RISK MANAGEMENT IN SOFTWARE DEVELOPMENT

By

MARIET LABUSCHAGNE

submitted in part fulfilment of the requirements
for the degree of

MASTER OF SCIENCE

in the subject

INFORMATION SYSTEMS

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: MRS M G MILLER
CO-SUPERVISOR: MRS E SMITH

JANUARY 2000

ABSTRACT

This dissertation discusses risk management in the context of software development. It commences by investigating why so many software development projects fail. It then focuses on approaches to software development that emerged as attempts to improve the success rate. A common shortcoming to these approaches is identified, namely that they only cater for the tasks that need to be done, ignoring possible unexpected problems. After having motivated the need for risk management, the framework for a risk management methodology is discussed, outlining the steps in the risk management process. Decision-making guidelines and best practices follow, as well as a discussion about the way they should be implemented as part of the risk management effort. Guidelines are provided for the implementation of risk management as part of software development. Finally, the risks that may cause the failure of the implementation of risk management are identified and guidelines provided to address them.

TABLE OF CONTENTS

FIGURES

TABLES

ABBREVIATIONS

CMM	Capability Maturity Model
CRM	Continuous Risk Management
DoD	United States Department of Defence
ISO	International Standards Organisation
IVR	Interactive Voice Response
NIP	National Infrastructure Program
RAMP	Risk Assessment and Management Program
RE	Risk Exposure
RIT	Risk information template
ROI	Return on investment
RRB	Risk review board
RUP	Rational Unified Process
SDLC	Software Development Life Cycle
SDP	Software development project
SE	Software Engineering

SEE	Software Engineering Environment
SEI	Software Engineering Institute of Carnegie Mellon University
SRE	Software Risk Evaluation
TPM	Technical performance measurement
TRM	Team Risk Management
UML	Unified Modelling Language

1 INTRODUCTION

Most tasks that we attempt usually have some potential problems associated with them which, if these problems materialise, may cause the execution of the task to result in failure. These "problems" are called risks, and we usually try to proactively address them to prevent them from causing task failure. This process of identifying risks with the aim to at best prevent them, or at least to minimise them, is called risk management.

Risk management has been exercised in mature engineering disciplines for centuries. In 1547, Michelangelo was given the task of raising the dome of the St. Peter's cathedral. This task was fraught with risks such as possible materials failure, human error, as well as the potential collapse of the dome. However, for each of these major risks, Michelangelo prepared an alternative plan that could be implemented if it became necessary. He therefore identified the possible problems up-front, spending time to devise contingency plans that would ensure, or at least enhance, the chances of success of his project.

If we look around us, we find that risk management is implemented routinely in a number of areas such as financial planning, medical and other insurance, environmental programs, construction engineering, the aviation industry, and many others. But what about software development, the ultimate risky business in this information age? Do we practice risk management in the software development industry?

One must keep in mind that the software development industry is very young in comparison to other mature disciplines such as construction engineering. People have been building bridges for centuries, while computers have only been around for the last fifty years. In addition, the computer industry has changed rapidly during the past 20 years with new hardware as well as software technology emerging every few years, and sometimes even every few months. It is debatable whether we can call software development a mature discipline, since we have yet to reach a steady state where time can be devoted only to maturing the existing processes, instead of keeping up with new advancements in the computer industry. However, even though computer technology is still rapidly evolving, we have to spend time maturing the processes that will enhance the chances of success in software development projects (abbreviated SDP).

One of the activities that will increase the chances of SDPs being concluded successfully is risk management. The need for risk management in SDPs, therefore, provided the primary motivation for this study.

1.1 BACKGROUND TO THE STUDY

Do project managers practice risk management in SDPs? The answer to this question is yes, but only to a certain extent – risk management is practised by some organisations when doing software development, but the majority of software development still takes place without any comprehensive risk management.

The reason why risk management is not always implemented as part of the software development effort is because project managers are not always clear as to what is required to manage risks. Without a clear understanding of the requirements and benefits of proper risk management, the usefulness of such an exercise will not be realised. We identified, therefore, the need to provide an overview of the whole risk management process, in order to assist project managers in understanding the basics of risk management. Furthermore, we also identified the need to provide guidelines to project managers as to how to implement risk management in software development.

While doing research for this dissertation, we found a wealth of books regarding risk management in other disciplines such as engineering and insurance. However, risk management in software development is a fairly new field that only really came into existence in the late 1980's, early 1990's. We found a number of sources focussing on the theory behind risk management, with very little information about practical experiences in implementing risk management in SDPs. A number of articles provided insight into the implementation of certain parts of the risk management methodology, and we made use of these sources to point out how to implement different parts of the risk management methodology. The Software Engineering Institute (abbreviated SEI) of the Carnegie Mellon University was established in order to do research and to provide guidance to the software development teams of the US Department of Defence (abbreviated DoD) in terms of good software development processes to follow to enhance the chances of success. Most of the groundbreaking work in the field of risk management with regard to software development had been done by the SEI, and we made extensive use of these sources in writing this dissertation.

In addition to referring to publications about risk management in software development, we also observed how risk management is implemented in some South African companies. We refer to these results mainly in terms of examples highlighting problems experienced in SDPs and how these could have been avoided if proper risk management had been exercised.

1.2 PROBLEM DEFINITION

Project managers in the software industry do not always know how to go about implementing risk management. It is not necessary for software project managers to be experts at risk management in order to manage the risks to their projects effectively.

- Why do we need to manage the risks to SDPs?
- What do project managers need to know about risk management before they can start managing the risks in their SDPs?
- How do we make decisions about risks under uncertain conditions?
- How must they go about implementing risk management?
- What are the potential benefits to be reaped from risk management?

This dissertation addresses the needs for and benefits of risk management in SDPs. It also provides an overview of what project managers need to know with regard to risk management, as well as guidelines as to how to go about implementing risk management in SDPs for the first time.

1.3 TERMINOLOGY USED IN THIS DISSERTATION

The following terminology applies and can be used as a frame of reference throughout the rest of the dissertation.

1.3.1 Software development project

A software development project entails the implementation of company resources for a relatively short-term objective that has been established to complete specific goals and objectives. All references to "project" in this dissertation refer to a SDP, unless stated otherwise.

1.3.2 Risk

A risk to a project is anything that can cause the project to fail.

1.3.3 Risk exposure

Risk exposure is a measure of the probability and consequence of not achieving a defined project goal.

1.3.4 Risk management

Risk management is the process of identifying risks, analysing these risks, handling the risks and learning from past experiences in managing risks.

1.3.5 Risk handling capability

The risk handling capability of an organisation defines the maximum risk exposure that the organisation is equipped to handle – i.e. projects with a higher risk exposure than the risk handling capability of the organisation should not be attempted.

1.4 OVERVIEW OF CHAPTERS IN THE DISSERTATION

In chapter two, we highlight the reasons why we need to manage the risks in SDPs. We derive these reasons mainly from statistics based on the success and failure rate of SDPs in practice. We pay special attention to the efforts and attempts (other than risk management) of the software industry to improve the chances of success of

SDPs. However, the attempts of the software industry to improve the success rate of SDPs have not been adequately successful, since the failure rate of SDPs is still unacceptably high. Chapter two concludes with underlining the importance of managing the risks of SDPs.

Chapter three provides the background information that project managers need to be aware of with regard to risk management. We define risk and risk management in the context of software development. How risk management in software development as we know it today came about is discussed, with attention being given to early risk management initiatives such as the Spiral Model of Boehm. Chapter three concludes with the differences between traditional project management and risk management.

Chapter four is devoted to presenting a basic framework for a risk management methodology that can be used to implement risk management in SDPs. We focus on the four steps of the proposed framework, namely risk assessment, risk analysis, risk handling and learning from experience. We devote attention to specific methods and tools that can be used in implementing the different steps of the proposed risk management framework. Chapter four, together with chapter three, therefore addresses issues and information with regard to risk management that project managers need to be aware of before attempting to implement risk management.

We constantly need to make informed decisions about risks when implementing risk management. In Chapter five, we look at decision-making structures and principles with regard to risks. We observe how the maturity of a company, as defined by the Capability Maturity Model (abbreviated CMM), should influence the decision-making regarding risks. Chapter five also provides guidelines as to how to reach a consensus when decisions are made with regard to risks, as well as how to deal with uncertainty when making decisions.

Chapter six provides guidelines to project managers on how to go about implementing risk management in software development for the first time. We conclude with the potential benefits to be achieved when implementing risk management in SDPs, stressing the importance of risk management to project success once again. This chapter addresses issues that project managers need to be aware of before implementing risk management.

This dissertation culminates in chapter 7. In this last chapter, the author summarises the research that has been undertaken thus far, and gives some reflections on possibilities for further research.

2 THE NEED FOR MANAGING RISKS IN SOFTWARE DEVELOPMENT PROJECTS

2.1 INTRODUCTION

Failure of a software development project can have significant negative consequences. Apart from the fact that resources such as time and money has been wasted, the failure of a software project can also lead to opportunities being lost. This chapter is devoted to motivating the need for managing risks as part of the software development process. It gives attention to the success or lack of success in SDPs, and discusses reasons for the failure or lack of success. It mentions some of the practices in the Software Development Life Cycle (abbreviated SDLC) that have evolved as attempts to increase the success rate of SDPs.

2.2 THE SUCCESS OR FAILURE OF SOFTWARE DEVELOPMENT PROJECTS

We start this discussion by considering the criteria for classifying a SDP as being a success and then continue with a discussion of the reasons for project failure.

2.2.1 Project success

A successful SDP can be defined as one that is developed within the original planned time and budget, satisfying all the user requirements. Project success also includes the completion of the project

- at the proper performance or specification level;
- with acceptance by the customer / user;
- with minimum or mutually agreed upon scope changes;
- without disturbing the main flow of the organisation;
- without changing the corporate culture [Kerzner, 1995].

The value of the final product of the project can be expressed in terms of opportunity and benefit. An opportunity of a SDP may be that it may save time by cutting out manual procedures in a business process. An example of this is Computicket's Internet website. One can book a movie ticket over the Internet, using one's credit card as method of payment, saving one the time and effort of having to be at the cinema early to stand in a queue to obtain tickets. A possible benefit of this project is that higher client satisfaction can be achieved by saving the client time, and therefore money, this potentially causing the client to go to the movies more often. The sales agents on duty will also be able to perform a better service, since they personally have to deal with fewer customers.

SDP costs are addressed by variable cost and risk in units of money, time, and effort necessary for project delivery [Lister, 1997]. A project may add value, but for the project to be a success this added value must justify the costs.

An example of a project where the value did not justify the cost, is a project that was initiated by a South African airline to use interactive voice response (abbreviated IVR) technology as part of its reservation system. IVR technology usually provides the user with a menu that is read out verbally, allowing a user to choose an option by providing input using his telephone keypad. Depending on the input of the user, a specific voice response is given. This project was initiated by the airline to optimise the productivity of reservations agents by having the IVR units deal with most of the queries of clients, allowing the agents to answer more queries in less time. This was a high-risk project since the technology used was unfamiliar to the software development firm that was awarded the contract. The first attempt at the project subsequently failed, and four and a half years later, the system was still not operational.

The initial planned time for completion was six months. Approximately five times the initial budget had already been spent during the four years of trying to get the system operational. The current project manager is positive that the IVR system will become operational within the next three months (four years late). If the IVR system becomes operational, value will be added in terms of freeing up time of reservations agents, allowing them to answer more queries in less time, therefore increasing productivity. However, the opportunity of being one of the first companies in South Africa to successfully deploy IVR technology has been lost. The cost of getting the system operational definitely outweighs the value that will be added – it is therefore doubtful whether this project will ever be classified as a success, even after the implementation of an operational system satisfying the user's needs.

Operational systems satisfying the user's needs are often incorrectly perceived as successful if the cost involved in getting the system operational is not considered. This scenario typically occurs when software development is done in-house by the company's own software development section – no money is paid to an outside company, therefore the work done is perceived to be 'free'. During our research we came across quite a number of projects where the added value did not justify the costs and much more money was spent than were originally anticipated, with no additional value added than originally perceived.

2.2.2 Reasons why projects fail

According to surveys done by research groups such as the Standish Group [Standish, 1994] [Yourdon, 1995] on the success rate of SDPs, the following can be deduced:

On average, most SDPs take twice the original planned time, and cost twice as much as what was originally envisaged. Most of the time, only about two thirds of the originally specified features and functions are provided, while about a third of the SDPs that are ever started, are cancelled. Comparing these results with our criteria for successful SDPs given earlier, it is obvious that most SDPs are not successfully completed. Two questions may be asked as a result of the outcome of these surveys:

- Why do so many SDPs fail?
- What can be done to improve the success rate of SDPs?

In an attempt to answer the first question, we provide the following reasons: Inherently, a SDP is a model of the real world that is constantly changing. Building software for a changing world is difficult since no two SDPs are exactly the same – there are always some unknown or uncertain factors to be reckoned with. The title of the well-known article by Brooks [Brooks, 1986] says it all: "*No* Silver Bullet" – thereby saying it is highly unlikely that a silver bullet will be discovered that will solve all software problems.

Different software projects fail in different ways, but it appears that most of them fail because of a combination of the following root causes [Kruchten, 1998]:

- Ad hoc requirements management
- Ambiguous and imprecise communication
- Brittle architectures
- Overwhelming complexity
- Undetected inconsistencies in requirements, designs and implementations
- Insufficient testing
- Subjective project status assessment
- Failure to attack risk
- Uncontrolled change propagation
- Insufficient automation

Lister [Lister, 1997] argues that another reason why most projects are functionally (do not fulfil requirements) and/or calendar late and so few on schedule, is that most project plans account only for the work recognised as absolutely necessary to build the software. These projects maintain no significant contingency fund or time for dealing with all the risks that might or might not manifest as actual problems. Another survey by Rothfeder [Rothfeder, 1988] showed that at least 35 percent of companies have at least one runaway software project. Barry Boehm [Boehm, 1991] argues that most software-project disasters could have been avoided or at least strongly reduced if there had been an explicit early concern with identifying and resolving their high-risk elements and making provision for contingency plans in the original project plan.

A high risk in the IVR project discussed earlier, was the use of new technology that was unfamiliar to the vendor to which the contract was awarded, as well as interfaces into a number of diverse systems of the airline. These risk areas should have been identified early in order to minimise the possibility of project failure. A possible attempt to resolve the problems could have been to obtain an experienced resource with the necessary knowledge of the technology to assist the vendor with the project.

Most SDP failures can be traced back to circumstances not originally planned for. Some of these circumstances might have been identified early in the SDLC and in this way provide for the proactive management of these risks in order to improve the chances of success of the project.

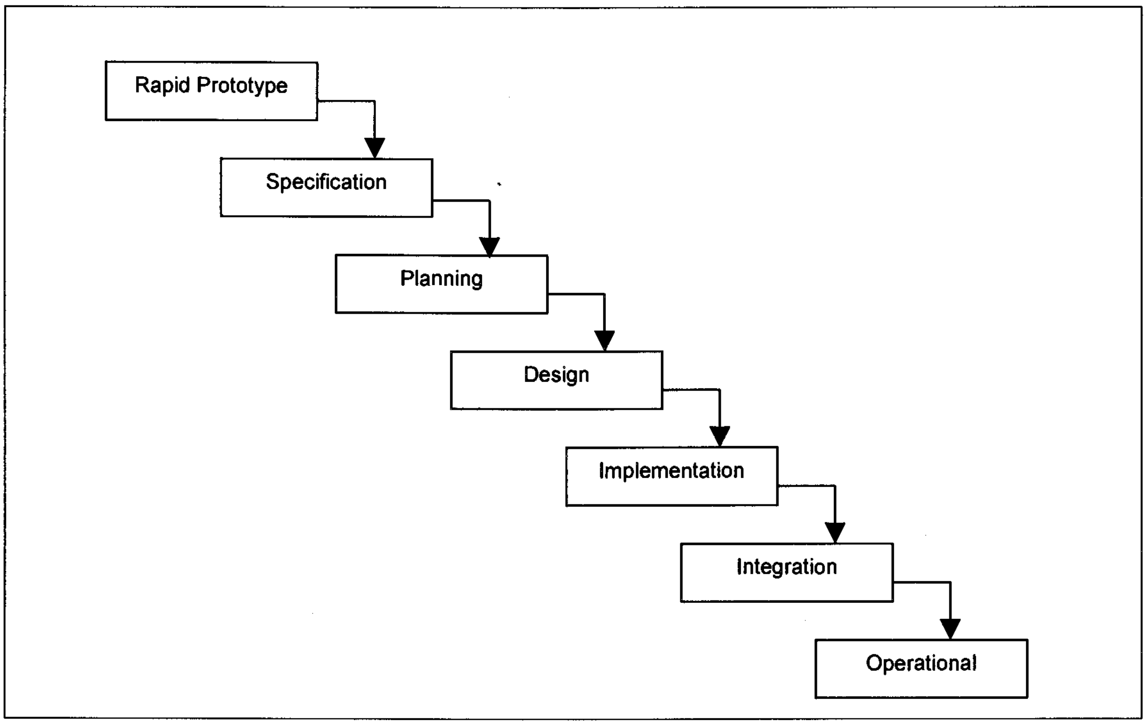
2.3 ATTEMPTS AT IMPROVING THE SUCCESS RATE OF SOFTWARE DEVELOPMENT PROJECTS

Due to the high project failure rate and the consequences of project failure, it becomes obvious that something has to be done to reduce the chances of project failure in one way or another. If we treat the root causes of SDP failures, we will be able to eliminate the symptoms of these causes, as well as being in a much better position to develop and maintain quality software in a repeatable and predictable fashion. That is what the best software practices entail: commercially proven approaches to software development that, when used in combination, strike at the root causes of software development problems [RATL, 1999]. A number of practices have been developed over the years to improve the success rate of SDPs.

2.3.1 Software Engineering

Software Engineering (abbreviated SE) evolved as an attempt to improve the success rate of software projects. Schach [Schach, 1993] defines SE as the study and application of tools and methodologies for developing software. The goals of SE include the controlling and predicting of the cost of software development, producing software that satisfies the specifications and needs of the customer. SE also aims at producing efficient, reliable and maintainable systems with fully documented requirements, specifications, and development plans. SE defines the SDLC. The waterfall method [Schach, 1993] is a way of describing the SDLC, that is, arranging the phases of the SDLC (requirements specification, planning, design, implementation, integration, operation) in terms of their input, function, and output. The waterfall method also explains different methods to achieve the defined output of each phase of the SDLC [Schach, 1993]. Figure 2.1 below provides a simplified overview of the waterfall method:

Figure 2.1: SDLC as defined by the waterfall model



2.3.2 Software Engineering Environment

A Software Engineering Environment (abbreviated SEE) provides automated support for each of the phases in the SDLC. It therefore provides an environment in which large software systems can be developed by integrating a set of tools to support a collection of development methods within an infrastructure that allows tools to communicate and co-operate in a controlled way. The SEE framework provides a set of common services appropriate to software development, which can be used by the tools [Brown, 1992]. An example of an SEE is a set of Unified Modelling Language (abbreviated UML) compliant tools, such as the Rational Suite tool-set, developed by Rational Corporation, which supports the Rational Unified Process (abbreviated RUP). The RUP is an iterative SDLC model. The Rational Suite tool-set provides tools for requirements definition, design and development, testing, documentation and change control [RATL, 1999]. Other tools such as Microsoft Project 98 could be used in conjunction with this tool-set in order to support other functions such as project planning in the SEE. The output of each phase of the SDLC, when using the specific tools, should satisfy the quality standards of the company.

2.3.3 Standards

Another way to improve the success rate of software projects is to enforce certain standards. The International Standards Organisation (abbreviated ISO) provides guidelines to companies for drawing up their own quality standards. ISO 9000 is a three-part cycle including planning, controlling, and documentation. Planning ensures that the objectives, goals, authority, and responsibility relationships of each activity are properly defined and understood. Controlling of the standards defined in the planning cycle must ensure that the goals and objectives are met. The documentation produced in the third task defined by ISO 9000 (documentation) is used for feedback on how well the quality management system is performing to satisfy the customer's needs, and what changes may be necessary [Kerzner, 1995].

2.3.4 Quality Management

Quality management and standards go hand in hand since the quality management process ensures that the standards applicable to a specific process are met. Quality management has been introduced to solve some of the problems of projects, especially those related to user requirements not being met. Project quality management includes the processes needed to ensure that the project will satisfy the needs for which it was launched, as well as satisfying the quality standards of the company [PMBOK, 1998]. Quality management defines the process of enforcing the set standards. Project quality management further includes all activities of the overall management function that determine the quality policy, objectives, and responsibilities, and implement them by means such as quality planning, quality control, quality assurance, and quality improvement within the quality system.

2.3.5 Project management

It becomes clear, by looking at ways to increase the success rate of projects, that the management of all processes throughout the SDLC is imperative for success. It is therefore necessary to closely monitor the whole software development process in order to pick up variances from the project plan. We have to manage these variances to prevent project failure in terms of overrunning deadlines, overspending on budget, or under delivering on user requirements.

Kerzner [Kerzner, 1995] defines project management as the planning, organising, directing, and controlling of company resources for a relatively short-term objective that has been established to complete specific goals and objectives. Proper project management will therefore further increase the chances of success of SDPs.

2.4 CONCLUSION

In this chapter, we have discussed some of the reasons for the low success rate of SDPs. Due to this low success rate, certain practices such as SE, standards, project management and quality management were developed as attempts to improve the low success rate of SDPs.

