

ASSESSMENT AND RECOMMENDATION OF REQUIREMENT MANAGEMENT TOOL
BASED ON USER NEEDS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Aditi Mohpal

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

November 2012

Fargo, North Dakota

North Dakota State University
Graduate School

Title

ASSESSMENT AND RECOMMENDATION OF REQUIREMENT

MANAGEMENT TOOL BASED ON USER NEEDS

By

ADITI MOHPAL

The Supervisory Committee certifies that this *disquisition* complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall Nygard

Chair

Dr. Simone Ludwig

Dr. Gursimran Walia

Dr. Chanchai Tangpong

Approved:

11/08/2012

Date

Dr. Kenneth Magel

Department Chair

ABSTRACT

Over recent years, requirement management has played a very vital role in materializing IT projects. Whether it is the healthcare sector, finance sector, banking sector, or any other industrial sector, requirement management is crucial for the successful completion of IT projects. Due to a tight time to market and severe competition, requirement management becomes one of the most critical activities for the software-development lifecycle process [1]. Requirements change all the time during the software-development lifecycle. Unmanaged changes lead to project overruns. Most companies are using requirement-management tools to manage requirements. However, many companies end up choosing a requirement-management tool that does not address the project's needs and problems. In this research, requirement-management tools available on the market have been identified, and a "decision-making application" has been developed for systematic evaluation of requirement-management tools based on user needs so that a project team can identify and choose the correct requirement-management solution.

ACKNOWLEDGEMENTS

I would like to express my gratitude by thanking everyone who has supported me during the completion of this research paper. I would like to thank Dr. Kendall Nygard for guiding me all these semesters. His support and vital encouragement contributed a lot in completing my research. I would also like to thank the Computer Science department for offering Requirements Management as a core course that helped me complete different chapters for my research paper. I would like to thank my project manager, Jennifer Wong, who gave me an insight into the “requirements change management” process and its importance for managing requirements. I would like to thank Stephanie Sculthorp (Administrative Secretary) of the Computer Science department for constant reminders and motivating me to complete my paper on time. Last but not least, I would like to thank my parents, Manju Mohpal and Shiv Prakash Mohpal, for guidance, love, much needed support, and making it possible to study at North Dakota State University.

TABLE OF CONTENTS

| | |
|--|------|
| ABSTRACT | iii |
| ACKNOWLEDGEMENTS | iv |
| LIST OF TABLES | vii |
| LIST OF FIGURES | viii |
| LIST OF ABBREVIATIONS | ix |
| CHAPTER 1. INTRODUCTION | 1 |
| 1.1. Problem Description | 1 |
| 1.2. Motivation..... | 3 |
| 1.3. Research Question and Major Contribution | 4 |
| 1.4. Paper Organization..... | 6 |
| CHAPTER 2. REQUIREMENT MANAGEMENT..... | 8 |
| 2.1. Requirement-Management Concepts..... | 8 |
| 2.1.1. Capturing Requirements that Drive the Design, Implementation, and Testing of Software to Ensure that the Final System Fulfills the End User’s Needs..... | 8 |
| 2.1.2. Organization and Documentation of Requirements..... | 8 |
| 2.1.3. Requirement Traceability..... | 9 |
| 2.1.4. Manage Requirement Change..... | 9 |
| 2.1.5. Team Communication..... | 10 |
| 2.1.6. Requirement Accessibility | 10 |
| 2.2. Why Do We Need a Requirement-Management Tool?..... | 10 |
| CHAPTER 3. SELECTING THE REQUIREMENT-MANAGEMENT TOOL | 13 |

| | | |
|--|--|-----------|
| 3.1. | Tool Options | 13 |
| 3.1.1. | IBM Rational RequisitePro | 13 |
| 3.1.2. | HP Quality Center | 14 |
| 3.1.3. | Gatherspace | 14 |
| 3.1.4. | Contour | 14 |
| 3.2. | Guidelines for Selecting the Requirement-Management Tool | 14 |
| CHAPTER 4. RESEARCH APPROACH FOR DEVELOPING THE DECISION MAKING APPLICATION | | 16 |
| 4.1. | Features in the Requirement-Management Tool..... | 16 |
| 4.2. | Requirement-Management Tool Evaluation..... | 20 |
| 4.3. | Tool Selection | 22 |
| 4.4. | User Decision..... | 24 |
| 4.5. | “Decision-Making” Application and User Analysis..... | 28 |
| CHAPTER 5. RESEARCH RESULTS..... | | 39 |
| CHAPTER 6. FUTURE WORK AND CONCLUSION | | 43 |
| 6.1. | Conclusion | 43 |
| 6.2. | Future Work | 43 |
| REFERENCES | | 45 |

LIST OF TABLES

| <u>Table</u> | <u>Page</u> |
|---|-------------|
| 4.1. Requirement Feature Categories..... | 20 |
| 4.2. Requirement-Tool Evaluation..... | 21 |
| 4.3. Total Normalized Score Calculation..... | 22 |
| 4.4. Case Scenarios for Utility and Cost..... | 25 |
| 5.1. Cost and Utility (Score) for Each of the Four Tools Based on a User’s Priority Selection for a Tool’s Category and Its Features. | 40 |

LIST OF FIGURES

| <u>Figure</u> | <u>Page</u> |
|--|-------------|
| 1.1. Third-Party Logistic Provider Database Tool..... | 5 |
| 1.2. Result of Logistics Provider..... | 6 |
| 2.1. Requirement Traceability..... | 9 |
| 4.1. Utility vs. Cost Plot for Contour and Gatherspace. | 25 |
| 4.2. Utility vs. Cost Plot for Contour and Requisite Pro..... | 26 |
| 4.3. Utility vs. Cost Plot for HP Quality Center and Contour. | 27 |
| 4.4. Solution Explorer MVVM Components..... | 30 |
| 4.5. Class Diagram Decision-Making Tool. | 31 |
| 4.6. Category and Feature Definition in XML..... | 32 |
| 4.7. Tool and Its Feature Addition in an XML File..... | 32 |
| 4.8. User Interface..... | 33 |
| 4.9. User Rating Section. | 34 |
| 4.10. Pie Chart: Normalized Score. | 35 |
| 4.11. Pie Chart: Normalized Score per Unit Cost..... | 36 |
| 4.12. Compare Features in Tools..... | 36 |
| 4.13. Score vs. Cost Plot. | 37 |
| 4.14. Tools and Features. | 38 |
| 5.1. Utility vs. Cost Plot for each of the Four Tools Based on a User's Priority Selection for a Tool's Category and Its Features..... | 39 |
| 5.2. First Dominant Front Points Marked in Red..... | 41 |
| 5.3. Second Dominant Front Points Marked in Red. | 42 |
| 5.4. Non-Dominant Front..... | 42 |

LIST OF ABBREVIATIONS

| | |
|-----------|--|
| CCB..... | Change Control Board |
| CM..... | Change Management |
| SDLC..... | Software-Development lifecycle |
| RE..... | Requirement Engineering |
| RM..... | Requirement Management |
| QC..... | Quality Control |
| RTM..... | Requirement Traceability Matrix |
| XML..... | Extensible Markup Language |
| XAML..... | Extensible Application Markup Language |
| MVVM..... | Model View ViewModel |

CHAPTER 1. INTRODUCTION

The chapter highlights the Problem Description and briefly describes the Motivation behind the research. The chapter closes with an outline of this paper along with its major research contribution.

1.1. Problem Description

The problem is to develop a “decision-making application” that would guide a project manager, a system analyst, or any other IT team member in comparing and selecting the correct requirement-management tool for projects.

The application should meet the following objectives:

- Users should be able to rate features of the requirement-management (RM) tool based on how important those features are for their projects.
- Based on a user’s rating, the decision-making application should show the graphical representation of normalized scores for different tools from which a user has to choose.
- The decision-making application should also show a graphical representation of the score per unit cost for tools that a user is comparing to help the users with tool selection.
- The application should plot a graph of score vs. cost so that the user can make a subjective decision based on how much a customer is willing to pay for features in the RM tool.

An organization is made up of two broad units, a business unit and an IT unit. The business unit always comes up with a need or a problem for which it wants a solution. For the solution, the business unit approaches the IT unit. The IT team provides solutions for business problems.

Business units are governed by different rules and regulations depending on the sector to which they belong and the country in which they operate. Due to changes in the government, finance, and economic rules and regulations, businesses evolve; therefore, the software systems supporting these businesses need to change. For e.g., [2]“The Centers for Medicare and Medicaid Services (CMS) and the National Center for Health Statistics (NCHS), two departments within the U.S. Federal Government’s Department of Health and Human Services (DHHS) provided different guidelines for coding and reporting using the International Classification of Diseases, 10th Revision, Clinical Modification (ICD-10-CM).” All different healthcare firms, whether they were providers or payers, had to conform to this new regulation and had to enhance all their software systems that supported ICD-9 to support ICD-10.

Change is inevitable sometimes. It is wise to implement requirement changes after a careful impact and risk analysis at the beginning of software development life cycle (SDLC) instead of waiting until the end to see a project fail.

If the requirements are not managed and documented well, the IT team suffers a lot when it is time to enhance the existing application. The project team can manage requirement changes effectively and efficiently if the team conducts a rigorous impact analysis based on the traceability matrix. Requirement-management tools help with the successful completion of IT projects on time and within budget.

Companies use a variety of requirement-management tools available on the market for large system-development projects. Most companies fall prey to marketing advertisements and unrealistic claims made by vendors selling these requirement-management tools. Companies do not establish clear objectives and needs before buying a tool. After purchasing the tool, they discover that the tool cannot be customized, making it hard for project teams to enforce their

processes, or that, when the vendor did a tool demo, the tool looked great, but now it does not apply to their real world.

