Deducing Requirements From Agile Software Processes

by

Ponle Salu

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Applied Science

in

Management Sciences

Waterloo, Ontario, Canada, 2014

AUTHOR'S DECLARATION

Abstract

In classical engineering practice, the elicitation of requirements is an important early project phase. Requirements help to define the project goals and scope, they serve as a basis for cost estimation, and in validated projects they are the cornerstone of the traceability matrix. However, requirements elicitation is difficult because of the abstract nature of the process and because there is uncertainty at the start of a project about what can be done.

In recent software development practice, waterfall methods have fallen into disfavor, and agile methods are preferred. Agile methods avoid formal requirements specification, and instead use techniques such as scrums and user stories to specify development phases that are performed iteratively. In agile methods, requirements remain implicit and undocumented.

While agile may avoid the difficulties of formal elicitation of requirements, it may in the process bypass the activity of analysis of user needs, and the generation of a baseline against which the implemented system can be validated.

In this thesis we show that requirements can be deduced from the user stories and process maps that result from agile methodologies. A modified failure mode effects analysis approach is used to identify risks, failure modes, and countermeasures, and to evaluate risks and countermeasures by computing severity and likelihood of the risks, and the benefits of the countermeasures.

The deduction of requirements from agile artifacts encourages an agile team to think through its preferences and proposed implementations, and objectively rate them. It captures the rationale for the user stories and process maps, and provides traceability from business goals to the functional requirements.

Acknowledgements

First, I want to thank God Almighty, who in His infinite mercies has made this dream a reality.

My deepest appreciation goes to my thesis supervisor, Dr. Darrell Raymond whose depth of knowledge and subject matter expertise proved instrumental to the success of this research endeavor. Without his continuous encouragement, enviable spirit of adventure, countless hours of brainstorming, convincing arguments, exemplary work ethic and selflessness, this research work would not have been possible. You will always be a mentor.

I will be forever grateful to my co-supervisor, Professor Frank Safayeni, Chairman Department of Management Science at University of Waterloo for ensuring I get all the administrative, technical and scholarly support I needed to succeed in this undertaking.

I would also like to acknowledge Dr. Mark Hancock and Dr. Stanko Dimitrov for reading this thesis. I am gratefully indebted to both of you for your time and valuable comments.

The case study used in this thesis was conducted as a collaborative industry-university research project. I would like to thank everyone that played a role in ensuring the objectives of the project were successfully realized. Special thanks to Steve Mai and Todd Ronald (industry partners) for the opportunities and supports availed us throughout the project; to MITACS for funding the project through the MITACS Accelerate research internship program; Jennifer Tedman-Jones of MITACS for supporting the project in every way possible; Ada Hurst (lecturer, Industry Liaison) for raising the possibility of a project in the first place; Geovania Pimenta (fellow intern) whose project management skills gave the project the impetus to succeed; Rob Duimering (lecturer, project co-supervisor) for all his efforts and the support he gave to see the project to success.

Special gratitude is extended to Dr. David Fuller (Associate Chair, Graduate Studies) for throwing his weight behind me in pursuing this research endeavor.

I would like to extend my sincere thanks to the management, lecturers and staff of the Management Sciences department of The University of Waterloo for the high level of support, dedication and professionalism. Studying here has been highly rewarding, enlightening and nourishing.

Many thanks to Bright Iheanacho’s family for their support upon my arrival in Canada, the Akinniyi family for their big heart, Marcia Halliday (landlady) for being hospitable, members of Our God Hears Foursquare Christian fellowship, My Dad, Prophet Olatunji Gbadebo Salu for relentlessly encouraging me to do this, the families of Deji Olusoga, Gbenga Salu ,Kemi Dairo, Adelola Adesida, Akinboro, my larger family members, in-laws and friends.

Last but not the least; I want to profoundly thank my wife, Abimbola Salu and my son, Timi Salu. Your overwhelming show of real support and understanding is thorough and peerless. We envisioned this together, painstakingly went through the stages together and now, we can see it come about together. You are awesome!

Dedication

I dedicate this thesis work to my mother, Victoria Iyabo Salu who passed on to glory while I was far away from home doing this program. Your words of advice to serve God, serve my fellow human beings and be the best I can be continues to ring in my ears.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 The importance of requirements

According to the Institute of Electrical and Electronics Engineers, a *requirement* is a condition or capability needed by a user to solve a problem or achieve an objective. It is also stated to be a condition or capability that must be met or possessed by a system or system component in order to satisfy a contract, standard, specification, or other formally imposed document (IEEE, 1990). Simply put, requirements are the necessary behaviors a system must exhibit to fulfill desired objectives.

It is a truism of software design that many software projects fail because their requirements are poorly understood or poorly managed (Dorsey, 2000). A system's requirements are considered important for the following reasons:

- They are a key step in evaluating and defining the scope of the project and in prioritizing user needs and desires (Karlsson & Ryan, 1997)
- They capture the needs of both the users of the system and the constraints of various other stakeholders, such as the IT support group, and explain why those needs and constraints should be in place (Nuseibeh & Easterbrook, 2000)
- They provide a description of what the software system should do without specifying how it should do it. They can thus serve as a checklist against which various vendor software or candidate designs can be compared (Westfall, 2006b)
- They provide a baseline for software validation, which tests the question "did you build the right system?"[1] (Magsarjav, 2004)

Failure to capture requirements adequately can lead to the following problems:

- If requirements are missing, then important needs may have not been addressed in the design, and so the system fails in use because it does not meet those needs (K. E. Wiegers, 2009)

[1] As opposed to most kinds of software testing, which test "did you build the system right?"

- If requirements are missing or are not specified, then the prioritization of various system features may not have been well done, and so effort will have been expended on features that are of less importance than those which are missing from the system (Lehtola, Kauppinen, & Kujala, 2004)
- If requirements are missing then the scope of the project is not fully understood, and so planning will not be adequate. There may be a need for extensive rework when requirements are discovered during design, development, testing, or rollout, and so the project exceeds its timeline or budget. Conversely, if requirements are overstated, then the project will have a timeline or budget that is excessive compared to what could have been done if requirements were better understood (Heindl & Biffl, 2005)
- If requirements are not stated as "what" and instead as "how" (that is, if implementation is provided instead of requirements) then the problem solution space is artificially restricted, and the resulting system will not make the best use of the possible solutions (Firesmith, 2007)
- If requirements are not well stated, then it will be difficult to validate the system. If requirements are lacking, then validation of the system will lead to false confidence that the system was the right one to build (Firesmith, 2007)

Since these problems can cause substantial rework, it has always been considered important to do an effective job in capturing requirements, to minimize the problems that occur in downstream phases of a project. For this reason, the classical "waterfall[2]" method of software development puts the requirements phase at the very beginning of the process. The waterfall method is still very common in regulated industries, such as aerospace and medical device development, and the importance of requirements is such that the process is not allowed to proceed until requirements have been signed off by all stakeholders.

1.2 Problems eliciting requirements

Requirements elicitation and specification is the task of understanding needed behaviors and determining the implementations needed to achieve them. However, the task of eliciting requirements is usually not easy, the information needed to formulate solutions is rarely available in explicit form,

[2] The waterfall (Royce, 1987) method proposes a linear, sequential approach to software development consisting of five phases – analysis, design, coding, testing, and maintenance

and information is often distributed across multiple sources, some of which could be conflicting. Goldsmith states that the elicitation task is “exceedingly difficult” (Goldsmith, 2004). Maynard-Zhang et al. state that requirements engineering and especially, early-phase designs, are inherently uncertain (Maynard-Zhang, Kiper, & Feather, 2005).

The term "elicitation" is preferred to "capture", to avoid the implication that requirements are out there to be collected simply by asking the right questions—instead they must be elicited from the users (Jirotka et al. 1994). The process of eliciting requirements spans both problem and solution domains. Eliciting requirements in the problem domain involves learning, extracting and determining as precisely as possible the problems that are or could be faced, the context within which that problem exists, and any rules that will constrain the essential features of solutions to the problem. In the solution domain, elicitation focuses on the formulation of methods to transform a potential or existing problem into desirable outcomes. Feng and Eyster assert that the greatest impact of a system development process occurs during the requirements elicitation and concept formation stages (Feng & Eyster, 2013). According to Rechtin, this process requires a great amount of creativity, but since creativity is one of the least understood of human activities, we are at some difficulty to explain the requirements process (Rechtin, 1991).

The need to build, change, correct or extend a system is usually as a result of some overarching business objectives such as the need to comply with regulations, the need to develop a new product, or the need to re-engineer a business process. These goals are usually broadly and vaguely expressed. The goals are sometimes not detailed enough to be implemented by a developer, nor specific enough to be verified by a tester. They may also not be sufficient for cost estimation (Herrmann & Paech, 2007).

The following problems are common in requirements elicitation (Christel & Kang, 1992), (Sommerville, 2004); (Avison & Fitzgerald, 2006):

- Users are not familiar with the requirements process and have difficulty thinking of needs in the abstract. Frequently users will want to specify an implementation (e.g. “we need a folder for work-in-process that has permissions set for only the editors”) instead of specifying only what the system needs to do (e.g. “work in process must be visible only to editors”)

- Prioritization of needs is not often based on empirical data or well-defined costs, and instead is an exercise in voting for “favorite” capabilities. Users may overestimate the actual cost saving of features they personally like, or spend too much time thinking about user interfaces compared to underlying functionality
- Users are not trained in what features are available in software, so they cannot judge what features are easily provided or which ones will require significant customization or configuration (with additional future support issues when software is upgraded)
- Participants in the requirements process may not know what they want until they actually see an implementation
- The output of the requirements process is a document or entries in a requirements management system, stated in the form “The system shall do X” and “The system should do Y”, and hence is abstract rather than practical. Participants in the requirements process can easily have difficulty visualizing whether the resulting system is really the one they want
- The process of collecting requirements involves interviews, meetings, and formal descriptions. Many participants find these activities to be tiresome, conflict-prone, and do not build confidence in the final result
- Requirements are sometimes elicited by technicians who either have little training in the subject, or who do not understand the full purpose of requirements for system maintenance and future upgrades, and so the elicited requirements are not as comprehensive as they should be

1.3 Agile methods

Partly because of the problems typical in elicitation of requirements, there has been growing interest in software development methods that are generally known as *agile*. Agile encompasses a large variety of techniques, but they share the same general notions (Hazzan & Dubinsky, 2009):

- Systems should be built in increments known as iterations, which are short (less than a month), result in working software (even if it performs only a very few tasks) and which build on one another

- Working software is more important than documentation
- Development should be test-driven; that is, tests should be built before the software itself is developed
- Development teams should be comprised of a mix of developers, users, and other stakeholders, who meet regularly to evaluate the current iteration and to decide on the content of each new iteration
- The system is done when the team decides that it is done
- Change is permitted and even encouraged during each iteration

In agile methods, there is no "requirements" phase to the project, nor is there a "requirements document". The closest one comes to the notion of requirements is that of *story*; a story is an explanation of how a specific type of process should work in the resulting system. When an iteration fulfills its stories and passes its tests, then it has in effect met its "requirements".

Agile attempts to avoid some of the problems that are known to occur in requirements elicitation (Kajko-Mattsson, 2008), (Daniel Turk, Robert, & Rumpe, 2005):

- By avoiding a requirements process, agile sidesteps the unfamiliarity of users with that process
- By having users respond directly to an iteration, agile makes it possible to obtain quick feedback and to refine users' desires, to discuss implementations and needs at the same time, to avoid having to think about an abstract statement of needs, and to learn just-in-time what software capabilities can be easily provided
- By avoiding the desire to predict the future and only evaluate what is in front of them, users and developers have a simpler task
- By managing implementation as a series of iterations, agile inherently prioritizes needs according to what is understood and what can be implemented at each stage, and reduces the tendency of users to request very elaborate systems or user interfaces
- By working on iterations immediately, the team does not feel like it is delaying the project with weeks of analysis and documentation

1.4 The thesis premises

It is clear that the points mentioned in the previous section are attractive aspects of the agile process, and if the agile process leads to higher quality software than other methods, then there is a strong argument to follow the process. But though we accept the proposition that agile methods do not require that a development process should *begin* with a formal statement of requirements, then it does not necessarily follow that a development process should not have as one of its *results* a formal statement of requirements.

It is a premise of this thesis that requirements are important for more than just the design of a software artifact: they are a formalism that has value in auditing, justifying, maintaining and evolution of software. Scrums and user stories have value in team communications, but they do not themselves fully capture the decision process in an auditable manner, nor do they require that the team consider software architecture, maintenance, installation, update, operation, or conformance to regulations[3].

It is the premise of this thesis that requirements remain important, even if agile methods are used to develop software, and that therefore we need to find a middle ground between up-front formal statements of requirements, as practiced in the "waterfall" method, and the no-requirements informal approach of agile methods. Other attempts have been made to find such a middle; in particular, the regulated medical device industry has tried various ways to practice agile software development while still meeting Food and Drug Administration (FDA) regulations for validated software development (Dean Leffingwell, 2011).

In this thesis we propose a method called Failure Mode Based Requirement Elicitation Method (FBREM), which can be used to deduce requirements from systems that have been developed through an agile process.

- FBREM can be applied after or during the agile process, depending on when a team sees the need for more formal analysis

[3] Nothing *stops* these issues from being considered in an agile process; the point is that nothing about the agile process *require*s them to be considered, and the output of an agile process is not easily audited to ensure that those considerations were taken into account.

- FBREM provides teams with an objective method for prioritization of requirements based on business goals and estimates of risk, which is better than leaving the prioritization to team guesses or development constraints
- FBREM structures requirements so that the rationale of any particular software feature can be traced back through levels of requirements to the business goal
- FBREM provides traceability between various levels of requirements and software features, which is useful when considering changes to the software or re-evaluating design decisions

In short, FBREM preserves the benefits of an agile development process, while still resulting in a formal requirements specification that is well-structured and based on business goals.

1.5 Contributions of the thesis

The contributions of this thesis are as follows.

1. We show how formal statements of requirements can be deduced from artifacts such as user stories and process maps that result from agile methodologies
2. We show that risk is a useful basis from which to deduce requirements. Empirically we observe the sensitivity of an agile team to its perceived risks; we then extend this observation to the idea that many, if not most, requirements are a response to some kind of risk
3. We show that requirements can be structured in levels, depending on the specificity of the countermeasure
4. We show that an objective prioritization of requirements is possible, based on countermeasure priority numbers
5. We show how FBREM structures requirements so that the rationale of any particular software feature can be traced back through levels of requirements to the business goal

   We show that FBREM provides traceability between various levels of requirements and software features, which is essential in software validation, and important when considering changes to the software or re-evaluating design decisions

1.6 Organization of the thesis

The thesis is organized as follows.

Chapter 2 presents a case study of the engineering automation company MACE, who automated their engineering processes in 2013—2014. This case study includes a project (in which the author worked) that employed a variant of the agile approach in the development of the system supporting process automation. A requirements document was not produced, but the project did create a working software system for managing the automated process.

Chapter 3 presents a method to elicit requirements from the working software developed through an agile process as described in Chapter 2. The method is based on failure mode and effects analysis, applied iteratively and intended to develop requirements. We call this method FBREM.

Chapter 4 discusses the FBREM method and its advantages in three areas: requirements prioritization, rationale, and traceability. Chapter 4 also compares and contrasts FBREM with other methods for obtaining prioritization, rationale, and traceability.

Chapter 5 contains our conclusions and suggestions for further work.

A full example of the FBREM method for one MACE process is found in Appendix A.

Chapter 2
MACE Case Study

2.1 Introduction

This chapter describes an agile development project which we conducted as part of this thesis. We developed a business process map using a software tool; the process map detailed the business activities of a medium-sized engineer-to-order firm. The agile development approach was used to conduct the business process-mapping project. This case study raised interesting observations about the benefits and defects of agile software development, and led us to the thesis contribution: the deduction of requirements from agile software development artifacts.

The chapter is organized as follows:

Section 2.2 provides a background of the engineer-to-order firm and the process-mapping project

Section 2.3 outlines the methodology used in conducting the project and discuses the activities and processes undertaken at various stages of the project

Section 2.4 presents a highlight of the methods used in conducting the process-mapping project and how they relate to various agile principles

Section 2.5 assesses the goodness of the agile approach as applied in our case study, bringing out the benefits of the agile approach as well as its drawbacks.

2.2 Background of case study

This case study was conducted at a company that supplies custom automated manufacturing and testing equipment solutions for diverse manufacturing needs in a variety of industry sectors, including health sciences, transportation, mining, telecommunications and energy. For reasons of confidentiality, we will refer to this company as MACE. MACE's services include the complete development of equipment, mechantronics engineering, management information systems, and product deployment and installation.

With a workforce of over 150 and with capabilities in applications development, project management, mechanical engineering, controls (hardware and software), fabrication, paint and sand blasting and production (tools, assembly and electrical), MACE offers a complete suite of custom automation service including pre-automation services, project management and post-installation support services such as training, spare parts management, process optimization, and long term service agreements.

MACE's tailored engineer-to-order process begins with a sales lead or request for quotation. An engineering solution that meets the customer's expectations is then proffered in a quotation; if the customer issues a purchase order, the project will be planned and the equipment designed. The project then proceeds through the manufacturing and assembly, integration and acceptance, tear-down and ship phases until the equipment installation is finalized a customer's plant and other project close-out activities are conducted to conclude the order.

The case (unit of analysis) in our study is a business process mapping project. In order to improve operational effectiveness and support business growth, MACE developed a process blueprint that maps their major process steps and workflow involved in the engineer-to-order business operation. The case study was a funded effort to build on the existing process map by describing business processes in greater detail, and by identifying a flexible, user-friendly software tool to implement the process model. As part of this case study we documented the personal knowledge of work processes held by individual employees and managers, so that this knowledge could be incorporated into the revised business process.

2.3 Methodology

The empirical data for this case study was collected in semi-structured, open-ended interviews that were conducted by a team of two (2) researchers[4] within a 3-month period at the research site.

Interviews were conducted with eight (8) members (including managers) from the Sales and Applications department, and five (5) managers from other departments that play a role in the Sales and Quotation phase. In total, 24 interviews were conducted, with several managers being interviewed two to four times.

The interviews were audio-recorded and subsequently transcribed. Handwritten notes were also taken during the interview. The interview data were supplemented with company documents such as training manuals, quotation templates and sample quotation documents.

[4] The researchers were the thesis author and Geovania Pimenta.

We next outline in detail the processes and activities undertaken to obtain the business process model.

2.3.1 Project preparation stage

The preparatory stage of the project involved meeting with the top management to understand the business needs, goals and objectives of the process-mapping project. The company president gave an overview of the company's business and conducted a walk-through of a preliminary model of the company's business model. This gave us a baseline understanding of the business operations and familiarized us with the structure of the company. We also met briefly with key heads of units and visited the manufacturing facility to have a first-hand look at some of the manufacturing activities. The output of the preparation stage was a deeper understanding of the case study scenario, and an understanding of what the company does and how activities are performed at the macro level. Interactions at the stage introduced us to some of the key individuals in the company. We also obtained and studied existing documentation such as the company's organizational chart that showed us in a graphical format the company's chain of authority and names and roles of staff members. Other documents obtained include training manuals, quotation templates and sample quotation documents. The following project deliverables were required at the end of the project:

1. The process map of selected processes
2. A matrix outlining pros and cons of potential software solutions
3. A software implementation of the process map, using one of the solutions identified in the matrix

Since one of the major deliverables of the project was identifying potential process modelling software that the company could adopt and eventually use to execute the company's process, we began testing, screening and evaluating Business Process Management (BPM) tools at this stage. BPM tools are software applications that can be used to diagram and execute business process flows. These tools usually come with modeling interfaces intended for non-programmers, so that they can be involved in capturing relevant information about the processes. Some BPM tools support both the design and digitization of business process, so that the component of the business can be easily identified and adapted to the ever-changing business requirements. BPM tools support rapid prototyping and experimentation and would be an essential part of an agile team's approach to this kind of software problem.

The first step in the tool selection process was selecting a modeling notation standard for representing the model, since the chosen notation would determine the set of tools that can be considered. Among the available notations we identified were the following:

- XML Process Definition Language (XPDL)
- Business Process Execution Language (BPEL)
- Event-driven process chain (EPC)
- Unified Modeling Language Activity Diagrams
- Business Process Model and Notation (BPMN)

BPMN, which is maintained by the Object Management Group[5] (OMG) was selected, largely because of its niche in visual expressiveness and richness of language system set compared to the other notations (BIS, 2010).

Due to the large number of BPMN modeling tools available in the market, and the limited time available to conduct the tool evaluation, we introduced screening criteria such as cost of acquiring the evaluation copy of the tool, support for the industry-standard Business Process Model and Notation (BPMN) and easiness to set-up and configure, to reduce the number of tools to be considered. Eight (8) tools were eventually evaluated against the following quality criteria:

- Compliance to the BPMN notation standards and notation rules enforcement
- Installability – the system requirements to run the tool in the company system environment
- Interoperability – ability of the tool to integrate with existing infrastructure and file formats
- Learnability – availability of learning materials and ease of mastery of the tool
- Maturity – inclusion and rating in major market reports, licensing cost and vendor support

The table showing the tool screening and table showing the tool evaluation is available in Appendix E and Appendix F respectively.

[5]OMG is an international, open membership, not-for-profit computer industry standards consortium

Bizagi Process Modeler was eventually selected as the tool for modelling and documenting the MACE business process. Bizagi Process Modeler version 2.6 was used in this project.

2.3.2 Model development – Iteration 1

The Sales and Quotation phase of MACE's business process was selected by management as the process to be modeled first. The Sales and Quotation phase begins with either the identification of an informal sales lead, or the receipt of a formal Request For Quotation (RFQ) from a potential customer. The Sales and Quotation phase ends when the customer's issued Purchase Order (PO) is accepted by Sales, or when MACE decides not to bid the job.

Interviews were conducted to learn about the Sales and Quotation business process, the relationships between workers, the flow of activities and the documentation, as well as information systems involved in the process. Initial interviews captured the major steps and overall workflow of the Sale and Quotation phase, while later interviews focused more narrowly on specific steps and activity details, in order to validate earlier results and to address any remaining gaps in the emerging business process model.

During the first round of interviews, the managers and engineers from Sales and Applications department were asked to sketch the processes they participated in, to describe each process in detail, and to identify any database tools and documents used while performing each process. This made the modelling process participative, and often the first iteration of a sketch they drew was used as a thinking model upon which they reflected, discussed further, and then modified to something they considered better suited. A hand-drawn sketch of the process by one of the interviewees is shown in Figure 1, while Figure 2 is a computer reproduced version of sketches produced by two interviewees of the same process.

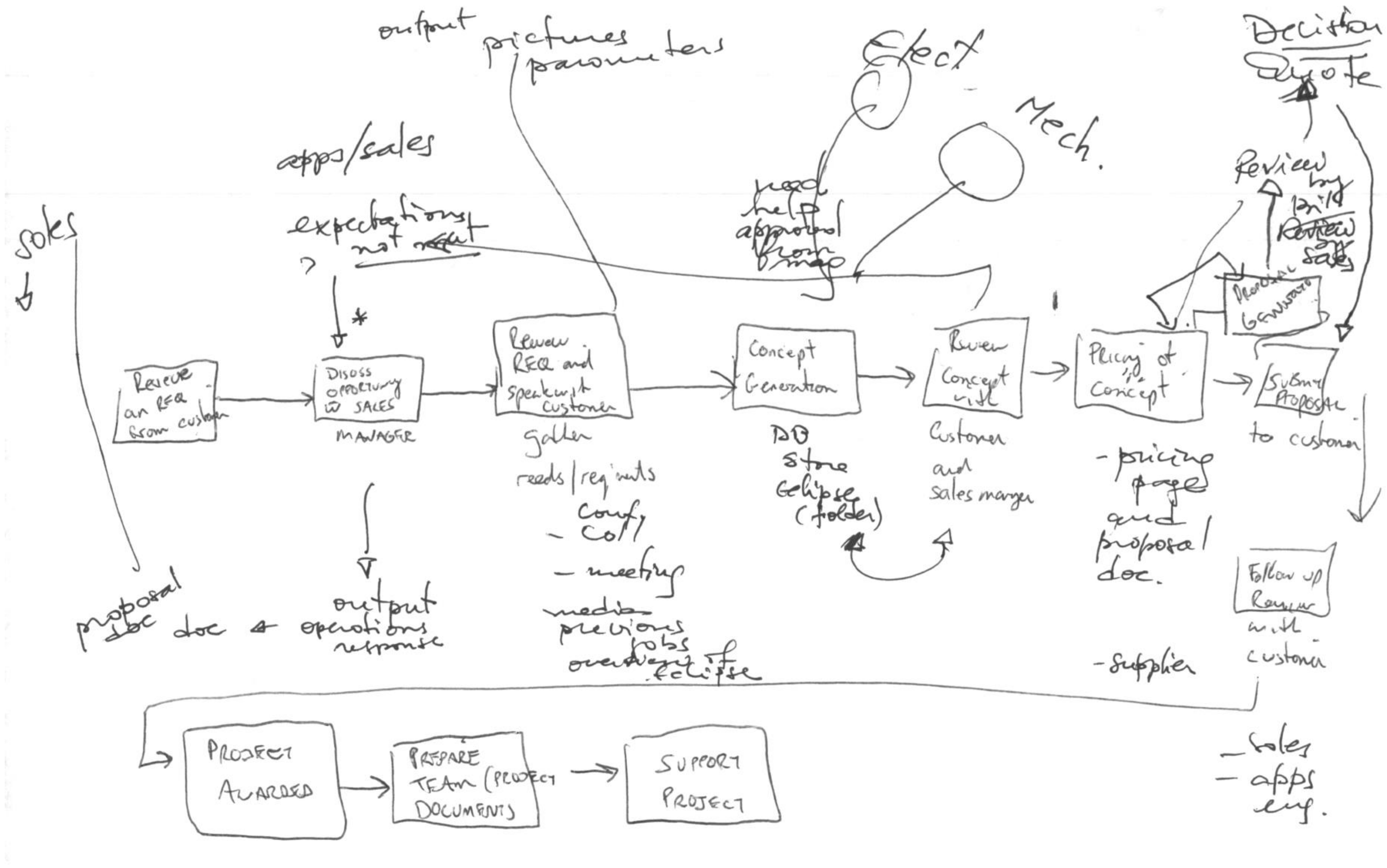

Review an RFQ from customer
Discuss opportunity w/ sales manager
Review RFQ and speak with customer
Concept Generation
Review Concept with
Pricing of concept
Submit proposal to customer
Follow up review with customer
Proposal generated
Project Awarded
Prepare team (project documents)
Support Project
sales
apps/sales
expectations not met
output
pictures parameters
Elect
Mech.
need help approval from mgr
Decision Quote
Review by mgmt
gather needs/req'ments
- conf. call
- meeting
media previous jobs overview of eclipse
DB store eclipse (folder)
Customer and sales manager
- pricing page and proposal doc.
- supplier
- sales
- apps eng.
proposal doc
doc & operations response
output

Figure 1: Raw process sketch

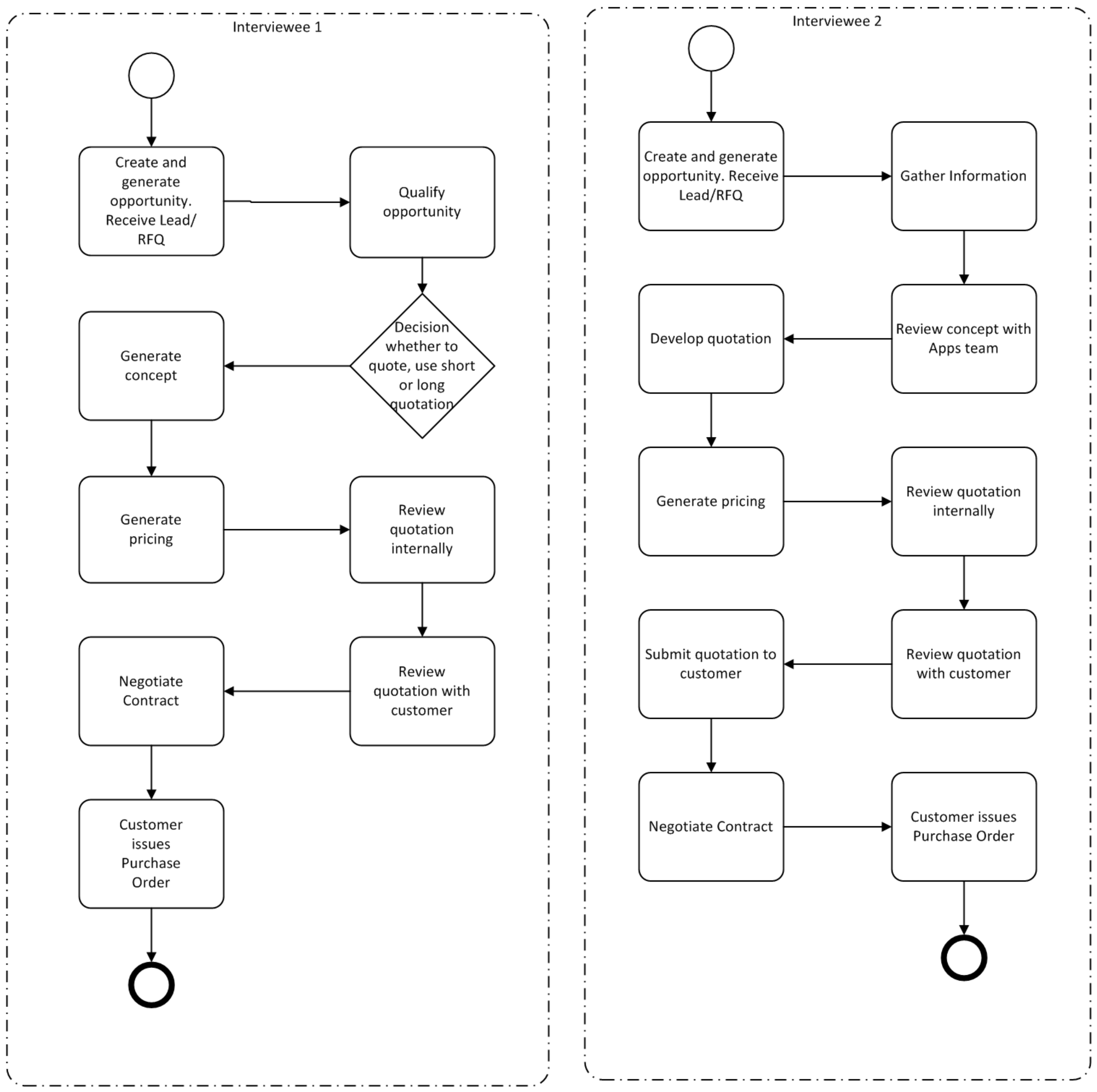

Interviewee 1
Create and generate opportunity. Receive Lead/ RFQ
Qualify opportunity
Decision whether to quote, use short or long quotation
Generate concept
Generate pricing
Review quotation internally
Review quotation with customer
Negotiate Contract
Customer issues Purchase Order
Interviewee 2
Create and generate opportunity. Receive Lead/RFQ
Gather Information
Review concept with Apps team
Develop quotation
Generate pricing
Review quotation internally
Review quotation with customer
Submit quotation to customer
Negotiate Contract
Customer issues Purchase Order

Figure 2: Computerized version of two interviewee sketches of the same process

The user's process sketches form a part of the stories that would drive the agile process.

The process sketches and descriptions from different individuals were compared to examine the degree of consistency in their perceptions of the Sales and Quotation process. It can be seen in Figure 2 that the process described by two interviewees bears some similarities and differences. Generally, interviewees listed similar tasks and similar task order at the start of the process, but as the flow continues, the tasks changed and their order also changed. Interviewee 1 included the task "decide whether to quote and what type of quotation format to use" but Interviewee 2 did not mention this task at all. Towards the end of the flow description, opinions about the tasks performed and the order in which they are performed also appear to converge.

Task descriptions were detailed to varying degrees. For example, while one interviewee only gave a cursory description of the task for developing the engineering concept of the automation equipment:

> *Applications engineer addresses the job from an engineering point of view by developing machine concept.*

Another interviewee described the process in detail as:

> *Design the machine layout using AutoCAD for 2-dimensional designs and SolidWorks for 3-dimensional designs, simulate the designs to demonstrate and test its abilities. Determine the cycle time of the machine based on the design using the cycle time sequence chart, determine the features and benefits of proposed engineering concept. Review the concepts with customer and team members and, then, the concept can be finalized. Pricing is also computed based on the finalized concept and the outcome of this process is reviewed with supervisors.*

Similarly, different individuals perceived the process structure differently. This was particularly noticeable when members of different departments describe the entire flow of the Sales and Quotation process. For instance, while members of the Sales department perceived the process of gathering information about the customer and the business opportunity as important and as one of the earliest activities to be performed, members of the Application department either didn’t mention this task as part of the process flow or did not have much to say about the task. Likewise, some other tasks performed predominantly by Applications were viewed a bit differently by the other departments involved in the process flow.

Apart from the interviews, we reviewed various existing organizational documents including original training documents and the company organizational chart, among others, to understand other details that could have been missed during the interviews. For instance, we requested to review the quotation documents submitted for different projects. Some opportunities began with well-defined customer specifications in the form of an RFQ, while others started without any formal specifications from customers, and even vague unspecific requests by customers that did not include feature preference for the equipment they requested. Other opportunities we reviewed were considered complex and difficult to achieve from the engineering perspective. We found that there are three different kinds of quotation; a *quotation letter*, a *budgetary estimate* and a *firm quotation*.

MACE has an in-house-developed enterprise resource planning (ERP) platform that it uses to manage its engineer-to-order business. Some of the features of the platform include customer relationship management, quotation management, work order management, job costing, scheduling and sequencing, and capacity management. We were given a walk-through of this system to understand how the business activities would make use of it, and so that we could extract process-related information.

To initially deduce the main tasks in the Sales and Quotation phase, we selected tasks that were common in the sketches we obtained from our first set of interviewees. Since interviewees tend to tell a compact story about the process, we used their stories to corroborate each other. In other words, if the majority of interviewees mentioned and/or sketched a given process, it was included as a major process in the model. Activities identified by relatively few interviewees were represented as sub-processes within major processes, or included as part of the detail of the process. Using this method, it was possible to distinguish between the overall workflow and the major processes and sub-processes involved in the Sales and Quotation phase. The inputs and outputs for each process step were also identified, including documents, database modules or information involved, and forms that required completion. Detailed descriptions of the activities involved in each process were written, and the organizational functions and roles performing each process were identified.[6]

Having abstracted the information using the method described above, we created the business process model by visually representing the fundamental structure, the details and the chain of activity of the Sales and Quotation phase in accordance with the BPMN standards using Bizagi Process Modeler. An excerpt of the first iteration of the model showing a portion of the layout (including symbols representing events, sequence flows, message flows, tasks and gateways, pools and lanes) is presented in Figure 3. The extract of the "Assign Resources" task, showing the performing role,

[6] Various other methods could have been used to identify and define the processes. Using the best method is not as important as user agreement about the result of the method. Agile techniques rely on user acceptance to justify their artifacts.

description and detail of the task, and the input /output for the task is presented in Figure 4. A diagram of the model developed in iteration 1 is attached in Appendix B.

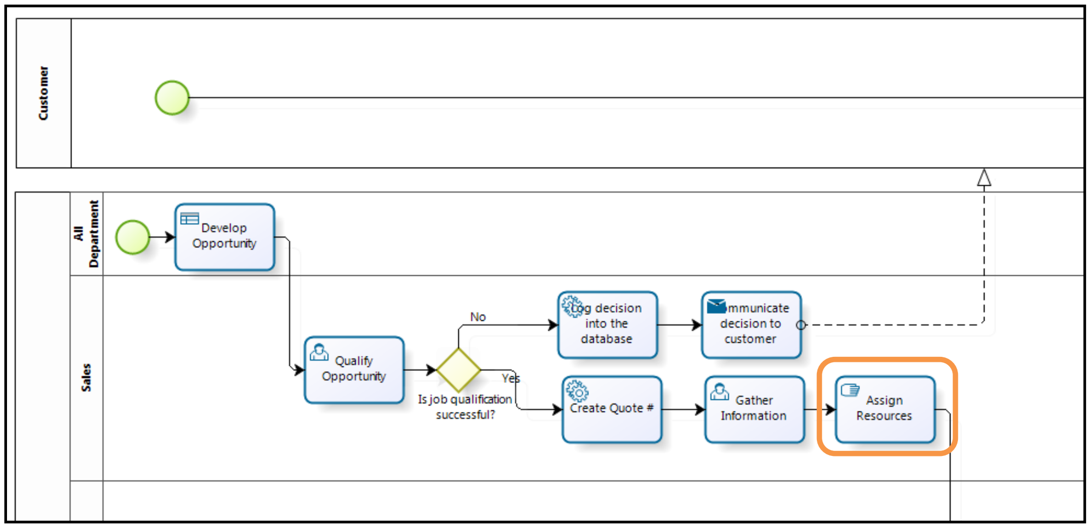
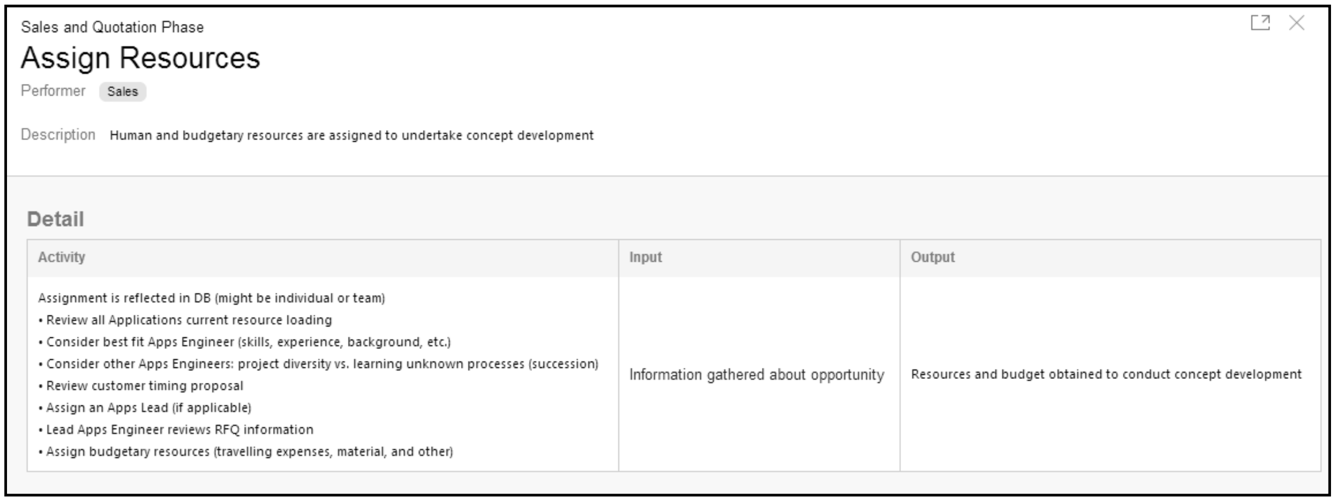


Figure 3: Excerpt of the first iteration of the process model



Sales and Quotation Phase

Assign Resources

Performer Sales

Description Human and budgetary resources are assigned to undertake concept development

Detail

Activity	Input	Output
Assignment is reflected in DB (might be individual or team) • Review all Applications current resource loading • Consider best fit Apps Engineer (skills, experience, background, etc.) • Consider other Apps Engineers: project diversity vs. learning unknown processes (succession) • Review customer timing proposal • Assign an Apps Lead (if applicable) • Lead Apps Engineer reviews RFQ information • Assign budgetary resources (travelling expenses, material, and other)	Information gathered about opportunity	Resources and budget obtained to conduct concept development

Figure 4: Detail of the "Assign Resources" task

2.3.3 Model development – Iteration 2

This iteration began with a review meeting with the project sponsor. The review meeting gave the project sponsor a chance to see the extent of work done and provide feedback to the team. It also gave the team a chance to demonstrate the features in the Bizagi Process Modeler, gauge the satisfaction level of the project sponsor, and gather additional information towards the further development of the model. Additionally, the meeting gave our team the opportunity to authenticate our interpretation of the information we had gathered about the company's activities. A key request we received from this review was that the model should separate departments into individual lanes rather than grouping all departments that performed the exactly the same task into a single lane. According to the reviewer, "separating the departments was a compulsory requirement that must be met before we can proceed with the job". His reason was that the appearance of the new model was so different from the original process map that it risked being rejected by the team that had produced the original process map. An excerpt of the change that was implemented in iteration 2 is displayed in Figure 5 to show the separation of the lanes, unlike in Figure 3 showing the departments in a grouped form. A diagram of the model developed in the second iteration is attached in the Appendix C.

