A PROJECT MANAGER’S ASSISTANT

A Chapter Style Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Software Engineering

Benjamin R. Garbers

College of Science and Health
Department of Computer Science/Software Engineering

August, 2006
A Project Manager’s Assistant

By Benjamin R. Garbers

We recommend acceptance of this manuscript in partial fulfillment of this candidate’s requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

Dr. Kasi Periyasamy
Examination Committee Chairperson

Date

Dr. Kenny Hunt
Examination Committee Member

Date

Dr. Mao Zheng
Examination Committee Member

Date

Dr. Vijendra Agarwal
Director, University Graduate Studies

Date
ABSTRACT

Garbers, Benjamin, R., “A Project Manager’s Assistant”, Master of Software Engineering, August 2006, Advisors: Dr. Kasi Periyasamy, Dr. Kenny Hunt

Software project management is a difficult endeavor due to the complexities of maintaining the cooperation and coordination of project team members in order to insure the ordered completion of numerous tasks with multifaceted inter-relationships. In addition to sharing software artifacts such as requirements documents, design documents, source code, and test cases, timely communication among team members is critical for the success of the project. Additional complexities arise when an individual team member assumes multiple roles (e.g., a project manager also acts as the requirements specialist) or when a single role is shared among multiple team members (more than one designer or programmer). Software project management tools vary from simple spreadsheets to complex software packages that allow the project manager to track project tasks, time sheets and billing. Most of these packages require the project manager and other users to input data at the right time. Moreover, the software artifacts such as the requirements document, design document and source code are not accessed via these tools. This paper describes a project management software tool supporting a wide array of project management activities. The tool was developed at the University of Wisconsin-La Crosse in coordination with IBM, Rochester, MN as an Eclipse plug-in
built on top of the Agent Building and Learning Environment (ABLE) framework. The tool provides a wide array of automatic and semi-automatic support for managing team communication; interactive development and dissemination of software artifacts (requirements and design documents); managed review of documents; time tracking; tracking skill-set profiles of project team members; the intelligent selection of project team members using personal profiles, and cost estimation.
ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my project advisors Dr. Kasi Periyasamy and Dr. Kenny Hunt for their valuable guidance. I like to thank the project sponsor, IBM (International Business Machines) who initiated this project and provided the support for this project. I would also like to express my thanks to the Computer Science Department and the University of Wisconsin-La Crosse for providing the computing environment for my project. Finally, I wish to thank my wife Nicole and our baby boy Brennan for their patience and encouragement during the tenure of this project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>v</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>GLOSSARY</td>
<td>xi</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 A Project Management Case Study</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Using ABLE Agents</td>
<td>5</td>
</tr>
<tr>
<td>1.3 PMA Toolset Overview</td>
<td>6</td>
</tr>
<tr>
<td>1.4 PMA Interaction</td>
<td>8</td>
</tr>
<tr>
<td>2. Project Plan Tool</td>
<td>11</td>
</tr>
<tr>
<td>3. Employee Experience Tool</td>
<td>13</td>
</tr>
<tr>
<td>4. PMA Task Assignment</td>
<td>14</td>
</tr>
<tr>
<td>4.1 Assigning Requirement Tasks</td>
<td>16</td>
</tr>
<tr>
<td>4.2 Assigning Design Tasks</td>
<td>19</td>
</tr>
<tr>
<td>4.3 Assigning Code Implementation</td>
<td>22</td>
</tr>
<tr>
<td>4.4 Assigning Test Plan Creation</td>
<td>25</td>
</tr>
<tr>
<td>4.5 Assigning Test Plan Execution</td>
<td>29</td>
</tr>
<tr>
<td>5. Requirements Editor</td>
<td>31</td>
</tr>
<tr>
<td>5.1 Function Point Analysis</td>
<td>32</td>
</tr>
<tr>
<td>5.2 DeMarco Model</td>
<td>38</td>
</tr>
<tr>
<td>5.3 Estimation</td>
<td>40</td>
</tr>
<tr>
<td>5.4 Requirements Editor Additional Functionality</td>
<td>43</td>
</tr>
<tr>
<td>5.5 Creating a Tentative Schedule from Requirements</td>
<td>43</td>
</tr>
<tr>
<td>6. Design Editor</td>
<td>46</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>7</td>
<td>Test Editor</td>
</tr>
<tr>
<td>7.1</td>
<td>Test Editor Additional Functionality</td>
</tr>
<tr>
<td>7.2</td>
<td>Project Managers Test Plan Execution Decisions</td>
</tr>
<tr>
<td>8</td>
<td>PMA Metrics</td>
</tr>
<tr>
<td>9</td>
<td>Example Case Study</td>
</tr>
<tr>
<td>9.1</td>
<td>Employee Experience Breakdown</td>
</tr>
<tr>
<td>10</td>
<td>Conclusion</td>
</tr>
<tr>
<td>11</td>
<td>Other Project Management Software</td>
</tr>
<tr>
<td>12</td>
<td>Continuing Work</td>
</tr>
<tr>
<td>12.1</td>
<td>Employee Experience Tool</td>
</tr>
<tr>
<td>12.2</td>
<td>Project Plan Tool</td>
</tr>
<tr>
<td>12.3</td>
<td>Requirements Editor</td>
</tr>
<tr>
<td>12.4</td>
<td>Design Editor</td>
</tr>
<tr>
<td>12.5</td>
<td>Test Plan Editor</td>
</tr>
<tr>
<td>12.6</td>
<td>PMA</td>
</tr>
<tr>
<td>13</td>
<td>Bibliography</td>
</tr>
<tr>
<td>14</td>
<td>Appendix A: Implementation Notes</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE 5.1 Unadjusted function point calculation.</td>
<td>34</td>
</tr>
<tr>
<td>TABLE 5.2 Example of increased complexity.</td>
<td>38</td>
</tr>
<tr>
<td>TABLE 5.3 Weight selections for each function type.</td>
<td>39</td>
</tr>
<tr>
<td>TABLE 9.4 Role break down.</td>
<td>59</td>
</tr>
<tr>
<td>TABLE 14.1 Users Table.</td>
<td>76</td>
</tr>
<tr>
<td>TABLE 14.2 Project.</td>
<td>77</td>
</tr>
<tr>
<td>TABLE 14.3 Employee experience table.</td>
<td>78</td>
</tr>
<tr>
<td>TABLE 14.4 Tasks Table.</td>
<td>78</td>
</tr>
<tr>
<td>TABLE 14.5 Design Tasks.</td>
<td>79</td>
</tr>
<tr>
<td>TABLE 14.6 Code Tasks Table.</td>
<td>79</td>
</tr>
<tr>
<td>TABLE 14.7 Test Plan Tasks Table.</td>
<td>79</td>
</tr>
<tr>
<td>TABLE 14.8 Test Plan Execution.</td>
<td>80</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIGURE 1.1</td>
<td>AGENT FRAMEWORK FOR DIFFERENT EMPLOYEE AGENTS</td>
<td>6</td>
</tr>
<tr>
<td>FIGURE 1.2</td>
<td>EMPLOYEE EXPERIENCE EXAMPLE</td>
<td>9</td>
</tr>
<tr>
<td>FIGURE 1.3</td>
<td>MILESTONES FROM SOFTWARE DEVELOPMENT PROCESS LIFE CYCLE</td>
<td>10</td>
</tr>
<tr>
<td>FIGURE 4.1</td>
<td>PROJECT PLAN EDITORS TOOLS THAT ARE USED IN THE PROJECT</td>
<td>14</td>
</tr>
<tr>
<td>FIGURE 4.2</td>
<td>PROJECT PLAN EDITOR TECHNOLOGY</td>
<td>15</td>
</tr>
<tr>
<td>FIGURE 4.3</td>
<td>EMPLOYEE EXPERIENCE WITH FUNCTIONAL REQUIREMENTS</td>
<td>16</td>
</tr>
<tr>
<td>FIGURE 4.4</td>
<td>EMPLOYEE RANKED FOR FUNCTIONAL REQUIREMENTS TASK</td>
<td>17</td>
</tr>
<tr>
<td>FIGURE 4.5</td>
<td>THE ENTRY 'STAFFING' IN THE PROJECT PLAN</td>
<td>18</td>
</tr>
<tr>
<td>FIGURE 4.6</td>
<td>EMPLOYEE EXPERIENCE WITH DESIGN DOCUMENTATION</td>
<td>19</td>
</tr>
<tr>
<td>FIGURE 4.7</td>
<td>ASSIGNING DESIGN TASKS</td>
<td>21</td>
</tr>
<tr>
<td>FIGURE 4.8</td>
<td>CODING EXPERIENCE FOR THE EMPLOYEE</td>
<td>23</td>
</tr>
<tr>
<td>FIGURE 4.9</td>
<td>PROJECT PLAN EDITOR CONFIGURATION MANAGEMENT</td>
<td>24</td>
</tr>
<tr>
<td>FIGURE 4.10</td>
<td>EMPLOYEES TEST CASE EXPERIENCE</td>
<td>26</td>
</tr>
<tr>
<td>FIGURE 4.11</td>
<td>RUNNING TEST CASE EXPERIENCE FOR AN EMPLOYEE</td>
<td>29</td>
</tr>
<tr>
<td>FIGURE 5.1</td>
<td>PROJECT MANAGER ASSIGNING REQUIREMENT AND TESTING TASKS TO SOFTWARE ENGINEER(s) AND SOFTWARE TESTER(s)</td>
<td>31</td>
</tr>
<tr>
<td>FIGURE 5.2</td>
<td>UNADJUSTED FUNCTION POINT</td>
<td>32</td>
</tr>
<tr>
<td>FIGURE 5.3</td>
<td>DEMARCOS MODEL CALCULATION</td>
<td>40</td>
</tr>
<tr>
<td>FIGURE 5.4</td>
<td>CALCULATED ESTIMATES FROM EACH FUNCTIONAL REQUIREMENT</td>
<td>41</td>
</tr>
<tr>
<td>FIGURE 5.5</td>
<td>COMPARISON BETWEEN ESTIMATION FORM REQUIREMENTS EDITOR AUTHOR AND DEMARCOS MODEL</td>
<td>42</td>
</tr>
<tr>
<td>FIGURE 5.6</td>
<td>RESPONSIBILITIES, HOURS PER DAY AND DAYS PER WEEK THAT AN EMPLOYEE WILL WORK</td>
<td>44</td>
</tr>
<tr>
<td>FIGURE 5.7</td>
<td>GENERATED SCHEDULE FROM THE FUNCTIONAL REQUIREMENTS</td>
<td>45</td>
</tr>
</tbody>
</table>
FIGURE 6.1 Project manager assigning design tasks and testing tasks to software engineer(s) and software tester(s) ........................................ 47