The inclusion of these so-called "best practices" (discussed in section 2.3) increased the success rate of SDPs. However, the failure rate of SDPs remains unacceptably high, even after implementing these best practices. It is therefore obvious that including these practices is not enough to increase the probability of project success. We therefore need to further explore ways to increase the success rate of projects.

The main shortcoming of the discussed practices is that they only cater for tasks that need to be done during the SDLC of the project, and not for things that may go wrong. Unfortunately, everything does not always go as planned, especially in the software industry. To increase the chances of successfully completing projects, we need to take into account those things that may go wrong. We therefore need to focus on the risks in our projects. Risks in SDPs have to be managed for the projects to be successful, unless we truly believe the optimist's version of Murphy's law: "What can go wrong will go wrong, just not *this* time!" [Lister, 1997].

Having given some motivation for the need for risk management in SDPs, we are ready to examine risk and risk management within the context of software development, in the next chapter.

3 RISK MANAGEMENT

3.1 INTRODUCTION

In the previous chapter, we discussed the success and failure of SDPs. We also gave attention to some of the best practices of software development that evolved as attempts to improve the success rate of SDPs. We saw that implementing these best practices does not guarantee the success of a SDP. We concluded that it is necessary to focus on the risks to our SDPs in order to try to plan for unforeseen events.

This chapter is devoted to providing some background information on risk and risk management. We define risk and risk management in the context of SDPs in section 3.2. Section 3.3 outlines the evolution of risk management as well as the reasons why risk management became part of the SDP. This chapter concludes with a comparison between risk management and traditional project management in section 3.4, highlighting the differences between these management techniques.

3.2 DEFINING RISK AND RISK MANAGEMENT

3.2.1 Risk

A risk can be defined as anything that can cause a project to fail. Risks include any variation to a project, which we may or may not have direct control over, that could either endanger or eliminate the possibility of project success. Possible risks are, for example, a key person in the development team leaving; the budget being reduced by management due to financial difficulties; equipment not arriving on time; or a product similar to the outcome of the current project becoming commercially available at a less expensive cost. Political, communication, schedule, legal, and technical risks can all threaten the success of a SDP [Lister, 1997].

Outsourcing the information technology of companies has become a common phenomenon in today's business industry. When a company's information technology functions are outsourced, there are usually *political* undercurrents involved since people are unsure of their job security and their futures. It is also quite common that more than one IT company contends for the position of the outsourcing partner, leading to an element of competition being present with employees usually favouring one party over another. Such a political undercurrent may seriously impact on the team spirit and morale of the employees, causing deterioration in productivity.

Lack of *communication* within the project team and between the project team and executive management can be a risk - executive management will not be aware of the risks or be able to assist with the risks if the development team does not communicate the risks to them.

An example of *schedule* risk is the Y2K (year 2000 compliance) problem – all initiatives to prevent problems with the century change had to be finished before 1 January 2000, otherwise the opportunity for proactively solving problems would have been lost. If any problems or errors with regard to date representation still existed on the 1st of January 2000, the impact of the risk would have had to be managed. On the 1st of January 2000, it would have been too late to reduce the probability of the risk occurring since it would have occurred already.

Legal risks are encountered where two or more parties have contractual obligations towards each other – for example if a vendor does not deliver the required equipment on time, the client can take legal action. However, even though the client may be protected by the contract, legal action will not solve the problem of preventing project failure – the client may enforce contractual penalties on the vendor, but the project may still be late and therefore unsuccessful.

Examples of *technical* risks are the use of new technology or complicated interfaces to existing systems. If the specific technology has not been used in similar projects in the past, there may be unknown complications that we may not be aware of up front. An example of this occurred in a document image processing project where it turned out that the device drivers of the scanners and the printers were incompatible, causing numerous problems to other peripherals used on the computers. It also took quite a while to identify that this was the problem, since it had not been experienced before.

Harm or loss to a project as a result of risk could be in the form of

- Diminished quality of the product
- Increased costs
- Delayed completion, or
- Total project failure.

To make informed decisions about priorities awarded to risks, we have to be able to distinguish between a high risk and a low risk.

3.2.1.1 Quantifying risks

Prioritising or quantifying risks is very important, since it is seldom possible to attend to all risks simultaneously; the risk with the highest priority should be attended to first. The most common way to quantify a risk takes both the *probability* of the project not being successful because of this risk, as well as the *consequence* of this failure to achieve the cost, performance and schedule objectives into account. Risk exposure (abbreviated RE) provides for a way to distinguish between high risks and low risks. RE is defined as the product of the probability of the potential loss to a software project, and the size or the impact of the loss [Kerzner, 1995]:

$$\text{RE} = \text{Uncertainty} \times \text{Impact}$$

or

$$\text{RE} = \text{Probability (Loss)} \times \text{Size (Loss)}$$

Both the uncertainty (probability of loss) and the damage or impact (size of the loss in monetary units) must, therefore, be considered when aiming to quantify a risk. In general, RE increases with the increase of either the probability or the impact. Those risks with the highest RE should be attended to first, since they have either a high probability of occurring, or a high impact, or both. An example of a risk with a high RE is the year 2000 compliance problem in bank applications – the probability of the century change is 100 % since it will definitely happen. If the dates are incorrect in working out payments on loans, the bank may suffer substantial losses due to incorrect loan contracts being issued to clients. Since both the probability of the risk occurring as well as the potential size of the loss is high, this risk has a high RE and should therefore be dealt with urgently. An example of a low risk may be that a fire may destroy the machine room where the file servers with all data of a certain SDP are stored. However, if it is the policy of the company to make backups of all data and store these tapes at an alternative location, such as in a backup machine room, this risk has a low RE. The probability of a fire occurring is not very high, and the impact if it occurs is low since the backup data will still be available and could be transferred to the servers in the backup machine room. The project manager therefore does not need to give much attention to this risk since the company infrastructure and policies already take care of such risks.

The term "risk" is commonly used in practice assuming it to mean the same as RE. A risk with a high RE is commonly referred to as a high risk.

3.2.2 Risk management

Risks need to be managed to minimise or prevent the occurrences thereof. Risk management, in the context of SDPs, entails the identification of the risks, quantifying and prioritising the risks and then handling the risks by either reducing or preventing the occurrence of the risks. The risks are quantified in terms of their RE. As discussed earlier, the RE of a risk determines its priority and the priority of a risk will indicate the urgency of dealing with the risk. The RE of risks is reduced either by reducing the probability of the occurrence of the risk, the impact of the risk, or both.

The outcome of risk management must be a risk that is either acceptable or unacceptable, in which case either the probability of the risk occurring or the projected impact of the risk should be reduced in some way. An example of an acceptable risk may be a member of the development team leaving. If the coding has been done modularly with adequate design and development documentation, and resources with skills in the specific development environment are available, this risk can be accepted. However, if the design and development documentation is inadequate and skilled resources are scarce, this risk may not be acceptable since the occurrence thereof will probably have an adverse effect on the project. If a risk is unacceptable, we must handle the risk by reducing the probability of the risk occurring as far as possible, until the risk becomes acceptable. In the case of the key resource leaving, we may reduce the probability of the risk occurring by entering into a contract with the resource, offering some incentive for the resource to stay until completion of the project.

Risk management therefore means more than identifying risks – it must quantify the risk and predict the impact on the project, as well as develop, select and manage options for handling these risks. The aim of risk management is, therefore, to prevent risks from occurring, or at least to minimise the impact of adverse risks.

3.3 THE EVOLUTION OF RISK MANAGEMENT

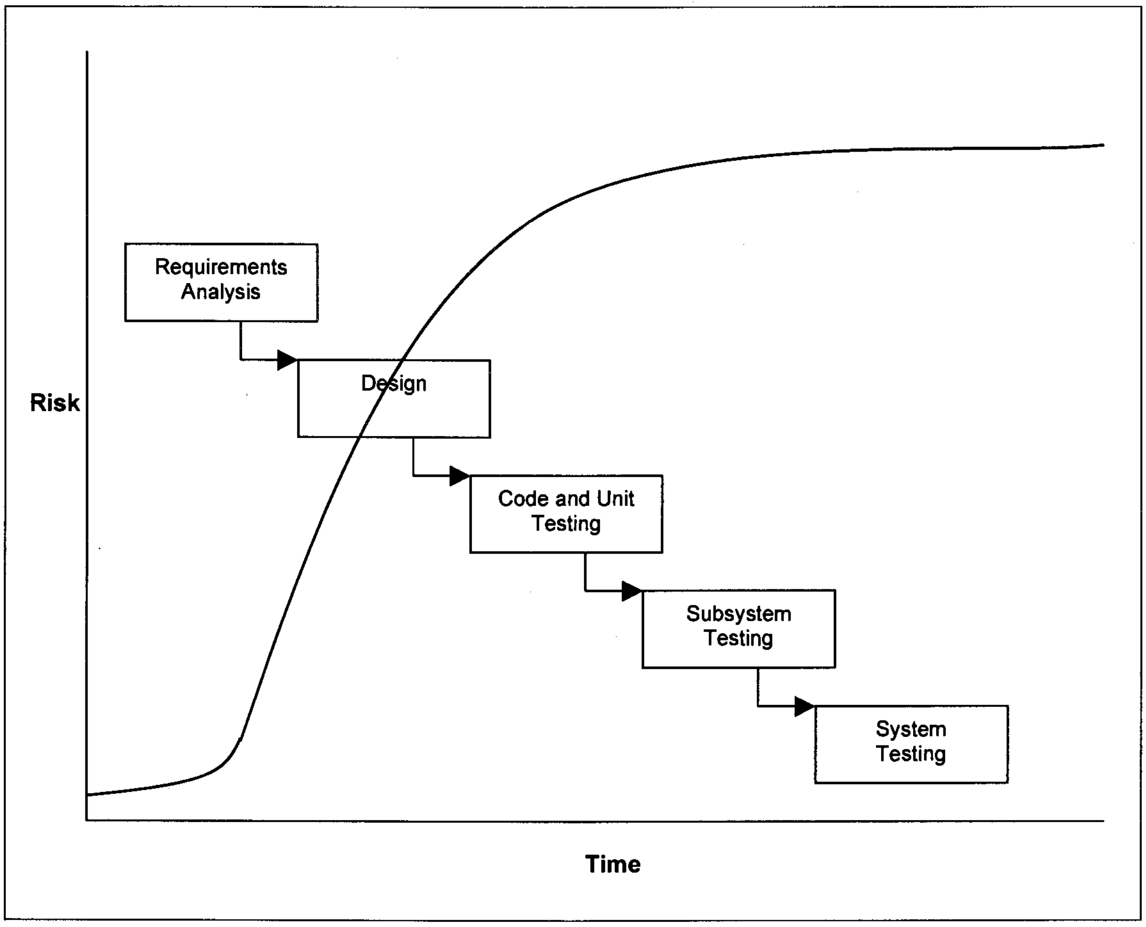
As it became evident that risk management of SDPs was required to increase the chances of project success, certain risk management frameworks were established in the software development industry.

3.3.1 The Spiral Model of the SDLC

Classic software development processes follow a waterfall life cycle, as illustrated previously in figure 2.1 (chapter 2). In this approach, development proceeds linearly from requirements analysis through design, code and unit testing, subsystem testing, and system testing. The fundamental problem of the waterfall model is that it pushes risk forward in time. This causes mistakes in the earlier phases to be costly to undo. The risk with respect to the phases in the waterfall life cycle model is depicted in figure 3.1.

This figure clearly indicates that risk is lower early in the SDLC. It is easier to handle a low risk than a high risk, and therefore it makes sense to handle risks as early as possible in the SDLC.

Figure 3.1: Risk in the waterfall life cycle [Kruchten, 1998]

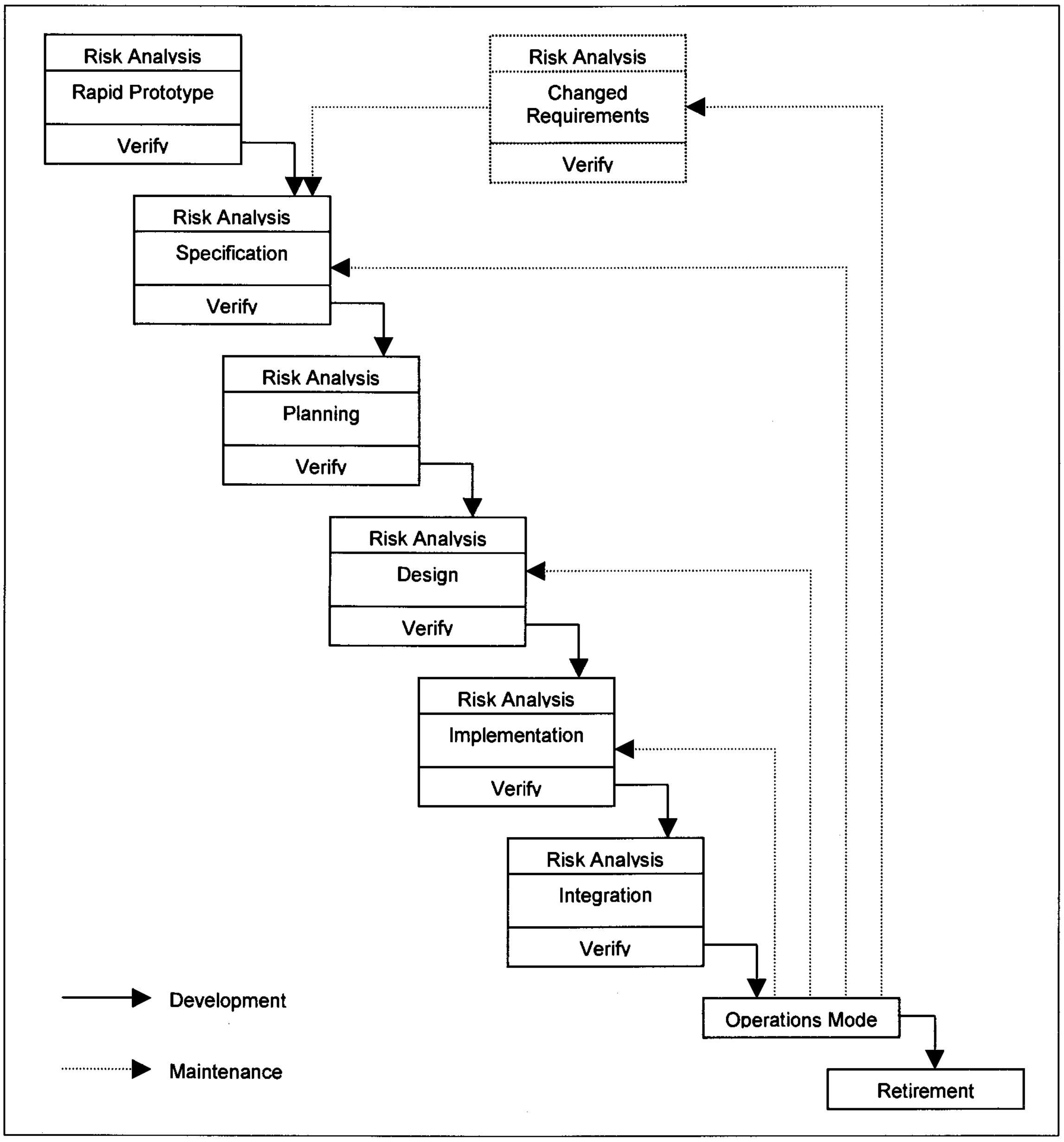


Barry Boehm can be seen as the risk management pioneer for SDPs [Schach, 1993]. He addressed the shortcoming of the waterfall model discussed in the previous paragraph by developing the Spiral Model of the SDLC shown in figure 3.2.

This model was the first model taking into account that there is almost always an element of risk involved in the development of software. These risks must be minimised as soon as possible to ensure project success in terms of functionality, schedule and cost. The idea of minimising risk via the use of prototypes and other means is the concept underlying the Spiral Model.

A simplistic way of looking at this process model, is as a waterfall model with each life cycle phase (prototype, specification, planning, design, implementation, integration, operations, retirement) preceded by a risk management phase entailing the identification of the risks, quantifying the risks and handling unacceptable high risks. Before commencing each phase, an attempt is made to control or resolve the risks. If it proves to be impossible to resolve all the significant (unacceptable) risks at that phase, the project is immediately terminated. The Spiral model has increased the visibility of risk as an issue to be seriously considered in the management of software projects, even though not always to the desired degree.

Figure 3.2: Simplistic version of Spiral Model [Schach, 1993]



The Spiral Model consolidated previously proposed process-model refinements into a single, unifying meta-process model, and made risk management a much more visible and important part of project management [Schach, 1993]. Consider figure 3.2. The Spiral Model uses a basic four phase, cyclic, risk-driven decision process as a meta-project management control mechanism. The first phase consists of the determination of the project objectives and constraints. The risks are then identified in the second phase and alternative courses of actions are evaluated to resolve the risks. In the third phase the selected course of action is implemented and its completion verified. Finally, in the last phase, it is determined whether the risks are now at an acceptable level to proceed to the next decision phase.

These four phases should be looped through continuously as the risks are tracked throughout the SDLC. If the risk is higher than the acceptable level at any stage throughout the SDLC, the risk should be reduced. If the risk cannot be reduced below an acceptable level, the project should be reviewed for its viability. A decision process therefore overlays each individual phase of the meta-process model. Before each life-cycle phase is initiated, management must decide whether the project situation is currently acceptable, feasible, and suitable. The Spiral model uses the level of RE, defined earlier, as a key decision metric for determining whether the project situation is currently acceptable. In the Spiral approach, how well we manage our risks is the overriding factor in developing and managing software [Schach, 1993].

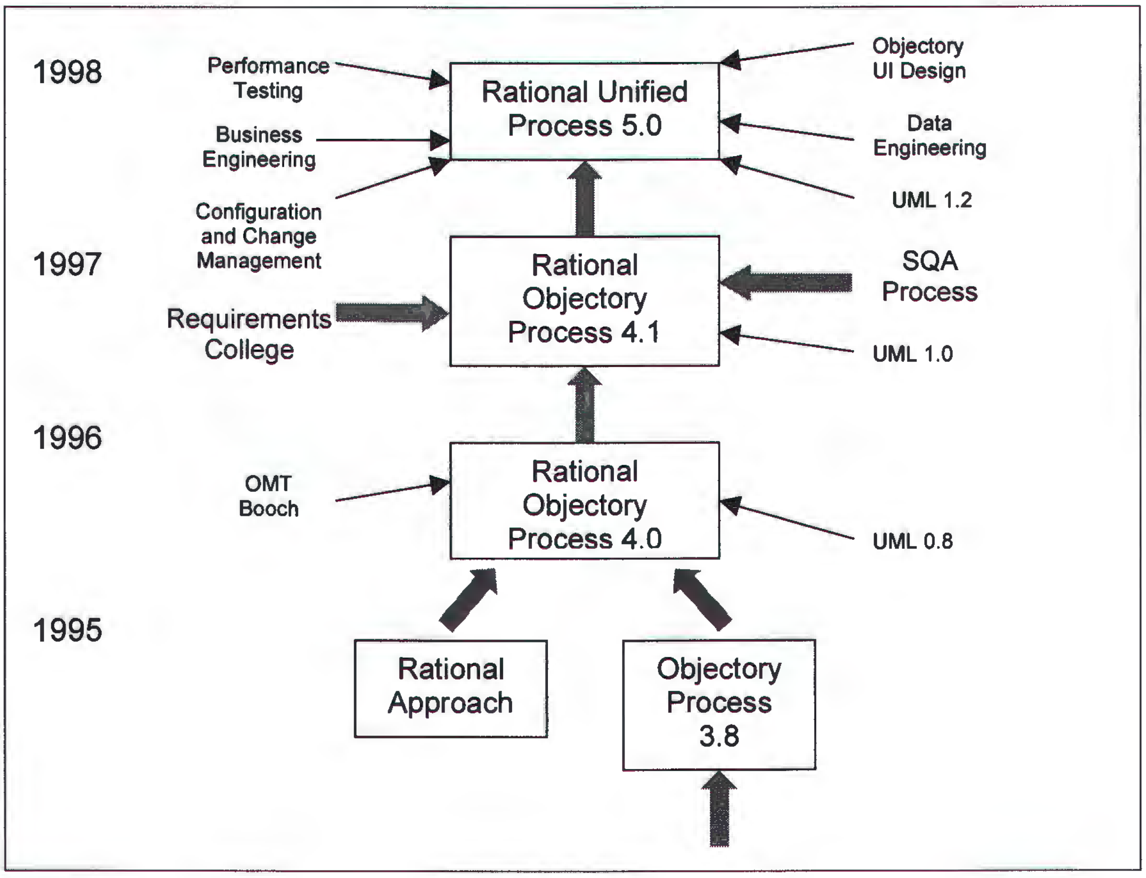
Several efforts to overcome the lack of risk management expertise in the software community have taken place since the Spiral Model first appeared.

3.3.2 The risk management framework of the Rational Unified Process

The Rational Unified Process (abbreviated RUP) has matured over the years, reflecting the collective experience of the many people and companies making up Rational Software's rich heritage. The history of the RUP is illustrated in figure 3.3.

As illustrated in figure 3.3, the RUP is the direct successor of the Rational Objectory Process. The RUP incorporates more material in the areas of data engineering, business modeling, project management, and configuration management than the Rational Objectory Process. From its Objectory background, the RUP has inherited its process model and the central concept of use case. From its Rational ancestry, the RUP gained the current formulation of iterative development and architecture. The RUP also incorporates material on requirements management obtained from Requisite, Inc., and a detailed test process inherited from SQL, Inc., companies that merged with Rational Software. The RUP was the first process to use UML [Kruchten, 1998].