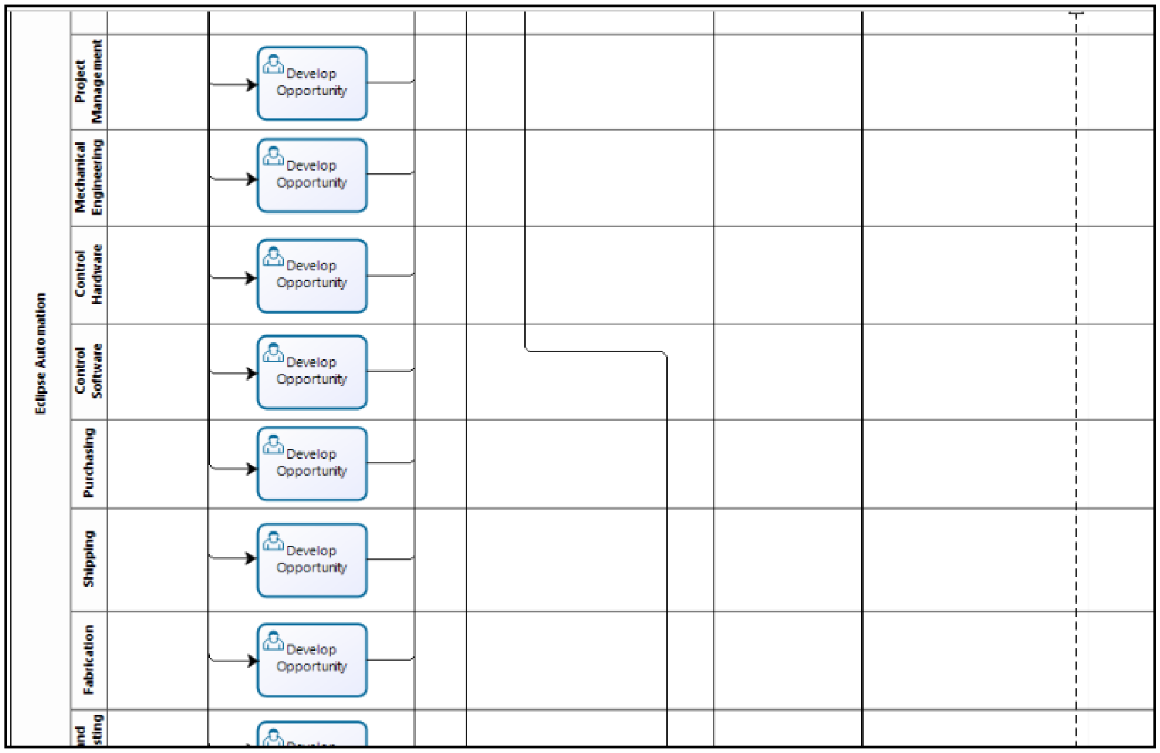


Figure 5: Excerpt of the second iteration of the process model

2.3.4 Model development – Iteration 3

The third iteration involved separate meetings with the managers of the Applications and Sales departments to receive their feedback on the model generated in the second iteration. This led to a few re-arrangements, both of the location of tasks within a lane (that is, the order in which the tasks are performed) and the location of tasks between lanes (that is, which department is responsible for the task). For example, the task “Assign resources” (a task involved with allocating human and budgetary resources to develop the concept), that was originally within the Sales department lane, was moved to the Applications department lane when it was agreed that the task is in fact performed by the Application department wherein the best-fit applications engineer is assigned to develop the concept, the assigned engineer reviews the RFQ, sets priority for the concept development task and requests budgetary resources (travel expenses, material, and other) needed to fulfill the task of developing the concept.

Apart from reviewing the iteration 2 model, we requested the managers to take us through an example of a real job that had been previously completed, starting from the point of developing the opportunity to the point of accepting the purchase order from their customer. The narration was done without referring to the model. The purpose of this exercise was to gather information about the dynamics present within the company. This method exposed us to the alternative workflow paths that exist within the system. We were then able to incorporate activities and sequence flows that might not normally fall within the “happy trail”, thereby making the model not just a model of an ideal process, but closer to a real model. Some of the ways we introduced dynamics into the model were to include feedback loops between tasks, and to append various symbols to some tasks to signify tasks usually performed repeatedly or tasks usually performed in parallel rather than sequentially. It was noted that the “Develop Concept” task is quite elaborate, since several other tasks such as machine concept design, quotation pricing, quotation document writing, RFQ to suppliers etc. were performed within this task. Consequently, this task was made into a subprocess within the main process in order to separately model the “Develop Concept” task and hide its complexity in the main model. An excerpt of the model showing the "Develop Quotation" task is shown in Figure 6.

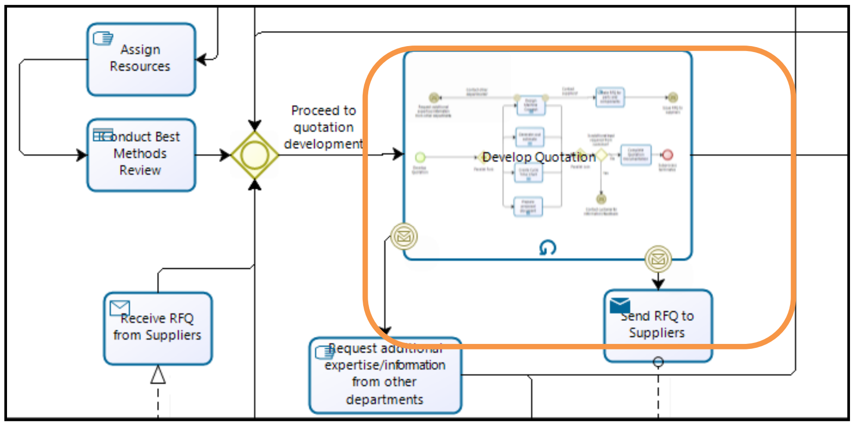

Assign Resources
Conduct Best Methods Review
Proceed to quotation development
Develop Quotation
Receive RFQ from Suppliers
Request additional expertise/information from other departments
Send RFQ to Suppliers

Figure 6: Excerpt showing the "Develop Quotation" task

Besides the qualitative data gathered through interviews, data from the company's ERP system was also accessed during this iteration to provide additional insight into the company's processes. Quantitative evidence is important because it can indicate relationships which may not be salient to the researcher or the interviewees. It can also keep the researcher from being carried away by vivid, but false, impressions in qualitative data, and it can bolster findings when it corroborates those findings from qualitative evidence (Eisenhardt, 1989).

We did a trace through the database of one of the actual jobs that the managers narrated to us. The data observed included the date the lead was registered in the database, the customer information available at that point, the date the lead became an opportunity to be pursued, the name of the applications engineer who handled the concept development, the parts and stations needed to build the machine, the engineering design, pricing for each machine component, cost of labor, and so on. We observed from the data that the version of the quotation document that was finally accepted by the customer was the seventeenth (17th) version. This information brought to light the fact that the sequence of flow from the point where the quotation is developed and submitted to the customer to the point where customer receives and reviews the quotation is bi-directional rather than uni-directional. This information prompted an update to the model.

The feedback received at this stage was incorporated into the model as the changes were being made. Consequently, the participants developed trust and a feeling that they had an impact on the development of the system. Versions of the model for every major revision were preserved.

Managers of other departments such as Accounting, Project Management, Controls, etc. were also interviewed specifically about their participation in the Sale and Quotation phase.

2.3.5 Model development – Iteration 4

In the fourth and final iteration, we conducted a group validation meeting. The validation exercise was conducted to confirm whether the model was a reasonable representation of the real-life process flow. The meeting was held in a joint session, so that the attendees from different departments could discuss and decide whether or not the information already captured in the model is what they believed to be the true representation of their business operation.

This facilitated validation session began with a run-through of the process map. Conflicting opinions that were earlier recorded were brought forward during this meeting for discussion. Changes were made to the model based on feedback from the validation meeting to present a unified view of the model. The goal of the joint session meeting was to make sure the participants were satisfied with the model and that consensus was being reached on conflicting ideas.

While discussing organizational processes and work activities during the interviews, it was common for interviewees to reflect on the pros and cons of the current process, and to identify potential areas for process improvement. We brought forward some of the concerns raised about the current process as recorded during the interviews. The purpose in highlighting concerns was to stimulate discussions among the attendees, thereby enabling us to confirm the authenticity of the concerns and also generate additional data for our documentation.

A sample identified area for improvement was the process for qualifying a lead (or opportunity). The qualification of a lead or business opportunity at MACE was frequently described by interviewees as a somewhat subjective, informal process. An important aspect of the subjectivity relates to the definition of "a qualified opportunity". Since what is considered a qualified opportunity differs from person to person, customer to customer, and opportunity to opportunity, there is a risk of inconsistent treatment of opportunities. It was thought that since the task is one of the earliest activities in the sales process which is performed to determine whether to pursue, nurture, or discard a possible business opportunity, an improvement in the qualification process in making the task more objective might improve the effectiveness of the task.

A diagram of the model produced in the final iteration is attached in Appendix D.

2.3.6 Project delivery

Prior to releasing a finalized version of the process map, a release candidate version of the map was distributed to a restricted group of staff to review the model and report any error with the content of the map. The review exercise provided an opportunity for members of the review group to familiarize with the web interface of the Bizagi process map and notify our team of any malfunction. Feedback obtained from this exercise, though minor, was considered and effected in the map.

An end-of-project review and close-out meeting were conducted to present the final model to the management of the company and to review the entire project experience. A report containing valuable project knowledge, such as the BPM Tool evaluation report, a comparison between the Bizagi model and the original process map, and potential areas for process improvements was presented and submitted to the management.

Recommendations for future work were also documented. For instance, we recommended that the Bizagi model could now be further expanded to include other business and technical processes performed at MACE. The interview and data collection methods developed for the Sales and Quotation phase could be readily adapted to other phases of MACE operations. We recommended continuity of the modeling work, starting with processes immediately downstream from the Sales and Quotation phase. Given that downstream activities are affected by decisions made upstream, it is possible that interviews with downstream roles could identify potential improvements and/or changes to the Sales and Quotation process to improve overall efficiency and organizational effectiveness. Thus, further refinement of the current Sales and Quotation model might result from efforts to model downstream processes.

Other administrative closeout activities needed to bring the project to an official close, such as signing off with the project sponsor and completing exit surveys as required by the funding partners, were also conducted.

2.4 Agile methodology as practiced in the case study

We next turn to a review of the agile methodology as practiced in the case study. Due to the need to deliver a functional MACE software system within the limited time and budget available to execute the project, we needed an approach that would rapidly produce a result. The agile approach enabled the project to begin early, and supported progressive discovery of what needed to be done. The rapid development of iterations helped to support the desired level of visibility of the project. Lastly, MACE did not request a requirements document or phase, and indeed there was no expectation of a specification or design document, a test plan, formal validation of the system, or other artifacts of a formal software development process.

In the following subsections, we present various agile approaches to development and then highlight how those approaches pertain to the MACE case study.

2.4.1 Iterative and incremental development

An agile approach involves breaking the development process into small development cycles (Shore & Warden, 2007). With each development cycle or iteration, additional features are designed, developed, tested and added to the previous increment, until a fully functional and finalized product is released to the customer. *Iteration* in this context refers to the cyclic nature of the development, while *increment* refers to the quantifiable outcome of each iteration. Generally we refer to iterative *refinement* when the process improves what already exists, and incremental *development* when the process results in progress against project objectives (Henney, 2007).

As practiced at MACE: The development of the MACE business process model went through several iterations. Each iteration was a learning process for us as researchers, since we needed to understand the company's processes and determine the level of variability in the business processes, reflecting these insights in the model. The gradual, incremental process of obtaining information and validating the model meant that interviewees and reviewers could focus on smaller and hence more manageable issues during each cycle.

At earlier stages of the MACE project, the information gathered from interviewees about the business process varied; however, as we cycled through the iterations and began to show the mapped process to the interviewees to review and approve, opinions about the process started converging, leading to stability in the model and a reduction in refactoring activities. Regular review meetings gave us an opportunity to demonstrate progress to the project stakeholders and build their confidence in our team. The "fail early" approach meant the potential cost of project failure was drastically reduced as experienced in one of the iterations, where the reviewer believed an important requirement was not properly interpreted. It was relatively easy at that early stage to rework the output of that iteration without a substantial impact on previous deliverables.

Most of the information we intended to capture was tacit knowledge held by staff; thus, it was unlikely that we could have obtained this information all at once or got everything right the first time. Information obtained during the cycle of interviews and reviews led to the continuous refinement of the MACE model.

2.4.2 People-oriented and collaborative development

Agile methods thrive on frequent face-to-face interactions between people, rather than focusing on structured processes or written documents (Shore & Warden, 2007). The main goal of frequent and ongoing communication is to ensure that information is quickly shared and the people involved can expressively communicate in ways a documentation-driven process does not support.

As practiced at MACE: Our experience in the MACE project showed that frequent and open communication with the project stakeholders provided additional clues that were not easily expressed in written form. For example, during the interactive sessions held, interviewees were able to provide us with sketches, system walk-throughs and explanation for the rationale behind some of the documentation based on past projects, all to communicate salient points that would have been difficult or lengthy to express in written form.

Since most of the stakeholders in the MACE project participated throughout the process of mapping and validating the mapped process, it was easy to successfully finalize the project, as they were already familiar with the outcome and were also accountable to ensure the resulting system met their expectations.

2.4.3 Change is welcome at any time

Agile methods welcome change, and each new cycle provides an opportunity for incremental refinement or iterative development (Shore & Warden, 2007). Changes can occur due to new management priorities, increased understanding by the users or project team, or changes in the technology being used. Highsmith describes the acceptance of change as an approach that acknowledges the reality that requirements change and are usually uncertain at the beginning of the project. Thus, development should not be managed with a fixed and rigid strategy, and instead plans should evolve based on the feedback from stakeholders and emerging constraints (Highsmith, 2013). “Complex problems in today's organizations require the interaction of many people, diverse information, out-of-the-box thinking, quick reaction, and, yes, rigorous activity at times" (Orr et al., 2001).

As practiced at MACE: Successfully executing the project required that we adapt quickly to the environment and adjust the project plan as the project cycled through the different iterations. Interviews were conducted to suit the busy schedules of the interviewees, and the mapped process was constantly adapted to align with the expectation of the stakeholders that were themselves changing during the project.

2.4.4 Tools for fast cycle times

Agile development depends crucially on tools that enable fast cycle times. "Ten minutes to green bar" is the agile rule of thumb (Kovitz, 2003); (Dan Turk, France, & Rumpe, 2002) . It is a common practice in agile development to use established standards and tools that can generate a significant part of the system automatically in order to deliver a working system fast.

As practiced at MACE: BPMN, a standard for process modelling, was used as the modeling standards in the project, and Bizagi Process Modeler was the tool that automatically generated a working system in a transparent and modular configuration.

2.4.5 Document stable knowledge, not speculative ideas

One of the core values of agile is the emphasis it places on working software over comprehensive documentation (Turk et al., 2002). Contrary to the traditional development methods which promote expansive production of documents such as the project plan, requirements specification, design documentation, test plan, user manual, and so on, the agile philosophy focuses on inter-personal communication rather than documentation. The agile concept asserts that valuable information and user needs are best obtained when users can see a working model of the system, even with limited functionality, so long as it is at the beginning of the project when uncertainties are at the highest and the knowledge about the project outcome is at its lowest. Highsmith & Cockburn put it this way; "Working code tells the developers and sponsors what they really have in front of them—as opposed to promises of what they will have in front of them. The working code can be shipped, modified, or scrapped, but it is always real" (Highsmith & Cockburn, 2001). In a bid to demonstrate or achieve process standardization, organizations usually develop comprehensive documentation or templates which tend to grow over time, and are sometimes not used in in day-to-day operations. Instead, agile affirms that documentation should be created only if necessary, and in a just-in-time manner when the process to be documented is already in a stable state, instead of documenting speculative knowledge that can often lead to rework or risk being obsolete due to changing business conditions (Ambler, 2007).

As practiced at MACE: In the case study, the main deliverable was an operational process map, so the agile approach required that project resources should be directed towards producing that outcome as soon as possible and with limited documentation. To keep communication effective and open in the MACE project, a web-based project site was set up to promote information sharing and information management within our team, while we maintained constant interaction with the staff of MACE who actively participated in the process. A detailed project report was created at the end of the project to preserve at least some aspects of the project experience. By not isolating needs and design specifications to the start of the project, the project was able to accommodate unanticipated events and trade-offs in options for actualizing the deliverables.

2.4.6 Communication

"Face-to-face conversations are the heart and soul of agile projects" (Layton, 2012). The agile method promotes face-to-face interactions, just-in-time documentation and just-enough documentation over the traditional method which suggests a plan driven, extensive up-front documentation. Paetsch et. al. explains that for us to be able to document all the information that is required build a system before actual development starts, as practiced in traditional method, we must be able to (1) anticipate future questions and (2) answer them in a concise and understandable manner, and both of these are difficult (Paetsch, Eberlein, & Maurer, 2003). There is also the risk of documenting more than is actually required, and the problem of keeping the documents up-to-date as changes occur. The agile community believes that face-to-face communication is better because it believes that more information can be gained through informal, personal communications than through formal documents. Turk et al. assert that the agile approach is based on certain assumptions and that for the agile method to thrive, some or all of those assumptions must hold true otherwise, the agile approach will not be able to deliver on its promises (Turk et al., 2002). A core assumption in agile is that people involved in a project must be engaged in face-to-face interaction through most or all of the life of the project. Customers are expected to provide input and feedback as need arises. This principle can be observed only when the participants are readily available for face-to-face planning and reviews. This suggests that the individuals involved are geographically collocated[7].

As practiced at MACE: In the case study project, face-to-face communications were used for the bulk of information transmission, both to the project team about the process and from the project team to the stakeholders about the resulting system. The project team traveled to the MACE site for all of these face-to-face communications.

2.4.7 Management involvement

The agile method advocates that for the process to be successful, all parties including subject experts and top management must be willing to participate in constant on-going conversation to ensure that the process and product knowledge is widely shared and that maintainers are familiar with the system even during development (Ambler, 2008). Ben Kovitz adds that authority figures with decision-making power and political will should be a part of this process to ensure success (Kovitz, 2003). Sillitti et. al. state that participants' availability is paramount to the success of the agile process, and that participants should be knowledgable and should have sufficient decision power (Sillitti & Succi, 2005).

As practiced at MACE: The project manager for the MACE project, who is also a vice-president of the company, was fully involved in driving participation and shielding the project from competing resource demands. The project sponsor was the CEO of the corporation, and he was completely committed to the success of the project.

2.4.8 Deliverables

The agile approach asserts the primacy of working software (Hazzan & Dubinsky, 2009): that is, the best method for demonstrating progress to the customer is by showing the user interface and demonstrating working features rather than relying on reports, specifications or work plans. The assumption is that systems can be broken down into loosely-coupled bundles that can be developed in short iterations[8]. Agile proponents suggest that it is more reliable to infer requirements and design

[7] This conflicts with the frequent practical reality that teams operate from geographically dispersed locations, spread across multiple time zones and multiple schedules.

[8] This is not always the case, especially in complex systems with features that are tightly dependent on one another.

specifications from software than to capture these specifications in documents, because requirements and specification documents are not likely to be kept up to date when the software changes. Agile methods believe the software should be the most accurate and reliable description of what a system does and how it was designed[9] (Ambler, 2010).

As practiced at MACE: The executable BPMN model, since it was the focus of the project, seemed to be sufficient and to meet the agile assumption of "code that documents itself". It was certainly the case that the model was easy to maintain because of the Bizagi software tool. Participants were able to assess the process map early to discover the possibilities and make valuable contributions to the emerging process map.

2.4.9 Testing

Agile methods recommend the test-driven design approach to software development. Kent Beck explains that the test-driven method requires that test cases are written first before the system is developed. If the system runs successfully against the test case, the system is deemed successful; else, the system has to be worked on further until it successfully passes the test (Beck, 1999). Using this approach, the system being developed and the documented test cases grow together, and the risk of the tests not matching the system, or of inadequate tests being developed, is greatly reduced. The tests case serves as executable documentation of the intent of system (Kovitz, 2003).

As practiced at MACE: We did not develop test cases; instead, we detailed each component of the model with its rationale so that as the model evolved, the rationale for each component represented in the model could be checked for consistency and relevance.

2.4.10 Executable specification

The notion of an "executable specification" involves putting information in the most appropriate place (Ambler, 2013). Instead of tucking relevant information in separate documents such as the maintenance manual, release notes, agile approaches urge that information should be put where it will be most useful and where people will most likely find it when they need it. Depending on the needs of the organization, design knowledge can be stored in test cases or as comments within code. This

[9] Thus introducing the problem: how do we know that a specific software element is a bug or feature? The assertion that the software itself describes what it is supposed to do (and that passes its tests) logically means that we cannot infer any other design intent.

approach also helps to ensure that supporting information is maintained in a single location, eliminating the need to update information in multiple sources which could lead to information inconsistency. Kovitz proposes that the sequence of programming instructions should be built in such a way that they in themselves are able to communicate the human intentions even without comments, and that developers who have never seen code at this level of refinement may not know what agile development demands (Kovitz, 2003).

As praticed in MACE: The MACE process model was built using a well-established mapping notation and a highly rated and well-documented mapping tool. All of the information provided by interviewees was either input into the BPMN model or else linked to the BPMN model. However, the rationale for some of the processes was not captured in the model and remained in interview notes and other project artifacts.

2.4.11 Follow the user view

Agile methods recommend that development should follow the users' view rather than the programmers' view (Leffingwell, 2011). In agile, requirements are captured as *user stories*: each story is a statement expressed in plain language of how we will use the system to achieve specific goals, rather than in the functional descriptions (such as "the system shall/should…") typically found in requirements documents. The reason for this is to ensure that the interpretations of the system by the developers are clear enough to the users so that they can easily identify and correct gaps and contradictions. The goal is to ensure that ambiguities will be mitigated and both the developer and the user have virtually the same picture of the requirements (Rubin & Rubin, 2010).

As practiced at MACE: The process sketches provided by users were the graphical "user story" as they showed how a user would think of MACE projects proceeding through the various departments. The use of the BPMN model to capture this story directly meant that we could never drift far from the original Visio process map, and all project participants could see a model that looked nearly identical to their original map.

2.4.12 Summary

In summary, then, the process used to develop the Bizagi-based application was an agile development process for the following key reasons:

- The project proceeded by iterations, with review of progress by various members of the project team
- Each iteration produced a higher level of functionality and content, and revised the functionality and content of the previous iteration
- The various process sketches in effect constituted the 'user stories', and were incorporated in the final system in an appearance and function very similar to what the users produced
- At no time in the MACE work did we develop a requirements document, and no one requested that a requirements document or even listing of requirements be produced
- The project team spent most of its time working with artifacts that looked like the eventual system, and not like formal software development artifacts

2.5 Has agile done the job?

We used an agile approach to develop and implement a working process model for MACE's Sales and Quotation process. This approach seemed to satisfy many needs:

- The process model was accepted by the sponsor and participants with enthusiasm
- The selected software was considered both quite affordable and quite usable
- The project was immediately given new funding to work on the other phases of MACE's business process

Overall, the system and the project were judged a success—all without ever having written a requirements document. Has this experience, then, validated the claims of agile methods that requirements are not needed? Is there any reason to be concerned? Consider the following issues:

- Although we heard many discussions of reasons for various process steps, we did not capture all the alternatives that were proposed, or the rationales behind those alternatives. Consequently, if a new project team were formed in the future to modify, maintain, or extend the MACE process, it would not be able to take advantage of knowledge we gained about alternatives
- Because we started with an existing process map, it was difficult to depart from it due to the feeling that "people would think their time had been wasted". Consequently, we were not

able to evaluate other possible process steps or flow that might have been more efficient or more reliable.   We were never able to seriously ask the question: *why* do you do it this way?

- There was no serious evaluation of alternatives, or indeed even of the existing process steps as practiced by MACE. There was no metric by which we could measure the benefit of having one more or one fewer process steps, or of including or excluding a department from a process step
- There was no overt connection or justification of the steps in the process map to MACE's overall business goals, except in the generalized sense that "if we all follow the same process, at least we'll have consistency"

The agile method focuses on deliverable software, and thus it is not a surprise that once that software has been delivered, the method is considered a success. But in most business situations, software is only part of the overall system that is needed; delivered software is only one step in a series of software versions; and change and improvement are possible even after the software is delivered.

Agile focuses so intently on delivered software because of the assumed high risk of not delivering software, as shown in many failed software projects. This risk is seen as so high that other risks are taken in order to avoid it, as shown in our considerations above.   However, we will show that the other risks are important too; indeed, the proper identification and management of risk is at the heart of the method we propose.

Chapter 3
The Failure Mode Based Requirement Elicitation Method (FBREM)

3.1 The FBREM Approach

To introduce requirements elicitation into the agile process, we propose Failure Mode Based Requirement Elicitation Method (FBREM). FBREM is a method that can be used to expand generic objectives, user stories, and other agile artifacts into system-specific, realizable and verifiable requirements.

FBREM identifies posssible *failure modes*, which are the set of undesirable phenomena imposed by the malicious objective that will ultimately cause the system to reach a state that is inconsistent with its goal (Lin, Ince, Moffett, Hall, & Mk, 2003). These failure modes are then quantitatively and qualitatively analyzed to determine the possible *effect* of the failures in terms of what the experience of the failure on end users, the impact of the failures on goals, the root *causes* of the failures, the *likelihood* of the causes of the failure occurring and ultimately, what *countermeasures* can be put in place to eliminate the root causes of the failures (or at least alleviate their effects).

As depicted in Figure 7, FBREM takes as input artifacts produced during agile work such as user stories and process components, and produces from these a formal requirements specification that is well-structured and based on business goals. In addition to producing and structuring requirements so that the rationale of any particular software feature can be traced back through levels of requirements to the business goal, FBREM also provides teams with an objective method for prioritization of requirements and establishing traceability between various levels of requirements and software features. FBREM can be applied after or during the agile process, depending on when a team feels the need for more formal analysis.

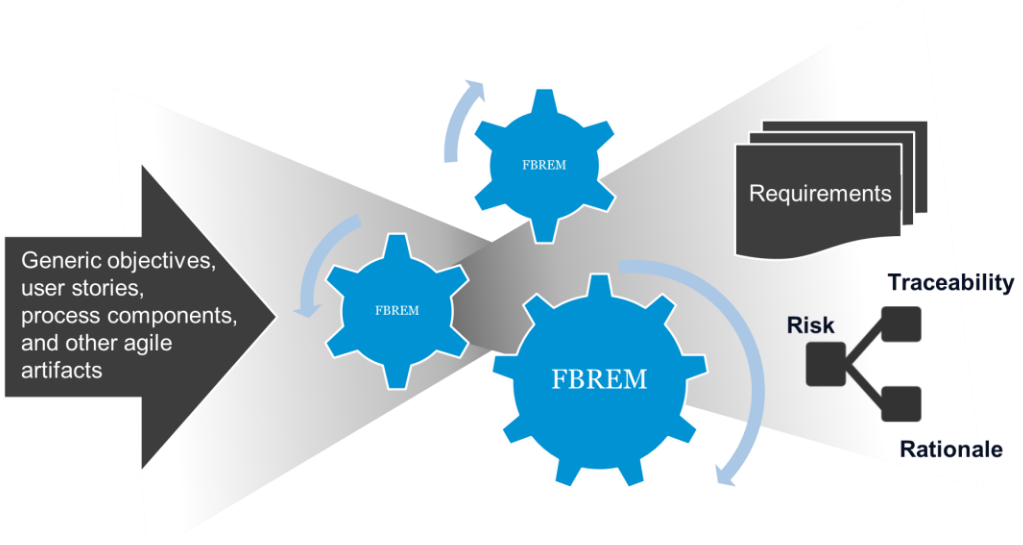

Generic objectives, user stories, process components, and other agile artifacts
FBREM
FBREM
FBREM
Requirements
Traceability
Risk
Rationale

Figure 7: FBREM input and output diagram

Our contribution with FBREM is to present a risk-driven method for systematically eliciting concretely specified requirements from agile artifacts in a way that the rationale behind every functional requirement can be traced to some business objective.

The premise of this approach is that a system exists largely because of the inherent need to eliminate one form of risk or another: whether it is the risk of not meeting regulatory requirements, the risk of losing market share, the risk of not meeting customers' expectations, the risk of exposing staff to safety or health hazards, or the risk of not making new sales, many business needs can be expressed as a response to risk.

Our method assumes that understanding risks will help in formulating the best possible requirements. Our risk-driven approach also attempts to address some of the problems that impede elicitation of requirements. Such problems include incomplete understanding of needs; incomplete domain knowledge; ill-defined boundary between the internal workings of the system and its external environment; difficult to substantiate intentions; unorganized bulky information sources and overlooking of crucial tacit assumptions (Tsumaki & Tamai, 2005).

The main risk analysis tool used in FBREM is Failure Modes and Effects Analysis (FMEA). FMEA is a well-established and widely-used reliability engineering tool. The purpose of FMEA is to identify possible failure modes of the system, evaluate their effects on system behavior, and advance appropriate counter-measures to eliminate or suppress these effects (IEC, 2008). FMEA will be discussed in detail in later sections.

3.2 Conceptual model of FBREM

We present in this section the various elements of FBREM using the process-deliverable diagram (PDD) shown in Figure 8. A PDD is a meta-modeling technique used in the creation of methods in order to show the stepwise activities and actions as well as the deliverables produced from each of the activities (Syed et al. 2008). The left side depicts the process steps of FBREM, and the deliverables produced in each of the activities performed are on the right side.

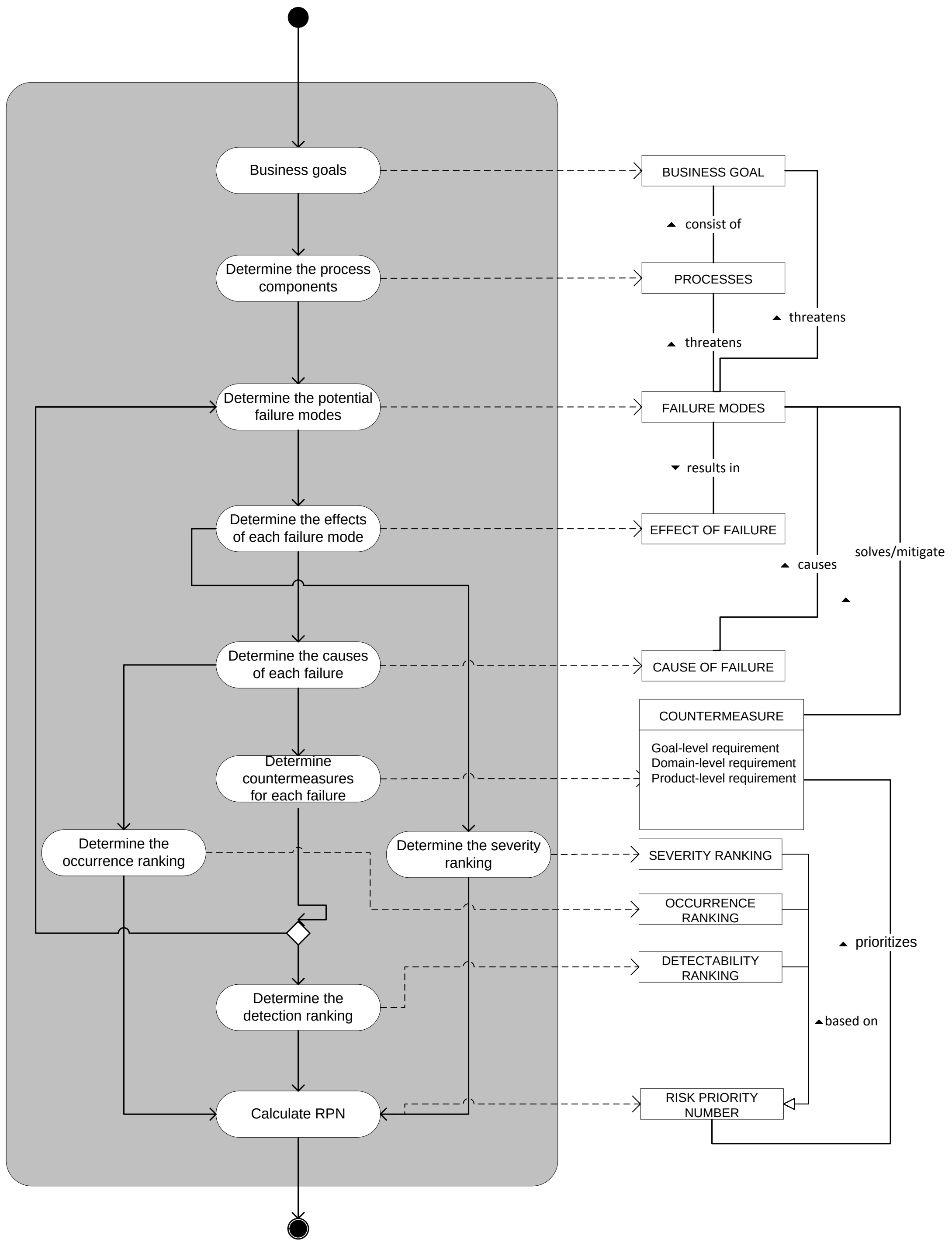


Figure 8: Process-Deliverable Diagram for the FBREM method

At a high level, FBREM proceeds as follows:

1. Determine business goals
2. Determine process components
3. Determine failure modes
4. Determine effects of failure
5. Determine causes of failure
6. Determine countermeasures
7. Determine detection ranking
8. Calculate CPN

Using the process deliverable diagram as depicted in Figure 8, we next describe each of the components of FBREM.

Business goals are the primary intentions of the business. They are the expected results and outcomes the business desires to achieve and hence, to which it is willing to commit resources. The business goals determine the nature of resources such as people, processes or tools that will be required; therefore, the business goal is a form of high-level requirement. Examples of such business goals could be “Reduce total greenhouse gas emissions by 20 per cent in six months”, “Increase sales by 30%”, “Ship goods to customers at minimal cost” and “Efficiently conduct the sales initiation and leads qualification process”. Business goals are usually stated in broad terms, as they represent general intentions and may not be directly translatable into functional behaviors. Business goals are typically the desired end result of user stories in an agile process.

Process components are the constituent parts of the business that work together to produce a result. Process components provide the set of related structured tasks and processes that is in place to achieve some activity. For a business goal that involves shipping goods to customers at minimal cost, some of the process components may include: scheduling manufacturing, finding a low-cost shipper, finding a low-cost insurer and so on. Process components contain information about business activities, business entities, workflow structure and other constraints related to realizing some activity. Process components can be visually modelled in a format that shows the workflow between the various components using a notation such as Business Process Modeling Notation (BPMN) used in our case study discussed in Chapter 2. Process components are typically described as user stories in an agile process.

Failure modes are the ways (modes) in which process components are potentially unable to meet business goals. Failure could mean failing to performing the task as intended, not performing the task within the expected time limit, a malfunction occurring while performing the task, or not performing the task at all (Carlson, 2012). Failure modes in process components or subsystems could also arise due to failures in a lower-level subsystem or could cause a failure in a higher-level component (Gan, Xu, & Han, 2011). A list of potential failure modes would be generated by conducting the "determine potential failure mode" task for the particular component, subsystem, or system that is being considered. Failure modes do not have a direct analog in agile methods, nor do any of the remaining components of FBREM.

In FBREM, failure modes are *anti-requirements*, that is, they correspond to "shall not" behaviors of the system. Examples of failure modes for a sale might be "customer's credit check not conducted", or "customer credit score wrongly computed". Identifying failure modes and then stating that these should not occur, is the FBREM approach to eliciting requirements.

Effects of failure modes are the consequences of a failure mode on the business goal, processes, systems or functions. Failure effects are described in terms of what a customer or end-user might experience. For the failure mode "customer's credit check not conducted" a potential effect could be "granting credit to a customer who has a bad credit history"; this could eventually lead to the effect "loss in revenue for the company". The effects of a failure mode can have impact in varying degrees; some effects are more severe than some others. Hence, there is a need to estimate the impact of the effect using the severity rating scale.

Severity is a numerical ranking of the impact an effect would have on the business goal, processes, systems or functions if the failure mode occurs. In our study, a scale of 1 to 5 was used, where 1 indicates an insignificant effect and 5 indicates an effect that critically impacts the intended result[10]. This scale is a relative ranking within the scope of the specific business goal, and is determined without regard to the likelihood of occurrence or detection (Carlson, 2012) . The severity scale used in our case study is shown in Table 1. For example, the severity of a failure mode will be ranked "5" or Critical if, when the failure mode occurs, the customer will not eventually issue a purchase order.

[10] The choice of scale from 1 to 5 is common in Risk Priority Number (RPN) practice, which is why we used it here. The FBREM method could use a different scale if that was determined to be more appropriate.

Ranking	Effect	Severity of Effect
1	Insignificant	None
2	Minor	RFQ rework, Clarification meetings
3	Moderate	Multiple proposal revision
4	Major	Inability to submit a complete proposal
5	Critical	Customer does not issue PO

Table 1: Severity ranking table

Potential causes of failure include causes both internal and external to the business goal, processes, systems or functions. For each mode of failure, causes are identified. An example of such a cause would be “software failure “which can result in delay (failure mode) in processing the credit check.

Likelihood is a numerical ranking indicating the likelihood that the potential cause of failure will occur. The likelihood ranking is a relative value and it is determined without regard to the severity of the effect of the failure or the likelihood of detecting a failure mode arising from a particular cause. As with severity, we used a scale of 1 to 5 in our study to indicate range of likelihood. A ranking of 1 indicates that the failure cause is very unlikely to occur (that is, the likelihood of the cause of failure is 1/100) whereas, a ranking of 5 indicates a frequency of 1/5 and it is described as very likely to occur. The Likelihood scale used in our case study is shown in Table 2. For example, a potential cause of failure is ranked “1” or Very unlikely, if its average frequency of occurrence is 1 in every 100 business opportunity considered. Likelihood is determined by the members of the project team.

Ranking	Likelihood	Frequency (1 in _)
1	Very unlikely	100
2	Unlikely	50
3	Possible	30
4	Probable	20
5	Very likely	5

Table 2: Likelihood ranking table

Countermeasures are mitigation, detection, or prevention mechanisms. By identifying countermeasures, we identify mechanisms that will provide the functionality that avoids the anti-requirements, or conversely, meets the requirements. Countermeasures may include actions, processes, devices, solutions, functionalities, systems or features intended to prevent the failure mode from compromising the business goal. Countermeasures as identified by FBREM are treated as requirements that have been elicited and rationalized by identifying unwanted failure modes and countering them with the countermeasures.

Risk reduction is a numerical ranking that assesses the likelihood that the countermeasure provided to prevent the cause of the failure mode from occurring will detect the failure mode (IMCA, 2002). Table 3 shows the scale used in our study, where 1 indicate that the countermeasure most certainly detects the failure mode and 5 indicate that the countermeasure cannot detect the failure. The risk reduction ranking is a relative ranking within the scope of the specific business goal and is determined without regard to the likelihood or severity of the failure (Carlson, 2012). For example, a countermeasure is ranked "1", that is, almost certain, if its chances of mitigating the potential failure mode is greater than 90%. Risk reduction is evaluated so we can compare countermeasures.

Ranking	Risk reduction	Chances
1	Almost certain	> 90%
2	High	> 60 to 90%
3	Moderate	> 40 to 60%
4	Low	>1 to 40%
5	Absolute uncertainty	Cannot reduce

Table 3: Risk reduction ranking table

The requirements elicited as countermeasures can be derived at different levels of detail, with each level addressing different needs. Westfall categorized different levels of requirement as *Business level*, *User level* and *Project level* (Westfall, 2006b). Adapting this categorization to FBREM, at the top we have the *business requirements*, representing the high-level detail of what needs to be done to mitigate the failure mode. The business requirement defines the scope from which the other levels and types of requirements will be derived to provide the desired solution.

