Stakeholder Relationship Management for Software Projects

BY

FRANCESCO MARCONI
B.S., Politecnico di Milano, Milan, Italy, 2010
M.S., Politecnico di Milano, Milan, Italy, 2013

THESIS

Submitted as partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois at Chicago, 2013

Chicago, Illinois

Defense Committee:

Prof. Ugo A. Buy, Advisor
Prof. Pier Luca Lanzi, Politecnico di Milano
Prof. Mark Grechanik
To my father Valter
ACKNOWLEDGMENTS

Thanks to all the people that supported me in this experience. Professors Anthony Finkelstein, Ugo Buy and Carlo Ghezzi, for their precious supervision of my thesis work. My family, that strongly encouraged me in these years and made everything possible. My American family, the Maranis, that made me feel like I was home. All my schoolmates and friends that lived this adventure with me. The people of SSR and all the people I met in this unforgettable period of my life.

Thank you all.

FM
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong> INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem Definition</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Thesis objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Structure of the thesis</td>
<td>2</td>
</tr>
<tr>
<td><strong>2</strong> BACKGROUND</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Requirements Engineering</td>
<td>4</td>
</tr>
<tr>
<td>2.1.1 Stakeholder Analysis</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Customer Relationship Management - CRM</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1 CRM Activities</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2 CRM Types</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2.1 Operational CRM</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2.2 Analytical CRM</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2.3 Sales Intelligence CRM</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2.4 Campaign Management</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2.5 Collaborative CRM</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2.6 Geographic CRM</td>
<td>13</td>
</tr>
<tr>
<td>2.2.3 CRM Functions</td>
<td>13</td>
</tr>
<tr>
<td>2.2.4 Customer Segmentation</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Text Mining</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1 Text Mining Approaches</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1.1 Keyword-Based Approach</td>
<td>15</td>
</tr>
<tr>
<td>2.3.1.2 Tagging Approach</td>
<td>15</td>
</tr>
<tr>
<td>2.3.1.3 Information Extraction Approach</td>
<td>15</td>
</tr>
<tr>
<td>2.3.2 Text Mining Tasks</td>
<td>15</td>
</tr>
<tr>
<td>2.3.3 Text Clustering</td>
<td>16</td>
</tr>
<tr>
<td>2.3.3.1 Vector Space Model (VSM)</td>
<td>16</td>
</tr>
<tr>
<td>2.3.3.2 Clustering Algorithms</td>
<td>18</td>
</tr>
<tr>
<td>2.3.4 Carrot²</td>
<td>20</td>
</tr>
<tr>
<td>2.3.4.1 Custom Data Clustering Tools</td>
<td>22</td>
</tr>
<tr>
<td>2.4 Tags</td>
<td>23</td>
</tr>
<tr>
<td>2.5 Return On Investment (ROI)</td>
<td>24</td>
</tr>
<tr>
<td>2.6 Agile Methods</td>
<td>25</td>
</tr>
<tr>
<td>2.6.1 Agile system lifecycle</td>
<td>25</td>
</tr>
<tr>
<td>2.7 Web Application</td>
<td>26</td>
</tr>
<tr>
<td>2.7.1 Web Application Architecture</td>
<td>27</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>EXISTING TOOLS AND METHODOLOGIES</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Requirements Management Tools</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Requirements attributes</td>
<td>30</td>
</tr>
<tr>
<td>3.1</td>
<td>Traceability</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>CRM Tools</td>
<td>32</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Salesforce</td>
<td>32</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Highrise</td>
<td>34</td>
</tr>
<tr>
<td>3.2.3</td>
<td>SugarCRM</td>
<td>34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PROPOSED APPROACH</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Common mistakes in managing requirements: the importance of stakeholders</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>From CRM to SRM for Software Projects</td>
<td>38</td>
</tr>
<tr>
<td>4.2.0.1</td>
<td>From Customer to Stakeholder</td>
<td>40</td>
</tr>
<tr>
<td>4.3</td>
<td>Simplifying Assumptions</td>
<td>40</td>
</tr>
<tr>
<td>4.3.0.2</td>
<td>Stakeholder Identification and Prioritization</td>
<td>40</td>
</tr>
<tr>
<td>4.3.0.3</td>
<td>Requirement Elicitation and Prioritization</td>
<td>40</td>
</tr>
<tr>
<td>4.3.0.4</td>
<td>Stakeholders and Requirements Format</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>Use Cases and Proposed Functionalities</td>
<td>41</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Project Management</td>
<td>41</td>
</tr>
<tr>
<td>4.4.1.1</td>
<td>Use Case: General view of projects</td>
<td>41</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Requirements Management</td>
<td>41</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Stakeholder management</td>
<td>42</td>
</tr>
<tr>
<td>4.4.3.1</td>
<td>Use case: general view of stakeholder information</td>
<td>42</td>
</tr>
<tr>
<td>4.4.3.2</td>
<td>Use Case: Stakeholder’s requirements</td>
<td>43</td>
</tr>
<tr>
<td>4.4.3.3</td>
<td>Use Case: Stakeholder’s history</td>
<td>43</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Tagging System</td>
<td>44</td>
</tr>
<tr>
<td>4.4.4.1</td>
<td>Use Case: general tag view</td>
<td>45</td>
</tr>
<tr>
<td>4.4.5</td>
<td>Stakeholder Group Management</td>
<td>45</td>
</tr>
<tr>
<td>4.4.5.1</td>
<td>Use case: Manually group stakeholders</td>
<td>45</td>
</tr>
<tr>
<td>4.4.5.2</td>
<td>Use case: Automatic Stakeholders segmentation via clustering</td>
<td>45</td>
</tr>
<tr>
<td>4.4.6</td>
<td>Communication Management</td>
<td>47</td>
</tr>
<tr>
<td>4.4.6.1</td>
<td>Filtering incoming communication</td>
<td>47</td>
</tr>
<tr>
<td>4.4.6.2</td>
<td>Manage stakeholder segments for outgoing communication</td>
<td>47</td>
</tr>
<tr>
<td>4.5</td>
<td>Return On Investment (ROI)</td>
<td>49</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Recalling ROI</td>
<td>49</td>
</tr>
<tr>
<td>4.5.1.1</td>
<td>Productivity</td>
<td>49</td>
</tr>
<tr>
<td>4.5.1.2</td>
<td>Total Life-cycle Cost</td>
<td>50</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Cost of change curve</td>
<td>50</td>
</tr>
<tr>
<td>4.5.3</td>
<td>ROI for Agile Methods</td>
<td>52</td>
</tr>
<tr>
<td>4.5.3.1</td>
<td>Costs</td>
<td>52</td>
</tr>
<tr>
<td>4.5.3.2</td>
<td>Benefits</td>
<td>53</td>
</tr>
<tr>
<td>CHAPTER</td>
<td>PAGE</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>4.5.3.3 Putting all together: ROI</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SYSTEM IMPLEMENTATION</td>
<td>55</td>
</tr>
<tr>
<td>5.1 The Web Application Choice</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>5.2 Database Design</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>5.2.1 Entities</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>5.2.1.1 Project</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>5.2.1.2 Requirement</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>5.2.1.3 Stakeholder</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>5.2.1.4 Tag</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>5.2.1.5 History Item</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>5.2.1.6 Stakeholder group</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>5.2.2 Relationships</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>5.2.2.1 Involvement</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>5.2.2.2 Stakeholders Relationships</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>5.2.2.3 History Relationships</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>5.2.2.4 Tagging Relationships</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>5.2.2.5 Stakeholders Grouping Relationships</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>5.3 Module details</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>5.3.1 Project Management</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>5.3.2 Requirements Management</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>5.3.3 Stakeholders Management</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>5.3.3.1 Stakeholder’s overview</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>5.3.3.2 Stakeholder’s requirements</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>5.3.3.3 Stakeholder Communication History</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>5.3.4 Tagging system</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>5.3.5 Communication Management</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>5.3.6 Group Management</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>5.3.6.1 Manual Group Management</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>5.3.6.2 Automatic Group Management</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>EVALUATION</td>
<td>73</td>
</tr>
<tr>
<td>6.1 Tool Evaluation</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>6.1.1 Tool Usability</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>6.1.2 Use Cases</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>6.1.2.1 Project Management</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>6.1.2.2 Requirement Management</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>6.1.2.3 Stakeholder Management</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>6.1.2.4 Tagging System</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>6.1.2.5 Stakeholder Group Management</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>6.1.2.6 Communication Management</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>CHAPTER</td>
<td>PAGE</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>7 CONCLUSION</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>7.1 Future Work</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>7.1.0.7 Short Term</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>7.1.0.8 Long Term</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>7.2 Thesis Summary</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>CITED LITERATURE</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>VITA</td>
<td>81</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>22</td>
</tr>
</tbody>
</table>

CLUSTERING ALGORITHMS PROVIDED BY CARROT$^{2[1]}$.  

---

viii
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stakeholder identification model described by Sharp and Finkelstein.</td>
</tr>
<tr>
<td>2</td>
<td>The Vector Space Model.</td>
</tr>
<tr>
<td>3</td>
<td>Hierarchical clustering dendogram example.</td>
</tr>
<tr>
<td>4</td>
<td>Partitional clustering.</td>
</tr>
<tr>
<td>5</td>
<td>Carrot2 Document Clustering Server quick start screen.</td>
</tr>
<tr>
<td>6</td>
<td>An example of tag cloud with terms related to Web 2.0</td>
</tr>
<tr>
<td>7</td>
<td>The three tiers of web application architecture.</td>
</tr>
<tr>
<td>8</td>
<td>Example of the Salesforce home screen</td>
</tr>
<tr>
<td>9</td>
<td>Highrise welcome page</td>
</tr>
<tr>
<td>10</td>
<td>Example of the SugarCRM home page</td>
</tr>
<tr>
<td>11</td>
<td>Use Case: General view of projects</td>
</tr>
<tr>
<td>12</td>
<td>Use Case: General view of requirements</td>
</tr>
<tr>
<td>13</td>
<td>Use Case: General view of stakeholder information.</td>
</tr>
<tr>
<td>14</td>
<td>Use Case: Stakeholder’s requirement display</td>
</tr>
<tr>
<td>15</td>
<td>Use Case: Stakeholder communication history</td>
</tr>
<tr>
<td>16</td>
<td>Use Case: general tag view</td>
</tr>
<tr>
<td>17</td>
<td>Use Case: Stakeholders’ group management (manual and automatic)</td>
</tr>
<tr>
<td>18</td>
<td>Use Case: Incoming communication filtering</td>
</tr>
<tr>
<td>19</td>
<td>Use Case: Outgoing communication management</td>
</tr>
<tr>
<td>20</td>
<td>The cost of change curve described in</td>
</tr>
<tr>
<td>21</td>
<td>Entity-Relationship diagram of the system database</td>
</tr>
<tr>
<td>22</td>
<td>Web application sections</td>
</tr>
<tr>
<td>23</td>
<td>Project list</td>
</tr>
<tr>
<td>24</td>
<td>Project overview</td>
</tr>
<tr>
<td>25</td>
<td>Project requirement list</td>
</tr>
<tr>
<td>26</td>
<td>Stakeholder overview</td>
</tr>
<tr>
<td>27</td>
<td>Stakeholder’s requirement list</td>
</tr>
<tr>
<td>28</td>
<td>Stakeholder’s history</td>
</tr>
<tr>
<td>29</td>
<td>Edit contact form</td>
</tr>
<tr>
<td>30</td>
<td>The Tag-Cloud section</td>
</tr>
<tr>
<td>31</td>
<td>The history filtering function, in the communication section</td>
</tr>
<tr>
<td>32</td>
<td>Manual group management functions</td>
</tr>
<tr>
<td>33</td>
<td>Group suggestion with cluster analysis</td>
</tr>
</tbody>
</table>
LIST OF ABBREVIATIONS

CRM Customer Relationship Management
CSS Cascading Style Sheet
DCS Document Clustering Server
GST Generalized Suffix Tree
HTML HyperText Markup Language
LOC Lines Of Code
PHP PHP Hypertext Processor
ROI Return On Investment
SFA Sales Force Automation
SQL Structured Query Language
SRM Stakeholder Relationship Management
STC Suffix Tree Clustering
TLC Total Life-cycle Cost
UIC University of Illinois at Chicago
VSM Vector Space Model
XML Extensible Markup Language
SUMMARY

Stakeholder Analysis represents a crucial task in the Requirement Elicitation phase for a software project. Understanding the needs of stakeholders is determining for the creation of an adequate software product. Nevertheless, Stakeholder Analysis can not be considered a mature task: it is often overlooked in the Requirement Elicitation process and there is a poor literature available about it. Various techniques have been adopted, but a standard method could not be identified. Some of the main problems related to these methods are the substantial lack of involvement and collaboration of the stakeholders, the short lifetime of relationships with them, the little quantity of information about stakeholders preferences and their relationships with other stakeholders.