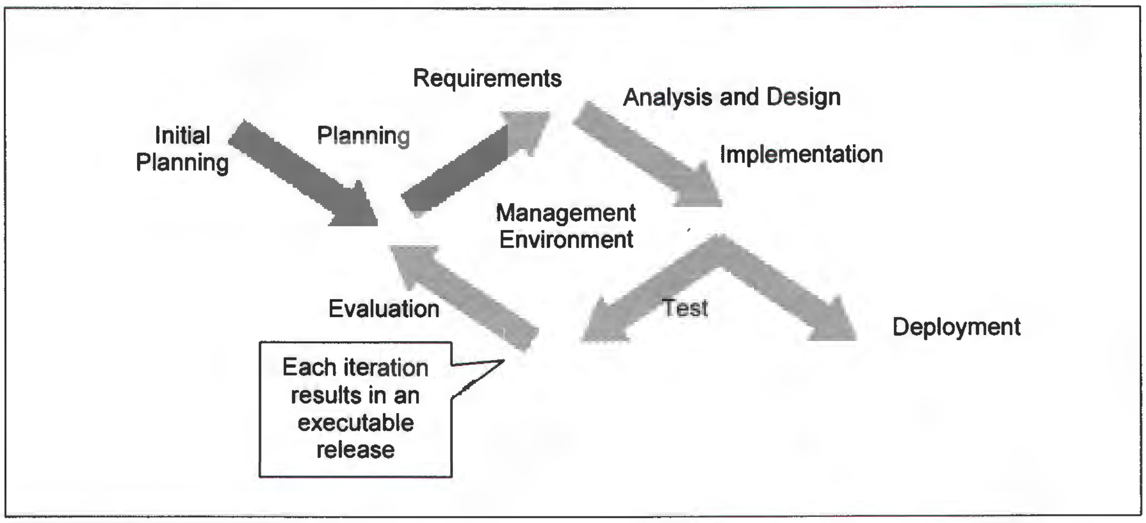
Figure 3.3: Genealogy of the Rational Unified Process [Kruchten, 1998]



The RUP recommends an iterative approach to software development. In this approach, building on the work of Boehm's spiral model, the identification of risks to a project is forced early in the life cycle, when it is possible to attack and react to them in a timely and efficient manner. Following this approach, risk management becomes part of the software development process, as opposed to being a separate activity.

The project team decides up front on the number of iterations the project will consist of, with each iteration resulting in an executable release. If the number of iterations are not defined prior to development commencing, the process may deteriorate into a "build-and-fix" model [Schach, 1993], leading to poor quality software. By defining the different iterations and their deliverables, the project is effectively split into smaller projects. By splitting the project into smaller deliverables, the overall risk of the project is significantly reduced since most risks are usually encountered during integration – in the RUP, integration is not one "big bang" at the end; instead, elements are integrated progressively. This approach is one of continuous discovery, invention and implementation, with each iteration forcing the development team to drive the project's deliverables to closure in a predictable and repeatable way. The RUP is portrayed by figure 3.4.

Figure 3.4: The risk management framework of the RUP [Kruchten, 1998].



Following this iterative process, we loop through the phases of the SDLC for each iteration. We therefore plan the iteration, define the requirements, perform analysis and design, implement and then deploy the executable release as a result of the iteration. After deploying the deliverable of an iteration, we have to evaluate the project to decide whether it is viable to proceed to the next iteration. Should the risk be too high to continue with the next iteration, the project may be cancelled. However, the iterations that have been deployed already will not be cancelled – the products of the delivered iterations can be used to add value.

Although software developers may desperately want to believe the opposite, the truth is that requirements usually change. The iterative approach allows us to take these changing requirements into account, without reorganising and delaying the project. This approach enables and encourages user feedback so as to elicit the system's real requirements, without making the users feel that they have to accept what has been decided on already. Serious misunderstandings are made evident early in the life cycle, when it is possible to react to them.

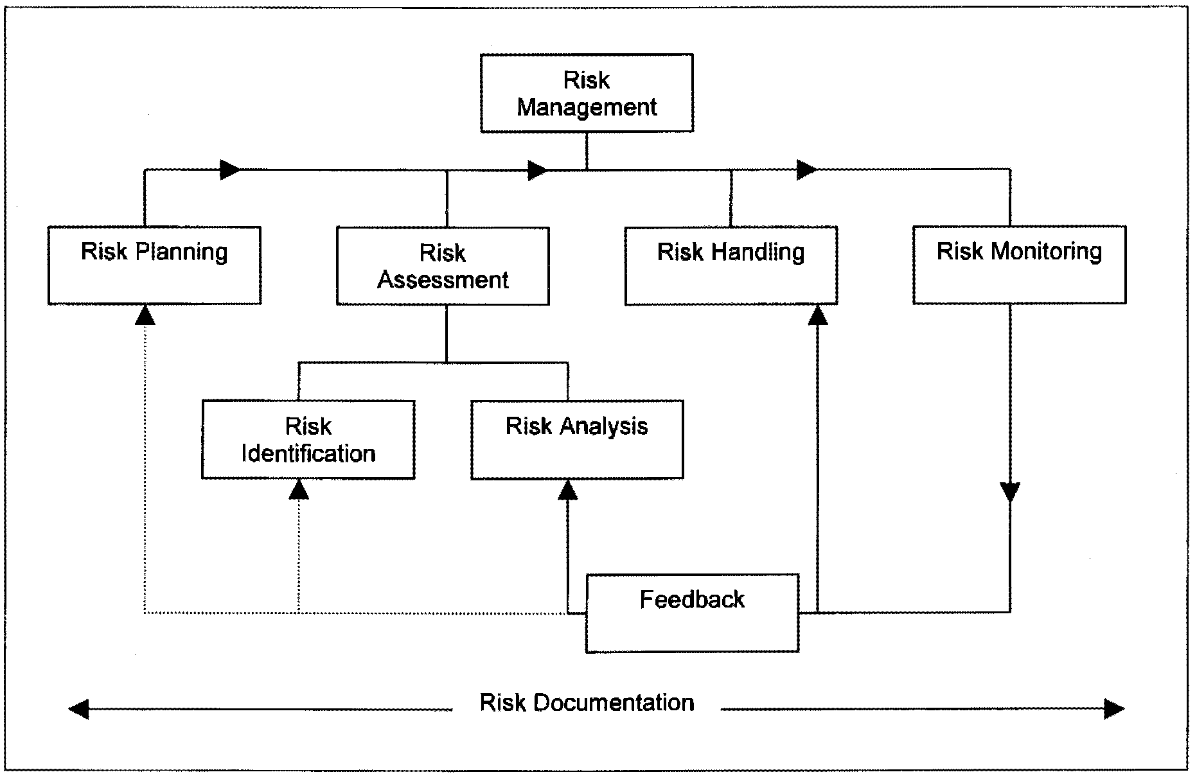
An iterative approach allows abstraction of the project's risks – the development team is forced to focus on those issues that are most critical to the current iteration of the project and are shielded from those issues that distract them from the project's real risks. Inconsistencies among requirements, designs and implementations are detected early.

It is very difficult to assess a SDP's status in the waterfall approach, since concrete evidence is only available closer to the end of the development life cycle. However, in an iterative approach, continuous, iterative testing facilitates an objective assessment of the project's status. Stakeholders in the project can be given concrete evidence of the project's status throughout the life cycle.

3.3.3 The United States DoD risk management framework

Risk management is especially important in mission critical systems where the impact of a risk can have far reaching consequences, such as the loss of human lives. Quite a number of the SDPs of the United States DoD can be categorised as mission critical. The DoD has, therefore, been evolving risk management frameworks to ensure their risk management approaches are complete and consistent. Figure 3.5 is an example of such a risk management framework. This risk management framework extends the framework (Spiral model) presented by Boehm.

Figure 3.5: DoD Risk Management Framework [Conrow, 1997]



Through their research, the DoD derived that good risk management consists of risk planning, risk assessment, risk handling and risk monitoring steps coupled in an integrated, closed-loop fashion and performed iteratively throughout the program's life. Risk planning constitutes the development of the risk plan specifying who is responsible for the risk management, when it must be done and what the outputs of the exercise should be. Risk assessment includes the processes of identifying, quantifying and prioritising the risks. Risk handling is the process of reducing the RE of the risks until it is at an acceptable level. The DoD's framework includes risk monitoring as a definite phase after risk handling; should it become evident that a risk that has been handled before has once again become a potential problem, the DoD framework makes provision for this risk to be looped back into the process of risk handling. It is very important to monitor the risks that have been handled in order to ensure that the RE of these risks does not rise above an acceptable level. If the RE of a risk rises above an acceptable level through the monitoring phase, this risk should be handled again. Should further analysis be necessary after handling and monitoring a specific risk, this risk should be analysed once again in the risk analysis phase.

This DoD framework also makes provision for continuous risk identification, not necessarily just at the beginning of the project. This framework therefore recognises the importance of making risk management part of the SDLC, and not a separate activity.

Each and every phase of this framework should be documented – from figure 3.5 it is obvious that risk documentation forms the backbone of the risk management process. By documenting the processes and arguments that were followed in identifying, analysing and managing the risks to a project, we provide an "audit trail" to be followed should it become necessary to retrace our steps in handling a specific risk. This documentation may also prove to be handy if this risk reoccurs in the project, or if a similar risk is identified in another project.

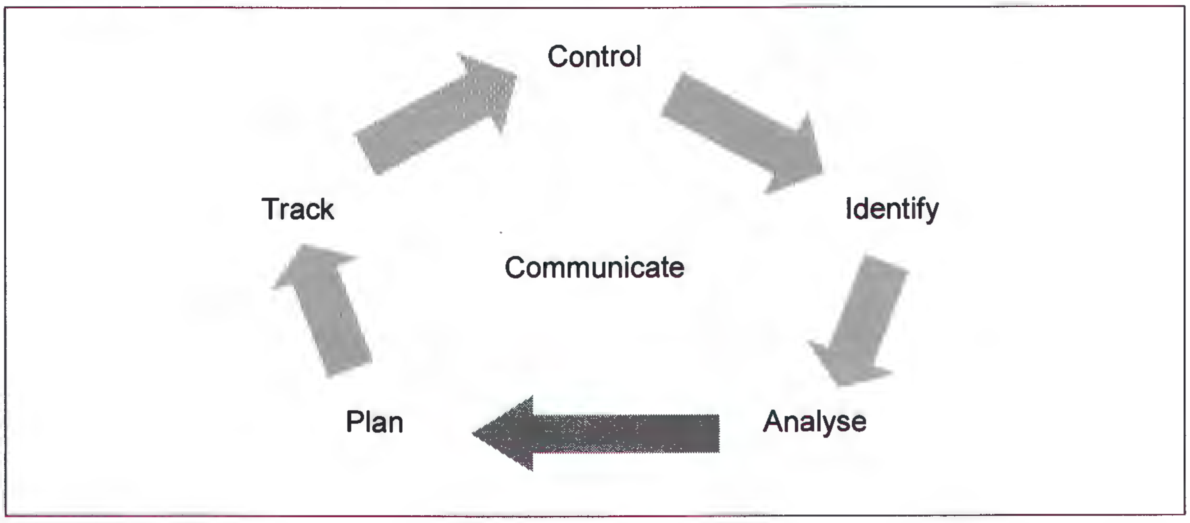
To summarise, the use of this framework enables us to establish the key risk areas early in the development cycle. We then identify the building blocks necessary to prevent these risks from materialising, and implement them immediately. By doing this, we successfully manage our risks through an iterative risk management process with documented feedback to both the project manager and the customer.

3.3.4 The Software Engineering Institute risk management framework

The Software Engineering Institute (abbreviated SEI) established a software risk management program in 1990 mainly to improve the risk management in United States DoD programs involving software-intensive systems. The aim of this program is to identify risk management processes, such as risk identification, risk assessment, risk handling and risk monitoring, as well as methods of implementing these processes that are practical and can be integrated into standard software project management processes. Figure 3.6 shows the SEI Risk Management Framework.

The SEI initially viewed risk management as a four-step cycle (plan-do-check-act), including risk planning, implementing the plan, checking that everything went according to plan, and acting on variances to the plan by adjusting the plan, and then initiating the cycle again. The SEI Risk Management Framework is an elaboration of the "plan-do-check-act" cycle of project management that captured the initial SEI view of risk management. This risk management framework differs from the classic "plan-do-check-act" cycle mainly in its identification step, where risks are recorded for later analysis, and in the communication step in which issues and concerns that could affect the project's success are shared across all working levels [Williams, 1997].

Figure 3.6: SEI Risk Management Framework [Williams, 1997]



The SEI Risk Management Framework, as depicted in fig. 3.6, corresponds to the steps followed in the Spiral model discussed in paragraph 3.3.1. Before moving on to the next phase of a project, risks are identified, analysed and risk handling plans made. The SEI Software Risk Evaluation (abbreviated SRE) is used to highlight specific risks. The SRE generates a list of risk statements (100 or more), evaluates them for their probability and impact, then classifies and ranks them in priority order. The next step of the SEI SRE is to determine risk-handling strategies for the risks. These risk statements are identified through a series of interviews using the SEI Taxonomy Software Development Risk and the Taxonomy-Based Questionnaire. These risks are then tracked and controlled before moving on to the next phase of the project, where the risk management framework cycle is initiated once again. Communication is central to this framework – all risks need to be communicated to the entire project team at all times.

The SEI developed the following risk management methods that follow the SEI Risk Management Framework:

- Continuous Risk Management (abbreviated CRM) – CRM consists of methods and tools project staff can use to ensure that timely risk identification and analysis are performed and surprises avoided.
- Team Risk Management (abbreviated TRM) – TRM is a method the customer and supplier can use to work together in managing project risks. TRM has been proven effective in establishing amicable and open relationships in which each party takes ownership of the risks they can mitigate.

Both the CRM and the TRM establish a risk baseline through the application of the SEI SRE. Both methods follow the SEI risk management framework in the sense that all the phases of the SEI risk management framework (identification, analysis, planning, tracking, controlling, communication) are present in these methods.

The SEI defined levels of maturity in companies with their Capability Maturity Model (abbreviated CMM) [Schach, 1993]. This model defines five different levels in terms of which the maturity of the software process used by an organisation can be measured. According to this model, a company at maturity level one, or the initial level, has the following characteristics: Such a company does not have sound SE management practices in place. Most activities happen as responses to crises. The software process is unpredictable and totally dependent on current staff. By contrast, a company at maturity level two, the repeatable level, has basic software project management practices in place. At this level, the company is also able to gain and learn from past experiences. At level two, measurements are taken to measure effectivity, and plans are made to improve effectiveness of software development.

Both the CRM and TRM are used with best results if the organisation has the infrastructure defined by the SEI Software CMM (discussed in detail in section 5.2.1) level 2 process areas in place [Carr, 1997]. However, these methods may also be used in organisations with maturity level 1 – in such cases, the project manager must act as the risk management champion, using risk management per project and not throughout the organisation.

3.4 RISK MANAGEMENT AS PART OF TRADITIONAL PROJECT MANAGEMENT

We have discussed a number of risk management frameworks in use in industry in this chapter up to now. We need to consider changes necessary to our current management procedures in order to incorporate risk management as part of the SDLC. Project managers need to ask the following questions: What do we need to change about our management of projects to be able to manage the risks? Is there a big difference between the traditional project management we have been practising and risk management?

As stated earlier, project management is the planning, organising, directing and controlling of company resources for a relatively short-term objective that has been established to complete specific goals and objectives. Classical project management is usually considered to have five functions or principles, namely planning, organising, staffing, controlling and directing [Kerzner, 1995].

A problem with traditional project management is that most project plans for SDPs account only for the work recognised as absolutely necessary to build the software. The project plans generally do not make provision for contingency funds, or time for dealing with all the risks that may or may not manifest as actual problems [Lister, 1997]. In traditional project management, risk management is treated as a separate activity with a separate performance cost and reporting section. Risk management is typically viewed as a separate section of the project review, with discussion about risk management typically being only 5 minutes long, and involving only risks with little uncertainty, or problems already solved.

What can we do to rectify the problems and inadequacies of traditional project management? Our view is that a fundamental shift in perception is necessary: risk is integral to nearly every problem being discussed and risk principles can be applied to any such discussion [Gemmer, 1997]. Project management should include managing the risks of the project proactively to resolve these risks before they become problems that may impede on the success of a project. By managing risks, a manager proactively recognises and manages problem areas (risks), while reactive (wait for a problem to occur) management is typical of traditional project management.

SDPs are becoming more and more complicated as technology changes and systems are required to interface with more and more diverse systems. Large-scale software projects are characterised by high decision stakes and high levels of system uncertainty – these projects cannot be managed using traditional project management, since they have objectives that are very complex. Ambiguity, continuous change, and complicated feedback loops generally dominate the decision making in project management of such large-scale projects. Quite a number of project issues are often unique, therefore the experience needed for planning and implementation does not exist [Charette, 1997]. To address these issues complicating the software development process, risk management will have to become the central objective of project management for these large-scale projects to be successful.

3.5 CONCLUSION

As software development became more complex, traditional life cycle models like the waterfall model became obsolete. The waterfall model was the first model to define the phases of the SDLC in terms of their inputs, processes and outputs. However, as with a waterfall, the information flow is generally only in one direction. It is very difficult to move back to a previous phase once the transition between phases has been made. The waterfall model usually works if the project is very well defined, with known requirements and very few surprises. Unfortunately, not many of today's software projects conform to this description. There is a need for models that take into account that things may go wrong or that requirements may change during development.

The Spiral Model was the first risk management initiative. Since then, a number of other risk management frameworks have been developed and are currently in use in industry. Most of these frameworks evolved from the Spiral model. An organisation that wishes to produce successful SDPs has to choose or develop a risk management framework in order to manage risks, since traditional project management does not ensure success for large-scale projects. Risk management needs to be a fundamental part of project management in software development.

In the next chapter, we propose a framework for a risk management methodology. This framework provides guidelines to organisations regarding the steps to be followed throughout the SDLC in order to manage risks properly.

4 FRAMEWORK FOR A RISK MANAGEMENT METHODOLOGY

4.1 INTRODUCTION

Risk management in SDPs needs to address issues particular to the organisation and not issues in general – it can therefore not be a rigid process. Organisations should evaluate risk management methodologies and then choose or derive a strategy that best suits their working culture and circumstances. Some customisation will be necessary to make provision for risks that are particular to the organisation.

We should not opt for any risk management methodology just to be able to say that we practice risk management, as a weak risk management methodology can introduce considerable doubt as to the accuracy and value of the results. An example of this may be an organisation that is using an outdated method, or identifying risks to their projects in the form of a questionnaire. If such a questionnaire does not cater for all the specific risk areas in the organisation, such as for example high Information Technology staff turnover, the project team may be experiencing a false sense of security since they may not be aware of all the risks of the project.

It is important that a risk management strategy be established early in a project in order to perform meaningful risk management. Risks must be addressed continually throughout the project life cycle and not just in the beginning phases, as risk rarely remains constant throughout the SDLC.

In this chapter, we discuss the different steps that should form part of a risk management methodology as derived from the risk management frameworks discussed in the previous chapter, as well as from alternate sources such as the National Infrastructure Program (abbreviated NIP) used by Transnet [Transnet, 1996]. This framework can be used as a guideline for companies to devise their own risk management strategy. The steps outlined in the proposed framework for a risk management methodology should be present in the risk management strategy of any company. However, the methods used for each step may differ, since different companies may use different methods for the same step of the risk management methodology. For example, one company may make use of a risk questionnaire in identifying risks to a project, while another company may make use of informal interviews.

In addition to discussing the steps that should form part of a risk management methodology, we also discuss and compare certain methods that could be used to implement the specific steps.

The high level steps of the risk management methodology will be the same across organisations – however, the way in which an organisation implements a specific step as well as the tools used may differ from that of another organisation. We mention certain tools that can be used in the specific phases – these tools may either be software tools or manual procedures.

The various steps that should form part of a typical risk management methodology can be summarised as follows:

- Risk assessment – the process of determining which risks are likely to affect the project
- Risk analysis – the quantification and evaluation of the probability of the risk's occurrence and the risk impact
- Risk handling – the process of defining mitigation steps for threats and enhancement steps for opportunities
- Experience gained – learning from past experience [PMBOK, 1998].

4.2 RISK ASSESSMENT

The first step towards risk management involves the assessment (or identification) of risks. This is the process of examining a situation and assessing and classifying the areas of potential risk, making them visible to all involved in the project. Effective risk assessment needs the participation of all parties involved in a brainstorming, no-holds-barred atmosphere [Lister, 1997]. The risk assessment process may include a survey of the project, customer, and users for concerns and problems. The thoroughness with which this assessment is done will determine the effectiveness of the risk management exercise. Not all risks are high-level risks that will crucially impact the project – however, the cumulative combination of a number of low-level risks could have a severe impact on the project [Kerzner, 1995].

Organisations that initially start doing risk management often fall into the trap of not having a structured, systematic method of assessing and recording risks. In the absence of a systematic, structured risk assessment method, workgroups tend to use vaguely worded statements that are easy to dismiss as impossible to prevent. The risk assessment exercise is usually only done during the initial phases of the project, where it should be continually looped through throughout the SDLC of the SDP.