FIGURE 7.1 Project manager assigning tasks to software engineer(s) based off of testing results. ............................................................... 51

FIGURE 8.1 Document creation and review time vs. the document estimation. .......................................................................................... 55

FIGURE 8.2 Requirements editor analysis of categorical words. .............. 56

FIGURE 8.3 Looks at one functional requirement and the total time in minutes it took to implement within each development stage.............. 58

FIGURE 9.1 Project manager creates the project. .............................................. 60

FIGURE 9.2 Software engineer joins the project. .............................................. 61

FIGURE 9.3 Project plan information. ................................................................. 62

FIGURE 9.4 Project manager asking for functional requirements advice .... 63

FIGURE 9.5 Experience of student1 ................................................................. 64

FIGURE 9.6 Student1 accepting the responsibility ............................................. 65

FIGURE 9.7 Scheduling advice ...................................................................... 65
ABLE
The Agent Building and Learning Environment is an artificial intelligence package developed by the Thomas Watson Research Center. Within this package a developer can use fuzzy logic, neural network and ruleset algorithms.

IEEE
An acronym for Institute of Electrical and Electronics Engineers, Inc., which is an international organization whose constitution describes their purpose as “scientific and educational, directed toward the advancement of the theory and practice of several engineering fields including computer science.”

IEEE 830-1998 Standard
This is a recommended practice describing approaches for the specification of software requirements. It emphasizes the development of unambiguous and complete requirements specification documents.

LOC
Lines of code which is the number of lines of text within development code.

PDF
An Adobe technology representation for formatting documents so that they can be viewed and printed using the Adobe Acrobat reader.

PMA
The Project Management Assistant which is the name of the software application that was developed for this capstone project. This application will help assist project managers in the overall management of a software project.

PMI
The Project Management Institute is an organization that develops project management education and certifications.
SRS
Software Requirements Specification is a document format supplied by the IEEE
830-1998 Standard for specifying the requirements of a software system.
1. Introduction

The ability to create quality software on time within budget is very challenging. Empirical evidences show that 50 – 70 percent of software projects fail [1] because of poor project management. According to Pankaj Jalote [4], “delivering high quality software within budget and on schedule must be an oxymoron”. Today’s software projects are so large and complex that a lot of time is invested just to track the various activities for every project. For example, a typical data-oriented software project involves all activities of the software life cycle such as requirements gathering and analysis, requirements specification, architectural and detailed design, implementation, testing and maintenance. Each one of these tasks may be assigned to one or more individuals. These individuals are expected to cooperate and coordinate with each other in order to successfully complete a project before the deadline and within the budget. The complexity increases with the number of projects and the number of members working on the projects. Since the complexity is so high, project managers are assigned to help manage the different projects. The project manager has a number of responsibilities that must be clearly identified.

Today’s project managers face a number of challenges that include managing the technical activities such as assigning work load to team members, reviewing documents and coordinating with team members at the right time. In addition, project managers are also responsible to find the right group of people for each technical task, motivating them to do their assigned task within deadline and budget, and finally properly documenting all activities of a project. The role of a project manager must be well defined and well-understood. In [1] Richard Whiteheads book, a project manager is described as a person who is responsible for planning, budgeting, liaising with management and negotiating with customers. In a larger project, or in a multidisciplinary project, a project manager would direct the work of several team leaders and is critically responsible for the delivery of the project on time and within budget [2]. The Project Management Institute (PMI) defines Project Management as a set of proven principles, methods and techniques for the effective planning, scheduling, controlling and tracking of deliverables (results) that help to establish a
sound historical basis for planning of future projects[3]. Depending on the organization, the responsibilities required for a project manager may vary.

1.1 A Project Management Case Study

Consider game scheduling software used by a recreational sports department. The game scheduler application should allow an administrator to manipulate facilities, the times the facilities are available, players, teams, and leagues. Once all the leagues have been created, a schedule should be dynamically created based on the information provided. Let us assume that three individuals have been assigned to complete this project. Further, it is also assumed that each of these individuals have adequate software development experience including development, testing and project management. However, the tools used within software development and software testing approaches used are different among the three employees. Employee A and employee B have Visual C experience and use the Visual Studio IDE while employee C has significant experience with the Java programming language and uses the Eclipse IDE. Employee A only works two days a week while the others work 3 days a week. The project manager must assign appropriate tasks based on this information.

One of the responsibilities of the project manager is to assign specific tasks to each individual. These tasks depend on the employee’s experience, the tools and processes used by the employee, the tools that are planned to be used and the processes that are followed as per the project plan. The milestones for the development process should consist of requirements gathering and documentation, design documentation, code implementation and test plan creation and execution. Each employee will be given the task assignment based on the milestones. For example, the game scheduler application is going to use Java technology for code implementation which would mean employee C would be the most qualified person for implementation. The project manager must also look at the availability of each employee before assigning the task along with the project completion date. In case of problems with meeting milestones or deadlines, the project manager may assign a task to other employee(s) in the team that is available. There might even be a circumstance that an employee, with no experience on a particular tool or language, should be given an assignment to
use that tool or language. In this case, the employee should be given adequate time to learn the tool or language and the deadline should be adjusted. These are a few of several things that must be taken into consideration when assigning particular individual tasks.

When assigning the tasks, it is also important to track each assigned task so that the project manager can make sure that the project will meet the expected deadline. Assume that the recreational sports department expects the application to be delivered within 3 months. Requirements have to be completed in order to discuss with the customer whether 3 months is going to be an acceptable time frame. After analyzing all requirements and looking at the estimated time frame for each development milestone, the 3-month deadline may be not practical or not attainable. If this is the case, it must be communicated to the customer and a project completion date must be negotiated. Once the project completion date is agreed upon, it is up to the project manager to make sure the project is on target. The project manager must also give advices on quality checks and implementing those checks within the target deadline.

In the case that development milestones start to slip or are possibly delayed, it is the responsibility of the project manager to have a contingency plan. Perhaps the customer may not be available for requirements clarification, thus causing the completion date to be moved. The project manager must communicate this to the customer, making sure they are aware of the situation. Perhaps he will allocate more resources from his team for requirements gathering and documentation. Status on each required milestone must be communicated to the customer by the project manager along with upper management.

Another challenging responsibility of a project manager is to make sure the project activities are carried out within budget. A customer needs to understand how much a project will cost before they approve the project. Cost estimation must be done during the requirements stage and must be communicated to the customer. Once the customer agrees on the cost, it is up to the project manager to make sure they stay within the given budget. Any time spent after the completion date means additional expenses for the developer that were not accounted for by the customer.
As one can see there are many challenging responsibilities for a project manager. Assigning tasks to individuals, monitoring the assigned tasks, making sure the project is within budget, everyday communication with upper management and the customer are just a few of the many responsibilities of an efficient project manager.
1.2 Using ABLE Agents

Since a project manager is required to perform many responsible and critical activities and is also required to closely monitor the assigned tasks, it is important to have a tool that will improve and possibly automate decision making. The aim of this thesis is to develop such a tool called the Project Manager’s Assistant (PMA) to assist a project manager in his/her work. Using this tool, the project manager can perform a variety of tasks, and can also gather and analyze information that support decision making. The tool was built using the Agent Building and Learning Environment (ABLE). The ABLE framework was developed at IBM lab in Rochester, MN and it allows several software agents to be built and run.

An agent is a smart piece of software that accomplishes certain specific tasks. Within PMA, each employee is treated as an agent. The structure of this agent and the tasks that can be performed by this agent depend on the information inputted by the employee. Figure 1.1 shows three different agents – a project manager agent, a software engineer agent and a software tester agent. The Project Manager agent will assign certain tasks to the Software Engineer Agent. The Software Engineer Agent then reports status back to the Project Manager agent on the specific task. The same interaction would occur with the Software Tester Agent.
Figure 1.1  Agent framework for different employee agents.

The ABLE framework enables to gather employees’ personal information which helps project managers and team leaders to decide the tasks to be assigned to each individual. Milestones need to be set and tracked so that the project manager knows where the project is at any given time. Estimated schedules must be created to understand the software application delivery. The Project Manager Assistant (PMA) was designed to do just this.

1.3 PMA Toolset Overview

PMA relies on a number of software artifacts to be created and maintained throughout the software development lifecycle. These include a project plan, requirements document, design document and test plan along with employees’ experience reports. Each software artifact can be created and modified by separate tools that PMA uses. The different tools within PMA that accomplish the above tasks are described below:
**Project plan tool.** This tool is used by the project manager at the beginning of the project life cycle to plan the project activities. The project manager is required to input all pertinent information about the project such as project description, tools and languages to be used, number of team members required, skill set required to complete the project and so on.

**Employee experience tool.** This tool can be used as a standalone tool by the employees to create their profiles. The project manager will access this information through PMA in order to select the right team member for the project.

**Task assignment functionality within PMA.** The project manager uses this functionality to assign various tasks to the team members. For example, a project manager may assign the task of developing the requirements document to one of the team members. Typically, this functionality provides interactive communication between the project manager and team members.

**Requirements Editor.** This tool provides an editor to develop an IEEE compliant requirements document. It also provides a user to enter function-point metrics, DeMarcos Model information and development life cycle milestone estimations. These three different types of estimations can be used to help understand how long it will take to complete the project. In addition, the editor also provides feedback mechanisms through which a team member reviewing the requirements can send the feedback on the requirements.

**Design editor.** This tool is very similar to the requirements editor but for developing an object-oriented design document based on the requirements. It also provides the mechanism for design reviewers to provide feedback on the design.