This work aims to focus on the stakeholders in order to gather useful information about project requirements, trying to involve them and to extend the duration of the relationships with them for a continuous support to project decisions. The proposed approach to achieve this goal takes inspiration from the popular business strategy of Customer Relationship Management (CRM). A web application has been designed to provide a support to manage relationship with stakeholders. It allows to keep track of all the interactions with them, to keep them updated with selective information about project requirements, and to automatically extract useful information from their profiles by means of text mining techniques. Moreover the work deals with the problem of defining a Return on Investment (ROI) for this kind of approach, which is a common aim for CRM in business environment while it is not so common for software projects.

Use cases have been prepared to define the needed functionalities of the system.
CHAPTER 1

INTRODUCTION

According to the Standish Groups Chaos 2006, about 20 percent of the software projects investigated failed, 46 percent of projects exceeded time or budget constraints. It is reported the importance, for the project success or failure, of how the system requirements are handled: in particular the communication of the requirements. Most of the errors in system development projects originate during the phase of requirements engineering\(^4\). These errors are often discovered in later project phases or even after the system deployment. The system is implemented according to what the requirements say; incomplete, not clearly formulated or wrong requirements result in a system that does not have the properties the customer needs. The cost of fixing a defect is higher if done in a later project phase instead of an earlier one, like the software requirements engineering phase. Stakeholders are among the main sources of requirements, good relationships with them is crucial. The requirements engineering is a systematic and disciplined approach to the specification and management of the requirements with the following goals\(^5\):

1. Knowing the relevant requirements, achieving a consensus among the stakeholders about these requirements, documenting them according to given standards, and managing them systematically
2. Understanding and documenting the stakeholders desires and needs
3. Specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders desires and needs

All three goals address important aspects of requirements engineering: process orientation, stakeholder focus and importance of risk and value consideration. The stakeholder is a person or organization that has a (direct or indirect) influence on a systems requirements\(^6\). Indirect influence also includes situations where a person or organization is impacted by the system. Interaction with stakeholders is essential in the system design, and strongly relies on the communication strategy of the requirements engineers. Very often the failures of software projects are due to the lack of communication or involvement of stakeholders.
1.1 Problem Definition

The main problems related to the stakeholder analysis in software projects can be identified as: lack of communication between stakeholders and analysts, lack of involvement, short lifetime of the relationships with stakeholders and poor information about preferences and relationships among them.

1.2 Thesis objectives

The work proposes a stakeholder-centric approach for software projects management, by means of practices and techniques typically adopted in Customer Relationship Management. The objective is to better understand stakeholder needs, focusing the attention on relationships with them, on communication with them and on a structured way of managing all the information gathered from them. The more stakeholder needs are met by requirements, the more project business value will be positively affected. Given these intents, a web tool has been designed to provide an effective implementation of these techniques. For simplicity and time constraints, some of the functionalities have been developed in a basic version and some minimalistic assumptions have been adopted.

Finally it has been investigated the possibility of estimating the return on investment (ROI) for the above-mentioned approach, by means of methods such as Total Life-cycle Cost.

1.3 Structure of the thesis

Chapter 2 gives an overview of the main background topics of the thesis. It starts describing Requirement Engineering, with particular accent on Stakeholder Analysis techniques, then it presents Customer Relationship Management and in the end gives an overview of Text Mining, Return on Investment, Agile Methods and Tagging system. Various Definitions are provided in order to let the reader be familiar with the technical terms that will be used.

Chapter 3 shows the most relevant tools adopted to perform Requirement Engineering and CRM tasks. The objective is to highlight the main features that inspired the design of the tool proposed in this work.

Chapter 4 describes the contribution of this thesis work, presenting the concept of Stakeholder Relationship Management, the features that have been selected referring to a set of use cases, and the conceptual design of the tool performing SRM.

Chapter 5 goes through the actual design of the Web Application, describing how the system has been implemented employing PHP, jQuery and MySQL technologies.
Chapter 6 provides some critical considerations to evaluate the proposed approach, considering how the starting problems have been faced, how the identified use cases have been successfully implemented.

Chapter 7 summarises the thesis contribution, suggesting possible improvements and extensions for the tool.
CHAPTER 2

BACKGROUND

In this chapter will be presented an overview of the main background concepts of the thesis. Definitions will be provided to better understand the context and the approach later described.

Section 2.1 presents the Software Engineering field of Requirements Engineering, providing an overview of the general practices with a particular focus on those related to the stakeholders.

Section 2.2 introduces the subject of Customer Relationship Management.

Section 2.3 illustrates the topic of Text Mining, a specific application of Data Mining for textual data.

Section 2.5 provides a brief definition of the concept of Return on Investment.

Section 2.6 describe the general details about Agile methodologies, focusing on the stakeholder-centricity of those methods.

Section 2.4 deals with the topic of tags or metadata, describing their usage in modern Web 2.0 systems.

Section 2.7 outlines the characteristics of web applications and their architectures.

2.1 Requirements Engineering

According to Sommerville[7], Requirements Engineering is the process of discovering, documenting and managing the requirements for a computer-based system. About the difficulty of Requirements Engineering, Brooks[8] writes:

“the hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later”.

It is an iterative extraction and refinement process between the client and the designer, often the client does not know what he wants, he has almost never thought of the problem in the detail necessary for specification. It is very difficult for a client to specify completely, precisely, and correctly the exact requirements of a modern software system. Requirements engineering is also defined[9] as:
“[...] the branch of software engineering concerned with the real-world goals for functions of
and constraints on software systems. It is also concerned with the relationship of these factors
to precise specifications of software behaviour, and to their evolution over time and across
software families”.

Another definition (by Nuseibeh[10]) says that a software system is developed starting from
real-world goals, and from these goals we know what system to build and why we build it.
The definition refers to precise specifications that provide the basis for analysing requirements,
validating that they are indeed what stakeholders want, defining what designers have to build,
and verifying that they have done so correctly upon delivery. These specifications evolve over
time and across software families.

Unfortunately it seems there is no common definition of requirements engineering and exist-
ing definitions differ in their focus. For example, some definitions concentrate on the elicitation
of requirements and thereby focus on the interaction with the user, whereas the focus of others
is more on the documentation (specification) of the requirements.

The term requirement is defined[5] as:

1. A need perceived by a stakeholder.
2. A capability or property that a system shall have.
3. A documented representation of a need, capability or property.

During the requirements engineering process four tasks are performed namely the elicitation,
documentation, validation / negotiation, and management.

Requirements are first of all elicited, and in a second step documented. Next the doc-
umented requirements are negotiated and validated according to some quality criteria. The
various stakeholders negotiating about the requirement, agree on it or change it accordingly;
the requirements must correspond to the original user/customer needs. Requirements manage-
ment comprises any measures necessary to prepare them, to maintain their consistency and
ensure their implementation.

The main product of the requirements engineering process is the requirements specifica-
tion which should state what a system should do and not how it should do it. The goal of
requirements is to understand and specify the problem to be solved rather than the solution.
There is the problem space and the solution space, the analysis and specification of require-
ments belongs in the problem space. Figuring out what the right problem is can be difficult,
the relevant knowledge about the system is not available from a particular user or customer, is distributed among many stakeholders which might have conflicting viewpoints and interests. Therefore, the identification of the relevant sources and appropriate consideration of them is essential. We need not only to know what a system should do, but why such a system needs to be implemented and who should be involved in the process. The system or the problem world is structured along three dimensions\textsuperscript{[11]}: why, what and who. The objectives that justify the new system which is going to be implemented, are located in the why dimension. The objectives constraints, their ramifications and interactions in the domain in which the system originates must be known in depth. This is the requirements elicitation and domain understanding. The objectives of this phase are\textsuperscript{[11]}:

- Understand the system as-is and its context
- Identify the problems and opportunities calling for a new system.
- Discover the real needs of stakeholders with respect to the new system.
- Explore alternative ways in which the new system could address those needs.

The knowledge of the structure of the organization that needs the new system with its business objectives, policies and responsibilities must be acquired. In this phase stakeholders or representatives of them are identified, and some criteria may be used in the choice\textsuperscript{[11]}, like:

- Relevant position in the organization.
- Effective role in making decisions about the system-to-be.
- Level of domain expertise.
- Exposure to perceived problems.
- Influence in system acceptance.
- Personal objectives and conflict of interest.

As already said there might be conflicts among stakeholders, because of diverging views, competition and different priorities. Sometimes it is not possible to get regular, direct access to key stakeholders or their representatives. We need to work with our stakeholders to identify what they want, produce something based on what we have understood, get feedback from our stakeholders, and then update our product. This is possible if we work in an evolutionary and collaborative manner, which means that we must work closely and regularly with stakeholders.
Traditional system development is a serial process where requirements are defined in great detail early in the life-cycle, these may or may not undergo a review by the stakeholders before they are passed to developers. This traditional method doesn’t take into account that requirements may change in time, because the organization may change or because the understanding of requirements by stakeholders may change. To get the right stakeholders involvement, the support of senior management is fundamental to decide the stakeholders level of involvement. Once we have identified the stakeholders and got the right access to them, the interaction with stakeholder is not without problems. At this point communication skills are important, we need to use the right terminology depending on the stakeholder’s background. The ability to listen to stakeholders is fundamental, as well as to create solid interpersonal relationships.

### 2.1.1 Stakeholder Analysis

A stakeholder in an organization is (by definition) “any group or individual who can affect or is affected by the achievement of the organization’s objectives”\(^{[12]}\). In software engineering stakeholders are groups or individuals that will be affected by the system and that will have an influence on the system requirements. Stakeholders include end-users, managers, software engineers responsible for system development and maintenance, customers that will pay for the system. Stakeholder analysis is an important technique for analysing their needs and for their identification. The aim of stakeholder analysis process is to develop a view of the relationships between the different stakeholders and the issues they care about most. In a project it is important to identify:

- the interests of all stakeholders;
- key people for information distribution;
- stakeholders that should be encouraged to participate in different stages of the project;
- ways to reduce potential negative impact of some stakeholders.

Engaging stakeholders throughout the project life-cycle, managing their expectations, the priority to be given to each stakeholders view are keys to projects success. According to Sharp and Finkelstein\(^{[2]}\), in software engineering is more appropriate to think of stakeholders as roles rather than specific people. They\(^{[2]}\) have proposed an approach to discovering all relevant stakeholders of a system. The approach starts with a set of baseline stakeholders roles from users, developers, legislators, and decision-makers; among these stakeholders the method
identifies supplier stakeholders that provide information or supporting tasks to the baseline and client stakeholders that process or inspects the products of the baseline. Other stakeholders called satellite interact or support the baseline in different ways. Figure 1 illustrates the main elements of this process of stakeholder identification. This step is repeated for each newly identified role and the result is added to the list of stakeholder roles. In this approach the interactions between stakeholders is more important than interactions between the system and the stakeholder. The result is a network of stakeholders with each stakeholder being a node and each edge being the relationship between the two stakeholders.

StakeNet\textsuperscript{[13]} is a tool based on Social Networks that identify and prioritise stakeholders. It was applied to a large scale software project at University College London (UCL). With StakeNet each stakeholder participates in the stakeholder analysis, their relations are used to identify and prioritise them. Stakeholders are asked to recommend other stakeholders, in this way StakeNet builds a Social Network with stakeholders as nodes and their recommendations as links; Social Network measures are used to prioritise stakeholders. StakeNet uses concepts like scope that is the work required for completing the project successfully, stakeholder role which is the part it plays in the project, salience for the level of influence the stakeholder has on the project. StakeNet works in six steps:
1. Determine the project scope is preparatory step to describe the boundary of the project in order to know which stakeholders should be involved.

2. Identify roles based on the first step identifies an initial set of stakeholder roles from the predefined categories of users, developers, legislators and decision-makers.