Careful selection of the requirement-management solution is very important. These tools have diverse features and can be applied to projects differently. Requirements change throughout the software-development process; therefore, changing a requirement-management tool in the middle of the project is not wise and can prove to be very costly. If companies end up with the wrong requirement-management tool, they not only pay high licensing costs, but also suffer loss as their objective of managing requirements is not fulfilled. Therefore, there is a critical need to evaluate and to select the correct requirement-management tool based on project needs.

1.2. Motivation

I worked at different companies as an IT intern. I realized that, no matter what sector to which a company belongs, requirement management is very critical to project success. I remember one of my friends calling me and telling me how the project he was working on got scrapped after 5 years of effort. At the end of 5 years, a customer told the company, “This is not what we want.” After doing a root-cause analysis for the project failure, the company found that it happened because requirements were not clarified, communicated, and managed effectively because the requirement-management tool being used lacked the capabilities for notifying team members about changing requirements and did not manage the requirement-change history. The incident of project failure really affected me because I could have been in that situation of working for 5 years just to realize, in the end, that my effort and all weekends sacrificed for my project team were unfruitful. Careful selection of the requirement-management tool is, therefore, very critical.

1.3. Research Question and Major Contribution

The main research question addressed by this paper is as follows: What approach should a user take to select the RM tool, and how will the “decision-making application” be built so that the user selects the correct requirement-management tool based on the project needs?

There is a lot of existing work on the evaluation and assessment of RM tools for different industries. Eric D. Clark [3] wrote a thesis analyzing a variety of requirement-management tools utilized by shipbuilders. The thesis discussed the best practices of requirement management and evaluated four requirement-management tools: Analyst Pro 5.3, Core 5.1, Cradle 5.3, and Doors. The author listed the tools and compared them based on 10 different criteria; he ranked the tools based on weighted criteria to evaluate what tool was best for the shipping industry.

Other research was done by Rajat R. and James D. Arthur [4]. They researched requirement-management tools and a qualitative assessment of those tools. They listed some questions that project teams should ask before selecting a requirement-management tool.

Although all the authors [4] [3] did a very good job identifying the RM tools and comparing them based on criteria, no application was developed to help users do an analysis of RM tools. Project needs can differ from company to company and from industry to industry. Project designers might want to consider the 20 different tools available on the market to see what tool best fits their project needs.

There are many applications developed on the market in other areas that take a user’s input and recommend a solution based on the need. One such application is a database tool that helps a user find a third-party logistics provider based on the user’s need. However, the application is not user friendly. It does not provide enough information for user that would help when selecting a provider. There are no quantitative data provided to user that would show why

a provider was suggested by the database tool [5] .

Figure 1.1 shows the user interface for the logistics tool. A user selects different criteria by checking the check-boxes, however, there is no option for the user to rate the criteria by importance. The logistics tool has a database that stores all the third-party logistics providers. The tool returns the list of providers to a user, however, the tool does not have an option for the user to compare the providers to see what unique criteria they have among them.

Figure 1.2. shows the results a user obtains on a different page when the user hits the “Find 3PL Providers” button shown in Figure 1.1. The result is not dynamic; the user has to go back and forth to change selections that were made previously to obtain the new results.

To find companies that match your specific needs, check as many boxes as apply from the criteria below.

| General Information | |
|--------------------------------------|--|
| <input type="checkbox"/> Asset-based | <input type="checkbox"/> Non asset-based |

| Areas Served | |
|--|---|
| <input type="checkbox"/> Asia | <input type="checkbox"/> Middle East and North Africa |
| <input type="checkbox"/> Southeast Asia & India | <input type="checkbox"/> South America |
| <input type="checkbox"/> Europe | <input type="checkbox"/> United States |
| <input type="checkbox"/> Eastern Europe & Russia | <input type="checkbox"/> Canada and Mexico |

| Certifications | |
|---|------------------------------------|
| <input type="checkbox"/> SmartWay Transport Partner | <input type="checkbox"/> Lean |
| <input type="checkbox"/> ISO | <input type="checkbox"/> Six Sigma |
| <input type="checkbox"/> C-TPAT | <input type="checkbox"/> LEED |

| Markets Served | |
|--|--|
| <input type="checkbox"/> Manufacturing | <input type="checkbox"/> Distributors/Wholesalers |
| <input type="checkbox"/> Retail/E-Business | <input type="checkbox"/> Service Industries/Government |

| Vertical Specializations | |
|--|---|
| <input type="checkbox"/> Aerospace | <input type="checkbox"/> Electronics |
| <input type="checkbox"/> Agriculture | <input type="checkbox"/> Energy (Renewable) |
| <input type="checkbox"/> Apparel & Textile | <input type="checkbox"/> Food & Beverage |

Find 3PL Providers

Figure 1.1. Third-Party Logistic Provider Database Tool.

Find a Third-Party Logistics (3PL) Provider

You searched for:

- **General Information: Asset-based**
- **Areas Served: Eastern Europe & Russia**
- **Logistics Services: Global Trade Services**

24 results found.

<< Try Another Search Send RFP to Selected Companies

| | |
|--|---|
| A&R Logistics Phone: 800-542-8058 Website: http://www.arlogistics.net Average Customer Size: All sizes | ADD TO RFP <input type="checkbox"/> |
| APL Logistics Phone: 602-586-4800 Website: www.aplogistics.com Average Customer Size: All sizes | ADD TO RFP <input type="checkbox"/> |
| Big Dog Logistics Phone: 713-996-8171 Website: www.bigdoglogistics.com Average Customer Size: All sizes | ADD TO RFP <input type="checkbox"/> |
| Corporate Traffic Logistics Phone: 904-727-0051 Website: www.corporate-traffic.com Average Customer Size: All sizes | ADD TO RFP <input type="checkbox"/> |

Figure 1.2. Result of Logistics Provider.

The main idea in this research is, therefore, to develop an application that will guide a user in selecting the correct RM tool and that will serve as an informative decision-making tool for user. The idea is also to develop this application in such a way that it can be extended to compare multiple tools with different RM features in the future.

1.4. Paper Organization

Excluding this chapter, the remainder of the paper is organized in the form of five chapters.

Chapter 2 focuses on the Requirement-Management Concepts and identifies the importance of the requirement-management tool in the software industry.

Chapter 3 discusses four different requirement-management tools. The chapter also highlights general guidelines that a user should keep in mind when selecting a requirement-management tool.

Chapter 4 describes the approach taken to develop a decision-making application. The chapter begins by describing different features that a RM tool can have. The chapter then

discusses three different options from which a user can choose to select the best tool for his needs. The first option is selecting a tool based on the tool's feature score; the second one is selecting a tool based on the per-unit cost; and the third one is selecting a tool based on Pareto Optimality.

Chapter 5 reports the Research Results and builds on the insights gained in Chapter 4.

Chapter 6 concludes the paper by summarizing the major result. In addition, several techniques are described for improving the decision-making tool and are intended as a guide for future researchers who wish to extend and build upon this paper.

CHAPTER 2. REQUIREMENT MANAGEMENT

The chapter discusses different requirement-management designs and briefly describes the need for a requirement-management tool.

2.1. Requirement-Management Concepts

2.1.1. Capturing Requirements that Drive the Design, Implementation, and Testing of Software to Ensure that the Final System Fulfills the End User's Needs

Capturing good requirements is a very important aspect of requirement management. The gathering requirement is not a one-time effort. It is a continuous process. In order to have successful requirement management, it is very important that customers are involved throughout the software-development lifecycle. Customers are the only source who can tell IT team whether the requirements that we are capturing are correct. From my experience, I know that some analysts are very good at eliciting requirements from end users. The requirement elicitation skill comes from experience. While gathering requirements, the main focus should be to make sure the solution (requirement) addresses the problem the customer is facing.

2.1.2. Organization and Documentation of Requirements

Requirements cannot be managed if they are not organized and documented well. Therefore, it is crucial to prioritize them. Prioritizing requirements based on risk is very important. It is imperative to identify the risk associated with each requirement, and it is also important to assess the severity and consequences for each risk. The requirement with a very high-level risk should receive high priority, etc. There can be many other factors that guide the prioritization process because every project is different and because the goal for each is different.

2.1.3. Requirement Traceability

Requirement traceability, as shown in Figure 2.1, suggests that the requirements should be traced to other artifacts, such as design, test artifacts, and user-documentation plan.

