Copyright

by

Teema Chandrakant Benadikar

Spring 2011

The Thesis Committee for Teema Chandrakant Benadikar
Certifies that this is the approved version of the following thesis:

Managing Software Requirements for Small Sized Company

APPROVED BY

SUPERVISING COMMITTEE:

Supervisor: ______________________________

Anthony P Ambler

____

Bruce McCann

Managing Software Requirements for Small Sized Company

by

Teema Chandrakant Benadikar

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2011

Dedication

This thesis is dedicated to my parents and husband.

Abstract

Managing Software Requirements for Small Sized Companies

Teema Benadikar, MSE

The University of Texas at Austin, 2011

Supervisor: Tony Ambler

This thesis considers the requirement engineering process from the small scale industries point of view. It begins with a brief introduction to software requirements and then proceeds to a detailed study of the requirement engineering process. The later part of the thesis considers Business Process Modeling and how it helps in understanding any business in a better way and how conceptual model helps in extracting business requirements from any particular business scenario. The Object Oriented Technique is used for building the conceptual model. The thesis concludes with a case study.

Table of Content

1 What are Requirements and Types of Requirements?

According to the study conducted by Boehm on Requirement Engineering practices in the early 80ies, catching errors in the requirement phase costs 10 to 100 times less than catching errors in the development and maintenance phases. Software requirements define the business problem to be solved. Software requirement define most importantly what a system must do and the activities it must be able to perform. The definition of a successful program is that it is 100% compliant with its initial requirements. If these requirements are incorrect, incomplete, and ambiguous or are unclear then it is difficult to correct the problem in the later stages of the software development. In particular understanding a requirement well drives the effort required to develop the software, the time required to complete the software and the inherent reliability of the software. Effective and accurate requirement gathering leads to more user satisfaction.

1.1 Definition of Software Requirement

Although many definitions of software requirements have been used throughout the years, the one provided by requirement engineering authors Dorfman and Thayer[1990] is quite workable:

1) A software capability needed by the user to solve a problem to achieve an objective.

2) A software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documentation.

Software requirements as defined by the IEEE standard glossary of software engineering terminology as being:

1) A condition or capability required by a user to solve a problem or achieve an objective;

2) A condition or capability that must be met or possessed by a system component to satisfy a contract, standard, specification, or other formally imposed document;

3) A documented representation of a condition or capability as in 1 or 2.

In other words, requirements are the features of the system that describe what the system is capable of doing in order to fulfill customer needs.

The requirements of a system define what it needs to do- but not how.

1.2 Levels of Requirements

Requirements are identified at several levels – business, user, functional and non-functional requirements.

Business Requirements: constitute high level requests from users to the developers of the system. Executives and managers might define the business requirements that will help their company operate more efficiently and compete successfully in the market place.

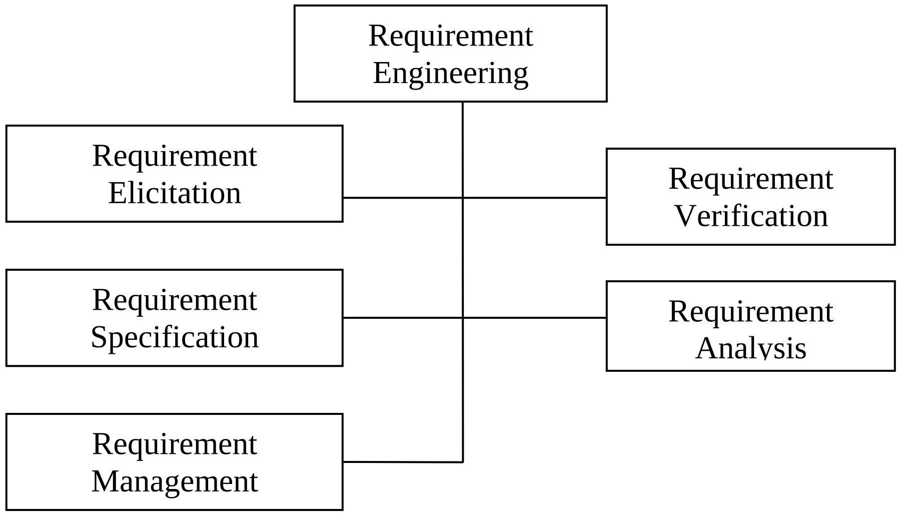
User Requirements: present a set of tasks that a user should perform while using the system. These requirements describe the functionality and the characteristics of the product in various dimensions that are important to the users and developers.