3. Find stakeholders for each role, that is the person taking up the role or a group representative.

4. Get recommendations and stake for each stakeholder, when we know the stake we know how the stakeholder influences or is influenced by the project. A recommendation is a triple $<\text{stakeholder}, \text{stakeholderrole}, \text{salience}>$ where salience is a number from 1 to 5: 1 means that the stakeholder has a marginal impact on the project while 5 means an important influence on the project.

5. Build the social network with stakeholders as nodes and their recommendations as directed edges; the edges have weights depending on salience.

6. Prioritise stakeholders by applying social network measures.

Another recent method of stakeholder analysis\cite{14} defines a process for ranking stakeholders based on needs and the relative importance of stakeholders to others in the network. This analysis proceeds according to two principles:

1. Establish and prioritize the needs of a given stakeholder based on the importance to them.

2. Establish and prioritize the stakeholders based on their importance to the organization.

According to these two principles the delivery of value is an exchange between two actors: the project and the beneficial stakeholder. Benefits are provided, with a cost, from one actor to the other and vice versa. Exchange takes place when the outputs of the project meet the needs of the beneficial stakeholder, and the outputs of the beneficial stakeholder meet the needs of the project. This exchange creates a loop called value loop. If the exchange is not successful there is no value delivery.

### 2.2 Customer Relationship Management - CRM

There are several definitions for the concept of Customer Relationship Management. One of the best known, coming from a CRM guru\cite{15}, states:
“CRM is a philosophy and a business strategy supported by a system and a technology designed to improve human interaction in a business environment.”

(Paul Greenberg)

Another definition\[16\] seems to be more pragmatic, and mentions the technological aspects of CRM strategies:

“At the core, CRM is an integration of technologies and business processes used to satisfy the needs of a customer during any given interaction. More specifically, CRM involves acquisition, analysis and use of knowledge about customers in order to sell more goods or services and to do it more efficiently.”

(Ranjit Bose)

Customer Relationship Management, or CRM, is a structured, integrated process to manage relationships and knowledge of the customers, in order to build long or medium term customized connections, increasing customer satisfaction level and company business value. Provided this customer-centric approach, the customer becomes one of the most important elements for company business strategy.

Even though the goals which companies try to reach with the adoption of CRM are various, the three main objectives that are always present and valid are:

- retaining existing customers;
- acquiring new customers;
- increasing company business value by means of sales increase;

2.2.1 CRM Activities

The main CRM activities that can be identified are the following:

- **Front office operations**: direct interaction with customers (e.g. face to face meetings, phone calls, e-mail, online services etc.).

- **Back office operations**: operations that ultimately affect the activities of the front office (e.g., billing, maintenance, planning, marketing, advertising, finance, manufacturing, etc.).
• **Business relationships:** interaction with other companies and partners, such as suppliers/vendors and retail outlets/distributors, industry networks (lobbying groups, trade associations). This external network supports front and back office activities.

• **Analysis:** key CRM data can be analysed in order to plan target-marketing campaigns, conceive business strategies, and judge the success of CRM activities (e.g., market share, number and types of customers, revenue, profitability, etc.).

### 2.2.2 CRM Types

Provided the basic activities, there are different kinds of approach to CRM, depending on the needs of the company.

#### 2.2.2.1 Operational CRM

Operational CRM provides support to “front office” business processes, such as sales, marketing and service staff.

Interactions with customers are generally stored in customers’ contact histories, and staff can retrieve customer information as necessary. The contact history provides staff members with immediate access to important information on the customer (products owned, prior support calls etc.), eliminating the need to individually obtain this information directly from the customer. Operational CRM processes customer data for a variety of purposes:

- Managing Campaigns
- Enterprise Marketing Automation
- Sales Force Automation

Particularly relevant is **Sales Force Automation** (SFA), which automates sales force-related activities such as scheduling sales calls or mailings, tracking responses and generating reports.

#### 2.2.2.2 Analytical CRM

Analytical CRM consists in procedures and tools which allow to improve company’s knowledge about the customer by means of extraction, processing and analysis of data gathered from Operational CRM. Customers’ data are analysed for a variety of purposes:

- Designing and executing targeted marketing campaigns
- Designing and executing customer acquisition, cross-selling, up-selling
• Analysing customer behaviour in order to make decisions relating to products and services (e.g. pricing, product development)
• Management decisions (e.g. financial forecasting and customer profitability analysis)
• Analytical CRM generally makes heavy use of data mining.

2.2.2.3 Sales Intelligence CRM

Similar to Analytical CRM, it is intended as a more direct sales tool. Its main features include alerts sent to sales staff regarding:

• Cross-selling/Up-selling/Switch-selling opportunities
• Customer drift
• Sales performance
• Customer trends
• Customer margins

2.2.2.4 Campaign Management

Campaign management combines elements of Operational and Analytical CRM. Its main functions include:

• Target groups formed from the client base according to selected criteria
• Sending campaign-related material (e.g. on special offers) to selected recipients using various channels (e.g. e-mail, telephone, post)
• Tracking, storing, and analysing campaign statistics, including tracking responses and analyzing trends

2.2.2.5 Collaborative CRM

Collaborative CRM covers aspects of a company’s dealings with customers that are handled by various departments within a company, such as sales, technical support and marketing. Staff members from different departments can share information collected when interacting with customers. For example, feedback received by customer support agents can provide other staff members with information on the services and features requested by customers. Collaborative CRM’s ultimate goal is to use information collected by all departments to improve the quality of services provided by the company.
2.2.2.6 Geographic CRM

Geographic CRM (GCRM) combines geographic information system and traditional CRM. Geographic data can be analysed to provide a snapshot of potential customers in a region or to plan routes for customer visits.

2.2.3 CRM Functions

Another perspective over CRM and its application is given by this categorization\cite{17} of the main modules present in most of the CRM tools, that are:

1. Marketing
2. Sales
3. Service

Marketing related functions include:

- Build up Marketing Campaigns
- Event marketing Management (reminders, updates by email)
- Lead Capture to segment leads and customers
- Trackable links in template emails to monitor click-through and conversation rates
- Track Marketing investments
- Track responses and success rates from Marketing Campaigns
- Know effectiveness of Marketing elements
- Follow different phases of marketing pieces
- Measure marketing ROI

Sales Module functions are:

- Manage leads
- Keep track of the communication with prospects, referrals and customers
- Appointment Management and Time Management
- Calendar Management
- Reporting
Customer Service functions are needed after marketing and selling operations to provide support to the customer. The main customer service functions are:

- Product Database
- Knowledge Base Search
- Customer Tracking
- Contract Tracking
- Customer History

2.2.4 Customer Segmentation

One of the features that is usually implemented by CRM systems is the segmentation of customers, made applying the marketing strategy called Market Segmentation. Market segmentation is defined as “a marketing strategy that involves dividing a broad target market into subsets of consumers who have common needs”\(^\text{[18]}\). Marketing Campaign can then be designed to target those specific segments addressing needs or desires that are believed to be common in those segment, using appropriate communication media.\(^\text{[18]}\)

There are various approaches to Market Segmentation; most of them, however, rely on the data mining techniques of Cluster Analysis.

2.3 Text Mining

Text Mining is an interdisciplinary field bringing together techniques from data mining, linguistics, machine learning, information retrieval, pattern recognition, statistics, databases and visualization, to the issue of quickly extracting information from large databases\(^\text{[19]}\). Differently from traditional data mining, the data sources are document collections, and interesting patterns are found not among formalized database records but in the unstructured textual data in the documents in these collection\(^\text{[20]}\).

2.3.1 Text Mining Approaches

There are several kind of approaches to Text Mining, depending on the data they take as input, and on the objective they pursue as mining tasks.

The three main approaches that can be identified are the keyword-based approach, the tagging approach and the information extraction approach\(^\text{[21]}\).
2.3.1.1 **Keyword-Based Approach**

The keyword-based approach takes as input a set of keywords or terms in the document. It relies on Information Retrieval techniques. The knowledge it can provide is quite limited: it can discover relationship at a relatively shallow level, such as rediscovering of compound nouns (e.g., “database” and “system”), or co-occurring patterns with less significance (e.g., “terrorist” and “explosion”). For these reasons, it is often not able to provide much deep understanding of the text.\[^{21}\]

2.3.1.2 **Tagging Approach**

For the tagging approach the input consists in a set of tags that can be provided by *manual tagging* (which is costly and infeasible for large collection of documents\[^{21}\]), or by *automated categorization algorithm*, (which can process a relatively small set of tags and requires to define the categories beforehand).

2.3.1.3 **Information Extraction Approach**

In the case of information extraction approach the input is represented by semantic information, such as events, facts, or entities uncovered by *information extraction*. It is the more advanced approach among these, and it may lead to the discovery of some deep knowledge\[^{21}\], but it requires semantic analysis of text by means of *natural language understanding* and *machine learning* methods.

2.3.2 **Text Mining Tasks**

There are various text mining tasks that can be performed using the previously described approaches. The most important are: *Document Clustering*, *Document Classification*, *Information Extraction*. For what will be discussed in this thesis, the only task which will be described in detail is Document Clustering.

2.3.3 **Text Clustering**

Text clustering (or Document clustering) is an automatic technique to find disjoint subsets or clusters of the texts such that:

- Texts within a cluster are very similar
- Texts in different clusters are very different

Text clustering methods can be used as a user interface for browsing a large collection of documents. The information need may be formulated as a metaphor of the way the information
is searched in a book\textsuperscript{[22]}. If the user has a specific question in mind he consults the index which then links to some pages; this corresponds to a text search method. If the user wants to have an idea of the contents of the book then he may go through the table of contents which describes the logical structure of the book; this corresponds to a cluster-based browsing of documents. There are numerous clustering algorithms for text applications, however the most frequently used algorithms are based on vector space.

Text clustering is applied mainly in two ways: the first as a tool to improve retrieval performance and the second as a way to organize large text collections. Text clustering for retrieval purposes originates from the cluster hypothesis\textsuperscript{[23]}, which states that closely associated texts tend to be relevant to the same requests, this means that relevant texts will be separated from irrelevant ones. An example of text clustering, as a tool to organize a large collection of texts, is the grouping of the results from a web search engine. Text clustering applies the same analysis process of data clustering to textual information. To group items or texts it is necessary to identify some item features or characteristics, and to measure the similarity between items.

2.3.3.1 Vector Space Model (VSM)

A text in which a term of a query occurs more often has more to do with the query, therefore the text should be assigned a high score. Here the query is in free form, that is just a sequence of words or terms without any connecting search operators. A term is assigned a weight or term frequency depending on the number of occurrences of the term in the text. There is a dictionary of terms, and for each term there is a list that records which text the term occurs in. The text is viewed as an unordered collection of words. This simplified representation\textsuperscript{[24]} is called bag-of-words model, the terms frequency is used as a feature of the text. Two texts represented by similar bag-of-words are then similar in content. Each text is viewed as a vector with one component for each term, the components are computed using some weighting scheme. The score of a text\textsuperscript{[24]} is the sum, over all query terms, of the number of times each of the query terms occur in the text, or instead of occurrences, the sum of weights computed with any weighting function. The representation of a set of texts as vectors in a common vector space is known as the vector space model, which is used in information retrieval operations, text classification and text clustering\textsuperscript{[24]}. A visual representation is provided in Figure 2.

In this model a query is viewed as a vector in the same vector space as the collection of texts. Following\textsuperscript{[24]}, the vector derived from the text \( t \) is denoted by \( V(t) \), with one component in the vector for each dictionary term. The set of a collection of texts then may be viewed as a
set of vectors in a vector space, in which there is one axis for each term. The similarity between two texts is computed with the cosine similarity of their vector representations $V(t_1)$ and $V(t_2)$

$$sim(t_1, t_2) = \frac{(V(t_1) \cdot V(t_2))}{\|V(t_1)\|\|V(t_2)\|}$$ (2.1)

where the numerator represents the dot product of the vectors $V(t_1)$ and $V(t_2)$ while the denominator is the product of their Euclidean lengths.