To depend on managers to recognise and articulate all possible problems or risks may lead to identifying only a subset of the possible risks since the managers may only foresee what their experiences have conditioned them to look for. It is important to identify all possible parties that may assist in assessing risks and to involve them in this exercise. The whole project team should work together when assessing risks. Williams [Williams, 1997] found a situation where separate mitigation teams on a project developed their own contingency plans, workarounds, and special tools to reduce the project's risk with respect to a support tool. This proved to have been wasted resources since the teams never sought or found a common solution to the risks.

Open communications within a project team are extremely important if we want to successfully assess all (or at least most of) the risks relevant to our project. Project managers consistently found issues that they were not aware of after undergoing a SEI SRE, where prior to the risk evaluation, they thought they had open communications within their project teams [Carr, 1997]. We cannot assume that we have open communications within our project team without providing the infrastructure or framework for discussions relevant to our projects. An example of enabling open communications may be a weekly progress meeting at which each member of the project team gets chance to raise his / her concerns with regard to the project, as well as to make suggestions for improving the chances of success of the project.

In chapter 3, we defined the concept risk with regard to software development as an ongoing or impending concern that has a significant probability of adversely affecting the success of major milestones in a SDP. We have to be aware of the different types of risks that we may encounter if we want to successfully assess most of the risks that may impede on our projects' success.

In working with organisations, Carr [Carr, 1997] found that for the most part risk identification and analysis is performed on an ad hoc basis, generally at the beginning of a project through brainstorming sessions by senior engineers. The risks identified in these ad hoc sessions are generally the global risks that the organisation is willing to accept. There is a lack of systematic methods to ensure that all aspects of the projects have been examined for risk and that the project is periodically re-examined to identify new risks.

The SEI often works with groups, assisting them in identifying the risks prevalent to their SDPs. The SEI guides these groups to use risk statement constructs as a model for consistency. Each risk statement consists of a condition and at least one consequence of the condition. The condition is something perceived to be true today and is a problem, something neutral or a good thing, from which undesirable outcomes are expected. This model consisting of the risk statements then becomes a benchmark for all new risks [Williams, 1997]. This method provides a structured way of identifying and documenting the risks in SDPs.

To avoid bias when assessing risk, project teams must be coached to identify all possible sources of uncertainty. These sources include political, social, financial, environmental and technical areas, as well as the relationships between these areas. Without understanding the uncertainty underlying risks and the relationships between them, the project team is powerless to affect the odds and therefore cannot practice proactive risk management.

Classification of the identified risks around a common structure for all projects can provide an added bonus to the organisation. It allows categorisation, analysis, and retrieval of risk information from across projects, allowing us to identify common risks, successful and unsuccessful risk handling strategies, as well as organisation-wide trends. Risks are typically more complex than individual risk statements, and often encompass, or overlap, with other risks. More than one risk statement can emerge from an individual risk statement, pointing to different aspects of the same risk. Global risks that can be solved together can be found by classifying or grouping risk statements into categories based on shared characteristics [Williams, 1997].

To summarise, the following points are important to implement in order to successfully assess risks to a SDP:

- Identify all the parties that could provide risk information for the project and involve them in the risk assessment exercise.
- Derive a systematic and structured method such as following a risk list (questionnaire) to assess risks.
- Use a consistent method of documentation for all risks.
- Repeat the process periodically throughout the SDLC.

4.2.1 Risk categories

By defining certain risk categories, we can enhance the process of risk assessment. The categorisation allows us to focus on each of the individual categories, ensuring that we consider all aspects that can pose a risk to our SDP.

We distinguish between two main categories of risk in the business context, namely business risks and insurable risks. Business risks provide organisations with opportunities of profit and loss, while insurable risks only provide an organisation with a chance of a loss. Business risks and insurable risks can be further categorised as follows: external-unpredictable, external-predictable, internal (non-technical), technical and legal risks [Kerzner, 1995].

4.2.1.1 Business risks

A good example of a business risk is the risk that airlines take when overbooking flights. Overbooking means selling more tickets for a specific flight than are physically available. If statistics show that on average, only 88 percent of the people that book for a flight actually arrive for the departure of the flight, the airline may decide to sell more seats anticipating that 12 percent of the people that booked will not claim their seats. Assume an airline sells 110 seats for a flight that only has 100 seats in total. If 12 percent of the passengers does not arrive in time for the departure of the flight, this means that 97 people will arrive for the specific flight, therefore only three of the seats will not earn revenue for the airline. However, if the airline only sold the 100 seats that were actually available, 12 seats would have been empty and therefore provided no revenue. It may also happen that all 110 passengers that booked seats for a flight arrive for the departure of the flight. It is the policy of most airlines to allow these passengers to fly on the next available flight free of charge (reimbursing them for the flight they were booked on) and to accept responsibility for any additional accommodation or transportation costs that the passengers may have incurred. It is therefore only viable for the airline to take the risk of overbooking flights if, on average, the airline earns more revenue than the costs incurred due to the overbooking of flights.

4.2.1.1.1 External – unpredictable risks

Risks in this category include all risks that can be caused by external influences and situations that are impossible to foresee by the project development team.

An external-unpredictable risk can be the rand dropping in value with regard to the dollar. A South African airline budgets a certain amount for fuel costs in one year. However, the fuel has to be paid for in dollars, therefore the deterioration of the rand with respect to the dollar may cause great losses to the airline.

4.2.1.1.2 External-predictable risks

An external-predictable risk can be illustrated by the fact that the South African government changed regulations with regard to domestic passengers being carried on flight legs of international flights. A South African airline provides a flight from Miami, via Cape Town to Johannesburg. In the past, the Cape Town – Johannesburg flight could have been used as a domestic flight, allowing passengers to be taken aboard in Cape Town. However, due to the fact that the domestic passengers do not need to go through customs in Johannesburg, while the passengers en route from Miami do, the government changed the regulation with regard to allowing domestic passengers on a flight leg of an international flight. It was too easy for the international passengers to pass on goods to the domestic passengers during the flight from Cape Town to Johannesburg.

The airline loses a large amount of revenue per year by having to let the aircraft from an international flight fly with a load factor far below capacity between Cape Town and Johannesburg. We can argue that the airline should have foreseen the government changing this law, since large amounts of money were lost in terms of customs taxes, and the possibility of smuggling was also enhanced.

External risks (both unpredictable and predictable) are outside of the project manager's control but may impact on the direction of the project.

4.2.1.1.3 Internal (non-technical) risks

Internal risks may be controlled by the project manager and present uncertainty that may have an impact on the project.

Providing proper documentation on risk information can be classified as an internal, non-technical risk. Part of implementing a successful risk management methodology is to learn from past mistakes. If we want to prevent ourselves from making the same mistakes over and over, we need to document the experience we have gained, making this information available to whoever may benefit from it within our SDP teams. Not documenting the lessons learnt will cause the same risk to materialise time and again, without us making use of past experience to prevent this risk from occurring. The reason why we classify this risk as a business risk is that we may actually benefit from applying the lessons from the past to provide a better product within a shorter development time.

4.2.1.1.4 Technical risks

Technical risks have to do with the technology being used and the impact it has on the direction of the project.

Do we use any new technology that has not previously been successfully implemented by the software development team? Using new unfamiliar technology always poses a risk to the success of a SDP. We need to identify where new technology is being used in order to be aware of problems that may occur. The IVR project discussed in section 2.2.1 (chapter 2) provides a good example of technical risks.

4.2.1.1.5 Legal risks

Legal risks are typically always present where companies depend on one another and this dependency is described in a contract. If the client and the developer are from different companies, a contract will be drawn up between the two parties to specify the rules pertaining to their involvement. If the contract states that the developer receives some sort of incentive if he delivers on time or ahead of schedule, the developer stands the chance of gaining from this legal risk. However, the contract will in all probability also state that the developer will incur penalties should the product not be delivered on time.

4.2.1.2 Insurable risks

The second main category of risk, namely insurable risks, is those risks that can only lead to a loss to the company, with no possibility of a profit.

An example of an insurable risk for an airline may be the risk that circumstances may cause a flight to be delayed. Such a delay can cost the airline authorities a great deal such as, amongst others, charges for utilising an aircraft bay for longer than the bay allocation and reimbursing clients for connecting flights missed.

As is the case with business risks, insurable risks can also further be categorised as external-unpredictable, external-predictable, internal, technical and legal.

4.2.1.2.1 External – unpredictable risks

Due to the unpredictable nature of these external risks, it is impossible to compile a list containing most of the risks that will be encountered in this category. Possible risks in this category may be a change in requirements due to a misunderstanding between the user and the analyst who initially compiled the user requirements.

4.2.1.2.2 External-predictable risks

To depending on the expert judgement of a specific person, may pose an external-predictable risk to a SDP. What if this person is not the expert we believe him to be?

Would it not be safer to consult different, independent experts? We have to identify all dependencies on expert judgement in our SDPs and make sure that we can trust the judgement of the so-called expert(s).

Making certain assumptions may also pose a risk to a SDP. We need to identify all assumptions, i.e. those things we believe to be true but that we do not have proof of. We need to document our assumption analysis – should our assumption prove to be false, we need to know how we derived this assumption to identify where we went wrong.

4.2.1.2.3 Internal (non-technical) risks

We found that, especially in South African state-owned companies, good programmers tend to be promoted to become SDP managers. However, the skills necessary to be a good manager differ considerably from those needed to be an outstanding programmer. Managers therefore need to be equipped with the skills needed for good management. Good management methods need to be implemented explicitly – we cannot assume that all managers are aware of good management practices. We therefore need to identify areas where it is not clear as to which management method to implement in order to ensure the success of our SDP. Ineffective project management (multiple levels possible) will probably have an adverse effect on the outcome of a SDP.

A work environment that is not conducive to productive development may have an adverse effect on the success of a SDP. An open plan office may be beneficial for the team working together, but if basic rules in terms of noise are not adhered to, productivity may suffer.

4.2.1.2.4 Technical risks

The user requirements form the foundation upon which our whole SDP is built. We need to be quite certain that we have established the correct requirements and that these requirements will remain reasonably stable throughout the SDLC. Requirements that change during development may cause a risk to our project.

Questions we need to ask ourselves include the following. Is it possible to validate that we have elicited the correct requirements? Can we build a prototype of the system to be validated by the involved user groups? We need to identify all the problems we may experience when developing realistic scenarios and test data to demonstrate conformance to the requirements. We also need to identify whether we may experience problems when testing the product.

The requirements specifications should not contain any technical implementation detail [Schach, 1993]. Implementation decisions should only be made at the detailed design stage; prior to that, implementation details may cloud other, more important issues.

4.2.1.2.5 Legal risks

Whenever the success of a SDP is in any way dependent on a deliverable by an alternate party, legal risks will be encountered. All interactions between the parties involved should be described explicitly in a contract, binding both parties to fulfil certain predetermined and agreed-upon requirements. An incomplete contract poses legal risks to all parties involved. From the viewpoint of software development, legal risks are usually perceived to be insurable risks, i.e. risks that can only lead to a loss. Even if the contract states that another party must pay penalties to the software developers in case of non-compliance to the contract, the success of the project will still be jeopardised.

Questions we need to ask ourselves include the following. Does the contract specifically state ownership in terms of the components of the SDP? Will the developer be allowed to use the requirements and software for work to be performed for other companies? Will the developer be allowed to perform similar work for other companies in the same industry, possibly business rivals of the company for which the current development is done?

The requirements document of a SDP is in effect a contract between the user and the developer. Both parties have to agree that this document contains all information pertaining to what the user expects as well as what the developer has to implement. If the requirements have not been devised or documented adequately, this may lead to a software product that does not really satisfy the user's needs. Inadequate documentation of requirements may lead to legal repercussions if requirements have not been explicitly stated. The requirements document also forms the basis for the rest of the SDLC – poor requirements may cause immense problems throughout the SDLC, in all probability causing the project to fail.

4.2.2 Software development areas to be examined when identifying risks

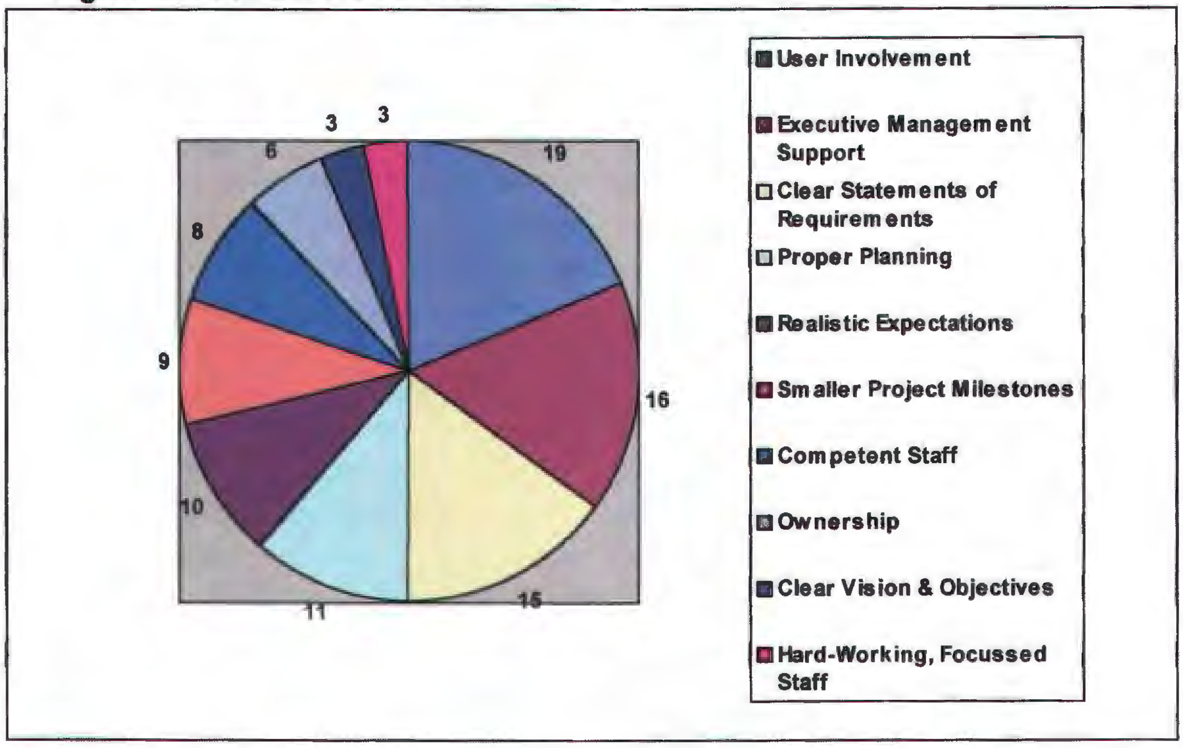
It should be clear by now that some degree of risk will always be present in any worthwhile project. These risks (as categorised in section 4.2.1) may originate in any project area.

The Standish Group [Standish, 1999] identified ten project success factors (project areas) through researching the opinions of IT executive managers on why SDPs fail. The ten project success factors are weighted in terms of how important they are for the success of a SDP. The three major factors that were identified are user involvement, executive management support, and a clear statement of requirements. Although there are other success criteria, with these three basic elements in place, the chances of success are much greater. Without these three criteria, the chance of failure increases dramatically. The ten success factors can be summarised as follows, stating the most important factors first:

- User involvement
- Executive management support
- Clear statements of requirements
- Proper planning
- Realistic expectations
- Smaller project milestones
- Competent staff
- Ownership
- Clear vision and objectives
- Hard-working, focussed staff

The importance of these success factors to a SDP, is illustrated in figure 4.1.

Figure 4.1: Ten success factors for SDPs



From the pie chart in figure 4.1, we can see the importance of the ten success factors relative to one another. We should pay special attention to identifying risks in the areas that are most important to project success, since these risks may have a significant impact on the success of our SDPs. Any risks impeding on user involvement, executive management support or the clear statement of requirements should therefore receive special attention, since these three areas determine about 50% of the success of a SDP.

4.2.3 Possible risk assessment methods

If we are not aware of the risks to our project, we cannot prevent these risks from causing project failure. The results of the risk assessment step, namely the list of risks in our project, form the basis for the rest of the risk management effort. The problem experienced when assessing risks is that there does not exist a single method that we can follow that will exhaustively identify all the risks in our SDP.

There are numerous methods for assessing or identifying risks – any source of information that recognises potential problems can be used for risk assessment. In this dissertation, we focus on a specific risk assessment method, namely the use of a risk questionnaire or checklist. There are a number of risk questionnaires available commercially that can be used to assess risk. These checklists provide a structured way to identify the main risk areas for a specific project.

Risk questionnaires guide us in assessing risks by focussing our thoughts on certain issues that usually pose risks in any SDP. The questions making up the questionnaire actually prompt the project team to think about and identify possible risks in their SDP. However, such a questionnaire should never be implicitly trusted as to identify all risks in a software project. Each company and each project has its own particular set of circumstances that must be considered when assessing risks. When choosing a risk assessment questionnaire, we have to evaluate the questionnaire in order to determine whether it covers most of the risks we usually encounter in our SDPs. We may need to add questions to help us identify those risks that we have encountered in the past, but that are not covered by the questionnaire. We can evaluate a risk assessment questionnaire by comparing the questionnaire with the risks generally encountered in SDPs in practice.

In this section, we combine our own experience and research into risk assessment practices in use in industry, with the work of Barki, Rivard and Talbot [Barki, 1993], Conrow and Shishido [Conrow, 1997], and Moynihan [Moynihan, 1997].

Barki, Rivard and Talbot reviewed in-house as well as custom development for external clients, as well as Boehm's work on software risk items. They then identified 35 variables, which they used as the basis for creating scales in a project risk assessment instrument. Conrow and Shishido identified 150 candidate risk issues through examining many studies that have been done to identify risk contributors in software-intensive projects.

Tony Moynihan did a survey to determine the risks most project managers are concerned about. He surveyed 14 experienced application system developers. His survey focussed on three main areas, namely

- Which characteristics of the customer or the application, do experienced software project managers consider important when planning development projects for new clients?
- How do these characteristics relate to accepted software project risk drivers?

- Do most project managers characterise new projects in generally the same way, or do different project managers use different perceptual lenses when viewing new projects?

Through his survey, Moynihan elicited 22 main constructs. These constructs should be represented by certain questions in the questionnaire being used to assess a project's risk.

4.2.3.1 Evaluating risk assessment questionnaires

"The notion of building a single, all-encompassing risk taxonomy for use by all software developers is probably unrealistic. We may need different risk taxonomies for different project contexts" [Moynihan, 1997].

Risk questionnaires or checklists cannot be regarded as definitive statements of precise risks, but should rather be used as an indicator of the areas that are more likely than not to cause problems. An organisation may (and should) derive its own checklist from a number of other lists – organisations are often prone to specific risks due to factors such as environment, cultural aspects, and others. An example of this is that a South African organisation may be more prone to labour strike action than a Swiss organisation, therefore higher risk will be associated to a South African company in terms of possible human resource problems.

It is very important for a project manager to ensure that a comprehensive set of perspectives is taken when identifying risks in new SDPs – otherwise some of the risk areas may be ignored. The project manager therefore needs to make sure that the questionnaire being used to identify the risks satisfies the requirements for the organisation, as well as the specific project by including all (or most) of the risks that are usually encountered in such a project and organisation. Evaluating the questionnaire according to a specific evaluation framework can ensure that the questionnaire satisfies the requirements.

Tony Moynihan [Moynihan, 1997] derived such an evaluation framework for risk questionnaires through his survey of risks in SDPs. He elicited twenty-two main constructs that should be represented by questions in each risk identification questionnaire. Moynihan evaluated two widely used questionnaires against these constructs, namely Barki's risk variables and the SEI SRE. Barki [Barki, 1993] identified 35 variables to be used as the basis for creating scales in a project risk assessment instrument. The SEI Taxonomy-Based Risk Identification instrument consists of 194 questions [Carr, 1993].

We evaluated another risk assessment questionnaire against Moynihan's constructs, namely the questionnaire used by Transnet for risk identification [Transnet, 1996]. The Harvard Business School under the guidance of Prof. Warren McFarlan developed this risk assessment tool. This questionnaire focuses on three risk areas, namely size, structure and technology. This questionnaire has been included in the NIP used by Transnet as part of its management processes.

The results of Moynihan [Moynihan, 1997] and our own evaluations are shown in table 4.1 below.

Table 4.1: Comparison of Barki Variables, SEI Taxonomy Questions and NIP Risk Assessment Questionnaire against Moynihan's Construct Themes.