Functional Requirements: describe the interaction between the system and its environment. These requirements are stated in terms of input and outputs to the software or between elements of the software.

Non-functional Requirements: describe restrictions that are imposed on the system. These requirements include standards, regulations and contracts to which the system must confirm. Nonfunctional requirements place constraints on how the functional requirements should be implemented.

2 Requirement Engineering Process

The Requirement engineering process is the processing of the requirements right from the beginning to the end of the software development. The following diagram will illustrate the Requirement Engineering process more clearly.



2.1 Requirement Elicitation

This is the most important step of the Requirement Engineering process. In this step the user and the developer discover, review, articulate, and understand the users need and the constraints of the software and the development activity. This step helps the user and developer know what can be expected from the software to be developed. During this phase a lot of initiative has to be taken from the developer side.

Requirement Elicitation Techniques:

1) Interviewing and Questionnaire

2) Requirements Workshop

3) Brain storming and Idea Reduction

4) Story Boards

5) Use Cases

6) Role Playing

7) Prototyping

1) **Interviewing and Questionnaire:** In this technique the developer conducts an interview with the user based on a questionnaire with context-free questions. Such as 'Who is actual user?' 'What are their needs?'

This technique helps in

a) Knowing the customers' needs well

b) Assessing the problem well

c) Understanding the user environment

d) Knowing the user's understanding about the software

e) Easier in suggesting solutions

2) **Requirements Workshop:** In this technique all the key stake holders and developers come together for a short but intensely focused period where an exchange of ideas take place. An agenda is decided and a format is laid out for working. An outside experienced facilitator ensures the success of the workshop. The facilitator helps in carrying the workshop efficiently by:

    a) By taking care that the session takes a good progressive route as it advances.
    b) Controlling the workshop well, by sticking to the agenda throughout the session.
    c) Seeing that each stakeholder participates in the workshop.
    d) Managing the time of the workshop.

3) **Brainstorming and Idea Reduction:** This technique involves both idea generation and idea reduction. Various voting techniques are used to prioritize the ideas. This is a very effective requirement elicitation technique because many creative and innovative ideas come from seemingly unrelated ideas. The idea reduction phase helps in pruning the ideas that are not worthy of further discussion and grouping the similar ideas together.
4) **Storyboarding:** Storyboarding identifies the players, explains what happens to them and how it happens. Its helps resolve the "yes, but" reactions in the requirement elicitation phase.
5) **Use Cases Techniques:** This is the most common technique used in the requirement elicitation phase. This phase like the storyboarding technique identifies the who, what and how of the system behavior. It describes the interaction between the user and developer. Giving an idea of what the software does for the user. It gives an idea of the systems/software's functional behavior.

6) **Role playing:** Role playing techniques allows the user and the developer exchange roles and thus understands each other's perspective better. It helps the user understand the constraints of the software.

7) **Prototyping Technique:** In this technique a partial implementation of software is done. It helps the user and developer understand the requirements in a better way. Prototyping takes care of the fuzzy requirements those that are poorly defined or poorly understood.

Challenges faced during the Requirement Elicitation step:

1) **"Yes, but" syndrome –** This problem stems up from the human nature. The user is not necessarily an IT friendly person. Sometimes they fail to understand constrains and limitations of the software. They are unable to see the software as a physical device. Thus this problem comes up when the expectations from the software and the developers are too high and the functionality of the software is not clear to the user.

2) **The undiscovered ruins syndrome – "**In many ways, the search for requirements is like a search for undiscovered ruins: the more you find, the more you know remain" [Dean Leffingwell, Don Widrig]. If the requirements are discovered in the later stages of the development phase it gets difficult to take care of them then, thus increases the cost of the software.

3) **The User and developer syndrome -**This takes place because of miscommunication between user and developer. The user many times is not able to articulate his /her requirements well thus making a completely different demand. Analysts think they understand the user's problem better than the users, thus creating confusion. Techniques like storyboarding and role play help in dealing with this kind of situations. They also help in understanding the requirement from multiple viewpoints, thus giving a clearer picture.
4) **Living with the sins of your Predecessors –** The user's past bad experiences with software development can make situations difficult and make it difficult to gain trust from the users.