Figure 2: The Vector Space Model.
### 2.3.3.2 Clustering Algorithms

Clustering algorithms group a set of texts or documents into subsets or clusters. Texts within a cluster are as similar as possible, while they are as dissimilar as possible from those in another cluster. The cluster hypothesis\textsuperscript{[24]} states that texts in the same cluster behave similarly with respect to relevance to information needs. The hypothesis states that if there is a text from a cluster that is relevant to a search request, then it is likely that other texts from the same cluster are also relevant. This is because clustering puts together texts that share many terms. Clustering algorithms based on vector space can be divided into two categories: hierarchical and partitioning, that is according to the type of structure they produce\textsuperscript{[19]}

- **hierarchical** clustering techniques produce a nested series of partitions, with a single cluster at the top of the hierarchy and all the clusters composed of the single starting elements at the bottom. A structure called dendogram is created (Figure 3).

- **partitional** (or flat) clustering techniques produce a one level (unnested) partitioning of the data (Figure 4).

Hierarchical clustering methods are preferable for detailed data analysis and provide more information than partitional clustering. Partitional clustering are less versatile than hierarchical methods, but they are preferable for their efficiency. Hierarchical clustering algorithms can be top-down or bottom-up. Bottom-up algorithms treat each text as a single cluster and then successively merge or agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all texts. Bottom-up hierarchical clustering is therefore called hierarchical agglomerative clustering. There is a form of hierarchical clustering called top-down clustering or divisive clustering. It starts with one cluster with all texts at the top. The cluster is split using a flat or partitional clustering algorithm. This procedure is applied recursively until each text is in its own single cluster.

Partitioning algorithm divide the set of n objects into k clusters so that the objective criterion is optimized. Objective criterion is usually chosen to minimize (maximize) some similarity function(e.g. distance) between objects. The most popular partitioning-based algorithm are K-means and its variants. The main purpose of text clustering\textsuperscript{[25]} in document browsing is that of describing and summarizing a larger set of documents. One of the the primary objectives of a clustering method should be to provide good, descriptive cluster labels. To distinguish this clustering from classical clustering, it is called descriptive clustering. Descriptive clustering
is a problem of discovering diverse groups of semantically related documents described with meaningful, comprehensible and compact text labels [25]. Two main algorithms used in descriptive clustering are: Lingo and Descriptive k-Means. The Lingo algorithm for clustering search results is capable to discover diverse groups of documents and at the same time keep cluster labels sensible. All candidate cluster labels in Lingo are discovered directly from the input text by selecting frequently repeated phrases and terms (so-called recurring phrases). Concurrently to candidate label discovery, a vector space model is built for all documents in the input. Lingo uses dimensionality reduction methods applied to the term-document space to find synthetic vectors that approximate topics present in the input documents. These vectors are then used to select final cluster labels from the set of all candidates. In the final step, the selected cluster labels are treated as queries to a conventional search system and documents matching a given label are assigned to it forming a final cluster [25]. One of the reasons the descriptive k-Means algorithm was created was to deal with thousands of short and medium documents.


2.3.4 Carrot$^2$

Carrot$^2$ is a library and a set of supporting applications that can be used to build search results clustering engine.

The Lingo, $k$-Means and STC algorithms are implemented in the Carrot$^2$ framework. The algorithms differ in terms of the main clustering principle and hence have different quality and performance characteristics.

The key characteristic of the Lingo algorithm in Carrot$^2$ is that it reverses the traditional clustering pipeline: it first identifies cluster labels and only then assigns documents to the labels to form final clusters. To find the labels, Lingo builds a term-document matrix for all input documents and decomposes the matrix to obtain a number of base vectors that well approximate the matrix in a low-dimensional space. Each such vector gives rise to one cluster label. To complete the clustering process, each label is assigned documents that contain the label’s words.

The other algorithm present in Carrot$^2$ is STC. The key data structure used in the Suffix Tree Clustering (STC) algorithm is a Generalized Suffix Tree (GST) built for all input documents. The algorithm traverses the GST to identify words and phrases that occurred more
Figure 5: Carrot2 Document Clustering Server quick start screen.

than once in the input documents. Each such word or phrase gives rise to one base cluster. The last stage of the clustering process is merging base clusters to form the final clusters.

Lingo and STC have two features in common. They both create overlapping clusterings, in which one document can be assigned to more than one cluster. Also, in case of both algorithms a certain number of documents can remain unclustered and fall in the Other Topic Group.

Bisecting k-means in Carrot$^2$[1] is a generic clustering algorithm that can also be applied to clustering textual data. As opposed to Lingo bisecting k-means creates non-overlapping clusters and does not produce the Other Topics group. Its current limitation is that it labels clusters
using individual words and not all cluster’s documents may correspond to the words included in the cluster label.

A comparison among the three algorithms with respect to key features is provided in table I.

2.3.4.1 Custom Data Clustering Tools

Carrot$^2$ offers a number of supporting tools that can be used to quickly set up clustering on custom data, further tuning of clustering results and exposing Carrot$^2$ clustering as a remote service:

- Carrot$^2$ Document Clustering Workbench: a standalone GUI application for experimenting with Carrot$^2$ clustering on data from common search engines or custom data.
- Carrot$^2$ Document Clustering Server: exposes Carrot$^2$ clustering as a REST service.
- Carrot$^2$ Command Line Interface: applications that allow invoking Carrot$^2$ clustering from command line.
- Carrot$^2$ Web Application: exposes Carrot$^2$ clustering as a web application for end users.

Carrot$^2$ Document Clustering Service (DCS) will be used in this thesis work, being the only solution that allow PHP integration. It can cluster documents from an external source (e.g. a

<table>
<thead>
<tr>
<th>Feature</th>
<th>Lingo</th>
<th>STC</th>
<th>k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster diversity</td>
<td>High, many small (outlier) clusters highlighted</td>
<td>Low, small (outlier) clusters rarely highlighted</td>
<td>Low, small (outlier) clusters rarely highlighted</td>
</tr>
<tr>
<td>Cluster labels</td>
<td>Longer, often more descriptive</td>
<td>Shorter, but still appropriate</td>
<td>Shorter, but still appropriate</td>
</tr>
<tr>
<td>Scalability</td>
<td>Low. For more than about 1000 documents, Lingo clustering will take a long time and large memory</td>
<td>High</td>
<td>Low, based on similar data structures as Lingo.</td>
</tr>
</tbody>
</table>
search engine) or documents provided directly as an XML stream and returns results in XML or JSON formats. Figure 5 shows Carrot\textsuperscript{2} DCS quick start screen.

2.4 Tags

In information systems a tag is a non-hierarchical term assigned to a piece of information. A tag is a metadata, that is \textit{data about data}, which allows systems to collocate related information, and helps users find relevant information by browsing or searching. Tags are created by users of the documents and media for their own individual use, that are also shared throughout a community.

This thesis uses \textit{descriptive metadata}, which identifies and organize information based on its intellectual content. Tagging is an important feature of many Web 2.0 services. Tags are terms in a flat namespace: there is no hierarchy, no specified parent-child relationships between these terms. Tags are keywords used by users to describe and organize content with any vocabulary they choose, because of this there are ambiguities. Ambiguity arises when the same tag is applied in different ways, or different tags are used for the same concept. If a vocabulary is shared throughout a community different users may apply terms to documents in different ways.

The activity of finding interesting content can be done by browsing or searching. Browsing is exploring a problem space to formulate questions and is opposed to direct searching, that is looking for answers to specifically formulated questions. Browsing a system and its interlinked related tag sets is great for finding things unexpectedly in a general area. Tagging is able retrieve the data and share data efficiently, it is considered more a categorization process than a classification process.

Categorization creates groups or categories whose members share some similarity. It is generally less rigorous and boundaries are less clear.

Classification schemes assign a term to one class within a group of mutually exclusive classes, it has clear relations and may be hierarchical. In tagging the set of terms is a flat namespace: there are no clearly defined relations between the terms in the vocabulary. Stewart Butterfield\textsuperscript{[26]} argues:

\begin{quote}
\textquote{Aside: I think the lack of hierarchy, synonym control and semantic precision are precisely why it works. Free typing loose associations is just a lot easier than making a decision about the degree of match to a pre-defined category (especially hierarchical ones). It’s like 90\% of the value of a proper taxonomy but 10 times simpler.}\
\end{quote}
Tags present in the websites are often displayed as tag clouds. A *tag cloud*[^27] is a visual representation for text data, used also to visualize free form text. An example of tag cloud is shown in Figure 6.

![Figure 6: An example of tag cloud with terms related to Web 2.0](image)

### 2.5 Return On Investment (ROI)

In finance return on investment (ROI) is the ratio of money gained or lost on an investment relative to the amount of money invested[^28]. The amount of money gained or lost may be referred to as profit or loss. The money invested may be referred to as the cost basis of the investment. ROI is usually expressed as a percentage. The general formula that defines return on investment is the following:

\[
ROI = \frac{\text{Benefits} - \text{Costs}}{\text{Costs}}
\]  

(2.2)

ROI is a measurement instrument to design strategies to achieve the maximum benefits. In the context of software projects, major benefits include increased stakeholder satisfaction, productivity, quality, cost savings, and cycle time reduction.
2.6 Agile Methods

Agile methods are an alternative to traditional project management that is the waterfall method, which assumes that every requirement of the project can be identified before any design or coding occurs. Programmers, testers, project managers, modellers, and stakeholders that build the system, need to work together effectively in such a manner that software is produced often and quickly, giving the users what they prefer. Only the stakeholders can tell what they want but usually they do not have the skills necessary to specify the system. Over time, the business environment changes, technology changes and the stakeholders will likely change their minds. Change is an aspect of software development that must be taken into account. Agile methodology allows developers to adapt to change during the software life-cycle through incremental, iterative work cadences, known as sprints. A sprint is a time period (can be anything from 2 to 8 weeks) in which the team develops software according to the requirements. During a Sprint, the team iterates towards a final product solution, improving on what they already have and adding some increments. The development during a Sprint is iterative and incremental. The Agile philosophy expresses its preferences in the Agile Manifesto\cite{29} where it values the items on the left (in bold) more than the items on the right:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

2.6.1 Agile system lifecycle

Modern Agile methods are in favour of an iterative, incremental and highly collaborative approach.\cite{3} defines the Agile methods as *serial in the large, iterative in the small*, and delivering incremental releases over time. The life-cycle is serial in the large because it identifies four phases:\cite{30}:

1. *Inception*: the goal of inception is to develop a shared understanding of the scope of the new system and to define a candidate architecture.

2. *Elaboration*: The goal of elaboration is to expand the team’s understanding of the system requirements and to validate the candidate architecture.
3. **Construction:** during construction the development of the system is completed.

4. **Transition:** during transition system testing is completed and the system is deployed to production.

Disciplines, that is activities development team members perform to build, validate, and deliver working software which meets the needs of their stakeholders, are iterative in the small and development releases are delivered at the end of each iteration. The disciplines are\[^3\]:

- **Model:** the goal of this discipline is to understand the business of the organization, the problem domain being addressed by the project, and to identify a viable solution to address the problem domain.

- **Implementation:** the goal of this discipline is to transform your model(s) into executable code and to perform a basic level of testing, in particular unit testing.

- **Test** the goal of this discipline is to perform an objective evaluation to ensure quality. This includes finding defects, validating that the system works as designed, and verifying that the requirements are met.

- **Deployment:** the goal of this discipline is to plan for the delivery of the system and to execute the plan to make the system available to end users.

- **Configuration Management** the goal of this discipline is to manage access to your project artefacts. This includes not only tracking artefact versions over time but also controlling and managing changes to them.

- **Project Management** the goal of this discipline is to direct the activities that takes place on the project. This includes managing risks, directing people (assigning tasks, tracking progress, etc.), and coordinating with people and systems outside the scope of the project to be sure that it is delivered on time and within budget.

- **Environment** the goal of this discipline is to support the rest of the effort by ensuring that the proper process, guidance (standards and guidelines), and tools (hardware, software, etc.) are available for the team as needed.

2.7 **Web Application**

A web application is an application that is accessed by users over a network such as the Internet or an intranet\[^31\]. Web applications usually follow the *client-server* architecture, and
two main software parts can be identified. The first part is coded in a browser-supported programming language and is reliant on a common web browser to render the application executable. The second part consists in the software that is running on an application server. Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility.\[31\]

2.7.1 Web Application Architecture

Web application are commonly composed of three logical parts, also known as tiers. Each of these tiers has a role assigned. Though many variations are possible, the most common structure is the three-tiered application. In its most common form, described in Figure 7, the three tiers are called presentation, application and data. A web browser is the first tier (presentation), an engine using some dynamic Web content technology (such as ASP, ASP.NET, CGI, ColdFusion, JSP/Java, PHP, Perl, Python, Ruby on Rails or Struts2) is the middle tier (application logic), and a database is the third tier (data). The web browser sends requests to the middle tier, which services them by making queries and updates against the database and generates a user interface.\[31\]
Figure 7: The three tiers of web application architecture
CHAPTER 3

EXISTING TOOLS AND METHODOLOGIES

3.1 Requirements Management Tools

A requirements management tool\[32] is a software system that helps users to manage various manually intensive tasks in the requirements development and requirements management processes. The decision of using requirements management tool may be based on the number of requirements or on the size of project teams\[32].