**Test editor.** Software testers use this tool to input test cases and test case steps to form a test plan. The test plan can be reviewed where other members of the team can send feedback; this functionality is similar to the functionality in the requirements and design editors. The tester can run the test cases once code implementation is completed. Results of the test cases can be entered using the Test editor and can be sent to the project manager.
1.4 PMA Interaction

PMA first requires a project plan to be created at the beginning of a project. This helps the development team understand the organization of the project. The project manager inputs several project-related information such as the tools that will be used, technical skills required, dependencies, risks, etc. This information helps understand what skills are required to make this project successful.

Once the project plan is created, the project manager must choose the most qualified employees for this project. Typically, without the PMA, this is done manually by the project manager by scanning through the employees’ personal records. PMA has built-in features to assist the project manager in this case. Accordingly, every employee must enter his/her personal profile (qualifications and past experience) using PMA. The project manager then selects the most suitable employees using the suggestions provided by PMA. The project manager then consults with the employees with regard to their selection and assignment of tasks. Once the employees are selected, they are added to the project. PMA stores this information internally for future references. The technical skills required for the project as described in the project plan, can be cross referenced to the employees’ skills to understand possible education or training that might be required. Figure 1.2 shows a sample scenario where Employee A will need possible education due to the fact that he/she does not have XML or Eclipse experience which is required for the chosen project.
As stated in an earlier section, the project manager is required to set and monitor several milestones for the project. These milestones in turn depend on the tasks assigned to various individuals. PMA helps in setting and keeping track of the tasks and milestones. Within PMA, these tasks include gathering and reviewing requirements, writing and reviewing design documentation, implementing the design and then writing, reviewing and implementing a test plan. Figure 1.3 shows the milestones that are tracked within PMA.
Each milestone has a software artifact (document) associated to it. For example, the requirements engineering process creates a requirements document which is later reviewed by another individual or group. The same holds true for the design documentation and the test plan documentation. Separate tools are used to input information for the documents. The information used within these documents can be analyzed by the project manager at any time. This helps the project manager to closely monitor the project and provide assistance and advice to the team members when needed.

The creation of project plan and employees’ profiles, the requirements editor, the design editor and the test plan editor tools will be described in detail in the following sections. All these tools can also be used externally outside of PMA where standardized files can be saved for review.
2. Project Plan Tool

The project plan tool is required to be used at the beginning of the project. The project plan consists of important details describing the project. Over the duration of the project, more information may be added to the project plan. For example, if an employee quits the project and a new employee is hired, the project plans must be updated. The project plan tool can be used within PMA or can also be used as a standalone tool. Since this is a working document throughout the project, the project plan is saved for future reference. PMA creates a .pnap file while saving a project plan. This file can also be viewed externally. If the project plan is created within PMA it can then be saved within PMA’s database which then uses the information to make certain project management decisions. The .pnap file can even be imported into PMA if an employee used the standalone project plan tool. The following information can be entered using the project plan tool:

- Project Description – description of the project including project start date and expected project end date.
- Background – information on why the project is needed.
- Responsibilities – the roles of each employee along with estimated hours per day and days per week the employee will work.
- Tools Used in Project – tools that will be used for the requirements, design, coding and testing phases of development.
- Technologies Used – technologies that will be used when developing the project.
- Staffing – the number of project managers, software engineers and software testers required for this project along with required skills.
- Training – education that will be given for this project.
- External Relationships and Dependencies – the external relationships and dependencies that this project has on another project and the external relationships and dependencies that other projects have on this project.
- Risks – risks that may occur during this project.
• Procedures and Quality – the different procedures that will be used during this project along with the quality measures that will be taken during each phase of the development life cycle.
• Configuration Management – the repository that will be used to store the different documents and code.
• Progress Monitoring -- the understanding of how time and cost will be monitored throughout this project along with what management meetings and reports that are required.
• Testing – the testing strategy that will be used in this project.
• Bugs and Issues – the process of how bugs and other issues will be reported.
• Release Procedures -- the procedures for each software application release.
• Customer Acceptance – the process of how customers will accept the product.
• Licensing – the licensing strategy for the application being distributed.
• Security -- the way security will be implemented within the released application.
• Requirements Change Procedure – the procedure to be followed when requirements change.
• Hardware Required – the hardware that will be required to produce this product.
3. Employee Experience Tool

In this framework, every employee must create a personal profile that includes his/her qualifications and past software development experience. The past experience information can be entered through the standalone employee experience tool or within PMA. The employees experience can be saved and then later be distributed to other members of the team. PMA makes use of the employee’s personal profiles in certain decision making instances. PMA can also import .experience files that have been created using the standalone experience tool. The following information can be input within the employee experience tool:

- **Project Description** – date the employee started and the date when the employee finished the project along with a description.
- **Requirements** – time spent gathering and writing requirements, education spent on understanding requirements and the tool used to write the requirements.
- **Design** – time spent writing design documentation, education spent on understanding how to write an effective design and the tool used to write the design document.
- **Code** – programming languages used within the project, time spent writing code, education spent on understanding the programming language and the Integrated Development Environment (IDE) used.
- **Write Test Cases** – number of test cases written, the total time spent on creating test cases using functional requirements, the total time spent on creating test cases using design document and the tool used to create test cases.
- **Run Test Cases** – the total number of test suites run and the tool used.

**Soft Skills** – a rating between 1 – 10 where 1 is low and 10 is high on self awareness, self regulation, motivation, empathy and social skills.
4. PMA Task Assignment

When the project manager assigns certain tasks to employees, the technologies used, tools used in project and configuration management from the project plan are cross referenced against the employees’ experience. The project manager will assign certain tasks to either software engineers or software testers. Software engineers will be assigned requirements, design and code implementation tasks while the software testers will be assigned writing the test plan and implementing the test plan. Figure 4.1 shows the project plan editor where the tools that are going to be used within this project can be entered. This is important to cross reference against the employees experience. Obviously an employee who has experience in a particular tool that the project is using will be an easy choice for the project manager.

![Project Plan Editor for GAME SCHEDULER](image)

Figure 4.1 Project plan editors tools that are used in the project.
The technologies that will be used within the project are also compared to the employee’s experience when choosing certain tasks. The input within the project plan editor for a technology is shown in figure 4.2.

![Figure 4.2 Project plan editor technology.](image)

The following sections explain how each development task is decomposed and how the information is extracted from the project plan and the employees’ profiles.
4.1 Assigning Requirement Tasks

The project manager can get advice from PMA on who the best employees are for gathering and writing functional requirements. This is done by looking at the project experience that the employee has in writing requirements, the time spent by the employee in writing functional requirements and the time spent on education along with the tool. These are cross referenced against the tool used within the project plan for requirements. In figure 4.3 the screen that allows an employee to enter functional requirement experience is shown.

![Employee Resume Editor for LLOYD CROMWELL](image)

**Figure 4.3 Employee experience with functional requirements.**

Each employee’s project experience for functional requirements is referenced against the tool within the project plan. The amount of experience the employee has is calculated using an ABLE ruleset. A value is created for each employee where each one is ranked.

The more experience and education in writing requirements with the specific tool within the project plan, the
higher this employee will be rated against other employees. Along with a ranking of each employee a concern is also shown. For instance, if there is no one available with experience PMA will inform the project manager that this is a concern and that the employee requires education. In figure 4.4 the advice is shown to the project manager.

![Figure 4.4 Employee ranked for functional requirements task.](image)

It is up to the project manager to decide the software engineer(s) that will gather and write the functional requirements. PMA has the ability to dynamically assign employees to do functional requirements thus simplifying the manual work for the project manager. The entry ‘staffing’ in the project plan is referenced to see how many Software Engineers are required to do the task of functional requirements. Once the project is started meaning a project plan has been created and employee experience entered the functional requirements, requirement tasks can be dynamically assigned. Figure 4.5 shows the staffing entry in project plan where the skills needed is indicated as ‘Gather and write functional requirements’. 
The automatic task assignment can be set within the preferences of PMA. When a software engineer is assigned this task they are required to use the requirements tool that is referenced within the project plan. A requirements tool can also be used outside of PMA. This is covered in section 5. Once the software engineer(s) have completed writing the requirements document they will inform the project manager through PMA by marking this task complete. The software engineer(s) will submit the requirements document to the project manager for review within PMA. The project manager will then look to see who the required reviewers are and schedule a time for a review. Each required reviewer must read the requirements document thoroughly where comments are brought to the review meeting. Once the review is done the software engineer(s) must answer all comments. When all comments are answered the software engineer(s) will notify the project manager that this development milestone is complete.

The project manager has the option to assign software testers the task of writing a test plan from the requirements. This will allow them to implement black box test cases.
with the functional requirements and the GUI functional requirements. In parallel, the project manager will look at the functional requirements and assign design tasks to different software engineers.

4.2 Assigning Design Tasks

The project manager can get advice from PMA on who the best employees are for developing a design from the requirements document. This is done by looking at the project experience that the employee has in design. The time they spent writing design documentation and time spent on education along with the tool are cross referenced against the tool used within the project plan for requirements. In figure 4.6 the screen that allows an employee to enter design documentation experience is shown.

![Employee Resume Editor for LLOYD CROMWELL](image)

*Figure 4.6 Employee experience with design documentation.*

Each employee’s project experience for design documentation is referenced against the tool within the project plan. The amount of experience the employee has is calculated using an ABLE ruleset. A value is created for each employee where each
one is ranked. The more experience and education writing design documentation with the specific tool within the project plan the higher this employee will be rated against other employees. There may be some instances where the employee has not as much experience as other employees however they are the only employee available. In this case a concern is shown by PMA that will inform the project manager that education should be required. Figure 4.4 is identical to what would be seen accept the software engineers would be ordered in the qualifications they would have for design documentation.