Requirement Elicitation guidelines:

1) Assess system feasibility.
2) Be sensitive to organizational and political consideration
3) Take multiple viewpoints into consideration
4) Record requirement sources
5) Identify and consult system stakeholders.
6) Prototype poorly understood requirements
7) Reuse requirements.
8) Look for domain constraints

2.2 Requirement Analysis

In the requirement analysis step the requirement gathered in the Requirement elicitation step are studied and analyzed. This is a very critical step in the software development process. Steps in Requirement Analysis phase:

1) **Fix software boundaries-** In this step a study is made of all the requirements collected from the stakeholder. This study helps in understanding and setting up boundaries to the software project.
2) **Identify the user –** From all the data collected during the requirement elicitation step, it becomes easy to identify which user is doing what. Study the requirement from each user and have a brief idea as what is the requirement of each user.

2.3 Requirement Specification

Requirement Specification is the step in which all the requirements that are understood and analyzed are put on a document. This document is often called Software Requirement Specification (SRS). It is a proof of all the requirements that are finalized by the user and developer. SRS specifies the functions and the capabilities that will be carried by the software.

A well designed SRS specifies the functional and non-functional requirements only it doesn't specify the design suggestions. SRS works as a guideline in moving ahead with the development work. It acts as checklist and help in ensuring that all the requirements are taken care off. “The SRS often works as the ‘Parent’ document because all subsequent documents like the design specification, statement of work, design architecture specification, testing and validation plans and documentation plans are related to it”, article by Donn Le Vie.

A well designed and well-structured SRS takes care of four things:

1) SRS provides a proper feed back to the users. It is an assurance to them that the development organization understands the issues and the software behavior necessary to address those problems.
2) SRS   helps in organizing the data, drawing borders around the problem, solidifying ideas and breaking down the problem domain in various components.
3) SRS serves as an input to the design specification.
4) SRS also serves as a product validation check.

Content of Software Requirement Specification (SRS):

Several standard organizations (including the IEEE) have identified nine important key elements that should be taken care of while writing and designing an SRS:

1) Interfaces
2) Functional Capabilities
3) Performance levels
4) Data structures/Elements

5) Safety

6) Reliability

7) Security/Privacy

8) Quality

9) Constraints and Limitations

A good SRS document should include the below mentioned things:

1) Template

2) A method for identifying requirements and linking sources

3) Business Operation rules

Let's consider each in detail:

1) **Template:** A template basically provides an outline for the SRS document. One can use the existing template and modify it according to the current project or can completely design a new template from scratch.

The following table shows the basic SRS template:

1. Introduction

1.1 Purpose

1.2 Document Convention

1.3 Intended audience

1.4 Additional Information

1.5 Contact information/ SRS team members

1.6 References

2 Overall Description

2.1 Product perspective

2.2 Product functions

2.3 User classes and Characteristics

2.4 Operating Environment

2.5 User Environment

2.6 Design/Implementation constraints

2.7 Assumption and Dependencies

3 External Interface Requirements

3.1 User Interface

3.2 Hardware Interface

3.3 Software interface

3.4 Communication Protocols and interfaces

4 System Features

4.1 System Feature A

4.1.1 Description and Priority

4.1.2 Action/result

4.1.3 Functional Requirements

4.2 System Feature B

5 Other Non-Functional Requirement

5.1 Performance Requirement

5.2 Safety Requirement

5.3 Security Requirement

5.4 Software Quality Attributes

5.5 Project Documentation

5.6 User Documentation

6 Other Requirements

Appendix A : Terminology/Glossary/Definitions List

Appendix B : To be determined

Taken from the article by Donn Le Vie, Jr on "Writing Software Requirement Specification".

2) Identify and link requirements with Sources: As mentioned earlier SRS contains functional and non-functional requirements. These functional requirements have a source. It is very important to identify and link the sources because it justifies the requirements and prevents unwanted and spurious requirements from interrupting in between.

To link requirements to their sources, each requirement included in the SRS is identified by a unique identifier that can remain valid over time as requirements are added, deleted or changed.

3) Establish business rules for contingencies and responsibilities: It is important to take into consideration that the most thought out requirement may also change due various reasons. Hence it is important to keep room for such known or unknown contingencies. An SRS has to anticipate the change and keep an action plan ready. For example, if a customer wants to change a requirement that is tied to a government regulation, which may not be ethical and/or legal to do so. A project manager may be responsible for ensuring that a government regulation is followed as it relates to a project requirement .However if a contingency is required, then the responsibility for that requirement may shift from the project manager to a regulatory attorney. The SRS should anticipate such actions to the furthest extent possible. Thus a systematic SRS acts as a good guideline for the entire Software Project.

Ref. Article by Donn Le Vie on “Writing Software Requirement Specification”.