Establishing traceability plays a very important role in complex projects because there are thousands of items involved and because establishing traces help project teams with impact analysis whenever there is a requirement change. Requirement traceability, therefore, helps the team save time and costly rework. [6]

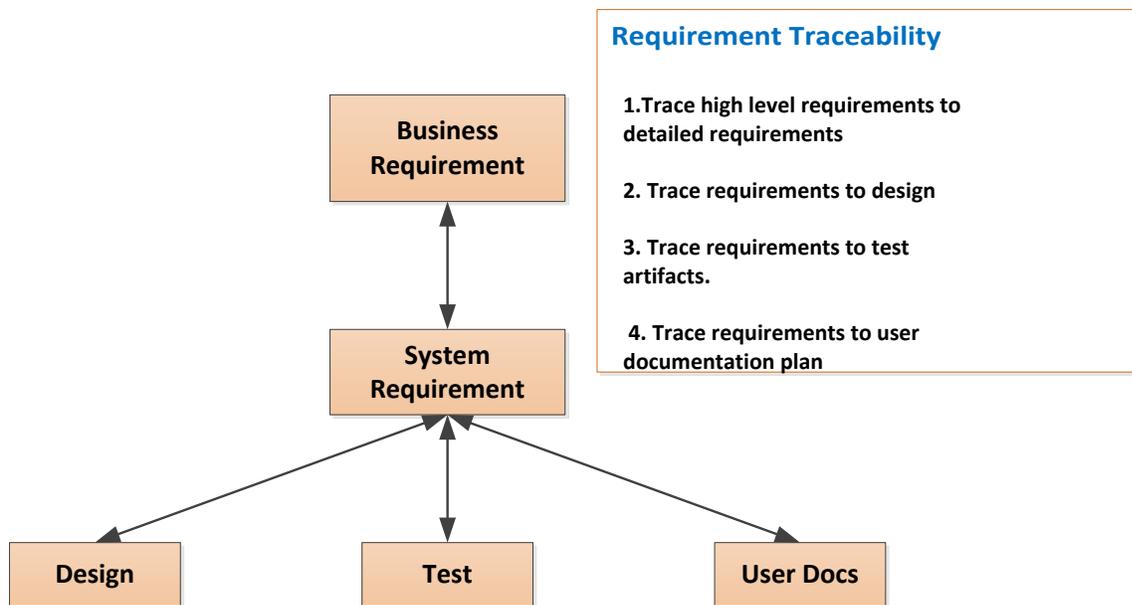


Figure 2.1. Requirement Traceability.

2.1.4. Manage Requirement Change

As a business analyst or a team member, I would hope to get all definite requirements at one time from my customers so that developers can work on to build a system. However, this scenario is not always the case. Requirements change throughout the SDLC process. A customer may say, “I don’t need this feature anymore.” Another customer may say, “I need a new feature,”

or “I need modification to an existing feature.” Any deletion, addition, or modification of the requirements can impact different artifacts. Therefore, managing change becomes important. Change management can be done effectively if we have established a requirement traceability plan that defines the requirement traces with different artifacts of different phases in the SDLC as depicted in Figure 2.1.

2.1.5. Team Communication

Project teams are not always at the same location. They can be spread across multiple geographies. The main idea here is to stay connected. Depending on the team size, project information can be communicated to team members through meetings, phone calls, documents, or a tool like “WebEx” to help keep the team connected.

2.1.6. Requirement Accessibility

For instance, a user communicated a change, and only the development team was notified, not the testing team. Developers will change the code, but testers will not update the test cases to examine the functionality that has been now changed. This miscommunication can lead to a very severe problem. Again, it is important that all team members have access to the most current requirements. This scenario is possible only if there is a central repository that is maintained and is up to date.

2.2. Why Do We Need a Requirement-Management Tool?

Many times when two or more business analysts work on the same requirement document, one of them labels a requirement document as version 1.2 only to find that the second business analyst has also made edits to version 1.1; now, there are two variations of version 1.2

that have to be merged. Often, team members are unsure whether requirements are complete or if some of requirements are missing in the Word document or the Excel document.

Many times, it also happens that, for a single project, there are more than 100 requirements and that, when a requirement Word or Excel document is sent to 15 different business units, they suggest more than 50-60 changes in 30 different emails. It becomes a nightmare to track all changes in different versions of the Word or Excel document and to conduct an impact analysis for the changing requirements.

Some of the above challenges are the reasons why project teams look for a requirement-management tool. Requirement-management tools promote communication, collaboration, and requirement-change management.

Requirement-management tools are critical to project success. Today, most teams are all over the globe, and the tool acts as a central repository for different teams to access the requirements. Teams do not need to exchange emails and spend time on the phone trying to contact a person for a requirement document. With the tool, teams look at the current requirements all the time. There is no confusion about different versions of the requirement document.

Requirement-management tools allow source requirements to be traced at a detailed level, thereby facilitating impact analysis.

To summarize, the main reasons why project teams should consider using requirement-management tools are as follows:

- To manage versions and changes.
- To manage the requirements using requirement attributes such as priority, status, difficulty, stability, source, etc.

- To ensure end-to-end traceability between different requirement types.
- To ensure that proper access-level rights are given to team members for editing different requirements, thereby restricting the editing rights by role on a team.
- To generate reports related to the requirement changes and requirement attributes.

CHAPTER 3. SELECTING THE REQUIREMENT-MANAGEMENT TOOL

Selecting a requirement-management tool is not an easy task. It is a very time-consuming process when it comes to evaluating different features that meet a project's need. This chapter highlights some of the RM tools that create, retrieve, update, and delete requirements and that help trace requirements to their source. A rigorous approach to requirement capturing and tracking is needed when it comes to huge and complex system development. Customers are happy only when we deliver the product that meets their expectations.

3.1. Tool Options

Because the number of requirement-management tools on the market is huge, the study did not include all of them. Four tools were chosen for the study:

- IBM Rational Requisite Pro
- HP Quality Center
- Gatherspace
- Contour

3.1.1. IBM Rational RequisitePro

“Rational RequisitePro is a requirement management tool that was originally developed by Rational but then was acquired by IBM. The tool helps project teams to manage their requirements, and other deliverables like use cases and system requirements, design documents, and many more. RequisitePro combines both document-centric and database-centric approaches. It integrates Microsoft Word with a multi-user database and allows user to organize, prioritize, trace relationships, and easily track changes to the requirements.” [6]

3.1.2. HP Quality Center

“HP Quality Center has a requirements management solution module. It enables creation of different requirement types and provides a centralized location for managing requirements. The tool provides real time visibility of requirements coverage and associated defects so that project can evaluate business risks. It provides creation of reporting templates for extracting information from the repository. The reports can be exported to Excel for further manipulation.” [7]

3.1.3. Gatherspace

“Gatherspace™ is an intuitive web-based On-Demand requirements management solution that promotes collaboration between business and technical teams in managing changing requirements throughout the software development and product lifecycles.” [8]

3.1.4. Contour

“Contour is made by Jama Software, it is a web-based tool for requirements management and collaboration. Contour provides a consistent way of managing requirements and their changes to avoid omissions or duplications. Contour is collaboration software for development teams tackling complex projects.” [9]

3.2. Guidelines for Selecting the Requirement-Management Tool

When a user wants to select a requirement-management tool for his project, it is very important for a user to understand the tool itself.

- The user should go through all the features and functionalities of the new tool that he is willing to buy or implement for requirement management so that he can see if the tool fits the existing requirement-management process.

- It is important for the user to understand the current process for managing requirements in the absence of a tool. If the requirement-management process is effective, the process will have a positive influence on the efficiency of the tool and vice versa.
- It is also important for the user to determine how the existing requirements would be migrated to the new tool.

Some questions that need to be answered include:

- Which artifacts or deliverables need to be moved to the new tool?
- Where are the artifacts and deliverables documented (MS Word, Excel)?
- Who will migrate the artifacts in the tool?
- Who will make sure the migration was successful?
- Who will make sure that the artifacts are correctly imported in the RM tool?

When selecting a requirement-management tool, it is also important to know the need or business problem that a team is trying to solve. A very high-level goal, whether it relates to improving quality, meeting deadlines, communicating and collaborating, or any other problem, must be identified. Once the goal is clear, it is easy to select the requirement-management tool.

CHAPTER 4. RESEARCH APPROACH FOR DEVELOPING THE DECISION MAKING APPLICATION

This chapter is solely devoted to the main research task- to build a decision making application that would guide user in selecting the right requirement management tool.

In order to build the application, the very first step was to list important features that a RM tool should have and to group these features into categories.

4.1. Features in the Requirement-Management Tool

For the study, 27 requirement-management features are identified. Requirement management is very crucial for a successful project because organizations rely more on software for completing the projects.

The requirement-management features were identified based on industry experience. Organizations spend a huge amount of time and money reworking activities, leading to cost overruns. Most rework effort is directed towards correcting requirement-management information. In order for organizations to manage requirements, they have to make sure they capture stakeholder needs, communicate requirements to all team members, prioritize and organize requirements, and manage changes throughout the software-development lifecycle.

Companies that do not use requirement-management tools argue that email, face-to-face meetings, phone calls, and web meetings are sufficient for managing requirements. However, these methods do not ensure that requirements, especially the requirement changes that occur throughout the development process, are communicated to all parties.

Although twenty seven features have been identified, the idea is to build an application in such a way that future researchers can extend the application by simply adding more features to

the XML file which will be discussed later in the chapter. The RM features for the study are listed below:

- Requirement Creation in tool: The tool should allow the user to create/write requirements directly in the tool.
- Import Requirements from Word: If a user has documented requirements in MS Word, the tool should allow users to import requirements from Word to the tool, helping users save time because they will not have to rewrite all requirements in the tool again.
- Import Requirements from Excel: Many companies write their requirements in MS Excel. If a company wants to start using a requirement-management tool, the tool should allow users to import requirements from MS Excel.
- Custom Requirement Attributes Creation: Requirements can be managed using different attributes such as status/priority, etc. The tool should allow users to create different attributes for different requirement types.
- Association of Attribute to Individual Requirement: The tool should allow the user to associate attribute values to each individual requirement.
- History of Requirement Text Change: The tool should automatically store the history of a requirement before and after it is changed.
- Requirement Modified By: The tool should automatically store the history of users who modified the requirement text.
- Requirement modified date: The tool should automatically store the date and time when the requirement text was changed.
- Traceability between different requirement types: The tool should allow the user to trace source-level requirements to detailed-level requirements.