It is possible to distinguish three projects according to the number of requirements to manage:

- Small projects with less than 200 requirements
- Medium-size projects with 200-2000 requirements
- Large projects with over 2000 requirements

In small projects spreadsheets, wikis or simple databases are used. With these tools it is difficult to make coordinated and timely updates resulting in a lack of access to up-to-date information. There is a need of using a lightweight or middleweight requirements management tool.

Medium-size projects can benefit from using requirements management tools, because spreadsheets and databases do not provide the functionalities of requirements management tools.

Large projects need requirements management tools to avoid project delays and defects. For projects with a team size less than 5, spreadsheets or wikis would suffice. For larger teams and large distributed teams the use of good requirements management tools is required.

There are some commercial requirements management tools available in the market. An example for large project is the IBM Rational RequisitePro\[1]. This is a fully integrated system that is able to manage the entire life-cycle, it requires heavy investments.

The Accompa Requirements Management Tool is a commercial tool for medium-size projects. It is a flexible and easy to use tool, it doesn’t require heavy investments.

For small companies or small projects the best tools to be used can be spreadsheets, databases, email based collaborations and wikis.

---

It is important to understand that once identified, the requirements will not remain unchanged until the project is delivered, it is an evolution process, new issues will arise during the project life-cycle. Since the changes must also be recorded, a robust process in place for managing the requirements is needed. A robust change control process should include the following steps:\[32\] :

- Logging of request
- Analysis and assessment of requests
- Prioritization and authorization (or not) of request
- Negotiation for resources
- Implementation of request
- Documentation and dissemination of the change in a timely manner.

At every change that is made all the documentation need to be updated and then all the documents need to be re-issued so that all the stakeholders are working from the same set of documents. A requirement should have a status code in order to implement status tracking. Some of the status codes may be a defective requirement, a new requirement or a requirement that has been implemented.

3.1.1 Requirements attributes

In order to track and monitor requirements, their attributes, that is only the data that is going to be useful for the application, need to be defined. The following is a list of requirements attributes:\[32\] :

- **Unique ID**  the id of the specific requirement this must be a unique.
- **Date Created**  when the requirement was created . This helps to indicate whether or not the requirement was part of the requirements baseline or a later addition.
- **Current Version**  version control is important to ensure that everyone is working on the same version of the requirement.
- **Requirement Author**  the name of the user or class of user that raised the requirement.
- **Assigned To**  the person to whom the design/development task has been delegated. This allows the generation of reports by person so that progress and workload issues can be monitored effectively.
• **Requirements Status**  this should indicate the stage at which the requirement is for example proposed; verified; approved; deleted; implemented; and rejected.

• **Priority**  this should indicate either the system priority (mandatory/critical/optional) or the priority in relation to other requirements (High/Medium/Low).

• **Complexity/Difficulty/Risk**  the risk level of this requirement in relation to the success of the project.

• **Origin/Source of Requirement**  where the requirement originated.

• **Requirement Rationale**  the reason for inclusion/rejection of the requirement.

• **Stability**  the likelihood to the requirement changing for instance mature; stable; or still changing.

• **Due Date**  the date of delivery for this particular requirement.

• **Requirement Stakeholders**  the stakeholders that should be involved should any change requests be made that impact this requirement.

### 3.1.2 Traceability

Traceability[^33] is one of the essential activities of good requirements management. Traceability is used to ensure that the right products are being built at each phase of the software development life cycle, to trace the progress of that development and to reduce the effort required to determine the impacts of requested changes. There is forward and backwards traceability. Forward traceability looks at[^33] :

- Tracing the requirements sources to their resulting product requirement(s) to ensure the completeness of the product requirement specification.

- Tracing each unique product requirement forward into the design that implements that requirement, the code that implements that design and the tests that validate that requirement and so on. The objective is to ensure that each requirement is implemented in the product and that each requirement is thoroughly tested. Backwards traceability looks at[^33] :

- Tracing each unique work product (e.g., design element, object/class, code unit, test) back to its associated requirement. Backwards traceability can verify that the requirements have been kept current with the design, code, and tests.
• Tracing each requirement back to its source(s).

The classic way to perform traceability is by constructing a traceability matrix. A traceability matrix allows bidirectional tracing of requirements. It can be used at varying stages during a software project to ensure nothing is missed. Another mechanism for implementing traceability is trace tagging. Again, each requirement, each requirement source and each work product element must have a unique identifier. In trace tagging however, those identifiers are used as tags in the subsequent work products to identify backwards tracing to the predecessor document.

Benefits of requirements management tools There are many benefits of using requirements management tools, for example:

• Reduction in costs
• Reduction in time to market of products
• Quality improvements
• Increased levels of customer satisfaction

Just using a requirements management tool is not going to make any project successful, rather requirements management tools have the potential to improve the chances of a success. Some of the benefits requirements management tools tries to achieve are:

• Avoiding loss of requirements
• Avoiding unnecessary development
• Reducing defects
• Making the control and communication processes more robust

3.2 CRM Tools

3.2.1 Salesforce

Salesforce is a Customer Relationship Management tool by Salesforce.com Inc., a global enterprise software company. It provides CRM for sales, services and marketing, plus collaboration tools and an integration platform. It also offers contact and company management with
all the traditional information. It can integrate with company e-mail system so correspondence with customers and prospects is captured automatically.[34]

Lead tracking and opportunity management help to keep track of lead generation sources and any resulting opportunities. It is possible to track opportunities through a standard or custom sales process, and report on sales funnel by stage. Salesforce provides reminders and ticklers for follow up tasks such as phone calls, and for upcoming events or contract renewals.

1www.salesforce.com
It is possible to attach proposals or other documents directly to a contact or an opportunity. It allows to input reason codes that enable win/loss analysis.

Campaign management lets the user create and manage campaigns and to relate leads and revenue directly to the campaigns so he can finally measure marketing effectiveness.

Service management allows one to input tickets and track calls from start to resolution. Metrics such as average resolution time and problem frequency can be tracked.

Salesforce is a cloud-based application available from any browser. There are also iOS iPhone and iPad, Android, Blackberry and Windows Mobile versions for mobile access. Mobile devices can access Salesforce either through their native browser or the mobile app.

### 3.2.2 Highrise

Highrise\(^1\) is a “shared contact management” web application which supports basic CRM tasks\[^{35}\]. The application was launched publicly by the company 37signals on March 20, 2007\[^{36}\].

The application centers on person and company pages, which collate information such as images, notes, and contact detail. “Cases”, which are pages/categories within which related notes, images, and people can be kept\[^{37}\]. With Highrise it is possible to import contacts from Microsoft Excel.

Highrise does not integrate with email applications directly like some of its competitors. Instead, it uses a concept called a dropbox. Each user has a dropbox email address in their Highrise account. User can copy or blind copy this address on all email correspondence and Highrise will attach it to the right contact. Alternatively, it is possible to forward the email to the dropbox later and it will attach it to the contact then. It is also possible to attach email to a case to deal, including the case name at the top of the email\[^{38}\]. Highrise is accessible from all the main internet browsers, also from mobile devices.

### 3.2.3 SugarCRM

SugarCRM\(^2\) is a CRM system geared towards small to medium-sized businesses that is produced by the company of the same name. SugarCRM offers a free open source application known as Sugar Community Edition or as the for-pay Sugar Enterprise and Sugar Professional.

\[^1\]http://highriseshq.com/

\[^2\]http://www.sugarcrm.com/
The open source version of the customer relationship management software contains about 85% of the code found in Sugar Professional edition.

SugarCRM\textsuperscript{39} can be deployed as a cloud-based application which offers advantages such as accessibility and scalability. Two new editions are available: Corporate, which will occupy the space between Sugar Professional and Sugar Enterprise, and Sugar Ultimate, which will

Figure 9: Highrise welcome page
apparently offer everything there is in Sugar Enterprise and then some. Sugar Professional offers the same interface, dashboarding features and Microsoft Office plug-ins as Sugar Enterprise, as well as the administration, customization and integration features of the enterprise edition.

Both Sugar Professional and Enterprise offer unlimited online and email support, with premium support available. Sugar Enterprise offers the widest range of features. This version gives organizations the ability to use MySQL and Oracle as the back-end databases, and includes Sugar Mobile, which allows users and administrators to access and manage the software via an iPhone.

Sugar Enterprise also includes a self-service portal and SQL reporting features.

Sugar Community Edition is unsupported and does not come with any technical support. It includes a very limited set of dashboards and collaboration and reporting tools, and lacks the finer-grained security of the for-pay versions.

Figure 10: Example of the SugarCRM home page
CHAPTER 4

PROPOSED APPROACH

In this chapter we present the main contribution of this thesis. In Section 4.1 some of the main mistakes in software requirements management are highlighted, in order to point out how an inadequate management of stakeholders could turn into inadequate management of requirements.

Subsequently, in Section 4.2, the possibility of applying practices and methodologies of CRM to the context of software projects is discussed. Various functionalities have been taken into consideration to highlight possible analogies and contact points between an approach oriented to customers in the business world and an approach intended to be applied in the software project context.

In this way the concept of Stakeholder Relationship Management starts to take shape, and in Section 4.4 it is more extensively defined. That section outlines the application area of this approach and the kind of support it has to provide for software projects. This description is assisted by use cases, that define in a graphical way the fundamental functionalities.

In section 4.5 we discuss the possibility of defining the return on investment of the presented approach, which is a prerequisite for any business decision making. To do this, Agile methods are taken as a reference, because of their stakeholder-centric aspects.

4.1 Common mistakes in managing requirements: the importance of stakeholders

Mistakes in managing requirements can lead to project failure, software overspend and overrun. The most common are: \[32\]:

- *No comprehensive set of requirements*: going through the requirements management process of elicitation, analysis, specification, validation and management will help avoiding the problem.

- *Ambiguous requirements*: The probability that the requirements are interpreted differently by different stakeholders should be low. During validation all ambiguities should be removed, requirements should be well documented and include test cases and system requirements.
• Not getting all the stakeholders involved in the requirements process: it is very important to get all stakeholders involved, once identified they should be fully engaged with throughout the life-cycle of the project.

• No adequate managements of the requirements after the initiation stage: good management and control are required, there should be a process for reviewing, validating and approving change requests.

• Developers including functionality not in the requirements document: developers should be managed in the same way as the other stakeholders. Sometimes developers add un-requested functionality. This problem is mitigated by ensuring that traceability of the functionality is bidirectional to the requirements and to the code.

While the first two points in the list are more connected to the activities of requirements management (such as requirements elicitation, analysis, specification), the other three are more related to a poor management of the relationship with stakeholders.

Getting the stakeholders involved is a key aspect to consider for the success of a software project.

In the phase of reviewing, validating and approving change requests it is also important to get the feedback from the right stakeholders, that have to be involved and informed in an appropriate way.

As stated above, to avoid the presence of unrequested functionalities, developers should be treated as the other stakeholders: communication with them is crucial to ensure that their work is always aligned to the system goals, and just the requested functionalities are implemented.

All these aspects should be taken into consideration in order to get real value from stakeholders. As described in section 2.2, Customer Relationship Management represents a strategy that focus on customer needs and highlights the importance of the communication with the customer. For this reason CRM has been taken as reference for the the stakeholder-centric approach to the management of software projects that is described in this thesis work.

4.2 From CRM to SRM for Software Projects

In order to define the kind of support needed for an effective stakeholder relationship management, several practices and functionalities of CRM will be reviewed.
Operational CRM is the approach focused on “front-office” operations, namely all the activities that imply direct interaction with the customer. It makes use of customer contact history in order to store all the communication with the customer, and all the records relevant for the company, such as product choices and recorded preferences.