2.4 Requirement Verification/Validation

Verification phase tells us if we are building the product right. It means that each “stage” of development follows processes and standards. Quality is the main goal. Validation tells us if what we are building is right. It also checks if the overall product and process meets the user needs. User satisfaction is the goal. It is a process of ensuring that the SRS is in compliance with the users' needs, conforms to the documents standards of the requirement phase and is an adequate base for the architectural design phase. (David A Cook)

Requirement Verification and Validation techniques:

Some of the most commonly used requirement verification and validation techniques are:

1) Tracing approaches- This approach helps in keeping a check on goals against tasks, features and requirement. It is generally done using a traceability matrix.
2) Prototyping- This technique helps in demonstrating the requirements to the users. Test scenarios are developed which provide broad coverage of all the requirements.
3) Testing- Inventing test cases to check on the requirements is an effective validation technique. It is very useful and essential if functional requirements have test cases associated with them.
4) User manual writing- Writing user manual is one very effective way of checking on the validity of the requirements. It describes the functionality and implementation of the system.

5) Reviews and Inspections- In this technique a group of different people read and analyze the requirements, look for problems, discuss them and agree on an action plan to address these problems.

   Types of requirement reviewing techniques:

   a) Reading a document and signing for endorsement.
   b) Formal Inspections: Very structured and detailed review, defined roles for participants and preparation is needed.

2.5 Requirement Management

In this step requirements are managed and controlled.

The following things are carried out in the Requirement Management Phase:

1) Controlling the baseline requirement.
2) Keeping a track of the changed requirement and maintaining history of the same.
3) Process proposed changes to the requirement.
4) Keeping requirement consistent with the plans and work products.

How does requirement Management help?

1) In keeping track of the budget.
2) The history helps in reusing the requirement.
3) Improves efficiency.

Thus Requirement Management is a very important step of the Requirement Engineering.

2.6 Requirement issues faced by small and medium sized Companies

Small and medium companies are considered “motors” of industrial growth. They are dynamic, innovative and efficient. From software engineering consultant point of view these companies are, demanding customers with a high potential for growth. Problems faced by small scale industry and startups:

1) **Absence of IT department**- Small scale industry (SSI) and startups do not have a separate IT department. The backup strategy is poor and there are no efficient steps adopted to accommodate security failures. Thus they do not have enough infrastructures to support software development.
2) **Cannot afford software available in the market –** Startups and SSI cannot afford the software (like SAP, Oracle apps) available in the market. Also software available in the market is sometimes too big in to fit into the small set up of these industries. Thus buying the readymade software isn’t an effective solution.
3) **Lack of technical know-how and insufficient training–** In SSI and startups employees are overwhelmed with their day to day activities, thus have very little time for strategic issues such as quality and process improvement. Also

their software engineering maturity level is very low. They are not an IT friendly. Thus there is a large need for know-how transfer.

4) **Use of illegal software –** Many SSIs and startups use unlicensed copies of software. Licensed copies ensure fastest recovery time when failure occurs.
5) **Virus Exposure –** Businesses and users fail to implement protection. A survey conducted by the National Cyber Security Alliance revealed that 67% of the respondents did not have up to date antivirus software.

However, the rising number of tools for requirement management and modeling on the market can be taken as an indication that SSIs and startups are moving from ad hoc software development to more systematic practices.

The Software Technology Initiative (STI) is an association of more than thirty companies, consultants and public organizations which offers consulting and variety of seminars, training programs and workshop mainly for Small and Medium Enterprise (SMEs). This study is based on the STI workshop on requirement engineering. Conclusion drawn from the experience with SMEs is

1) SMEs do not subscribe to large co-operation project, but rather to small slices of work.
2) SMEs require well-packaged and mature results being transferred in short periods.
3) Are rather price sensitive.

The STI workshop on requirement engineering draws some important observations and conclusions. The workshop consisted of five sessions. Each session consisted of experience report from the participant, a lesson on requirement engineering topic, a small case study or experiment and discussions. The SMEs participating in the workshop were operating in the wide range of application domain. The workshop worked around the following issues: product and project characteristics, requirement sources and elicitation activities, requirement engineering process and system evolution.

2.7 Product and Practices

The requirement engineering practices differ a lot between the SMEs

1) Type of system: market-driven vs. customer-specific systems.
2) Degree of adaptability: user –configurable vs. vendor-configured system.
3) Type of software development vs. sub-contracting.

According to the workshop a majority of the SMEs develop market driven systems. In a market driven system two strategies are visible:

1) Starting directly with a market concept

2) Generalizing the solution for one initial customer to the needs of other similar customers.

Projects in SMEs are usually triggered by

1) Technology changes (e.g. new operating /window system, new hardware)

2) New features to be integrated to market pressures.