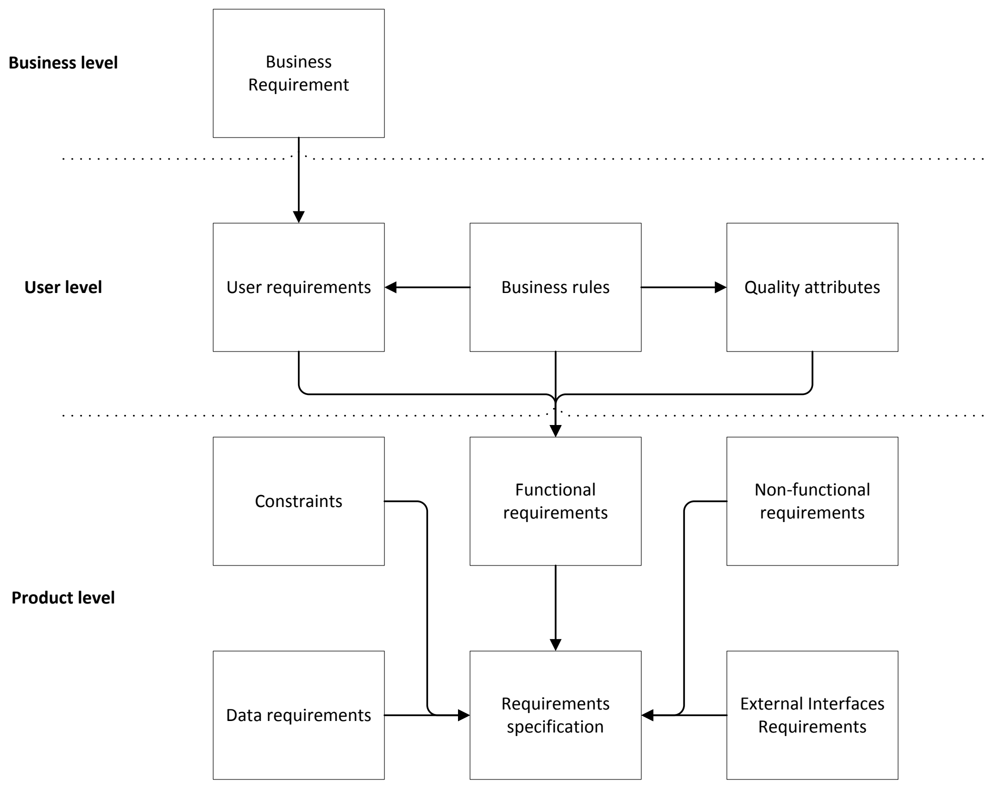


Figure 9: Levels and types of requirements

The second level (the user level) addresses the *user requirements*. This level describes what the users will need from the solution. It specifies how users will be able to interface with the solution in order to achieve the business goals. The other types of requirements generated at this stage include *business rules* which defines the structure that controls the operation of the intended solution, they include policies and practices, and *quality attributes* (such as usability, efficiency, portability, and maintainability) which are characteristics that define the qualities of the intended solution (Wiegers, 2000).

The requirements derived at user level can be used to generate the third requirement level, which is the *product level*. This level identifies specific behaviors that must be exhibited by the intended solution in order to fulfill the user level requirements, business level requirements and ultimately, the broadly stated intentions of the business goals (Wiegers, 2000). Types of requirements specified at the product level include *solution constraints* which define any restrictions on the solution design, the *external interfaces* requirements which define the requirements for sharing information with parties or systems external to the intended solution, *data requirements* which specifies the content and structure of the data for solution, and the *functional requirements* which specifies that functionality and features that the solution should have in order to fulfill the user requirement.

As depicted in the Process-Deliverable Diagram shown in Figure 8, iterating over the process through the “determine potential failure modes”, “determine the effect of each failure mode”, “determine the cause of each failure” and “derive the countermeasure for the failure” processes produces different levels and types of requirements.

For each failure mode identified, the causes and effects as well as the countermeasures that address the causes of the failure are determined, and the countermeasures identified are the requirements. Each of the requirements identified belongs to a requirements type and requirements level category. The first iteration usually produces business level requirements. For each countermeasure determined, possible ways in which it can fail are identified, along with the causes of the potential failure and their effects, and corresponding countermeasures. Similar to the first iteration, the countermeasures identified at this stage are also a type of requirement, but at a lower level. The iteration process can be continued until the desired level of detail of requirement and type of requirement is elicited. The final iteration should produce the product level requirements from which a *requirement specification* document which contains the constraints, functional requirements, non-functional requirements, data requirements, external Interfaces requirements and any other requirement that contain enough and all necessary information that is required to attain the business goal is documented.

For each pair (failure mode, countermeasure), a *countermeasure priority number* (CPN) can be calculated. Each failure mode gets a numeric score that quantifies

(a) the likelihood that the failure will occur

(b) the ability of the countermeasure to reduce the risk of the failure mode occurring

(c) the severity the effect of the failure will have on the business goal

The product of these three scores is the countermeasure priority number (CPN) for that failure mode

$$\mathrm{CPN} = \mathrm{Severity} \times \mathrm{Likelihood} \times \mathrm{Risk\ reduction}$$

CPN is based on the notion of risk priority number or RPN, which is the product of risk severity, risk likelihood, and risk detectability (IHI, 2013). RPN is a commonly employed metric in risk analysis

(Carbone & Tippett, 2004). CPN is similar in that it considers the severity and likelihood of the risk, but it includes the risk reduction estimate of the countermeasure; thus, it gives us a measure of the residual risk after the countermeasure is applied.

Multiple countermeasures can typically be generated for each failure mode. Since we do not always want to implement multiple countermeasures, there is a need to evaluate the countermeasures to determine which ones to use. CPN is a valuable tool for quantifying options to realize the business goals within the bounding condition, since for each particular risk, CPN tells us the relative goodness of each countermeasure. Other parameters such as cost of implementation, implementation time, nature of resources required and how urgently the countermeasures need to be implemented are some of other estimates that could benefit from CPN. Another important use of CPN is to assess the effectiveness of the countermeasures after they have been implemented. Calculating CPN before implementing the countermeasure and after the countermeasure (when we might have more empirical data about severity, likelihood, and risk reduction) could improve our ability to determine the effectiveness of countermeasures.

3.3 An application of FBREM

In this section, we illustrate FBREM by applying it to the MACE case study. Our illustration will be limited to the Sales and Quotation phase of the business process, just as in Chapter 2, although FBREM could be applied to any phase of any business process. The portion of the process under consideration is shown in Figure 10.

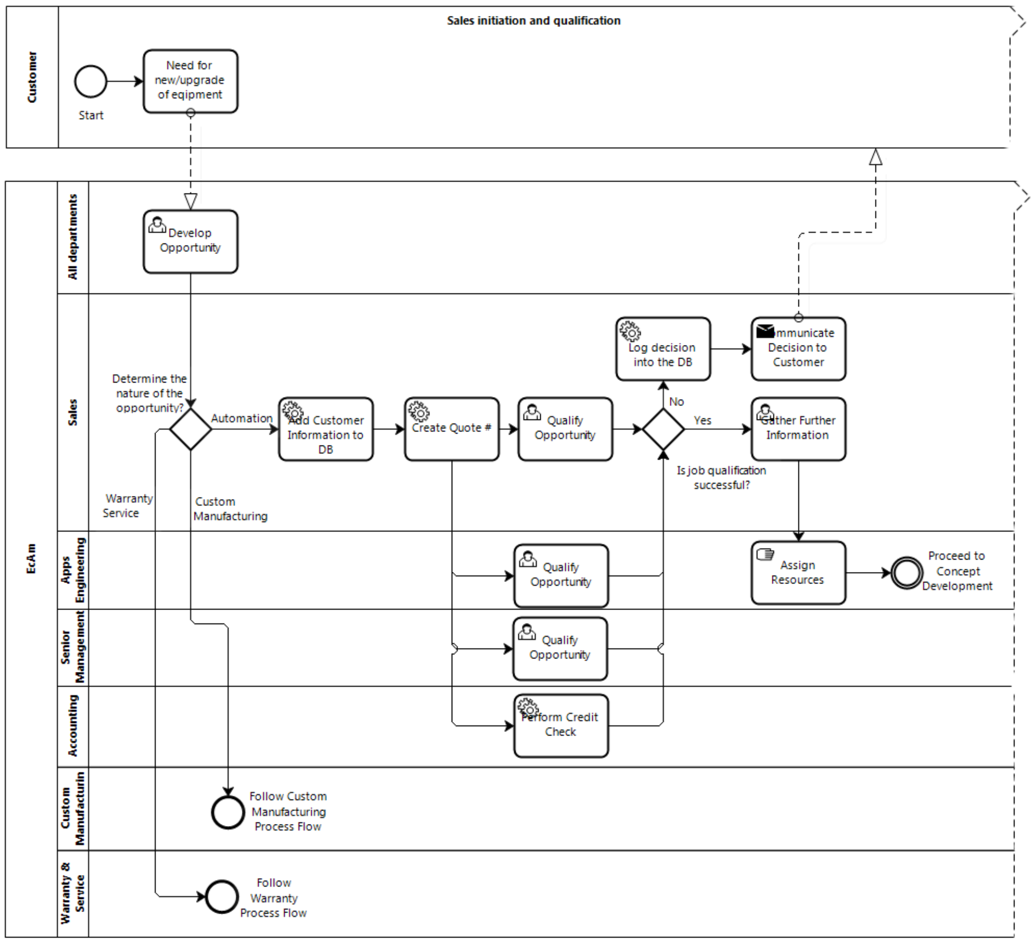

Sales initiation and qualification
Customer
Start
Need for new/upgrade of eqipment
EcAm
All departments
Develop Opportunity
Sales
Determine the nature of the opportunity?
Automation
Add Customer Information to DB
Create Quote #
Qualify Opportunity
No
Yes
Log decision into the DB
Communicate Decision to Customer
Gather Further Information
Is job qualification successful?
Warranty Service
Custom Manufacturing
Apps Engineering
Qualify Opportunity
Assign Resources
Proceed to Concept Development
Senior Management
Qualify Opportunity
Accounting
Perform Credit Check
Custom Manufacturin
Follow Custom Manufacturing Process Flow
Warranty & Service
Follow Warranty Process Flow

Figure 10: MACE's Sales and Quotation process

FBREM can be managed in spreadsheet tables, or by graphical structuring of the elements according to the analytic hierarchy process described on page 58. A graphical technique will be used to demonstrate selected elements of FBREM process in this section. The entire FBREM requirement elicitation process, using a spreadsheet format, can be found in Appendix A.

Step 1: Determine the business goal

The business goal of the Sales and Quotation process is to initiate the sales process and screen out undesirable leads.

Step 2: Determine the process component

Process components are the constituents parts of the business that will work together to produce the result intended by the business goal. The BPMN diagram shown in Figure 10 consists of process components including activities, events and gateways.

Activities are tasks that are performed within the process. The activities are:

1. Develop opportunity
2. Add customer information to the database
3. Create the quote information in the database
4. Qualify the opportunity
5. Perform credit check
6. Log decision whether or not the job is qualified into the database
7. Communicate decision not to quote job to customer is the job will not be quoted
8. Gather further information about the opportunity
9. Assign human and material resources to develop the equipment concept.

Events are occurrences that happen within the process. The events are:

1. Enter the custom manufacturing process
2. Enter the warranty process flow
3. Enter the concept development process

Gateways control the flow of the process. The gateways are:

1. What is the nature of the opportunity?
2. Is job qualification successful?

For demonstration purposes, we will apply FBREM only to the process components shown in Figure 11.

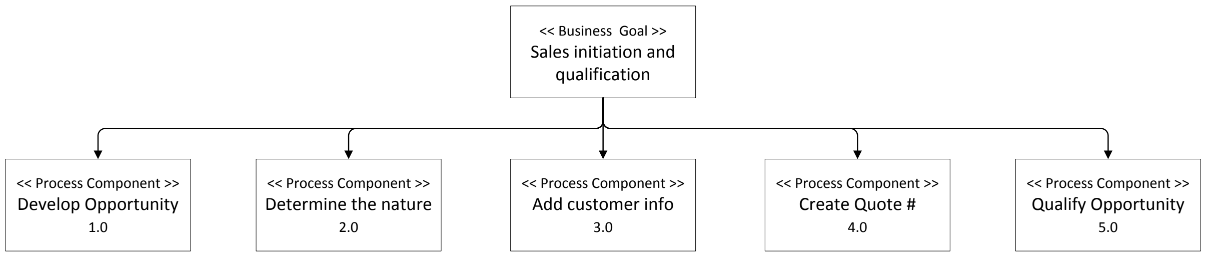


Figure 11: Process component

The “Develop opportunity” process component involves sourcing and identifying business opportunities for the company. These opportunities can come as business leads or in form of Requests for Quotation. “Determine the nature of the opportunity” is a quick assessment to determine if the opportunity fits the MACE business profile, or if it should be referred to some other division of the company. “Add customer information to the database” is the process of maintaining customers and opportunity-related information in the Enterprise Resource Planning (ERP) system. “Create quote number” involves initializing the quotation creation process in the ERP: a quote ID is created and quotation templates are generated. “Qualify Opportunity” is an opportunity pre-qualification activity, in which the opportunity is to be assessed for the likelihood of winning a purchase order (PO).

Step 3: Determine the potential failure modes

We next determine the failure modes for each of the process components. For this example, we limit the failure modes determination to the “Develop Opportunity” and “Qualify Opportunity” components, although they can be defined for any of the process components. Failure modes are determined by considering questions such as: In what way can the process fail to perform its intended function? In what way can the process perform an unintended function? What has gone wrong with the process in the past? How could the process be abused? (Carlson, 2012). For “Develop Opportunity”, the following failure modes were determined:

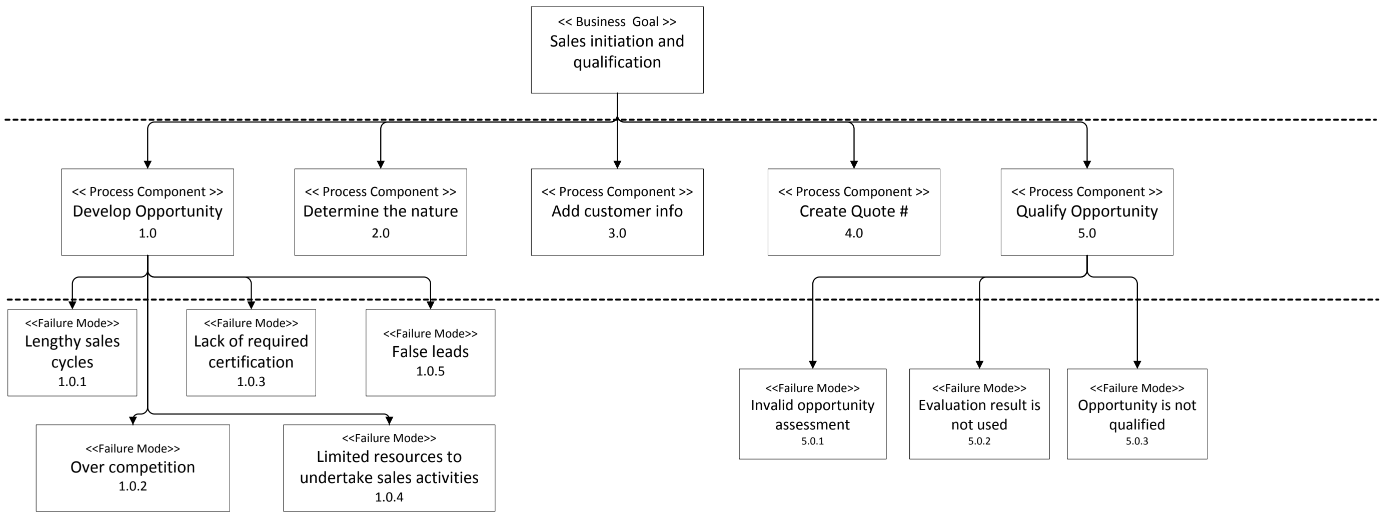
- Lengthy sales cycles (1.0.1)
- Over competition (1.0.2)
- Lack of required certification (1.0.3)

- Limited resources to undertake sales activities (1.0.4)
- False leads (1.0.5)

For “Qualify Opportunity” the following failure modes were determined:

- Invalid opportunity assessment (5.0.1)
- Evaluation result is not used (5.0.2)
- Opportunity is not qualified (5.0.3)

The graphical representation of the failure mode decomposition is shown in Figure 12.



<< Business Goal >> Sales initiation and qualification
<< Process Component >> Develop Opportunity 1.0
<< Process Component >> Determine the nature 2.0
<< Process Component >> Add customer info 3.0
<< Process Component >> Create Quote # 4.0
<< Process Component >> Qualify Opportunity 5.0
<<Failure Mode>> Lengthy sales cycles 1.0.1
<<Failure Mode>> Lack of required certification 1.0.3
<<Failure Mode>> False leads 1.0.5
<<Failure Mode>> Over competition 1.0.2
<<Failure Mode>> Limited resources to undertake sales activities 1.0.4
<<Failure Mode>> Invalid opportunity assessment 5.0.1
<<Failure Mode>> Evaluation result is not used 5.0.2
<<Failure Mode>> Opportunity is not qualified 5.0.3

Figure 12: Potential failure modes

Step 4: Determine the effects of each failure mode

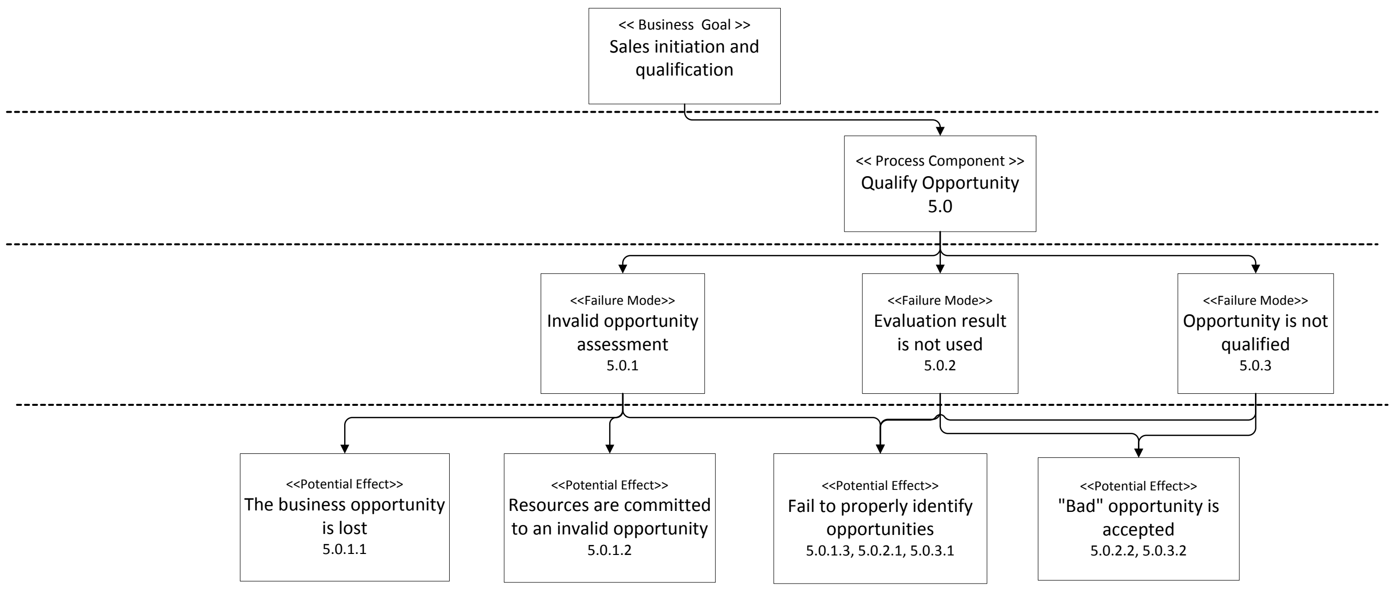
The effects of a failure mode are the impacts of that failure occurring. We determine effects of failure modes by asking questions such as “What adverse consequences could be experienced by the company, opportunity or customer if the failure mode occurs?” and “Could the failure mode result in the violation of a regulatory requirement?”

The effects of each of the failure modes for our example are graphically displayed in Figure 13. The consequences of “Invalid opportunity assessment” failure mode on the “Qualify opportunity” process are determined to be

- Potential loss of the business opportunity (5.0.1.1)
- Potentially committing resources to an invalid opportunity (5.0.1.2)
- Potentially failing to properly identify the nature of the opportunity (5.0.1.3)

The effects of the failure modes “Evaluation result is not used” and “Opportunity is not qualified” are determined to be:

- Potentially accepting a “bad” opportunity (5.0.2.2, 5.0.3.2)
- Failing to properly identify the nature of the opportunity (5.0.2.1, 5.0.3.1)



<< Business Goal >>
Sales initiation and qualification
<< Process Component >>
Qualify Opportunity
5.0
<<Failure Mode>>
Invalid opportunity assessment
5.0.1
<<Failure Mode>>
Evaluation result is not used
5.0.2
<<Failure Mode>>
Opportunity is not qualified
5.0.3
<<Potential Effect>>
The business opportunity is lost
5.0.1.1
<<Potential Effect>>
Resources are committed to an invalid opportunity
5.0.1.2
<<Potential Effect>>
Fail to properly identify opportunities
5.0.1.3, 5.0.2.1, 5.0.3.1
<<Potential Effect>>
"Bad" opportunity is accepted
5.0.2.2, 5.0.3.2

Figure 13: Potential effects of failure modes

Step 5: Determine the causes of each failure

The causes of a failure are the reasons why a failure mode occurs. The causes are determined by asking questions such as: “What could cause the kind of failure effects experienced?” and “Are there actions that can result in those effects if performed or if not performed?" and "Can a combination of causes result in a new kind of failure effect?"

The identified causes of the failure modes in our example are shown in Figure 14. It was discovered that each of the potential effects of failure modes can be traced to one or more causes. The causes were determined to be:

- Evaluation criteria not well defined
- Evaluation criteria not evaluated for opportunity
- Assessment is not done by trained individual
- Assessment is done by trained individual but they do not apply procedure correctly
- Evaluation result is not used
- Lack of sufficient data to do proper evaluation
- Lack of sufficient time to do proper evaluation
- Lack of standard operating procedure
- Disregard for standard operating procedure
- Lack of training on procedure

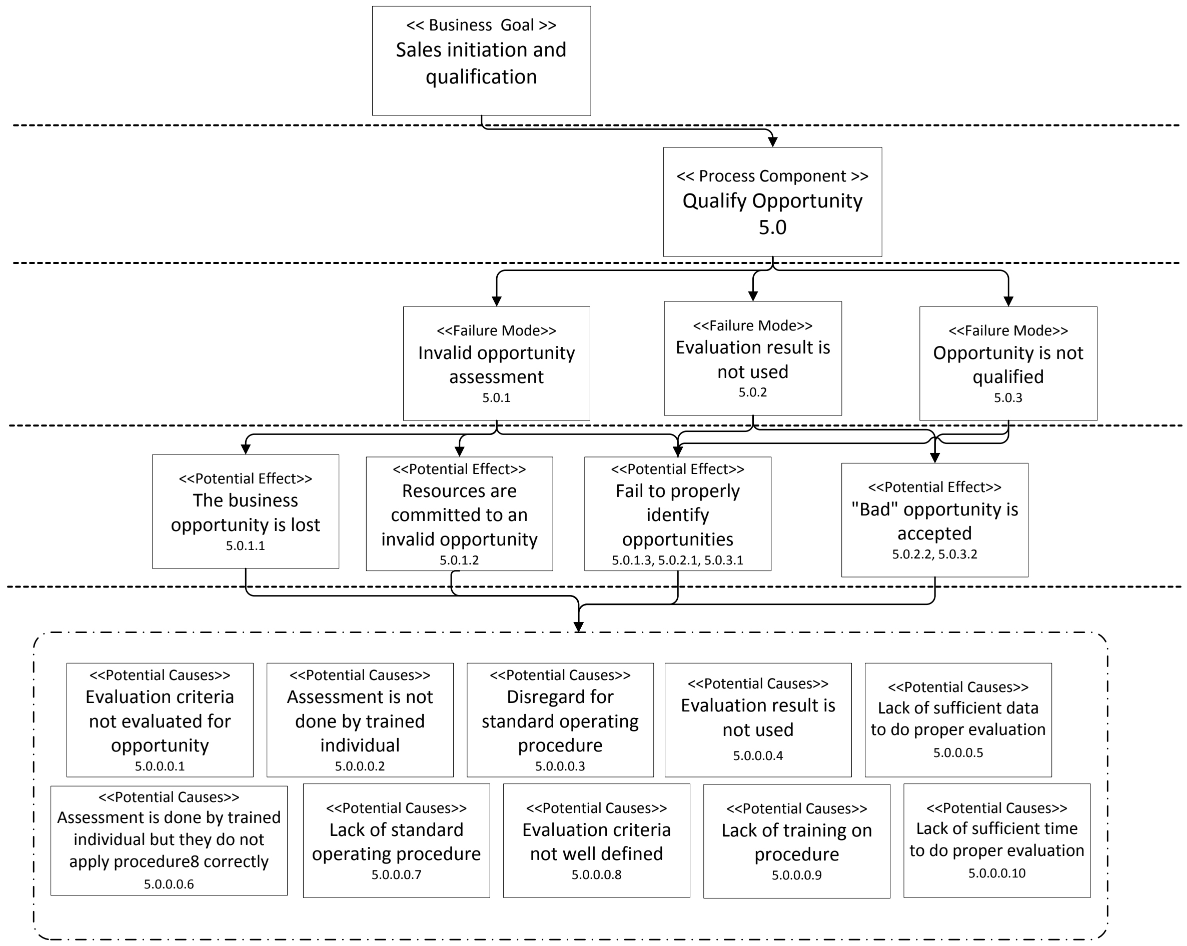

<< Business Goal >>
Sales initiation and qualification
<< Process Component >>
Qualify Opportunity
5.0
<<Failure Mode>>
Invalid opportunity assessment
5.0.1
<<Failure Mode>>
Evaluation result is not used
5.0.2
<<Failure Mode>>
Opportunity is not qualified
5.0.3
<<Potential Effect>>
The business opportunity is lost
5.0.1.1
<<Potential Effect>>
Resources are committed to an invalid opportunity
5.0.1.2
<<Potential Effect>>
Fail to properly identify opportunities
5.0.1.3, 5.0.2.1, 5.0.3.1
<<Potential Effect>>
"Bad" opportunity is accepted
5.0.2.2, 5.0.3.2
<<Potential Causes>>
Evaluation criteria not evaluated for opportunity
5.0.0.0.1
<<Potential Causes>>
Assessment is not done by trained individual
5.0.0.0.2
<<Potential Causes>>
Disregard for standard operating procedure
5.0.0.0.3
<<Potential Causes>>
Evaluation result is not used
5.0.0.0.4
<<Potential Causes>>
Lack of sufficient data to do proper evaluation
5.0.0.0.5
<<Potential Causes>>
Assessment is done by trained individual but they do not apply procedure8 correctly
5.0.0.0.6
<<Potential Causes>>
Lack of standard operating procedure
5.0.0.0.7
<<Potential Causes>>
Evaluation criteria not well defined
5.0.0.0.8
<<Potential Causes>>
Lack of training on procedure
5.0.0.0.9
<<Potential Causes>>
Lack of sufficient time to do proper evaluation
5.0.0.0.10

Figure 14: Potential cause of failure

Step 6: Determine the countermeasures for each failure mode

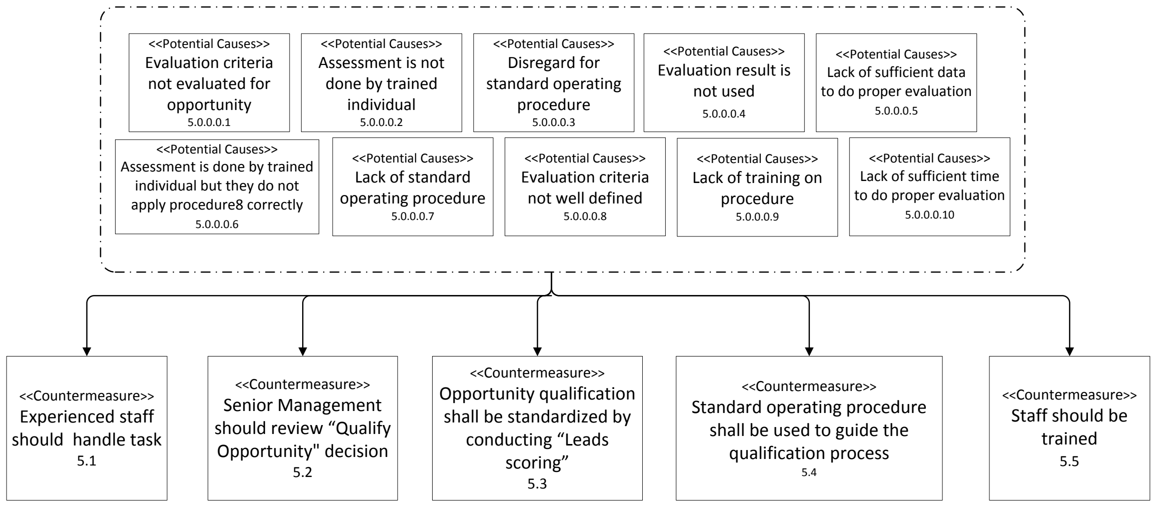
A countermeasure is a technique that will stop the cause of a failure mode, and thus reduce or eliminate the likelihood of the failure mode occurring. Some questions recommended by Carlson that could be considered in order to derive countermeasures include (Carlson, 2012):

- What can be done to reduce the impact of the failure to a safe level by modifying the process?
- If the process fails, how can the company be protected from breaching contracts or regulations?
- How can the current process be made more robust?

- What tests or evaluation techniques need to be added or modified to improve chances of detecting erros before they can occur?
- What warning signs mechasim can be built into the process?
- If the recommended actions are implemented, will they be sufficient to reduce the severity of and likelihood of failures?

The countermeasures shown in Figure 15 were determined to mitigate the potential risks faced by the business goal by carrying out the Qualify opportunity process component:

- Experienced staff should handle task (5.1)
- Senior management should review “Qualify Opportunity" decision (5.2)
- Opportunity qualification should be standardized by conducting “Leads Scoring” (5.3)
- Standard operating procedure should be created (5.4)
- Staff should be trained (5.5)



<<Potential Causes>> Evaluation criteria not evaluated for opportunity 5.0.0.0.1
<<Potential Causes>> Assessment is not done by trained individual 5.0.0.0.2
<<Potential Causes>> Disregard for standard operating procedure 5.0.0.0.3
<<Potential Causes>> Evaluation result is not used 5.0.0.0.4
<<Potential Causes>> Lack of sufficient data to do proper evaluation 5.0.0.0.5
<<Potential Causes>> Assessment is done by trained individual but they do not apply procedure8 correctly 5.0.0.0.6
<<Potential Causes>> Lack of standard operating procedure 5.0.0.0.7
<<Potential Causes>> Evaluation criteria not well defined 5.0.0.0.8
<<Potential Causes>> Lack of training on procedure 5.0.0.0.9
<<Potential Causes>> Lack of sufficient time to do proper evaluation 5.0.0.0.10
<<Countermeasure>> Experienced staff should handle task 5.1
<<Countermeasure>> Senior Management should review “Qualify Opportunity" decision 5.2
<<Countermeasure>> Opportunity qualification shall be standardized by conducting “Leads scoring” 5.3
<<Countermeasure>> Standard operating procedure shall be used to guide the qualification process 5.4
<<Countermeasure>> Staff should be trained 5.5

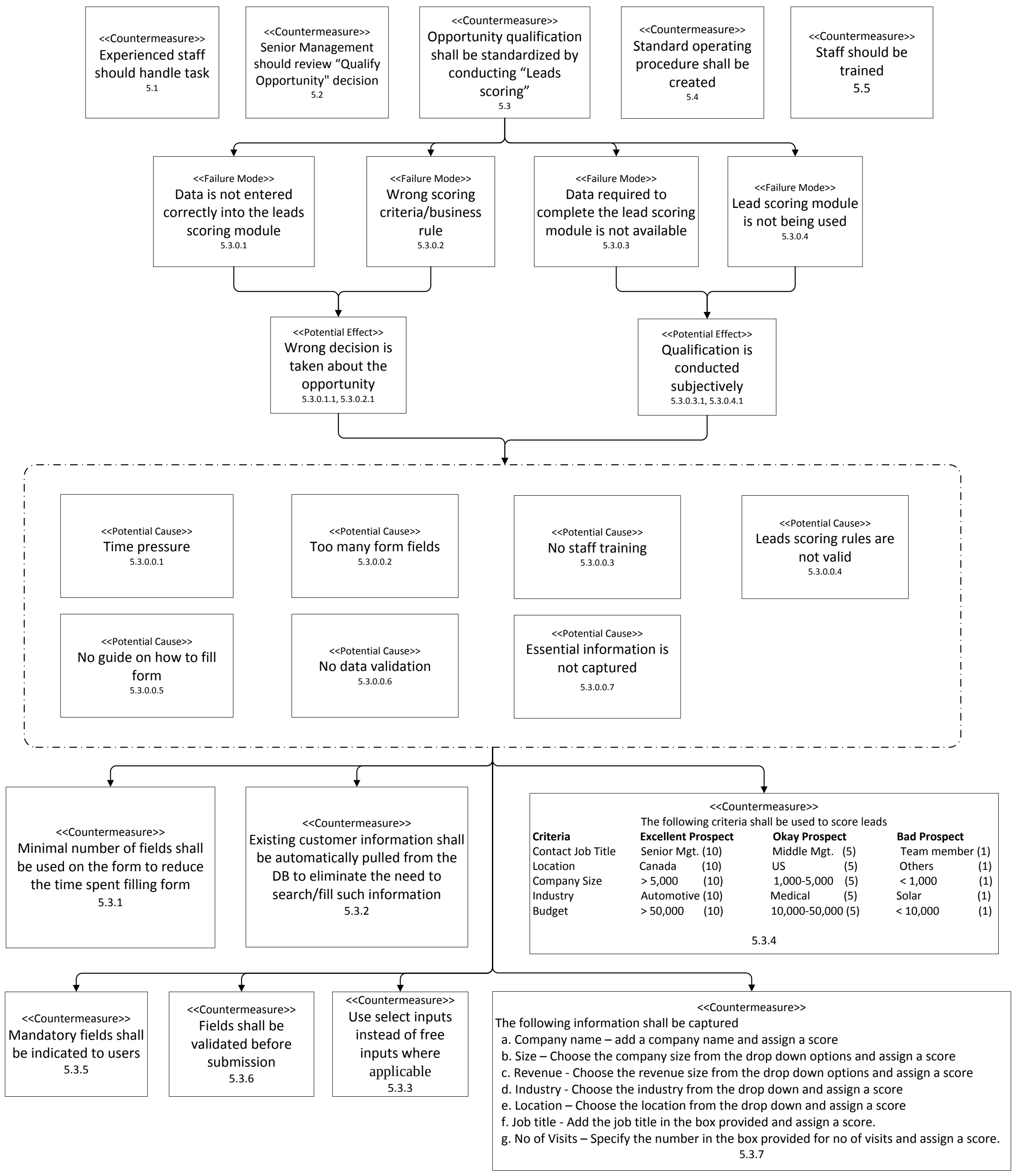
Figure 15: Countermeasures

Step 7: If countermeasure is not a product level requirement, go to Step 3 and iterate

In Figure 15, each of the elicited countermeasures describes *user level requirements*, including user requirements (e.g., opportunity qualification shall be standardized by conducting “Leads scoring” and standard operating procedure shall be used to guide the qualification process) and business rules (e.g., experienced staff should handle task, senior management should review “Qualify opportunity decision). User level requirements tells *what* should be done, however, we must derive the product level requirement that will specify *how* the user level requirements will be achieved. This involves determining the functional requirements, non-functional requirements, data requirements, external interfaces requirement and other constraints.

To derive the product level requirements, we will return to step 3 as stated in section 3.3 to determine the failure modes, effects of failure, causes and the appropriate countermeasures, except that this time we will apply that activity to each countermeasure, instead of each process component. The process will be repeated on the countermeasure derived in each iteration until we elicit explicit requirements that specify how the product needs to be put together to satisfy the business needs.

The outcome of the second iteration is shown in Figure 16.



<<Countermeasure>> Experienced staff should handle task 5.1
<<Countermeasure>> Senior Management should review "Qualify Opportunity" decision 5.2
<<Countermeasure>> Opportunity qualification shall be standardized by conducting "Leads scoring" 5.3
<<Countermeasure>> Standard operating procedure shall be created 5.4
<<Countermeasure>> Staff should be trained 5.5
<<Failure Mode>> Data is not entered correctly into the leads scoring module 5.3.0.1
<<Failure Mode>> Wrong scoring criteria/business rule 5.3.0.2
<<Failure Mode>> Data required to complete the lead scoring module is not available 5.3.0.3
<<Failure Mode>> Lead scoring module is not being used 5.3.0.4
<<Potential Effect>> Wrong decision is taken about the opportunity 5.3.0.1.1, 5.3.0.2.1
<<Potential Effect>> Qualification is conducted subjectively 5.3.0.3.1, 5.3.0.4.1
<<Potential Cause>> Time pressure 5.3.0.0.1
<<Potential Cause>> Too many form fields 5.3.0.0.2
<<Potential Cause>> No staff training 5.3.0.0.3
<<Potential Cause>> Leads scoring rules are not valid 5.3.0.0.4
<<Potential Cause>> No guide on how to fill form 5.3.0.0.5
<<Potential Cause>> No data validation 5.3.0.0.6
<<Potential Cause>> Essential information is not captured 5.3.0.0.7
<<Countermeasure>> Minimal number of fields shall be used on the form to reduce the time spent filling form 5.3.1
<<Countermeasure>> Existing customer information shall be automatically pulled from the DB to eliminate the need to search/fill such information 5.3.2
<<Countermeasure>> Mandatory fields shall be indicated to users 5.3.5
<<Countermeasure>> Fields shall be validated before submission 5.3.6
<<Countermeasure>> Use select inputs instead of free inputs where applicable 5.3.3

<<Countermeasure>>
The following criteria shall be used to score leads

Criteria	**Excellent Prospect**	**Okay Prospect**	**Bad Prospect**
Contact Job Title	Senior Mgt. (10)	Middle Mgt. (5)	Team member (1)
Location	Canada (10)	US (5)	Others (1)
Company Size	> 5,000 (10)	1,000-5,000 (5)	< 1,000 (1)
Industry	Automotive (10)	Medical (5)	Solar (1)
Budget	> 50,000 (10)	10,000-50,000 (5)	< 10,000 (1)

5.3.4

<<Countermeasure>>
The following information shall be captured
a. Company name – add a company name and assign a score
b. Size – Choose the company size from the drop down options and assign a score
c. Revenue - Choose the revenue size from the drop down options and assign a score
d. Industry - Choose the industry from the drop down and assign a score
e. Location – Choose the location from the drop down and assign a score
f. Job title - Add the job title in the box provided and assign a score.
g. No of Visits – Specify the number in the box provided for no of visits and assign a score.
5.3.7

Figure 16: Second Iteration

A list of the requirements derived in the second iteration is shown in Table 4.

Requirement level	ID	Requirement
Business Goal		Sales initiation and qualification
Business level requirements	5.0	Qualify business opportunity
User level requirements	5.3	Opportunity qualification shall be standardized by conducting “Leads scoring”
Product level requirements	5.3.1	Minimal number of fields shall be used on the form to reduce the time spent filling form
	5.3.2	Existing customer information shall be automatically pulled from the DB to eliminate the need to search/fill such information
	5.3.3	Required fields shall be indicated to users
	5.3.4	Fields shall be validated before submission
	5.3.5	Use select inputs instead of free inputs where applicable
	5.3.6	The following criteria shall be used to score leads

Criteria	Excellent Prospect		Okay Prospect		Bad Prospect	
Contact Job Title	Senior Mgt.	10	Middle Mgt.	5	Team member	1
Location	Canada	10	US	5	Others	1
Company Size	> 5,000	10	1,000-5,000	5	< 1,000	1
Industry	Automotive	10	Medical	5	Solar	1
Budget	> 50,000	10	10,000-50,000	5	< 10,000	1

Requirement level	ID	Requirement
	5.3.7	The following information shall be captured a. Company name – Add a company name and assign a score b. Size – Choose the company size from the drop down and assign a score c. Revenue - Choose the revenue size from the drop down and assign a score d. Industry – Choose the industry from the drop down and assign a score e. Location – Choose the location from the drop down and assign a score f. Job title - Add the job title in the box provided and assign a score g. No of Visits – Specify the no of visits and assign a score

Table 4: Requirements derived

Step 8: Determine the Severity, Likelihood, Risk reduction and CPN scores

Severity, Likelihood and Risk reduction rankings are made of the effects, causes and countermeasures respectively. Countermeasure Priority Number, which is the product of the severity, likelihood and risk reduction ratings, is calculated as shown on page 42. CPN shows the relative likelihood of a failure mode with a particular countermeasure: the higher number, the higher the failure mode. From CPN, a critical summary can be drawn up to highlight the areas where action is most needed (Hekmatpanah, Shahin, & Ravichandran, 2011).

Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) or Mechanism(s) of Failure	Likelihood	Countermeasure	Risk reduction	CPN
Invalid opportunity assessment	The business opportunity is lost	5	Evaluation criteria not well defined	4	Create leads scoring module - Standardize the qualification criteria	2	40
	Resources are committed to an invalid opportunity	3	Evaluation criteria not evaluated for opportunity	5	Create standard operating procedure	3	45
	Fail to properly identify opportunities	2	Assessment is not done by trained individual	2	Staff training	1	4
			Assessment is done by trained individual but they do not apply procedure correctly	1	Create standard operating procedure	2	4
			Evaluation result is not used	4	Create standard operating procedure	2	16
			Lack of sufficient data to do proper evaluation	4	Create standard operating procedure	4	32
			Lack of sufficient time to do proper evaluation	2	Create standard operating procedure	4	16
Evaluation result is not used	Fail to properly identify opportunities	3	Lack of standard operating procedure	5	Create standard operating procedure	1	15
	"Bad" opportunity is accepted	5	Lack of adherence to the standard operating procedure	2	Staff training	2	20
			Lack of training on procedure	5	Staff training	1	25
Opportunity is not qualified	Fail to properly identify opportunities	4	Lack of standard operating procedure	5	Create standard operating procedure	1	20
	"Bad" opportunity is accepted	5	Lack of adherence to the standard operating procedure	2	Staff training	2	20
			Lack of training on procedure	5	Staff training	1	25

Table 5: CPN Table

3.4 Definition of FBREM method

FBREM is a risk-driven method for eliciting and specifying unambiguous, consistent, traceable and testable requirements from broad, high level business goals. The FBREM method aims at:

- Systematically eliciting requirements by having the project team identify risks and countermeasures
- Progressively refining the business level requirements to derive all necessary information that is required to implement the best design
- Presenting the requirements in a format that is understandable to both decision makers who require information to help justify their decision, and to implementers who require specific implementation details
- Providing a means of evaluating requirements in order to assess the impact that the requirements, if implemented, might have on the business goals

The method consists of four main phases:

Business process modelling This phase involves abstracting the functioning of the business process into a model. Weske (Weske, 2007) expounds that a

> *Business process model consists of a set of activity models and execution constraints between them. A business process instance represents a concrete case in the operational business of a company, consisting of activity instances. Each business process model acts as a blueprint for a set of business process instances, and each activity model acts as a blueprint for a set of activity instances. These activities jointly realize a business goal.*

Modeling the real system is a basic step in the identification and understanding of the important elements of the system. In this thesis, an additional goal of this stage is to identify the behaviors of the elements making up the processes, and to elicit the requirements implicit in elements of the business process.

The main artifact of this stage is a business model, which is a graphical representation of the inputs, outputs, tasks, events, decision, flow of logic and roles involved within the business process.

Failure modes and requirements generation This phase analyzes the process model in order to determine the features and attributes that enable the components to achieve the desired business goal.

This phase is conducted by eliciting the various ways the components can fail to achieve their desired goal, and then determining countermeasures to those failure modes. The countermeasures are the requirements that can detect, prevent or mitigate the failure modes, thereby constraining the system or process to produce the desired outcome. Failure mode and effect analysis (FMEA) is used to determine the risk factors associated with the business goal and to ascertain possible countermeasures to mitigate the risk.

This phase begins by applying FMEA on the process components to determine appropriate countermeasures. FMEA can then be recursively applied to the requirements (countermeasures) generated in each iteration to further decompose the requirements to derive product level requirements that are verifiable and testable.

The main artifact of this stage is a set of requirements specifications. Analytic Hierarchy Process (AHP), a decision-making method based on the division of problem spaces into hierarchies, is used to visually represent the failure modes and the requirements generation process (Saaty, 1990). The tree-like structure of the AHP representation provides a means of visually connecting each level of the FMEA decomposition in a way that supports the rationalization of requirements and design. The rationale behind every design decision can be traced through the various levels of requirements and up to the business goals in a structured hierarchical way.

Requirement prioritization The aim of this phase is to prioritize the requirements generated in the previous phase. FBREM leverages the systematic and semi-quantitative nature of FMEA to derive quantitative estimates of the severity of failure modes, the likelihood of the failure, and the risk reduction implicit in the countermeasure. The values are multiplied together to derive the countermeasure priority number (CPN) for each countermeasure. Conventionally, CPN is an indication of the priority that should be given to the failure mode, that is, more effort should be put into mitigating failure modes with higher CPNs. We have however, adopted the CPN as a prioritization metric for determining which of the possible countermeasures will have the most impact on the business goal. Also, because multiple countermeasures can be elicited for a single failure mode, the CPN value can be used as a guide to prioritize requirements based on the resource and time available to implement them.

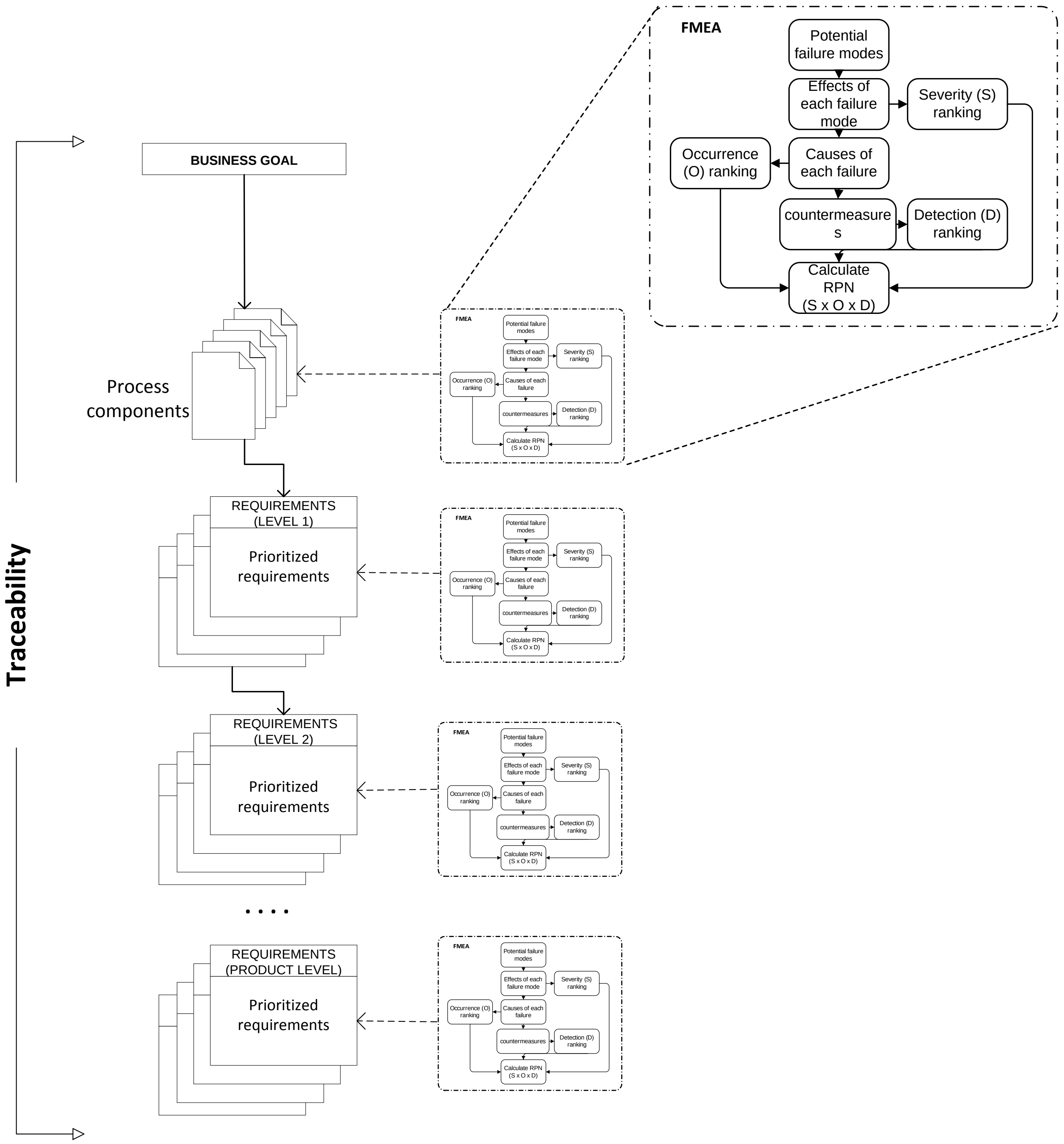


Figure 17: FBREM Workflow

Severity, likelihood, risk reduction and CPNs determined are used to prioritize the requirements derived in the previous stage. The main artifacts of this phase are a selected set of countermeasures that have the lowest CPN value.

Traceability The aim of this phase is to establish relationships between requirements and design artifacts for the purpose of demonstrating decision rationale, and to structure decomposed business goals into product level requirements. Gotel et. al. state that traceability is the ability to follow the life of a requirement in a forward and backward direction (Gotel & Finkelstein, 1994). While the traceability component of FBREM provides a means of justifying the design decisions made based on the requirements elicited, it also provides a means of establishing interdependencies between the requirements and the business goal. Requirement traceability provide support for impact analysis, change management, verification and validation processes. The main artifact of this phase is a traceability schema that captures the relationship which establishes the alignment of design, decisions and requirements with business goals. The traceability of FBREM is shown in Figure 17.

Chapter 4
Discussion

The case study led us to the FBREM method described in Chapter 3. From the small example studied in Chapter 3, FBREM shows promise as a method for eliciting requirements from agile artifacts such as process components. In this section, we review the motivation for using risk as a basis for eliciting requirements. We also look in more detail at three aspects of requirements elicitation in which FBREM provides important advantages: prioritization, rationale and traceability.

4.1 Why emphasize risk?

FBREM is a risk-based method for eliciting requirements. Why should risk be a good basis for this task?

4.1.1 Case study observations

Our attention was first drawn to risk as a method for eliciting requirements by analysis of the MACE case study. When we began this study, we did not have a particular interest in risk, and were only planning to observe the emergence of requirements in what we knew would be an agile methodology. However, requirements as such were never developed during the project, which proceeded with a typical agile process in which iterations occurred and users and developers collaborated on the design. We noticed with interest that at three points during the case study, MACE representatives became concerned about the project and made definite statements about what "must" be done:

1. At the very start of the project, they insisted that the project must result in the selection of some software tool
2. At the first iteration, they insisted that the process map must look the same as the one they had drafted internally
3. At various stages of the project they insisted that the tool must link to their existing ISO documentation

MACE representatives gave overriding importance to these three issues, and did so in a manner[11] that left no doubt in our minds that they were of the opinion that the project would be a failure if any of the above had not been satisfied. In fact, each of these three issues was seen as a serious risk by the MACE representatives.

Failure to address requirement 1 meant to MACE that they would have no automation for the process model, and therefore the process model would fail to be adopted by their company because it would be too unwieldy to use manually. In FMEA terms, the failure mode they identified was lack of adoption of the process model; the effect would be chaotic processes; and the cause would be lack of software to enforce the process model. Thus, MACE insisted on automation as a countermeasure.

Failure to address requirement 2 meant to MACE that the process map would look different from what the company had developed over the previous year, and therefore participants would think their effort had been a waste of time. This risk was considered so high by MACE that they did not want to engage in any process improvement no matter how beneficial, since that would result in a process map that looked substantially different, and therefore incur the risk of rejection. In FMEA terms, the failure mode they identified was lack of adoption of the process model; the effect would be chaotic processes; and the cause would be a new and unfamiliar process map. Thus, MACE insisted on similarity as a countermeasure.

Failure to address requirement 3 meant to MACE that the process map would either not direct users to existing ISO documentation, or else that screenshots and excerpts from the ISO documentation would have to be included in the software as duplicates. In the first case, the failure mode would be that personnel would not use existing ISO documentation; the effect would be possible loss of ISO certification (if auditors discovered that ISO documentation was not used); and the cause would be lack of connection between the process map and the ISO documentation. In the second case, the failure mode would occur when ISO documentation was updated and changes were not made to the process map; the effect would be failure to follow current ISO documentation; and the cause would be discrepancies between the ISO documentation in its “home” location and the copies in the process map, resulting from lack of updates to all copies. Thus, MACE insisted on linking as a countermeasure.

[11] They used emphatic language and expressions to emphasize the importance of these points.

MACE representatives did not formally outline these risks to us as part of our discussion, and they certainly did not engage in explicit FMEA. They simply stated very firmly what kind of implementation they wanted and suggested to us the effects they wanted to avoid and their beliefs about how the implementation would counter those effects. The requirements were implicit in their statement of necessary implementation, but un-elicited by us.

From an agile perspective, once users have agreed to an implementation, and it passes its tests, then development is successfully completed. But this approach can easily bypass the process of evaluating other possible options[12]. For example, in the MACE study, requirement 1 could have also been met by incorporating a process map in their current ERP system, without new software; requirement 2 could have been met by reviewing a modified process map with the original stakeholders and obtaining their agreement on the modified map; and requirement 3 could have been met by having an overall index that would lead users to both the right phase of the process map and the existing ISO documentation. These options received little consideration during the case study. The option of including the process map in the ERP system was rejected because it was felt that the process map needed to be modifiable by end users, while the ERP system was not (or in risk-based terms, the use of ERP would introduce a risk of inflexibility, and the countermeasure was not to use ERP). At the time of the case study the ERP system was undergoing a redesign, so it is at least theoretically possible that a user-modifiable process map system could have been made part of the ERP system. A requirements process would have kept this possibility open, whereas an agile process closed it off because of early decisions about what could be implemented.

Our point here is not that the implementation was non-optimal, or that the MACE representatives should have specified requirements and not implementations. Our point is to observe that the key requirements for this system were all strongly grounded in risk assessment. In fact, the entire effort of the MACE process map (of which our case study was only a small part) was based on MACE's implicit risk assessment of their business expansion plans: although informal and word-of-mouth use of their processes was sufficient when all work was conducted by long-term employees at the home location, a large risk was perceived in the planned expansion of the business to new employees at two new geographically widespread locations. These new locations and employees

[12] A standard agile philosophy is to deliver working software with the minimum amount of work; this stance can easily lead to accepting the first option that is expected to work.

would be much less likely to use standard processes, and that was perceived to put the whole business at risk. Hence, it was essential to create a process map as a countermeasure to this risk.

Thus, our case study drew our attention to the perception of risk and the development of countermeasures as an important facet of requirements elicitation.

4.1.2 Historical perspective

A notable instance of the long history of the relationship between setting objectives, assessing risk and decision-making was portrayed in the Thucydides's's[13] account of the eulogy given by Pericles[14] to honor Athenians killed in the Great Peloponnesian War (Spielvogel, 2014):

> We Athenians, in our persons, take our decisions on policy and submit them to proper discussion…the worst thing is to rush into action before consequences have been properly debated. And this is another point where we differ from other people. We are capable at the same time of taking risks and assessing them beforehand. Others are brave out of ignorance; and when they stop to think, they begin to fear. But the man who can most truly be accounted brave is he who best knows the meaning of what is sweet in life, and what is terrible, and he then goes out undeterred to meet what is to come.

These words are a profound expression of value placed on risk assessment in the decision-making process.

The subject of risk and the knowledge gained from conducting risk-related analysis has been an area of interest in both academic and professional circles. Peter Bernstein in his book *Against the Gods: The Remarkable Story of Risk*, highlighted several remarkable stories of how an understanding of risk, defining what may happen in the future, and choosing among alternatives has become one of the drivers of modern society (Bernstein, 1997). He describes how in 1952, future Nobel Prize winning economist Harry Markowitz, then a young graduate student studying operations research at the University of Chicago, devised modern portfolio theory. Markowitz demonstrated mathematically why putting all your eggs in one basket is an unacceptably risky strategy and why diversification is the investor's best option. His theory also demonstrated how no additional expected return can be

[13] Thucydides (460 – c. 395 BC) was an Athenian historian, political philosopher and general. He survived the war that killed Pericles.

[14] Pericles (495 – 429 BC) was a prominent and influential Greek statesman, orator and general of Athens.

gained without increasing the risk of the portfolio. Another example is Daniel Bernoulli (Bernoulli, 1954) whose work *Exposition of a New Theory on the Measurement of Risk* was the foundation of the theory of risk aversion, a systematic process by which most people make choices and reach decisions. His theory explained why some gamblers prefer a sure outcome even though it has a lower expected value, while others who are less risk-averse would make riskier choices in hope of a higher expected value.

4.1.3 Defining risk

There is no one universally accepted definition of risk. Some of the definitions pertinent to this thesis are listed here:

- Garvey defines risk as an event that, if it occurs, adversely affects the ability of an engineering project to achieve its objectives (Garvey, 2008). This definition asserts two important concepts associated with risk: its occurrence probability and its impact (or consequence) to the system
- Modarres defines risk (or potential loss) as associated with the exposure of the recipient to a threat, and can be expressed as a combination of the probability or frequency of the threat and its consequences (Modarres, 2006). Modarres's definitation supports Garvey's, but with emphasis on the recipient, which could be a system, project, objective or persons
- Rosa defines risk as a situation or event where something of human value (including humans themselves) has been put at stake and where the outcome is uncertain (Rosa, 1998). Consequently, for a situation to be termed risky something of value must be at stake and the certainty of whatever is at stake must be of a probability value between 0 and 1
- Alwang et al. characterize risk by a known or unknown probability distribution of events (Alwang, Siegel, & Jorgensen, 2001). These events have magnitude (including size and spread), frequency and duration, and history. This definition included the time component; Alwang et al. thereby note that the immediacy or the span of time of the effect of the hazard is an important factor in the risk. Time transforms risk, and the nature of risk is shaped by the time horizon: the future is the playing field (Bernstein, 1997). For example, the risk of not finding survivors of a missing airplane increases with time

- An interesting quantitative definition is given by Kaplan & Garrick (Kaplan & Garrick, 1981). They defined risk as the answer to the following three questions:

  (i) What can happen? (i.e., what can go wrong?)

  (ii) How likely is it that it will happen? (i.e., what can go wrong?)

  (iii) If it does happen, what are the consequences?

- ISO 31000 acknowledges that we operate in an uncertain world: risk is defined as the “effect of uncertainty on objectives” which can result in a positive or negative deviation from the expected. In order to achieve objectives, risk has to be reduced to the minimum (ISO 31000, 2009)

4.1.4 Risk Assessment

Risk assessment is the process of identifying and dealing with risks that could potentially prevent the achievement of an objective.   A useful chart to demonstrate the risk assessment process is shown in Figure 18 (NORSOK, 2001).

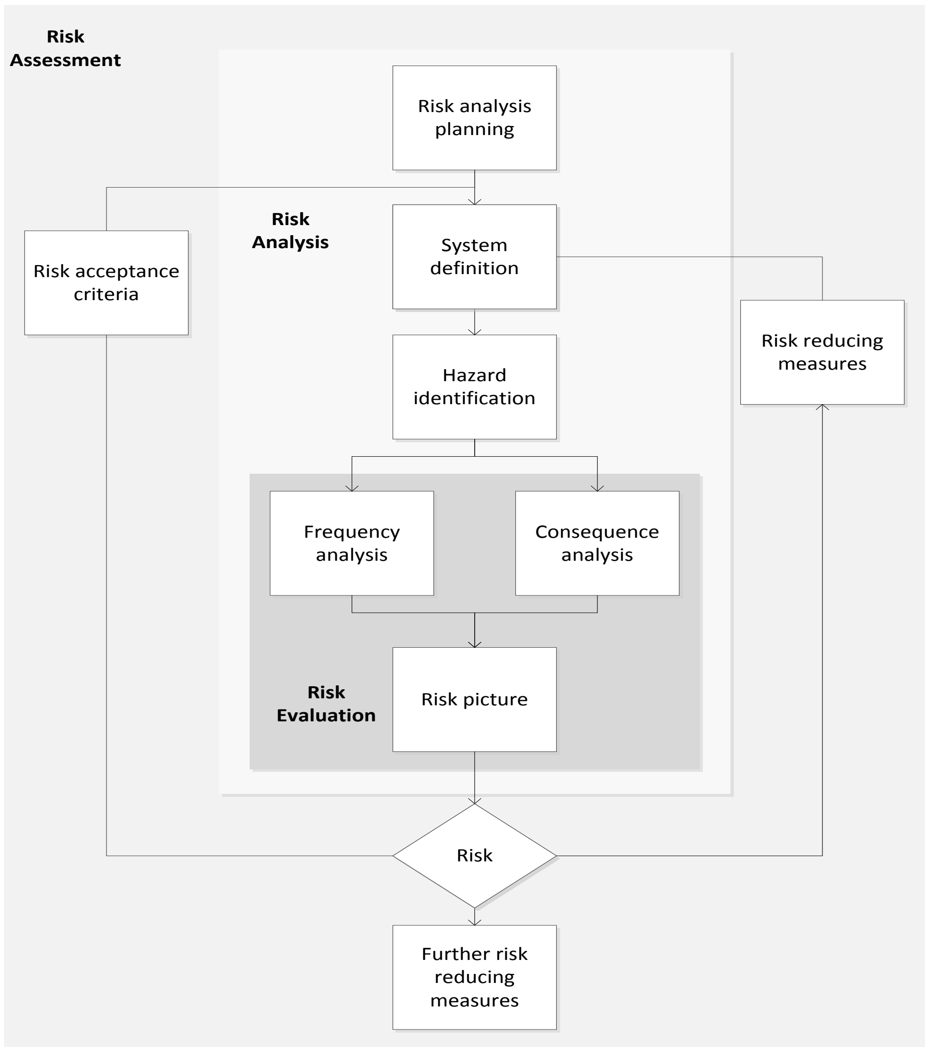


Figure 18: Risk assessment process

The general process of risk assessment has been well described by several authors (Berg, 2010), (Modarres, 2006), (Aven, 2008). The following are the key steps:

- **Establish the goal and context**: understand the objective, constraints and environment of the entity involved in the risk scenario. This stage also includes understanding the risk tolerance level of the customer.
- **Identify hazards**: identify the undesirable events that may adversely alter the identified objective. Process documentation, interviews, audit reports are some of the sources of hazard information

- **Analyze risk**: estimate the likelihood and the consequence of each undesired risk event. Existing measures put in place to control risk are also analyzed to determine their effectiveness. Risk analysis can be conducted qualitatively using simple methods such as brainstorming, or quantitative methods such as Fault Tree Analysis (FTA), or methods with a blend of quantitative and qualitative aspects, such as the Failure Mode and Effects Analysis (FMEA)
- **Evaluate risk**: compare risk information with the pre-defined risk tolerances to ascertain the acceptability of the risk involved. Youssef et al. argued that beside traditional risk evaluation factors, that is, probabilty and severity, there are other factors that should not be overlooked in risk evaluation (Youssef & Hyman, 2010). Such factors include
    - Detectability–The ability to detect the hazard before loss occurs. This is because the better the controls in place are able to detect the chances of the risk occurring, the less likely the potential loss will happen
    - Correctability–The relative ease of eliminating or mitigating a certain risk. A highly detectable risk with a low correctability can still result in a severe loss. In this regard, technical practicability and economic feasibility will be factors to consider in determining the correctabilty of a risk
    - Product utility–This factor implies integrating benefit into risk. This involves weighing the benefit derived from having a feature or undertaking an enterprise against the possible loss that could be encountered. If the estimated benefit outweighs the risk, then the risk may be acceptable rather than expending resources otherwise. The process of estimating benefit can be challenging and overstating benefit is a possible pitfall
- **Reduce risk**: Based on the results of the evaluation, measures are to be taken to reduce the likelihood (or consequence) of the risk depending on the resources available and the risk tolerance level

It can be seen that FBREM follows this standard pattern:

- Goal and context are established by the agile process artifacts
- Hazard identification is done through failure modes

- Risk analysis is done through severity and frequency analysis
- Risk evaluation and risk reduction is done through identification of countermeasures and estimating their correctability

Risk analysis in FBREM follows the industry-standard Failure Modes and Effects Analysis (FMEA). Based on a survey conducted by Carlson et. al., to determine the current important reliability practice in the industry in which over 450 reliability practitioners participated, FMEA was chosen both as the most important task in their reliability program, and the most important task practitioners think they should be doing in cases where they haven't started doing it (Carlson, Sarakakis, Groebel, & Mettas, 2010). FMEA enjoys wide application in a variety of industries and it forms an important aspect of various standards such as the US Department of Defense MIL-STD-1629A standards (DoD, 1980), International Electrotechnical Commission Standard, IEC 60812: 'Analysis Techniques for System Reliability—Procedure for Failure Mode and Effects Analysis (FMEA) (IEC, 2006)', British Standards Institution, BS 5760: 'Reliability of Systems, Equipment and Components' (BSI, 1991), International Organization for Standardization Technical Specification ISO/TS 16949:2009: 'Particular requirements for the application of ISO 9001:2008 for automotive production and relevant service part organizations'(ISO, 2009) and American Society of Quality (ASQ) Six Sigma Black Belt certification (ASQ, 2011).

FBREM differs from FMEA and the risk assessment shown in Figure 18 in that it is applied recursively. We adopted this approach from the Analytic Hierarchy Process (AHP) concept (Saaty, 1990). AHP hierarchically structures requirements at various levels of detail and specificity, thereby aiding the users of the model to focus on the specific level of information in which they are interested. Top-level goals can be decomposed into subcategories, and each subcategory can be further decomposed and analyzed independently, depending on the level of detail required. According to Saaty, each level may represent a different cut at the problem. Elements at each level can provide complimentary, competing or conflicting solution to the problem. David & Saaty state that using specific metrics, decision makers are able to measure the relative weight of requirements, their benefits, costs, risks and resource demands (David & Saaty, 2007)

4.1.5 Development methodology as risk reduction

Assessing risks of a new product or service is a common engineering activity, and the FMEA process is a technique with a long history. Assessing risks on specific projects (such as risks to schedule and cost) is also a common engineering activity in larger projects. But we suggest that project methodologies themselves are, to some extent, based on notions of risk assessment, and are designed to reduce what they view as core project risks. Consider the agile methodology:

- Since many waterfall projects fail to deliver software on schedule, agile delivers (minimal) software as soon as possible[15]
- Since users are frequently dissatisfied with the results of systems they have commissioned, agile requires that users work directly with software developers during the entire project and so get their comments early and often
- Since waterfall-delivered software is sometimes incomplete and buggy, agile puts testing ahead of software development

Similarly, waterfall methods can also be seen as tactics to avoid risk:

- Since errors in requirements can cause excessive rework downstream, waterfall methods put requirements elicitation first to reduce the risk of poor or unstated requirements
- Since documented requirements and design are important for maintenance, auditing, and updating of the software, waterfall methods reduce the risk of problems in those areas by requiring good documents
- Since change is a common vector for introducing bugs and other problems, waterfall methods involve formal change management to try to limit the introduction of bugs

Each methodology highlights specific ways that projects can fail—that is, failure modes—and the methodology contains countermeasure to those failure modes. From this view, the "best" methodology for software development is not one or the other; the answer can only be relative to the actual failure modes that are experienced (or avoided) in practice. If you are running a development team that is at risk for not delivering software, or frequently dissatisfies its users, or develops software that is incomplete and buggy, then perhaps agile has identified the risks and countermeasures for you. If your development team has a good delivery record, and users are

[15] Note that we are stating claims that agile proponents make about waterfall software development. We do not need to agree with these claims to make the observation that the claims are implicitly based on risk assessment and risk countermeasures.

satisfied with its systems, but you are concerned about reducing rework, change, and passing regulatory audits, then perhaps waterfall has best identified the risks and countermeasures for you.

4.2 Requirements Prioritization

There are several reasons to prioritize requirements. First, the requirements elicitation process usually produces more requirements than can or will be implemented. Second, requirements are derived from many viewpoints, each person introducing requirements that may be in conflict with others or able to serve as alternatives to one another. Third, solutions are implemented over a long period of time, necessitating the need to batch requirements into phases or releases.

Prioritizing requirements is the next logical task to be performed once requirements have been elicited (Ramzan, Jaffar, & Shahid, 2011). Prioritization helps to identify the most valuable requirements from the entire set by distinguishing the critical few from the trivial many (Berander & Andrews, 2005). By arranging the requirements in a prioritized order, it is easier to develop the system in a more realistic and structured form. Requirements can be prioritized to realize which subset can be delayed so that more urgent requirements can be implemented first; considering which requirements belong to earlier or later stages of the development cycle is frequently done in order to optimize one form of constraint or another (Ramzan et al., 2011). Ruhe et al. state that “The challenge is to select the ‘right’ requirements out of a given superset of candidate requirements so that all the different key interests, technical constraints and preferences of the critical stakeholders are fulfilled and the overall business value of the product is maximized” (Ruhe, Eberlein, & Pfahl, 2002). Karlsson and Ryan emphasize that requirements prioritization helps in making acceptable trade-offs among sometimes-conflicting goals such as quality, cost, and time-to-market. It can also benefit in quantifying the cost and schedule required to implement the elicited requirements (Karlsson & Ryan, 1997). A more comprehensive list of benefits to requirements prioritization compiled by Berander & Andrews (Berander & Andrews, 2005) and Gottesdiener (Gottesdiener, 2005) includes the following:

- To plan staged releases for incremental deliveries
- To decide on the core requirements for the system
- To balance the business benefit of each requirement against its cost
- To balance the implications of requirements on the software architecture and future evolution of the product, taking into account those associated costs

- To select a subset of the requirements that still produces a system that will satisfy the customer[16]
- To control scope creep
- To minimize rework and schedule slippage
- To handle contradictory requirements, focus the negotiation process, and resolve disagreements among stakeholders
- To establish the relative importance of each requirement and provide the greatest value at the lowest cost

We can see from the foregoing discussion that requirements prioritization is an essential part of requirements engineering.

4.2.1 Criteria used for prioritization

Various criteria can be considered in determining the priorities assigned to requirements. The common ones include *importance, cost*, *time* and *scope* (Berander & Andrews, 2005). Depending on the motivation for prioritization, one or more criteria can also be considered jointly in requirements prioritization. *Importance* is the criticality of the requirement in achieving the business goal. Requirements rated less critical can be accorded less resources or shifted to another development phase. *Cost* is the expense expected to be incurred in implementing the requirement. Cost can be expressed in terms of person-hours, capital expenditure, training needs, skill level or effort required to carry out the requirement. *Scope* is the amount of features/functions and nature of work required to be performed to deliver the stated requirements. Requirements with larger scopes can be implemented later in the process or moved to another phase of the project. Other criteria used for prioritization include the value placed on the requirement by the customer, the difficulty of implementation, economic benefit gained by implementing the requirement, the need to comply with regulatory demands, the ease of deployment, and provision of a competitive advantage (Gottesdiener, 2005).

The *risk* associated with implementing (or not implementing) a requirement can also be used as a criterion for prioritization. “Risk-based decision making is a process that organizes information about the possibility for one or more unwanted outcome and the impact of such unwanted outcome

[16] This aspect of prioritization is a prime focus of agile methods.

into a broad, orderly structure that helps decision makers make more informed choices" (Macesker, Myers, & Guthrie, 2002). A risk-based requirements prioritization approach involves identifying the potential failures that could occur if the requirement is not implemented, and then for each failure we identify the likelihood that the failure will occur, the impact or severity such a failure would have on the goal, and how we might detect such a failure. These risk factors can be combined into a priority for the requirement.

4.2.2 Requirements prioritization techniques

Several requirements prioritization methods have been described in the literature.

1. *100-Dollar Test* (Dean Leffingwell & Widrig, 2003) is a type of voting system where 100 imaginary dollars are given to participants to be divided among the requirements. This is a system that uses the metaphor of purchase as a way to make prioritization more concrete. For example, if there are five requirements to be prioritized and a participant allocates 20 dollars to each one, this indicates that all the requirements are equally important to that participant. This voting system can be performed by many people, and the average value assigned to each requirement can be used to prioritize the requirements.
2. Quality Function Deployment Matrix (Akao, 1994). This method involves organizing requirements into areas on a "House of Quality" matrix. The attributes from the matrix are then mapped to appropriate technical specifications and performance targets. Ultimately, specific elements of the mapped technical specification can be quantified and prioritized.
3. Wieger's Method (K. E. Wiegers, 2009). This method addresses prioritization from the customer's perspective. The value the customers place on each requirement is divided by the sum of the cost, risk and other trade-offs associated with that particular requirement. The ratio realized from this calculation is viewed as the rating of the requirement compared to the cost of implementing the requirement. This ratio is used to prioritize the requirements.
4. Numerical Assignment (Grouping) (Bradner, 1997). This class of methods involves grouping requirements into different priority groups. Sample grouping includes "mandatory", "desirable", and "unessential". Another usage of the grouping method involves using keywords such as "shall have" to denote critical requirements, "should have" to denote recommended requirements, and "may have" to denote optional requirements.

The methods described above offer means of prioritizing requirements, but they also have shortfalls. The 100-Dollar Test method is not suited for prioritizing a large number of requirements, because there are not enough dollars and participants' judgments become more questionable. For example, it is impractical to assign to use this method when requirements to be prioritized are in the thousands or even hundreds (Dean Leffingwell & Widrig, 2003). Similarly, the grouping method tends to constraint stakeholders to fix requirements into the available groups. Stakeholders may tend to put requirements that satisfy their interest in the "shall have" group, independent of their general value. A characteristic common to the methods discussed so far is that prioritization is done on a scale that promotes subjective rather than objective values. Because requirements are subjective, introducing a new requirement into a prioritized set may mean that the process of prioritization will have to be completely re-done (Herrmann & Paech, 2009).

Risk has been proposed as a method for prioritizing requirements by several writers (Berander & Andrews, 2005). Assessing the risk associated with requirements can help in estimating the benefit of each requirement and hence, prioritizing the requirements (Gottesdiener, 2005). When doing risk-based prioritization, we can avoid the two problems mentioned above: risk can be evaluated on an objective scale, and risk can be evaluated across a large set of requirements. The reason for this is that risk is evaluated independently for each requirement, while preference methods (such as the 100 dollar and/or grouping) tend to involve asking participants to look at the whole set of requirements at the same time.

FBREM provides a risk-based technique for prioritizing requirements based on an absolute risk value associated with each requirement. In FBREM, each requirement is quantified on the basis of the severity and likelihood of potential failures adversely impacting the goal if the requirement is not implemented, and the ability of the implementation to detect the failure before it occurs. Severity, likelihood and risk reduction are each rated on a scale of 1 to 5. Severity, likelihood and risk reduction ratings are multiplied together to derive the countermeasure priority number (CPN) for each failure mode associated with the requirement. CPN ranges from 1 to 125. CPN is a measure of the suitability of a countermeasure on three dimensions: the severity of the effect of a failure, the likelihood of the failure, and the likelihood that the countermeasure will prevent the failure, along a single dimension so that requirement can be prioritized and compared (Bowles, 2004). Specifically, CPN indicates how much the countermeasure, if not implemented, will adversely impact on the goal for which the failure mode was identified. Feather et al. describe values such as CPN as the indicator of “how much of a risk-reducing effect a requirement, should it be applied, has on reducing each risk (either by decreasing the risk’s likelihood, or by reducing the severity of the risk’s impacts on Requirements; the nature of the requirement dictates which kind of reduction takes place)” (Feather et al, 2006). CPNs can be ranked and used to prioritize the time and other resources that should be allocated to each of the countermeasures.

Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) or Mechanism(s) of Failure	Likelihood	Countermeasure	Risk Reduction	CPN
Data is not entered correctly into the leads scoring module	Unreliable lead score	4	Time pressure	3	Minimal number of fields shall be used on the form to reduce the time spent filling form	2	20
					Existing customer information shall be automatically pulled from the DB to eliminate the need to search/fill such information	4	48
			Too many form fields	2	Minimal number of fields shall be used on the form to reduce the time spent filling form	2	16
			Invalid data type input	3	Fields shall be validated before submission	5	60
					Select inputs shall be used instead of free inputs where possible	3	36
			Knowledge gap	1	Staff training shall be conducted	4	16
					User manual shall be provided	2	8
					Hints shall be provided for each form field	5	20

Table 6: Requirement Prioritization CPN Table for one Failure Mode

An extract of the requirements determined in our case study using FBREM is shown in Table 6. The table contains the countermeasures elicited with their respective CPN values. Using the CPN values as the basis for prioritization, the requirement “Fields shall be validated before submission” with the highest CPN value (60) is considered the countermeasure that will have the greatest impact in mitigating the risk posed by failure mode “Data is not entered correctly into the leads scoring module”. Put differently, this is the requirement that will have the most risk-reducing effect among the set of requirements elicited. The countermeasure “User manual shall be provided” is the requirement with the least risk-reducing effect because of its CPN value of 8.

In addition to using CPN to order the requirements according to their priority, CPN can be used to determine how to proceed with further requirements work. We may decide to further decompose requirements whose CPN value is greater than a given threshold into lower level requirements by conducting FMEA recursively on these countermeasures (as depicted in Figure 17), or we may decide not to implement any requirement with CPN lower than a certain minimum. For example, if the CPN value 50 is chosen as the threshold for further analysis and 10 is the minimum for implementation, then the requirement “Fields shall be validated before submission” will be further analyzed to determine its failure modes and subsequently elicit requirements for the failure mode, whereas the requirement “User manual shall be provided” will not be implemented at all.

4.3 Rationale

The rationale for a decision is the justification or reasoning behind that decision (Dutoit, McCall, Mistrik, & Paech, 2007). Burge et al. describe rationale as the expression of how decisions are made, what alternatives were considered before making the decision and what parameters were used in evaluating the alternatives. Rationale is the reason underlying decisions made and actions taken (J. E. Burge, Hall, & Brown, 2007). While requirements states the conditions or capabilities desired to produce an intended result—that is, the “what”—rationale explains “why” those requirements exist in the first place (Miller & Chavez, 2002). Leveson asserts that requirements are a set of instructions useful for implementers to create an intended solution, and that this necessitates that the stated requirements are correctly interpreted (Leveson, 2000). To ensure proper interpretation, requirements should be accompanied by their rationale. Rationale provides a bridge between formal and informal aspects of the requirements. Rationale provides the underlying ideas, assumptions, psychology and environmental basis for requirements. Simply specifying requirements without describing the rationale for those requirements does not provide much assistance to the implementers, because they do not know why the system should satisfy those requirements and therefore cannot easily evaluate whether their implementation embodies the rationale.

4.3.1 Uses of rationale

Several authors have suggested various uses of rationale.

Burge et al. suggest that rationale provides a means of actively shaping the process of reasoning about decisions and it serves as a record of the reasoning associated with those decisions (J. Burge, Carroll, McCall, & Mistrik, 2008). The rationale behind decisions taken in previous phases or projects can serves as valuable input in producing consistent, well thought-out requirements for subsequent activities. Rationale documentation also serves as memory aid. The reasoning underpinning design decisions can be easily forgotten in time, especially in large and complex projects (Tang, Babar, Gorton, & Han, 2006b). Documented rationale provides a resource database for querying the basis for decisions made in the past. The need to revisit previous decisions may arise when changes are to be made to existing products, when new systems are being acquired to interface with existing systems, or during quality processes such as validation.

Dutoit et al. highlight other benefits for documenting rationale in a requirement engineering process. Documented rationale provides support for communication during requirement elicitation and negotiation. The process of deciding which of the elicited requirements to implement becomes part of the requirement specification, since it involves communication between various stakeholders (Dutoit et al., 2007). Providing and documenting the justification supporting each of the requirements will aid the decision-making process and help to resolve conflicting requirements. It can also help in cases where there is need to probe a decision or requirement in further detail. Requirements reuse can also benefit from documented rationale. Requirements reuse is the ability to share a requirement across projects without unnecessary duplication of artifacts (Akers, 2008). Rationale provides additional information about requirements which helps to determine in what way a requirement is reusable and in what situation is it reusable.

Other benefits of rationale are described by (Leveson, 2000), (J. Burge et al., 2008), (Tang, Babar, Gorton, & Han, 2006a), (Miller & Chavez, 2002), (Dutoit et al., 2007):

- To improve management of dependencies among requirements
- Support for elicitation of downstream requirements
- Support for communication with management to justify project schedule and/or cost
- Support for prioritization of requirements by providing supplementary information
- Support for risk assessment and contingency planning
- To aid in the understanding of requirements by external stakeholders who may have little background knowledge about the requirement

- To aid in testing, audit and problem resolution activities
- To facilitate configuration management by making configuration options explicit
- To facilitate the operation, support and maintenance of the system
- To provide a record of decision alternatives and their evaluation to facilitate the redesign or refactoring of the system
- To assist in traceability of requirements by identifying the origin of systems features

Burge and Brown summarized the use for rationale as follows (J. E. Burge & Brown, 1998):

Design verification—to verify that the requirement meets the intent

Design evaluation—to assess requirement alternatives

Design maintenance—to determine what will be affected and needs to be taken care of if changes are to be made to the requirement

Design reuse—to determine the portion of the requirement that can be reused and how

Design education—to teach people who are unfamiliar with the system

Design communication—to facilitate communication and provide better insight into the decision-making process

Design assistance—to improve the requirements by considering such things as constraint/dependency checking

Design documentation—to present and preserve the knowledge acquired in the process of creating the requirement

4.3.2 Documenting rationale

Rationale documentation can be informal, formal or semi-formal. Informal documentation is easily created since it involves capturing requirement elicitation in raw form, using natural text, video or audio recording. This form is however difficult to process due to its lack of structure. Formal documentation is a structured form of documenting rationale; it involves the use of data types and data relationships. Formal documentation of rationale is captured in formats that can be computer processed and easily queried. Formal documentation involves substantially more effort than informal documentation. Semi-formal documentation combines the benefit of the other two methods: rationale is captured in a partially-structured format and is stored using natural language (Heindl & Biffl, 2006).