It is up to the project manager to decide the software engineer(s) that will write design documentation. Design documentation is based off of functional requirements. Functional requirements must be reviewed and marked complete before design documentation tasks can be assigned. When a software engineer is assigned a design documentation task the functional requirement must be referenced. The project manager has the capability of assigning the different software engineers design documentation. Figure 4.7 shows the screen that will allow the project manager to assign design tasks to a software engineer where functional requirements must be chosen.
PMA has the ability to dynamically assign employees to do design documentation thus taking out the need for manual work by a project manager. The project plans staffing is referenced to see how many Software Engineers are required to do the task of design documentation. The completed functional requirements are used for the software engineer to know what the design documentation should be written from. The functional requirements are broken up into sections when written. Each section is assigned to a different software engineer depending on how many are suppose to do design documentation. So once the project plan has been created, employee experience entered and functional requirements are complete the design documentation tasks can be dynamically assigned. Figure 4.3 shows the project plan editors staffing where the skills needed is indicated as ‘Write design documentation’.

When a software engineer is assigned this task they are required to use the design tool that is referenced within the project plan. A design tool was created that runs outside of PMA that can be used. This is covered in section 6. Once the software
engineer has completed writing the design document they will inform the project manager through PMA by marking this task complete. When informing the project manager the design document is submitted for review. The project manager will then look to see who the required reviewers are and schedule a time for a review. Each required reviewer must read the design document thoroughly where comments are brought to the review meeting. Once the review is done the software engineer(s) who wrote the design document must answer all comments. When all comments are answered the software engineer(s) will notify the project manager that this development milestone is complete.

Once design is complete the project manager will assign software engineers to write the code from the design document. They also have the option to assign software testers the task of writing a test plan from the design document. The software tester can look at the pseudo-code within each method of the design classes to write white box test cases.

4.3 Assigning Code Implementation

The project manager can get advice from PMA on who the best employees are to implement code. This is done by looking at the project experience that the employee has in code implementation and comparing it to the project experience input within the employee experience tool. The programming language, time spent writing code, time spent on education along with the tool are cross referenced against the tool used for coding is entered within the employee experience tool. In figure 4.8 the screen that allows an employee to enter code implementation experience is shown.
Each employee’s project experience for code implementation is referenced against the tool within the project plan. The amount of experience the employee has is calculated using an ABLE ruleset. The code repository is also taken into consideration. A software engineer may have experience saving code into a code repository like CVS which could be used within the project. This can be entered above within the employees programming experience. The code repository used for the project can be entered in the configuration management screen shown in figure 4.9.
A value is then created for each employee where each one is ranked. The more experience and education implementing code with the specific tool within the project plan, the higher this employee will be rated against other employees. Figure 4.2 is identical to what would be seen accept the software engineers would be ordered in the qualifications they would have for code implementation.

It is up to the project manager to decide the software engineer(s) that will implement code. When assigning this task the project manager must inform the software engineer of what design classes they are required to implement. To do this design documentation is required to be reviewed and complete.

PMA has the ability to dynamically assign employees to do code implementation thus taking out the need for manual work by a project manager. The project plans staffing is referenced to see how many Software Engineers are required to do the task of code implementation. Once the project plan has been created, employee experience entered and design documentation is reviewed and completed the code
implementation tasks can be dynamically assigned. Figure 4.3 shows the project plan editors staffing where the skills needed is indicated as ‘Write code’.

When a software engineer is assigned this task they are required to use the IDE tool that is referenced within the project plan. Once the software engineer has completed writing the code and unit testing they will inform the project manager through PMA by marking this task complete. The project manager will then have the option of either having a code review with a number of members from the team or to begin test execution. In the case that a code review is held the required reviewers must be notified. The project manager will create a schedule time. Comments must be submitted to the software engineers who created the code. It is the software engineer(s) responsibility to answer those comments. Once all comments are completed the software engineer will inform the project manager. After this test execution can occur.

4.4 Assigning Test Plan Creation

The project manager can get advice from PMA on who the best employees are for writing test cases. This is done by looking at the project experience that the employee has in writing test cases that is required to be inputted within the employee experience tool. The total number of test cases written, total time to create test cases from functional requirements, total time to create test cases from design document and tool used are all required to be inputted within the employee experience tool. In figure 4.10 the screen that allows an employee to enter test case experience is shown.
The tool that is used within the project plan for writing test cases is compared to the tool that the employee has used in past projects. The total number of test cases written from functional requirements is calculated for the employees each past project and compared to other employees. This is used only in the case that functional requirements have been completed and the project manager has assigned software testers to write test cases. The total number of test cases written from design documentation is used when a project manager assigns a software tester to write test cases based off of the design. The amount of experience the employee has in writing test cases is calculated using an ABLE ruleset. A value is created for each employee where each one is ranked. The more experience and education writing test cases from either the functional requirements or design document with the specific tool within
the project plan the higher this employee will be rated against other employees. Figure 4.2 is identical to what would be seen accept the software engineers would be ordered in the qualifications they would have for writing test cases.

It is up to the project manager to decide the software tester(s) that will write test cases. When assigning software testers to do black box testing the functional requirements document must be reviewed and completed. When assigning software testers to do white box testing the design documentation must be reviewed and completed. The requirements document and design document is where test cases can be created.

Since the requirements document has functional requirements and GUI functional requirements broken down by section the project manager can assign certain functional requirements sections for the software tester to write test cases against. In the case that white box testing is required the project manager will assign design classes to the software tester so that test cases can be implemented.

PMA has the ability to dynamically assign employees to write test cases thus taking out the need for manual work by a project manager. The project plans staffing is referenced to see how many software testers are required to do the task of writing test cases. The project plan must already exist with the staffing section filled out. Employees experience must also be entered and either functional requirements or design documentation must be reviewed and completed before a project manager can assign a software tester to write test cases. Figure 4.5 shows the project plan editors staffing where the skills needed is indicated as ‘Write test cases.

When a software tester is assigned this task they are required to use the testing tool that is referenced within the project plan. A test case tool was created that runs outside of PMA that can be used. This is covered in section 7. Once the software tester(s) have completed writing the test plan they will inform the project manager through PMA by marking this task complete. The project manager will then look to see who the required reviewers are and schedule a time for a review. Each required reviewer must read the test plan document thoroughly where comments are brought to the review meeting. Once the review is done the test plan owner must answer all
comments. When all comments are answered the software tester(s) will notify the project manager that this development milestone is complete.
4.5 Assigning Test Plan Execution

The project manager can get advice from PMA on who the best employees are to run test cases. This is done by looking at the project experience that the employee has in running test cases that is required to be inputted within the employee experience tool. The total number of test suites run and the tool that was used is required to be inputted within the employee experience tool. Figure 4.11 is the screen that allows an employee to enter test cases run.

![Employee Resume Editor for LLOYD CROMWELL](image)

Figure 4.11 Running test case experience for an employee.

The tool that is used within the project plan for running test cases is compared to the tool that the employee has used in past projects. The total number of test cases written from functional requirements is calculated for each employee’s past project and compared to other employees. The amount of experience the employee has is
calculated using an ABLE ruleset. A value is created for each employee where each one is ranked. The more experience of running test suites the higher this employee will be rated against other employees. Figure 4.2 is identical to what would be seen accept the software testers would be ordered in the qualifications they would have for running test cases.

It is up to the project manager to decide the software tester(s) that will write test cases. This task can only be assigned once the test plan has been reviewed and completed. Certain test cases are assigned to different software testers.

PMA has the ability to dynamically assign employees to run test cases thus taking out the need for manual work by a project manager. The project plans staffing is referenced to see how many software testers are required to do the task of running test cases. The project plan must already exist with the staffing section filled out. Employees experience must also be entered and the test plan must be reviewed and completed before a project manager can assign a software tester to run test cases. Figure 4.3 shows the project plan editors staffing where the skills needed is indicated as ‘Run test cases’.

When a software tester is assigned this task they are required to use the testing tool that is referenced within the project plan. A test case tool was created that runs outside of PMA that can be used. This is covered in section 7. The software tester(s) after being assigned the task to run the test cases will generate a report that has the test case results. It is the project manager’s responsibility to analyze the bugs and assign a certain software engineer the task of fixing the bug. Eventually, after all bugs are fixed the project manager will reschedule the test plans to be run again. This process can keep iterating in the case that bugs keep reoccurring. However, within the test plans execution report the bugs do have a severity. A project manager in some instances may decide to only re-run the test plan for a bug that has a high severity and to let the application be released with low severity bugs.
5. Requirements Editor

Within PMA requirements must be entered using the requirements editor. This task is assigned by the project manager to one or more software engineers. When entering the requirements within the requirements editor, the software engineer(s) must also enter the required reviewers. Once the gathering and writing of requirements are completed the project manager will be updated that the requirements are ready for review. The project manager will then take a look at the required reviewers and schedule a review. Once the review is completed the requirements will be marked complete. The Project Manager will then assign testing tasks to the Software Testers where they will use the requirements document to write their test cases.

Figure 5.1 Project Manager assigning requirement and testing tasks to Software Engineer(s) and Software Tester(s).
When entering each requirement the estimated time it takes to complete design, implementation and testing of the requirement can be entered. This information helps the project manager estimate the schedule for the project. Since time estimation is very important additional functionality is placed within the requirements editor tool that will use function point analysis and DeMarcos Model.

5.1 Function Point Analysis

Traditionally, project managers estimated the total time for an application by looking at how many Lines Of Code (LOC) would be written. The problem with this estimation is that no one really knows how many lines of code it will take to write an application after the requirements have been established.

It was discovered that instead of using lines of code to estimate a project, it made more sense to analyze the functional requirements. This application will allow a user to input each functional requirement by adding unadjusted function points to the Function Point Analysis. More details of Function Point Analysis can be found in J. Brian Dregers book titled Function Point Analysis [3]. Figure 5.2 shows what is required to be input when creating an unadjusted function point.

**Figure 5.2 Unadjusted Function Point.**
Each unadjusted function point consists of a business function name, business function description, business function category along with data items referenced, files referenced or logical record format relationships. The business function category can be described as an output, inquiry, input, file or interface. The following are defined below:

- **Outputs** -- items of business information processed by the computer for the end user. Outputs will deal with Files that are referenced.
- **Inquiries** -- considered a simple output; more precisely, they are direct inquiries into a data base or master file that look for specific data, use simple keys, require immediate response, and perform no update functions. Inquiries will deal with Files that are referenced.
- **Inputs** -- items of business data sent by the user to the computer for processing and to add, change, or delete something. Inputs will deal with Files that are referenced.
- **Files** -- data stored for an application, as logically viewed by the user. Files deal with logical record format relationships.
- **Interfaces** -- data stored elsewhere by another application but used by the one under evaluation. Interfaces deal with logical record format relationships.