3) Customer specific adaptation.

2.8 Requirements Sources and Elicitation Activities

The major sources for requirements are customers and users, observations of the market and in-house domain expert. In the initial phase interviews with the prototypical customer are conducted and domain experts are involved. In the later stages workshops are held with potential and future customers to elicit further requirements. The workshop also helps the customer gain deep practical insight about the software.

2.9 Requirement Specification, Verification and Validation

Documenting requirements strongly depends upon the type of software development; in-house development or by sub-contracting the project. It also depends upon the degree of adaptability: user-configurable or vendor configured system. The sub-contracting SMEs must have a complete SRS. The SMEs that develop vendor-configured systems usually state the requirements that have particularly customer-specific adaptation. These become part of the legal contract which the customers possess.

In many market-driven systems that are developed in-house (where the developer and the requirements sources are located at the same place) requirements are communicated verbally and not documented. Hence there is absence of the requirement document. Some SMEs use the user manual as an SRS but face problems as user manuals are not detailed enough.

2.10 System Evolution

SMEs that did not produce an SRS faced problems during the testing phase since the first task of the test personnel is to elicit the requirement again in order to create test cases. Thus a large amount of time and effort is spent on testing and rework. During the maintenance phase the new requirement can cause unforeseeable interaction with the already implemented requirement.

3 Business Modeling

In today's competitive and fast world every company no matter small or big, is adopting the latest and fastest technology available to survive. The companies that fail to update their manufacturing, management and financial services will fail to succeed in the long run.

The main aim of Small scale and start up business is:

1) Increasing the number of clients/customer- As a start up the main aim of the company is increasing the number of clients. This comes by maximum customer satisfaction and by producing quality products. Meeting the customer needs by customizing products according to their specification and needs. This also helps increase reputation and good will among customers.
2) Making and maintaining reputation- Building reputation and trust takes years. Proper care needs to be taken right from the very first project. Even a smallest mistake can ruin all the efforts. Hence companies need to take care of product quality, customer satisfaction, employee satisfaction, develop and maintain good management processes.
3) Increasing profits- Increasing profits is the aim of all companies but particularly startups and small scale industries because that improves their chances in the bigger market.
4) Improving infrastructure- Good infrastructure plays a very important role in the success of the company.

5) Globalization- With increasing globalization even the smallest of the companies have the opportunity to go global. Hence a business should be capable to meet the global needs and should be able to acquire the required international industry standards.
6) Quality Management – Maintaining the quality of one's product is an essential part of any business.

3.1 Purpose of Business Modeling

Business modeling helps in deciding what kind of a business system is required and what place this system should occupy in the company structure. Business modeling is an abstract form of presenting business functions. To extract value from an innovation a start-up needs an appropriate business model.

The business model gives the software developer a better idea of the business process and makes it easier to develop a system. A business model is a simplified view of the business.

The main aim of business modeling is:

- To understand the structure and dynamics of the company.
- To ensure that the customers, end-user and developers possess a common understanding of the company.
- To make it easy to follow the requirement engineering steps and gather requirements.

- To understand how to deploy new systems to facilitate productivity and which existing system may be affected by that new system.

In the context of the information system the model helps answer the following questions:

Why build the information system?

Where will it be located?

Who all are the prime users and how do they interact?

How will the functionality of the system help reduce human errors?

How will the information system reduce the efficiency of the business?

In short, business modeling gives the team a logical approach to understand and define where a software application can improve the productivity of the business and help determine requirements for those applications.

Business modeling does have its limitations and is not complete and accurate in all the scenarios. For example,

1) The changes in a company environment can alter the basis on which the model was created.
2) Very complex model can make the model as complex as the business and thereby hard to understand.
3) Stakeholders might resist overly complex models due to their intricacy.

4) Not every detail can be fixed during the modeling process because each method / technique has its evident restrictions.

Modeling of complex business requires multiple views to focus on particular aspects of the business. Errikson and Penker argue that there are four different views of a business:

The business vision: An overall vision of the business. It describes the goal structure for the company and illustrates problem that must be solved in order to reach these goals.

The business process: the view that represents the activities and values created in the business and illustrates the interaction between processes and resources.

The business structure: the way the resources in the business are structured, such as organization of the business or the structure of the products created.

The business behavior: the individual behavior of each key resource and process in the business model.

Amongst all the above mentioned views the business process describes the process in the business along with goal, resources and activities in the business.