We next described some systems for capturing rationale.

Issue-Based Information System (IBIS)

The first method is the Issue-Based Information System or IBIS described by Kirschner et al. (Kirschner, Buckingham-Shum, & Carr, 2003). IBIS uses the following elements:

- *Issues*: the requirement item being considered, which is specified as a question
- *Positions*: answers to the issue
- *Arguments*: statements that supports or contradicts the position
- *Resolutions*: decisions made that document the rationale behind the requirements from different perspectives

Once these elements are decided for each requirement, an “issue-map” is created, which documents the relationships existing among the various elements of the representation. The elements are denoted as nodes on the issue-map. The business goal, referred to as the root issue, can be expanded into child issues, each with its own corresponding arguments and resolution. A web of relationships is then created among the rationale elements forming the rationale documentation. This method is semi-formal and graphical.

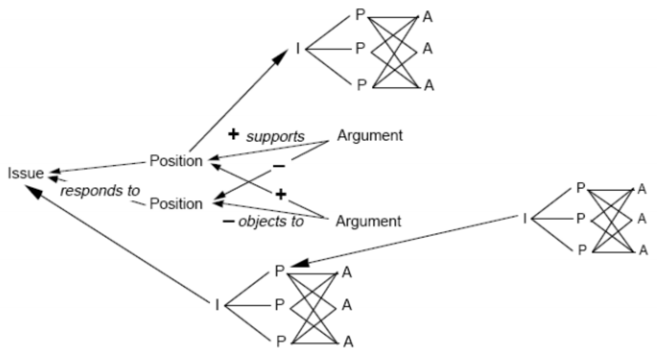


Figure 19: IBIS – Structure (Adhikari & Reinhart, 2006)

A sample root issue such as “How should the company’s intranet be implemented?” can lead to “Build in-house” and “Outsource” alternative positions. These positions can then be expended further to associate supporting arguments such as “company will have more control over the intranet implementation” or negating arguments such as “project stands the risk of being de-prioritized” to the “build in-house” position. The issue can be expended further as shown in Figure 20 to depict the rationale for the decisions taken.

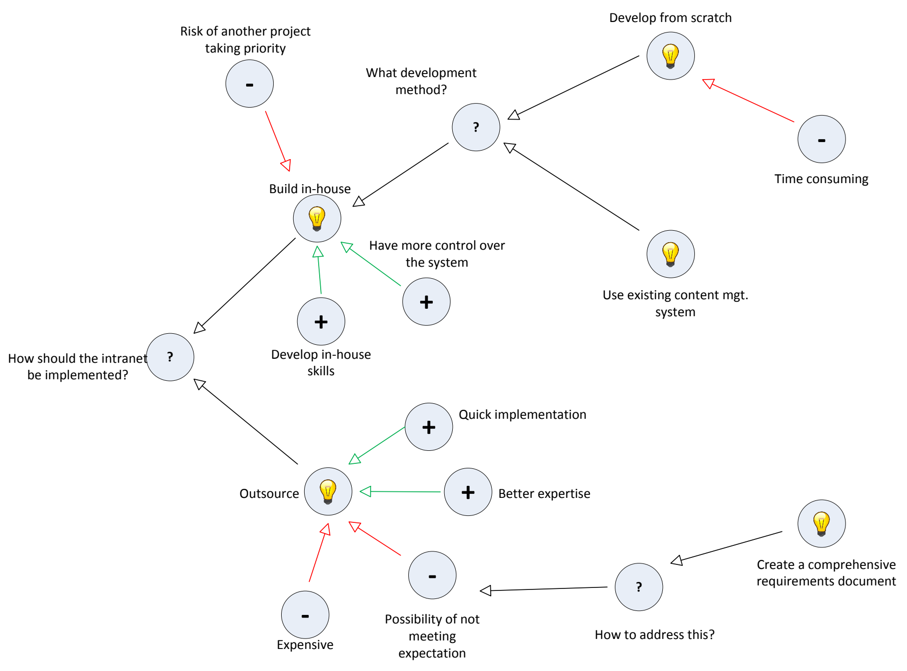

Risk of another project taking priority
Develop from scratch
What development method?
Time consuming
Build in-house
Have more control over the system
Use existing content mgt. system
How should the intranet be implemented?
Develop in-house skills
Quick implementation
Outsource
Better expertise
Create a comprehensive requirements document
How to address this?
Possibility of not meeting expectation
Expensive

Figure 20: Sample IBIS map

Decision Representation Language (DRL)

The second method is Decision Representation Language or DRL as described by Lee (J Lee, 1991). This method uses decision graphs to map the *issues* to be decided (decision problems), *alternatives* (way of addressing the issues), *goals* (the results the alternatives are set to achieve) and *claims* made about the outcomes of the goals, which can support or refute the goals. An additional element is *groups*, which describe any relationship existing among the elements of the model. DRL is a semi-formal rationale representation.

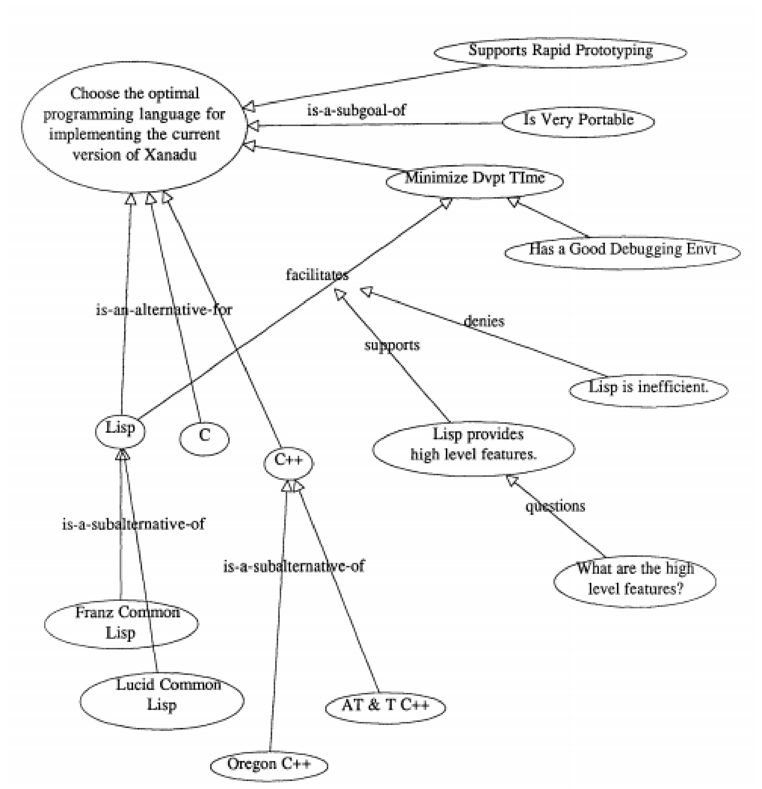


Figure 21: An example of DRL Decision Graph (Jintae Lee, 1989)

Unlike IBIS, which provides the arguments supporting and opposing an issue, DRL only provides positive arguments to support goals. This difference is significant in that claims evaluation in DRL may not be as effective, since we cannot consider arguments that actually inhibit the achievement of the goal. However, claims made in DRL have attributes, such as plausibility, degree (extent to which claim is true) and evaluation (function of plausibility and degree) (Stumpf, 1997). Using these attributes, DRL produces additional data for evaluating decisions, rather than just a method for exploring the design space and elaborating design rationales.

Device Modeling Environment (DME)

The third method is Device Modeling Environment or DME as described by Gruber (Gruber, 1990). DME is a formal and graphical representation of requirements that can be queried for rationale. A system to be developed is simulated in an environment similar to the production environment and is manipulated to produce the possible outcomes of various inputs. The results of each observed behavior and the reasons for such behaviors are stored as pre-enumerated set of rationale.

FBREM

FBREM, our method for eliciting requirements, is effectively a semi-formal method of representing rationales. Rationale is presented in hierarchical format such that the overarching business goal is presented at the top of the hierarchy, while the rationale supporting each decision made throughout the elicitation process can be traced as a response to the various failure modes.

Figure 22 shows the documentation of rationale from our Chapter 3 case study. The example traces the rationale of the requirements from the product level "Form shall be validated before submission" up to the process component "Sales Initiation and qualification".

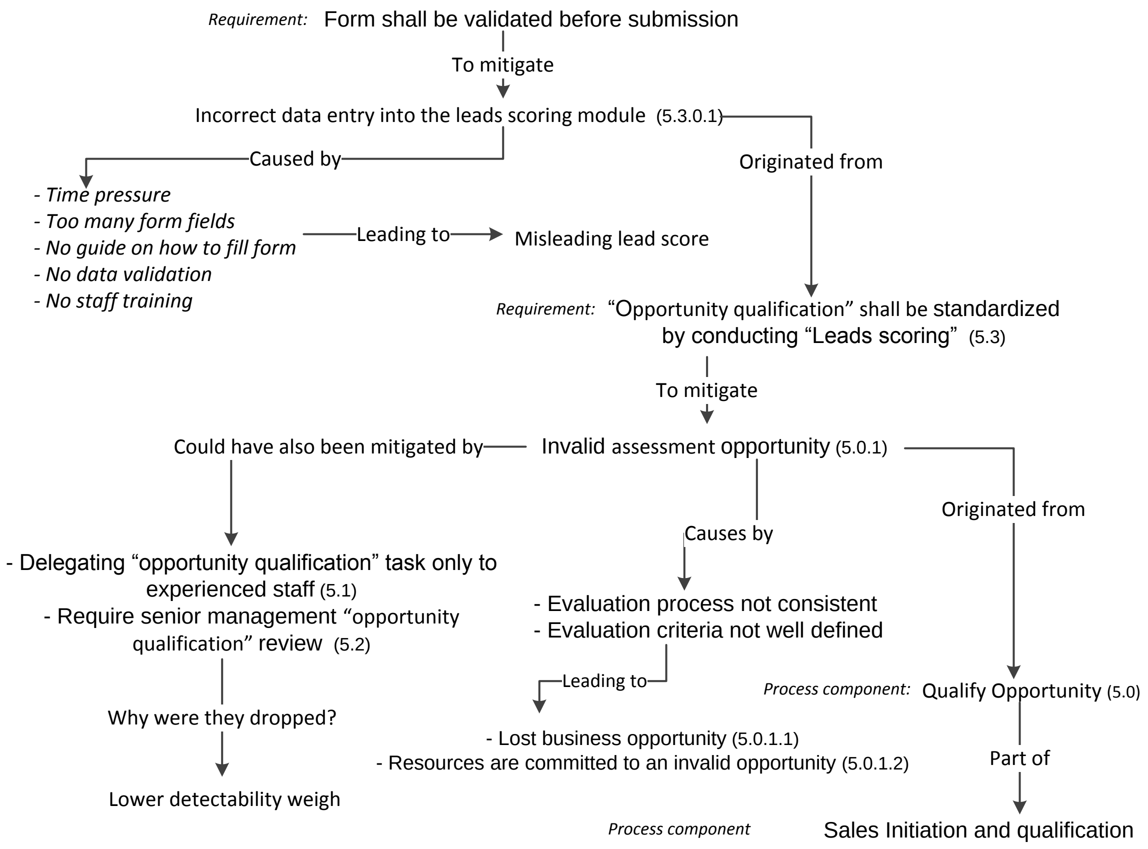

Requirement: Form shall be validated before submission
To mitigate
Incorrect data entry into the leads scoring module (5.3.0.1)
Caused by
Originated from
- Time pressure
- Too many form fields
- No guide on how to fill form
- No data validation
- No staff training
Leading to
Misleading lead score
Requirement: "Opportunity qualification" shall be standardized by conducting "Leads scoring" (5.3)
To mitigate
Could have also been mitigated by
Invalid assessment opportunity (5.0.1)
Originated from
Causes by
- Delegating "opportunity qualification" task only to experienced staff (5.1)
- Require senior management "opportunity qualification" review (5.2)
- Evaluation process not consistent
- Evaluation criteria not well defined
Leading to
Process component: Qualify Opportunity (5.0)
Why were they dropped?
- Lost business opportunity (5.0.1.1)
- Resources are committed to an invalid opportunity (5.0.1.2)
Part of
Lower detectability weigh
Process component
Sales Initiation and qualification

Figure 22: Rationale and traceability with FBREM sample

FBREM provides a structured approach for presenting the justification for each requirement by including extra information such as failure mode, effect, causes, and prioritization, thus describing the reasoning surrounding each requirement and justifying the choices that were made. In addition, other possibilities generated during the elicitation process that did not form part of the prioritized set of requirements are also presented, along with the justification for their elimination.

FBREM is useful both for prescriptive and descriptive reasoning purposes. For prescriptive purposes as it can be referenced in reasoning out new possibilities or updating existing requirements. It provides information about existing dependencies in the system and ways in which new requirements can support or conflict with existing requirements. It can be used in reflecting on the decisions taken and for determining alternative requirements. It can be useful for descriptive purposes as the reasoning behind the decisions made, which provide information for support and maintenance activities. Documented rationale can also be referenced when similar projects are carried out or when similar situation is experienced (J. Burge et al., 2008).

4.3.3 Rationale documentation barrier and FBREM

In this section we discuss some of the challenges in documentating rationale, and describe how the FBREM method addresses those challenges.

Rationale is usually either documented in passing, or else captured as a separate process outside the elicitation process. This causes contextual information related to the rationale to be lost, and may lead to misinterpretation of requirements, interpreting requirements out of context, or loss of valuable rationale information. Loss is particularly likely if the people who determined the requirements are not available later when the rationale needs review and the captured data becomes the only source of information (Dutoit et al., 2007). In cases where rationale is tacit knowledge (Kruchten, Capilla, & Dueas, 2009), that is, knowledge that not stated in explicit form (Dale, Siesfeld, & Cefola, 1998), rationale can be unintentionally omitted.

FBREM attempts to solve the challenge of lost rationale by providing a structured process that guides requirement elicitation, ensuring that processes are well-documented at the same time they are considered. Though FBREM may not completely eliminate the possibility of omitting rationale from the documentation, it structures the elicitation process so that the rationale is “automatically” obtained as the requirement elicitation is being carried out.

A second problem with documenting rationale is the retrieval problem. Some data is generated during requirements elicitation that does not end up as requirements, but serves as valuable input into rationale documentation. In many elicitation processes, these data are either not tracked or are not structured in any particular way, thereby making the rationale difficult to retrive. Dutoit et al.suggests indexing as a solution to this problem (Dutoit et al., 2007). Though indexing rationale documentation requires additional effort on the part of the designer, its benefit to implementers and reviewers can outweigh the cost (Gruber & Russell, 1996).

FBREM helps with retrieval since it is designed to ensure the logical sequencing of both the requirements and their rationale. Data created throughout the entire requirement elicitation process is indexed to facilitate fast and accurate retrieval of requirements and their rationale. FBREM also provides an overall structure of the reasoning process for easier reference. Using the index to trace through Figure 22, it can be seen that the requirement "Form shall be validated before submission (5.3.0.1)" originated from requirement "Opportunity qualification shall be standardized by conducting leads scoring (5.3)" which originated from process component "Qualify Opportunity (5.0)". Rationale can be traced in a similar way, by observing the labels on the arrows which specify the deductions.

4.4 Traceability

The concept of traceability is the aspect of requirement engineering concerned with showing the relationship of requirements to future activities in the software development process and past reasoning about the requirements. Traceability is a key component of a software validation process, where traceability of tests to designs and of designs to requirements forms the basis of the validation task. Requirements traceability is an important aspect of requirement engineering, as it is the way to associate the reasoning underlying the creation of an artifact with the artifact itself, as well in assessing the consequences and impact of change to requirements (Nuseibeh & Easterbrook, 2000). There are many definitions of traceability in the literature, each highlighting a different aspect of its importance:

- According to Wright, the term " requirements traceability" was framed by the US Department of Defense, and it is used to concisely communicate to vendors the need to "prove" that the requirements are understood, the product delivered fully complies with requirements, and that no unnecessary feature or functionality is added to the delivered product (Wright, 1991).

- One of the more commonly cited definition is that of Gotel and Finkelstein, who define traceability as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)" (Gotel & Finkelstein, 1994)
- IEEE 830-1998 defines requirements as traceable "if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation" (IEEE, 1998)
- Hull et al. suggest that traceability is "how" high-level requirements transform into low-level requirements (Hull, Jackson, & Dick, 2005)
- Murray & Griffiths defines traceability as the "ability to identify requirements at different levels of abstraction, and to show that they have been implemented and tested" (Murray & Griffiths, 2002). This definition emphasizes traceability across the various levels of requirements as a means demonstrating completion
- Ramesh et al's. definition states that "Requirements traceability is a characteristic of a system where requirements are linked to their sources and to the artifacts created during the system development lifecycle based on those requirements " (Ramesh, Stubbs, Powers, & Edwards, 1997)
- Spanoudakis describes traceability as "the ability to relate requirements specifications with other artifacts created in the development life-cycle of a software system" (Spanoudakis, 2002)
- Greenspan and McGowan define traceability as "The property of a system description technique that allows changes in one of the three system descriptions—requirements, specifications, implementation—to be traced to the corresponding portion of the other descriptions. The correspondence should be maintained through the lifetime of the systems" (Greenspan & McGowan, 1978)

These definitions generally agree, although there are two areas in which there is substantial difference: what is or should be traceable, and the orientation or direction of the traceability.

What is traceable: Spanoudakis and Ramesh both explicitly link traceability from requirements to other development artifacts. Greenspan and McGowan hint that traceability exists between requirements and artifacts, but they they limit the scope to include only requirements, specifications and implementation. Traceability can also be made to other artifacts such as test cases, user manual, defects records, etc. Traceability between requirements is known as *inter-requirements traceability*, while traceability between requirements and other artifacts is known as *extra-requirements traceability* (Pinheiro, 2004), as shown in Figure 23.

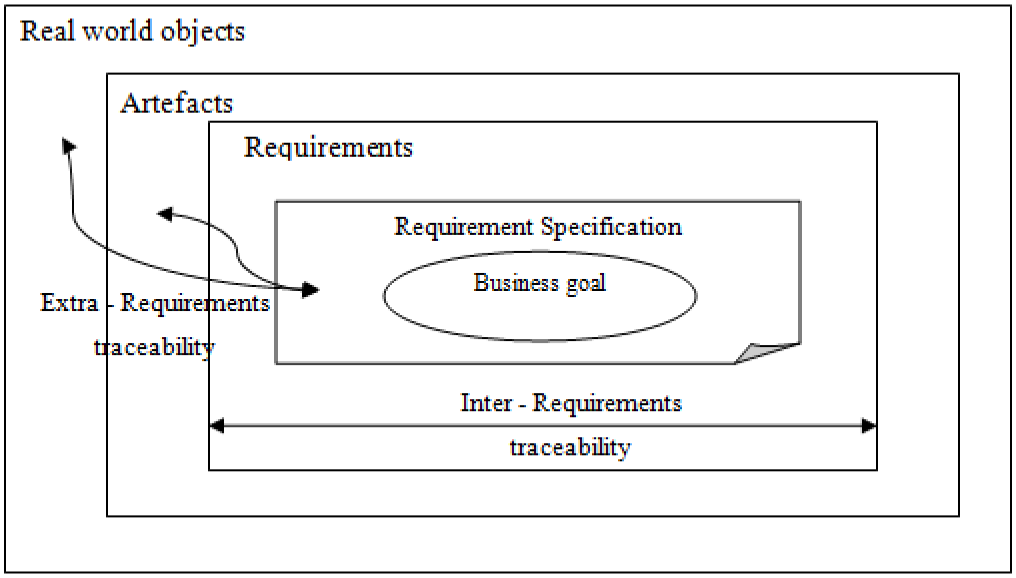

Real world objects
Artefacts
Requirements
Requirement Specification
Business goal
Extra - Requirements traceability
Inter - Requirements traceability

Figure 23: Extra- and inter-requirements traceability

The orientation of traceability: Some definitions consider traceability as being both "forward" and "backward", while others only consider one direction. *Forward traceability* refers to tracing from the source of the requirement (business goal, management direction, regulatory requirements, need for corrective and preventive actions) to requirements, the design elements which make up the implementation, the actual implementation, and tests of the implementation. *Backward traceability* on the other hand is used to trace tests, design, and other software development artifacts back to the source requirements. According to Westfall, forward traceability ensures that the evolving product is representative of the original intent (that we are building the right thing) and helps to ensure the completeness of software development activities (Westfall, 2006a). Backward traceability ensures that software development activities do not create additional elements that expand the scope of the project beyond the original scope. For example, if a test cannot be traced back to a design element, or a design element can not be traced back to a requirement (as shown in Figure 24), then we can question if the test is needed or if some features has been added along the way that should not be part of the system. Westfall describes backward traceability as helpful to ensure that we "built the product right" (Westfall, 2006a).

We can demonstrate forward and backward traceability in FBREM using our case study as an example. In Figure 22, the requirement 5.3 (*Opportunity qualification shall be standardized by conducting lead scoring)* was elicited from the higher level requirement "*Form shall be validated before submission*". If no subsequent requirement or design element would be traceable to requirement 5.3, then we would know the software design was incomplete.

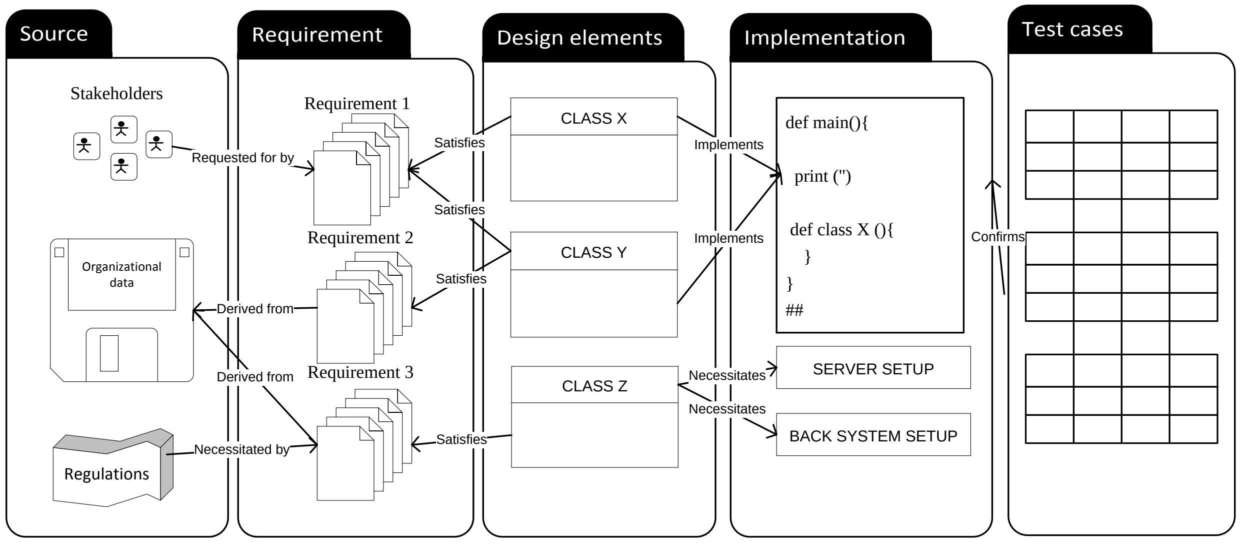


Figure 24: Forward and backward traceability

We summarize our observations as follows:

1. Traceability is about establishing relationships between layers of information
2. Relationships can exist within a single abstraction layers or across layers of abstraction
3. The traceability relationship can be forward from a source or backward to the source
4. Stakeholders may be interested in different layers and directions of a traceability relationship

4.4.1 Motivations for requirement traceability

Determining the requirements for an intended system and using the requirements to guide the development process is critical. It is important to track the changes that may arise as the elicitation process evolves and to ensure that the activities performed, as well as artifacts produced along the way, are identifiable and trackable. The literature is clear on the benefits of traceability (Galvao & Goknil, 2007), (Hull et al., 2005), (Bashir & Qadir, 2006), (Jaber, Sharif, & Liu, 2013):

For certification purposes Demonstration of traceability of requirements used in process and product development is a requirement for quality standards certification. "Maintain Bidirectional Traceability of Requirements" is a goal in theRequirements Management process area of CMMI (SEI, 2000). Similarly, traceability is an important quality requirement in ISO quality standards

To aid collaboration Traceability provides context and visibility to shared artifacts, which enhances stakeholder engagement and collaboration

To aid maintenance activities Traceability provides a means of documenting interrelated aspects of the system in a way that can be leveraged on in support and maintenance tasks

To aid audit activities Traceability provides guidance to auditors in knowing what the rules are and to what extent there is compliance. Traceability information helps auditors to trace the sources of data, check if data is being updated, how often and with what methods it is updated

For impact and change analysis "The ability to perform correct impact analysis of changes is often referred to as the most important motivation for establishing requirements traceability" (Turban, 2013). Impact analysis involves determining the consequence of change and how change can be successfully carried out without perturbing a stable system

To preserve memory Traceability helps to identify and organize background information, assumptions and justification for decisions taken for future reference

To verify completeness Traceability improves accountability in the development process since the expected result can be matched against the actual result to verify completeness

4.4.2 Traceability techniques

There are a number of techniques for managing traceability. Four of the techniques are briefly described here.

Constraint network   This technique establishes traceability between requirements and artifacts by explicitly capturing the constraining influences the requirements exert on each other and on other artifacts. A constraint specifies the relationships that must be satisfied between the components of the system for the requirements to be fulfilled. The network of such constraints is captured by the method (Bowen, O'Grady, & Smith, 1990).

Hypertext is an architectural framework for generating a glossary of links from the requirements to the artifacts, based on textual reference. The essential components of the technique are nodes and links (Bigelow, 1988). The requirements are stored in nodes, which link to the appropriate resource in a way that allows for the organization of data and explicit presentation of the dependencies between requirements and artifacts (Kaindl, 1993)

Traceability matrices This technique documents traceable relationships between pairs of the products of the development process. A typical example is the traceability matrix used in software validation that presents the relationship between design elements and test cases or requirements and design elements. A traceability matrix presents the basic relationship between elements without showing the detail of dependencies among the elements, or any complex relationships. It is usually presented in tabular or tree formats (ESA, 1994).

Cross references and indexing schemes Cross references and indexing schemes are "implemented as references made across several artifacts, to indicate links between them; or as lists of indices containing the related artifacts for each entry" (Pinheiro, 2004). Cross references can also be transformed and viewed as a traceability matrix. Like the traceability matrix, cross referencing is only used to represent the relationship between pairs. Hierarchical or extended dependencies cannot be shown with the cross-reference technique (Lauber, 1982).

4.4.3 FBREM as a traceability technique

FBREM supports the delineation of the hierarchical relationship existing between the various levels of a requirements elicitation process. As described earlier, business goals are taken through various levels of refinement, starting from breaking the business goal into process components and then progressively applying FMEA on each of the process components until product level requirements are determined. As the process is being conducted, as shown in Figure 26, the path through which the process occurs is automatically preserved within the FBREM framework. This becomes the requirement traceability technique through which the life of the requirement can be followed in both the forward and backward directions.

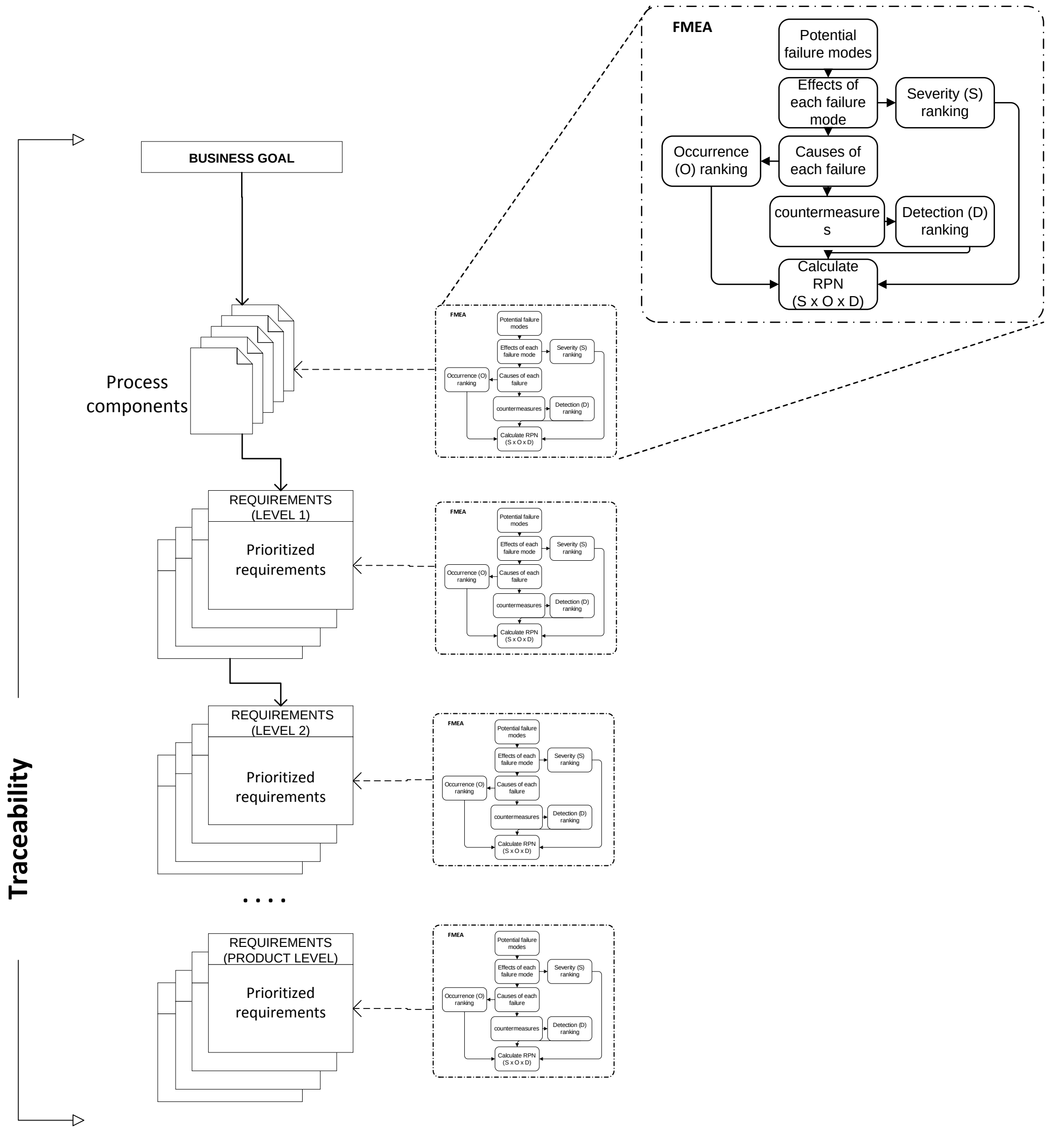
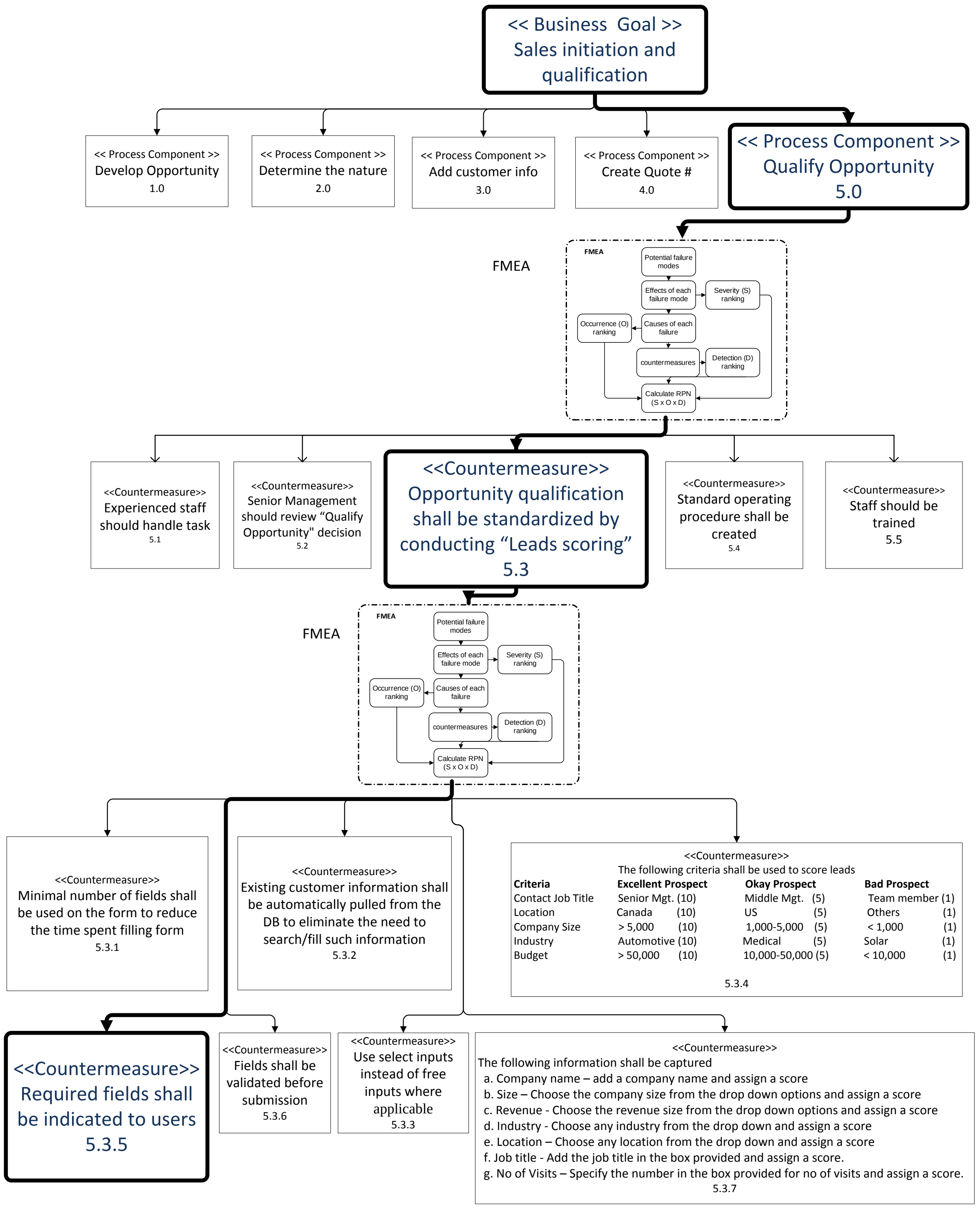


Figure 25: FBREM traceability

Other software development artifacts such as test cases and implemented modules have been included in the diagram to show that the trace produced using FBREM can be extended to other development activities.

Figure 26 is a stripped-down version of the rationale and traceability diagram depicted in Figure 22. Prioritization and rationale-related information described in sections 4.1 and 4.3 respectively are also created along elicitation process and they add additional information on the traceability relationship.



<< Business Goal >> Sales initiation and qualification
<< Process Component >> Develop Opportunity 1.0
<< Process Component >> Determine the nature 2.0
<< Process Component >> Add customer info 3.0
<< Process Component >> Create Quote # 4.0
<< Process Component >> Qualify Opportunity 5.0
FMEA
Potential failure modes
Effects of each failure mode
Severity (S) ranking
Occurrence (O) ranking
Causes of each failure
countermeasures
Detection (D) ranking
Calculate RPN (S x O x D)
<<Countermeasure>> Experienced staff should handle task 5.1
<<Countermeasure>> Senior Management should review "Qualify Opportunity" decision 5.2
<<Countermeasure>> Opportunity qualification shall be standardized by conducting "Leads scoring" 5.3
<<Countermeasure>> Standard operating procedure shall be created 5.4
<<Countermeasure>> Staff should be trained 5.5
<<Countermeasure>> Minimal number of fields shall be used on the form to reduce the time spent filling form 5.3.1
<<Countermeasure>> Existing customer information shall be automatically pulled from the DB to eliminate the need to search/fill such information 5.3.2
<<Countermeasure>> Required fields shall be indicated to users 5.3.5
<<Countermeasure>> Fields shall be validated before submission 5.3.6
<<Countermeasure>> Use select inputs instead of free inputs where applicable 5.3.3

<<Countermeasure>>
The following criteria shall be used to score leads

Criteria	Excellent Prospect	Okay Prospect	Bad Prospect
Contact Job Title	Senior Mgt. (10)	Middle Mgt. (5)	Team member (1)
Location	Canada (10)	US (5)	Others (1)
Company Size	> 5,000 (10)	1,000-5,000 (5)	< 1,000 (1)
Industry	Automotive (10)	Medical (5)	Solar (1)
Budget	> 50,000 (10)	10,000-50,000 (5)	< 10,000 (1)

5.3.4

<<Countermeasure>>
The following information shall be captured
a. Company name – add a company name and assign a score
b. Size – Choose the company size from the drop down options and assign a score
c. Revenue - Choose the revenue size from the drop down options and assign a score
d. Industry - Choose any industry from the drop down and assign a score
e. Location – Choose any location from the drop down and assign a score
f. Job title - Add the job title in the box provided and assign a score.
g. No of Visits – Specify the number in the box provided for no of visits and assign a score.
5.3.7

Figure 26: Traceability case study example

In addition to showing how high-level requirements, objectives, goals, needs, and so on are transformed into low-level requirements, the diagram also illustrates how requirements are traceable horizontally and vertically. *Horizontal traceability* (Jaber et al., 2013) refers to traces between requirements on the same level of abstraction, while *Vertical traceability* refers to traces between requirements across levels of abstraction (Jaber et al., 2013). A sample horizontal traceability portrayed on Figure 26 is the link between “Experienced staff should handle task”, “Senior Management should review qualify Opportunity decision” and “Opportunity qualification shall be standardized by conducting leads scoring” while the trace between “Opportunity qualification shall be standardized by conducting leads scoring” and “Required fields shall be indicated to users” portrays vertical traceability. Horizontally traceable requirements could be dependent, independent, complementary or conflicting, while vertically traceable requirements represent a dependent relationship.

A major advantage FBREM offers is that the traceability recording process is integrated into the requirement elicitation process. Therefore, associating requirements to each other does not have to be a separate activity, as it is with some of the techniques discussed in Section 4.4.2. FBREM is thus less likely to suffer consistency and completeness errors. Also, FBREM allows us to extract traceability information at different levels of abstraction. For example, an engineer’s interest may focus on how a data requirement implements the external interface requirement, while an auditor may only be interested in how the business rules are fulfilled by a functional requirement, and management is interested in knowing which requirement could serve as an alternative to a particular requirement. Such independent traceability information can be extracted from an FBREM result.

4.5 Comparison of FBREM with related techniques

In this section, we compare FBREM with other requirement elicitation and analysis methods reported in the literature.

4.5.1 KAOS

Kaos is a goal-oriented requirements approach to eliciting requirements (Dardenne, Lamsweerde, & Fickas, 1993) (Respect-IT, 2007). KAOS starts by specifying high-level abstract goals that describe the system that is being envisioned, and then continuously refines the goals into sub-goals and the agents responsible for the goal until low level executable requirements are determined. “The main emphasis of KAOS is on the formal proof that the requirements defined for the envisioned system match the goals" (Rubin & Rubin, 2010).

