OptiSPAR

Design Project Report

Developer Name : Mark Aquilina
Developer ID : 3096249

Supervisor Name : Alec Simcock

Client Name : Tom Whitehouse (DSTO)
Summary

OptiSPAR is a program that optimises FORTRAN files that are based on the SPAR system model. SPAR is a tool that models a system, and predicts and analyses the life cycles of the system. What SPAR does is model these systems using Reliability Block Diagrams (RBD), which are made up of different components. These components are Line Replaceable Units (LRU) and Subsystems (which contain LRUs and other Subsystems). What OptiSPAR does is read in the FORTRAN files and converts them into Linked Lists. A Linked List can be seen as a representation of a RBD. Each node in a Linked List is either a LRU or a Subsystem. After the Linked List is created, numerous algorithms are used to optimise the Linked List, including moving LRUs up a system, sequentially organising LRUs, and most importantly, converting series and parallel components into a certain parallel structure. Finally, the optimised RBD is written back into the FORTRAN File. The Optimised RBD will now, in effect, be simulated more quickly than the original RBD.

The project was completed in 5 phases, although two of these phases were integrated together. These phases were the research phase, specification phase, design phase, and the implementation and testing phase. The research phase involved investigating what the requirements for the project would be, as well as looking into methods for solving the problem. This included deciding the best programming language for the project, and what data structures I would use. Java was chosen as the preferred language, and I decided to use a linked list as the main data structure. During the specification phase I came up with numerous ideas and solutions to problems, and was able to lay the groundwork for the design phase. The Design phase was a very important part of the creation of OptiSPAR. It was also the most difficult. During this time I created the class diagram, which took numerous attempts to complete. The last phase of the project was the implementation and testing phase. These phases were integrated together since I done much of the coding and testing at the same time. This phase was the most challenging, and the most rewarding.

OptiSPAR successfully optimises a RBD. A number of problems were encountered during the different phases of the project, such as not taking into consideration the scenario where two or more Parallel2 systems are located in a single RBD level. This was overcome eventually by using an “Invisible” subsystem, where LRUs and subsystems are contained in a subsystem that really doesn’t exist, so it is never written in a FORTRAN file. I have also learnt much while working on the project. I have gained invaluable experience in research, designing and implementing a piece of software, as well as working for a client and under a supervisor.
Table Of Contents

1. Introduction........................................................................................................5

2. Design Processes..................................................................................................6
   2.1 Problem Statement.........................................................................................6
   2.2 Software Requirements Specification.........................................................6
   2.3 Software Design Document..........................................................................6
   2.4 Test Plan.......................................................................................................7
   2.5 Implementation............................................................................................7

3. Project Management............................................................................................7
   3.1 Research Phase.............................................................................................7
   3.2 Specification Phase.......................................................................................8
   3.3 Design Phase................................................................................................8
   3.4 Implementation and Testing Phase...............................................................8
   3.5 Discussion.....................................................................................................9

4. Evaluation..........................................................................................................9
   4.1 Problems.........................................................................................................9
   4.2 Enhancements................................................................................................10
   4.3 Learning Aspects...........................................................................................10
   4.4 Improvements...............................................................................................11
   4.5 Conclusion....................................................................................................11

5. Conclusion.........................................................................................................12

6. Bibliography.....................................................................................................13

Appendix A - Software Requirements Specification........................................14
Appendix B - Software Design Document.........................................................15
Appendix C - Test Plan........................................................................................16
Appendix D - Time Summary...............................................................................17
Table of Figures

Figure 3.1 – Time Distribution by Percentages.............................................................. 9
1. Introduction

OptiSPAR is a program that will optimise a FORTRAN file and update a data file based on a SPAR system model. The reason for optimising the SPAR system model is because the current simulation time is quite large, and a method to reduce this time has been discovered, but needs to be implemented into a program.