A data item referenced is a piece of data that is neither printed nor stored in an organized data set, file or table. The number of data items referenced must be chosen. A file reference is a file that stores or accesses data. File references are only shown if the business category is a data input, output or inquiry. A logical record format relationship is a logical record such as a database that stores or accesses data. This is only shown to the user if they have selected a data input or interfaces or files.

Once the information is entered the unadjusted function point can be calculated. The algorithm is different for each business category. Table 5.1 shows how an unadjusted function point value is calculated.
Table 5.1 Unadjusted function point calculation.

A value is assigned to the unadjusted function point depending on the number of files referenced and the number of data items referenced.

Once the Unadjusted Function Points are added to the application, some adjustment factors must be selected.

Below are 14 adjustment factors and their explanations.

- **Data Communications** -- Data communications means that data or control information used in the application is sent or received over data communications facilities -- this not only includes various networks, concentrators, multiplexers, and private lines, but also includes terminals locally connected. On-line systems, therefore, will always have at least some data communications influence.

- **Distributed Data or Processing** -- Distributed data or processing indicates the application uses data stored, accessed, or processed on a storage or processing system other than the one used in the main program routines. Note that presence of this factor increases the data communications influence previously defined.
• Performance Objectives -- Application performance objectives influence system design, development, implementation, and support when specific, user-approved demands for exceptionally high throughput or fast response times have been made.

• Heavily-Used Configuration -- If the application will be run in a heavily used, tight, or crowded configuration, extra design and programming care is required to minimize storage access routines and swapping code segments in and out of main memory. This factor would be especially important to a user already lacking computer capacity but unable to purchase or acquire more hardware or upgraded software; the system is 'running out of gas'.

• Transaction Rate -- A high transaction rate can occur when the network consists of many data entry or inquiry terminals, when each screen transmitted contains a lot of input information, or when the frequency of screen transmission is high. Greater design efficiency is required due to the increased competition for line and central processor facilities.

• On-Line Data Entry -- On-line inquiry and data entry (including control and security functions) are always more difficult to accommodate than similar batch systems; hardware, application software, and operating system software are all affected by the additional requirements of an on-line system.

• End User Efficiency -- When end user efficiency is emphasized, more human-factor features are required; these are designed to increase the level of 'user-friendliness' and include such things as conversational data entry (requiring multiple sequenced screens), help screens, 'next format' fields, paging capabilities, more descriptive documentation (including users manuals and 'learner-friendly' training materials), second-language input/output screens and messages, and additional edit, error trap, and exception handling routines. If you are using structured systems analysis and design techniques, the boundary of the physical system -- the 'human-machine interface' -- is high.

• On-Line Update -- As on-line inquiry and data entry more difficult than batch, so is on-line update of files and data sets more difficult because of the short turnaround time and its widespread effects on all system design
components. On-line update requires non-sequential disk files and reliable transaction files for both production and audit/recovery purposes.

- Complex Processing -- Complex processing refers to the situation in which an application requires substantially greater than average difficulty in input or output processing; in logic, file, or numeric manipulation; or in exception handling routines. For example, an application requiring multiple control interactions and decision points would be classified as complex. So would one requiring extensive logical or mathematical equations, such as forecasting future sales using multiple forecasting models. As a third example, an application which sorts out many transactions for correction and reprocessing ('rework') would also be considered complex. Finally, processing requiring high security measures or sensitive controls would be accounted for here.

- Reusability -- Code reusability refers to the situation in which some of an application's routines, subroutines, or other procedures have been designed or written with users in mind other than just the program under evaluation. That is, portions of the application are specifically intended at the time of design or writing to be used and supported in other applications, perhaps even at other sites. Additional analysis, planning, co-ordination, and design is required to meet the unique needs of the other programs or processing sites.

- Conversion and Installation Ease -- Conversion and installation ease increases the difficulty of application development but reduces the number and severity of problems in testing and implementation. This factor may be evidenced by the presence of a well-designed conversion or installation plan which was fully tested and modified as necessary during the system test phase. Installations performed on the fly (which usually translates to 'winging it'), regardless of apparent ease, do not meet this standard.

- Operational Ease -- Operational ease is not the same as end user efficiency. This factor, like the previous, requires a well-defined plan that was fully tested and modified as necessary during the system test phase. Its purpose is to provide effective but easy startup, backup, error recovery, and shutdown procedures, and to minimize such manual activities as mounting tapes or
special forms, handling paper, or responding to requests for information at the operator console.

- **Multiple-Site Use** -- When the application has been specifically designed, developed, and supported for installation at multiple sites, for multiple organizations, additional coordination, review, and approval is required even if no site-unique code needs to be written. This factor may be distinguished from code reusability, in which only some of the application will contribute to a new and different program, perhaps only at the writing site; with multiple-site installation planning, the same application will be used at all sites. Any differences in business functions provided between the various sites would be accounted for in step one, business functions identification, not in this step, which is concerned with the application and production environment complexity as a whole.

- **Facilitate Change** -- When the application has been specifically designed, developed and supported to facilitate change, it requires increased attention to and planning for future maintenance and modification needs. For example, business information subject to change -- such as tax rates and shipping costs -- is organized into tables easily accessed and maintained by the user rather than hard-coded and scattered throughout the program. As another example, providing a flexible query capability facilitates future change because stored data or information is more easily identified as needing maintenance.

By using the Unadjusted Function Points and the 14 adjustment factors, the deadline for an application can be estimated using the following formula;

\[
\text{Final Adjusted Function Point} = \text{Total Unadjusted Function Points} \times (.65 + (.01 \times \text{Total Adjustment Factors}))
\]

Once the final adjusted function points are calculated the user can enter the value of how long it will take them to develop one function point. The default of this is 20 hours for common object oriented applications but could decrease depending on the experience of the developer. Increasing function points will increase the complexity
of the application. To determine the complexity, a user can input the percentage of complexity. This will increase each function point by the percentage of complexity. For example, if we have a final calculation of 2 for the adjusted function points and 20 hours is estimated for one function point with a complexity percentage of 2, the following would occur.

<table>
<thead>
<tr>
<th>Function Point</th>
<th>Formula</th>
<th>Hours</th>
<th>Total Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>--</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>((1 + \frac{2}{100}) \times 20)</td>
<td>20.4</td>
<td>40.4</td>
</tr>
</tbody>
</table>

**Table 5.2 Example of increased complexity.**

The number of hours for one function point will vary depending on the developer. Once developers go through a number of projects the number of hours for one function point can be more accurately defined. It is up to project managers to record these values for each project for particular developers so that project estimation can be more accurate. This will of course depend on each developer’s skill. Project managers perhaps can take an estimation of the developers past projects that can be used on the current project. Again, this is up to the project manager.

5.2 DeMarco Model

Additional cost estimation is done using DeMarco Model [16]. DeMarco model is based on function primitives, which are found within the lowest level of a data flow diagram. The required information for DeMarco model is included within each functional requirement making it easy to calculate. When entering each functional requirement inputs and outputs must be entered along with a weight. A total sizing for the functional requirement can be calculated by taking the weight of the functional requirement multiplied by the total inputs and outputs multiplied by log2 of the total inputs and outputs.

Total Sizing = Weight * <Total inputs and outputs> * (log2 * <Total inputs and outputs>)
The list of different weight selections along with the calculated value for each is shown in table 5.3.

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compose or decompose data</td>
<td>0.8</td>
</tr>
<tr>
<td>Update information</td>
<td>0.5</td>
</tr>
<tr>
<td>Analyze data and take action</td>
<td>1.0</td>
</tr>
<tr>
<td>Evaluate input data</td>
<td>0.8</td>
</tr>
<tr>
<td>Check for internal consistency</td>
<td>1.0</td>
</tr>
<tr>
<td>Text manipulation</td>
<td>1.0</td>
</tr>
<tr>
<td>Synchronize interaction with users</td>
<td>1.5</td>
</tr>
<tr>
<td>Generate output</td>
<td>1.0</td>
</tr>
<tr>
<td>Perform simple calculations</td>
<td>0.7</td>
</tr>
<tr>
<td>Perform complex calculations</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Table 5.3 Weight selections for each function type.**

Once all functional requirements have been entered the estimation of the project deadline can be calculated using the following formula:

\[
\text{Total Months for Project} = \text{ConstantA} \ast \text{Total Size} ^ \text{ConstantB}
\]

Where

\[\text{ConstantA and ContstantB can be adjusted depending on the experience of the development team. Past project information can be kept so that more accurate values can be created. For example, a development team that has worked together for 10 years may have a lower ConstantA than a team that is straight out of college with minimal technical skills.}

Figure 5.3 shows the calculation for each functional requirement and the total sizing after calculation occurs.
5.3 Estimation

Estimation of each development milestone can also be entered for each functional requirement. The development milestones that are defined is actual time to write functional requirement, design documentation complete, code implementation complete and test case creation and execution. This is based on each functional requirement. Eventually, all functional requirement estimations can be added to find the total time it would take to complete this application. Figure 5.4 shows each
functional requirement and the sizings entered along with the total value for the project within each development milestone.

Figure 5.4 Calculated estimates from each functional requirement.

The values entered for each functional requirement can be added together and compared against the DeMarcos Model calculations. This gives an opportunity for requirements writers to be more accurate next time when they input the estimated time for each functional requirement. Figure 5.5 shows the comparisons between the estimation that the requirements document author input and the sizings that were calculated using DeMarcos Model.
Figure 5.5 Comparison between estimation form requirements editor author and DeMarcos Model.

In order for the project manager to be able to come up with realistic schedules, requirements must be thoroughly analyzed. This allows realistic schedules to be set which will allow a more realistic budget. Many times within software development, requirements are not analyzed enough thus making it hard to come up with realistic schedules. By using theoretical cost estimation formulas such as function point analysis and DeMarcos model schedules can be methodically calculated.
5.4 Requirements Editor Additional Functionality

The requirements editor also provides additional facility to print the requirements into a pdf file or html file.