Construct Theme	Barki Risk Variable	SEI Risk Question	NIP Risk Question
The client's knowledge / understanding / clarity regarding the requirements / problem to be solved.		Are requirements changing as the product clarity is being produced? Are requirements missing or incompletely specified?	Replacement of automated /manual system, or new. Percentage functionality to be replaced. How knowledgeable is the user in data processing? How knowledgeable is the user in the application?
The existence / competence / seniority / commitment of the project patron / owner.	Lack of top management support.		Involvement of senior management.
IT competence and experience of customer / users.	Lack of IT experience.	Does the customer understand software? Does the customer understand the technical aspects of the system?	Programming language to be used? New system software or operating system? Is system software new to the vendor? How knowledgeable is the project team in the application?
Need to integrate / interface with other systems.	Number of links to existing systems; Number of links to future systems; Extent of linkage of system to other organisations.	Are the external interfaces completely defined?	Number of subsystems. Dependence on sub-systems? Amount of networking involved?
Scale / co-ordination complexity of the project (number of disciplines, need to share resources, need to sub-contract)	Number of hardware suppliers, software suppliers, people on team; Relative project sizes; Team diversity.	Do requirements specify a larger/more complex product or require a larger team than the developer is used to?	Total systems and programming man-months at full availability; Total estimated calendar time

Construct Theme	Barki Risk Variable	SEI Risk Question	NIP Risk Question
Main source of control over the project (developer versus client versus third parties)		Does the program have any dependencies on outside products / services?	Who will perform the work? The number of external vendors involved.
Level of enthusiasm / support / "energy" for the project in the client's organisation	Lack of user support		General user attitude
Level of change to be experienced by the client (to procedures, workflow, structures)	Extent of changes (to user tasks, organisation structure); Degree of computerisation of the present system.		Severity of procedural changes; Procedures / methods used for first time or major breakthrough in implementation of new procedures? Degree to which the user must change?
The need to satisfy multiple groups of disparate users versus the need to satisfy one group of similar users	Number of users outside the organisation; Number of users inside the organisation; Number of departments involved; Number of hierarchical levels occupied by users		Number of non-data processing departments or locations; Approximate number of user department personnel; Number of geographic locations
Who we will be working through: users versus the IT department, individuals versus committees.		Is there a poor interface with customer, other contractors, senior / peer managers? Are all customer factions involved in reaching agreements?	Do you have a joint data processing / user team?
Developers' familiarity with platform / environment / methods	Lack of development expertise in the team (regarding platform, methods, tools)	Is there prior company or project member experience with the development system?	Which programming languages are used?
Developers previous experience with the application.	Team's lack of experience with the application; Team's lack of experience with the task	Do the requirements specify something generally never done before or never done by your company before? Is the staff lacking domain knowledge?	What system software / operating system is new to the user? Is the system software new to the vendor?
Logical complexity of the application	Task complexity		Number of subsystems? How many systems does it interface to? What is the level of the system complexity?
Ease of solution validation (such as possibility of prototyping)		Is there any problem with developing realistic scenarios and test data to demonstrate conformance with requirements? Is the product difficult or impossible to test?	To what degree are software packages being used?
Client's willingness / capability to handle implementation			
Freedom of choice of platform / development environment			
Criticality / reversibility of the roll-out of the new system			Is the system batch or online? Does it use a distributed or on-line database? Does it use

Construct Theme	Barki Risk Variable	SEI Risk Question	NIP Risk Question
			distributed hardware?
Developer's knowledge of country / culture / language			
Stability of the client's business environment			
Maturity of the technology to be used	Need for new hardware; need for new software	Any state-of-the-art requirements for technologies, languages, hardware, and so on?	The extent to which new or unfamiliar user-related hardware is needed; what additional hardware is needed? What hardware is the first of its kind for the vendor or vendor's local support group?
Developer's knowledge of client's business sector.			How knowledgeable is the project team in the application?

From the above evaluation of the three risk assessment questionnaires, it is evident that not one of the evaluated questionnaires addresses all of Moynihan's constructs. Also, Moynihan's constructs do not necessarily cover all the risks that are identified by the questionnaires. This proves that we cannot totally depend on any single risk identification questionnaire to enable us to identify all the risks in a SDP.

Risk management describes what is *different* about our project from all others. What is our unique set of circumstances and issues for this particular effort? We need to evaluate the risk identification questionnaire that we chose in the context of our particular circumstances, i.e. taking into account our organisation as well as the specific SDP. By doing this, we need to identify the shortcomings of our chosen questionnaire and expand on it to include those issues that we have identified as not being covered by the questionnaire.

Now let us assume we have found and expanded on the perfect risk assessment questionnaire for our specific software project. We used this questionnaire to guide us in identifying all the pertinent risks that may impede on the success of our SDP. What do we do now that we know what our risks are? How do we prioritise these risks? How will we handle our risks and how will they affect our plans?

4.3 RISK ANALYSIS

We can assume that, after having applied step one of the risk management methodology (risk assessment) to a SDP, we have a list with most of the risks that may impact on our project. Once the project risks have been identified in the risk assessment step, these risks need to be examined in terms of specific causes and characteristics in order to narrow the focus on that which is important, avoiding confusion by focusing on unimportant details [Gemmer, 1997]. During this risk analysis step, we need to attach a value to the identified risks to prioritise them.

The risk analysis step entails conducting an analysis to determine the probability of the risk materialising, as well as the consequences associated with such a situation. The output of the risk analysis step must therefore be the RE of the identified risks as discussed earlier (section 3.2.1, chapter 3), the product of the probability of the risk occurring and the impact of the occurrence. We therefore need to estimate both the probability and the impact of the risk on the project, should this risk materialise. The purpose of risk analysis is, therefore, to discover the cause, effects, and magnitude of the risk perceived. From this output of the risks analysis step, we need to develop and examine alternative options of either mitigating the risks, or accepting the consequences during the following steps of the risk management methodology [Kerzner, 1995].

4.3.1 Causes and characteristics of risks

Charette [Charette, 1990] identified the following issues to cause a risk element to be present.

- Uncertainty in time
- Uncertainty in control
- Uncertainty of the information decisions is based on

If we could only be certain of the exact time that certain events will take place, we would have the ability to plan and react to these occurrences proactively in a pre-planned way, ensuring that our projects are not impacted negatively. Not knowing when or even if an event will occur, we may choose to ignore the possibility of an occurrence, therefore not planning for the possible event. However, should the event occur, depending on the RE of the specific risk, it may pose a definite threat to the success of our project. On the other hand, we will never be able to plan for all possible occurrences of all risks. We therefore have to make an informed guess as to whether or not a risk will materialise, as well as to when this risk will occur. Based on this "guess", we must decide how to prioritise the risks. The priorities of the risks determine which risks we plan for first, as well as which risks we choose to disregard for the moment.

Decisive decisions under uncertain conditions are sometimes required to handle risks efficiently. Uncertainty in control is a risk in itself, but this also increases the RE of other risks to projects. Inadequate authority of the relevant person(s) to make or influence decisions and inconsistency in management processes are predominant causes of risk.

As mentioned before, we need to make decisions under uncertain conditions to handle risks. We do not always have all the information we need to base our decisions on, simply because all the information is not always available or we may not be aware of it. The uncertainty involved in the trustworthiness of the information we base our decisions on is therefore also a major cause of risk to projects.

Just as we need to be on the lookout for common risks, we also need to be aware of the causes of risks in order to try to prevent these causes from aggravating the risks in projects. By identifying the causes of risks, we may be better equipped to identify and analyse possible risks.

When analysing a risk, we need to examine certain characteristics of the risks. These characteristics make it easier to attach a priority to the risk. Charette [Charette, 1990] identified the following intrinsic characteristics of risks to be taken into account when analysing risks:

- Impact
- Probability
- Time frame
- Coupling
- Uncertainty

The nature and magnitude of the risk's consequences determine the impact of a risk. The risk may have consequences or an impact in terms of the three concepts that define project success, namely cost overrun, schedule overrun, or not satisfying the customer's requirements.

The likelihood that the risk's consequences will be realised if the current situation is allowed to continue defines the probability of the risk. This value is described by the probability distribution function, where the probability function is influenced by certain variables determined by the specific circumstances experienced.

There is usually a specific time frame during which the project team can exercise proactive choices associated with the risk. During this time frame, the team exercises options to minimise the probability of the risk occurring, thereby trying to prevent the occurrence of the risk. Beyond this point, choices are eliminated and the only available option is to try to minimise the impact of the risk on the project.

The coupling associated with the risk indicates the possible effect of the risk's occurrence on other risks or opportunities. Questions to ask when determining the risk's coupling include the following: If the risk becomes a problem, will it increase the probability of other risks, increase their effect, limit the dealing choices, or reduce the time frame for making choices regarding the other risks? It is imperative that a risk with high coupling be given a high priority, since such a risk may have a domino effect by increasing the probability or impact of other risks.

The lack of understanding of the nature of the risk's probability distribution function, or how it may vary over time, influences the uncertainty that characterises a risk. The probability of a risk occurring may vary over time and may be influenced by other circumstances such as the financial position of the company, restructuring within the company, and so on.

4.3.2 Documenting the risk analysis process

"The first step to successful risk management is to write down the risks and to make them visible to all – it is harder to ignore if they are written down" [Williams, 1997]. With this statement, Williams stresses the importance of documentation to the successful implementation of risk management in software development. We need to document the important information pertaining to each of the steps in the risk management methodology.

It is extremely important to provide details about the process and rationale followed when analysing risks. If we do not adequately document the supporting rationale for our estimates of probability, impact, time frame and coupling, we will not be able to retrace our analysis reasoning should it become necessary in the future. This information may also prove to be worthwhile should we encounter a similar risk in the future.

The estimates assigned to risks have little meaning without supporting details of how they were derived. It is also difficult to communicate the estimates, to gain consensus on them and to track changes over time if insufficient details have been stored with regard to the derivation of the estimates. If the details have not been captured, participants may have different understandings of the risk – it is also very difficult to take unified action without details. The details have to be captured adequately – according to Gemmer [Gemmer, 1997], all participants must be able to understand at least 90% of the risk by examining the captured details, without any supporting explanation. The general overview of risks does not reflect subtle changes over time. However, it is necessary to reflect these subtle changes over time in order to track the effectiveness of strategies and to modify these strategies.

4.3.3 Quantitative (numerical) versus qualitative (ordinal) risk assessment scales

The output of the analysis phase of the risk management methodology should be the list of risks determined in the risk assessment phase, prioritised in terms of urgency. The risk with the highest priority should be attended to first.

The following questions arise: How do we prioritise the risks? Do we need to attach a numerical value to the risk, or can we prioritise them as high, medium or low? What will be the most effective way of prioritising the risks?

As discussed earlier, risk (RE) represents the combined effect of the probability and consequence terms. Quantitative cost and schedule risk assessment methodologies can ideally estimate risk, since the probabilities of occurrence result from the simulation process and the consequence term directly represents either cost or schedule [Conrow, 1997]. Such risk assessment tools that allow the project manager to arrive at a numerical value are useful in the sense that a percentage or number is assigned to the risk. This makes decision-making an easy task – once the risk handling capability (what risk value is the team usually able to handle without causing the project to fail) of a specific organisation or project team has been determined, it is easy to determine whether to continue with a project or not.

However, your run-of-the-mill software development team does not always include seasoned statisticians or mathematicians. We therefore have to rely on existing tools or calculations when quantifying risks. By basing our trust on a number attached to the risk, we may loose focus of other characteristics of the risk. What if we calculated the value of the risk incorrectly? What if the tool that we used to calculate the risk value did not include all the risk areas experienced within a specific company? The numerical value might not always be a true indication of the risk. In certain cases it is sometimes more beneficial not to have a numerical value assigned to the overall risk, since this can cause false security or even false alarm.

It is important to realise that it is sometimes impossible to attach a numerical value to a risk. An example of such a case can be found in the medical environment. How can we attach a numerical value to the risk that a patient might die due to the fact that the network was down and his / her medical information could not be obtained?

Ordinal (qualitative) risk assessment scales can also be used to perform risk analysis. Ordinal assessment scales do not represent risk as a numerical value. They express the level of uncertainty and / or the state of maturity for the "uncertainty" item being evaluated, or the level of consequence for the impact associated with that item being evaluated. Ordinal risk assessment scales do not require assumptions about probability distributions.

It is Conrow's [Conrow, 1997] belief that true risk values can almost never be mathematically computed from ordinal "uncertainty" consequence scales – he is of the opinion that the results from such operations on uncalibrated scales are generally meaningless and may hide true risk issues. In contrast, Gemmer [Gemmer, 1997] believes that the rationale and supporting evidence for a risk's characteristics are more important than the values assigned to them, since different parties may have different perceptions of risk values. Numbers by themselves are useless and dangerous – they imply a degree of accuracy that isn't present. The quality of decisions based on intuitive estimates (guesses) may be worse than making decisions without them [Gemmer, 1997].

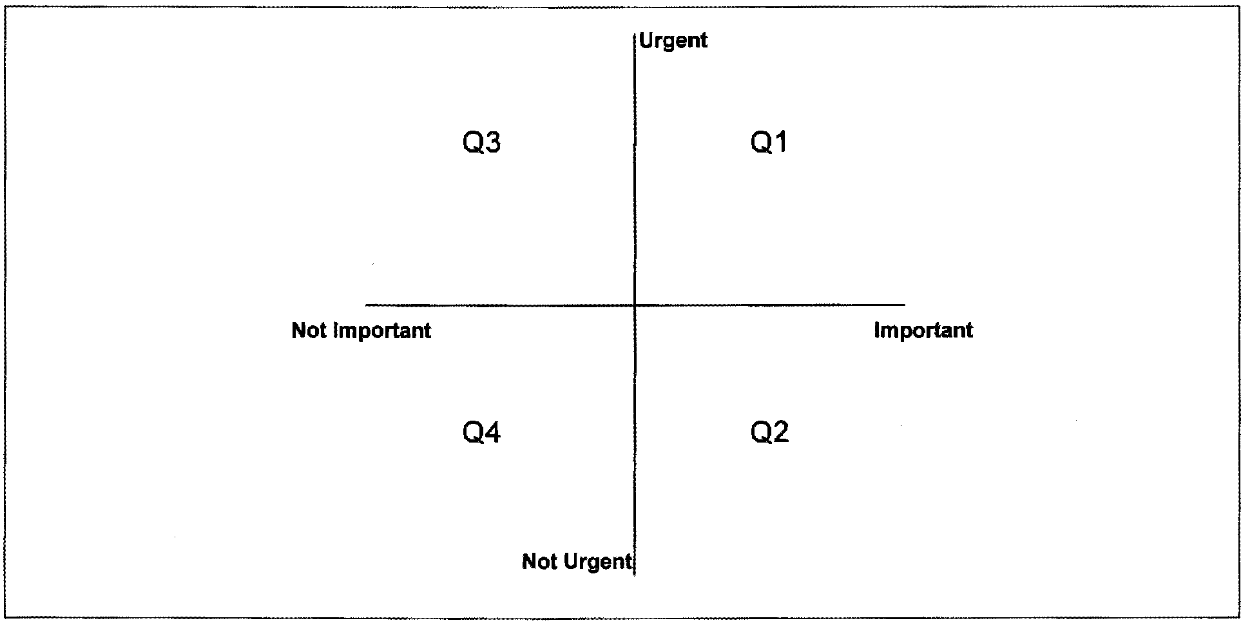
So should organisations be using qualitative or quantitative risk analysis techniques? Some say that risk cannot be adequately managed if you are not able to quantify the probability and impact. However, insistence on quantification can lead to "paralysis of analysis" [Williams, 1997] – a breakdown in communication about the risk. There are rare instances where you can assess the probability of a future event and estimate its cost. Williams found the following to be effective evaluation techniques of new risks. Work groups using "quick and dirty" estimates such as high, medium or low, or categorising risks into categories such as "need to look now", "keep an eye on it" or "ignore – not significant" were able to manage risks effectively. Ineffective evaluation techniques observed by Williams were groups trying to make a detailed quantitative assessment of probability and impact of all identified risks. The group spent as many resources evaluating the risk as would have been spent if the risk had materialised. Unless a risk has a significant impact on the program, early quantification of the impact and the probability is unnecessary.

The SEI found that most risks that they helped groups to identify were related to unstable requirements or personnel issues. They also found that none of the groups they worked with had to quantify risk probability and impact to effectively manage them. Identifying a risk is already a starting point – a risk can be managed even if it is not possible to assign a numerical value to it.

We found Covey's third habit [Covey, 1998] of the seven habits of highly effective people to be an effective ordinal way to prioritise the risks determined in the risk assessment phase. Covey's third habit is entitled "First Things First". According to this, all our activities can be divided into four quadrants. These quadrants are illustrated in figure 4.2.

Activities in quadrant 1 are important and urgent. These are generally our high-risk activities that need to be handled immediately to prevent project failure. Activities in quadrant two are important, but not urgent. We should always try to attack the risks in this quadrant and solve them before they move into quadrant one and become critical. Activities in quadrant three are urgent, but not important. This may typically include activities such as paying a bill – not paying the bill will probably not cause project failure, but is still important to remain credit worthy. Activities in quadrant four are not urgent and not important. We can handle these activities in our spare time.

Figure 4.2: Covey's four quadrants:



The key to effective prioritisation of risks is that this cannot only be done once – the priorities of risks should be re-assessed periodically. An effective way of prioritising risks turned out to be where a workgroup created a prioritised list with the top 8 risks clearly identified and the remaining risks ordered according to the most recent evaluation of their probability and impact. The top risks were periodically reordered by voting or by using pairwise comparison techniques based on group consensus about the most important criterion at the time, such as quality and cost [Williams, 1997]. Williams experienced a workgroup's weekly reprioritisation of the top risks as resulting in constant thrashing with some risks moving on and off the priority list. With limited mitigation resources, action on risks started and stopped in direct response to changes in the priority. Risks should not be re-prioritised too often. It also proved to be ineffective to plan mitigation plans for most risk statements determined during the risk assessment stage [Williams, 1997]. The effort to develop plans for minor risks turned out to be greater than the impact on the project if most risks had materialised.

In drawing a conclusion about whether to use a quantitative or qualitative risk analysis method, we have to say that both of these methods have their place. In a mission critical system where human lives may be at stake, it would probably be required to use a quantitative risks analysis method if possible, that is based on sound mathematical and statistical principles. However, in such a case, trained statisticians should be part of the project team, guiding the risk analysis effort. It is important to keep in mind that even if human lives are at stake, it may not always be possible to attach a numerical value to a risk – in such a case, we may not have any other choice but to make use of an ordinal risk analysis method. Ordinal risk analysis proved to be more effective than qualitative analysis methods in non-mission critical SDPs [Williams, 1997].

4.4 RISK HANDLING

As input to the risk-handling step, we have the risks that were identified and prioritised during the risk assessment and analysis phases. We now have to decide what we are going to do about these risks to prevent them from causing our project to fail.

It is impossible to mitigate each and every risk in a SDP. A workgroup has to be established to take the risk issues into account and to choose which of the identified critical risks from the analysis phase have to be watched for significant changes. The workgroup needs to identify which risks can be accepted, which risks they can live with if they become problems, and which risks can be assigned to someone better able to manage them. An organisation must decide on the risks they will mitigate – this is dependent on the number of resources they have available for this task, as well as the type and magnitude of risks the resources are equipped to handle. Williams [Williams, 1997] found that groups new to risk management generally have trouble dealing with more than 10 risks – more experienced groups can better judge how much effort to spend on each risk and can thus deal with more risks simultaneously. The chosen workgroup will therefore identify the 10 most crucial risks and then decide how to handle these risks.

Deciding how to handle risks can be quite difficult for an inexperienced workgroup. There are some tools or guidelines available to guide organisations in their decisions about risk handling. The Planning Flowchart of the "Continuous Risk Mitigation Guidebook" [Dorofee, 1996] is a decision making structure that assists workgroups in deciding which risks on the list to mitigate, and which risks can be handled by less resource-intensive means. This chart has a three-stage decision flow that assigns responsibility for the risk, determines the approach for dealing with it, and defines the scope of the mitigation plan. In essence, this flowchart guides a project team step-by-step in planning how to handle a risk.

Effective risk handling consists mainly of two activities. The first activity is the continuous making of informed decisions about risks, those future negative events that may affect the success of an effort. The second activity is to take appropriate action to eliminate or minimise the effect of risks. Informed decisions require sufficient information to choose among the various available options for handling a given risk [Carr, 1997]. As output of the first two phases of the risk management methodology, the risk assessment and analysis phases, we should have sufficient information pertaining to a specific risk to make informed decisions about the options available with regard to handling the risk. In some instances, it may even be worthwhile to watch and wait – however, effective risk handling makes selection of the do-nothing alternative a conscious decision rather than an oversight [Kerzner, 1995].

4.4.1 Techniques for risk handling

The handling of risks includes techniques and methods developed to reduce or control the risk. There is no real risk management if there are not provisions for handling the identified and quantified risks. Kerzner [Kerzner, 1995] identified the following categories of techniques for reducing and controlling risks:

- Risk avoidance
- Risk reduction / mitigation (prevention or control)
- Risk assumption (retention)
- Risk transfer (deflection)
- Knowledge and research

4.4.1.1 Risk avoidance

This technique of risk handling entails choosing not to accept a risk because of potentially unfavourable results. To avoid a risk is to avoid the potential consequence of failure and / or the probability of failure. This can be achieved by choosing an alternative path – one risk is often traded for others that are more acceptable or easier to deal with [Charette, 1990]. In other words, to avoid the risk, we can reorganise the project in such a way that it cannot be affected by the risk. However, not every risk can be wholly avoided – by avoiding one risk, we may actually simply transfer that risk to another area, which may imply overlapping risk-handling techniques. The key to remember when choosing this risk handling option is to make sure that the new risks that we may encounter because of avoiding the original risk do not pose greater threat to the project's success than the original risk.

What follows is an example of risk avoidance. A South African airline has a computerised system for tracing which tools were used during maintenance on specific aircraft. Traceability of tool usage on aircraft is a requirement specified by the Federal Aviation Administration. It is necessary to know if a certain tool was used for specific maintenance. For instance, if this tool is a calibrated tool, which is used to do measuring, and it is discovered that the tool was calibrated incorrectly, it is imperative to know where this tool has been used in order to check that the work performed using the tool was correct. This tool tracing system of the airline has been in use since 1988.

As part of their strategy to become Y2K ready, the airline reviewed all software systems for compliance to the date change. During the testing, it was discovered that the tool tracing system was not compliant. The airline faced two options: either modifying the old system to ensure Y2K compliance, or replacing the system with a new system.