From this kind of approach SRM took the idea of storing all the relevant historical information regarding each stakeholder, in order to increase the knowledge about his preferences and his relationship with the project, and in order to avoid the waste of time of asking for the same information more than once.

Analytical CRM extracts, processes and analyses data gathered from Operational CRM to design marketing campaigns, analyse customer behaviour, manage decisions, generally making use of data mining techniques.

Analogous operations could be performed with data gathered from stakeholders. Interactions with stakeholder could be consulted in order to make decisions about requirements, and stakeholder behaviour such as his level of engagement, could be a relevant indicator to understand how to behave with him in future. Moreover, it could be useful to process stakeholders data in order to segment them, or to find relevant groups that share interests (stakes) or involvement in similar requirements.

Campaign management deals with the design of and management of promotional campaigns addressed to customer segments. Tracking, analysis and statistics are then performed to estimate the effectiveness of the campaigns.

Stakeholder segments or groups could be at the same way exploited to manage incoming and out-coming group communication. Stakeholders could be organized in groups based on different criteria, depending on the kind of communication that has to be performed.

In software projects the evaluation of ROI is not so common because of its complexity which is increased by the presence of factors that influence ROI in an indirect way, called intangible assets.

Before the implementation of SRM functionalities a decision was made: to check if there would be a return on investment (ROI). Unfortunately there was no data available to estimate it. To get a qualitative ROI estimate, the stakeholder centric Agile methodology was investigated instead, because the two methodologies share the same factors that enable a good ROI. This topic is discussed in section 4.5.
4.2.0.1 From Customer to Stakeholder

It is also useful to point out the differences between customers and stakeholders, in order to better understand the difference between the support provided by CRM and the kind of support that should be provided by SRM.

A customer\textsuperscript{40} is the recipient of a good, service, product, or idea, obtained from a seller, or supplier for a monetary or other valuable consideration. Customers of a company are the external entities who purchase the product.

A stakeholder\textsuperscript{41} is any individual or group who can be positively or negatively impacted by, or cause an impact on the actions of an organization.

Categories of stakeholders in software engineering, include end-users, managers and others involved in the organizational processes influenced by the system, engineers responsible for system development and maintenance, customers of the organization who will use the system to provide a service, external bodies such as regulators, domain experts, and so on. They will each have different goals, and will try to satisfy their own without recourse to the others\textsuperscript{6}.

4.3 Simplifying Assumptions

Since SRM involves a variety of Requirement Engineering topics, abundantly covered by studies and specific tools, it is important to outline which are the boundaries of this thesis work. Some simplifying assumptions have been made, in order to focus the efforts on the central topics.

4.3.0.2 Stakeholder Identification and Prioritization

The task of identifying and prioritizing the stakeholder is out of the scope of the thesis. The assumption of stakeholder already identified has been made: stakeholders and their roles are provided from an external procedure.

4.3.0.3 Requirement Elicitation and Prioritization

At the same way, there are several tools and methodologies dealing with requirement elicitation and prioritization. In order to not “reinvent the wheel”, requirement are also assumed to be already elicited.

4.3.0.4 Stakeholders and Requirements Format

Stakeholders of the projects are assumed to be provided with their essential information, such as name, surname, email address, stakeholder role and possibly connections with other stakeholders. The requirements are represented as simple strings. A more specific description can be added to each requirement. Requirement list has a hierarchical structure.
4.4 Use Cases and Proposed Functionalities

In this section the functionalities provided by the system are described, starting from a set of proper use cases.

4.4.1 Project Management

4.4.1.1 Use Case: General view of projects

When a project is selected from the project list, all its relevant information have to be accessed from its general view section: the overview, the general description, the requirement list and the stakeholder list (Figure 11).

![Figure 11: Use Case: General view of projects](image)

4.4.2 Requirements Management

For each project, it is possible to specify a hierarchical list of requirements. Requirement management is limited to very basic functionalities, summarized in the use case shown in Figure 12. User should be able to easily see and edit all the requirement information, such as description, stakeholder that are involved in it, the historical items related to it. Requirements can also be tagged in order to add more information.
4.4.3 **Stakeholder management**

Stakeholders have a central role in the system.

4.4.3.1 **Use case: general view of stakeholder information**

Figure 12: Use Case: General view of requirements

Figure 13: Use Case: General view of stakeholder information.
Project managers should have a complete view of the information about each of the stakeholders, such as their roles, requirements which they are involved in, their interests and stakes, their relationships with other stakeholders, the history of their involvement in the project (Figure 13).

4.4.3.2 **Use Case: Stakeholder’s requirements**

![Use Case: Stakeholder’s requirement display](image)

Figure 14: Use Case: Stakeholder’s requirement display

From the stakeholder’s view, it should be possible to see the requirements in which he is involved (Figure 14). Other requirements can be added, and it should be immediate to see which are the other stakeholders that have a stake in those requirements.

4.4.3.3 **Use Case: Stakeholder’s history**

As stated in Section 2.2, relatively to Operational CRM, it is important to keep track of all the relevant interactions with the customers, in our case the stakeholders. In this way there is no need to ask for information that have been already provided, and a more conscious and complete view of the communication is easily accessible from the project manager.
To “keep track” means to store in a structured way all the pieces of communication that have been exchanged between the stakeholder and the project manager. This is what is called stakeholder history (Figure 15). In order to support this, the system should provide automatic redirection of emails, and the possibility of storing also custom notes to manually save information (e.g. the summary of a phone call or the result of a personal meeting). It is also possible to add tags relatively to other involved stakeholders and requirements.

4.4.4 Tagging System

The concept and usage of tags in modern web 2.0 applications has been discussed in Section 2.4. In the system the use of tags is employed to add information to the basic elements such as requirements, stakeholders and historical items. Tags belong to a free dictionary, and is up to the user defining and adding them in a way that is helpful to better organize the entities in the system.

There are tags used to describe stakeholder interests and stake, and they refer to the knowledge project manager has about the stakeholder. Interest tags are also displayed in the stake-
holder’s overview section, and are useful to give a first look to the possible topics, functionalities or benefits the stakeholder is interested or implied in.

Other tags can be added to the requirements. Those tags have more the meaning of expressing the keywords of the requirement they refer. They are helpful to store the essential information about requirements in addition to their “conversational” description, and can be used, as well as the interest tags, to perform mining tasks to detect new relationships among stakeholders. Given that often the tags related to a requirement are just some words already present in their name or description, it could be helpful to employ some automatic feature extraction technique or tool in order to automatically tag requirements (or provide suggestions for tags) and save time.

Ultimately, tags allow to express relationships that go beyond the simple involvement of stakeholders with specific requirements.

4.4.4.1 Use Case: general tag view

Whenever a tag is selected, it should be possible to have immediate access to an overview of its presence in the system. Since it can be added to requirements, people or history items, all of these elements should be accessible from the general tag view(Figure 16).

4.4.5 Stakeholder Group Management

4.4.5.1 Use case: Manually group stakeholders

Stakeholders can be freely grouped according with project manager’s preferences. Groups can be either disjointed or overlapping. Since a group simply expresses a relationship among multiple stakeholders, each stakeholder can belong to different groups.

4.4.5.2 Use case: Automatic Stakeholders segmentation via clustering

As mentioned in Section 2.2, one of the features provided by CRM tools is to segment the customers which are found to have common needs, in order to better organize relationship with them and to differentiate the approach with each of the segments, depending on the marketing objective. Customer segments are then exploited to design specific marketing campaigns.

Applying segmentation strategies to the SRM approach is not so straightforward. While in the marketing world the objective is clearly to sell particular products to different group of customers, in the management of software projects stakeholders the goal is less definite.

In the case of this study, the objective of such functionality is to extract previously unknown knowledge about relationship among different stakeholders based on their interests, the requirements they are involved in and their communication history.
To perform automatic segmentation, an approach based on text clustering techniques has been chosen. For each stakeholder a *stakeholder profile* is created. It consists in a text document containing a set of tags that have the function of describing the stakeholder’s interests and stakes. The tags are easily took from the stakeholder’s overview and from the requirements he is involved in (Figure 17). The profiles are then provided to a text clustering tool, which process them and outputs the found clusters according to the selected clustering algorithm.
4.4.6 Communication Management

4.4.6.1 Filtering incoming communication

Once the communication has been stored in a structured way in the system, it can be exploited to get a better knowledge of the stakeholders’ needs. Communication can be filtered in order to have a fast access to history elements based on specific criteria. This functionality extends what is already possible with the stakeholders and requirements histories. It is the possible to combine different criteria such as tags involved, requirements and stakeholders in either inclusive or exclusive way (Figure 18).

4.4.6.2 Manage stakeholder segments for outgoing communication

The groups that have been defined with the functionalities described above, can then be exploited when managing outgoing communication (Figure 19). It should be possible to contact directly a single stakeholder or to communicate with all the people belonging to a group regarding their common interest. It should also be possible to set up an automatic alert in order
Figure 18: Use Case: Incoming communication filtering

Figure 19: Use Case: Outgoing communication management

to keep the stakeholders updated whenever there is an important change in a requirement they are involved in.
4.5 Return On Investment (ROI)

So far the importance of stakeholders has been considered crucial. But are there methodologies that apply SRM principles in software projects and that have an economic value in terms of return on investment (ROI)?

The answer is yes, and to show this the application of the Agile methodology to software projects has been examined. The choice of Agile methods has been made because, as highlighted in section 2.6, they are customer-centric or better stakeholder-centric, which is what has been proposed so far with CRM and SRM.

The work in this thesis aims at adding functionalities that make the SRM in software projects more valuable. Boehm in\cite{42} was critical about the management of many software projects and addressed the importance of stakeholders. The concept “build the software and the benefits will come” from The Information Paradox\cite{43}, assumes that the desired business outcome will happen automatically. This assumption is almost never met in practice: the right approach is to take into account, besides the software, system benefits. Evaluating cost and benefits is a prerequisite for investment decision making. This is just as true for software projects as for any other investment.

4.5.1 Recalling ROI

Return On Investment (ROI) is an indispensable tool for determining the business value of a system across its life cycle. We already defined ROI in section 2.5 as:

\[
ROI = \frac{(\text{Benefits} - \text{Costs})}{\text{Costs}}
\]

Costs include salaries and actual project effort. The goal is the improvement of software development and maintenance. There are two ways to increase business value or ROI:

1. increasing volume and revenue while maintaining current costs
2. reducing costs while maintaining current volume and revenue

In this thesis we focus on the second, specifically on the reduction of maintenance cost by means of early detection of software defects in development phase.

4.5.1.1 Productivity

Productivity is defined as the rate, speed or capacity of a software process\cite{44}. In our case, it is a measure of how fast and how many software products and services are rendered over a
given period of time\textsuperscript{[44]}. In the examples provided in this thesis, productivity will be defined as:

$$\text{Productivity} = \frac{\text{Lines of Code}}{\text{Hour}}$$  \hspace{1cm} (4.1)$$

For example, a productivity of 30 lines of code per hour (LOC/hour) means that, for every project hour, an average of 30 lines of code is produced. But this measure does not refer only to the work performed by the programmers. It relates the total lines of code produced by a team to the total amount of time spent on a project. In order to provide a clearer example, let’s assume that a team is composed of software analysts, architects, programmers and testers, all working on a project. In this project, software analysts have spent 50 hours developing requirements, architects have spent 50 hours on design, programmers have spent 50 hours on programming and the testers have spent 50 hours on integration, for a total of 200 project hours. If the programmers have produced 6000 LOC, the productivity is obtained by dividing the lines of code by the total number of project hours of the whole team. In our example, it will be $6000/200 = 30 \text{ LOC/hour}$.

### 4.5.1.2 Total Life-cycle Cost

Total life cycle cost\textsuperscript{[44]} is a method of determining both software development and maintenance costs. It quantifies analysis, design, development, test, and maintenance costs. Often software cost estimation estimates the time it takes for analysis, design, coding and testing while maintenance cost are not estimated. Almost never the benefits of a new software practice are estimated for the software maintenance phase of a software project. The maintenance time might be 20 times the development time, so the maintenance phase can be the part of the total life cycle where the greatest benefits can be found.

General formula of Total Life-cycle Cost is:

$$\text{TLC} = \text{development} + \text{maintenance}$$  \hspace{1cm} (4.2)$$

whose measure is expressed in hours.