SPAR is a modeling tool for predicting and analyzing the life cycle of systems. The tool takes various pieces of information about the system, like its reliability, what its intended use is and its maintenance schedule, and it uses this information about the system to predict its future. The SPAR system model is described using RBDs (Reliability Block Diagrams), which are designed in the SPAR tool. Each RBD is made up of components, or LRUs (Line Replaceable Units).

The SPAR system model can be a platform, such as a frigate or a destroyer, or a system, such as an engine, which contains other subsystems and LRUs. All these subsystems and LRUs are arranged in series and parallel. In order to optimise this system, as many LRUs as possible should be arranged in a certain parallel structure so it can be simulated more quickly than the original structure.

What the DSTO hopes to use the SPAR tool for is to sample the failure and repair distribution of LRUs in a platform over a length of time, such as a voyage over six months. Many variables are taken into consideration to sample this distribution. With these results from the sampling, different information is shown by the program, such as how many times a system will be estimated to fail, and how many spare parts might need to be taken. My project has nothing to do with this side of the program. It does however work with the RBDs of these LRUs, and this is where the optimisation takes place.

Overall, the program should allow simulation to be completed in a much quicker time than a simulation without optimisation.

The process of completing the project has been done in numerous stages. Firstly the DSTO briefed me about the program that needs to be created. This includes explaining what the SPAR modeling tool does. Then the model itself was explained. Finally, my role was explained. Next I created two documents outlining the needs of the program, and what the program will require and contain. The outline of the program was described in the Problem Statement, and the program requirements were contained in the Software Requirements Specification. A detailed design was created and documented in the Software Design Document, and then a testing plan was created. Finally the program was coded and tested.

---

This document contains information on the processes and outcomes of this design project. It also contains descriptions of project management and evaluation. Also included are the previous documents, and the code that makes the program.

2. Design Processes

OptiSPAR has been designed and implemented in designed and implemented using 5 different stages: A Problem Statement, Software Requirement Specification (Appendix A), Software Design Documentation (Appendix B), Test Plan Appendix C), and the coding.

2.1 Problem Statement

The Problem Statement briefly described what OptiSPAR (at that stage the program was called “SPAR Optimiser”) was for. The document contained information on the existing system, the proposed system, the fictional requirements, and the non-functional requirements. Through these explanations the theory behind the SPAR system was explained, along with the expectations of the program.

2.2 Software Requirements Specification

The Software Requirements Specification document gave a more detailed explanation of what is required for OptiSPAR (still referred to as “SPAR Optimiser” at that stage”). Included in the document was a more detailed description of what the program was about. The Stakeholders in the project were revealed, being my client, the DSTO, my supervisor Alec Simcock, and fellow students, Chow and Stan. The scope of the product was explained, including the users of the system and the different scenarios the program could take. A detailed explanation of the functional requirements, data requirements, and non-functional requirements was documented, explaining what was required of the program, and how to go about fulfilling these requirements. Finally the acceptance criteria were explained, being a program that will optimise a SPAR system model, thus reducing the simulation time of the model. The program must also record any number changes to LRU’s, and that any simulation results must be the same in the optimised version as the unoptimised version.

2.3 Software Design Document

Next the Software Design Document was created to describe how OptiSPAR was to be implemented. The design development was documented and explained two different methods that could be used to implement OptiSPAR. Firstly and Action-Orientated design is explained, then an Object-Orientated design is explained. Although the Action-Orientated design wasn’t used to create OptiSPAR, it was documented as a reference. The architectural design was the detailed, which included a detailed class diagram and description for OptiSPAR. Note that the class diagram in the Software Design Document changed slightly from the actual class diagram implemented. The reason behind this is that as I started coding the program, it became evident that the way that I wanted to implement the program would not be able to exactly match the design of the project. A
detailed design of the class diagram was then given, including Pseudocode for different methods in some classes and actual code in other classes, and where the requirements from the Software Requirement Specification were met. An interface design was created using screen mock-ups, with a description of what each was to be created for. Finally a data dictionary explained the use of each class in OptiSPAR, describing the links between each class.