In addition to not being Y2K compliant, none of the original programmers (since the system's inception in 1988) are employed by the airline any longer. The only documentation available is the training manual, with no documentation with regard to the coding. Many changes have been made to the system over the years, with these changes often satisfying one requirement, only to negatively impact on another part of the system. High coupling (interdependency) between different procedures and programs causes this negative impact of changes on other parts of the program. The system was also written using technology that had since become obsolete, making it difficult to find resources with adequate skills to maintain the system.

Due to the facts mentioned in the previous paragraph, it was decided to replace the system rather than change it to become Y2K compliant. By taking this route, the airline avoided the risks of finding scarce resources, and making changes to a poorly documented program. However, the risks encountered by taking this approach included the schedule risk that the new system had to be up and running before the dreaded date change, with all the users having been trained to use the new system. The airline also decided to develop the system using Intranet technology, which could pose performance risks since this was the first web-development to be done by the airline. However, taking this route also had the potential of saving time in training users since the new system is Windows-based. By using Intranet technology, the problems experienced with distributed installations would also be reduced.

4.4.1.2 Risk reduction / mitigation

Although we may prefer to avoid risks, this is not always possible. Some risks are inherently part of the development process – we need to prevent these unavoidable risks from causing project failure. When it becomes clear that we cannot avoid a risk, we need to accept the risk and do two things. We should take immediate, proactive steps to reduce the probability or the impact of the risk. Secondly, we need to define a contingency plan in order to determine the course of action to take if the risk becomes an actual problem. We also need to continuously monitor this risk in order to pick up variances from our plans, so that we can adjust our plans accordingly. The project manager or risk workgroup must take the necessary measures required to prevent or control the risk by continuously re-evaluating it and developing contingency plans or fallback positions. This risk handling strategy is called risk reduction or risk mitigation.

The ultimate purpose of risk management is risk mitigation. Risk mitigation is the act of revising a project's schedule, budget, cost, or quality so that a project's uncertainty is reduced without any significant impact on the original project objectives. Risk mitigation requires risk analysis to assess the impact of the risk on the project [Kerzner, 1995].

This risk handling strategy is aimed at reducing the probability and / or the impact of the risk. Risk control is the process concerned with the continuous monitoring of the program condition. It includes the development of options and fallback positions in order to reach lower-risk solutions. According to Kerzner [Kerzner, 1995], the best known sensors used for control are technical performance measurement (abbreviated TPM) and cost/schedule control system criteria [Kerzner, 1995]. Risk control options include developing alternative sources for production, parallel development for critical research and design components, or getting priority for critical materials.

As mentioned in chapter 2, a high percentage of SDPs fail. Sometimes these projects fail because of valid, unavoidable risks. However, a number of projects fail due to risks that could have been mitigated or avoided with little or no loss of benefit to the product, and at only marginal expense. Lister [Lister, 1997] goes so far as to term these risks “stupid risks”. It is unacceptable that projects should fail due to such risks that could have been mitigated.

Managers need good metrics to make mitigation decisions about risks. These measures assist managers in watching out for significant changes in either the risk or its mitigation plan. If the mitigation goal is clear, the workgroup and management should be able to determine when the mitigation effort is not working. If this is the case, an alternative mitigation plan will have to be used or the risk accepted. Williams [Williams, 1997] found that some workgroups new to risk management closed risks as soon as any risk mitigation activity was completed, without first verifying the effect of the mitigation activity. These “closed” risks were sometimes identified as a new risk at a later stage, with the potential impact to the program even greater then, and not preventable due to the changed circumstances. These workgroups failed to mitigate the risk – the risk then had to be accepted.

During the development of the tool tracing system for a South African airline (discussed in previous paragraph), it became evident that the development schedule would not be reached. The project manager then negotiated with the developers to work every second weekend in order to catch up on the backlog of development work. The developers were given the choice of either receiving overtime payment, or to receive time off after the project's conclusion. By working over weekends, the development schedule deadline was met. The project manager could not prevent the development from falling behind, but he could implement a plan to reduce the impact of this backlog on the development schedule.

Boehm [Boehm, 1997] found that software managers in India perceived personnel turnover as their biggest risk. There is a huge demand for software developers; therefore projects loose many of their people which may potentially result in a substantial and costly disruption of the project's schedule. The notion of RE (where RE = Probability (Loss) X Size (Loss)) helped these managers to focus on reducing the probability (loss) and the size (loss). Assessing, addressing and monitoring the annual turnover rate reduced the probability of the loss. The software managers implemented good strategies to reduce the probability of loss. These strategies included empowering performers, teambuilding, establishing significant incentive bonuses for successful project completion, recognising exceptional efforts, and structuring career paths around the organisation's product lines. They also managed to reduce the size of the loss. Software inspections were held to find defects as well as to spread information of the software product's components across the organisation. Modular software architectures and encapsulation confine the effects of personnel turnover to small parts of systems. Proper documentation further helps to reduce the effect of staff turnover. Software development files and good configuration management make it easier for new replacements to master existing software modules. In addition to reducing problems caused by staff turnover, focussing on these strategies can make an organisation more competitive in a high-turnover marketplace, while also making it a more satisfying place to work for.

4.4.1.3 Risk assumption

Risk assumption is the acknowledgement of the existence of a risk, together with a decision to accept the consequences of the risk should they materialise. All risks cannot be mitigated – most projects must assume some risks. Identification, analysis, and the selection of handling techniques assist the project manager in assuming the right (usually low) risks. Those risks that will have too great an impact to assume may be (fully or partially) transferred to a contractor at an appropriate cost [Kerzner, 1997].

The project manager therefore acknowledges the existence of the risk and its possible consequences. He makes the decision to wait and observe what happens, without doing anything to mitigate the risk. The project manager is prepared to accept the consequences if the risk should occur, and will do the absolute minimum to get by.

An example of risk assumption can once again be found in the tool tracing system discussed earlier. Using the "old" system, when a technician requested a tool from a tool store, he used to fill out a form with a unique number in barcode format on the form. A portion of this form was then detached and hung in the place of the tool in the store. This was done so that in case of system failure or down-time, it would still be possible to see who was using a specific tool by referencing the information on the form-portion that is hung in place of the tool. This number (barcode) on the form was used to uniquely reference the specific transaction. However, the form with the barcode on it is quite expensive and costs the airline money each year. As part of the new system, in order to save money, it was decided not to use the bar-coded forms anymore, but to issue each technician with a number of metal chits (small metal plaques) with the unique employee number of each technician on it. When receiving a tool, the technician would hand in one of his chits – this chit is then hung in place of the tool, making it possible to identify, without using the computerised system, which technician is currently using a specific tool.

The risk involved in this approach of replacing the barcode system with the chit system was that the technical maintenance department of the airline needed to organise the chit-system. They needed a manual system as back up to the computerised system. Should the risk have materialised that the chit system was not in place once the project was to be rolled-out, an alternative approach could have been taken, such as still using the barcode system, without the unique number (barcode) being the reference of a transaction. The crux of the manual system is to be able to see, without using the computerised system, which technician currently has booked out which tools – it is therefore only necessary to have some sort of identification hung in place of the booked-out tool.

This risk had a low RE in terms of possible impact on the project, although its probability of materialising was quite high. The project manager assumed this risk and waited to see what would happen. An alternative plan would only be made when it becomes necessary, and not before the time.

4.4.1.4 Risk transfer

This risk handling option entails reorganising the project so that someone else or something else bears the risk, such as the customer, vendor, bank or another element. By opting for this risk handling technique, the project manager shares the risk with others or transfers the entire risk. This option is typically taken for insurable risks as discussed in risk categories (section 4.2.1). This is a means of deflection – the project manager (customer) transfers all or a part of a risk to another party (contractor) by some contractual agreement. Options for the risk transfer from the customer to the contractor include product performance incentives, warranties, cost incentives, bonding, and contracting out. This corresponds to buying insurance – the contractor agrees that he will assume the consequent cost of failure for the agreed-to price. The contract type determines the degree of risk sharing between the contractor and the customer [Kerzner, 1995].

4.4.1.5 Knowledge and research

Knowledge and research as a method for risk handling, is a continuing process enabling participants to perform risk reductions through both probability and consequence modification by early initiation of development activities, implementation of extensive development testing, and the development of simulations to predict performance. The project manager develops extensive testing and simulation plans to predict the result.

To follow this risk handling technique is especially helpful in development using new technology. By first researching the concepts through building limited prototypes, the developers familiarise themselves with the development tools prior to development, thus reducing the learning curve during the development phase of the SDLC. An iterative life cycle approach is conducive to knowledge and research, since the project is divided into a number of iterations, making it easier to deliver iteration by iteration than to deliver the whole project at once.

4.4.2 Choosing a risk handling strategy

For each risk that we have identified during the risk assessment phase, we need to evaluate the risk handling strategies in terms of how it affects the risk's causes and characteristics – this assists the project manager in making cost-benefit trade-offs in risk planning. We need to find the most efficient, most cost-effective way to handle a risk in our project.

When avoiding a risk, the team must consider the characteristics and the causes of the risk; the new risk may be acceptable because of its characteristics. When mitigating a risk, the team must weigh the changes in the risk's probability and impact against the cost of the actions taken. In order to mitigate a risk, the goal and constraints must be known. The goal will usually be to eliminate the risk or to reduce the impact by a certain percentage. In mitigating risks, problem solving and analytical techniques must be used to develop strategies and to guide actions. If the risk is not acceptable, the team must consider the risk's causes – risks caused by uncertainty in time are generally more difficult to deal with than risks due to uncertainty in control of information. It is usually worthwhile to opt for risk transfer if risks due to uncertainty in time can be "traded" for risks due to uncertainty in control of information.

We must keep in mind that all is not over and done with after we have chosen and implemented a risk handling strategy. We need to monitor and control the risks that we have handled to ensure that they do not re-occur as problems to our project. It may also be possible that the risk handling strategy that we choose is not successful – we must then re-evaluate the situation and possibly choose an alternative risk handling technique.

Controlling a risk means

- altering the mitigation strategy when it becomes ineffective;
- taking action on a risk that becomes important enough to require mitigation;
- taking pre-planned contingency action;
- dropping to watch-only mode at a specific threshold, or
- closing the risk when it no longer exists.

A risk may not always be solved by following a specific risk handling strategy – it may, from time to time, be necessary to combine some of the handling strategies in order to handle risks.

4.5 EXPERIENCE GAINED

Today's major software systems are complex and costly – the increased complexity can decrease a project manager's ability to identify and manage risks. There is only one mistake that is acceptable, and that is a new mistake! Managers cannot afford to repeat past mistakes because of being unaware of successful risk management actions applied elsewhere. The smart use of information technology designed to capture risk management information allows project managers to learn from and share with others by tapping into centralised resources of system engineering knowledge, advice, and contacts [Garvey, 1997]. The last step in the risk management methodology, therefore, requires that we document the lessons about risk that we learnt through experience, so that we can utilise this knowledge in future.

As mentioned before, it is extremely important to document the risk analysis process. However, all information pertaining to risks should be documented. Even after a risk has been successfully mitigated, it is worth keeping this information. Other teams within the organisation may learn from this experience, or the risk may recur at a later stage. An effective way of retaining this information is to retain it in a risk information database after closing the risk – this information will be beneficial to the organisation on future projects. This information can also be used in order to determine or adjust the organisation's risk handling capability. Information to be kept in this database includes when the risk was closed, the rationale for closing the risk, successful and unsuccessful actions, assumptions proven wrong, mitigation costs, and project savings or return on investment (abbreviated ROI) [Williams, 1997].

Open communication about risks is central to effective risk management and to the work of the software community as a whole – one can learn from each group's approach to risk and mitigation. We need to share our experiences and the practices we found worked best. We also need to participate in experimental risk management approaches – only through this experimentation and sharing can we develop the most effective methods for managing risks in SDPs.

It will not be beneficial to document all this risk information just to file it away somewhere. Information is only useful if it is accessible and easy to understand. Williams [Williams, 1997] found electronic databases more effective than paper-based ones. All the data must be captured in this database, and integrated with other types of data such as problem and safety reports. A variety of different reports such as weekly summaries, complete information on each risk, condensed risk information for senior management, trending reports, and so on, present risk data to various audiences in meaningful ways. To have this data online and accessible to all personnel lets them enter new risks as soon as they are identified, making this information accessible to other interested parties. The value of the information will be diminished if access to the database is limited to only some personnel that must do all the data capturing and report generation – an information bottleneck may result in and out of the database leading to a breakdown in risk communication. This information does not belong to a specific individual or group of people – it should be made available to all parties so that the company as a whole can benefit from the lessons the different project teams learnt through experience.

Information concerning risks and implications of risk management actions must, therefore, flow freely across projects; otherwise incomplete or sub-optimal decisions are likely to result. Not to utilise past experiences corresponds to a definition of insanity – when a person, failing at a task, tries the same thing over and over again, expecting a different result [Charette, 1997].

4.6 TOOLS AND METHODS SUPPORTING THE RISK MANAGEMENT PROCESS

How do companies currently deal with risks? According to Lister [Lister, 1997], "the vast majority of software projects use the 'genuflect on the foul line' method: you beseech the deity of your choice to beseech his or her grace upon the effort underway." Irrespective of how religious you are, this is clearly not the optimum solution. Just as with other areas of the software development process, methods and tools to assist with risk management encapsulate some expertise, easing the implementation for the user, especially the first-time user. We will now mention some of these risk management tools in use in industry.

Through our research for this dissertation, we paid attention to how leading software development institutions manage their risks. We found that the majority of published information regarding risk management has been published by the SEI. The SEI came into existence to assist the United States DoD in establishing sound software development methodologies in order to enhance the chances of success for SDPs. The SEI is also the leading authority on risk management practices for software development. Through their publications, we identified risk management tools that are widely in use in the software development industry.

4.6.1 RiskTrak

The United States DoD stresses the importance of risk management in their SDPs in directive DoD 5000.2-R. This directive states that project managers and other acquisition managers must continually assess project risks. These risks must be well understood, and risk management approaches should be developed before any decisions can be made to authorise a project to proceed into its next phase. RiskTrak, from Risk Services & Technology has a risk management software tool designed to help meet the mandate of DoD 5000.2-R. RiskTrak uses customisable Expert "wizards" to generate project-specific databases and can be shipped with an adaptation of the SEI Taxonomy Based Questionnaire. The software can help to manage risk throughout Program Structure, Design, Assessments, Decision Reviews, and Reporting stages, as well as meeting Cost As an Independent Variable guidelines [Conrow, 1997].

4.6.2 Risk Assessment and Management Program

The Mitre organisation designed and implemented a software solution to risk management, called Risk Assessment and Management Program (abbreviated RAMP). RAMP is a risk management information system that was developed to provide interactive support for identifying, analysing, and sharing risk mitigation experiences. RAMP operates on Mitre's Intranet, allowing authorised users to access project-specific risk mitigation experiences across the corporation from worldwide locations. RAMP contains a database of system engineering related project risks and mitigation strategies, as well as links to other World Wide Web sites. RAMP is a highly integrated and connected system, allowing users to browse or query the system in order to:

- Identify and collaborate with domain experts in specific risk and technology areas
- Query experts via e-mail
- Examine RAMP's risk information templates (abbreviated RIT) that document lessons
- Obtain summaries and mitigation strategies on analogous projects

- Create custom portfolios on similar projects, their risks and mitigation strategies.

Another feature of RAMP is its RiskCheck Application – this tool can be used to provide intelligent risk suggestions. RiskCheck assists users to locate projects similar to their own, examine the risks these projects have faced and provide a list of contact people who dealt with the risks. RiskCheck supports project teams that are performing, or planning to perform, a risk assessment.

RAMP has three types of RITs namely:

- Consolidated RIT – summarises the risk mitigation strategies and experience gained that have been applied by many people in the organisation.
- General RIT – contains risk mitigation insights about particular risk areas that are independent of specific projects.
- Project RIT – contains risk mitigation experiences about risk areas associated with a specific project.

When first starting to implement risk management in a project, the project manager would typically look at the consolidated RIT in order to get an overview of the strategies followed and experience gained throughout the organisation. When needing specific information pertaining to a particular risk area, the project manager would refer to the general RIT. If only to review risk information pertaining to a specific project, the project manager would refer to the project RIT [Garvey, 1997].

4.6.3 Methods and tools in use by TRW Defence Systems Group

TRW is a company that develops and builds advanced technology for ground, space, and high-energy laser systems for the United States DoD. Capitalising on a growing privatisation trend within the government and industry, TRW secured a number of outsourcing contracts in 1998. These contracts include a $240 million award from the U.S. Air Force Strategic Command to manage its data processing activities, a $264 million contract to manage the Defence Travel System, and a $187 million contract from the U.S. Census Department for Census 2000. In the commercial information technology market, TRW received a $70 million award from Wheeling Pittsburgh Steel Corporation to outsource and modernise its information technology and telecommunications systems [TRW, 1999]. TRW is clearly a leader in the field of software development.

Excessive, immature, unrealistic or unstable requirements were the main reasons why some of the TRW's large software-intensive projects experienced cost overruns and schedule slips. To address the major risk, the TRW Ada Process Model was used to specify current procedure requirements and plans for the likely directions of growth and change. Parnas' information hiding techniques were used in order to modularise the software architecture to facilitate anticipated changes. Other methods that assisted the project team to contain the potential impact of risk were the use of a defined and agreed-upon change control process, metrics to track requirements growth and stability, and allocated capabilities to the various increments.

The TRW establishes a risk review board (abbreviated RRB) for each project, led by the project manager. The RRB meets on a monthly basis during the project's intensive requirements definition, design and integration phases to apply an iterative risk management process with feedback to the involved parties. Each risk is documented, including a description of the risk type, severity, risk mitigation plan, and status of the risk mitigation activity. All members of the project team, as well as the users, are encouraged to identify risks during any management or technical meeting. Once a risk item is placed under the control of the RRB, a risk mitigation plan is created and assigned to the responsible person. The RRB then evaluates and approves the plan for implementation and review the status of the active risk items, mitigation plans and modified mitigation plans. The risk items are tracked until the risk is reduced to an acceptable level. Throughout this process, the risk items are discussed with the customer through daily interactions and project management reviews of the top 10 risks or concerns. The early risk analysis and decisions typically resolve many of the user concerns.

It was found that large projects with human computer interaction often failed to meet user expectations due to the lack of user involvement. As part of early reviews, extensive prototyping to demonstrate the functional capabilities was used to address these risks. The prototype evolved into the operational system – the users could therefore see the displays and interactions to be built into the final system before project completion.

Performance shortfalls can usually not be identified prior to system integration, which usually occurs only after coding and unit level testing are complete. The TRW System Integration Group started system integration during the design phase and could therefore address performance issues early by using an iterative approach to software development. The TRW also used the Cocomo Cost model to determine a workable combination of functionality, budget and schedule to address the risk of unrealistic cost or schedule estimates.

Ineffective project management caused problems in the SDPs of TRW. It was found that frequent project reviews, together with the use of the relevant metrics, ensured that potential problems were identified early and handled before affecting the problem significantly. Using consistency checking during the design phase rather than waiting until integration testing to check interfaces, accomplished significant integration prior to coding. This addressed the risk of ineffective integration, assembly and testing, and quality control. Monitoring metrics throughout the project development to ensure that the process model was working addressed the risk of immature or untried design, processes or technologies.

Based on the TRW's prior experience, detailed plans and solid configuration control are very important to the success of the project. The TRW developed, maintained and used detailed work plans and configuration management procedures throughout the development cycle. The use of automated version control and software change tracking process, further assisted them in handling the risk of inadequate work plans or configuration control. The quality assurance personnel conducted periodic audits of the various configurations to ensure that the configuration control was maintained from the beginning of the project up to the completion of the operational system.

It was found that potentially good people left TRW's employment due to poor training. To address the risk of poor training, the required and suggested training for each software engineer on the project was tracked, with training plans drawn up and adhered to in order to accommodate the career development plans of each of the participants in the project [Conrow, 1997].

4.6.4 Risk spreadsheets

A risk spreadsheet can help workgroups summarise risk information for the top risks (those risks to be mitigated). Such a spreadsheet must include the risk identification number uniquely identifying which risk is referred to, and a risk statement concisely stating the risk. The spreadsheet should also include the priority attached to the risk, the probability and impact of the risk, who it is assigned to and what the current status of the risk is. Those risks for which the mitigation strategies are proving to be ineffective, should be flagged to indicate that action is required [Dorofee, 1996].