- Impact Analysis: The tool should assist users in doing impact analysis whenever requirements are modified.
- Online Change Notification: Whenever any modification is done to requirements, the tool should send an online notification to tool users regarding the change.
- Requirement-Level Security: The tool should allow the project administrator to specify which users can edit what requirements, and who has read-only rights at the requirement level.
- Package-Level Security: The tool should let the user define read-only/edit rights at the folder or package level.
- Project-Level Security: The tool should let the administrator define read/write rights at the project level.
- Portability and Backend Compatibility (Operating System): The tool should be compatible with different operating systems and other vendor products.
- Static Report: The tool should show results about the project at the present time.
- Trend Analysis Report: The tool should use time-sensitive filters to analyze changes in requirements, traceability, and attributes and to generate reports.
- Thick Client: The tool should have a thick client for installing software on a machine.
- Web Client: The tool should have a web-based version so that the artifacts in the tool can be accessed globally.
- Typo Spelling Error: The tool should be able to check for typing errors in the requirement text.
- Duplicate Requirements: The tool should be able to check for duplicate requirements.

- Requirement Query Based on Attribute: The user should be able to query and filter requirements based on attributes.
- Requirement Query Based on Traceability: The user should be able to filter requirements based on requirement traceability.
- Requirement Baseline: The tool should allow the creation of a baseline (snapshot) for project requirements.
- Comparison of Baselines: The tool should be able to compare different versions of the requirement-baseline documents.
- Tool Support: The vendor should be able to provide tool support for configuration/customization and any other support.
- Cost: The user should be able to select the range for the cost/user/month that he is willing to pay.

The features listed above are some of important features that a user would want for a project's requirement-management tool. All the features listed in this section are categorized into 14 different categories. Each category groups related features from the list. Categorization is done so that it becomes easy for a user to rate the related features. The fourteen categories are shown in Table 4.1.

Table 4.1. Requirement Feature Categories

| Category# | Category Name |
|------------------|--|
| 1 | Requirement Creation |
| 2 | Manage Version and Changes |
| 3 | Requirement Management with Attributes |
| 4 | Change Management |
| 5 | Security Level |
| 6 | Portability and Backend Compatibility |
| 7 | Report Generation |
| 8 | Tool Installation Type |
| 9 | Inconsistency Check |
| 10 | Query Support |
| 11 | Traceability to Other System Elements |
| 12 | Avg. Cost/User/year |
| 13 | Requirement Baseline and Comparison of Baselines |
| 14 | Tool Support |

4.2. Requirement-Management Tool Evaluation

The second step in the approach was to identify different features for each of the four RM tools. This information is important because it serves a basis for tool-score calculation and for calculating the features' per unit cost for developing an application.

Table 4.2. lists the features for each tool.

Table 4.2. Requirement-Tool Evaluation

| # | Tool → Features | Rational Requisite Pro | HP Quality Center | Gatherspace | Contour |
|-----|--|------------------------------|-------------------------|------------------|------------------|
| 1. | Requirement Creation Directly in Tool | X | X | X | X |
| 2. | Import Requirements from MS Word | X | X | - | X |
| 3. | Import Requirements from Excel | X | X | X | X |
| 4. | Custom Requirement Attributes Creation | X | - | - | X |
| 5. | Association of Attribute to individual requirement | X | X | X | X |
| 6. | History of Requirement Text Change | X | X | X | X |
| 7. | Requirement Modified By: | X | X | X | X |
| 8. | Requirements modified date | X | X | X | X |
| 9. | Traceability between different requirement types | X | X | X | X |
| 10. | Impact Analysis on requirement change | X | - | - | X |
| 11. | Online Change Notification | - | X | - | X |
| 12. | Requirement level security | - | X | - | - |
| 13. | Package level Security | X | X | X | - |
| 14. | Project level Security | X | X | X | X |
| 15. | Portability | X | X | X | X |
| 16. | Backend Compatibility | X | X | X | X |
| 17. | Static Report | X | X | X | X |
| 18. | Trend Analysis Report | X | X | - | X |
| 19. | Thick Client | X | - | - | X |
| 20. | Web-based Client | X | X | X | X |
| 21. | Typo/Spelling error | - | X | - | - |
| 22. | Duplicate Requirements | - | X | - | - |
| 23. | Requirement Query based on Attribute | X | X | X | X |
| 24. | Requirement Query based on Traceability | X | X | X | - |
| 25. | Requirement Baseline | X | X | X | X |
| 26. | Comparison of baselines | X | - | X | X |
| 27. | Tool Maintenance and Support | X | X | X | X |
| 28. | Cost/yr./user | \$5020.00 | \$5400.00 | \$3540.00 | \$3120.00 |

4.3. Tool Selection

The third step in the approach is calculating the normalized score and the scores (utility) per unit cost. The features are classified in different categories. A user can rate features on a scale of 0-4; “0” indicates the least-important feature that is not needed by a user, and the rating increases to “4” which indicates the most important feature that a user wants in the tool.

In our study, the features are used to evaluate similar tools. We have four RM tools for our study. In order to calculate scores for each tool, the application does not assign ratings to the features. The user decides how he wants to rate each feature within a category.

Calculation of normalized score: In the example in Table 4.3., the user rates “Feat 1,” “Feat 2,” and “Feat 3” as 4, 3, and 4, respectively.

Table 4.3. Total Normalized Score Calculation

| Req# | Features | Rating Score (0-4) |
|--------------------|---------------------------------------|---------------------|
| Category 1 | Requirement Creation | |
| Feat 1 | Requirement creation directly in tool | 4 |
| Feat 2 | Import requirements from MS Word | 3 |
| Feat 3 | Import requirements from Excel | 4 |
| Total Score | | 11 |

If, in the above scenario, HP Quality Center has all three features, “Feat 1,” “Feat 2,” and “Feat 3,” the total utility value, or score, for the HP Quality Center tool would be **11**. If, in the above case, Requisite Pro tool has just two features, “Feat 1” and “Feat 2,” the total utility value, or score, for Requisite Pro would be **7**.

$$\text{Normalized \% Score for Tool 1: } (\text{Score for Tool 1} / \sum \text{score of all tools}) * 100$$

Based on this calculation for each tool, the decision-making application would show graphical representation (in the form of a pie chart) to illustrate the distribution of normalized scores for different tools. If a user has to select between Requisite Pro and HP Quality Center based only on the normalized score, irrespective of cost, then the user would select HP Quality Center over Requisite Pro because HP Quality Center would have a normalized % score of 61.11% $((11/18) * 100)$ over Requisite Pro which would be 38.88% $((7/18)*100)$.

Calculation of Normalized scores per unit cost: In the study, the cost for all four tools is based on one concurrent license/user/year. This expense is a fixed cost.

Calculation of Normalized Score per Unit Cost: $(\text{Normalized Score}/\text{Cost of tool})$

If we reconsider the example stated in Section 1.4, the normalized utility value, or score per unit cost, for HP Quality Center is .00611 $(33/\$5400)$. The normalized utility value, or score per unit cost, for Requisite Pro is .00418 $(21/\$5020)$.

Based on this calculation for each tool, the decision-making application would show graphical representation (in the form of a pie chart) to illustrate the distribution of normalized scores per unit cost for different tools. In this case, if the user wants to choose a tool based on the utility per unit cost, then the user will select HP Quality Center above the Requisite Pro tool because the utility value per unit cost for the HP Quality Center is higher than that for Requisite Pro.

However, in the real world, there can be a lot of scenarios. A user might want to select a tool which is a little costlier but offers more features, or a user can have a situation where there are two or more tools with very minimal differences in normalized score and normalized cost. In those cases, a user will look at available options and, based on subjective analysis, will select the

preferred choice. The decision-making application would plot normalized score vs. cost graph that a user analyzes to make his decisions.

In order to handle these scenarios, we can apply the Pareto Optimality concept for solving a problem that a user faces while making choices from multiple options. Before we talk about different Pareto Optimality cases that would apply to our study, let us look at the concept of Pareto Optimality.

4.4. User Decision

According to non-dominance theory, individual A dominates individual B if

- A is no worse than B in all objectives
- A is strictly better than B in at least one objective

A user has a problem choosing the correct management tool for the project among a set of N tools. In order to choose the correct tool (from N number of tools), each solution, or tool, should be compared with every other N-1 solutions to find if it is dominated. In other words, for selecting the correct management tool, the user can follow a dominant strategy to see if one tool is better than the other tool for both objectives:

- Cost Minimization
- Utility/Score Maximization

Tool A will dominate Tool B if choosing Tool A always gives as good as or a better outcome than choosing Tool B. There are 2 possibilities:

- A strictly dominates B: Choosing A always gives a better outcome than choosing B, no matter if we look at the cost-minimization objective or the utility-maximization objective.
- A weakly dominates B: There is at least one set of objectives for which A is superior and another set of objectives that gives A at least the same payoff as B.

Let us analyze some scenarios to understand the user’s tool choice based on the dominance principle. Let us assume that a user is given an option to choose from two tools.

There can be 4 cases that may apply to Tool A over Tool B (Table 4.4.).

Table 4.4. Case Scenarios for Utility and Cost

| Case | Utility/Score | Cost |
|------|---------------|---------|
| I. | MAXIMUM | MINIMUM |
| II. | SAME | MINIMUM |
| III. | MAXIMUM | MAXIMUM |
| IV. | MAXIMUM | SAME |

Case I. Utility is maximum and cost is minimum for Tool A over Tool B.

In Figure 4.1, we have cost on the X axis and total utility/score on the Y axis. The graph is plotted based on user ratings for features.