3.2 Business Process Model

The term business has many meanings. The Amos WEB economics on-line dictionary defines business, as "a profit-motivated organization that combines resources for the production and supply of goods and services. The term business usually refers to a productive organization that is privately owned and motivated by the pursuit of profit”.

The second source, Merriam Webster’s online collegiate dictionary defines a business as being “

a) Usually commercial or mercantile activity engaged in as means of livelihood;

b) A commercial or sometimes an industrial enterprise;

c) Usually economic dealings”

The meaning of business in the context of this paper is: an organization profit making, not for profit, health care etc. that uses resources (human, financial, information) to produce services or products. In the context of this thesis we will stick with the above meaning as this will help in better understanding of the requirements of the business from the software point of view.

There are a myriad of definitions for the word “process”. Given below are the ones which fit properly in the context of this thesis:

- Collin considers a process to be a “sequence of activities performed on one or more inputs to deliver an output to the customer of the process”.
- Whereas Davenport extends the previous meaning to encompass “a specific ordering of work activities across time and space, with a beginning and an end, and clearly identified inputs and outputs: a structure for action”.

- Ould defines process according to its key features “1) It contains a purposeful activity. 2) It is carried out collaboratively by a group.3) It often crosses functional boundaries.4) It is invariably driven by the outside world”.

Every organization has an aim. It may be to manufacture equipment, to provide health care, to build and sell cars etc. To achieve its aim the organization has to take care of a number of things like finance, sales, marketing, billing, human resource etc. Many organizations have different departments to take care of each of the above mentioned activity. Each department has its own purpose and responsibilities. All these departments work in co-ordination to achieve the main aim of the organization. This ‘work in co-ordination’ forms a process.

Finally this paper assumes a business process to be a sequence of business activities performed by the people that exist within a company, use different resources as their inputs and produce an observable end-output and are driven by an external event or the company’s internal environment.

Business process model is about how a business carries out its process in the most efficient manner and how it supports the staff to achieve this. It is about designing an IT system to support the people rather than to have people do what the system tells them to do.

3.3 Building a Business Process Model

To build an effective business model one needs to understand all the processes in the business. The following questions help to understand the business from business modeling point of view.

- How does the process contribute to the high level goal?
- The physical steps involved
- The sequence in which steps should be undertaken. The skills required by the people who undertake the work
- What decisions do people involved in that process have to take?
- The organizational structure
- How do the customers and the staff communicate and interact (e.g. how and what information is requested and provided)?
- Why documentation is produced and where it goes?
- What kind of documentation is used? (Training manuals, user guides, checklists, references etc.)
- The information required by people to do their jobs. People having restricted access versus people having access to everything. How does each process link internally and externally?
- The part of the processes that are supported by the IT systems.

3.4 Use of Business Process Model

Business Process model is useful in understanding the organization as follows:

- Understand and document what the organization is doing currently.
- Understand and document the future goals of the organization.
- Identify where the IT system supports the organization and where does it fit in well. Provide a starting point for the design of such systems.
- Become the basis for training staff.

3.5 Business Process Modeling Technique

Based on industrial experience it can be said that without models that are shared by the stakeholders, business is a hard activity to understand and manage. These models must reflect the integrated, dynamics and the distributed nature of the business. There exists a number of Business Process modeling techniques. Choosing the correct technique can be a challenge.

4 Introduction to Business Process Modeling Technique

Every business is unique and has its own way of being carried out. Hence there are a variety of modeling techniques available. All these methods and tools have something in common: they attempt to solve critical problems. These modeling techniques help all the users from business managers, workflow specialists, software developers; participate in the development of software solutions that support business process.

It is necessary for all the parties involved to understand the essence of the business process model through communication that spans the entire lifecycle of the project. Possessing a common language, understanding the structure and behavior of the process, formulating the requirement in an unambiguous manner, mapping the business requirement to the software component, interpreting the business rules correctly and presenting the software solution to the users of the future system are all very important factors which determine the success or failure of a project.

There are basically three approaches for business process modeling: process - oriented, data-oriented and object-oriented approaches. Object oriented is the latest adopted methodology. In this thesis we will study the object oriented approach. Object oriented approach combines data and process (methods) into single entities called objects. Objects easily relate to things that a business oriented software system deals with such as customers, suppliers, contracts etc. Object oriented model is able to completely represent a complex model. The goal of the object oriented approach is to make system elements reusable, thus improving system quality and the productivity of system analysis and design.

4.1 How does object oriented mechanism work?

Objects, encapsulation, inheritance and polymorphism are the base of the object oriented mechanism. To understand and express the essential and interesting features of an application in the complex world, an object oriented model is built around the objects. An object encapsulates both data and behavior.