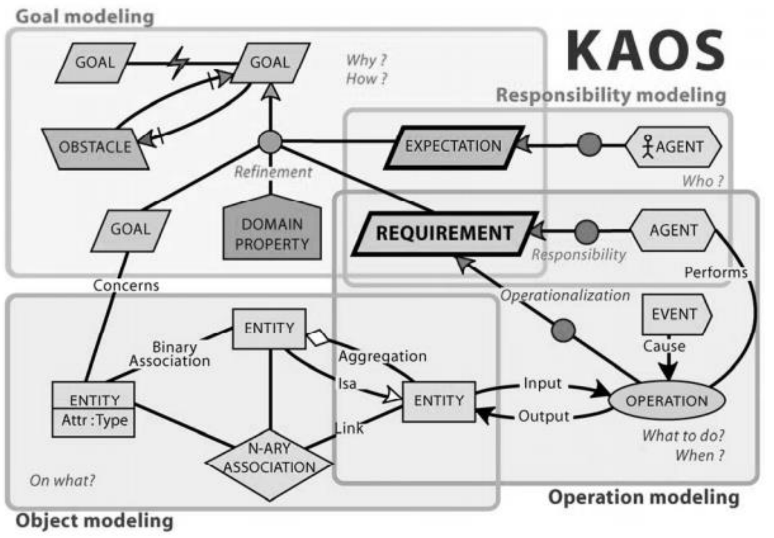


Figure 27: KAOS technique

4.5.2 Misuse cases

Misuse cases are a concept derived from the traditional "use case" that describes functions that the system should be able to perform. Misuse case is the inverse of use case: misuse cases represent behavior not wanted in the system, or threats to the system's goals (Sindre & Opdahl, 2001) (Sindre & Opdahl, 2004). The misuse case method involves identifying assets of the system to be developed, determining the misuse cases for those assets and then determining requirements to mitigate the misuse cases.

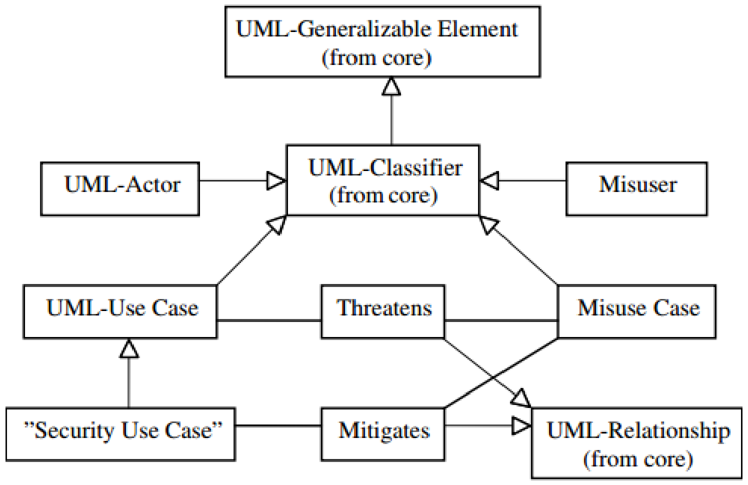

UML-Generalizable Element (from core)
UML-Actor
UML-Classifier (from core)
Misuser
UML-Use Case
Threatens
Misuse Case
"Security Use Case"
Mitigates
UML-Relationship (from core)

Figure 28: Misuse case technique

4.5.3 NFR framework

The non-functional approach starts with soft goals which describe the global quality of the system (Mylopoulos, Chung, & Nixon, 1992). Examples of soft goals include security, reliability, usability, and performance.. The soft goals are decomposed into subgoals, and analyzed to resolve conflict and dependencies among the subgoals. The NFR framework provides a structure for recording the decomposition and reasoning process in tree structure known as soft goal interdependency graph. The operation of the framework can be viewed as an incremental and interactive construction, elaboration, analysis and revision process. An evaluation procedure is used to determine when a soft goal has been satisfied by its sub goals.

4.5.4 GBRAM

GBRAM is a another method that uses goals as a means of systematically eliciting and analyzing requirements (Anton, 1996) (Fabian, Gürses, Heisel, Santen, & Schmidt, 2009). The method consists of two phases; goal analysis and goal refinement. Goal analysis is concerned with identifying and exploring available information sources for goals and classifying the goals; goal refinement involves identifying obstacles to the goals and operationalizing the goals into requirements.

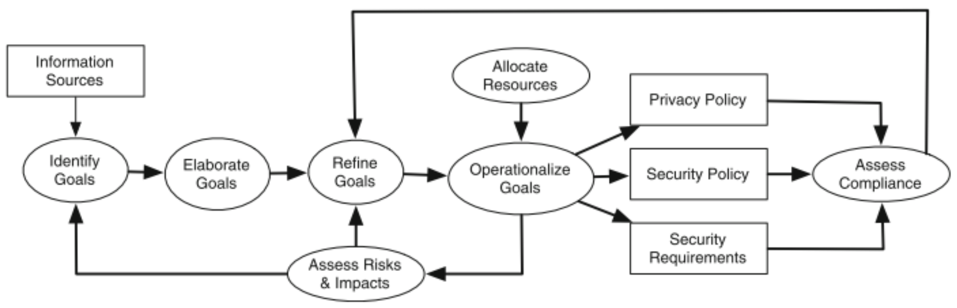


Figure 29: GBRAM technique

4.5.5 CORAS

CORAS is a stepwise and systematic risk analysis method with the overall objective of understanding the limitations of existing systems in order to design new features that will fill identified gaps (Braber, Hogganvik, Lund, Stølen, & Vraalsen, 2007) (Stølen, 2011). CORAS is conducted in eight steps which include: setting the scope and focus of the analysis; presentation of the goal of the analysis and setting of targets; refining the targets using asset diagrams in order to have a more refined understanding of the targets; approval and agreement on the targets; scope and other details of the project; identify all the possible potential threats, vulnerabilities and threat scenarios; conduct risk estimation to determine the likelihoods and consequences of the identified risks; evaluate the risk to determine which of the identified risks must be considered for possible treatment; conduct risk treatment in order to reduce the impact and likelihood of unacceptable identified risks.

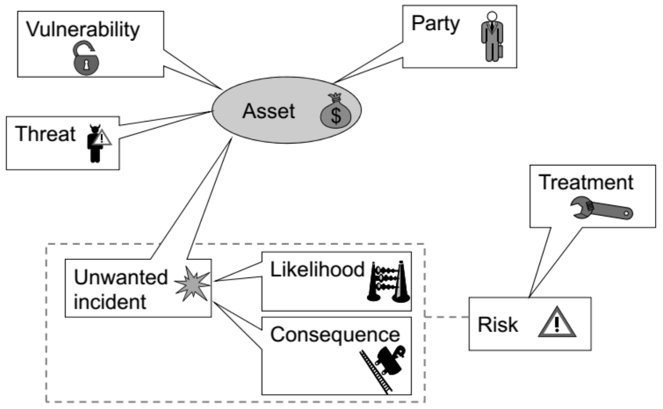


Figure 30: CORAS technique

4.5.6 ATAM

ATAM (Kazman et al., 1998) is a structured risk-mitigation technique for determining the suitable architecture for a system. Quality attributes are extracted from goals and then used to create scenarios. These scenarios are used in conjunction with architectural approaches to create an analysis of trade-offs, sensitivity points, and risks (or non-risks). ATAM aids in analyzing requirements along multiple dimensions to understand the effect of each of the requirements under different scenarios. Some of the benefits of ATAM include: improved requirements, more complete architectural documentation and earlier identification of risk factors.

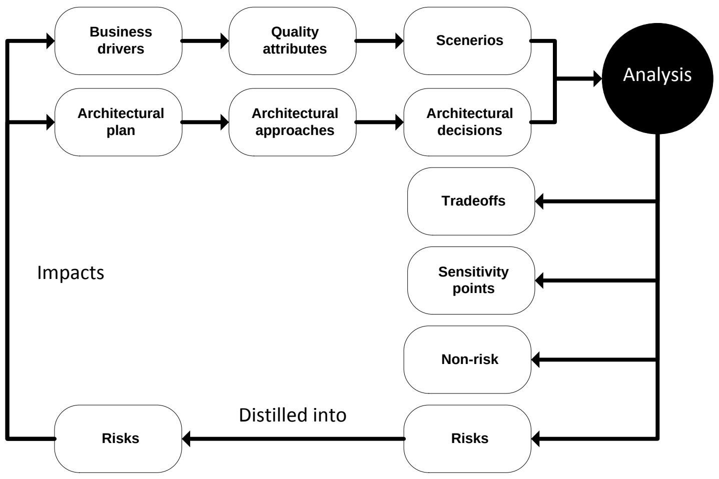


Figure 31: ATAM technique

4.6 Comparison of the techniques

FBREM and the other techniques share several characteristics, but as shown in Table 7 below, FBREM seems the most complete technique.

	FBREM	KAOS	MISUSE	NFR	GBRAM	CORAS	ATAM
Focus on risk/threat	✓		✓			✓	✓
Connection to goals	✓			✓	✓		✓
Prioritizing	✓	✓	✓		✓	✓	✓
Traceable	✓	✓	✓		✓	✓	✓
Retention of rationale	✓						✓
Levels of requirements	✓	✓			✓		
Recursively applied	✓						

Table 7: Completeness of techniques

Table 8 through 10 contain a more detailed comparison between FBREM and the other methods.

	FBREM	**KAOS**	**Misuse Cases**	**NFR**	**GBRAM**	**CORAS**	**ATAM**
Process	Derive requirements methodically using the notion of failure mode and countermeasure	Uses various models to refine goals into requirement	Employs the concepts of use cases and UML	Derives requirement from goals, using the softgoal dependency graph	Analyze goals considering obstacles and scenarios to uncover requirements	Stepwise risk analysis of system limitations eliciting appropriate requirements	Map quality attributes with scenario and architecture to determine minimal risk requirements
Driver	Risk	Goal	Risk	Goal	Goal	Risk	Risk
Levels or types of requirements	Requirements are hierarchically organized as business, user and product level	Separates functional from non-functional requirements	None	None	Separates user from system goals	None	None

Table 8: Method comparison: Process

	FBREM	KAOS	Misuse Cases	NFR	GBRAM	CORAS	ATAM
Criteria	Severity (SEV), likelihood (OCC), risk reduction (DET)	Heuristics	Mitigation cost (Implied)	None	Stakeholders negotiation	Risk assessment values	Vote by stakeholders
Factor Combinatio	CPN = SEV x OCC x DET	Yes	No	No	Yes	Yes	Yes
Objectivity	Relatively objective	Subjective	Objective		Subjective	Relatively objective	Subjective

Table 9: Method comparison: Prioritization

	FBREM	KAOS	Misuse Cases	NFR	GBRAM	CORAS	ATAM
Traceability, rationale	Builds a hierarchical model that contains the justifications for decisions taken and establishes traceability among requirements and other artifacts	Traceability by the KAOS relationships between model components	Misuse cases facilities traceability between various component s of the use case	Arguments in support of goals are represented with "claims" & "make" links	Method traces from goals to requirements	Stepwise process builds risk models which becomes the basis for traceability	Method generates scenarios which are mapped info the architecture to preserve rationale details

Table 10: Method comparison: Rationale and Traceability

Chapter 5
Conclusions and Future Work

Requirements are an important component of the development process. Requirements provide a description of what a system should do; they identify the boundaries of the system, as well as its features, attributes and qualities. Requirements are central to the concept of validating systems, and essential for establishing traceability between the various elements of the system. Requirements provide a baseline for quantification of system effort and for resource planning, and they contribute to system maintenance and update. Failure to capture requirements adequately can lead to missing functionality, improper allocation of resources, project rework leading to budget overrun, scope creep and delays, and difficulty in conducting validation and quantification activities.

The unsatisfactory experience typical in formal requirements elicitation is one of the main reasons why the agile approach is gaining in popularity. However, while agile may avoid the difficulties of formal elicitation of requirements, it also bypasses the analysis of user needs and the generation of a baseline against which the implemented system can be validated.

The research presented in this thesis is an effort towards showing that requirements can be deduced from the user stories and process maps that result from agile methodologies. We developed the Failure Mode Based Requirement Elicitation Method (FBREM) to systematically refine agile artifacts into system-specific, realizable and verifiable requirements. The requirements deduced using FBREM are presented in a format that will preserve the justification for decisions taken, and show traceability between the various levels of requirements and their rationale. The practicality of FBREM was examined in a case study.

5.1 Contributions of the thesis

The contributions of this thesis are as follows.

1. We showed how formal statements of requirements can be deduced from artifacts such as process maps that result from agile methodologies. We gave examples from our case study, and we showed how the method can be extended to more general use.

2. We showed that risk is a useful basis from which to deduce requirements. Empirically we observed the sensitivity of an agile team to its perceived risks; we extended this observation

to the idea that many, if not most, requirements are a response to some kind of risk. FBREM quantitatively and qualitatively analyzes the components of the agile artifacts to determine possible failure modes for the components as well as their causes and effects. The risk the failure modes pose are evaluated on the basis of their severity and likelihood. Countermeasures are then elicited to reduce the root causes of the failures (or at least alleviate their effects).

3. We showed that requirements can be structured in levels, depending on the specificity of the countermeasure. FBREM provides a means of eliciting various levels of requirement. The method derives requirements by progressively cycling through failure modes in such a way that the countermeasures of one level become the input for failure mode consideration in the next level. Eliciting requirements at different stages of abstraction help in managing implementation, demonstrating completion and conducting level-specific tests.

4. We showed that an objective prioritization of requirements is possible, based on countermeasure priority numbers. FBREM guides agile teams through an evaluation of requirements, not based on subjective preferences, but based on the severity and frequency of risks, and the risk reduction of proposed countermeasures. Prioritizing requirements objectively based on estimates of risk is better than leaving the prioritization to team guesses or development constraints.

5. The literature states that maintaining rationale is important in a requirements process. We showed that FBREM structures requirements so that the rationale of any particular software feature can be traced back through levels of requirements to the business goal.

6. The literature states that traceability is important in a requirements process. We showed that FBREM provides traceability between various levels of requirements and software features, which is essential in software validation, and important when considering changes to the software or re-evaluating design decisions.

5.2 Future Work

There are several areas in which future work could be conducted.

5.2.1 Complete the current case study

Our case study with MACE led us to the FBREM approach, which we have applied to part of MACE's business process. We have yet to review the elicited requirements with MACE executives, although they have expressed an interest in this analysis of their business process. It is likely that this review could lead MACE to suggest other failure modes, modify our assessments of severity and likelihood, and develop other possible countermeasures. We would expect certain aspects of FBREM to be validated through this exercise, while other aspects would be challenged and probably modified. We could also apply FBREM to all the other phases of the MACE business process.

5.2.2 New case studies in FBREM

Our empirical experience is valuable but is limited to our single case study. It would be important to evaluate and verify FBREM in several more case studies that may differ in the following parameters:

- Project size and length
- Geographical distribution of the project team
- Team familiarity with agile processes
- Projects with more strenuous risk requirements (such as those in regulated industries)

Although we have described FBREM as a method for deducing requirements from agile artifacts, it is also possible to use FBREM in a waterfall process, simply by starting with the business goals and using FBREM to elicit failure modes, causes, effects, and countermeasures from the requirements team.

5.2.3 Validate CPN

Our Countermeasure Priority Number (CPN) is a simple linear product of severity, likelihood, and effectiveness, following the existing notion of Risk Priority Number (RPN). However, we have not demonstrated that this linear product results in an accurate result, where by "accurate" we mean "corresponds to the actual reduction in risk that the countermeasure provides in practice". It would be useful to confirm that this linear product is accurate, or else develop some other mathematical function of these parameters (possibly non-linear) that gives a more accurate result. CPN is also subject to the same concerns that exist regarding accuracy of the estimates of severity and likelihood that exist with virtually all other risk models, such as RPN.

5.2.4 Recursive process

FBREM is a recursive process, in which one recursively develops countermeasures and then searches for the failure modes within the countermeasures. A natural question is: when should one stop the recursion? As described in this thesis, we rely on development teams to use their judgment in deciding when to stop the recursion, but it would be better if teams could perform some quantitative assessment to make this decision. Developing such a quantitative assessment is an area for further work. For example, a team may set up a certain level of “risk cost” that they are willing to absorb, and then apply recursion until the residual risk across a goal has been reduced below the risk cost threshold.

5.2.5 Tool support

A software tool that could be used to automate some aspects of FBREM is an area for future work. The main purposes of such a tool are:

1. To provide the capability for rapidly conducting the FBREM analysis
2. To automatically calculate CPN from severity, likelihood, and risk reduction parameters
3. To calculate and manage total risk across all goals
4. To rank the goals and components that remain at high risk (and therefore lead the team towards the work left to do in the FBREM approach)
5. To incorporate multiple models of CPN (as suggested in Section 5.2.3) and thus provide a tool for exploring risk using different models

Important features of the tool might include:

- A catalog of pre-defined components that can easily be adapted for specific needs using an intuitive property window to set preferences
- Templates and building blocks that represent best approaches for specific types of scenarios, disciplines or development styles

- Context-specific intelligence such as help, hints, recommendation, warnings, validation etc. that can provide support and guide the analyst in correctly applying the FBREM principle and using the tool
- Both a graphical and spreadsheet interface

One architectural design for such an FBREM tool can be found in Figure 32.

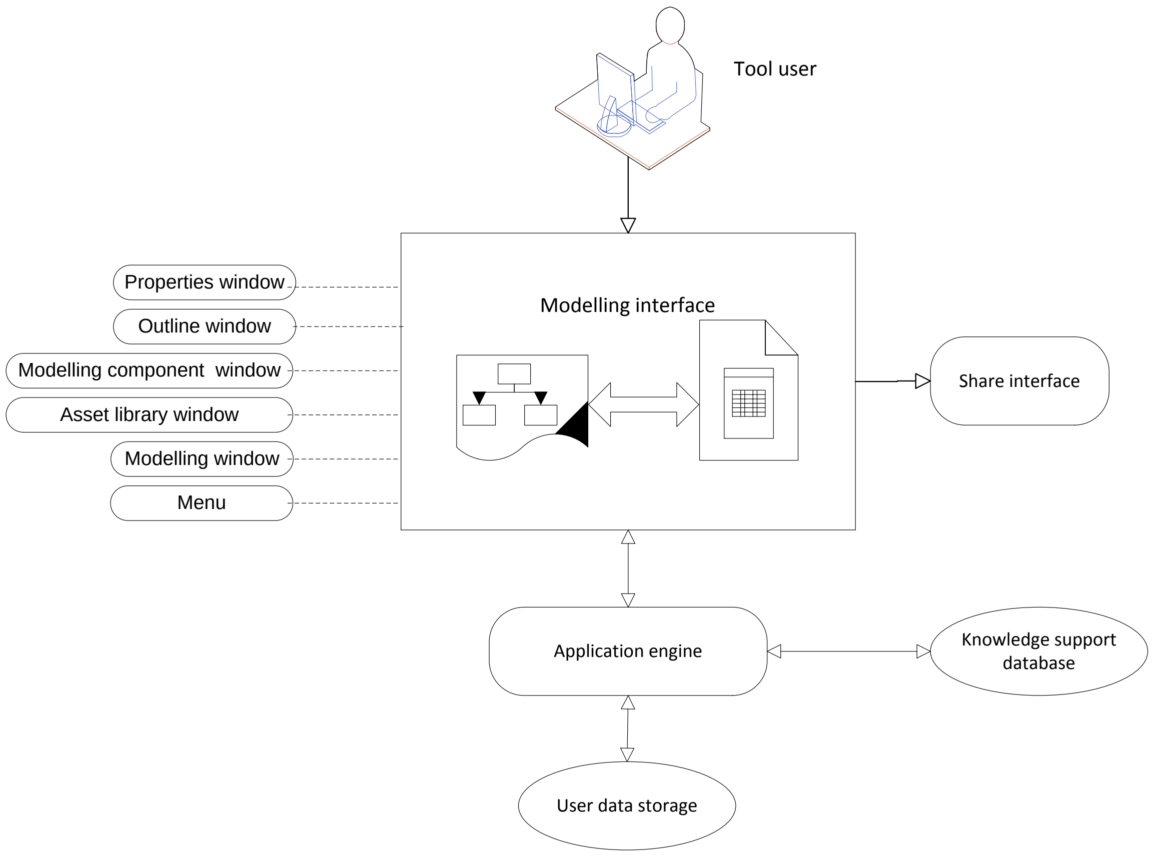


Figure 32: Architecture of FBREM software tool

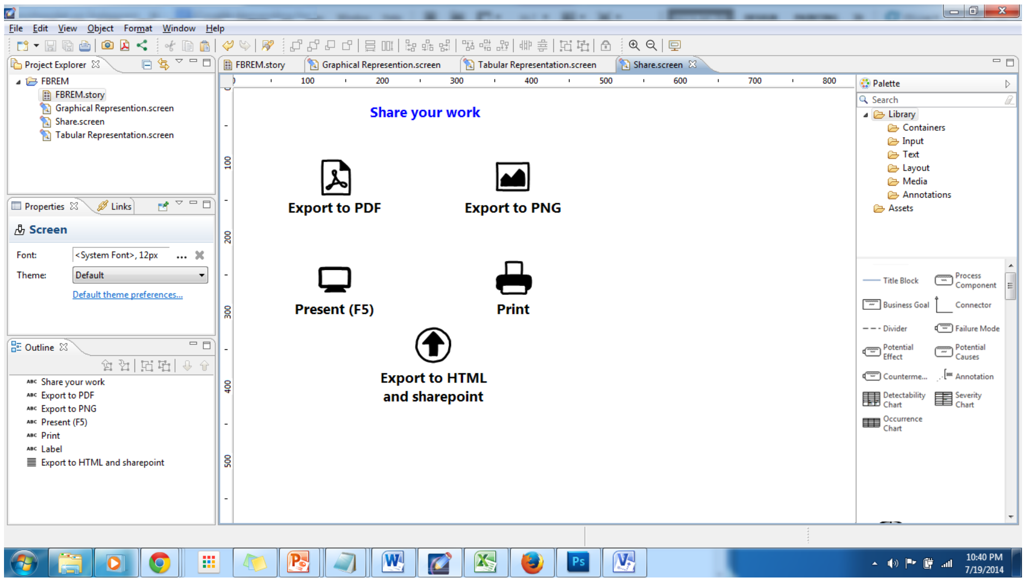
The architecture consists of five main components.

Application engine runs all programs.

User data storage stores all information about users, projects, and specific FBREM input and output.

Knowledge support database stores the knowledge of the FBREM model, including help, process validation rules, and all other information needed to ensure that as users interact with the tool they are always producing a valid FBREM model.

Share interface provides functionality to import and export data to the tool. A mockup of the share interface is shown in Figure 33.



Font: <System Font>, 12px

Theme: Default

Figure 33: Tool share interface

Modelling interface This is where tool users interact with the application. This component supports both the graphical mode and the spreadsheet mode for FBREM. Both presentation modes can also be convertible to each other. A mockup of the graphical mode presentation interface is shown in Figure 34 while the spreadsheet format is shown in Figure 35.

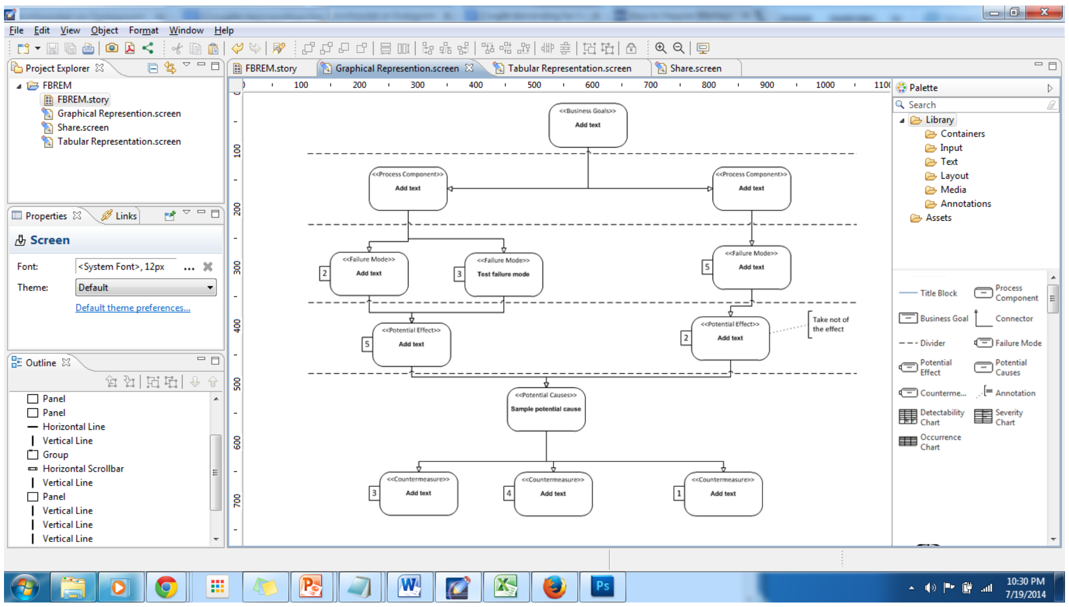
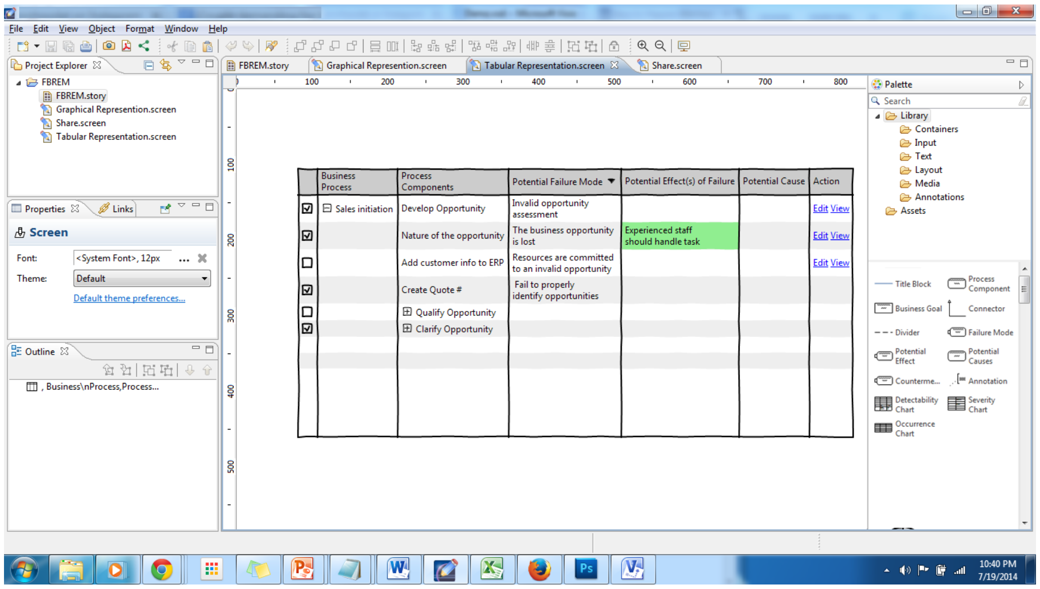


Figure 34: Tool Graphical Interface



	Business Process	Process Components	Potential Failure Mode	Potential Effect(s) of Failure	Potential Cause	Action
☑	⊟ Sales initiation	Develop Opportunity	Invalid opportunity assessment			Edit View
☑		Nature of the opportunity	The business opportunity is lost	Experienced staff should handle task		Edit View
☐		Add customer info to ERP	Resources are committed to an invalid opportunity			Edit View
☑		Create Quote #	Fail to properly identify opportunities			
☐		⊞ Qualify Opportunity				
☑		⊞ Clarify Opportunity				

Figure 35: Tool Spreadsheet Interface

5.2.6 Epilogue

This research presents an intriguing perspective to some of the challenges experienced in the various attempts at getting the best out the requirement engineering process. The research benefited from both theoretical and practical viewpoints in developing the Failure Mode Based Requirement Elicitation Method (FBREM) as a viable tool for achieving agility in the development process while not sacrificing the formal requirements analysis objectives.

Though we acknowledge that FBREM is only one of the efforts towards providing improving the requirements elicitation and analysis, to the best of our knowledge, it is the only method that is created from attempting to jointly avoid the risk both the agile and the traditional developments methods attempt to avoid individually.

It is our hope that this challenging but interesting research work will provoke new ways of thinking about development approaches and provide useful insights for industry requirement engineering professional as well as academic researchers.

Appendix A
Worked example of the FBREM method

The following tables show a fully worked example of the FBREM method for the ten top level tasks of the MACE Sales & Quotation phase. Each row of the table shows failure modes, effects of failure, severity of failure, potential causes, likelihood, various countermeasures and their risk reducing impact, the CPN for each countermeasure, and the number of the countermeasure. Countermeasures shaded in green are the most effective for that particular failure mode. The set of requirements (that is, the most effective countermeasures) elicited for these tasks in the Sales & Quotation phase are then summarized in a table at the end of this appendix.

LEVEL 1

1 - Develop Opportunity Develop business opportunities from RFQs or sales leads								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Lengthy sales cycles	Excessive cost of sale (human & budgetary resources are being used up)	3	Contact does not have decision power	2	Train staff on customer profiling and relationship management	4	24	1.1
			Customer unsure of what they want	4	Train staff on information elicitation	2	24	1.2
					Create parts and products catalog	3	36	1.3
			Customer is not buying yet	3	Train staff on customer profiling and relationship management	3	27	1.4
			Lengthy customer internal process	3	Train staff on customer profiling and relationship management	3	27	1.5
			Customer budget not allocated or dependent on other contract	3	Train staff on customer profiling and relationship management	4	36	1.6
Excessive competition	Excessive cost of sale leading to reduction in	3	Saturated market	2	Profile potential customers in order to create a niche offering	1	6	1.7

	profit margin				Seek new markets	3	18	1.8
			Customer already has a preferred vendor	4	Profile customer and create a niche offering	2	24	1.9
			Not enough reasons to choose Eclipse over competitors	3	Profile customer and create a niche offering	2	18	1.10
Lack of required certification	The business opportunity is lost	5	The company has not executed a similar project hence requiring certification to convince customer	2	Create a sales forecast module in ERP to proactively segment markets/customers and predict future market/customer requirements	2	20	1.11
			New legislation/regulation/customer	2	Create a sales forecast module in ERP to proactively segment markets/customers and predict future market/customer requirements	3	30	1.12
			Foreign market	3	Create a sales forecast module in ERP to proactively segment markets/customers and predict future market/customer requirements	3	45	1.13
	Delay in pursuing opportunity	4	The company has not executed a similar project hence requiring certification to convince customer	2	Create a sales forecast module in ERP to proactively segment markets/customers and predict future market/customer requirements	2	16	1.14
			New legislation/regulation/customer	2	Create a sales forecast module in ERP to proactively segment markets/customers and predict future market/customer requirements	3	24	1.15
			Foreign market	3	Create a sales forecast module in ERP to proactively segment markets/customers and predict future market/customer requirements	3	36	1.16
Limited resources to undertake sales activities	Opportunities are inadequately pursued	4	Resource constraints / too many opportunities at the same time	4	Outsource sales	2	32	1.17
					Contract part time staff	1	16	1.18
					Employ full time staff	3	48	1.19
					Prioritize the opportunities to be pursued	2	32	1.20
			Opportunities not well managed, resource poorly used	3	Train staff on sales management	4	48	1.21
	Sales cycles are prolonged	3	Resource constraints / too many opportunities at the same time	4	Outsource sales	2	24	1.22
					Contract part time staff	1	12	1.23
					Employ full time staff	3	36	1.24
					Prioritize the opportunities to be pursued	2	24	1.25
			Opportunities not well managed, resource poorly used	3	Train staff on sales management	4	36	1.26
	Reduced sales	4	Resource constraints	4	Outsource sales	2	32	1.27
					Contract part time staff	1	16	1.28
					Employ full time staff	3	48	1.29

					Screen opportunities to be considered	1	16	1.30
			Opportunities not well managed, resource poorly used	3	Train staff on sales management	2	24	1.31
False leads	No sale	5	Inaccurate information	2	Validate information by peer review	2	20	1.32
					Train staff on information elicitation	2	20	1.33
			Incomplete information	4	Validate information by peer review	2	40	1.34
					Train staff on information elicitation	2	40	1.35
			Evaluation criteria not well defined	5	Validate evaluation criteria and adjust accordingly	1	25	1.36
			Opportunity assessment is not being performed	3	Train staff on evaluating opportunities	2	30	1.37
No new leads	No new sale	5	No process to identify new opportunities/markets	4	Outsource leads generation	3	60	1.38
					Train staff on leads generation	2	40	1.39
					Make commission based deals with lead source partners (e.g. suppliers, customers of customer)	2	40	1.40
					Offer a compelling reward to returning customers	3	60	1.41
			Not enough marketing effort	5	Offer a compelling referral reward to current customers	3	75	1.42
					Recruit sales staff from competitor	3	75	1.43
					Explore new or expand reach by participating in trade shows, Fairs & Exhibitions, new media etc.	2	50	1.44

2 - Determine the nature of the opportunity Preliminarily determine the details of the opportunity								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Nature of the opportunity is wrongly determined or Task not performed	Wrong or below standards decisions are taken about the opportunity (Resources are wrongly assigned, Time is wasted in pursuing the opportunity wrongly, Fail to properly identify opportunities)	5	Nature of opportunity is not clear	2	Create screening checklist to filter opportunities	3	30	2.1
					Train staff on information elicitation techniques	4	40	2.2
					Escalate to manager	2	20	2.3
			Staff is not experienced enough to determine the nature of the opportunity	2	Train staff on how to determine the nature of the opportunity	1	10	2.4
					Discuss opportunity screening result with colleagues	2	20	2.5
			Not enough information to determine the nature of opportunity	3	Escalate to manager	2	30	2.6
					Train staff on information elicitation techniques	4	60	2.7
					Create required information checklist to guide elicitation	5	75	2.8
			Lack of standard operating procedure (SOP)	2	Create SOP for performing task	2	20	2.9
					Train staff on the use of the SOP	3	30	2.10
			Lack of adherence to the standard operating procedure	2	Train staff on the use of the SOP	3	30	2.11
					Institute consequence management program for non-compliance	2	20	2.12
	Opportunity is lost due to the wrong assessment	5	Wrong assessment of opportunity	4	Train staff on determining nature opportunity procedure	2	40	2.13
					Screen opportunities to be considered	1	20	2.14
					Discuss opportunity screening result colleagues	2	40	2.15
Nature of opportunity is indeterminate	Time is wasted in determining the nature of the opportunity	3	Nature of opportunity is not clear	2	Create screening checklist to filter opportunities	3	18	2.16
					Train staff on information elicitation techniques	4	24	2.17
					Escalate to manager	2	12	2.18
			Staff is not experienced enough to	2	Train staff on information elicitation	2	12	2.19

			determine the nature of the opportunity		techniques			
					Create screening checklist to guide opportunity screening exercise	2	12	2.20
			Not enough information to determine the nature of opportunity	3	Escalate to manager	2	18	2.21
					Train staff on information elicitation techniques	3	27	2.22
					Create screening checklist to guide elicitation	2	18	2.23

3 - Add customer info to ERP

Add information for new or unsecured customers to the DB

Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) or Mechanism(s) of Failure	Likelihood	Countermeasure	Risk reduction	CPN	#
Available information is incorrect/incomplete	Opportunity cannot be properly tracked in the ERP	3	Correct/complete information unavailable	3	Review opportunity with colleagues	3	27	3.1
					Escalate to manager	2	18	3.2
			Correct/complete information not requested	2	Validate information by peer review	3	18	3.3
					Train staff on information elicitation	3	18	3.4
	Wrong/incomplete customer information stated in quotation	3	Correct/complete information unavailable	3	Validate information by peer review	2	18	3.5
					Validate data to detect incorrect/incomplete data	2	18	3.6
			Correct/complete information not requested	2	Validate information by peer review	2	12	3.7
Inaccurate/Incomplete information is added	Opportunity cannot be tracked in the ERP	2	Correct/complete information unavailable	3	Validate information by peer review	2	12	3.8
			Data entry error	3	Validate data to detect incorrect/incomplete data	2	12	3.9
					Provide standard operating procedure	3	18	3.10
					Train staff on data entry	4	24	3.11
	Wrong/incomplete customer information stated in quotation	3	Correct/complete information unavailable	3	Validate information by peer review	2	18	3.12

			Data entry error	3	Validate data to detect incorrect/incomplete data	2	18	3.13
					Provide standard operating procedure	3	27	3.14
					Train staff on data entry	4	36	3.15

4 - Create Quote #
Register the quotation information in the DB

Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) or Mechanism(s) of Failure	Likelihood	Countermeasure	Risk reduction	CPN	#
Task not performed or delayed	Information about the quotation is not being entered into the ERP	2	Lack of standard operating procedure	4	Provide standard operating procedure	3	24	4.1
			Lack of training on procedure	4	Train staff on performing task	2	16	4.2
					Institute consequence management program to address non-compliance	3	24	4.3

5 - Qualify Opportunity
Pre-qualify the business opportunity

Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) or Mechanism(s) of Failure	Likelihood	Countermeasure	Risk reduction	CPN	#
Invalid opportunity assessment	The business opportunity is lost	5	Evaluation criteria not well defined	4	Create leads scoring module to standardize the qualification criteria	2	40	5.1
	Resources are committed to an invalid opportunity	3	Evaluation criteria not evaluated for opportunity	5	Create standard operating procedure on conducting opportunity assessment and using assessment result	3	45	5.2
					Institute consequence management program to address negligence	5	75	5.3
	Fail to properly identify opportunities	2	Assessment is not done by trained individual	2	Train staff conducting opportunity assessment	1	4	5.4

			Assessment is done by trained individual but they do not apply procedure correctly	1	Create standard operating procedure on conducting opportunity assessment and using assessment result	2	4	5.5
			Evaluation result is not used	4	Create standard operating procedure	2	16	5.6
			Lack of sufficient data to do proper evaluation	4	Create standard operating procedure	4	32	5.7
					Escalate to manager	3	24	5.8
			Lack of sufficient time to do proper evaluation	2	Create standard operating procedure	4	16	5.9
					Escalate to manager	2	8	5.10
Evaluation result is not used	Fail to properly identify opportunities	3	Lack of standard operating procedure	5	Create standard operating procedure on conducting opportunity assessment and using assessment result	1	15	5.11
	"Bad" opportunity is accepted	5	Lack of adherence to the standard operating procedure	2	Train staff on using on standard operating procedure	2	20	5.12
					Institute consequence management program to address negligence	4	40	5.13
			Lack of training on procedure	5	Train staff on how to use evaluation result	1	25	5.14
Opportunity is not qualified	Fail to properly identify opportunities	4	Lack of standard operating procedure	5	Create standard operating procedure on conducting opportunity assessment and using assessment result	1	20	5.15
	"Bad" opportunity is accepted	5	Lack of adherence to the standard operating procedure	2	Train staff on following standard operating procedure	2	20	5.16
					Institute consequence management program to address negligence	4	40	5.17
			Lack of training on procedure	5	Train staff qualifying opportunity	1	25	5.18

6 - Perform Credit Check

Assess the financial capability of the customer

Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) or Mechanism(s) of Failure	Likelihood	Countermeasure	Risk reduction	CPN	#
Limited credit information	Fail to properly qualify opportunities	4	Limited customer credit information	2	Escalate to manager	4	32	6.1
Credit report	Fail to properly identify	4	Error from credit agency	2	Multi credit agency checks	1	8	6.2

dispute	opportunities				Escalate to manager	4	32	6.3
			Inaccurate information transmitted to credit agency	2	Train staff on task	3	24	6.4
					Institute consequence management program to address negligence	4	32	6.5
					Validate information by peer review	2	16	6.6
	Relationship with customer is strained	4	Error from credit agency	2	Multi credit agency checks	1	8	6.7
					Escalate to manager	4	32	6.8
			Inaccurate information transmitted to credit agency	2	Train staff on task	3	24	6.9
					Institute consequence management program to address negligence	4	32	6.10
					Validate information by peer review	2	16	6.11
	The business opportunity is lost	5	Error from credit agency	2	Multi credit agency checks	1	10	6.12
					Escalate to manager	4	40	6.13
			Inaccurate information transmitted to credit agency	2	Train staff on task	3	30	6.14
					Validate information by peer review	2	20	6.15
					Institute consequence management program to address negligence	4	40	6.16
Response delay from credit agency	Relationship with customer is disrupted	4	Delays in making request to credit agency	3	Train staff on task	3	36	6.17
					Institute consequence management program to address negligence	4	48	6.18
					Add calendar and task management module to ERP	2	24	6.19
			Delays in receiving response from credit agency	3	Add calendar and task management module to ERP	2	24	6.20
					Escalate to manager	4	48	6.21
	Quotation process is stalled	3	Delays in making request to credit agency	3	Train staff on task	3	27	6.22
					Institute consequence management program to address negligence	4	36	6.23
					Add calendar and task management module to ERP	2	18	6.24
			Delays in receiving response from credit agency	3	Add calendar and task management module to ERP	2	18	6.25
					Escalate to manager	4	36	6.26
Credit agency data is not reliable	Fail to properly identify opportunities	4	Error from credit agency	2	Multi credit agency checks	1	8	6.27
					Escalate to manager	4	32	6.28
			Inaccurate information transmitted to credit agency	2	Train staff on task	3	24	6.29
					Institute consequence management program to address negligence	4	32	6.30