Tool Utility vs Cost Plot (Upper left corner tools are better)

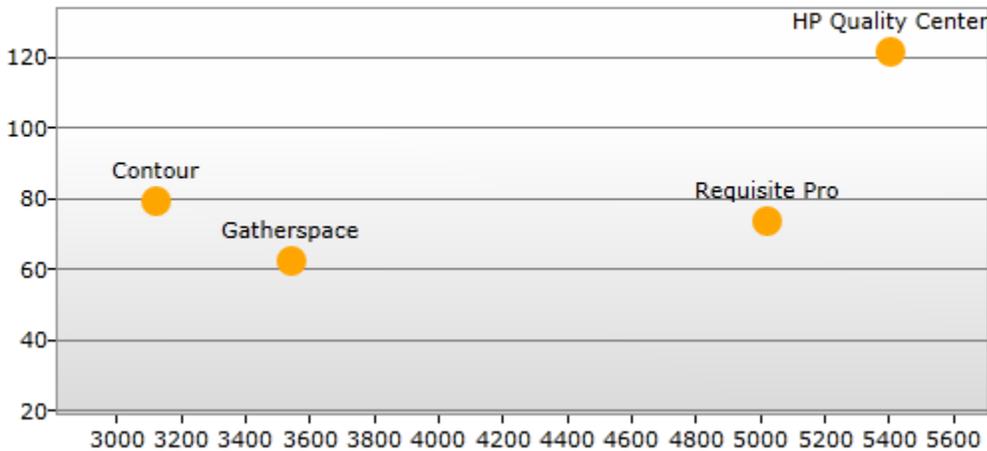


Figure 4.1. Utility vs. Cost Plot for Contour and Gatherspace.

If we compare two tools, Tool A: Contour and Tool B: Gatherspace, we will observe several things: Contour has a utility/score value of 80, which is higher than the utility/score value of Gatherspace which is 60. Contour clearly outperforms Gatherspace in terms of utility/score.

Contour costs \$3120/user/year, and Gatherspace costs \$3540/user/annum. Therefore, Contour is cheaper compared to Gatherspace. Contour, therefore, dominates Gatherspace both in terms of cost minimization and utility maximization.

Result: If Gatherspace and Contour are the options from which to choose, the user will choose Contour.

Case II. Utility/Score is the same in both Tool A and Tool B, and the cost of Tool A is lower than Tool B.

In Figure 4.2, we have cost on the X axis and total utility/score on the Y axis. The graph is plotted based on the user's priority choices for the tool category and tool features.

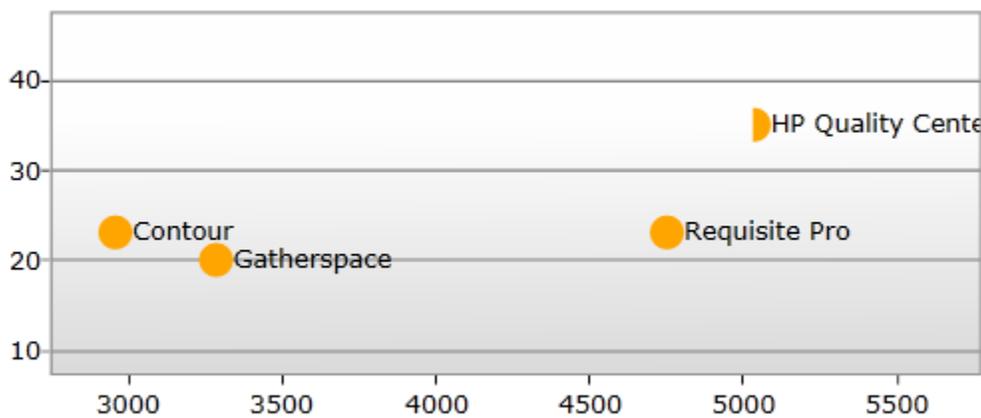


Figure 4.2. Utility vs. Cost Plot for Contour and Requisite Pro.

If we compare two tools, Tool A: Contour and Tool B: Requisite Pro, we will observe the following things:

Contour has a utility/score value of 23, which is the same as the utility/score value of Requisite Pro which is also 23. Contour costs \$3120/user/year however, Requisite Pro costs \$5020/user/annum. Therefore, Contour is cheaper compared to Requisite Pro. Contour, therefore, weakly dominates Requisite Pro in terms of cost minimization.

Result: If Requisite Pro and Contour are the two tools from which to choose, a user will select Contour.

Case III: The utility/score is higher and the cost is higher for Tool A over Tool B.

Tool Utility vs Cost Plot (Upper left corner tools are better)

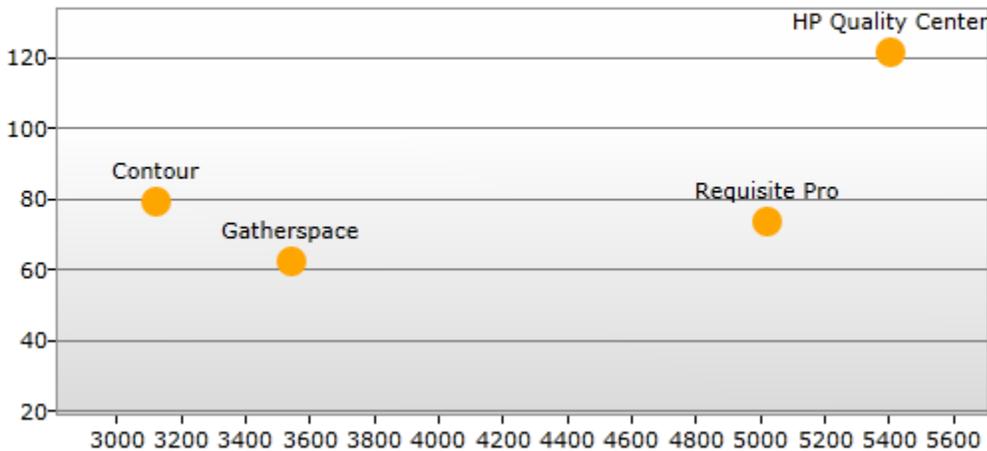


Figure 4.3. Utility vs. Cost Plot for HP Quality Center and Contour.

In Figure 4.3, we have cost on the X axis and total utility on the Y axis. The graph is plotted based on the user’s priority choices for the tool category and tool features.

If we compare two tools, Tool A: HP Quality Center and Tool B: Contour, we will observe the following things. HP Quality Center has a utility/score value of 120, which is higher than the utility/score value of Contour which is 80. HP Quality Center clearly outperforms Contour in terms of utility.

HP Quality Center costs \$5400/user/year, and Contour costs \$3120/user/annum.

Therefore, Contour is cheaper compared to HP Quality Center. Both tools dominate each other on one objective.

Result: The user will have to compare both tools and see whether he is willing to pay more for extra features. If yes, then the user will compare features from Tool A and Tool B to see if the extra features with Tool A provide value to the customer. If yes, then he would choose Tool A. However, if the user feels that the extra features are of no added value, the user would choose Tool B which has fewer features compared to Tool A but is cheaper than Tool A. In order to help the user make that decision, an option will be provided in the decision-making application, allowing users to compare the two tools at the same time to look at uncommon features in both tools.

Case IV: The utility/score is more for Tool A over Tool B, but the cost is the same for both tools.

Result: The user will always choose Tool A over Tool B.

4.5. “Decision-Making” Application and User Analysis

Based on the approach discussed above, a decision-making application has been developed using Microsoft Silverlight technology.

The decision-making application was built in the .Net framework using C# code and Silverlight. Silverlight is a very powerful development tool. It is used to create engaging, interactive user experiences for Web and mobile applications. It is a free plug-in and is powered by the .Net framework [10]. The main reason why Silverlight was chosen was because it is

compatible with multiple browsers, devices, and operating systems. There are many benefits of using Silverlight [11]:

- It delivers an engaging user experience for almost all the browsers.
- It is compatible with most operating systems.
- There is a speedy deliver of high-quality audio, video, animation, and graphics.
- It easily integrates and migrates with existing Asp .Net web applications.
- The Silverlight Toolkit contains controls such as the Chart control to make data visualization easy. The kit contains controls that support dynamic charting.
- Having the .Net runtime in Silverlight helps users code the logic in any language: C#, VB, Python, etc.

The application uses data stored in an XML file. The XML file contains a list of different tools and the features for each tool.

The main reason XML was chosen was because information coded in XML is easy to read and to understand. In addition, it can be processed easily by computers. There is no fixed set of tags. New tags can be created as they are needed. XML documents can contain any possible data type, from multimedia data (image, sound, or video) to active components (Java applets or ActiveX) [12].

The Model View ViewModel (MVVM) was used to develop the application. MVVM is mainly used for modern User Interface (UI) development platforms, e.g., Silverlight, that support event-driven programming.

MVVM facilitates a clear separation to develop the graphical user interface (either as a markup language or GUI code) from the development of the business logic or backend logic known as the model (also known as the data model to distinguish it from the view model). [13]

The view model of MVVM is a value converter, meaning that the view model is responsible for exposing the model's data objects in such a way that those objects are easily managed and consumed. In this respect, the view model is more model than view, and it handles most, if not all, of the view's display logic (although the demarcation between what functions are handled by which layer is a subject of ongoing discussion [5] and exploration). The view model may also implement a mediator pattern that organizes access to the backend logic around the set of use cases supported by the view.