### 2.4 Test Plan

The Test Plan was documented to explain the integration and tests that would be required for OptiSPAR. A bottom-up approach is used in OptiSPAR, which means that each object that is called by another object will have already been tested. Tests for each class are described, detailing the explanations for inputs and outputs. Unit testing was documented to give actual tests for the classes, and the results that are to be expected. System Testing was then described explaining how each Functional Requirement should be met and any performance testing, acceptance testing, and installation testing that would be required.

### 2.5 Implementation

Finally the program was implemented. Through implementation the coding of each class was completed. Testing of each class was done throughout the implementation stage, both during and after the creation of each class. Throughout the implementation stage much research was done for different methods that could be used to complete each part of the program. This included ways of creating Linked Lists for the Reliability Block Diagram, methods for string extraction from files, creating arrays of variable sizes to accommodate unknown model sizes, and various other pieces of research.

One thing that I researched that eventually became very handy was a class file called ArrayList. It behaves like a normal array, except that it can be resizable, and of an infinite size. I used ArrayList widely in many of my classes, and made the implementation of the project a lot easier.

### 3. Project Management

The OptiSPAR project has been created in 5 stages. These are:

1. Research
2. Specification
3. Design
4. Implementation
5. Testing

#### 3.1 Research Phase

The research phase involved investigating the requirements for the project. Much of this research was done in the Problem Statement and Software Requirement Specification
documents. Approximately 26 hours was spent in researching the project, and this was done from weeks 1 to 6. The first week was used deciding what project to do, and a little research was done to understand the problem. I also decided what language I would create the project in (Java), as well as how I would go about implementing the project, such as choosing a linked list for a data structure. In week two, a couple of meetings to begin understanding what the project was about and how to go about implementing it were attended. In week three, another meeting with the DSTO was organised and was used to learn more of what the project was about. Following this, much investigation was done into how to design OptiSPAR, including getting the FORTRAN file and putting it into a linked list, optimising the Linked List, and then putting the Linked List back into a FORTRAN file. Research was also done in the different setups of a Reliability Block Diagram, including series, parallel, and subsystem systems. A few graphical user interface mockups were also created. Time was also spent during implementation to research a number of topics in Java.

Time Spent – 22.3 Hours

3.2 Specification Phase

Much of the specification phase was spent creating the Software Requirement Specification document, but time was also spent on preparing and demonstrating oral presentations. During specification, numerous ideas and solutions to problems were devised, and these were used to satisfy the functional requirements. Much of the specification phase laid the groundwork for the design phase, including creating the initial class diagram.

Time Spent – 22 Hours

3.3 Design Phase

The design phase was probably the most difficult phase of the project. It involved much work in deciding how to go about creating OptiSPAR. Much of this work is documented in the Software Design Document. A lot of difficulty was encountered when trying to create the class diagram with all the methods. Eventually it was finished, but only after several attempts at designing it.

Time Spent – 26.5 Hours

3.4 Implementation and Testing Phase

The implementation and testing phase has been incorporated together due to the fact that while implementing the program, testing was performed. Many hours were spent implementing and testing the program. During this time, the Test Plan was also created to work out how each part of the program should be working, and to find any faults in the program. Many hours have been spent in this phase coding classes, and testing that these classes are working and working with the other classes. Numerous problems were
detected while coding and testing, and these had to be overcome by either redesigning the class, or changing parts of the code. I found this phase the most time-consuming, the most challenging, but it was also the most rewarding.