### 4.5.2 Cost of change curve

Figure 20 depicts the traditional cost of change curve\textsuperscript{[3]}, which shows that the longer it takes you to find a defect then on average the more expensive it is to address it, the average cost rises exponentially the longer that you wait.
Agile techniques all have very short feedback cycles, often on the order of hours or even less, this minimize the risk and reduce the cost of addressing any defects. Traditional techniques have feedback cycles on the order of weeks or months, so increasing the risk and making a late and expensive discovery of mistakes. Feedbacks in Agile techniques map on the initial part of the curve where it appears almost flat, while feedbacks in traditional techniques map in the steeper part of the curve.

The greater the feedback cycle the greater the average cost of fixing a found defect. In traditional methods of software engineering as the Waterfall method studied in\textsuperscript{[4]}, the X-axis is divided in phases like requirements, design, code, development test, acceptance test and operation.\textsuperscript{[4]} showed that requirements errors are the most costly, the earlier in the development process an error occurs and the later the error is detected, the more expensive it is to correct.
Moreover, the relative cost rises quickly. As shown in the following table, an error that costs a dollar to fix in the requirements phase may cost 200 dollars to fix if it is not corrected until the system is in the maintenance phase.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Relative Repair Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>1-2</td>
</tr>
<tr>
<td>Design</td>
<td>5</td>
</tr>
<tr>
<td>Coding</td>
<td>10</td>
</tr>
<tr>
<td>Development test</td>
<td>20</td>
</tr>
<tr>
<td>Acceptance test</td>
<td>50</td>
</tr>
<tr>
<td>Maintenance</td>
<td>200</td>
</tr>
</tbody>
</table>

The SRM approach, focusing on the communication with stakeholders, aims to shorten the feedback cycle, as well as it happens in the Agile methods software development. In this way costs are reduced, benefits come from a better alignment between stakeholder needs and requirements implementation and ultimately ROI is positively impacted.

### 4.5.3 ROI for Agile Methods

Since the stakeholder collaboration, interaction and satisfaction is one of the most important points in this thesis, we take as an example the cost and benefits of Agile methods, which are a stakeholder-driven approach to software development.

#### 4.5.3.1 Costs

In a study reported by Rico\textsuperscript{[45]} the average productivity of an Agile methods team is 21.23 LOC/hour, if we assume a software size of 10,000 LOC, the effort to produce the software is 10,000/21.23 = 471 hours that translated in economic terms is a cost of $47,100 at $100/hour. The average defects with Agile methods is 1.7972 defects/KLOC, the maintenance effort in hours is 1.7972 x KLOC x 100 if it takes 100 hours to fix a defect. So it takes 1.7972 x 10 x 100 = 1,797 hours. The estimated maintenance cost is $179,72 at $100/hours. The total life cycle cost is the sum of the 471 hours for the development and the 1,797 hours for the maintenance, that is 2,268 hours. In economic terms the cost of using Agile methods is $226,80 at $100/hours for a software size of 10,000 LOC.
4.5.3.2 Benefits

There are two kinds of benefits: tangible and intangible. Tangible benefits are those that can be measured and quantified in financial terms while intangible benefits, such as stakeholder satisfaction, are not so easily measured. Tangible benefits can take the form of increased revenue or they might be expressed in terms of cost savings. Good quality of stakeholder collaboration and communication produce few defects during the software development phase, which means that the maintenance cost is low. If we assume it costs 100 hours of work to fix a defect during the maintenance phase, and we are able to decrease the number of defects let’s say by 10, the saving is 1000 hours. Agile methods were devised to increase the business value focusing on stakeholder interaction and teamwork, following the estimates of Rico\cite{45}, we will see their impact in economic terms. Traditional methods average about 0.85 LOC/hour furthermore they average 33.33 defects/KLOC, the total life cycle (TLC) is development + maintenance efforts. For a software size of 10,000 LOC or 10 KLOC the total life cycle is 10,000/.85 + 33.33 x 10 x 100 = 45,094 hours. The total life cycle cost for Agile methods that average 21.23 LOC/hours and 1.7972 defects/KLOC is 10,000/21.23 + 1.7972 x 10 x 100 = 2,268 hours. The benefits of the Agile methods is the difference between the two total life cycle costs, that is of the traditional methods and the Agile methods. In the above case 45,094 - 2,268 = 42,826 hours which is in economic terms $4,282,600 at $100/hour.

4.5.3.3 Putting all together: ROI

The idea behind ROI is that the benefits of an investment should be larger than its costs. In the case we examine, that is Agile methods, the benefits are larger than the costs. Traditional methods have large costs mainly due to great documentation efforts and lack of communications between stakeholders and developers, and this produces more errors. Agile methods involve stakeholder collaboration, teamwork and iterative development producing less errors and increasing productivity and quality. The total lifecycle cost of Agile methods is lower than traditional methods. For our model\cite{45} we assume a software size of 10,000 LOC or 10 KLOC and that the cost to fix a defect during the maintenance phase is 100 hours of work. Teams using Agile methods produce an average of 21.23 LOC/hour and 1.79 defects/KLOC\cite{45}. The total lifecycle cost is LOC/21.23 + 1.7972 x KLOC x 100 = 10,000/21.23 + 1.7972 x 10 x 100 = 2,268 hours. Traditional methods produce an average of 0.85 LOC/hour and 33.33 defects/KLOC\cite{45}; the total lifecycle cost is 10,000/.85 + 33.33 x 10 x 100 = 45,094 hours. The benefits of using Agile methods are the difference between the cost of traditional methods and the cost
of Agile methods that is 45,094 - 2,268 = 42,826 hours. Since the benefits of Agile methods are 42,826 hours and the costs are 2,268 hours, we use the formula ROI = (Benefits - Costs) / Costs = (42,826 - 2,268) / 2,268 = 17.88 that is ROI = 1788%.\[45\]
CHAPTER 5

SYSTEM IMPLEMENTATION

This chapter presents in a more detailed way the tool that has been implemented in order to apply the approach proposed in this work.

Section 2.7 will introduce the general design choices, explaining the reasons behind the choice of the web application architecture.

Section 5.2 will describe the database design, by means of the Entity-Relationship diagram of the system database.

Subsequently, all the system modules, which implement the previously proposed functionalities, will be presented.

5.1 The Web Application Choice

For the implementation of the tool we chose to develop a web application. We described the web application architecture in Section 2.7, and we already mentioned some of the reasons of its success.

In the case of the SRM tool, there are many reasons for the choice of web application. It allows easy access from the web and from multiple kind of devices, there is no need to install it locally, it is operative system independent, there is no need to update for the user, because the last updated version is always accessible online by all the users.

The tool has been implemented using standard web technologies such as HTML, CSS, jQuery for the client-side (browser) and PHP for the server side. For the database management PHP communicates with MySQL, that is compatible, reliable and open source.

5.2 Database Design

In Figure 21 is represented the Entity Relationship Diagram which describes how data are organized in the system database.
Figure 21: Entity-Relationship diagram of the system database
5.2.1 Entities

The main entities involved in the system are listed below.

5.2.1.1 Project

It represents the projects stored in the system. Each project is stored with a unique id, a name, a description, open and close dates (optional), the project status and a public attribute. Database entries have the following format:

\[
< \text{id}, \text{name}, \text{owner\_id}, \text{description}, \text{open\_date}, \text{close\_date}, \text{public}, \text{status} >
\]

5.2.1.2 Requirement

Each requirement is identified by a name, a description, the id of the project containing it and, the id of the father requirement (set to 0 when father requirement is absent.).

\[
< \text{id}, \text{name}, \text{description}, \text{father\_id}, \text{project\_id} >
\]

5.2.1.3 Stakeholder

The relevant information about the stakeholders are name, surname, email address and role.

\[
< \text{id}, \text{name}, \text{surname}, \text{email}, \text{role} >
\]

5.2.1.4 Tag

Tags are saved with their unique name and a weight to express how many times they are present in the system.

\[
< \text{id}, \text{name}, \text{weight} >
\]

5.2.1.5 History Item

It refers both to emails and manually entered notes. The stored attributes are the date and time (in order to sort the items in temporal order), the content, the subject (that is the title for notes), the item type (note or email), and, just for the emails, the sender and receiver.

\[
< \text{id}, \text{time}, \text{subject}, \text{type}, \text{content}, \text{sender}, \text{receiver} >
\]
5.2.1.6 **Stakeholder group**

Represents the stakeholder groups and it is described by a name, an optional description and the project identifier.

\[
< \text{id, name, description, project\_id} >
\]

5.2.2 **Relationships**

5.2.2.1 **Involvement**

Expresses the involvement of a stakeholder in a requirement. Each entry has the format:

\[
< \text{stakeholder\_id, requirement\_id} >
\]

5.2.2.2 **Stakeholders Relationships**

Expresses the relationship of personal knowledge or recommendation between stakeholders. Entries has the format:

\[
< \text{stakeholder\_id\_1, type, stakeholder\_id\_2} >
\]

5.2.2.3 **History Relationships**

Represent the connection between history items and the other entities in the system, namely stakeholders and requirements. There are two database tables, one for the stakeholders with the format:

\[
< \text{history\_item\_id, stakeholder\_id} >
\]

and the other for requirements, with the format:

\[
< \text{history\_item\_id, requirement\_id} >
\]

5.2.2.4 **Tagging Relationships**

Describe the connections between tags and the tagged elements in the system. There are three tables expressing these relationships. The first is for tagged stakeholders and has the format:

\[
< \text{tag\_id, stakeholder\_id} >
\]
the second is for tagged requirements and has the format:

\[ < \text{tag} \_ \text{id}, \text{requirement} \_ \text{id} > \]

the third is for tagged history items, and has the format:

\[ < \text{tag} \_ \text{id}, \text{history} \_ \text{item} \_ \text{id} > \].

5.2.2.5 Stakeholders Grouping Relationships

Connects the stakeholders with the groups they belong to. The table representing this relationship has the following format:

\[ < \text{stakeholder} \_ \text{id}, \text{group} \_ \text{id} > \]

5.3 Module details

Given the main entities and the functionalities described in Section 4.4, the web application has been organized in five main sections, as displayed in Figure 22.

Figure 22: Web application sections
Those section are:

- **Projects section**: provides access to all the stored projects.
- **Contacts section**: provides access to all the contacts present in the systems. Contacts can be stakeholders of multiple projects.
- **Tags section**: allows direct access to the *tag-cloud* representing all the tags present in the system.
- **Communication section**: lets the user go directly to the communication functions, such as communication history filtering and outgoing communication management.
- **Manage Group section**: allows instant access to the stakeholder group management.

Most of the sections in the tool are organized as “tabs”, to provide an intuitive navigation experience. In order to obtain this graphical effects, the jQuery Tab\(^1\) Plugin has been used.

\(^1\)http://jqueryui.com/tabs/
5.3.1 Project Management

As shown in Figure 23, selecting the Projects section it is possible to see the list of the existing projects with their general data. Clicking on the action button it is also possible to edit general information about them.

![Figure 23: Project list](image)

When a specific project is selected, the overview can be consulted (Figure 24), and clicking on the other tabs it is possible to see all the other relevant data about the project, such as description, stakeholders and requirements. With the specific button it is possible also to add or remove stakeholders directly choosing them from the contact list.

5.3.2 Requirements Management

Requirements are displayed in a hierarchical list and can be easily edited or deleted directly from the list view with the option button (Figure 25). In addition, it is possible to add sub-requirements in order to specify a larger level of accuracy in the needed functionalities.
Clicking on the requirement name or on the specific link from the option button, it is possible to see the requirement view. All the relevant details about the requirement are shown, such as description, involved stakeholders, tags, and related history items.

### 5.3.3 Stakeholders Management

#### 5.3.3.1 Stakeholder’s overview

As well as for projects, when a stakeholder is selected, his overview is displayed, and all the basic information such as name, role, email and interest tags are visible (Figure 26).

Selecting the specific tabs is possible to navigate among all the data related to that stakeholder which are stored in the system: the requirements he is involved in, the history of its interaction with the system, the other stakeholder he is connected to and the other projects that count him among the stakeholders.