The main reason to use MVVM for development was, therefore, to facilitate the separation of view-layer development from the rest of the pattern by removing virtually all GUI code ("code-behind") from the view layer. Instead of writing GUI code, the framework markup language (XAML) was used, and bindings were created to the view model. This allowed the application layers to be developed in multiple work streams for higher productivity. [13]

Figure 4.4 is from the code in Visual Studio that shows different elements of the MVVM (Model, ViewModel, and Views) in the solution explorer.

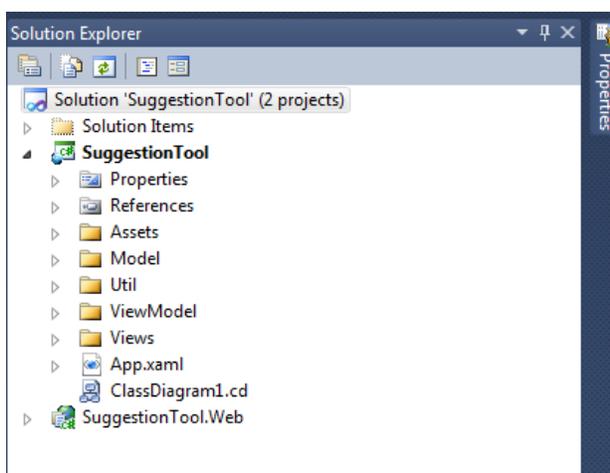


Figure 4.4. Solution Explorer MVVM Components.


```

<?xml version="1.0" encoding="utf-8"?>
<Features xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Headers>
    <Header Name="Requirement Creation">
      <Attributes>
        <Attribute Name="Requirement creation directly in tool" Id="1"/>
        <Attribute Name="Import requirements from MSWord" Id="2"/>
        <Attribute Name="Import requirements from Excel" Id="3"/>
      </Attributes>
    </Header>
    <Header Name="Manage Version and Changes">
      <Attributes>
        <Attribute Name="History of requirement text change" Id="4"/>
        <Attribute Name="Requirement modified by" Id="5"/>
        <Attribute Name="Requirement modified date" Id="8"/>
      </Attributes>
    </Header>
    <Header Name="Requirement Management with Attributes">
      <Attributes>
        <Attribute Name="Custom requirement attributes creation" Id="6"/>
        <Attribute Name="Association of attribute to individual requirement " Id="7"/>
      </Attributes>
    </Header>
    <Header Name="Change Management">
      <Attributes>
        <Attribute Name="Impact analysis on requirement change" Id="9"/>
        <Attribute Name="Online change notification" Id="10"/>
      </Attributes>
    </Header>
  </Headers>
</Features>

```

Figure 4.6. Category and Feature Definition in XML.

Similarly, a user can add more tools to the XML file and define a tool's features in the form of Tool Name and Attribute ID as shown in Figure 4.7.

```

<Tool Name="Gatherspace" Cost="3540">
  <AttribIds>
    <AttribId>1</AttribId>
    <AttribId>3</AttribId>
    <AttribId>4</AttribId>
    <AttribId>5</AttribId>
    <AttribId>7</AttribId>
    <AttribId>8</AttribId>
    <AttribId>12</AttribId>
    <AttribId>13</AttribId>
    <AttribId>14</AttribId>
    <AttribId>15</AttribId>
    <AttribId>16</AttribId>
    <AttribId>19</AttribId>
    <AttribId>22</AttribId>
    <AttribId>23</AttribId>
    <AttribId>24</AttribId>
    <AttribId>26</AttribId>
    <AttribId>28</AttribId>
    <AttribId>29</AttribId>
    <AttribId>30</AttribId>
  </AttribIds>
</Tool>
<Tool Name="HP Quality Center" Cost="5400">
  <AttribIds>
    <AttribId>1</AttribId>
    <AttribId>2</AttribId>
    <AttribId>3</AttribId>
    <AttribId>4</AttribId>
  </AttribIds>
</Tool>

```

Figure 4.7. Tool and Its Feature Addition in an XML File.

The user interface of the application is shown in Figure 4.8.

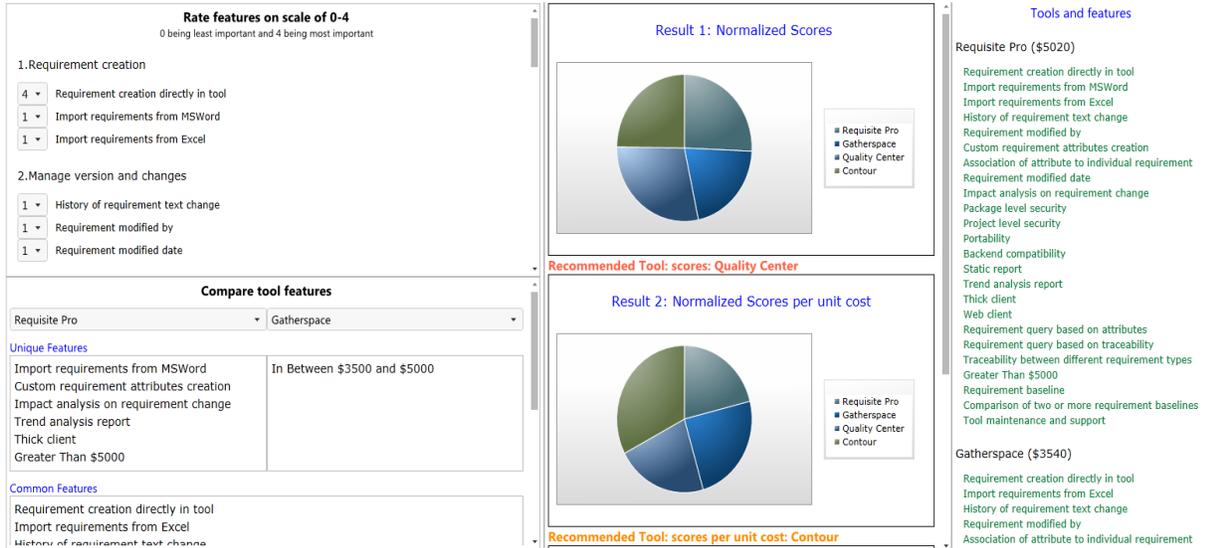


Figure 4.8. User Interface.

With the help of a decision-making application, a user can rate features on scale of 0-4. Based on the input received, the application calculates the utility/score value of each tool in the form of a pie chart. The application also calculates the utility/score per unit cost that helps a user understand and compare different tools. The application also creates the utility vs. cost graph for different tools so that a user can choose the best option.

Let us look closely at each section of the application and discuss them in detail

Figure 4.9 shows a user-rating section which is the top leftmost part of the User Interface. When a user runs the application, this area where he would rate different features on scale of 0-4, with 0 being least important for the project needs and 4 being the most important feature. All features are sorted into different categories, such as requirement creation, manage version and changes, etc., to group related features together.

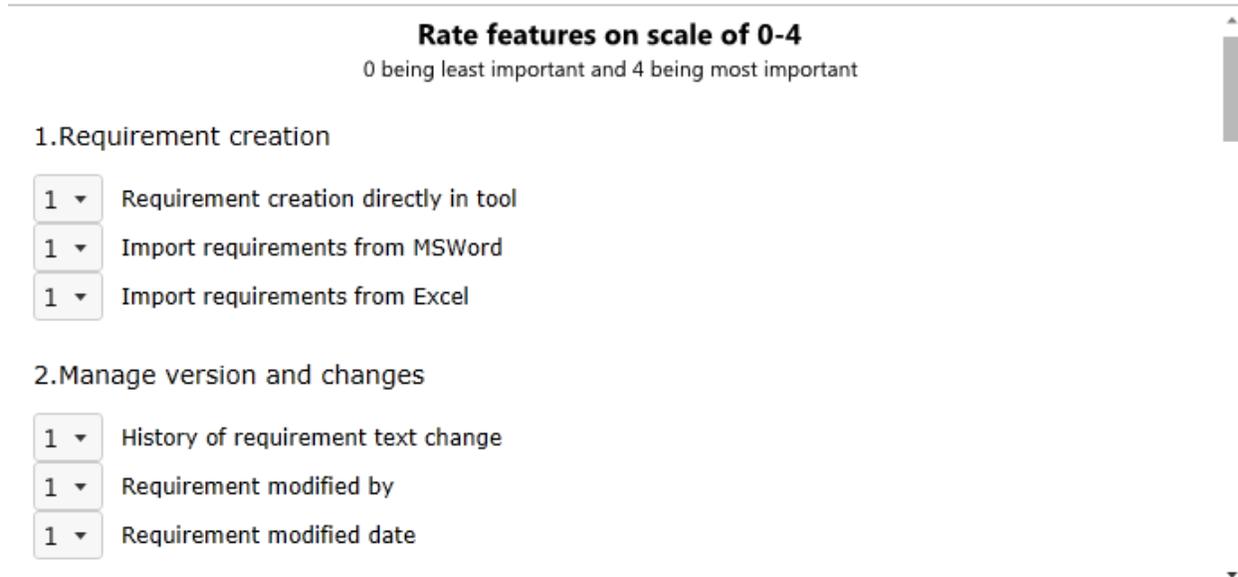
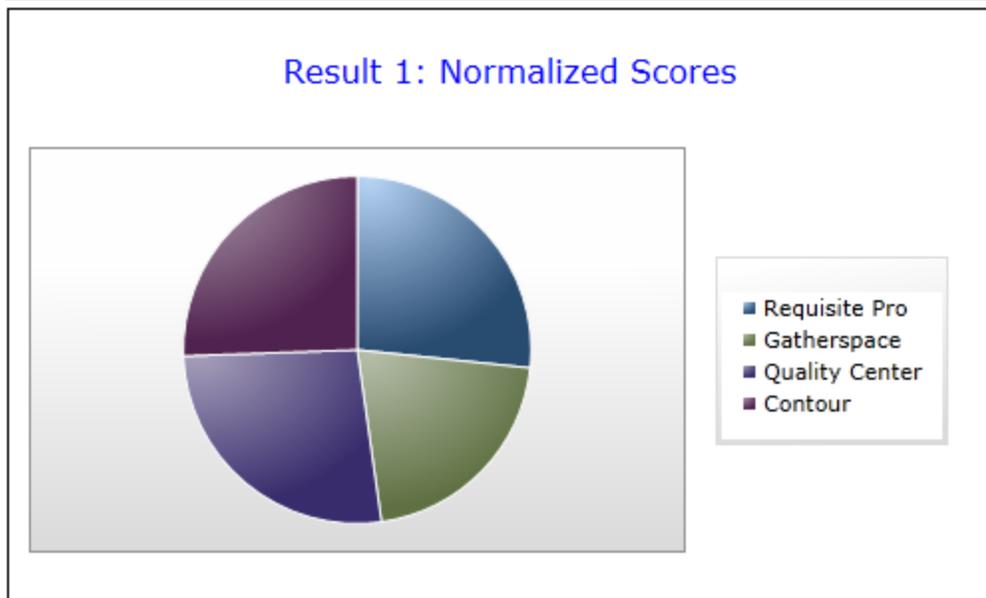