The requirements are input using a subset of the IEEE-830-1998 standardized format. GUI screens can be embedded within the tool, which can then be referenced by GUI functional requirements. Once finished they can be saved in a .rmap file where they can be distributed and loaded into the standalone requirements tool by another user. The requirements also can be saved in a pdf file and printed for review. Comments from a reviewer can be created and distributed back to the requirements document author(s). The comments can then be reviewed by the requirements document author(s) where resolution to the comment is saved within the document. The analysis of certain words within the requirements document is calculated. Certain words can be very ambiguous and are not recommended while there are more precise words to use.

5.5 Creating a Tentative Schedule from Requirements

Once the requirements are submitted the estimations are analyzed by the project manager to come up with a tentative schedule. Estimations are cross referenced against the project plans responsibilities where the hours per day and days per week are filled in for each employee. Figure 5.6 shows the responsibilities that can be entered for each employee within the project plan.
Figure 5.6 Responsibilities, hours per day and days per week that an employee will work.

The calculated schedule can be shown to the project manager. The calculated schedule consists of the following tasks that were gathered from the functional requirements document:

- design that will be done from the functional requirement document
- code implementation from the design document.
- test plan creation from the requirements document.
- test plan creation from the design document
- test plan execution

Figure 5.7 shows an example of a schedule that was generated from the functional requirements.
When generating the schedule the employee will be assigned functional requirement sections that consist of a number of individual functional requirements. It is the project manager’s responsibility to assign the tasks and track which functional requirements have been completed within each task. Once a task gets assigned to an employee the functional requirement will be cross referenced. This allows the project manager to make sure all functional requirements have been implemented at every development milestone stage. In some cases the project manager may not require the functional requirement to be implemented within particular test cases. However, the functional requirement should definitely be able to be tracked within the design documentation and the code implementation. By having a tentative schedule created the project manager has an idea of when the project can be completed depending on staff. In the case that the project manager sees a particular stage falling behind additional employees can be added.
6. Design Editor

After requirements are created and reviewed the next step is to create the design document. Once the project manager sees that requirements have been completely reviewed, he/she must assign software engineer(s) a task to write design documentation. The software engineer(s) assigned will use the design editor. The design editor will allow classes and methods to be created where functional requirements will be cross referenced. For each method in the design, pseudo-code can be written helping define exactly what the method will look like. The actual time it took to write the design classes is entered along with the estimation that it will take to do implementation and testing. This actual time is then used by the project manager to analyze the status of the schedule. After the software engineer(s) have completed writing the design documentation the required reviewers must also be added before the task is marked ready for review. When marked ready for review the project manager will be notified and will schedule a meeting with the required reviewers. Once the review is complete and the design document is updated the project manager will notify the software tester(s) that test cases can be written based off of the design. This is shown in figure 6.1.
Figure 6.1 Project manager assigning design tasks and testing tasks to Software Engineer(s) and Software Tester(s).

6.1 Design Editor Additional Functionality

Similar to the requirements editor, the design editor also allows a user to print the design into a pdf or html file. This allows other employees to have a hard copy to review. In addition, the author of the design document can save the design document in a .dnap file and send it to the required reviewers. The reviewers can load the .dnap file within the design editor and create review comments that can be sent back to the author. The author can go through all reviewed comments and answer them. When the author answers the reviewed comments it is required to put a resolution along with how long it took for them to answer all the comments.
6.2 Design Documentation Estimation vs. Actual Time

When the completed design document is submitted to the project manager after the review, the estimated time to develop the design document from the requirements document can be compared to the actual time it took. This will allow the project manager to see how well the author(s) were at estimating the design documentation milestone. The inputted estimates for the design milestone within the requirements document from the author can be used by the project manager to understand if the author is good at making estimates. In the case that the author is accurate schedules can be determined and maintained. However, when the estimates are not satisfactorily accurate, a project manager might not want the employee to calculate estimates within the requirements document. Estimations have a lot to do with experience of the employee writing the requirements document and also experience of the employee writing design documentation. In many cases both have to be taken into consideration by the project manager when the requirements and design documentation tasks are assigned. A project manager can probably trust an experienced team. However, when new employees are assigned certain tasks, this may not be that easy. The talent of the new employee is still unknown. This is why the requirements tool uses DeMarcos Model and Function Point Analysis to help the author create estimates. Over time the project manager will have a better feel of each employee along with the team. The author of the requirements document will be able to assign estimations better if they have worked with the employee developing the design.

6.3 Design Documentation Requirements

It is also the project manager’s responsibility to make sure all requirements are being considered within the design document. When a class is created within the design document it must reference the functional requirements that the class will implement. Once the design document is submitted to the project manager the completed requirements document is retrieved. The functional requirements from the requirements document are then compared to the cross referenced functional requirements from the design document classes. The project manager must
communicate any missing functional requirements to the design document authors. By doing this the project manager can guarantee to the customer that no functional requirements have been missed.

Once the design document is submitted for review, PMA will dynamically inform the design document author of any missing functional requirements by creating a comments file. Additional comments can be created by the project manager which is then sent to the design document author.
7. Test Editor

Once the requirements and design are complete, test cases can be written. This was shown above in figures 5.1 and 6.1. The test editor can be used for writing the test plan where test cases and test case steps can be inputted. The test cases are then cross referenced with the functional requirements and design classes. By being able to cross reference functional requirements black box testing can be done. Black box testing is done to ensure that all requirements are implemented within the application. By referencing design document classes white box testing can be done. White box testing ensures that the logic is working correctly.

The required reviewers need to be input after all test cases are completely written within the test plan. Once the test plan is completed it will be sent to the project manager who will schedule a review with all the required reviewers. It is the responsibility of the required reviewers to send comments to the test plan author. Once the test plan is updated from the comments given by the reviewers, the project manager will verify that implementation is complete. Test cases will be run manually and reports generated and sent to the project manager. Within the report bugs will be identified. It is the project manager’s responsibility to look at the bugs reported and assign the correct software developer to fix them. This process is shown in figure 7.1
Figure 7.1 Project Manager assigning tasks to Software Engineer(s) based off of testing results.

7.1 Test Editor Additional Functionality

After a test plan is completed the test plan author can save a .tnap file and send it to the project manager and other required reviewers. The .tnap file can be loaded into the test editor where the required reviewers can create comments. When done the comments can be extracted and sent to the test plan author. The test plan author can review the comments. While reviewing them a resolution to the problem along with the time it took to review the comment is inputted and saved at the end of the test plan.
7.2 Project Managers Test Plan Execution Decisions

Once the project manager is informed that code implementation is complete the software testers will be notified. The software testers can then load the test plan and manually execute each test case step. It is their responsibility to mark the test case step as a success or a failure. In certain circumstances a test case may fail. In this case a priority is set being either low, medium or high. After the test plan is completely executed it will be sent to the project manager. The project manager will look at the failed test cases and assign certain software engineers to fix the problem. Assigning the task of fixing bugs to a certain employee depends on if the project manager wants the owner of the code, owner of the design or owner of the functional requirement to take responsibility. There may be an individual on the team that looks at all bugs and then filters the ownership down to the correct owner. In some projects the owner of the code should not be assigned the bug because of possible stress. In other projects it makes sense for the owner of the code to take ownership of the bug which should mean a quick turnaround.

The project manager can manually assign the bug to the developer or PMA can look at the preferences set and assign the bug dynamically once the test plan report is generated. PMA will look at the preferences that are set by the project manager for maintenance. Since the project manager knows the entire past work of each individual it is easy to trace the owners of different tasks. The test case within the test plan also cross reference design document class and functional requirements making the owner even easier to track. The project manger has the following choices to assign the bug-fixing activity:

- Assign bug to code implementer
- Assign bug to design document author
- Assign bug to requirements document author
- Assign bug to certain employee (where the employee would have to be entered)
The person responsible for the bug will have to submit to the project manager if the problem occurred within the requirements, design or code implementation along with a description of the fix. After this is done, they will notify the project manager who will make a decision to execute the test plan. In certain circumstances a project manager may not want to execute the test plan again. For instance, if only low priority bugs were reported the project manager may not feel it is worth the time to fix them. The project may be close to being at the end of the negotiated project completion date and may have no time to execute the test plan again. The project manager may not have enough resources to execute the test plan. If they have another test plan execution occur the projects budget may be affected.

PMA allows the project manager to put this information in the testing preferences. In doing so, number of iterations and the priority of bugs are taken into consideration. The project manager might only want 2 iterations of test plan execution done no matter what. In some cases the project manager will want to execute the test plan until there are no bugs or perhaps only low priority bugs. PMA will look at this after the test plan results are sent to the project manager. Software engineers will dynamically be assigned based off the preferences set for maintenance.
8. PMA Metrics

Since PMA stores the software artifacts from different development phases, it also generates several metrics for later analysis. Metrics are crucial in understanding the status of a project. As an example, the project manager may find that the number of bugs reported from one particular employee’s work is somewhat the same for every project. In this case, the project manager may look into the metrics of bugs reported for each employee and take corrective action (may send the employee for proper training or education).

The project manager will schedule a review for requirements, design and test plans when the authors have notified them of completion. Once the author has resolved all comments from the required reviewers, the document will be submitted to the project manager. The project manager can then view how many comments were sent to the author, how many actual bugs were found from the comments, total time it took to write the requirements, design or test plan document, total time it took to review the document and the total time it took to resolve the comments.

The first thing a project manager will look at is the number of bugs found from the reviewer comments. If this number is higher than expected, the project manager can have a feeling that it was carefully reviewed. The total amount of time that the reviewers took can be compared against the number of comments and the number of bugs. The more time that was spent on the review the more comments should have been submitted thus leading to more bugs found. This is the first thing that the project manager wants to verify. There may be some circumstances where this would not be true. Perhaps the authors of the requirements, design or test plan documents were very good but the reviewers might have spent a lot of time reviewing and would have come up with minimal comments.