*Time Spent – 137.5*

### 3.5 Discussion

Overall, most of the time spent was during the Implementation and Testing phase. As figure 3.1 shows, over half of the time was spent in the Implementation and Testing phase, while the Research, Specification, and Design phase were divided similarly. The full table of time spent on each phase, as well as time spent during each week is shown in *Appendix D.*

![Figure 3.1 – Time Distribution by Percentages](image)

4. Evaluation

My program, “OptiSPAR”, solves the problem of optimising a Reliability Block Diagram. It successfully reads in the FORTRAN file, converts it into a linked list, optimises the linked list, and then writes the optimised linked list back into a FORTRAN file. In comparison to tested optimised files, there is no difference in the structure of the files. The corresponding data file has also been successfully implemented.

#### 4.1 Problems

During the course of the implementation a problem was discovered where numerous parallel2 systems could be in series in one level. The problem was that this wasn’t taken into consideration. To overcome this problem I altered the design of the Reliability Block Diagram Class to implement an “Invisible Subsystem”. The invisible subsystem is contained in a normal subsystem, with multiple subsystems in series. Each parallel2 system is contained in one of these invisible subsystems, and any optimisation takes into account these systems, but when the FORTRAN file is being written, they are not taken written, hence the name “Invisible Subsystem.”
Another problem was encountered where multiple parallel1 systems could be contained in one level. The problem with this is how it is written in the FORTRAN code. This also wasn’t taken into consideration at the start of the program, and has had to be overcome by altering the design of the project.

One simple, but time consuming, problem that needed to be overcome was the formatting of the output files. The problem occurred with the carriage return symbols. When Java outputs a new line character (\n) it writes it as a Unix new line character. However, SPAR is unable to read this character, as it requires a DOS new line character. After much investigation, it was discovered that for a DOS new line character, and carriage return (\r) followed by a new line (\n) was needed. This was changed wherever required in the code, and the problem was overcome.

Various other problems were discovered during the implementation of the program. These include problems from simple programming errors to faults in the design of the program. As more code was written, fewer errors were made due to better experience and preparation.

In one of the example optimisation files, it was discovered that an error was made in the format of the FORTRAN file. Fortunately, OptiSPAR didn’t encounter this error.

As far as is known, the project will be able to read, optimise and write any RBD required. Numerous tests will be conducted at the DSTO do make sure all is working well with the program.

### 4.2 Enhancements

During the course of programming, numerous enhancements were made to the code to improve the time it takes the program to optimise the RBD. Two notable changes occurred in the FORTRANReader class. The FORTRANReader class reads one line of the FORTRAN file in and stores it in a single String. In order to do this a new String object is created for every line read. This was very time consuming for a large FORTRAN File. Another String class in Java, called StringBuffer, has a method that allows the program to append to the String, instead of creating a new String every time. This cut the reading in time down dramatically.

Another improvement in the FORTRANReader occurred when the class was altered to read in only parts of a FORTRAN file instead of the whole FORTRAN file. This improved the reading in of FORTRAN files three times.

### 4.3 Learning Aspects

While creating OptiSPAR, I have been able to learn many aspects of software engineering and programming. I learnt numerous software engineering practices, including how to create various documents such as the SRS and SDD. I have also learnt how to create a class diagram and how to add methods to the class diagram. OptiSPAR was implemented using a bottom-up implementation technique. I used this technique
because I researched the different ways of implementing programs, and decided that this would be the best for OptiSPAR.

Apart from learning software engineering techniques, I also vastly improved my programming. I learnt much about implementing algorithms and data structures since OptiSPAR heavily dependent on these concepts. I also enhanced my Java programming skills, and learnt many new ways of creating linked lists, infinite capacity arrays, reading in files, writing out files, using the StringTokenizer, and many other things. Overall I believe that the project has increased my understanding of programming.

I also gained knowledge and experience in working for a client and working under a supervisor. In working for a client I had to communicate in numerous ways, such as talking over the telephone, talking face-to-face, and emailing. I also had weekly meetings with my supervisor to report on my progress in the project, and my plans for the upcoming days.