Figure 4.9. User Rating Section.

Based on a user's rating, the scores and score per unit cost are calculated in real time, providing very useful information for a user selecting the tool.

If there is a lot of textual information provided to users, they tend to get confused and find it hard to understand the application. However, if the text is presented as a picture (i.e., a graph), then it is easy for users to make quick decisions. Charts and graphs are colorful and are a pleasant method of communicating information to users.

Figure 4.10 is the graphical representation of the normalized scores based on users' ratings for each feature. The application is developed in such a way that the graph changes dynamically as and when a user changes a feature rating. Users can see the score and normalized % score for each tool by hovering over each section of the pie chart.



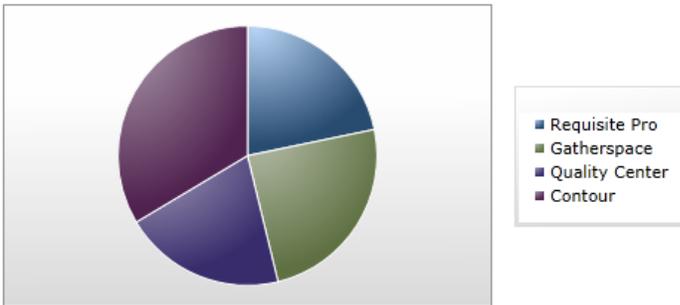
Recommended Tool: scores: Quality Center

Figure 4.10. Pie Chart: Normalized Score.

The main idea is to make as much information available to users as quickly and as clearly as possible so that they do not have to spend lot of time analyzing the data. The graphing and charting functionality in the decision-making application helps users understand the choices between different RM tools.

Figure 4.11 is the graphical representation of the normalized scores per unit cost based on users' rating for each feature. The application is developed in such a way that the graph changes dynamically as and when a user changes a feature rating. The user can see the core per unit cost and normalized % score for each tool by hovering over each section of the pie chart.

Result 2: Normalized Scores per unit cost



Recommended Tool: scores per unit cost: Contour

Figure 4.11. Pie Chart: Normalized Score per Unit Cost.

Figure 4.12 depicts a feature in the application that allows the user to compare any two features at the same time. The user can look at unique features available with one tool or common features for both tools when making a decision about tool selection.

Compare tool features

Requisite Pro Gatherspace

Unique Features

| | |
|--|------------------------------|
| Import requirements from MSWord Custom requirement attributes creation Impact analysis on requirement change Trend analysis report Thick client Greater Than \$5000 | In Between \$3500 and \$5000 |
|--|------------------------------|

Common Features

| |
|---|
| Requirement creation directly in tool Import requirements from Excel History of requirement text change |
|---|

Figure 4.12. Compare Features in Tools.

Figure 4.13 is the graphical representation of scores vs. cost. This graph can be used by a user to make a subjective decision to select the RM tool.

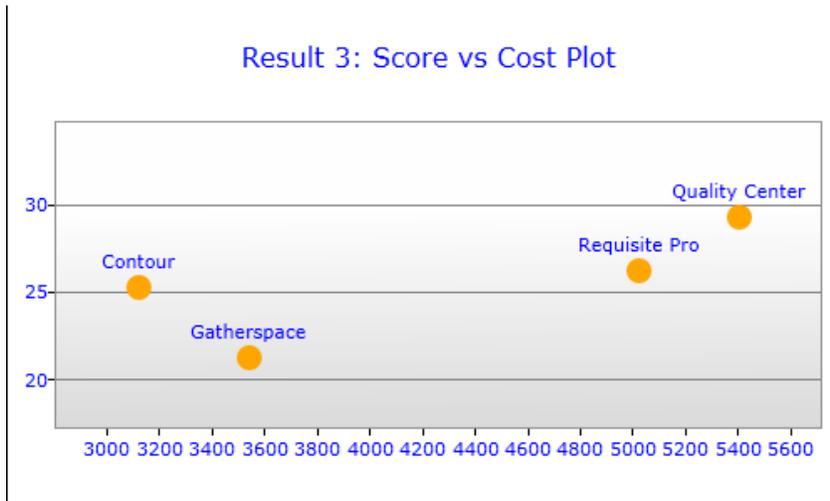


Figure 4.13. Score vs. Cost Plot.

The user interface was designed using Ben Shneiderman's Eight Golden Rules of Interface Design. The users will like this decision-making application because it provides informative feedback.

Whenever a user changes the ratings for different application features, the pie charts update automatically and dynamically, and the application makes suggestions about which tools to select depending on data from both the score pie chart and the score per unit cost pie chart. The dynamic suggestion is quite helpful because users do not have to take additional actions to obtain the results.

Figure 4.14 is a part of the application that displays all features for each tools. This section of application keeps the user informed about the available features for all tools being evaluated.

| Tools and features | |
|-------------------------------|---|
| Requisite Pro (\$5020) | <ul style="list-style-type: none"> Requirement creation directly in tool Import requirements from MSWord Import requirements from Excel History of requirement text change Requirement modified by Custom requirement attributes creation Association of attribute to individual requirement Requirement modified date Impact analysis on requirement change Package level security Project level security Portability Backend compatibility Static report Trend analysis report Thick client Web client Requirement query based on attributes Requirement query based on traceability Traceability between different requirement types Greater Than \$5000 Requirement baseline Comparison of two or more requirement baselines Tool maintenance and support |
| Gatherspace (\$3540) | <ul style="list-style-type: none"> Requirement creation directly in tool Import requirements from Excel History of requirement text change Requirement modified by |

Figure 4.14. Tools and Features.

The user ratings and results appear on one page which really helps users because they do not have to go back and forth to view different pages when rating features and getting results. Users are the initiators. Until and unless a user selects ratings, the results are not displayed. Users can change the rating at any time if they end choose the wrong rating for a feature, thus allowing easy reversal of an action.

CHAPTER 5. RESEARCH RESULTS

The main idea of the paper was to help a user make the correct tool choice for managing requirements. Based on the approach in Chapter 4 and with the help of the decision-making application, the user can select the best choice if he carefully analyzes different graphs produced by the application.

A user has two objectives. The first objective is to minimize cost, and the second objective is to get as many features as possible for project needs at a minimum cost. In other words, we can say that the second objective is to maximize utility. Let us look at the total utility vs. cost graph (Figure 5.1) closely.

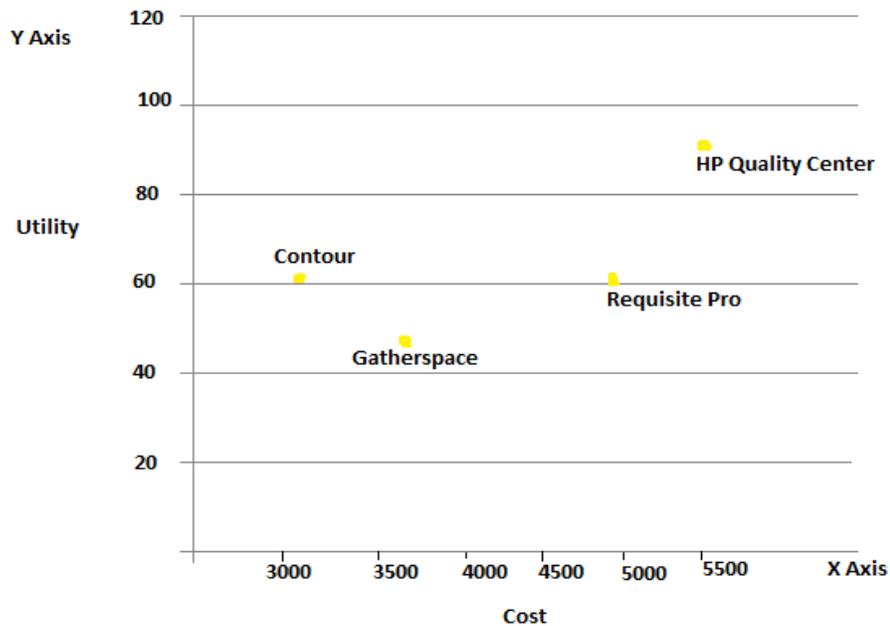


Figure 5.1. Utility vs. Cost Plot for each of the Four Tools Based on a User's Priority Selection for a Tool's Category and Its Features.