The total time it took to write a document can also be compared to the total time it will take to review the document. A document that took more time should take comparable amount of time to review. In the case that the reviewers did not spend a lot of time reviewing a large document means that the review might not have been done very well. In this situation the project manager may not be surprised when bugs are found later on within the software development lifecycle. A project manager may
even have a second review if he/she does not feel comfortable with the statistics from the first review.

Each software development phase should be looked. The total time it took to create the document and the total time it took to review the document should be compared against the estimates. By looking at this, the project manager will have a good understanding of how accurate the estimations are. Figure 8.1 shows the estimates against the total time it took to create each document and the total time it took to review.

![Figure 8.1](image_url)  

**Figure 8.1 Document creation and review time vs. the document estimation.**

Within each software development phase certain metrics can be analyzed. We will go through each software development milestone and see what metrics are gathered.

The requirements document can be analyzed for certain weak phrases or options. For example, the word ‘can’ gives an option to the individual who implements the design. The author of the design document is not sure if the requirement should be implemented. All they know is that they can do it if they want to. Weak phrases or options should be minimized within the requirements document. The project manager can look at the number of weak phrases or options within the requirements
document. The more weak phrases or options the greater the ambiguity of the document. The project manager can view the requirements document before it is reviewed by the reviewers where comments can be sent to the author of the requirements document to get rid of weak phrases or options. The requirements editor will actually show the author all of the weak phrases or options. In figure 8.2 a screen shot is shown which allows the author of the requirements editor to view the weak phrases or options. There are other categories of words that are monitored within the requirements document. This consists of imperatives, directives and continuances. The higher the number of such words used in the requirements document, the higher is the quality of the document, according to NASA [17].

![Figure 8.2 Requirements editor analysis of categorical words.](image)
Along with the analysis of words functional requirements can also be analyzed. The total estimated time of each development phase can be divided by the number of functional requirements. This will return the average amount of time to complete one functional requirement. This gives the project manager a good understanding of how big the project is.

The same metrics can be done within the design document where the total estimated time for the coding and testing stages can be divided by the number of classes. This will give the project manager the average amount of time it takes to implement a class from the design document within the coding and test plan stages.

The most popular metric is the defect. The defect can be monitored when a test execution report is sent to the project manager. The defect can be traced back to either the requirements, design, code implementation or the test case creation. Once identified this can be tracked to the owner of that particular development stage where the defect occurred. The more defects for a particular employee the more concern a project manager has on that individual’s performance.

From the metrics on defects, the time it takes to fix a defect can also be analyzed. In many circumstances the maintenance of an application is more than the actual time it took to develop. If this starts to occur the project manager will understand the quality of the product. Possible re-design will have to occur thus costing the project more time and money. Figure 8.3 shows a functional requirement and the actual time it took to implement it for each development stage in minutes.
Figure 8.3 Looks at one functional requirement and the total time in minutes it took to implement within each development stage.
9. Example Case Study

We will revisit our case study from section 1.1. This example is a real case scenario taken from a graduate computer science class where 3 members were assigned to a particular team. The project consisted of writing a game scheduler application that would allow a schedule to be dynamically created for any league. We will break down what was decided without using PMA and what could have happened when using PMA.

9.1 Employee Experience Breakdown

On day 1 the roles were discussed and broken up. I will refer to the students as Student1, Student2 and Student3. Table 9.1 shows the roles of each student.

<table>
<thead>
<tr>
<th>Development Milestone</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Plan</td>
<td>Student1</td>
</tr>
<tr>
<td>Requirements Document</td>
<td>Student1</td>
</tr>
<tr>
<td>Design Document</td>
<td>Student2</td>
</tr>
<tr>
<td>Code Implementation</td>
<td>Student3</td>
</tr>
<tr>
<td>Test Plan Document</td>
<td>Student1, Student2</td>
</tr>
<tr>
<td>Test Plan Execution</td>
<td>Student1, Student2</td>
</tr>
<tr>
<td>Maintenance of Code</td>
<td>Student3</td>
</tr>
</tbody>
</table>

Table 9.4 Role break down.

The owners were in charge of the documents and all students were required in the reviews. The roles were verbally communicated between each other. It was decided that since Student1 had large experience writing requirements and was doing his thesis on this particular development phase, it made sense for him to do this work.
Student2 was working on an Independent Study that dealt with design documentation so it made sense for him to be the owner of the design document. We all agreed that most of the work would be done by the person doing the code implementation and that the language of choice would be Java. Since Student3 had a large experience in Java it made sense that he would be assigned that role. Test Plan documentation and execution was assigned to Student1 and Student2. Student3 had a lot of experience with testing. However, the completion of the project had to be done in 3 months and it only made sense for Student3 to maintain the role of implementing and maintaining the code while Student1 and Student2 would do the test plan.

This was verbally decided at the beginning. However, after entering in each students experience PMA would have had the same conclusion. To do this within PMA the project manager must create the project. Figure 9.1 shows the sign on screen to PMA where a project can be created, a user can join a project or a user can log into a project.

![Project Manager creates the project.](image)

Once the project is created, other individuals must join the project. There are 3 different user types which include project manager, software engineer and software tester. A project manager will be in charge of assigning tasks, making sure the
project is on schedule and within budget. Additional communication can be done by the project manager between members of the team. A software engineer can be responsible for requirements and design documentation along with code implementation and maintenance. The software tester’s responsibility is to create and execute test plans. There may be some situations where one individual is only going to do software testing and nothing else. In this scenario they would only have to join the project once. However, in the case that one individual will act as a project manager, a software engineer and software tester they will have to join the project three separate times for each user type. When joining the project employee experience can be imported or manually entered. When importing the experience the stand alone employee experience tool can be used. If one does not exist the individual can input past projects. Figure 9.2 shows Student2 joining the project.

![Figure 9.2 Software Engineer joins the project.](image)

Once all students have joined the project the project manager must make sure the project plan information is inputted. This can be done directly within PMA. Figure 9.3 shows the project plan editor within PMA. Information from the project plan can be exported and sent to others, imported from the stand alone project plan tool or saved within our database.
Figure 9.3 Project Plan information.

Once the project plan is inputted the project manager can ask PMA for particular advice. Figure 9.4 shows the project manager logged in asking for requirements document advice.
Figure 9.4 Project Manager asking for functional requirements advice.

Not only can the project manager select advice on functional requirements, but can also get advices on design and code. This will take a look at the students’ past experience along with what is required within the project to see who the best individual is for the particular task. From this advice the project manager can manually assign the task of writing functional requirements. The project manager can view each employee’s past project experience to draw the conclusion when assigning tasks. Figure 9.5 shows the employees experience.
In the case that the project manager does not want to evaluate each student when assigning tasks PMA will dynamically do this. It will look at the required employees to do a certain task within the project plan along with the required skills and compare the employee’s information. Tasks will be dynamically assigned. By looking at the project plan only one student will be doing functional requirements. The most talented student is picked which is Student1. Student1 will log in and get a message that they need to write the functional requirements. Student1 will accept this responsibility. The communication of this is shown in figure 9.6.
Figure 9.6 Student1 accepting the responsibility.

Student1 can reject this if he/she decides. If this occurs the project manager will get a reply saying that Student1 rejected and why it was rejected. It is then the responsibility of the project manager to assign the task to another employee. However, Student1 accepted this responsibility. The project manager will get a message back saying that it was accepted. The project manager will then wait till the requirements document is ready for review. Warning messages will be sent to Student1 before the date requirements should be reviewed so that Student1 is aware of the timeline. Once Student1 has completed the requirements document for review they will go into PMA and select ready for review. When selecting ready for review a dialog will be shown asking Student1 to attach the requirements document. This should be a .rmap file which will then be dispersed to the required reviewers. It is the responsibility of the required reviewers to send the comments back to Student1 who will integrate them into the requirements document. Once all comments have been reviewed Student1 will mark this as being completed which will again ask for the .rmap file. Student1 will attach this which will be used by the project manager to come up with an estimated schedule.

An estimated schedule can be generated by the project manager getting scheduling advice. Figure 9.7 shows this.

Figure 9.7 Scheduling Advice
An estimated schedule will be shown to the project manager that can be used to make sure is done with each development phase on time.

Once the schedule is created the project manager can assign design tasks to Student2 along with test plan tasks to Student1. When assigning the test plan tasks PMA will look at the project plan to see how many individuals are needed to assign test cases based on the requirements document. Since there is only one, Student1 is chosen. PMA does have the ability to assign design tasks by functional requirement section. However, since the project plan has only one student developing design document this is not needed. The entire design document is then assigned dynamically to Student2. Student2 will accept this responsibility and when done advise the project manager that they are ready for review. When writing the design document Student2 is required to use the design editor which will produce a .dnap file that must be sent to the project manager. This is similar to figure 9.6 accept the tasks deal with design document instead of functional requirements document. The project manager will schedule a review and notify the required reviewers. Once Student2 resolves all comments from the required reviewers the design document will be marked complete. PMA will again look at the project plan and see one person will have the responsibility to write code and one person will have the responsibility of writing test cases based on the design. The project manager will then dynamically assign Student2 the responsibility of writing test cases based on the design document. Student3 will then be assigned the code implementation.

When assigning code implementation the same functionality will exist that is in figure 9.6. Student3 will accept responsibility of the task. When done he will notify the project manager. However, there will be no document sent to the project manager. Instead it is optional on where the code is stored. Student3 may have used a code repository or is just using a pc to store the code on. It is not required in PMA to submit the code for review. This is optional by the project manager. A code review can be scheduled if the project manager feels that it is needed. However, if this is the case PMA does not support the communication of where the code is. In the
case that no code is required to be reviewed this task will be marked complete where the software testers will be notified that test plan execution can occur.

While code implementation is occurring Student1 and Student2 are writing test plans that will eventually need to be reviewed. Student1 accepted to do the black box testing that is based on the functional requirements. Student2 accepted to do the white box testing that is based on the design document. When done Student1 and Student2 notified the project manager who scheduled a review with all the required reviewers. Student1 and Student2 updated their test plans from the comments they received from the review and notified the project manager. The project manager now is aware that test plan execution can start at anytime once Student3 is done coding.