4.4 Improvements

There are a number of things that I believe could have been done differently. This includes that way that I implemented the program at the start. Now that I look at the code I could have done things differently, more efficiently, and better. Even code that I wrote recently I think could have been done better, but time restraints didn’t allow me to try to rewrite the classes. Even though the design of the project reflected what was created in the program, a number of things could have been better. This includes the setup of the classes, and the methods involved in these classes.

4.5 Conclusion

All in all, I believe that I have gained much from my design project. I have vastly improved my skills with using Java and programming in general, I have gained knowledge in software engineering, I have learned to organise my time to complete various tasks required for the design project as well as do work on my project while having other subjects to study for.

I have run into a number of difficulties while working on my project, including design problems and coding problems. I was able overcome these difficulties, although I did find that sometimes after solving one problem, often another problem would crop up. There are also some issues that I wish I addressed differently (such as how I read in the FORTRAN File), but unfortunately I didn’t have to do this.
5. Conclusion

Overall, OptiSPAR was successfully designed and completed in time. The program did what it was supposed to do, which was optimise a SPAR system, and successfully passed the numerous tests that it was put through.

Although numerous difficulties were encountered throughout the project, these were overcome. The project was not only successful in optimising a SPAR system, but it also helped me learn and improve many skills.

OptiSPAR was implemented in 5 stages, although two of the stages were done concurrently. First came the researching stage, where I investigated the requirements of the project, and came up with a number of ideas on how to create the program. Next came the specification stage, where I furthered my ideas on how to implement the problems. Then the design stage was used to put my ideas onto paper and create the architecture for OptiSPAR. Finally the implementation and testing stage was completed, where each class was programmed and integrated, and testing was completed concurrently.
6. Bibliography


Appendix A - Software Requirements Specification

This appendix contains the up to date Software Requirements Specification for OptiSPAR.
Appendix B - Software Design Document

This appendix contains the up to date Software Design Document for OptiSPAR.
Appendix C - Test Plan
This appendix contains the up to date Test Plan for OptiSPAR.
Appendix D - Time Summary

The following tables contain summaries of time spent of different tasks every week of each semester.

<table>
<thead>
<tr>
<th>Week</th>
<th>R</th>
<th>S</th>
<th>D</th>
<th>I &amp; T</th>
<th>Weekly Total</th>
<th>Running Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td>3.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>5.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td>5.4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>7.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>8.4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td>12.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>12.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td>12.4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>17.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>19.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>20.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>24.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td>25.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>26.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td>26.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td>27.2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td>27.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>30.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>34.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>36.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>39.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td>39.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>41.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14.3</td>
<td></td>
<td></td>
<td></td>
<td>42.2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td>42.2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>46.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>47.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>49.2</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>52.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>57.2</td>
<td></td>
</tr>
<tr>
<td>Totals</td>
<td>17.3</td>
<td>22</td>
<td>18</td>
<td></td>
<td>57.2</td>
<td></td>
</tr>
<tr>
<td>Week</td>
<td>R</td>
<td>S</td>
<td>D</td>
<td>I &amp; T</td>
<td>Weekly Total</td>
<td>Running Total</td>
</tr>
<tr>
<td>------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5.5</td>
<td>7.5</td>
<td>15.5</td>
<td>22.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>20</td>
<td>20</td>
<td>42.5</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>20</td>
<td>20</td>
<td>69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3.5</td>
<td>3.5</td>
<td>7</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>7</td>
<td>103</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>5</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>3</td>
<td>103</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>5</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>130</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Break</td>
<td>8</td>
<td>8</td>
<td>138</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>14</td>
<td>144</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>16</td>
<td>146</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>10</td>
<td>206</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub</td>
<td>5</td>
<td>22</td>
<td>36.5</td>
<td>187.5</td>
<td>206</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>22.3</td>
<td>22</td>
<td>36.5</td>
<td>187.5</td>
<td>263.2</td>
<td></td>
</tr>
</tbody>
</table>

**Legend**

- R – Research Phase
- S – Specification Phase
- D – Design Phase
- I & T – Implementation and Testing Phase