By a concept called polymorphism, functionality that is conceptually similar among differing objects is extracted to a global level. This process limits the production of parallel functionality and streamlines the information interface. Polymorphism directs the specification writer to understand the functionality of a process and make it available to any object that requires a similar instance of functionality (Cackowski 2000).

4.2 Unified Modeling Language

The Unified Modeling Language (UML) is an object-oriented language for specifying, visualizing, constructing, and documenting the artifacts of the software system. UML uses simple and intuitive notations which non-programmers can easily understand. Developers, customers and implementers can understand a UML diagram. The UML diagrams help them to understand and agree on the intended functionality, thereby improving their chance of creating an application that truly solves a business problem.

The UML notations are useful for graphically depicting object oriented analysis and design models. These notations allow one to specify the requirement of the process and capture the design decisions. They also help in promoting communication among key persons involved in the development process.

Object oriented approach can be classified into three categories (Monarch & Puhr, 1992) 1) Combinative approaches use different modeling techniques in different stages of the system development process; 2) Adaptive approaches apply existing techniques (e.g. data flow diagram and entity relationship approach) in object oriented ways to analyze the problem domain 3) Pure approaches adopt an object oriented perspective in systems analysis and design.

4.3 Conceptual Modeling

The goal of conceptual modeling is to develop the conceptual model of the problem domain – the structural model of the UML analysis model, which defines key abstractions (that is a concept), as well as their properties and relationships. It defines the vocabulary of the problem domain because it specifies the meaning of the concept and terms from the problem domain.

The process of conceptual modeling is very challenging, difficult and intensive. This stage comes after the requirement elicitation step in the requirement engineering process. After carefully listening to the user by whatever elicitation technique is applied, the software developer should try to identify the concepts and their relationships with the problem domain. Users typically present key abstractions from the problem domain in two ways:

- Explicitly by trying to introduce and define terms and concepts from the domain.
- Implicitly by describing use cases, that is to describe desired functionalities of the system and scenarios of its behavior in use.

Once the concepts and their relationships are identified the developer can design the UML conceptual model. The model is a rough draft of the users' requirements also called the candidate model. This candidate model should satisfy the users' requirement. The developer should be able to interpret the scenarios just like the future system would do and see whether the candidate model can satisfy the use cases and provide information desired by the users.

The developer should also have a vision and be creative enough to discover the use cases that can be expected by the application. The candidate conceptual model should also be robust enough to satisfy all the unexpected requirements that come later in the project.

Users' requirements are often not very clear and sometimes ambiguous. To address this ambiguity the developer should first identify all possible and acceptable alternatives of the users' statements during the elicitation process. To select the right alternative the system analyst should carefully select the scenarios of behaviors of the system that qualify one and disqualify the other alternatives. Thus the process of conceptual modeling is iterative and incremental. The developers iteratively improve the understanding of the problem domain while building the conceptual model. It is built by extending, modifying and refining the initial understanding and idea about the problem domain. Thus when the developers gain a good understanding of the problem domain and obtain a conceptual model they are ready to describe it in the textual requirement specification document.

4.3.1 Guidelines for building a good conceptual model

- Listen carefully to the user.
- Identify abstractions, generalize things, and invent formal concepts.
- Check abstractions and candidate model against the required use cases.
- Invent other expected use cases to check your candidate model.
- Propose generalized concepts and discuss them with the users. Explain the concepts to the user by using domain-specific use cases.

If faced with ambiguities, discover alternatives. To select the right alternative, invent scenarios of behavior that qualify one and disqualify the others. Describe the scenario to the user and select an alternative. If several alternatives are acceptable, select the simplest for implementation or the most flexible one.

5 Case Study on Syscon Engineers

5.1 Introduction

Syscon Engineers is a mechanical manufacturing company in India. It manufactures mechanical equipment like heat exchangers, reactors, pressure vessels and stack liners. It also takes up turnkey projects and site work. It is a small company and was established in 1995.It comes under the small scale industry category according to Indian standards. It has seen a steady growth since 1995. With increasing growth there came a need for more organization and better management.

The stakeholders started feeling the need for a system which would take care of their purchase, sales, work orders and work status. Accounts were taken care by software called Tally which works on the similar line of Quick Books. Hence the mission of developing software was undertaken. I was the only software engineer in the company and hence took up the responsibility. I got one more person to work with me. Together we built in-house software called the Syscon Management System (SMS).