Student1 and Student2 will accept this and when done will send the test plan execution to the project manager. PMA will dynamically review the results to see where bugs were found. PMA will look at the preferences to see who should dynamically be assigned the task of looking at this bug. In this case study Student3 was chosen who is the individual implementing the code. Student3 will accept the task and then notify the project manager that the bug was fixed. The preferences are then looked at by PMA to see if test plan execution should occur again.

During all of this the project manager is comparing the schedule from the requirements and monitoring the tasks results. The schedule may need to be calculated again once design documentation or code implementation is complete.
10. Conclusion

This thesis describes the design and implementation of a tool called Project Manager’s Assistant (PMA). It assists a project manager in several decision making processes during a software project’s life cycle. It simplifies several manual tasks that will otherwise consume a lot of time for a project manager such as selecting an employee for a particular phase of development and assigning the task to that employee.

PMA will solve this problem by finding the correct individual for the task and assigning them right away depending on which development phase is completed. A schedule is put in place after requirements and monitored by PMA. The individual assigned to a particular phase or task will be notified before the expected completion. This makes sure each individual is aware of the schedule and the date that is required to be met.

The use of PMA In a typical project was illustrated through a course project which involved three graduate students. Within the case study all students were required to communicate on where they were at within their particular task. Verbal communication was definitely the key to effectively maintain the three-month deadline. PMA takes this kind of communication out of the picture. Tasks can be assigned dynamically and status reported back manually. PMA will also remind the individual owner of the task.

PMA also enforces a software team to follow a disciplined development process that consists of requirements, design, code implementation and testing. In some circumstances, this is not the case. Depending on schedules, some development phases may be overlooked. This is not recommended within PMA because it compromises the quality of the project. PMA also ensures that all requirements stated in the requirements document are addressed in each development phase. It provides a tracking mechanism by which a design or code fragment or test case can be traced.
back to a requirement. This guarantees the quality of the product making sure requirements are not missed.

11. Other Project Management Software

As a part of this literature study, a number of project management software applications were found. Rational Portfolio Manager (RPM) is a tool used that allows tasks to be assigned to certain individuals users. Users must be created and tasks assigned with a start date and a completion date along with additional information about the task. This information is then put within a Gannt chart so that the project manager can view the project timeline. Any user can input documents within RPM that pertain to something within the project. The scope of the project along with risks are also maintained within RPM. Dashboards can also be created from the project information and distributed to executives so that they can understand the status of the project and where the project is heading. PMA focuses more on managing the software development life cycle where it is required to have certain stages of development. For example, requirements gathering and documentation is required within PMA where RPM does not require a project manager to assign the task of requirements gathering. The same holds true for design documentation, code implementation or testing.

RPM has the ability to create a portfolio which consists of a number of projects while PMA focuses on only one project at a time. RPM also has up to 60 different macro reports that can be run on particular projects while PMA has a limited amount of metrics that can be run. Much of the detailed information such as managing dependencies, assets and proposals are built within RPM where this information is kept within the project plan of PMA. RPM allows individual documents to be created and stored for any task while PMA only is concerned with documents that are focused on software development lifecycle tasks.
Another project management tool that was found is called @task. This allows roles to be defined and created. For example a role of a project manager can be defined. Timesheets can be created for each user. Instead of tasks a user will create issues. Any user can assign an issue to another user. In PMA the project manager is the only one who assigns tasks but in @task anyone can assign issues. Different issues that can be assigned include submitting a new issue, requesting an issue, changing the order of an issue or creating a bug.

There are a number of other project management tools available in the market. Most of them consisted of creating projects, defining employees to a project and then creating tasks. There was no project management tool that was found that followed a close development process lifecycle like PMA does. The user of this other project management software could enter a generic task to do such a thing as requirements documentation or design documentation. However, there was no estimation done from the requirements documentation, design documentation or code implementation within a project. Also there was no ability that I found in any project management software that would assign responsibilities or tasks to a particular user. There was no other project management software that would keep a cross reference of requirements throughout the development life cycle like PMA does.
12. Continuing Work

The following is the continuing work on this project. I have broken it down by each tool.

12.1 Employee Experience Tool

The employee experience tool will be able to take the inputted information and allow the user to save it as a .pdf file in the format as a resume. This will be useful so that an individual can fill out their experience in the format that PMA needs and also allow them to send their resume to other employers.

12.2 Project Plan Tool

Once the project plan is inputted it will be able to be saved as a .pdf file where it can be shared to other people curious about the project.

12.3 Requirements Editor

Currently there is no plans of enhancing the requirements editor.

12.4 Design Editor

Eventually, the Napkins Requirements Tool file (.rnap) will be imported so that the user can make sure all functional requirements are referenced within the design document. Additional work will be made to view the design document within a pdf file. This functionality will be similar in the way that the Napkins Requirements Editor can save its data to a pdf file.

12.5 Test Plan Editor

Eventually, the Napkins Requirements Tool file (.rnap) will be imported along with the design document tool file (.dnap) so that the user can make sure all functional requirements and design classes are referenced within the test plan. Additional work will be made to view the test plan within a .pdf file. This functionality will be similar in the way that the requirements editor can save its data to a .pdf file.
12.6 PMA

PMA still needs to add the testing functionality of having the project manager receive a test plan execution report and assign a particular developer the task of fixing the bug. Also schedules need to be compared between each development milestone. For instance, the design document has the estimates for coding and testing which can be compared against the values within the requirements document. Also, the software developer writing the code must be able to enter how long it took them to do each class and should submit a report of the classes they created along with where they are located. This could be used by the project manager if a code review was required.

The additional functionality of allowing a miscellaneous task to be assigned to user should also be implemented.
13. Bibliography


APPENDIX A
IMPLEMENTATION NOTES
PMA uses an Eclipse Plug-in client and interacts with the ABLE Platform which uses RMI. There are also stand alone clients that can be used within where xml files can be saved and imported within PMA. The stand alone tools consist of the requirements editor, design editor, test plan editor, employee experience editor and the project plan editor. The employee experience editor and project plan editor uses the Standard Widget Toolkit (SWT) for its GUI while the rest of the stand alone applications use Java Swing for the GUI. All stand alone clients run within a web browser using the Java Network Launch Protocol (JNLP). The employee experience and project plan editors are built into PMA as Eclipse plug-ins giving the user the ability to either use the stand alone tools or the Eclipse plug-in to enter the employee experience and project plan information. All information from the stand alone tools can be imported within PMA which saves the information within a database. The database that is used is running mySql to store the project information. A number of tables were created to help store the assigned tasks and document information.

The users table will have all users that have been inputted within PMA. It will allow us to see what project they current are on, their name, whether they are a project manager, software engineer or software tester and if they are available. This is shown in table 14.1.

<table>
<thead>
<tr>
<th>Column Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectName</td>
<td></td>
</tr>
<tr>
<td>UserName</td>
<td></td>
</tr>
<tr>
<td>UserType</td>
<td></td>
</tr>
<tr>
<td>Available</td>
<td></td>
</tr>
</tbody>
</table>

Table 14.1 Users Table.
The project table will have all the project information. Since this is used by the client we need the server IP address and server port of this project along with the name and the description. The project plan, requirements document review, requirements document complete, design document review, design document complete, test plan review and test plan complete are all XML documents representing those particular files. At anytime a member of the project can view this documentation by extracting it and loading it in the correct tool. The project table is shown in table 14.2.

<table>
<thead>
<tr>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerIpAddress</td>
</tr>
<tr>
<td>ServerPort</td>
</tr>
<tr>
<td>ProjectName</td>
</tr>
<tr>
<td>ProjectDesc</td>
</tr>
<tr>
<td>ProjectPlan</td>
</tr>
<tr>
<td>RequirementsDocumentReview</td>
</tr>
<tr>
<td>RequirementsDocumentComplete</td>
</tr>
<tr>
<td>DesignDocumentReview</td>
</tr>
<tr>
<td>DesignDocumentComplete</td>
</tr>
<tr>
<td>TestPlanReview</td>
</tr>
<tr>
<td>TestPlanComplete</td>
</tr>
</tbody>
</table>

**Table 14.2** Project.

The employee experience can be saved within PMA. The employees name is saved along with the past project information. Note that the project data is stored as an XML file which is identical to the .experience file that can be saved within the experience tool. Table 14.3 shows this.
Task information is assigned by the project manager for a particular employee. When this is done the user is assigned a task for a particular project. Any additional information is kept along with the status and the date. An example that would be stored within this table is a user writing requirements documentation. The tasks table is shown in table 14.4.

Since design tasks must cross reference a functional requirement this also must be stored. We need to make sure who is responsible for this so the task Id is stored which has the employees information along with the functional requirement id. The design tasks table is show in table 14.5.
Since code tasks must cross reference the design document the class definition must be stored. We need to make sure who is responsible for this so that the task Id is stored which has the employee’s information along with the class definition name. The code tasks table is show in table 14.6.

<table>
<thead>
<tr>
<th>Column Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskId</td>
<td></td>
</tr>
<tr>
<td>ClassDefinitionName</td>
<td></td>
</tr>
</tbody>
</table>

**Table 14.6 Code Tasks Table.**

Test plan creation must be assigned to the software tester where functional requirements and class definitions must be kept track of. This allows PMA to cross reference who was in charge of writing test cases for each functional requirement from the requirements document and each class definition which is from the design document. This is shown in table 14.6.

<table>
<thead>
<tr>
<th>Column Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskId</td>
<td></td>
</tr>
<tr>
<td>FRId</td>
<td></td>
</tr>
<tr>
<td>ClassDefinitionName</td>
<td></td>
</tr>
</tbody>
</table>

**Table 14.7 Test Plan Tasks Table.**

Test plans may be executed a number of times. Because of this a separate table was created that has the project name along with the test plan execution date and the
test plan execution results which is a large XML file with the results. The test plan execution can be loaded within the test plan editor. This is shown in table 14.8.

<table>
<thead>
<tr>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectName</td>
</tr>
<tr>
<td>TestPlanExecutionDate</td>
</tr>
<tr>
<td>TestPlanExecution</td>
</tr>
</tbody>
</table>

Table 14.8 Test Plan Execution.