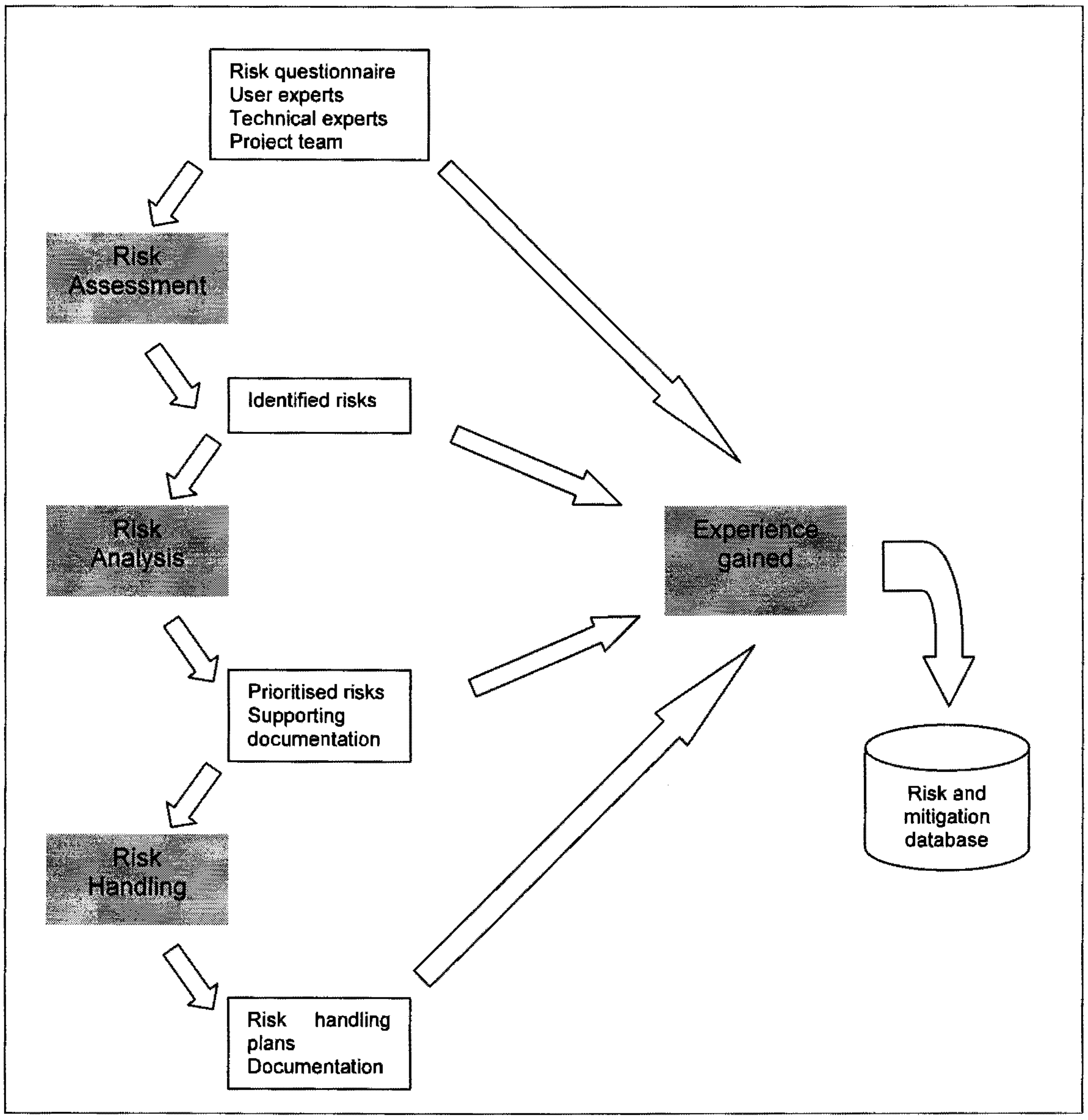
The Mitigation Status Report [Dorofee, 1996] can be used for critical risks that need complex tracking reports – this report provides for the effective portrayal of RE versus time [Williams, 1997]. Verbal reports and no documentation proved to be ineffective in tracking risks and mitigation plans – the workgroup could not remember what had been done or which risks were still open.

Whenever risk management is first implemented in an organisation, it is crucial to have some way of documenting and retrieving the information pertaining to the risks. It is not necessary to have a software tool that supports the whole risk management effort before implementing risk management. A basic electronic database, allowing all interested parties to browse or add information, will add value to the risk management process since project teams will be able to learn from each other's experiences, therefore preventing the same mistakes from being made over and over. As mentioned in this section, a risk spreadsheet can also add value to the process. Workgroups making mitigation plans or agreements without any documentation on the follow-up finally produced ineffective risk mitigation [Williams, 1997].

4.7 CONCLUSION

Our aim with this chapter was to propose a possible risk management methodology for SDPs. The different steps that should form part of a typical risk management methodology, with their respective inputs and outputs, are summarised in figure 4.3.

Figure 4.3: Inputs and deliverables of the steps of risk management



We discussed possible methods for implementing each of the steps of the risk management methodology, giving examples to further clarify the concepts. After reviewing this chapter, any software development project manager should have a fairly good idea as to what a risk management methodology should include – a project manager should also be able to use some of the mentioned methods as a starting point for implementing risk management.

Decisions are central to the risk management process. We need to constantly make decisions such as which risks will have the highest priority, who need to handle the risks, and so on. In the following chapter, we consider how decisions about risks are made under uncertain conditions, as well as the factors that influence these decisions.

5 DECISION MAKING IN RISK MANAGEMENT

5.1 INTRODUCTION

Decisions are very important in all projects, especially large-scale projects. The consequences of a few or even one risky decision can have disastrous effects, ultimately causing project failure. We do not always have all the information pertaining to what the effect of a certain decision will be, and it is seldom possible to predict this. Even though we may have information about the separate risks involved, this knowledge cannot prevent mistakes in the individual's intuitive judgement of the effects of decisions regarding risks.

As discussed in the previous chapter, managing risks means that risks must be assessed and analysed, risk-handling plans must be developed, and those plans must then be put into action. It is here, where the plans must be put into action, that decision-making information is generated [Carr, 1997]. A quality decision-making process with risk as overriding concern is essential – the decision-maker must actively search for risk in every decision, assess the risk to see if it is too great, and if it is, perform some positive action to reduce it. The primary characteristics of such a decision process are that it must be visible, repeatable, and measurable. This ensures that a base level of consistency is achieved. Understanding develops as to why and how decisions were made, how to improve future decisions and how decisions can be reversed with minimal damage [Charette, 1997].

The decision process must furthermore pragmatically support how project members make decisions involving risk on a daily basis. This process must be embedded into the everyday work practices – the process must support different decision makers' perspectives and the work contexts should help project managers or programmers answer specific questions about any risk or combination of risks involved in the decision making relevant to their work environment. It is imperative that risks are assessed at a fine level of granularity so that informed decisions can be made. However, risks must also be assessed at a level course enough so that actions taken to manage them are efficiently implemented.

In this chapter, we focus on decision making with regard to the whole project life cycle, starting with the decision whether to attempt a project or not as well as a multitude of decisions that is required during the project life cycle.

5.2 RISK HANDLING CAPABILITY

Not all organisations are capable of handling the same risks. For instance, the US DoD should be able to handle much higher risks than a software company developing educational software, since the DoD has more experience with high-risk projects, and possibly a more mature software development process.

The risk that an organisation is prepared to take must always be less than or equal to the risk the organisation is capable of handling. Each organisation should have a threshold defined for the risks they are capable of handling – no project with risk higher than this threshold should be attempted, since these projects will in all probability result in failure. The better the software process model used by an organisation, the higher the risk that the organisation will be able to handle.

So how do we determine the risk handling capability of an organisation? The level of a company on the CMM will play a role in determining the risk handling capability of the company. This risk handling capability of a company is extremely important since it must influence decisions about projects such as whether to continue with a project, or to terminate it.

5.2.1 The Capability Maturity Model

The CMM was first put forward by Watts Humphrey [Humphrey, 1989] of the SEI of Carnegie Mellon University. The CMM is not a software process model, but a strategy for improving the software process, irrespective of the process model used. The CMM provides guidelines to assist organisations in providing an infrastructure to achieve a disciplined and mature software process. The improved process should result in improvements in software quality, and the reduction of time and cost overruns.

The CMM aims to introduce change incrementally, since improvements in the software process cannot occur overnight. The CMM defines 5 different maturity levels. These maturity levels are an indication of how advance the software processes are that is used by an organisation. An organisation advances slowly in a series of evolutionary steps toward the higher levels of process maturity. The ultimate goal for any organisation should be to reach level 5 of the CMM.

To clarify this approach, we now describe the five levels of maturity [Schach, 1993]. Each maturity level builds on the processes of the previous level.

5.2.1.1 Maturity Level 1: Initial Level

At this level, the organisation typically does not have sound SE management practices in place. There is usually no proper project planning, and most activities occur in response to some or other crises. The software development process is unpredictable and totally dependent on the current staff.

5.2.1.2 Maturity Level 2: Repeatable Level

At this level, the organisation has basic software project management practices in place. The main difference between level 1 and level 2 is that at this level, the organisation learns from experience and attempts not to make the same mistakes. Measurements are taken, and goals are set to try to improve the software process.

5.2.1.3 Maturity Level 3: Defined Level

In an organisation that has reached the third maturity level of the CMM, the software production process is fully documented. In addition to complete documentation, the managerial and technical aspects of the software development process are also clearly defined, with clear role definitions of management and technical staff. At this level, new technology can be implemented to improve the quality and productivity of the software production process.

5.2.1.4 Maturity Level 4: Managed Level

At this level, project success is virtually guaranteed since the software development process is clearly defined and repeatable, and lessons are learnt from experience. The software development team can now focus on quality and productivity – quality and productivity goals are set for each project. Quantities with regard to quality and productivity are continuously measured and corrective action is taken when deviations from the set goals occur.

5.2.1.5 Maturity Level 5: Optimising Level

In an organisation at the optimising level, the goal of all SDPs is to continuously improve the software development process. At this level, statistical quality and process control techniques are used to guide the organisation. The knowledge gained from previous projects is utilised in future projects.

Improving the software development processes of an organisation does not occur overnight. Experience with the CMM has shown that advancing a complete level usually takes from 18 months to 3 years. However, moving from level 1 to level 2 sometimes take 3 or even 5 years – this is a reflection of how difficult it is to instil a methodical approach in an organisation that has been functioning on a purely ad hoc and reactive basis [Schach, 1993]. However, once an organisation reaches level 2, there are at least some defined, repeatable processes in place and therefore some discipline in the software development process. To move from level 2 upwards is thus easier than the jump from level 1 to level 2.

The CRM and TRM methods and tools developed by the SEI are best used by organisations on maturity level 2 or higher. In organisations on level 2 or higher, the risk handling capability of the organisation can be defined company-wide. However, this does not rule out risk management for projects in a level 1 organisation. In such a case, the risk handling capability will not be determined for the company, but rather for the specific team working on a specific project – at level 1, the risk handling capability for a specific project depends on the members of the software development team. The project manager must drive the risk management process in such a case [Carr, 1997]. The problem with risk management in a level 1 company is that the risk handling capability is dependent on team members, therefore the risk of a project will be significantly increased by staff turnover.

The risk handling capability of an organisation is directly related to the maturity level of the organisation – an organisation at maturity level 5 will be able to successfully complete projects with a much higher risk than an organisation at maturity level 1. By improving the software development processes, the organisation will therefore also improve its risk handling capability, leading to more projects being successfully completed.

5.3 WHEN TO TAKE A RISK

To risk or not to risk – that remains the question! In the previous section, we discussed the risk handling capability of an organisation at the hand of the CMM. An organisation should never attempt a project if the risk of the project is higher than the risk handling capability of the organisation, or of the project team in the case of an organisation at level 1 of the CMM.

We distinguished between insurable and business risks in section 4.2.1 (chapter 4). In the presence of insurable risks, we would always prefer only to take low risks, since the materialisation of an insurable risk can only lead to a loss. However, where business risks are concerned, an organisation may either benefit or loose from taking the risks. Organisations may not always be able to pursue only low-risk ventures, since risk sometimes represents a competitive advantage. Some higher risk ventures need to be pursued from time to time to capitalise on this competitive advantage. However, risks need to be managed so that the types and quantity of the risks are appropriate to the business needs as well as the risk handling capability of the organisation, otherwise the risk will be a wasted resource, possibly leading to project failure.

For the risk management effort to be effective with a positive outcome, risk management must start *before* project execution. The risks to a project must be determined up-front to the SDLC. The decision of whether to go ahead with a project must be based on the risks of the project – if the risk of the project is higher than the risk handling capability, the project should not be attempted. Once the development process has started, expectations are set and risk management is aimed at managing the risks – once expectations are set, they must be met irrespective of the risks encountered.

Charette [Charette, 1995] pointed out that risk management can be approached as a proactive, entrepreneurial activity. "Whenever they can, risk entrepreneurial companies reposition themselves to be progenitors of risk (mostly in the form of innovation) by causing one or more flash shifts to occur. The more shifts risk entrepreneurial companies can cause, in more arenas, the more risk they can manipulate, and the harder it is for competitors to keep up" [Charette, 1995]. An example of such a risk entrepreneurial activity is Hotmail.com, a website where people can obtain a free e-mail service. By providing a free e-mail service, the creators of Hotmail.com took a great risk, since they moved away from the traditional source of income of subscriptions for e-mail service providers. However, Hotmail.com became an instant hit, with revenue being generated via marketing and advertising. The creators of Hotmail.com therefore "created" a risk for traditional e-mail service providers, giving themselves as the authors of the risk time to deal with the new risk of not receiving any subscription fees, generating income through advertising.

Charette [Charette, 1995] furthermore states that we must develop an extra sensitivity to risk or more appropriately, a "risk-taking ethic" which holds that

- success entails taking on risks, sometimes very great risks, but doing so intelligently;
- risk is not something to be feared or avoided, but something we can profit from;
- change is not feared, but embraced;
- by mastering the details, the risks can be mastered as well.

One must keep in mind that every decision has the potential to cause negative consequences. The high uncertainty coupled with high decision stakes dictate that every decision, irrespective of who the decision-maker is, must be at a level of acceptable risk to the greatest possible degree. Each and every decision must therefore be assessed for risk and this risk must be at a level in accordance to the risk handling capability of the organisation to be accepted.

Management must articulate the point of unacceptable risk, where the project is no longer beneficial, the risks outweigh the rewards, or the project turns out not to be doable in its present form. To continue stubbornly past this point believing something can be salvaged is foolish and creates problems when the idea is attempted in future with a different approach [Charette, 1997]. It is often difficult for a manager that has been closely involved in a project to admit that the risks outweigh the rewards – it is a difficult decision to stop a project, as it is often perceived to be admitting to personal failure. However, it is imperative that management be objective when evaluating the viability of a project – the decision of whether to proceed with a project or not should be based on the facts of the project situation.

The question remains: When is it acceptable to take a risk? It is acceptable to take a risk if the risk handling capability of the organisation taking the risk is higher than the RE of the potential risk. Taking low risks is usually acceptable, but high risks should only be considered if their potential rewards outweigh the potential losses that can be caused by the risk.

Although it may seem clear-cut whether to take on a risk or not if we consider the risk-handling capability of an organisation, decisions are not always that easy to make. We need to discuss the risks with the stakeholders before making decisions. How do we go about discussing risks, and how do we make decisions involving these risks? What is an effective decision process?

5.4 RISK DISCUSSIONS AND DECISION-MAKING STRUCTURES

The most effective decision process accounts for all decisions from the initial project definition time until the project is retired. It is only when the decision process is fully documented that we can trace our steps to understand our reasoning in making certain decisions. We need this tracing ability, since we need to trace back our steps if something went wrong. Only by tracing back and determining exactly which decision caused the problems can we rectify the situation, or if that is impossible, at the very least learn from our mistakes.

The project risks are central to the decision process. The project risks must be the first input to any initial project definition and must be continually reassessed throughout the project's development and operation [Charette, 1997]. The risks in our project must dictate the decisions we make.

The goal of any risk discussion is to reach consensus on a risk's characteristics, as well as the action plan for dealing with the specific risk. However, disagreements often stem from different people using different reasoning or data to reach their conclusions. Individuals rarely have both the breadth and the depth of knowledge to act on only their own knowledge – the best decision will more likely be made from discussions among people with contrary beliefs, than from one person in isolation. To be the most meaningful, discussions should be focussed on areas of differing opinions rather than on common ground [Gemmer, 1997]. Each person with a specific belief differing from someone else's belief should be given the opportunity to explain the reasoning behind his belief. By explaining his reasoning, the other members participating in the risk discussion will have the opportunity to point out possible flaws in the reasoning, or be persuaded that the reasoning makes sense. An open risk discussion usually leads to the risk being better understood, with the derived risk-handling plan taking into account more of the risk's characteristics.

However, we still need to reach consensus on the risk characteristics. Gemmer [Gemmer, 1997] suggests that the risk's elements should be viewed at different levels of abstraction in order to understand it. To move down abstraction levels to eventually gain consensus, Gemmer used the Inference Ladder, a learning tool developed by Action Design [Action Design, 1995], a training institution. Table 5.1 explains how the Inference Ladder works.

Table 5.1: Inference Ladder as applied to Risk Discussion [Action Design, 1995]

Abstraction Level	Can we agree on ...	... using this risk information?
Highest	What should we do about this situation?	Risk-handling strategy and action plan
↑	What is our evaluation of the situation?	Risk statement and risk's probability, impact, time frame and coupling.
↓	What reasoning are we using to reach this evaluation?	Rationale for risk's probability, impact, time frame and coupling
Lowest	What data are we using to support our reasoning?	Evidence, root causes, and risk tolerance

At the highest level, the risk consists of a set of choices in the form of the plan of action. The risk's statement and characteristics, the reasoning that led to their estimated values and the evidence underlying that reasoning, support the proposed choices. The Inference Ladder guides discussions by helping participants to focus on the areas of disagreement, instead of focussing on those areas in which agreement has already been reached. If the participants cannot agree on the highest level (the choices to be made), the discussion moves to the next level of abstraction to focus on the evaluation of the situation (represented by the risk statement and it characteristics). If the participants still disagree at this level, the discussion moves to analysis, which leads to the formulation of the risk's characteristics (as discussed in section 4.3.1, chapter 4).

An example of a discussion using the inference ladder is given in table 5.2 below. This example should clarify how the Inference Ladder is used.

Table 5.2: Example of dialogue with the Inference Ladder [Gemmer, 1997]

Individual	Statement	Action to take
Person A:	"I don't agree with the action to be taken on this risk."	Recognise differing opinion
Person B:	"I base my choice on high probability of occurrence. Do you agree with the risk's probability?"	Move down a level of risk characteristics
Person A:	"I don't agree with the estimated probability. It is too high."	Identify area of disagreement
Person B:	"I believe the risk is highly probable because the probability is driven mainly by the likelihood the supplier will fail to deliver on time. Do you agree that it is the driving factor?"	Move down a level to supporting rationale
Person A:	"I agree they may be late but we won't need the part on the delivery date so it is not as likely to impact us if they slip a little. Do you agree?" and so on.	Identify use of different rationale

Following the levels of abstraction illuminates the need for additional investigation and data collection. The discussion therefore becomes a learning process – the capturing of new information, positions and actions prevent the same discussions from recurring. The discussion may stall at times when the team identifies evidence supporting or refuting certain positions. Stalling in the discussion can also be caused by difference in risk tolerance – each person's risk tolerance (the risk an individual feel comfortable with) may vary from situation to situation. Different business ventures may require tolerance levels that differ from that of the individuals involved [Gemmer, 1997]. In a situation like this, where the risk tolerance of the participants of the risk discussion differ, we should consider the risk handling capability or risk tolerance of the organisation within which the risk discussion takes place.

Gemmer [Gemmer, 1997] investigated decision-making structures in a number of organisations. He found that people typically presented risks in reviews where little discussions occurred. If the presented risks were discussed, the discussions did not always lead to consensus on how to handle the risk – these reviews had no way of resolving misunderstandings and disagreements. As mentioned before, the whole point of a risk discussion is to reach consensus on the risk's characteristics and to formulate a mitigation plan. If this is not achieved through the risk discussion, the discussion fails and does not add value to the risk management process.

Gemmer [Gemmer, 1997] also noted the following points about the risk discussions:

- The discussions typically surrounded decisions about situations with a good deal of uncertainty.
- The review meetings tended to be rehashes of the previous meetings.
- No one person has all the knowledge to make a decision.
- Depending on their role in the program, people prioritised what they heard differently.
- Almost all review discussions were about opportunities, risks, and problems. Most people called them "issues" – it was found that this term was perceived to sound safer and more positive. The "issues" list contained most of the uncertain and potentially dangerous decisions.

Little change in risk discussions was observed when implementing commonly accepted risk management techniques such as risk questionnaires. The team identified more risks, with each risk having a number or word associated to it signifying probability and impact. However, no explanation or discussion of uncertainty was attached to these estimates. In a project, if a common perspective of a risk cannot be reached, the team members may make decisions that drive the project in different directions. Also, risk is a perceived value – the estimates of probability and impact can be biased. Risk communication must provide perception as well as information – how people perceive risk must be monitored, since the decision-making process must act on that perception [Gemmer, 1997]. It is therefore imperative that the reasoning behind a certain decision be well understood and documented, since the decision itself does not explain the circumstances that led to it.

To solve the deficiencies in the risk discussion and decision process, a fundamental shift in perception is necessary. People need to realise that risk is nearly everything being discussed and that risk principles can be applied to any discussion. To move to this new way of thinking, a radical departure from the previous way of thinking has to take place. Previously, in traditional project management, we only focussed on the tasks that had to be performed, without giving much thought to what may go wrong. Whenever we identify a task to be performed, we should identify the risks that may impede on the successful finalisation of the task.

The meaningful discussion of risk should focus on *what* is discussed, *how* it is discussed, and the *actions* taken as a result [Gemmer, 1997]. Senior management should ask specific questions in order to create a "pull" for information – managers should be coached to recognise certain cues as triggers for certain questions to be asked (e.g. the words "issues" or "problems").