Table 5.1. Cost and Utility (Score) for Each of the Four Tools Based on a User’s Priority Selection for a Tool’s Category and Its Features.

| Tool Name | Cost (Minimize) X Axis | Utility (Maximize) Y Axis |
|-------------------|-------------------------------|----------------------------------|
| Contour | 3120 | 62 |
| Gatherspace | 3540 | 47 |
| Requisite Pro | 5020 | 60 |
| HP Quality Center | 5400 | 90 |

In order to select the best tool from the four tools as shown in Figure 5.1, the user will analyze Figure 5.1., and based on information in Table 5.1., user will do a step-by-step analysis.

Step 1. The user will list all tools for comparison.

- Contour
- Gatherspace
- Requisite Pro
- HP Quality Center

Step 2. The user will compare the first tool with the second tool. Does Contour dominate Gatherspace? Yes. (Based on Figure 5.1 and Table 5.1., the user will see the dominance and answer the question.) Therefore, Gatherspace will not move to the non-dominated set and will be deleted.

Step 3. The user will then compare the first tool with the third tool. Does Contour dominate Requisite Pro? Yes. Therefore, Requisite Pro will not move to the non-dominated set and will be deleted.

Step 4. The user will compare the first tool with the last tool. Does Contour dominate HP Quality Center? No. Does HP Quality Center dominate Contour? No. Because both tools do not dominate each other, HP Quality Center will become a part of the non-dominated set.

This is the first non-dominated front highlighted in red in Figure 5.2.

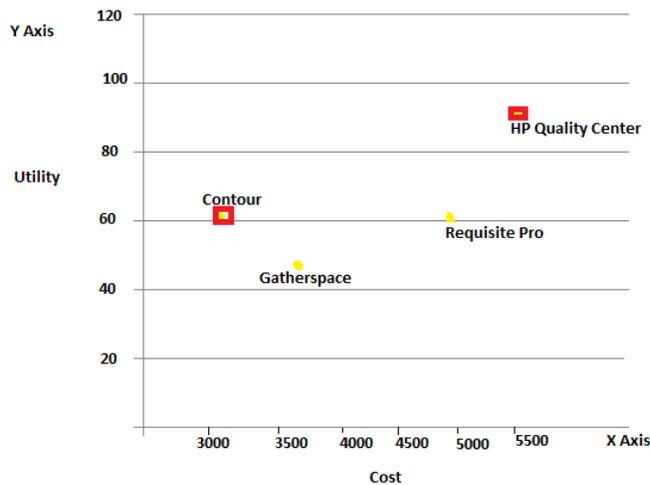


Figure 5.2. First Dominant Front Points Marked in Red.

A user will repeat the same step with the other remaining tools.

Step 6. The user will list the remaining tools for comparison.

- Gatherspace
- Requisite Pro

Step 7. The user will compare the first tool with the second tool. Does Gatherspace dominate Requisite Pro? No. Does Requisite Pro dominate Gatherspace? No. Because both tools do not dominate each other, the user will add Requisite Pro to the non-dominated set. The user will get the new non-dominated front as shown in Figure 5.3. We call it the second non-dominated front.

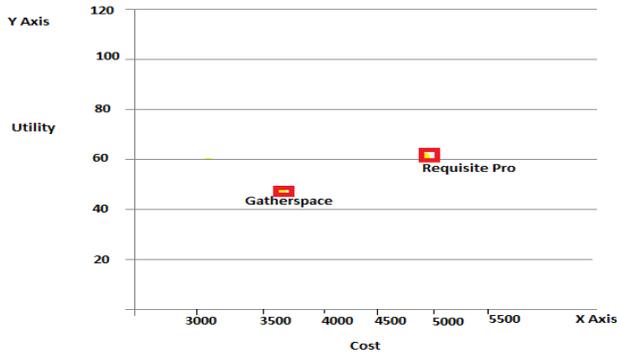


Figure 5.3. Second Dominant Front Points Marked in Red.

The user will get two dominated fronts, Front F1 and Front F2 (as shown in Figure 5.4, and can decide which one is the best tool for his needs.

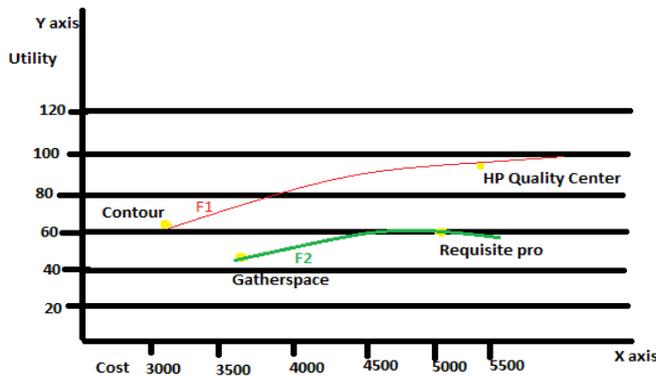


Figure 5.4. Non-Dominant Front.

A rational user will choose tools that are either on the Front F1 or on the upper-left corner of Front F1 because those tools will give more utility value for less cost. However, with our decision-making tool, we have only implemented a score vs. cost graph. Pareto fronts are not created in the application. The decision-making application is a very useful application that not only lets users rate different criteria they need for the RM tool, but also let users make objective and subjective decisions based on dynamic graphs.

CHAPTER 6. FUTURE WORK AND CONCLUSION

6.1. Conclusion

This paper compared and evaluated four requirement-management tools. The tools were evaluated based on their features and costs. The paper described different options for users to make a decision for selecting a requirement-management solution from a number of solutions available on the market. If the users want to select a tool based on the utility/score for each tool, they can refer to the normalized score pie chart in the suggestion tool application. If the users are more interested in choosing a tool based on the utility per unit cost, they can refer to normalized score per unit cost pie chart. If the users want an optimal solution, they can do an analysis based on the total utility vs. cost plot graph as suggested in the paper. The application thus developed as a part of the research met all the objectives listed in the Problem Description (Section 1.1). This application can be utilized to include more tools for evaluation. It is very easy to add more categories and features for tools in the XML file used with this decision-making application.

6.2. Future Work

The work presented in this paper is a very small step in developing a decision-making application. The work can be extended in many ways.

The application's user interface can be improved. The application does not produce non-dominant fronts as discussed in the paper. Future researchers can add the Pareto Optimality algorithm to show the Pareto optimal fronts in Total Utility vs. Cost graph.

Pareto Optimality: This concept is named after Vilfredo Pareto. Pareto Optimality is a measure of efficiency. Pareto Optimality is a domain-independent property that can be applied to the selection of a requirement-management tool based on a user's needs.

Pareto Optimality defines the maximum social welfare and exists where resources are allocated in such a way that it is impossible to make one individual better without making another individual worse. Given a set of alternative allocations for, say, goods or income for a set of individuals, a movement from one allocation to another that can make at least one individual better without making any other individual worse is called a Pareto Improvement. A solution is Pareto-optimal when no further Pareto Improvements can be made. A user can save time and quickly make his decision based on Pareto Fronts.

The application is developed using MVVM architecture, making the application extensible. Future researchers can add more tools and features, and can use this application to compare multiple tools.

REFERENCES

- [1] S. Heinonen, *Requirement Management Tool Support for Software Engineering in Collaboration*, Oulu: University Of Oulu, 2006, pp. 10-11.
- [2] The Centers for Medicare and Medicaid Services (CMS) and the National Center for Health Statistics (NCHS), *ICD-10-CM Official Guidelines for Coding and Reporting*, 2011, pp. 1-2.
- [3] E. D. Clark, *Analysis And Comparison of Various Requirement Management Tools For Use in The Shipbuilding Industry*, Monterey, CA, 2006, pp. 39-51.
- [4] R. R. S. a. J. D. Arthur, *Requirement Management Tools A Qualitative Assessment*, BlacksBurg, VA, 2003, pp. 2-7.
- [5] "Inbound Logistics," Thomas Publishing Company, 2012. [Online]. Available: <http://www.inboundlogistics.com/cms/search-tool/3pl/>. [Accessed 24 October 2012].
- [6] R. S. Corporation, *Rational requisite pro user's guide*, SUMMIT SERVICE COMPANY, 2003, pp. 10,15.
- [7] "Silverlight 4 Chart Example," 13 January 2011. [Online]. Available: <http://www.devcurry.com/2011/01/silverlight-4-chart-example.html>.
- [8] "Silverlight Advantages," Roots Infocomm Ltd., 2003-2011 . [Online]. Available: <http://rootsitservices.com/CustomPages/silverlight.aspx>. [Accessed October 2012].
- [9] "XML Benefits," Software AG, 2012. [Online]. Available: <http://techcommunity.softwareag.com/ecosystem/communities/public/Developer/webmethods/products/tamino/faq/XMLStarter/XMLBenefits.html>. [Accessed 24 October 2012].
- [10] "Model View ViewModel," 17 October 2012. [Online]. Available: http://en.wikipedia.org/wiki/Model_View_ViewModel. [Accessed 23 October 2012].
- [11] h. p. d. Company, *HP Quality center requirements management module*, 2010, p. 2.
- [12] "Gatherspace Requirements on Demand," Gatherspace, 2012. [Online]. Available: <http://www.gatherspace.com/>. [Accessed August 2012].
- [13] "Jama Contour," Contour, 2007-2012. [Online]. Available: <http://www.jamasoftware.com/contour/>. [Accessed August 2012].