#### 5.3.3.2 Stakeholder’s requirements

Selecting the Stakeholder’s requirements tab, the user can see which requirements the stakeholder is involved in. As shown in Figure 27, those requirements are highlighted in the requirement list with a different color.
Figure 25: Project requirement list

5.3.3.3 Stakeholder Communication History

Figure 28 shows an example of the stakeholder’s history tab. Under this section user can have a summary of all the past interactions between the stakeholder and the system. The email are stored in the database by means of a functionality similar to the “dropbox” mail address present in some CRM tools such as Highrise(see Section 3.2). To each project an email
address is associated. The project managers (or, in general, the users of the application in charge of managing communication), use that email address to redirect emails which are sent to and received from the stakeholders. Periodically (or whenever there is the need for the application to provide the history) a PHP script accesses to the email inbox to check whether
unread messages are present. The emails are then parsed and stored in a structured way in the database. From the user’s point of view, sent emails are stored in the system simply adding the “dropbox” email address as CC in the email, while the received ones need to be forwarded.

In this way project managers can continue using their personal email clients without the need of adopting another interface or email address in the application.

In order to store interaction occurred by means of other communication media (e.g. phone or skype call, personal meeting, text messaging), it is possible to add simple textual notes that can be dated and tagged.
Figure 28: Stakeholder’s history
5.3.4 Tagging system

The usage of tags, as remarked in Section 2.4, is useful to categorize elements and organize them in the system. In the database design (section 5.2) it has been described how the association between tags and other entities in the system is performed. The elements that can be tagged are stakeholders, requirements and also history items. Tags can be added and removed any time when editing an element. Figure 29 shows an example of the popup window containing the form that allow to edit stakeholders. Thanks to the specific jQuery Tag-it plugin\(^1\), the graphical appearance of tags is pretty similar to the ones present in the most common web 2.0 websites, enabling user to be immediately familiar with the tagging usage in the tool.

Tags are visible in the overview section of all the elements, and also in the main Tags section, always reachable from the main tabbed menu. In this section all the tags present in the program are displayed in the tag cloud format (mentioned in Section 2.4). Figure 30 shows an example of a tag cloud generated in the tool with the existing tags. For the tag cloud feature, the dedicated jQuery plugin\(^2\) has been used.

5.3.5 Communication Management

The Communication Management functionalities are divided in two sections: History and Communicate (Figure 31).

The History section allows to browse and filter all the historical elements (emails and notes) stored in the system. Four different filtering criteria can be applied in order to locate the desired communication items.

The Communicate section is dedicated to configure the outgoing communication with the stakeholder, specifically to set up email alerts in case of changes in requirements. As previously stated, the generic communication with stakeholders is managed from the personal email account “outside the tool” and is stored in the system thanks to the “dropbox” functionality. Requirement email alert is the only feature that the user has to use from the tool to communicate with the stakeholders. It mainly consists of defining a template that has been sent to stakeholders, in order inform them about the changes. Once the template is set and the alert active, whenever a requirement is modified, the system asks if involved stakeholders should

\(^1\)http://aehlke.github.com/tag-it/

\(^2\)http://addywaddy.github.com/jquery.tagcloud.js/
be informed about those changes. This functionality is not yet completed and its graphical interface will not be displayed in the thesis.

### 5.3.6 Group Management

As for Communication, also the Manage Groups functionalities are divided in two sections: the Manual and the Automatic group management.

#### 5.3.6.1 Manual Group Management

Manual group section, as can be seen in Figure 32, provides an essential and intuitive interface to manage the groups created in each project. Groups can be viewed, edited or deleted with the option button, and new groups can be added, choosing manually from the stakeholder list.
Other filtering options to group stakeholders based on specific criteria, such as roles, requirements or tags, are features in phase of development.

**5.3.6.2 Automatic Group Management**

The automatic group management function makes use of clustering techniques in order to automatically segment stakeholders. As described in Section 4.4.5, for each stakeholder a
profile is created, that is a text containing all the relevant tags about his interests or stakes and about the requirements he is involved in. User can choose which of these tags to take into consideration selecting the proper radio button (Figure 33).

The select menu allows to choose which clustering algorithm to use in order to perform the grouping. User can choose among the three algorithms provided by the Carrot\(^2\) clustering tool, described in Section 2.3.3: *Lingo*, *STC* and *K-Means*.

Differently from *Lingo* and *STC*, *K-means*, being a partitioning-based algorithm (see Section 2.3.3), provides non-overlapping clusters and is therefore the best choice if there is the goal of finding non-overlapping segments of stakeholders.

When the “Group Them All” button is clicked, the profiles of all the stakeholders of the selected project are encoded in an *XML* stream, which is sent to the *Carrot*\(^2\) DCS (Document Clustering Server). The request is then processed and the clustering results are sent back to
the web application in XML format. The XML is parsed and displayed to the user as shown in Figure 33. The jQueryUI Accordion\(^1\) Plugin has been used to have this graphical effect.

\(^1\)http://jqueryui.com/accordion/
At this point the user can choose whether or not to add any of the suggested groups, simply by selecting the “add group” button.
CHAPTER 6

EVALUATION

In order to evaluate the work an empirical test on a real world project should be performed. Unfortunately, it was not possible to find a suitable project with people available to collaborate. For this reason, in this chapter just a discussion about the tool functionalities is provided.

As in the previous chapters, the use cases will be taken as a reference, to highlight the positive as well as the negative aspects of the work performed.

6.1 Tool Evaluation

6.1.1 Tool Usability

The tool has been designed to be as simple and intuitive as possible. The “tabbed” structure is easy to navigate and all the functions have been kept very essential.

However, some of the functionalities are not complete, and both the graphical aspect and the level of customizability of some modules have to be improved.

The access to stakeholders and requirements data is fast and straightforward, but there is still a low level of interactivity when there is the need of editing data.

6.1.2 Use Cases

6.1.2.1 Project Management

The project management provides all the functions that have been identified in the use cases. Overview, requirements and stakeholders are immediately accessible from the project section.

6.1.2.2 Requirement Management

The requirements have been managed in a very simple way, not being the main subject of the thesis work. They are easily editable and the requirement view allows to know everything is connected with them in the system.

6.1.2.3 Stakeholder Management

All the needed “Operational CRM” functionalities described in the use cases have been implemented quite successfully.
6.1.2.4 Tagging System

The tagging system represents a useful feature to provide semantic interconnections between elements that are not directly connected by the normal “involvement” relationship.

A simple and immediate categorization can be made with a good use of tags.

Tagging system is suitable for small and medium projects, while for large projects it may lose its efficiency.

To mitigate the problems in the case of a large number of requirements, whose tags are often just keywords already present in their description, the usage of automatic feature extraction techniques should be investigated.

6.1.2.5 Stakeholder Group Management

The group management section would be complete with the addition of features that allow to create groups with the support of filters (e.g. filtering by role or tags or requirements).

The automatic group management is one of the most interesting features of the tool. It has the aim to provide suggestions for the grouping that are not trivial, and its usefulness, under this point of view, should grow with the dimensions of the project. It is possible to discover more about the usefulness and effectiveness of clustering using it with large amount of data.

6.1.2.6 Communication Management

Communication has been highlighted as an important aspect for the stakeholder management especially given the drawbacks of most of the stakeholder analysis methods.

The implementation choice, for this tool, has been to give great relevance to the stored communication, most of times referred as history, that can always be consulted and filtered with the dedicated modules.

Particular is the case of the outgoing communication, where the use of the tool could be limited to the case of automatic alerts in case of changes in requirements and communication with groups defined inside the tool. For general communication the “dropbox” function provided allows to exploit the personal, more powerful mail client and simply store the important pieces of communication by redirecting emails to the dropbox address. On the other hand, if the user wants to use exclusively the tool to send emails, it has some limitation due to the partial implementation of these feature. The management of the contacts import/export between the system and other messaging tools has not yet been implemented.
CHAPTER 7

CONCLUSION

This chapter provides a summary of the thesis, discussing the contributions and limitations, and suggest possible extensions and topics to be investigated for a future development of the work.

7.1 Future Work

In this section the possible extensions and improvements of the work are described.

7.1.0.7 Short Term

Some features have not been included in the tool, but should be added to provide a better support in project management.

- **Considering Stakeholders Priority.** So far in the system all the stakeholders have been considered as equally important. With few Changes it should be possible to take into consideration prioritized stakeholders and requirements, as provided from proper stakeholder identification tools such as StakeSource\(^{[46]}\).

- **Email Templates.** Most of times the lack of involvement and collaboration from the stakeholders is due to the inaccurate usage of communication. An example are the emails, should be composed in an effective way in order to get the desired feedback from the correspondent. The presence of proper email templates could be useful to project managers that would have predefined examples to personalize in order to write effective and attractive emails.

- **Automatic Feature Extraction.** Automatic feature extraction techniques of Information Retrieval could be exploited to provide automatic tagging from requirement names and description in order to deal with large volumes of data.

- **Data Import.** So far it is possible to import stakeholders and requirements from simple csv files, but the function should be improved. Contact import from standard format, such as excel tables, could be useful in order to easily synchronize the contacts in the SRM system with the contacts of any other contact management tool.
• **Data Export.** At the same way, exporting data in a well structured format could be useful to produce projects report and availability of data for other systems.

• **Defect Tracking for ROI.** Among the marketing features typically present in CRM tools, there is the estimate of the effectiveness of marketing activities, and the measure of marketing ROI, identifying trends in response rates, measuring sales results and viewing the status of related opportunities. As discussed in section 4.5, a good way to increase ROI in software projects is to minimize the cost of late correction of errors (or generally defects). It could be useful to devise a way to classify and track defects, because they are crucial to estimate the ROI of any set of activities aimed to deliver higher quality software. The data to be stored about defects may be:
  
  – Defect description
  – The software phase in which the defect was discovered
  – The software phase in which the defect was fixed
  – The requirement affected.

The knowledge about the phases in which the defects are discovered and fixed is necessary to estimate the total life-cycle cost.

• **Advanced Stakeholder Profiling.** Other features such as response tracking links in emails could be adopted to profile stakeholders and keep track of their level of collaboration. This could highlight which stakeholders are more effectively involved and which ones should be handled in a different way in order to obtain their collaboration.

7.1.0.8 **Long Term**

• **Collaborative interaction with stakeholders.** It should be possible, in order to increase involvement and collaboration of the stakeholder, to develop a complete Web 2.0 platform which allows direct interaction with stakeholders, possibly integrating with existing wide-spread social network frameworks.

• **Requirement Change Management** It could be useful to enhance the level of detail and the structure of requirements in order to provide a stakeholder oriented support system to deal with changes in requirements. Currently, the only function provided in the tool is the setup of automatic alerts whenever a requirement is substantially modified.
7.2 Thesis Summary

Chapter 1 presents the problems that motivates this work and define the objectives, introducing how the work will be performed in the remaining chapters.

Chapter 2 provided the basic knowledge needed to understand the concepts exposed in the following chapters and the related problems.

In Chapter 3 is provided an overview of some the most common tools and functionalities in the fields of Requirements Engineering and Customer Relationship Management, in order to understand what is the state of the art in those contexts.

Subsequently Chapter 4 presented the proposed approach of this work. Starting from some stakeholder-related problems of Requirements Engineering, it has been discussed how the functionalities and the practices of Customer Relationship Management could be applied and adapted to stakeholder management in software projects. The Stakeholder Relationship Management needed support is then defined with the usage of proper use cases.

In Chapter 5 the implementation of the tool which applies the proposed approach is described with examples that refers to the previously defined use cases.

Finally, in chapter 6, a critical analysis about the approach and its implementation has been performed, outlining the achievements and the limitations of the presented work.

The main problems addressed by this thesis work are the lack of involvement, the short lifetime of relationships and the inaccurate management of the little information available that characterize the stakeholder management in software projects.

The work identified the kind of support needed to apply CRM principles to the context of stakeholder management, and provided a basic version of a tool that implements those functionalities. Features like communication tracking, tagging system and the cluster analysis resulted a valid support for the management of stakeholders in small and medium size projects. Although for larger size projects many functionalities should be reviewed and extended, these functions constitute the basis for a novel approach to stakeholder analysis.

The main contribution of this thesis is to propose an approach that is able to leverage the relationship with the stakeholders to provide added value to the requirement management. In order to give an empirical evaluation of its advantages and drawbacks, this approach should be used in real software projects.
CITED LITERATURE


VITA

NAME: Francesco Marconi

EDUCATION: Bachelor of Science in Engineering of Computing Systems, Politecnico di Milano, 2009
Master of Science in Engineering of Computing Systems, Politecnico di Milano, 2013
Master of Science in Computer Science, University of Illinois at Chicago, 2013