					Validate information by peer review	2	16	6.31
	Bid on job for customer with bad credit	5	Error from credit agency	2	Multi credit agency checks	1	10	6.32
					Escalate to manager	4	40	6.33
			Inaccurate information transmitted to credit agency	2	Train staff on task	2	20	6.34
					Institute consequence management program to address negligence	4	40	6.35
					Validate information by peer review	2	20	6.36
Credit check is not done	Fail to properly identify opportunities or structure payments	4	Lack of standard operating procedure	4	Create standard operating procedure for performing task	3	48	6.37
			Lack of training on procedure	4	Train staff on credit check task	2	32	6.38
	Bid on job for customer with bad credit	5	Lack of standard operating procedure	4	Create standard operating procedure for performing task	3	60	6.39
			Lack of training on procedure	4	Train staff on credit check task	2	40	6.40

7 - Log decision into the ERP
Log decision not to proceed with the quotation in the DB

Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) or Mechanism(s) of Failure	Likelihood	Countermeasure	Risk reduction	CPN	#
Task not performed	Decision and related information is lost	2	Lack of standard operating procedure	4	Create standard operating procedure for performing task	3	24	7.1
			Lack of training on procedure	4	Train staff how to log decision into the database	2	16	7.2

8 - Communicate decision to Customer if not quoting Communicate decision not to quote to the customer								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Customer didn't receive information or task not performed	Company's reputation is negatively perceived	3	Message not sent through the appropriate channel/format	3	Create standard operating procedure for performing task	3	27	8.1
					Train staff on task	2	18	8.2
			Message sent to the wrong address	3	Create standard operating procedure for performing task	3	27	8.3
					Train staff on task	2	18	8.4
No quote message is not properly communicated	Company's reputation is negatively perceived	3	Lack of standard operating procedure	4	Create standard operating procedure for performing task	3	36	8.5
			Lack of training on procedure	4	Train staff on task	2	24	8.6
					Validate message by peer review	2	24	8.7
Task not performed	Company's reputation is negatively perceived	3	Lack of standard operating procedure	4	Create standard operating procedure for performing task	3	36	8.8
			Lack of training on procedure	4	Train staff on task	2	24	8.9

9 - Gather Information Gather information needed to successfully quote the opportunity								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Limited in-house experience on the technology required to execute job	The business opportunity is lost (unable to produce a viable quotation)	5	Required technology is new or emerging	5	Train staff	2	50	9.1
					Outsource activity	3	75	9.2
					Invest in R&D	2	50	9.3
					Employ personnel with requisite skill &	1	25	9.4

					experience			
			Staffs are yet to be trained	4	Train staff on the technology	1	20	9.5
			Suitable supplier or resource is yet to be identified	3	Outsource activity	3	45	9.6
	Increase in the number of proposal revisions due to rework	3	Required technology is new or emerging	5	Train staff	2	30	9.7
					Outsource job	3	45	9.8
					Invest in R&D	2	30	9.9
					Employ personnel with requisite skill & experience	1	15	9.10
Incomplete/no information gathered	The business opportunity is lost (unable to produce a proposal which addresses customer needs)	5	Customer may not know or reluctant to release information	3	Establish non-disclosure agreements to make customer comfortable	2	30	9.11
					Create information elicitation checklist	2	30	9.12
					Hold frequent meetings with customer	3	45	9.13
			We neglect to request information during period when it can be requested	3	Create information elicitation checklist	3	45	9.14
					Add calendar and task management module to ERP	2	30	9.15
					Institute consequence management program for non-compliance	4	60	9.16
			We do not know what questions to ask because we are not familiar with customer needs	3	Create information elicitation checklist	2	30	9.17
	Increase in the number of quotation revisions due to rework (implying that more cost is incurred)	3	Customer may not know or reluctant to release information	3	Establish non-disclosure agreements to make customer comfortable	2	18	9.18
					Create information elicitation checklist	2	18	9.19
			We neglect to request information during period when it can be requested	3	Create information elicitation checklist	3	27	9.20
					Add calendar and task management module to ERP	2	18	9.21
					Institute consequence management program for non-compliance	4	36	9.22
			We do not know what questions to ask because we are not familiar with customer needs	3	Create information elicitation checklist	3	27	9.23
					Train staff on information elicitation	2	18	9.24
Inaccurate information	The business opportunity is lost (unable to produce a proposal which addresses customer needs) or excessive rework leading to higher	5	Customer is misleading us in order to make the job cheaper	4	Train staff	2	40	9.25
					Create information elicitation checklist	2	40	9.26
			Customer representative is not knowledgeable	3	Create information elicitation checklist to guide customer on expectation	3	45	9.27

	cost				Escalate to manager	2	30	9.28
			Transcription errors	3	Validate entry	2	30	9.29
					Validate information by peer review	2	30	9.30
			Information is rapidly changing and we are not informed of changes	4	Freeze requirements and obtain sign-offs	2	40	9.31
					Hold frequent meetings with customer	1	20	9.32
			Ambiguous information is obtained from customer	4	Train staff	2	40	9.33
					Create information elicitation checklist	2	40	9.34
					Validate information by peer review	3	60	9.35
	Strained relationship between manufacturing and sales/app engineering	2	Customer is misleading us in order to make the job cheaper	4	Train staff	2	16	9.36
					Create information elicitation checklist	2	16	9.37
			Customer representative is not knowledgeable	3	Create information elicitation checklist to guide customer on expectation	3	18	9.38
					Escalate to manager	2	12	9.39
			Transcription errors	2	Validate data entry	2	8	9.40
					Validate information by peer review	2	8	9.41
			Information is rapidly changing and we are not informed of changes	4	Freeze requirements and obtain sign-offs	2	16	9.42
			Ambiguous information is obtained from customer	4	Hold frequent meetings with customer	1	8	9.43
					Train staff on information elicitation	2	16	9.44
Data required to quote the job is not available	Unable to properly quote job	4	Unavailability of information required to quote job	4	Escalate to manager	2	32	9.45

10 - Assign Resources Allocate human and budgetary resources to develop the concept								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Unavailable/limited human/budgetary resources to purse the quotation process	Poor quality quotation is developed	4	Too many other opportunities at the same time or Limited resources	4	Outsource task	2	32	10.1
					Contract part time staff	1	16	10.2
					Employ full time staff	3	48	10.3
					Prioritize projects	2	32	10.4
	The business opportunity is lost (unable to produce timely and appropriate quotation which addresses customer needs)	5	Too many other opportunities at the same time or Limited resources	4	Outsource task	2	40	10.5
					Contract part time staff	1	20	10.6
					Employ full time staff	3	60	10.7
					Prioritize projects	2	40	10.8
	Extra amount of other resources are committed to make up for the unavailable resource	3	Too many other opportunities at the same time or Limited resources	4	Outsource task	2	24	10.9
					Setup a staff compensation scheme	3	36	10.10
					Prioritize projects	2	24	10.11
Resources are not requested / Resources are not assigned	Other business/customers are disrupted as we try to handle unscheduled work	3	Resources are not properly scheduled	4	Add ticket management module to ERP	1	12	10.12
			Resource constraints	4	Contract part time staff	1	12	10.13
					Employ full time staff	3	36	10.14
					Prioritize projects	2	24	10.15
	Have to pay overtime or hire additional resources because of poor scheduling	3	Resources are not properly scheduled	4	Add ticket management module to ERP	1	12	10.16
			Resource constraints	4	Contract part time staff	1	12	10.17
					Employ full time staff	3	36	10.18
					Prioritize projects	2	24	10.19
	Strained relationship between manufacturing and sales/app engineering	2	Resources are not properly scheduled	4	Add ticket management module to ERP	1	8	10.20
			Resource constraints	4	Contract part time staff	1	8	10.21
					Employ full time staff	3	24	10.22
					Prioritize projects	2	16	10.23

LEVEL - 2

1.7 - Profile potential customers in order create a niche offering								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Ineffective profiling result	Inability to grow sales. Loss due to wasted efforts	4	Limited customer information	4	Outsource profiling activity	3	48	1.7.1
					Add customer profiling module to ERP in order to maintain customer data	2	32	1.7.2
					Create a customer feedback channel	2	32	1.7.3
			Limited resources to carry out profiling activity	3	Train staff on customer profiling	1	12	1.7.4
					Allocate budget for customer profiling	2	24	1.7.5
					Outsource profiling activity	3	36	1.7.6
					Recruit staff	2	24	1.7.7
Inability to create niche offering	Inability to grow sales	4	Limited know-how	4	Improve knowledge base by hiring skilled staff	1	8	1.7.8
					Train existing staff	2	16	1.7.9

1.11, 1.14 - Create a sales forecast module in ERP to proactively segment markets/customers and predict future market/customer requirements								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Module is not being used	The business opportunity is lost due to market uncertainty	5	Lack of adherence to the standard operating procedure or lack of training	3	Train staff on how to use module	2	30	1.11.1, 1.14.1
					Institute consequence management program for non-compliance	4	60	1.11.2, 1.14.2

			Lack of standard operating procedure	3	Provide standard operating procedure on entering data into the leads scoring module	3	45	1.11.3, 1.14.3
Result from module is not effective	The business opportunity is lost due to market uncertainty	5	Module is not being properly used	3	Train staff on how to use module	2	30	1.11.4, 1.14.4
			Sales forecast parameters is not appropriate	4	Properly set features and functions of the forecast module | | Target | Achieved | Pipeline Potential | | $ | | | | | % | | | |	1	20	1.11.5, 1.14.5

.

1.30, 2.14, 10,11 - Prioritize opportunities to be pursued								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Prioritization not done or not done correctly or result not effective	Fail to allocate resources appropriately to opportunities	5	Not enough information to prioritize opportunity	4	Escalate to manager	3	60	1.30.1
					Train staff on information elicitation techniques	2	30	1.30.2
			No standard process for prioritizing opportunities	4	Use the scoring quadrant | Low reward High risk (Avoid) | High reward High risk (Evaluate) | | Low reward Low risk (Evaluate) | High reward Low risk (Pursue) |	2	20	1.30.3
					Create standard operating procedure	3	60	1.30.4

			Lack of know-how	2	Train staff on conducting opportunity prioritization	2	20	1.30.5
			Political interest	3	Escalate to manager	3	45	1.30.6
					Create standard operating procedure	2	20	1.30.7

1.40 - Make commission based deals with lead source partners (e.g. suppliers, customers of customer)

Potential Failure Mode	Potential Effect(s) of Failure	Severity	Potential Cause(s) or Mechanism(s) of Failure	Likelihood	Countermeasure	Risk reduction	CPN	#
Program is not successful or loss is incurred or lack of commitment on the part of partners	Loss in revenue	5	Lack of support and coordination or clarity of purpose	3	Define the terms, condition and features of the commission based sales scheme	2	30	1.40.1
					Train partners on the workings of the commission based sales scheme	2	30	1.40.2
					Set up a project management office (PMO)	1	15	1.40.3
Partners use privilege information for other purposes	Unhealthy internal competition	4	Conflict of interest	3	Establish non-disclosure agreements with partners	3	36	1.40.4
					Create standard operating procedure	3	36	1.40.6
					Train partners on the workings of the commission based sales scheme	2	24	1.40.7
			Unethical behavior	2	Institute consequence management program for non-compliance	4	32	1.40.8
					Train partners on the workings of the commission based sales scheme	2	16	1.40.9
Dispute from sales monitoring	Strained relationship with partners	3	Lack of support and coordination, lack of clarity, conflict of interest or unethical behavior	2	Establish a dispute management channel	3	18	1.40.10
					Create standard operating procedure	4	24	1.40.11
Misrepresentation of facts by third party	Strained relationship with customers	4	Lack of support and coordination, lack of clarity, conflict of interest or unethical behavior	2	Create standard operating procedure	4	32	1.40.12
					Train partners on the workings of the commission based sales scheme	2	16	1.40.13
					Create sales confirmation/follow-up procedure system	2	16	1.40.14

2.20 - Create screening checklist to guide opportunity screening exercise									
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Module is not being used	Fail to screen out "Bad" opportunity early	4	Lack of adherence to the standard operating procedure or lack of training	3	Train staff on how to use module	2	24	2.20.1
					Institute consequence management program for non-compliance	3	36	2.20.2
			Lack of standard operating procedure	3	Provide standard operating procedure on operating the leads scoring module	3	36	2.20.3
Result from module is not effective	Fail to screen out "Bad" opportunity early	4	No standard process for screening opportunities	4	Checklist to screen opportunities early is as follows: 1. Is it real? (Funding, market, experiences) 2. Can we win? (Competition, resource, timing) 3. Is it worth it? (Cost, risk, returns, strategy)	1	16	2.20.4
			Lack of know-how	3	Train staff on conducting opportunity prioritization	2	24	2.20.5
			Political interest	2	Escalate to manager	3	24	2.20.6

3.9 - Validate data to detect incorrect/incomplete data								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Validation rule is deficient	Dirty data (inaccurate, incomplete or erroneous data) is stored in the ERP resulting in difficulty in querying ERP	3	Data not being validated	5	Validate user information upon submit. The validation rule is as follows: • Has the user left required fields empty? • Has the user entered a valid e-mail address? • Has the user entered a valid date? • Has the user entered text in a numeric field?	1	15	3.9.1

5.1 - Create leads scoring module to standardize the qualification criteria								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Lead scoring module is not being used	Fail to properly qualify opportunities	4	Lack of adherence to the standard operating procedure or lack of training	3	Train staff on how to use leads scoring module	3	36	5.1.1
					Institute consequence management program for non-compliance	4	48	5.1.2
Data is not entered correctly into the lead scoring module	Misleading lead score leading to poor decision concerning the opportunity	5	Time pressure sue to too many form fields	3	Use selected inputs instead of free inputs where possible. Minimal number of fields shall be used on the form to reduce the time spent filling form	2	30	5.1.3
			Lack of standard	3	Provide standard operating procedure on entering data into the leads	2	30	5.1.4

			operating procedure		scoring module			
			Lack of adherence to the standard operating procedure	3	Train staff on using on standard operating procedure	3	45	5.1.5
					Institute consequence management program to address negligence	4	60	5.1.6
			Lack of training on procedure	4	Train staff on entering data into and using the scoring module	3	60	5.1.7
			No data validation	3	Required fields shall be indicated to users	2	30	5.1.8
					Validate user information upon submit. The validation rule is as follows: • Has the user left required fields empty? • Has the user entered a valid e-mail address? • Has the user entered a valid date? • Has the user entered text in a numeric field?	2	30	5.1.9
Wrong scoring criteria/business rule	Misleading lead score leading to poor decision concerning the opportunity	5	Essential information is not captured Business rule is not valid	3	The following information shall be captured a. Company name – add a company name and assign a score b. Size – Choose the company size from the drop down options and assign a score c. Revenue - Choose the revenue size from the drop down options and assign a score d. Industry - Choose any industry from the drop down and assign a score e. Location – Choose any location from the drop down and assign a score f. Job title - Add the job title in the box provided and assign a score g. No of Visits – Specify the number in the box provided for no of visits and assign a score	1	15	5.1.10

					The following criteria shall be used to score leads	1	15	5.1.11

Criteria	Excellent Prospect		Reasonable Prospect		Bad Prospect	
Contact Job Title	Senior Mgt.	10	Middle Mgt.	5	Team member	1
Location	Canada	10	US	5	Others	1
Company Size	> 5,000	10	1,000-5,000	5	< 1,000	1
Industry	Automotive	10	Medical	5	Solar	1
Budget	> 50,000	10	10,000-50,000	5	< 10,000	1

Data required to complete the lead scoring form is not available or has not being obtained	Fail to properly qualify opportunities	4	Limited customer information	4	Escalate to manager	2	32	5.1.12
			Lack of standard operating procedure	3	Create standard operating procedure on obtaining for and entering data into the leads scoring module	3	36	5.1.13
			Lack of adherence to the standard operating procedure	3	Train staff on using on standard operating procedure	3	36	5.1.14
					Institute consequence management program to address negligence	4	48	5.1.15
			Lack of training on procedure		Train staff on obtaining for and entering data into the leads scoring module			

5.2, 5.5, 5.11, 5.15 - Create standard operating procedure (SOP) on conducting opportunity assessment and using assessment result								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Standard operating procedure does not	Company's processes are not standardized affecting repeatability	4	Lack of know-how on how to create SOP	2	Train staff on how to create and implement SOP	2	16	5.2., 5.5.,

address necessary issues	in service delivery							5.11., 5.15.
			SOP does not address the issues appropriately	4	The SOP should contain the following elements: • Rationale for SOP • Detailed description of procedure based on best practice/standards • Monitoring actions • Accountability • Corrective Actions • Date of last review or revision date	1	16	5.2.1, 5.5.1, 5.11.1, 5.15.1
Standard operating procedure is not being followed	Company's processes are not standardized affecting repeatability in service delivery	4	Lack of training on how to use the SOP	2	Train staff on how to apply the SOP	2	16	5.2.2, 5.5.2, 5.11.2, 5.15.2
			SOP format is not user friendly	2	Create the SOP using an interactive format	3	24	5.2.3, 5.5.3, 5.11.3, 5.15.3
			Lack to adherence to the SOP	2	Train staff on using on standard operating procedure	2	16	5.2.4, 5.5.4, 5.11.4, 5.15.4
					Institute consequence management program to address negligence	4	32	5.2.5, 5.5.5, 5.11.5, 5.15.5
			SOP is outdated	4	Set up SOP review committee to review SOP annually and as need arises	1	16	5.2.6, 5.5.6, 5.11.6, 5.15.6

6.2, 6.7, 6.12, 6.27, 6.32 - Multi credit agency checks								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Conflicting or erroneous report from credit agency	Delay in submitting quotation	4	Error from credit agency	2	Escalate to manager	3	24	6.2.1, 6.7.1, 6.12.1, 6.27.1, 6.32.1

6.19, 6.20, 6.24, 6.25 - Add calendar and task management module to ERP								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Calendar and task management module is not being used	Failure in remembering to carry out tasks leading delay or inability to submit quotation	4	Lack of training on how to use the SOP	2	Train staff on how and when to use the Calendar and task management module	2	16	6.19.1, 6.20.1, 6.24.1, 6.25.1
			SOP format is not user friendly	2	Create standard operating procedure on using the Calendar and task management module	3	24	6.19.2, 6.20.2, 6.24.2, 6.25.2
			Lack to adherence to the SOP	2	Institute consequence management program to address negligence	2	16	6.19.3, 6.20.3, 6.24.3, 6.25.3

Failure in remembering to carry out tasks	Delay or inability to submit quotation	4	Features of the calendar and task management module is not effective	4	Add calendar and task management module to ERP with the following features: • User-definable data fields • Quick, easy data entry with automatic field defaults, AutoCorrect and speed entry templates • Progress monitoring and indicators • Ability to set recurring tasks, jobs and projects • Automatic task scheduling • Automatic data backup • Optional task synchronization with Microsoft Outlook	1	16	6.19.4, 6.20.4, 6.24.4, 6.25.4
					Train staff on how on how & when to carry out tasks	2	32	6.19.5, 6.20.5, 6.24.5, 6.25.5
			Negligence	3	Create standard operating procedure on how & when to carry out tasks	3	36	6.19.6, 6.20.6, 6.24.6, 6.25.6
					Institute consequence management program to address negligence	4	48	6.19.7, 6.20.7, 6.24.7, 6.25.7

10.12, 10.16, 10.20 - Add ticket management module to ERP								
Potential Failure Mode	**Potential Effect(s) of Failure**	**Severity**	**Potential Cause(s) or Mechanism(s) of Failure**	**Likelihood**	**Countermeasure**	**Risk reduction**	**CPN**	**#**
Resources are not properly scheduled	Delay or inability to submit quotation	4	Features of the ticket management module is not effective	3	Add ticket management module to ERP with the following features: • Maintain accurate resource profiles with groupings, roles etc. • Define attributes for different resource types. E.g. Skills • Define primary & secondary task resource • Integrate application with outlook • Send notifications • Provide utilization & availability report • Forecast resource shortage and surplus	1	12	10.12.1, 10.16.1, 10.20.1

Summary of Risk-Based Requirements

1 - Develop Opportunity			
Level 1		**Level 2**	
1.1	Train staff on customer profiling and relationship management		
1.2	Train staff on information elicitation		
1.7	Profile potential customers in order create a niche offering	1.7.4	Train staff on customer profiling
		1.7.8	Improve knowledge base by hiring skilled staff
1.11, 1.14	Create a sales forecast module in ERP to proactively segment markets/customers and predict future market/customer requirements	1.11.1, 1.14.1	Train staff on how to use module
		1.11.5, 1.14.5	Properly set features and functions of the forecast module
1.18, 1.23, 1.28	Contract part time staff		
1.30	Screen opportunities to be considered	1.30.3	Use the scoring quadrant
		1.30.5	Train staff on conducting opportunity prioritization
		1.30.7	Create standard operating procedure
1.32	Validate information by peer review		
1.33	Train staff on information elicitation		
1.39	Train staff on leads generation		
1.40	Make commission based deals with lead source	1.40.3	Set up a project management office (PMO)

	Target	Achieved	Pipeline Potential
$			
%			

Low reward High risk (Avoid)	High reward High risk (Evaluate)
Low reward Low risk (Evaluate)	High reward Low risk (Pursue)

	partners (e.g. suppliers, customers of customer)	1.40.9	Train partners on the workings of the commission based sales scheme
		1.40.10	Establish a dispute management channel
		1.40.13	Train partners on the workings of the commission based sales scheme
		1.40.14	Create sales confirmation/follow-up procedure system
2 - Determine the nature of the opportunity			
Level 1		**Level 2**	
2.4	Train staff on customer profiling and relationship management		
2.14	Screen opportunities to be considered		
2.18	Escalate to manager		
2.19	Train staff on information elicitation techniques		
2.20	Create screening checklist to guide opportunity screening exercise	2.20.1	Train staff on how to use module
		2.20.4	Checklist to screen opportunities early is as follows: • Is it real? (Funding, market, experiences) • Can we win? (Competition, resource, timing) • Is it worth it? (Cost, risk, returns, strategy)
3 - Add customer info to ERP			
Level 1		**Level 2**	
3.2	Escalate to manager		
3.3, 3.7, 3.8, 3.12, 3.13	Validate information by peer review		
3.4	Train staff on information elicitation		
3.9	Validate data to detect incorrect/incomplete data	3.9.1	Validate user information upon submit. The validation rule is as follows: • Has the user left required fields empty? • Has the user entered a valid e-mail address? • Has the user entered a valid date? • Has the user entered text in a numeric field?
4 - Create Quote #			
Level 1		**Level 2**	
4.2	Train staff on performing task		
5 - Qualify Opportunity			
Level 1		**Level 2**	
5.1	Create leads scoring module to standardize the qualification criteria	5.1.1	Train staff on how to use leads scoring module
		5.1.3	Use selected inputs instead of free inputs where possible. Minimal number of fields shall be used on the form to reduce the time

			spent filling form
		5.1.4	Provide standard operating procedure on entering data into the leads scoring module
		5.1.8	Required fields shall be indicated to users
		5.1.9	Validate user information upon submit. The validation rule is as follows: • Has the user left required fields empty? • Has the user entered a valid e-mail address? • Has the user entered a valid date? Has the user entered text in a numeric field?
		5.1.10	The following information shall be captured a. Company name – add a company name and assign a score b. Size – Choose the company size from the drop down options and assign a score c. Revenue - Choose the revenue size from the drop down options and assign a score d. Industry - Choose any industry from the drop down and assign a score e. Location – Choose any location from the drop down and assign a score f. Job title - Add the job title in the box provided and assign a score g. No of Visits – Specify the number in the box provided for no of visits and assign a score
		5.1.11	The following criteria shall be used to score leads (see table below)
		5.1.12	Escalate to manager
5.4	Train staff on conducting opportunity assessment		

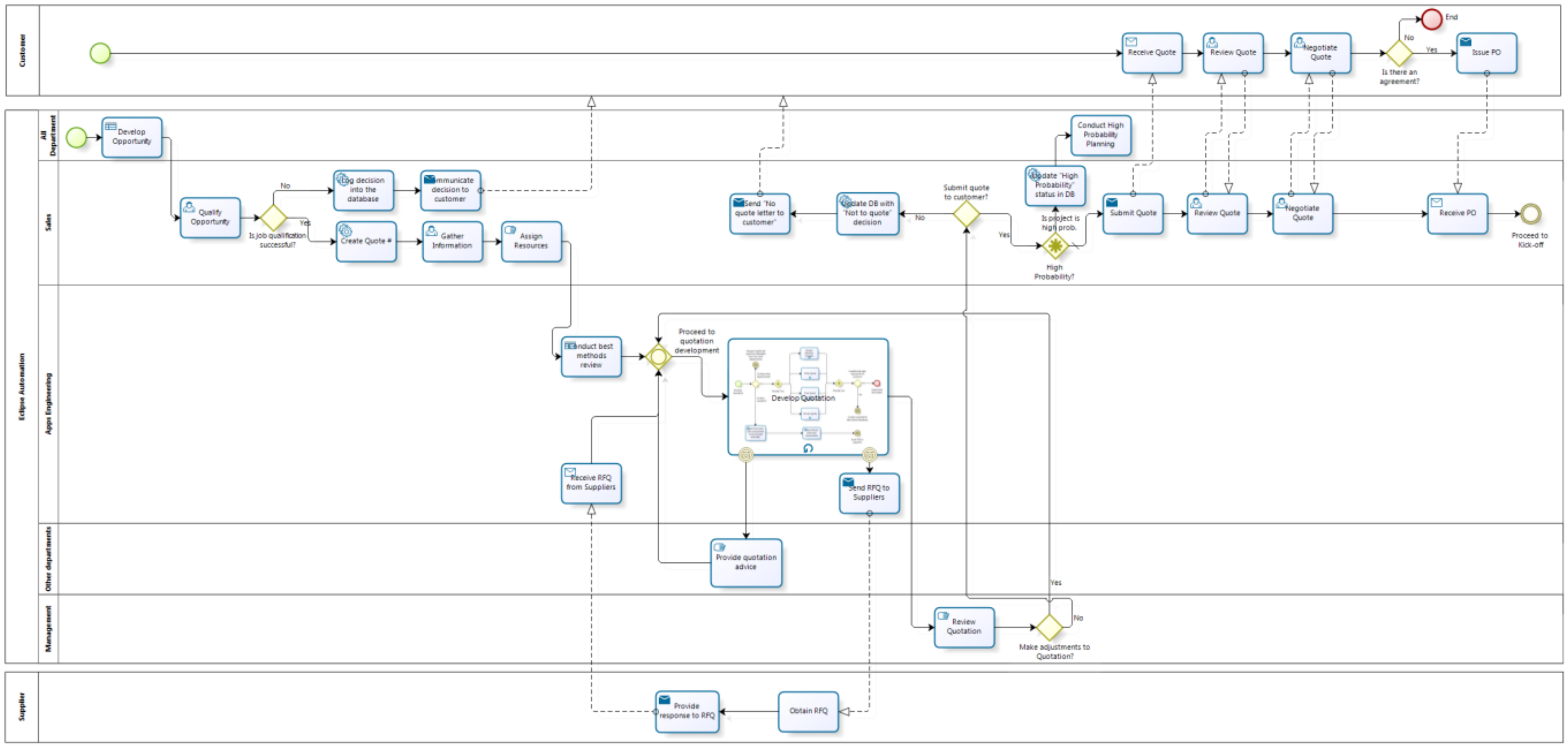
Criteria	Excellent Prospect		Reasonable Prospect		Bad Prospect	
Contact Job Title	Senior Mgt.	10	Middle Mgt.	5	Team member	1
Location	Canada	10	US	5	Others	1
Company Size	> 5,000	10	1,000-5,000	5	< 1,000	1
Industry	Automotive	10	Medical	5	Solar	1
Budget	> 50,000	10	10,000-50,000	5	< 10,000	1

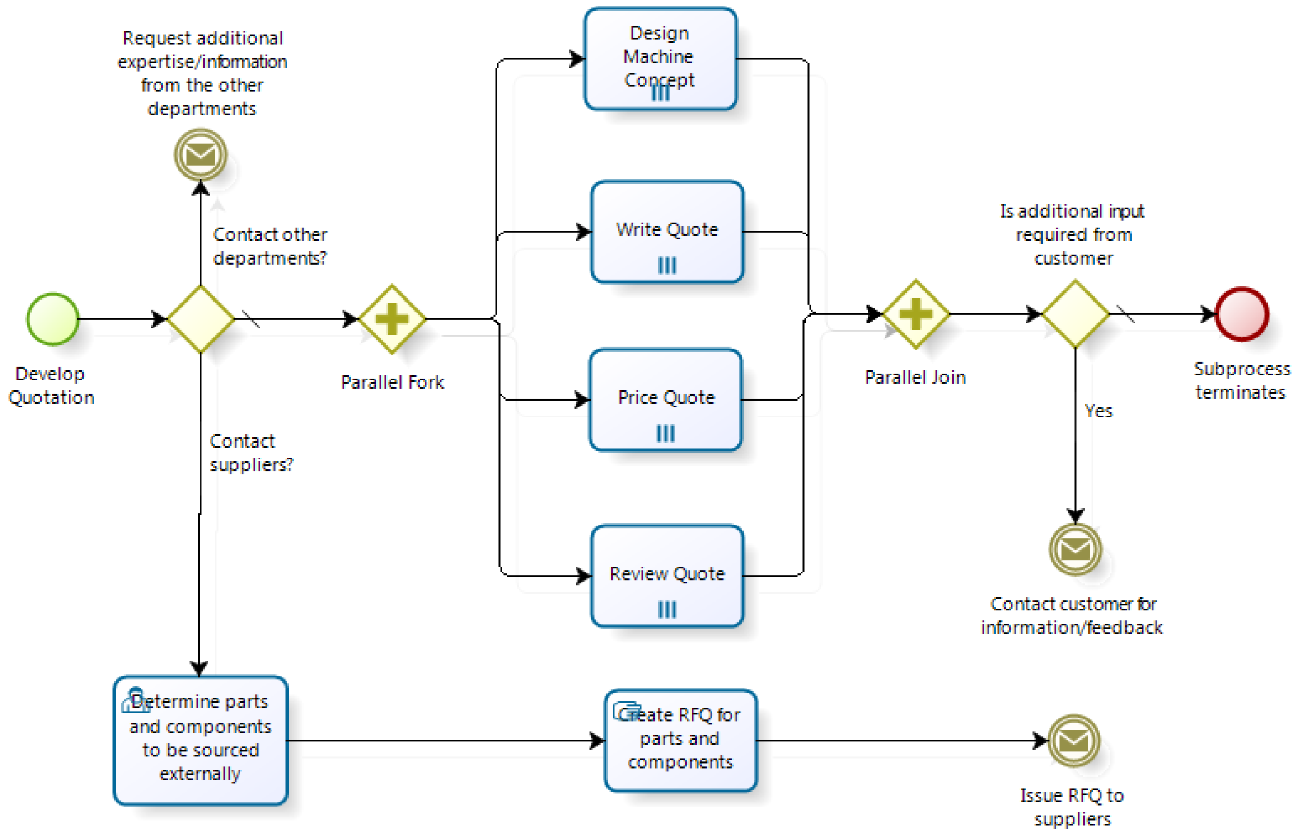
5.12, 5.16	Train staff on using on standard operating procedure		
5.2, 5.5, 5.11, 5.15	Create standard operating procedure on conducting opportunity assessment and using assessment result	5.2., 5.5., 5.11., 5.15	Train staff on how to create and implement SOP
		5.2.1, 5.5.1, 5.11.1, 5.15.1	The SOP should contain the following elements: • Rationale for SOP • Detailed description of procedure – based on best practice/standards • Monitoring actions • Accountability • Corrective Actions • Date of last review or revision date
		5.2.2, 5.5.2, 5.11.2, 5.15.2	Train staff on how to apply the SOP
		5.2.4, 5.5.4, 5.11.4, 5.15.4	Train staff on using the standard operating procedure
		5.2.6, 5.5.6, 5.11.6, 5.15.6	Set up SOP committee to review SOP annually and as need arises
6 - Perform Credit Check			
Level 1		**Level 2**	
6.1	Escalate to manager		
6.2, 6.7, 6.12, 6.27, 6.32	Multi credit agency checks	6.2.1, 6.7.1, 6.12.1, 6.27.1, 6.32.1	Escalate to manager
6.19, 6.20, 6.24, 6.25	Add calendar and task management module to ERP	6.19.1, 6.20.1, 6.24.1, 6.25.1	Train staff on how and when to use the Calendar and task management module
		6.19.3, 6.20.3, 6.24.3, 6.25.3	Institute consequence management program to address negligence
		6.19.4, 6.20.4, 6.24.4, 6.25.4	Add calendar and task management module to ERP with the following features: • User-definable data fields • Quick, easy data entry with automatic field defaults, AutoCorrect and speed entry templates • Progress monitoring and indicators • Ability to set recurring tasks, jobs and projects • Automatic task scheduling • Automatic data backup • Optional task synchronization with Microsoft Outlook

6.38, 6.40	Train staff on credit check task		
7 - Log decision into the ERP			
Level 1		**Level 2**	
7.2	Train staff how to log decision into the database		
8 - Communicate decision to Customer if not quoting			
Level 1		**Level 2**	
8.2, 8.4, 8.6, 8.9	Train staff on task		
8.7	Validate message by peer review		
9 - Gather Information			
Level 1		**Level 2**	
9.5	Train staff on the technology		
9.10	Employ personnel with requisite skill & experience		
9.11, 9.18	Establish non-disclosure agreements to make customer comfortable		
9.12, 9.17, 9.19	Create information elicitation checklist		5.1.10
9.15, 9.21	Add calendar and task management module to ERP		6.19, 6.20, 6.24, 6.25
9.40	Validate data entry		3.9
9.41	Validate information by peer review		
9.45	Escalate to manager		
10 - Assign Resources			
Level 1		**Level 2**	
10.2, 10.6, 10.13, 10.17, 10.21	Contract part time staff		
10.9	Outsource task		
10.11	Prioritize projects		
10.12, 10.16, 10.20	Add ticket management module to ERP	10.12.1, 10.16.1, 10.20.1	Add ticket management module to ERP with the following features: • Maintain accurate resource profiles with groupings, roles etc. • Define attributes for different resource types. E.g. Skills • Define primary & secondary task resource • Integrate application with outlook • Send notifications • Provide utilization & availability report • Forecast resource shortage and surplus

Appendix B

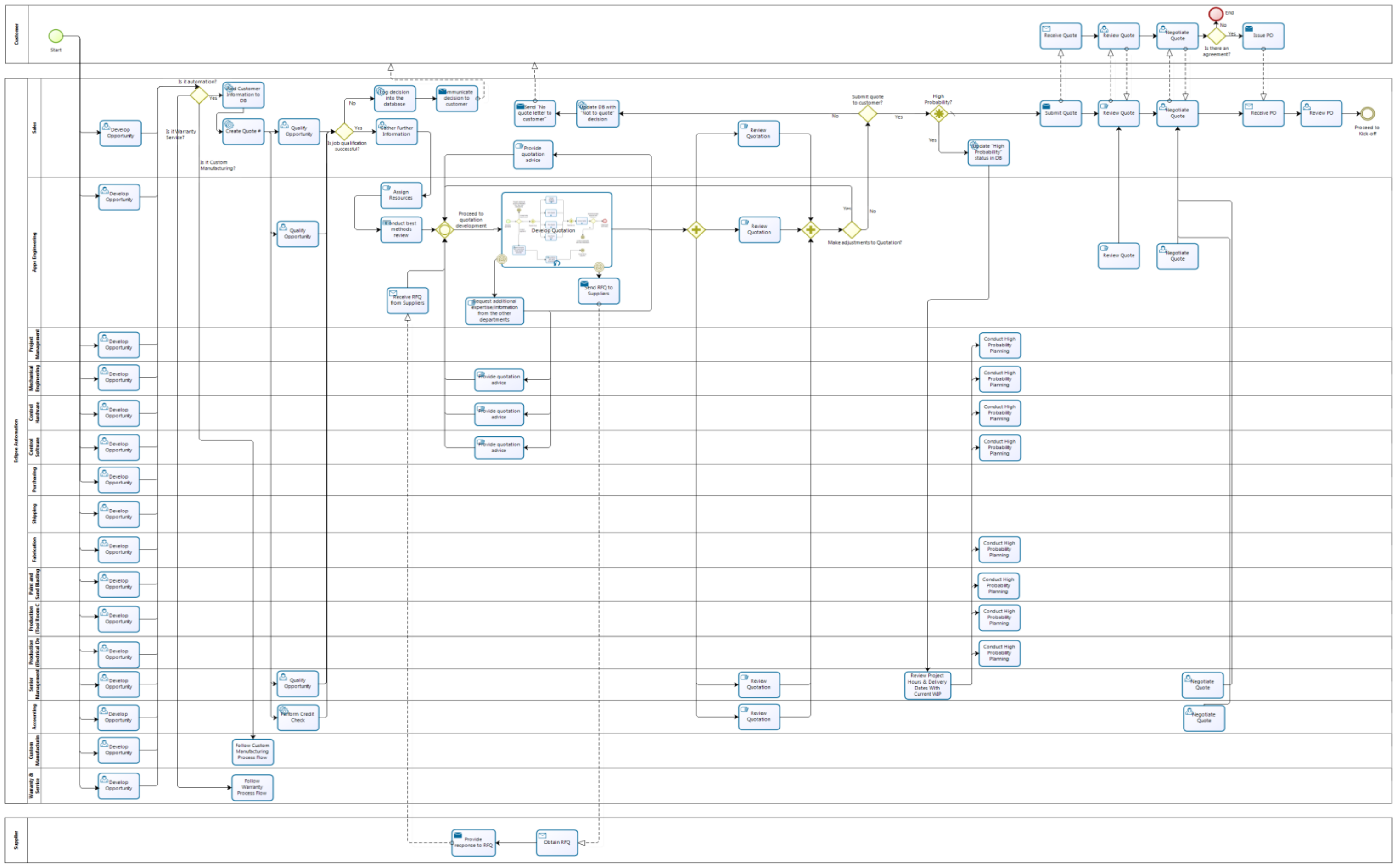
Model developed in iteration 1



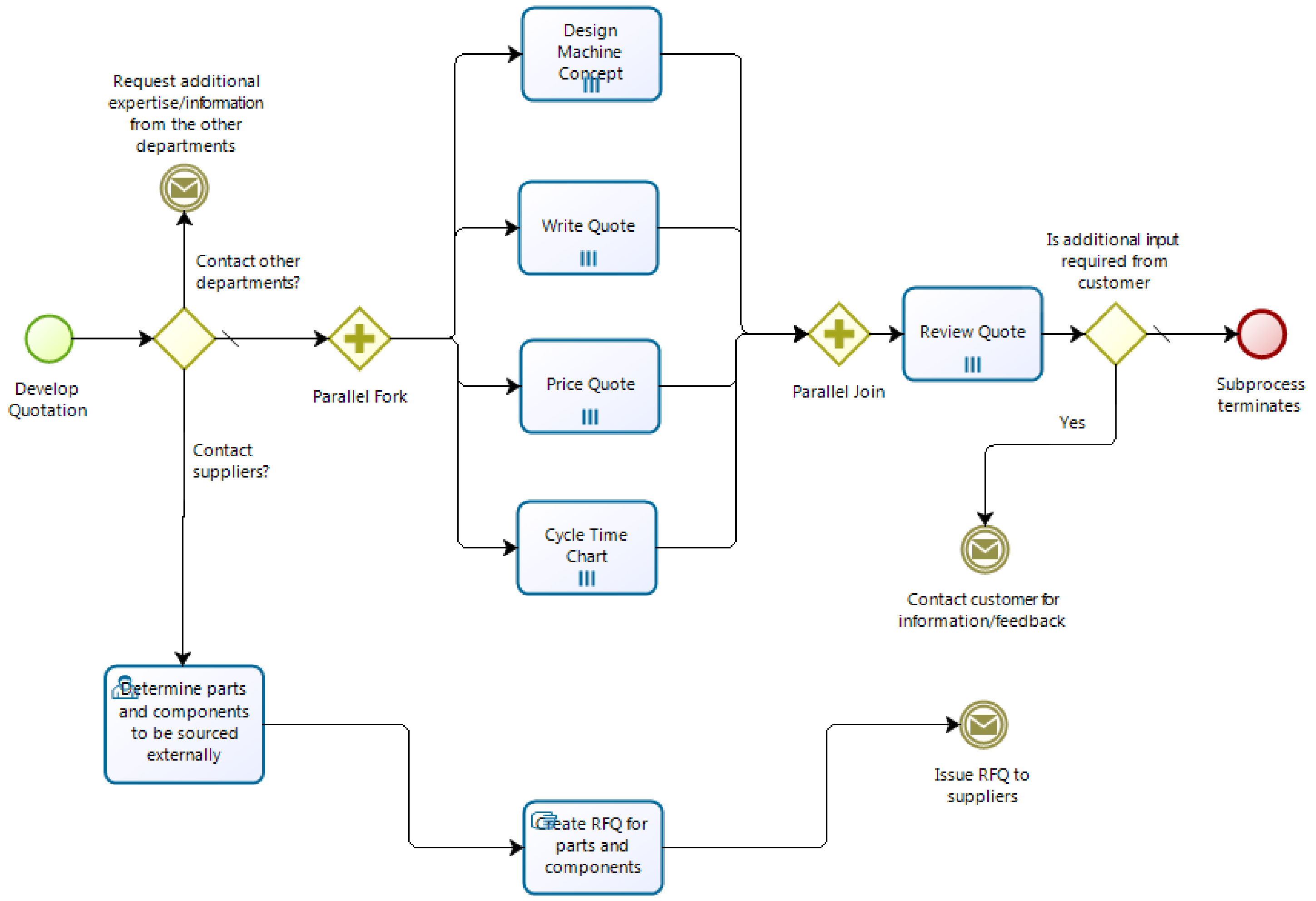

Request additional expertise/information from the other departments
Contact other departments?
Develop Quotation
Parallel Fork
Contact suppliers?
Design Machine Concept
Write Quote
Price Quote
Review Quote
Parallel Join
Is additional input required from customer
Yes
Subprocess terminates
Contact customer for information/feedback
Determine parts and components to be sourced externally
Create RFQ for parts and components
Issue RFQ to suppliers

