Java;public domain software;software libraries;JDK;Java development kit library;commercial software systems;library developers;library managers;open source software systems;software feature diffusion;software libraries;software products;Communities;Feature extraction;Java;Software;Software libraries;XML;Diffusion;Empirical Software Engineering;Exploratory Study},
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6462682&isnumber=6462577

[14] "What Is Lightweight? - Definition From Whatis.Com". WhatIs.com. N.p., 2017. Web. 7 Feb. 2017.

[15] Eriksson, U. (2012) Functional requirements vs non functional requirements. Available at: http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

[16] Kotonya and Sommerville (1998). Requirements Engineering, Processes and Techniques, Wiley.

[17] Kerzner, H. (2014) Project Recovery

[18] Pinto, J.K. (2013) Project Management, Achieving Competitive Advantage Global Edition

[19] Barker, S. and Cole, R. (2015) Brilliant Project Management.

Albuquerque, R. de O., Villalba, L.J.G., Orozco, A.L.S., Buiati, F. and Kim, T.-H. (2014) 'A Layered Trust Information Security Architecture', Sensors (Basel), .

[20] Hartman, B. (2009) New to agile? INVEST in good user stories. Available at: http://agileforall.com/new-to-agile-invest-in-good-user-stories/ (Accessed: 8 February 2017).

[21] Guindon, C. (no date) Eclipse desktop & web iDEs. Available at: https://eclipse.org/ide/ (Accessed: 8 February 2017).

[22] NetBeans IDE - Debugger and Profiler (no date) Available at: https://netbeans.org/features/java/debugger.html (Accessed: 8 February 2017).

[23] Home (no date) How to write a use case. Available at: http://www.bridging-the-gap.com/what-is-a-use-case/ (Accessed: 8 February 2017).

[24] UML 2 class diagrams: An agile introduction (2004) Available at: http://agilemodeling.com/artifacts/classDiagram.htm (Accessed: 8 February 2017).

[25] Search APM (2016) Available at: https://www.apm.org.uk/resources/what-is-project-management/ (Accessed: 8 February 2017).

[26] TeamGantt reviews: Overview, pricing and features (2016) Available at: https://reviews.financesonline.com/p/teamgantt/ (Accessed: 8 February 2017).

[27] IEEE standard glossary of software engineering terminology - IEEE Xplore document (2017) Available at: http://ieeexplore.ieee.org/document/159342/?reload=true (Accessed: 8 February 2017).

[28] Kushwaha, D. and Misra, A. (2008) Testing Using Junit [online]. 33 (3), pp.1-5.

[29] Understanding software testing concepts (2007) Available at: https://blackboard.uwe.ac.uk/bbcswebdav/pid-5277901-dt-content-rid-10430132_2/courses/UFCFED-30-M_16jan_1/jwalk.pdf (Accessed: 25 January 2017).