5.2 Challenges faced during the development of the SMS

We tried to follow the traditional method of software development while developing SMS. We faced the maximum challenges during the requirement gathering phase. Following are the challenges that we faced:

1) The company being a manufacturing unit did not have proper infrastructure for an IT project.

2) It was difficult to convince the users as to how the software was going to reduce their work load and improve their efficiency.

3) Users were so overwhelmed by their day to day activities that they didn't have time to think of requirements for a software project. Hence getting the users to talk was a big challenge.

4) Cost was another big concern. We had to find out many work around to reduce the increasing cost and keep the budget in limit.

5.3 Method adopted

Throughout the software development we adopted the prototyping method of requirement elicitation. We analyzed each requirement, discussed it with the users and demonstrated it by implementing partially. We maintained an SRS throughout the development phase.

5.4 Shortcomings in the requirement method adopted

As I studied the requirement engineering process from small scale industries point of view I realized a number of shortcomings in the method we had adopted for our project.

1) Though prototyping is a good method for requirement elicitation we should have considered another method which allowed more user involvement.

2) Our SRS was not up to the mark. It did not follow a consistent pattern throughout the development process. Also it failed to take care of all the nine key elements considered for writing a good SRS.
3) Apart from prototyping we should have also considered writing a user manual to validate the requirements.
4) Since we were developing a management system which took care of the company's sales, purchase and work status. we had to understand the company's business process thoroughly. We did that by conducting several meetings with the stakeholders and users. Hence considering the above mentioned object oriented technique and conceptual modeling technique could have helped us improve our understanding of the business and its software requirement in a better way. This would also provide a better understanding of the software to the user.

6 Conclusion

This thesis studies the requirement engineering process for small scale industries. The requirement engineering process consists of the requirement elicitation, requirement analysis, requirement verification, requirement validation and requirement management steps. Requirement elicitation and requirement management play a key role in the requirement engineering process.

The case study presented in the thesis shows that small scale industries face a major problems during the requirement elicitation phase. This is because the users are not very interactive, they are overwhelmed by their day to day activities and hence do not have enough time to participate in the requirement gathering process. Also they are not technically sophisticated and hence fail to understand the use of the software.

This thesis has considered business process modeling as the basis for requirement elicitation at small scale industries. Business process modeling using use cases helps in better understanding of the business structure. Business process modeling also helps in identifying where software will fit in the organization. It also helps in training the staff after the implementation of the software.

Conceptual modeling using the Object Oriented technique helps in better understanding of the software requirements. It helps in identifying abstractions and generalizing requirements.

Thus from the above study we can conclude that business process modeling followed by good conceptual modeling will allow small scale industries to greatly increase the business impact of their investments in informational technology projects.

7 References

1) Dragan, Milicev. Model Driven Development with executable UML. Indianapolis, IN: Wrox/Wiley, c2009.
2) Wiegers, Karl E. Software Requirements. Sebastopol: O'Reilly Media, Inc, 2009.
3) A.Aurum, C.Wohlin. Engineering & Managing Software Requirements.Berlin, London: Springer, 2005.
4) Leffingwell, Dean. Managing Software Requirements: Use Case Approach-2003. Boston: Addison-Wesley, c2003.
5) Yana Selioukova. Business Process Modeling in Software Requirement Engineering. Spring 2002.
6) Peter Oriogun. Innovations in Teaching And Learning in Information and Computer Sciences. University of North London,UK.
7) Erik Kamsties, Klaus Hormann, Maud Schlich. Requirement Engineering in Small and Medium Enterprises: State of the Practice, Problems, Solutions and Technology transfer. Fraunhofer Institute of Experimental Software Engineering, Germany
8) Roland Kaschek, Heinrich C Mayer. Characteristics of Object Oriented Modeling Methods. University of Klagenfurt.

9) Ilia Bider. Choosing Approach To Business Process Modeling-Practical Perspective. Research Report, Ibis soft, 2002.

10) Linda Westfall. Software Requirements Engineering; What, Why, Who, When, & How.

11) Jain Wang. Object – Oriented Analysis Methodology. University Of Missouri, St.Louis, Fall 2001.

12) Robin .F. Goldsmith. Software Requirements Management .Search Software Quality.com, E-Guide.

13) Erick Eckel.10 mistakes Small businesses make.

8 Vita

Teema Chandrakant Benadikar has done her bachelor of engineering in Computer Science from Mumbai University, India. She has a work experience of two years in IT industry. She then entered the Graduate School at The University of Texas at Austin.

Permanent Address: 12440, Alameda Trace Circle, Apt: 1236,
Austin, Texas 78727.
e-mail address: teemabenadikar@gmail.com