Appendix C

Model developed in iteration 2



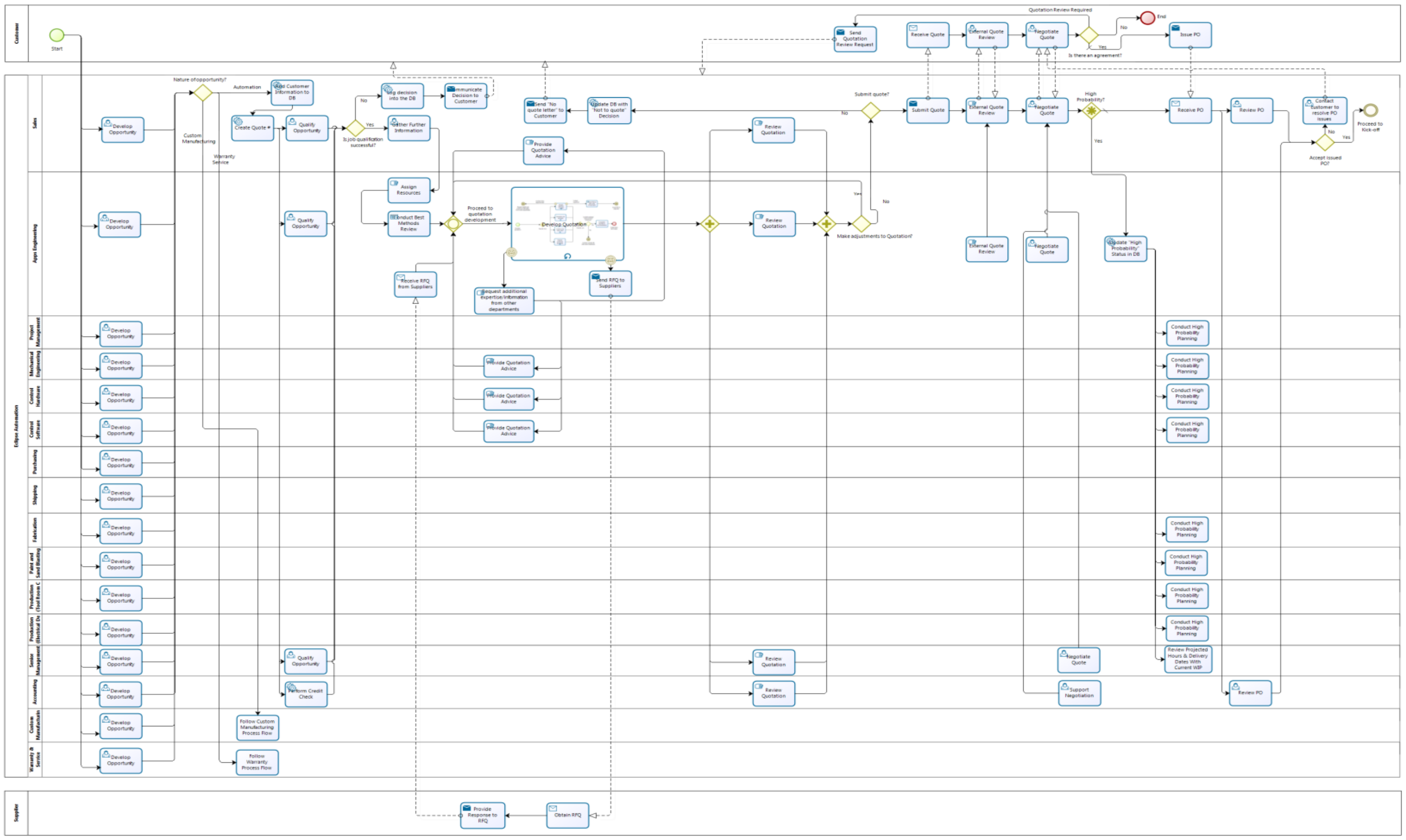
Customer
Start
Receive Quote
Review Quote
Negotiate Quote
Is there an agreement?
No
Yes
End
Issue PO
Eclipse Automation
Sales
Apps Engineering
Project Management
Mechanical Engineering
Control Hardware
Control Software
Purchasing
Shipping
Fabrication
Paint and Sand Blasting
Production (Tool Room C
Production (Electrical De
Senior Management
Accounting
Custom Manufacturin
Warranty & Service
Develop Opportunity
Is it automation?
Yes
Is it Warranty Service?
Is it Custom Manufacturing?
Add Customer Information to DB
Create Quote #
Qualify Opportunity
Is job qualification successful?
No
Yes
Log decision into the database
Communicate decision to customer
Gather Further Information
Send "No quote letter to customer"
Update DB with "Not to quote" decision
Provide quotation advice
Assign Resources
Conduct best methods review
Proceed to quotation development
Develop Quotation
Request additional expertise/information from the other departments
Send RFQ to Suppliers
Receive RFQ from Suppliers
Review Quotation
Make adjustments to Quotation?
Submit quote to customer?
High Probability?
Update "High Probability" status in DB
Submit Quote
Receive PO
Review PO
Proceed to Kick-off
Conduct High Probability Planning
Review Project Hours & Delivery Dates With Current WIP
Perform Credit Check
Follow Custom Manufacturing Process Flow
Follow Warranty Process Flow
Supplier
Provide response to RFQ
Obtain RFQ

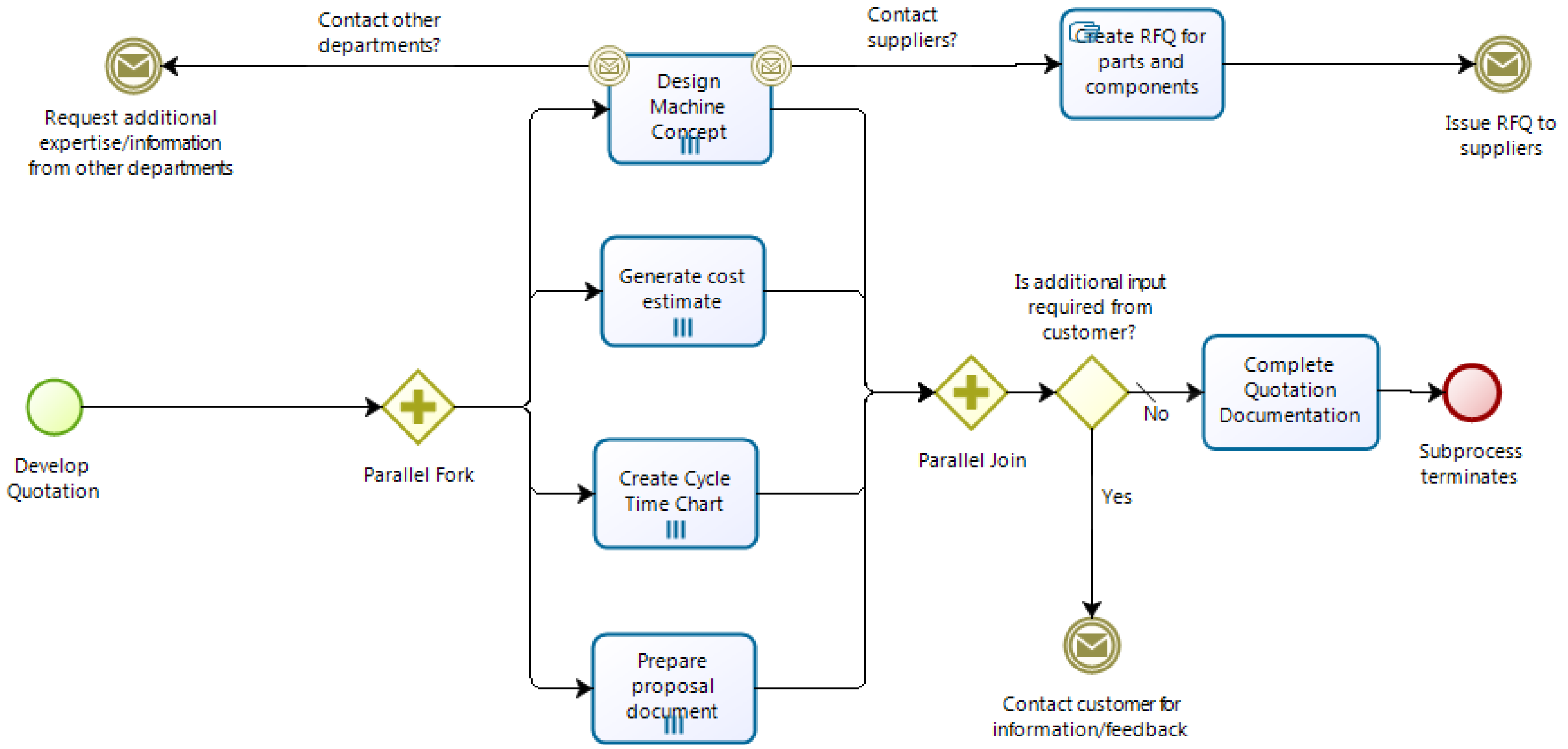


Request additional expertise/information from the other departments
Contact other departments?
Develop Quotation
Contact suppliers?
Parallel Fork
Design Machine Concept
Write Quote
Price Quote
Cycle Time Chart
Parallel Join
Review Quote
Is additional input required from customer
Subprocess terminates
Yes
Contact customer for information/feedback
Determine parts and components to be sourced externally
Create RFQ for parts and components
Issue RFQ to suppliers

Appendix D

Model developed in iteration 3




Contact other departments?
Contact suppliers?
Create RFQ for parts and components
Request additional expertise/information from other departments
Design Machine Concept
Issue RFQ to suppliers
Generate cost estimate
Is additional input required from customer?
Complete Quotation Documentation
No
Develop Quotation
Parallel Fork
Parallel Join
Subprocess terminates
Create Cycle Time Chart
Yes
Prepare proposal document
Contact customer for information/feedback

Appendix E

BPM Tool screening

Creator	Tool name	Meets cost criterion?	Meets ease criterion?	Meets BPMN criterion?
Bizagi	Bizagi	Yes	Yes	Yes
IBM	Rational	No	N/A	N/A
Microsoft	Visio	Yes	Yes	No
Software AG	Aris Express	Yes	Yes	Yes
BonitaSoft	Bonita BPM	Yes	Yes	Yes
Intellivate	IYORO	Yes	Yes	Yes
Lucid Software	Lucid	Yes	Yes	Yes
Visible Systems	Visible Analyst	No	N/A	N/A
The BOC Group	ADONIS	Yes	No	N/A
GeneXus Modeler	GeneXus	Yes	No	N/A
igrafx	igrafx Flowchater	Yes	Yes	Yes
Altova	Umodel	Yes	Yes	Yes
Oracle	Business Process Management (BPM) Suite	No	N/A	N/A
OpenText	Process Suite	No	N/A	N/A
PTC	PTC Windchill	No	N/A	N/A

Appendix F

BPM Tool evaluation

#	Quality Criteria	Parameter	Bizagi		Visio		ArisExpress	
			Creator: Bizagi Version evaluated: 2.6..0.4		**Creator: Microsoft** **Version evaluated: Visio 2010**		**Creator: Software AG** **Version evaluated: 2.4**	
1	Suitability	BPMN modelling	✓		✓		✓	
		BPMN version supported	BPMN 2.0		BPMN 2.0		BPMN 2.0	
		Modularity: Ability to break model into independent modules	✓	Reusesable subprocess can be created	✗		✓	Reusuable fragment of modules be created
		Version Management: Ability to maintain a revision control of model	✗		✗		✗	
2	Interoperability	Export capability	Image, Sharepoint, MS Word, Visio, html, PDF		Image, XML, Sharepoint, MS Word, AutoCAD, html, PDF		Image, PDF	
		Import capability	xpdl, xml, visio		AutoCAD, Image		xml drawing, Image	Limited Visio import
		Hyperlink to external resources	✓		✓		✗	
3	Compliance	BPMN rules enforcement	Available but optional		✗	Commercial plugin are available for BPMN validation	✗	Macros can however be built for syntax checking

4	Maturity	Inclusion in major market reports	• Forrester, 2013 - Listed as strong performer • Gartner, 2010 - Listed in the visionaries quadrant		✓	• Gartner, 2010 - Listed in the leaders quadrant	✓	• Forrester, 2013 - Listed as strong performer • Gartner, 2010 - Listed in the leaders quadrant
5	Learnability	Adequate documentation of tool usage	✓	Link	✓		✓	
		Online forums	✓	Link	✓		✓	Link
		Training courses	✓	Link	✓		✓	Link
6	Usability	Context-sensitive: interface provides context-sensitive help and meaningful feedback when errors occur	✓		✓		✓	
		Familiarity & Navigability: offers recognizable elements and interactions easily understood by the user; users can move around in the application in an efficient way	✓		✓		✓	
		Flexibility: whether the user interface of the software product can be tailored to suit users' personal preferences	✓	Allow usage of extended attributes	✓		×	Allow basic attributes
		Readability: ease with which visual content (such as text dialogs) can be understood	✓		✓		✓	
7	Resource behavior	Licensing cost	Freeware		Commercial	Link	Freeware	

8	Vendor support	Availability of offline support	✗		✓		✗	
		Availability of online support	✓	Paid	✓		✓	http://www.ariscommunity.com/forums/aris-community-support
9	Installability	Specific issues/requirements	✗		✗		✗	
		Operating system requirement	Windows		Windows		Windows	
		Hardware requirement	• Processor: 1 gigahertz (GHz). 32-bit (x86) or 64-bit (x64) • Memory: 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit) • Hard drive: 50 MB available hard disk space • Display: 800 x 600 or higher resolution		• Processor: 1 gigahertz (GHz) or faster 32-bit (x86) or 64-bit (x64) processor • Memory: 1 gigabyte (GB) of RAM for 32-bit (x86) processors or 2 GB of RAM for 64-bit (x64) processors		• Min. screen resolution: 1024x600 pixels • Min. free disc space: 275 MB • Min. free memory (RAM): 256 MB • Recommended free memory (RAM): 512 MB	

Bibliography

Adhikari, P., & Reinhart, B. (2006). Design Rationale Modeling Representations, 1–9.

Akao, Y. (1994). Development History of Quality Function Deployment. *The Customer Driven Approach to Quality Planning and Deployment. Minato, Tokyo 107 Japan: Asian Productivity Organization*, 339.

Akers, D. (2008). Real Reuse for Software Requirements Sharing Between Projects. Retrieved June 09, 2014, from http://www.methodsandtools.com/archive/archive.php?id=68

Alwang, J., Siegel, P., & Jorgensen, S. (2001). Vulnerability: a view from different disciplines, (2001), 1–12. Retrieved from http://siteresources.worldbank.org/SOCIALPROTECTION/Resources/SP-Discussion-papers/Social-Risk-Management-DP/0115.pdf

Ambler, S. (2007). Agile Documentation Strategies. Retrieved from http://www.drdobbs.com/architecture-and-design/agile-documentation-strategies/197003363

Ambler, S. (2008). Active Stakeholder Participation: An Agile Best Practice. Retrieved September 03, 2014, from http://agilemodeling.com/essays/activeStakeholderParticipation.htm

Ambler, S. (2010). Agile/Lean Documentation: Strategies for Agile Software Development. Retrieved September 03, 2014, from http://agilemodeling.com/essays/agileDocumentation.htm

Ambler, S. W. (2013). Best Practices for Agile Lean Documentation.

Anton, A. (1996). Goal-based requirements analysis. *Requirements Engineering, 1996., Proceedings of …*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=491438

ASQ. (2011). Six Sigma Certification - Become Black Belt (CSSBB) Certified | ASQ. Retrieved May 14, 2014, from http://asq.org/cert/six-sigma

Aven, T. (2008). *Risk analysis: assessing uncertainties beyond expected values and probabilities*. *Vasa*. Retrieved from http://medcontent.metapress.com/index/A65RM03P4874243N.pdf

Avison, D., & Fitzgerald, G. (2006). *Information Systems Development: Methodologies, Techniques & Tools* (4th ed.). New York: McGraw-Hill College.

Bashir, M. F., & Qadir, M. A. (2006). Traceability Techniques: A Critical Study. *2006 IEEE International Multitopic Conference*, 265–268. doi:10.1109/INMIC.2006.358175

Beck, K. (1999). *Extreme Programming Explained*.

Berander, P., & Andrews, A. (2005). Requirements prioritization. *Engineering and Managing Software Requirements. Springer Berlin Heidelberg*, 69–94. Retrieved from http://link.springer.com/chapter/10.1007/3-540-28244-0_4

Berg, H. (2010). Risk management: Procedures, methods and experiences. *RISK MANAGEMENT*, *1*(17), 79–95. Retrieved from http://gnedenko-forum.org/Journal/2010/022010/RTA_2_2010-09.pdf

Bernoulli, D. (1954). Exposition of a new theory on the measurement of risk. *Econometrica: Journal of the Econometric Society*, *22*(1), 23–36.

Bernstein, P. L. (1997, July). Against the Gods: The Remarkable Story of Risk. *Journal of Marketing*. doi:10.2307/1251793

Bigelow, J. (1988). Hypertext and CASE. *Software, IEEE*, *5*(2), 23–27. doi:10.1109/52.2007

BIS. (2010). *Business Information Systems: 13th International Conference, BIS 2010, Berlin, Germany, May 3-5, 2010, Proceedings* (p. 305). Springer. Retrieved from http://books.google.com/books?id=TGQA3__sLsIC&pgis=1

Bowen, J., O'Grady, P., & Smith, L. (1990). A constraint programming language for Life-Cycle Engineering. *Artificial Intelligence in Engineering*, *5*(4), 206–220. doi:10.1016/0954-1810(90)90022-V

Bowles, J. (2004). An assessment of RPN prioritization in a failure modes effects and criticality analysis. *Journal of the IEST*, 380–386. Retrieved from http://iest.metapress.com/index/Y576M26127157313.pdf

Braber, F. den, Hogganvik, I., Lund, M. S., Stølen, K., & Vraalsen, F. (2007). Model-based security analysis in seven steps — a guided tour to the CORAS method. *BT Technology Journal*, *25*, 101–117. Retrieved from http://dx.doi.org/10.1007/s10550-007-0013-9

Bradner, S. (1997). RFC 2119. Retrieved from http://www.ietf.org/rfc/rfc2119.txt

BSI. (1991). BS 5760-5:1991 - Reliability of systems, equipment and components. Guide to failure modes, effects and criticality analysis (FMEA and FMECA) – BSI British Standards. Retrieved May 14, 2014, from http://shop.bsigroup.com/ProductDetail/?pid=000000000000256425

Burge, J., Carroll, J., McCall, R., & Mistrik, I. (2008). *Rationale-based software engineering*. Springer.

Burge, J. E., & Brown, Jd. C. (1998). Design Rationale Types and Tools. Retrieved June 10, 2014, from http://web.cs.wpi.edu/Research/aidg/DR-Rpt98.html

Burge, J. E., Hall, K., & Brown, D. C. (2007). Supporting Requirements Traceability with Rationale.

Carbone, T., & Tippett, D. (2004). Project Risk Management Using the Project Risk FMEA. *Engineering Management Journal*, *16*(4). Retrieved from http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=10429247&AN=16605660&h=0ykRyIfVbOUhXZV8NMhRfd8KKGrsNqEt35rz2w9SQNm1qM/rO700V/5uwwU2PoOl9VWeyOo57nKidYMpgB/0RA==&crl=c

Carlson, C. (2012). *Effective FMEAs: Achieving Safe, Reliable, and Economical Products and Processes Using Failure Mode and Effects Analysis*. John Wiley & Sons, Inc.

Carlson, C., Sarakakis, G., Groebel, D. J., & Mettas, A. (2010). Best practices for effective reliability program plans. *Reliability and …*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5448073

Christel, M., & Kang, K. (1992). Issues in requirements elicitation, (September). Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA258932

Dale, N., Siesfeld, G. A., & Cefola, J. (1998). The economic impact of knowledge. *Routledge*.

Dardenne, A., Lamsweerde, A. Van, & Fickas, S. (1993). Goal-directed requirements acquisition. *Science of Computer …*. Retrieved from http://www.sciencedirect.com/science/article/pii/016764239390021G

David, J., & Saaty, D. (2007). Use analytic hierarchy process for project selection. *ASQ Six Sigma Forum Magazine*. Retrieved from http://dschoenherr.fatcow.com/sitebuildercontent/sitebuilderfiles/analytic_hierarchy_process.pdf

DoD. (1980). U.S. Department of Defense, MIL-STD-1629A: Procedures for Performing a Failure Mode Effects and Criticality Analysis, Cancelled in November, 1984.

Dorsey, P. (2000). Top 10 reasons why systems projects fail. *Retrieved February*, 1–9. Retrieved from http://www.rrsg.ee.uct.ac.za/courses/EEE4084F/Reading/Lect21-Dorsey_Top10ReasonsSystemsProjectsFail.pdf

Dutoit, A., McCall, R., Mistrik, I., & Paech, B. (2007). *Rationale management in software engineering*. Retrieved from http://books.google.com/books?hl=en&lr=&id=Ud9pZtSyHKUC&oi=fnd&pg=PR5&dq=Rationale+Management+in+Software+Engineering&ots=sV2h2n8wkc&sig=Gz7qPQwSELuq9__NBkT3hfd9x5o

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *The Academy of Management Review*, *14*(4), 532. doi:10.2307/258557

ESA. (1994). ESA Software Engineering Standards. *ESA Publications Division, ESA PSS-05*(2). Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:ESA+software+engineering+standards#1

Fabian, B., Gürses, S., Heisel, M., Santen, T., & Schmidt, H. (2009). A comparison of security requirements engineering methods. *Requirements Engineering*, *15*(1), 7–40. doi:10.1007/s00766-009-0092-x

Feather, M., Cornford, S., Kiper, J., & Menzies, T. (2006). Experiences using Visualization Techniques to Present Requirements, Risks to Them, and Options for Risk Mitigation. *2006 First International Workshop on Requirements Engineering Visualization (REV'06 - RE'06 Workshop)*, 10–10. doi:10.1109/REV.2006.2

Feng, D., & Eyster, C. (2013). Risk-based requirements management framework with applications to assurance cases. *2013 IEEE Aerospace Conference*, 1–11. doi:10.1109/AERO.2013.6496958

Firesmith, D. (2007). Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them. *Journal of Object Technology*, *6*(1), 17–33. Retrieved from http://www.jot.fm/issues/issue_2007_01/column2/

Galvao, I., & Goknil, A. (2007). Survey of Traceability Approaches in Model-Driven Engineering. *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, 313–313. doi:10.1109/EDOC.2007.42

Gan, L., Xu, J., & Han, B. T. (2011). A computer-integrated FMEA for dynamic supply chains in a flexible-based environment. *The International Journal of Advanced Manufacturing Technology*, *59*(5-8), 697–717. doi:10.1007/s00170-011-3526-9

Garvey, P. R. (2008). *Analytical Methods for Risk Management: A Systems Engineering Perspective* (p. 288). CRC Press. Retrieved from http://books.google.ca/books/about/Analytical_Methods_for_Risk_Management.html?id=dCLXq78GDVgC&pgis=1

Goldsmith, R. F. (2004). *Discovering Real Business Requirements for Software Project Success* (p. 215). Artech House.

Gotel, O., & Finkelstein, A. (1994). An analysis of the requirements traceability problem. *Requirements Engineering, …*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=292398

Gottesdiener, E. (2005). The Software Requirements: Memory Jogger: a Pocket Guide to Help Software and Business Teams Develop and Manage Requirements. *Goal/QPC*.

Greenspan, S. J., & McGowan, C. . (1978). Structuring Software Development for Reliability. *Microelectronics and Reliability*.

Gruber, T. R. (1990). Model-based Explanation of Design Rationale. *Proceedings of the AAAI-90 Explanation Workshop, Boston*.

Gruber, T. R., & Russell, D. M. (1996). Generative design rationale: beyond the record and replay paradigm. In T. P. Moran & J. M. Carroll (Eds.), (pp. 323–349). Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Hazzan, O., & Dubinsky, Y. (2009). *Agile software engineering*. doi:10.1007/978-1-84800-199-2

Heindl, M., & Biffl, S. (2005). A case study on value-based requirements tracing. *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering - ESEC/FSE-13*, 60. doi:10.1145/1081706.1081717

Heindl, M., & Biffl, S. (2006). Risk management with enhanced tracing of requirements rationale in highly distributed projects. *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner - GSD '06*, 20. doi:10.1145/1138506.1138512

Hekmatpanah, M., Shahin, A., & Ravichandran, N. (2011). The application of FMEA in the oil industry in Iran : The case of four litre oil canning process of Sepahan Oil Company, *5*(8), 3019–3027. doi:10.5897/AJBM10.1248

Henney, K. (2007). Iterative and incremental development explained. TechTarget. Retrieved from http://searchsoftwarequality.techtarget.com/news/1284193/Iterative-and-incremental-development-explained

Herrmann, A., & Paech, B. (2007). MOQARE: misuse-oriented quality requirements engineering. *Requirements Engineering*, *13*(1), 73–86. doi:10.1007/s00766-007-0058-9

Herrmann, A., & Paech, B. (2009). Practical challenges of requirements prioritization based on risk estimation. *Empirical Software Engineering*, *14*(6), 644–684. doi:10.1007/s10664-009-9105-0

Highsmith, J. (2013). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems* (p. 392). NY, USA: Dorset House Publishing Co., Inc.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=947100

Hull, E., Jackson, K., & Dick, J. (2005). Requirements Traceability. In *Requirements Engineering* (pp. 1–19). London: Springer London. doi:10.1163/9789004265820_002

IEC. (2006). IEC 60812 - Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA) | IEC Webstore | Publication Abstract, Preview, Scope. Retrieved May 14, 2014, from http://webstore.iec.ch/webstore/webstore.nsf/standards+ed/IEC 60812 Ed. 2.0?OpenDocument

IEC. (2008). Analysis Techniques for System Reliability: Procedure for Failure Mode and Effects Analysis (FMEA)., (26).

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). Los Alamitos. *CA: IEEE Computer Society*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:IEEE+Standard+Glossary+of+Software+Engineering+Terminology+(IEEE+Std+610.12-1990)#0

IEEE. (1998). IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998*, 1–40. doi:10.1109/IEEESTD.1998.88286

IHI. (2013). Institute for Healthcare Improvement: Risk Priority Number (from Failure Modes and Effects Analysis). Retrieved May 17, 2014, from http://www.ihi.org/resources/Pages/Measures/RiskPriorityNumberfromFailureModesandEffectsAnalysis.aspx

IMCA. (2002). Guidance on Failure Modes & Effects Analyses (FMEAs), (April).

ISO. (2009). ISO/TS 16949:2009 - Quality management systems -- Particular requirements for the application of ISO 9001:2008 for automotive production and relevant service part organizations. Retrieved May 14, 2014, from http://www.iso.org/iso/catalogue_detail?csnumber=52844

ISO 31000. (2009). ISO 31000 Risk Management Definitions in Plain English. Retrieved March 24, 2014, from http://www.praxiom.com/iso-31000-terms.htm

Jaber, K., Sharif, B., & Liu, C. (2013). A Study on the Effect of Traceability Links in Software Maintenance, *1*.

Kaindl, H. (1993). The Missing Link in Requirements Engineering. *SIGSOFT Softw. Eng. Notes*, *18*(2), 30–39. doi:10.1145/159420.155836

Kajko-Mattsson, M. (2008). Problems in agile trenches. *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '08*, 111. doi:10.1145/1414004.1414025

Kaplan, S., & Garrick, B. J. (1981). On The Quantitative Definition of Risk. *Risk Analysis*, *1*(1), 11–27. doi:10.1111/j.1539-6924.1981.tb01350.x

Karlsson, J., & Ryan, K. (1997). A cost-value approach for prioritizing requirements. *Software, IEEE*, *14*(5), 67–74.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., & Carriere, J. (1998). The architecture tradeoff analysis method. *... , 1998. ICECCS'98. ...*, *98*(c). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=706657

Kirschner, P. A., Buckingham-Shum, S. J., & Carr, C. S. (2003). Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making. *London: Springer-Verlag*.

Kovitz, B. (2003). Hidden skills that support phased and agile requirements engineering. *Requirements Engineering*, *8*(2), 135–141. doi:10.1007/s00766-002-0162-9

Kruchten, P., Capilla, R., & Dueas, J. C. (2009). The Decision View's Role in Software Architecture Practice. *Software, IEEE*, *26*(2), 36–42. doi:10.1109/MS.2009.52

Lauber, R. J. (1982). Development Support Systems. *Computer*, *15*(5), 36–46. doi:10.1109/MC.1982.1654023

Layton, M. C. (2012). Agile Project Management for Dummies. Retrieved April 29, 2014, from http://www.dummies.com/how-to/content/agile-management-communication-methods.html

Lee, J. (1989). Decision representation language (DRL) and its support environment, (325). Retrieved from http://dspace.mit.edu/handle/1721.1/41499

Lee, J. (1991). Extending the Potts and Bruns model for recording design rationale. *Proceedings of the 13th International Conference on Software Engineering (ICSE '13), IEEE Computer Society Press, Los Alamitos, CA*, 114 – 125.

Leffingwell, D. (2011). Agile Software Development with Verification and Validation in High Assurance and Regulated Environments. *Rally Software Development Corp*.

Leffingwell, D. (2011). Agile software requirements. *... Requirements Practices for Teams, ...*. Retrieved from http://www.scalingsoftwareagilityblog.com/wp-content/uploads/2010/08/0321635841_leffingwell_comp.pdf

Leffingwell, D., & Widrig, D. (2003). *Managing software requirements: a use case approach*. Addison-Wesley Professional.

Lehtola, L., Kauppinen, M., & Kujala, S. (2004). Requirements Prioritization Challenges in Practice, 497–508.

Leveson, N. G. (2000). Intent specifications: an approach to building human-centered specifications. *IEEE Transactions on Software Engineering*, *26*(1), 15–35. doi:10.1109/32.825764

Lin, L., Ince, D., Moffett, J., Hall, W., & Mk, M. K. (2003). Introducing Abuse Frames for Analysing Security Requirements.

Macesker, B., Myers, J., & Guthrie, V. (2002). Quick-reference Guide to Risk-based Decision Making (RBDM): A Step-by-step Example of the RBDM Process in the Field. *... Http://www. Au. Af. Mil/au ...*. Retrieved from http://www.au.af.mil/AU/AWC/AWCGATE/uscg/risk-qrg.pdf

Magsarjav, U. (2004). Requirements Documents Evolution and Synchronization with Activities in the Refined Requirements Generation Model. Retrieved from http://scholar.lib.vt.edu/theses/available/etd-09112004-210405/

Marina Jirotka, Joseph A. Goguen, Andrew F. Monk, B. R. G. (1994). Requirements Engineering: Social and Technical Issues (Computers and People). Retrieved May 19, 2014, from http://www.amazon.com/Requirements-Engineering-Social-Technical-Computers/dp/0123853354

Maynard-Zhang, P., Kiper, J., & Feather, M. (2005). Modeling uncertainty in requirements engineering decision support. *... Engineering Decision Support of ...*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.3695&rep=rep1&type=pdf

Miller, M. T., & Chavez, F. (2002). FGS requirement rationale [Flight Guidance System]. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st* (Vol. 2, pp. 13D1–1–13D1–7 vol.2). doi:10.1109/DASC.2002.1053012

Modarres, M. (2006). *Risk analysis in engineering: techniques, tools, and trends*. Retrieved from http://books.google.com/books?hl=en&lr=&id=ErjFzRWSne8C&oi=fnd&pg=PA1&dq=Risk+analysis+in+engineering+:+techniques,+tools,+and+trends&ots=oqLd_-PnYS&sig=5VEIrdabKUrDXxajdcYvUyhiN7k

Murray, L., & Griffiths, A. (2002). Requirements traceability for embedded software—An industry experience report. *Int. Conf. on Software ...*. Retrieved from http://www.actapress.com/PaperInfo.aspx?PaperID=24411

Mylopoulos, J., Chung, L., & Nixon, B. (1992). Representing and using nonfunctional requirements: A process-oriented approach. *Software Engineering, IEEE ...*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=142871

NORSOK. (2001). *NORSOK STANDARD Risk and emergency preparedness analysis*.

Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: a roadmap. *Proceeding ICSE '00 Proceedings of the Conference on The Future of Software Engineering*, 35 – 46.

Orr, K., Summit, C. C., Development, A. S., Bayer, S., Street, W., Development, R. S., … Explained, E. P. (2001). Chapter 23 . Adaptive Software Development, 173–179.

Paetsch, F., Eberlein, a., & Maurer, F. (2003). Requirements engineering and agile software development. *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, 308–313. doi:10.1109/ENABL.2003.1231428

Pinheiro, F. A. C. (2004). Requirements Traceability. In J. C. S. do P. Leite & J. H. Doorn (Eds.), *Perspectives on Software Requirements* (pp. 91–113). US: Springer. doi:10.1007/978-1-4615-0465- 8_5

Ramesh, B., Stubbs, C., Powers, T., & Edwards, M. (1997). Requirements traceability: Theory and practice. *Annals of Software Engineering*, *3*(1), 397 – 415.

Ramzan, M., Jaffar, M., & Shahid, A. (2011). Value based Intelligent requirement prioritization (VIRP): expert driven fuzzy logic based prioritization technique. *International Journal of Innovative …*, *7*(3), 1017–1038. Retrieved from http://www.ijicic.org/ijicic-09-0765.pdf

Rechtin, E. (1991). Systems Architecting: Creating & Building Complex Systems. Retrieved May 09, 2014, from http://www.amazon.com/Systems-Architecting-Creating-Building-Complex/dp/0138803455

Respect-IT. (2007). A KAOS Tutorial, 1–46.

Rosa, E. a. (1998, January). Metatheoretical foundations for post-normal risk. *Journal of Risk Research*. doi:10.1080/136698798377303

Royce, W. W. (1987). Managing the Development of Large Software Systems: Concepts and Techniques. In *Proceedings of the 9th International Conference on Software Engineering* (pp. 328–338). Los Alamitos, CA, USA: IEEE Computer Society Press. Retrieved from http://dl.acm.org/citation.cfm?id=41765.41801

Rubin, E., & Rubin, H. (2010). Supporting agile software development through active documentation. *Requirements Engineering*, *16*(2), 117–132. doi:10.1007/s00766-010-0113-9

Ruhe, G., Eberlein, A., & Pfahl, D. (2002). Quantitative WinWin: a new method for decision support in requirements negotiation. *… of the 14th International Conference on …*, 159–166. Retrieved from http://dl.acm.org/citation.cfm?id=568789

Saaty, T. (1990). How to make a decision: the analytic hierarchy process. *European Journal of Operational Research*. Retrieved from http://www.sciencedirect.com/science/article/pii/037722179090057I

SEI. (2000). *CMMI for Systems Engineering/Software Engineering, Version 1.02, Staged Representation (CMMI-SE/SW, V1.02, Staged)* (Vol. 02). Software Engineering Institute. Retrieved from http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5263

Shore, J., & Warden, S. (2007). *The Art of Agile Development*. O'Reilly Media, Inc. Retrieved from http://books.google.com/books?hl=en&lr=&id=g_ji7cRb--UC&oi=fnd&pg=PR9&dq=The+Art+of+Agile+Development&ots=vnHBDnTA9x&sig=P8juL4TmRrCfqDA1UxyI-1hSxqE

Sillitti, A., & Succi, G. (2005). Requirements engineering for agile methods. *Engineering and Managing Software Requirements*. Retrieved from http://link.springer.com/chapter/10.1007/3-540-28244-0_14

Sindre, G., & Opdahl, A. (2001). Capturing security requirements through misuse cases. *NIK 2001, Norsk Informatikkonferanse 2001, Http:// …*. Retrieved from http://www.nik.no/2001/21-sindre.pdf

Sindre, G., & Opdahl, A. (2004). Eliciting security requirements with misuse cases. *Requirements Engineering*, *10*(1), 34–44. doi:10.1007/s00766-004-0194-4

Sommerville, I. (2004). *Software Engineering* (7th ed.). Pearson Addison Wesley.

Spanoudakis, G. (2002). Plausible and Adaptive Requirement Traceability Structures. *In the Proceedings of SEKE '02*.

Spielvogel, J. J. (2014). *Western Civilization: A Brief History*. *The History Teacher* (8th ed., Vol. 13, p. 442). Boston : Wadsworth/Cengage Learning. doi:10.2307/491694

Stølen, K. (2011). The CORAS Method.

Stumpf, S. (1997). Argumentation-based design rationale-the sharpest tools in the box. *Computer Science Department, University College London*. Retrieved from http://openaccess.city.ac.uk/223/

Syed, M. R., & Syed, S. N. (2008). *Handbook of research on modern systems analysis and design technologies and applications*. Information Science Reference - Imprint of: IGI Publishing. Retrieved from http://www.cs.uu.nl/docs/vakken/me/downloads/2008 - Handbook - Meta-modeling for situational analysis and design methods.pdf

Tang, A., Babar, M. A., Gorton, I., & Han, J. (2006a). A survey of architecture design rationale. *Journal of Systems and Software*, *79*(12), 1792–1804. doi:10.1016/j.jss.2006.04.029

Tang, A., Babar, M., Gorton, I., & Han, J. (2006b). A survey of architecture design rationale. *Journal of Systems and Software*, *79*(12), 1792–1804. doi:10.1016/j.jss.2006.04.029

Tsumaki, T., & Tamai, T. (2005). A Framework for Matching Requirements Engineering Techniques to Project Characteristics and Situation Changes, 44–58.

Turban, B. (2013). *Rationale Management and Traceability in Detailed Discussion* (pp. 159–258). Wiesbaden: Springer Fachmedien Wiesbaden. doi:10.1007/978-3-8348-2474-5

Turk, D., France, R., & Rumpe, B. (2002). Limitations of agile software processes. *… and Agile Processes in Software …*. Retrieved from http://www4.in.tum.de/publ/papers/XP02.Limitations.pdf

Turk, D., Robert, F., & Rumpe, B. (2005). Assumptions underlying agile software-development processes. *Journal of Database Management (JDM …*. Retrieved from http://www.igi-global.com/article/journal-database-management-jdm/3342

Weske, M. (2007). *Business Process Management: Concepts, Languages, Architectures* (Vol. 2007, p. 368). Springer.

Westfall, L. (2006a). Bidirectional Requirements Traceability. *White Paper, The Westfall Team, Dallas*. Retrieved from http://www.westfallteam.com/Papers/Bi-Directional Requirements Traceability updated 2009 03 18.pdf

Westfall, L. (2006b). Software Requirements Engineering: What, Why, Who, When, and How. *The Westfall Team*, (2004).

Wiegers, K. (2000). Describes 10 Requirements Traps to Avoid. Retrieved May 16, 2014, from http://www.processimpact.com/articles/reqtraps.html

Wiegers, K. E. (2009). *Software requirements*. O'Reilly Media, Inc.

Wright, S. (1991). Requirements Traceability - What? Why? and How. In *Proc. of the Colloquium on Tools and Techniques for*.

Youssef, N. F., & Hyman, W. A. (2010). Risk Analysis: Beyond Probability and Severity. Retrieved March 28, 2014, from http://www.mddionline.com/article/risk-analysis-beyond-probability-and-severity