The following guidelines should be followed to discuss risk intelligently:

- Use precise terms
- Distinguish between risk, opportunities and problems
- Distinguish between uncertainty and probability
- Make the causes of risk explicit
- Provide details of the characteristics of the risk (probability, impact, time frame, coupling)
- Devise strategies for dealing with the risks (mitigate, avoid, transfer or accept)

5.5 DEALING WITH UNCERTAINTY

Kerzner [Kerzner, 1995] categorises decision making as follows: decision making under certainty, risk, and uncertainty. When making decisions under certainty, we assume, with certainty, that all of the necessary information is available to assist us in making the right decision, and we can predict the outcome with perhaps 100 percent confidence. As we progress from certainty to risk to uncertainty, the potential damage to our project caused by a wrong decision increases.

In reality, situations in which everything is certain do not often exist. However, the level of uncertainty may differ from one situation to the next. The difference between risk and uncertainty is that under risk there are assigned probabilities, and under uncertainty, meaningful assignments of probabilities are impossible.

We will always find an element of uncertainty whenever a decision needs to be made. Uncertainty may be present in the knowledge on which we base our decisions, our ability to carry out the actions associated with our decisions, or the effectiveness and side effects of our actions. Communications should be focussed on two types of uncertainty, namely uncertainty in perceiving risk impact and uncertainty in perceiving risk probability [Gemmer, 1997].

5.5.1 Impact perception

When making decisions under uncertainty, it is not possible to work out exactly the impact that a risk may have on the project. In a situation like this, it is not possible to use a quantitative analysis scale (refer to section 4.3.3, chapter 4), but an ordinal scale may assist us in attaching a "value" to a risk.

One criterion used by Gemmer [Gemmer, 1997] for reporting risks in project reviews is that a risk has a significant impact, regardless of its probability of occurrence. Senior management wants warnings. In order to provide such warnings, project teams must define "significant" in an impact model, where this model defines the thresholds of variances in performance to the project's expectations. The impact model provides a common scale against which to measure different types of impact.

Gemmer [Gemmer, 1997] performed studies with regard to impact models in a number of organisations. Each project team developed its own impact model until they realised that their own definition of a "big" or "significant" risk may not correspond to those of other projects, or to what senior management perceived as acceptable or not. It became obvious that the project team's perception of performance variances differed from those of the project's stakeholders. Senior management had to develop organisation-wide guidelines to assess impact.

The following is an example of an impact model. A risk is termed as significant if it can cause a project to slip 10 percent or more on its schedule. A risk is also termed as significant if it can cause overspending of 5 percent or more on the budget. If a risk causes performance to be more than 5 percent under the requirements, it is termed as significant.

The above model may have to be adjusted for different projects. For instance, if a project had a schedule of three months, a five-percent slippage on its schedule only entails a one-week slippage, which may be quite acceptable and therefore not a significant risk impact. However, if the project schedule was three years, five-percent slippage means slipping two months, which may not be acceptable in this case. It may therefore be necessary to adjust the impact model for certain projects.

This impact model is a two-way communications tool – it acts as a downward communications tool in the sense that senior management's desire for feedback from each project is communicated. It also outlines organisational guidelines for establishing priorities and making trade-offs. The impact model helps project teams to better understand their stakeholders' sensitivity to variances in multiple, often conflicting objectives. This provides additional perspective to the team's decision making. Through tailoring the organisation-wide impact model for each specific project, project teams communicate upward how their specific expectations relate to the organisational expectations. The project team can then communicate potential variances (risks) in terms that the stakeholders can appreciate. This improves the chances of the project team to receive the help it needs from management to manage risks.

5.5.2 Probability perception

The project team and senior management may also have different perspectives of the probability of a risk occurring. Different people have different connotations of words such as maybe, likely, certainly, unlikely, probably, possible, improbable, doubtful or expected, which are used to describe probability in ordinal risk analysis scales. Due to these differences in perception, the rationale and supporting evidence for a risk's characteristics are more important than the values assigned to them. Without rationale, the decisions made are based solely on the probability and impact, which is, according to Gemmer, little more than gambling [Gemmer, 1997], since we cannot say for certain what the probability and the impact of a specific risk are under uncertain circumstances.

In section 3.2 (chapter 3) we defined RE as the product of the probability and the impact of a risk. According to this formula, a risk will have a high RE if it has either a high probability of occurring, or a high impact. For the purpose of this section, let us focus on risks with a high RE because of a high probability, but with a low impact on the project. Assume this risk will occur, since it has a high probability. However, the impact on the project will be quite low – we have to ask whether it is really necessary to give much attention to such risks – will it not be easier just to assume these risks and handle their impact when they occur? Instead of wasting time on risks with high probability but low impact, we should focus on those risks with high probability as well as high impact.

When focusing on these risks with high RE due to high probability as well as high impact, we should identify the possible causes of the specific risks. We should then focus our efforts on preventing the risk from occurring by reducing its probability. These possible causes of the risk provide the rationale behind our decisions concerning the specific risk.

5.6 CONCLUSION

We constantly need to make decisions during the life cycle of a project. The first decision to make involves whether a project should be attempted or not. The answer to this question depends on whether the organisation or project team is capable of implementing a project with the associated risk or not. An organisation should never attempt a project with a RE higher than the risk handling capability of the organisation.

This risk handling capability of an organisation or project team is directly related to the level of maturity of the organisation with regard to the CMM. The more mature the software processes in use by an organisation, the higher the risk that the organisation will be able to handle. To improve their risk handling capability, organisations should therefore strive to increase the level of maturity of the software development processes used.

Once the decision has been made to go ahead with a project, decisions form a central part of the project life cycle. We need to decide on the priorities awarded to risks, how we will handle risks, who is responsible for what, and so on. Prior to any decision, the different stakeholders need to discuss the possible decisions, carefully considering the pros and cons of each decision. It is very important to document these risk discussions, since this information needs to be available if it becomes necessary to trace back why a certain decision was made. All the supporting rationale with regard to a certain decision needs to be documented so that it is clear from the documentation how and why a certain decision was reached. This information can then be used for future references when similar decisions need to be made.

It is important that the stakeholders agree when a certain decision is reached. It is not always easy to reach consensus on the rationale supporting decisions. The inference ladder can be used to guide risk discussions to a level of consensus. This inference ladder allows the risk discussion to move down abstraction levels to eventually gain consensus. This method of discussion guides the participants to focus on areas of differing opinions, reaching consensus on these areas, instead of focussing on areas of agreement.

The circumstances under which decisions need to be made can vary from certainty, risk to uncertainty. The potential damage that may be caused to our project by a wrong decision increases from certainty to risk to uncertainty. When making decisions under risk, we need to consider the risk exposure of the risks involved. The impact model can assist us in defining the impact perception general to an organisation. It is more beneficial to focus on risks with high impact that on risks with low probability, since the risks with high impact may have a profound negative effect on our project if they materialise.

In the next chapter, we consider how to go about implementing risk management in software development for the first time in an organisation.

6 IMPLEMENTING RISK MANAGEMENT

6.1 INTRODUCTION

In the chapters leading up to this chapter, we gave an overview of risk management in SDP's. We explained that risk management came from the need to improve the success rate of SDPs. We also looked at the evolution of risk management in SDPs, paying attention to risk management frameworks applied in industry. We then discussed a general risk management methodology, applied specifically to the software development process. We attempted to clarify the concepts of risk management by providing ample examples from the software development industry. Finally, we discussed and gave guidelines for effective decision-making structures and procedures where risks are concerned. These chapters up to now have thus given an overview of the whole risk management process in SDPs, answering the following questions asked in the problem definition:

- Why do we need to practice risk management in software development?
- What do project managers need to know about risk management before attempting to practice it?
- How do we make decisions about risks?

In this chapter, we focus on how to go about implementing risk management in software development for the first time, highlighting the potential benefits to be achieved when practising risk management.

Boehm [Boehm, 1997] found that software managers would be more inclined to acknowledge and manage their risks if they were more aware of comparable organisations that had already chosen to manage risks. Minimal evidence is published that software risk management is beginning to affect many companies and government agencies – the reason why little is published may be that doing software risk management makes good sense, but that publishing this exposes companies to legal liabilities. If a software product fails, the existence of a formal risk plan acknowledging the possibility of such a failure could complicate and even compromise the producer's legal position. Just because there is not much evidence that organisations are doing risk management, does not mean that software development companies do not exercise risk management in practice.

In our research for this dissertation, we found a wealth of books and articles on risk management in general, with a few directed specifically at software development. The information specifically related to risk management in SDPs contained ample theory, with scant information about practical experience. Most of the articles or textbooks focussed on a specific portion of the risk management process, with little guidance for the first-time implementation of risk management. We observed the need for an easy-to-use, informal guide to implementing risk management in SDPs for the first time in an organisation, without it being necessary for the implementation team to make a detailed study of, or be experts, at risk management.

This chapter provides guidelines for implementing risk management in SDP's in an organisation. We highlight potential problems that may be encountered when implementing risk management for the first time. By implementing the procedures discussed in this chapter, a project manager should be able to start managing risks in SDPs effectively. The steps in this chapter provide a starting point – the risk management process should evolve within a company to become more effective with time, as we learn from experience.

6.2 IS IT NECESSARY TO DO RISK MANAGEMENT?

From our discussions in chapter two, it became evident that it is necessary to manage the risks to our SDPs in order to increase the success rate of our projects. As a motivation to start using risk management, we can look at the existing success rate of SDPs in our organisations. From this, it should be obvious that those projects in which the risks are managed, even in an informal way, have a better chance of success than projects in which the risks are ignored. By identifying and dealing with risks early in the SDLC, we can reduce the long-term costs and prevent disasters [Boehm, 1991].

Kerzner [Kerzner, 1995] is of the opinion that risk management can be justified in almost all projects. The level of implementation can vary from project to project, depending on factors such as size, type of project, customer, relationship to the corporate strategic plan, and corporate culture. When asked whether it is necessary to do risk management and why, Lister [Lister, 1997] answered: "Because*no* project ever runs exactly as planned!" We have to agree with this statement of Lister – projects fail if everything does not go as planned. The high failure rate of SDPs (section 2.2, chapter 2) is proof that *something* has gone wrong *most* times!

Now that we have concluded that it is necessary to perform risk management in SDPs, we need to know how to go about implementing risk management as part of our software development process.

6.3 TREAT THE IMPLEMENTATION OF RISK MANAGEMENT AS A PROJECT

The implementation of risk management in software development can be treated as a project itself. As people with a software development background and experience, we understand why it is necessary to follow a life cycle model in our implementation of a software project. We can actually follow a life cycle model for the implementation of risk management too. In this section, we look at the basic life cycle phases of the SDLC, starting with the specification, applied to the implementation of risk management. We then consider planning, design, implementation, integration, and lastly the operational scenario of risk management in an organisation. We will incorporate into the life cycle of our project the identification and handling of risks that may cause the risk management implementation effort to fail.

Initial implementation of risk management in an organisation will correspond to a linear model such as a waterfall model. However, the risk management practices in an organisation will evolve and improve as experience in the field is gained, learn from mistakes and building on successes. It would therefore be an iteration through the stages of the SDLC, until an acceptable level of risk management is reached in the organisation.

6.3.1 Requirements specification for implementing risk management

As with the specification document of a SDP, the requirements specification for the implementation of risk management as part of software development must contain information of *what* is needed for proper risk management. The needs, or specifications, for successful risk management are summarised in the following three points [Gemmer, 1997]:

- Repeatable risk management process – a process that is visible and measurable, which allows it to be repeated and improved upon (Maturity level 2)
- Widespread access to adequate risk knowledge – knowledge sources fuel the risk management process.
- Functional behaviour – behaviour deals with human interactions, motivations and incentives, perceptions and perspectives, communication and consensus, and decision-making and risk tolerance.

This specification defines *what* is needed to implement risk management, and not *how* to do it. How to satisfy the requirements incorporated in the specifications will be determined during the design phase.

As we have been propagating throughout this dissertation, it is extremely important to identify the risks to our projects as soon as possible in the SDLC. Just as we do with SDPs, we need to analyse the identified risks, prioritising which of these risks pose the greatest threat to the project. We then have to try and avoid these risks by reducing the probability of the risk occurring, or alternatively making plans to mitigate the most crucial risks.

6.3.2 Risks associated with implementing risk management

We need to assess the risks that may impede on the successful implementation of risk management in our organisation. In this section, we look at problems (risks) that are commonly experienced when initially implementing risk management. We also consider guidelines and possible solutions to overcome these risks. A risk-averse culture and its effect on the implementation of risk management is discussed. In addition, we consider an immature software development process, inadequate management infrastructure and risk documentation, lack of continuous risk management, inadequate decision-making processes and the lack of commitment from key stakeholders. We discuss the effect of inconsistent risk management methods between client and supplier as well as problems experienced when focussing either on probability or impact, without taking the other into consideration. By addressing these problems, we will enhance the chances of successfully implementing risk management in SDPs.

6.3.2.1 Risk-averse culture

The culture of an organisation generally depends on the history of the organisation, the management structure, processes followed and the type of people that work for the organisation. For instance, we may find that organisations with younger management tends to have a more relaxed culture, materialising in different ways, such as flexi-time, a relaxed dress-code, less red-tape etc.

Gemmer [Gemmer, 1997] did an investigation into current risk management practices in organisations. He found, in many respects, that risk was a four-letter word, used only by a few. These results were consistent with the cultures of other organisations – this behaviour was due primarily to the organisation's history, structure, processes, and reward system.

When attempting a project, we generally do not want the project to be "risky", except under exceptional circumstances such as business risk where the ROI would be significant. A project's chance of success should be significantly greater than the likelihood of failure; otherwise the project might not be approved. Some project managers perceive admitting to "riskiness" as admitting to not fully understanding the problem, being pessimistic or not being a team player. This association of risk with something being wrong leads to cognitive dissonance – a belief that is held in spite of evidence to the contrary [Charette, 1997]. The culture within organisations should be changed so that the risk stigma is removed – all projects have risk and it is no shame to admit to that. In fact, hiding the risks will cause more damage than admitting to them.

Executive management is responsible for ensuring a risk-aware as opposed to a risk-averse culture at all levels of an organisation. This means that executive management recognises all development work has risks – hardware as well as software. Executive management should be concerned about tracking and controlling risks, and ensure that this message has been received by all in the organisation [Carr, 1997]. Management should encourage project teams to identify as many risks as possible to their projects. Traditionally, the culture in software development organisations has been risk averse – by admitting to risks, project teams felt as though they were admitting possible failure and therefore preferred to hide or ignore the risks, focussing on positive aspects of the development effort. In these circumstances, it was often perceived that the team member who brought up the risk was to blame for it. Management should move away from this "shoot-the-messenger" culture and rather reward team members when they identify critical risks, even if these risks become problems. By identifying risks early, project teams can attack and mitigate these risks before they cause a crisis in the software development effort. The aim of risk management is to prevent crises by proactively managing risks, instead of reactively handling the crises.

The culture of the organisation should change as not to reward crisis management. It is often encountered that the person fending off a crisis is perceived as the hero saving the project. However, in most cases the fact that a crisis occurred is actually a sign of bad management. If the risk(s) that caused the crisis had been identified and managed in time, the crisis would never have materialised. Heroism should be viewed as problem avoidance, and not problem solving.

Telltale signs of a risk-averse culture include the following [Gemmer, 1997]. Management does not trust all people to make decisions – they equate decision-making capability with IQ. People are not allowed to make a clean start after failure; instead of being allowed to learn from experience, people are judged by past performances and not given a chance to improve on them. People do not trust others to make decisions and do not share information, since information is perceived to be power. Management usually tends to reward the "lone rangers".

Furthermore, in a risk-averse culture, uncertainty is almost always perceived as a negative. Rather than giving thought to risks, team members believe that the team cannot fail by denying the uncertainty of events and decisions. The team tends to think of everything as either black or white – they are only able to do this by ignoring those issues that they cannot be certain of. The team members also do not expect variances in performance – if they were successful in the past they will expect success again. However, they will also disregard the opinions of those people that have failed in the past, since they do not expect variances in performance. This is unfortunate, since it is actually those people that failed in the past that have the experience that we can learn from.

In such a culture, risks are perceived as something that can always be solved. Project teams tend to ignore the "soft stuff"; engineering aspects are kept away from business and marketing aspects, and management tasks are viewed as administrative overhead. Teams usually only deal with technical issues and solutions, ignoring the people issues. The prevailing sentiment is that "we cannot go wrong". Past decisions are always perceived to have been correct and will not be reversed, irrespective of the circumstances. Decisions are usually not made until the outcome is guaranteed – this makes decision-making easy, since options are eliminated by the passing of time, usually reducing the options left. However, by eliminating the options, the remaining options may not be sufficient to ensure project success. Decisions are based on emotion, intuition or gut feeling rather than logic. Closure is never reached on difficult issues – people talk about these issues, but do not document these discussions. Silence is perceived as a sign of consensus or agreement, instead of as a lack of information. Project teams are reactive in the sense that they deal with the symptoms of problems rather then the causes – they deal with immediate and specific problems, rather than systemically dealing with possible causes of problems.

Commitments are made without determining the probability of success – people believe that they will be successful somehow, without investigating how they will obtain success. New requirements or constraints are added to the project without questioning if success is still feasible, and success is promised unconditionally to the client. Probabilities are assessed intuitively, without any scientific evidence or procedure. Project teams always only plan for the best-case scenario since they do not take into account that something may actually go wrong.

In a risk-averse culture, project teams pretend projects can be made to succeed by sheer force of will. The best people are assigned to crises, assuming that a miracle will once again occur to save the project. Hard work is rewarded, instead of smart work, and the comeback is usually glorified.

The risk-averse culture goes hand in hand with maturity level 1 of the CMM. With organisations at this level, there is no repeatable process of software development. People tend to make the same mistakes over and over, without learning from experience. Everybody is usually so relieved when a project is finally implemented, that they do not want to hold a "post mortem" to learn from their experiences.

6.3.2.2 Immature software development process

In the previous section, we discussed the effects of a risk-averse culture on the software development process. We saw that one of the trademarks of such a culture is that people do not learn from past experiences, thus repeating mistakes from the past. It is very difficult to change such a culture in an organisation still on maturity level 1 of the CMM.

It is Carr's [Carr, 1997] opinion that risk management cannot be instituted within the confines of one project manager's project. If executive management is averse to risk, that value will be known through all the levels of the organisation – people "down in the trenches" will know that talking about risks is not condoned and will therefore not sound an alarm when they become aware of risks. The project manager thus gets blindsided by critical problems because the risk information is not communicated. It should be the organisation's first priority to move from level 1 to level 2 of the CMM before attempting to change the culture from risk-averse to risk-aware.

The lack of a systematic and repeatable method to identify, analyse, and plan risk handling inhibits organisations' ability to manage risks effectively [Carr, 1997]. This inability to do proper risk management will in all probability persist if an organisation is on level 1 of the CMM, since no repeatable processes are present at this level. However, should the organisation move from maturity level 1 to maturity level 2, repeatable processes can be established to identify, analyse and plan risk mitigation. Executive management should guide the organisation in reaching the next level of maturity. If it is left up to individuals to increase the maturity of the organisation, this will in all probability never happen, since the individuals will not have the authority to enforce the changes that are necessary to move to the next level of the CMM.

6.3.2.3 Inadequate management infrastructure

As already mentioned, two of the main problems experienced when trying to implement risk management are those of a risk-averse culture, and the lack of a repeatable, systematic process for software development. With both these problems, it is up to management to make the necessary changes to establish repeatable processes, as well as a risk-aware culture. However, to be effective, these changes need to be made across the organisation, and not just in individual sections.

In order to be able to make these changes, a good management infrastructure, uniting management in a common goal, is required. If managers of different sections operate independently without taking into account the goal of the organisation, it will be very difficult to make these changes organisation-wide. Management will not be able to bring the necessary changes about if it is not agreed throughout all the sections that the changes need to be implemented. It is therefore imperative that a strong management infrastructure exists binding management together, so that management can act together to bring about the necessary changes.

The closer you get to executive management, the fewer and less severe the risks identified and discussed. SREs done by Carr [Carr, 1997] uncovered consistent evidence of this: the severity of the risks identified was inversely proportional to the observer's hierarchical position within the company. At the lowest level, people consistently ranked risks higher than did respondents at the project level and above. Executive management should therefore initiate risk communications with people at all levels in the organisation – only then will they have a true view of the risks experienced in the organisation.

To perform proper, effective risk management, we need a project management team that can provide leadership and actively control risks, even if all risks are fully and completely understood by all parties potentially affected, and risks must be continuously and visibly managed. As mentioned before, management must be proactive as opposed to reactive. Management must do more than "merely taking the initiative" - they must create a culture where risk is not synonymous with disaster [Covey, 1989]. Proactive risk-taking project management stresses co-operation, collaboration, integration, and balance of action [Charette, 1997]. This means that everybody works together – management together with the project teams, to prevent risks from materialising.

Risk management is often viewed as a self-evident activity – software projects "obviously" involve risks that need to be managed. The problem is that few managers see any compelling reason to make risk management a separate formal activity, which is very unfortunate. In Charette's [Charette, 1997] experience, the project success is severely limited by the perspective that risk management is a self-evident activity. Executive management should make managers and other employees of an organisation aware of the importance of risk management as a definite activity of the project management process. By specifically identifying and focusing on risks, risk management can be proactive and effective, handling risks before they become problems. However, if risk management is seen to be a self-evident activity of the project management process, risks will probably only be identified and managed once they cause a crisis in the SDLC of the project.

Some organisations initially feared that risk management would open them up to micro-management by senior management – implying that senior management would want to be involved at a detailed level in the project. Gemmer [Gemmer, 1997] observed the opposite to be true – some of the highest risk projects exhibits the best functional behaviour. By making senior management aware of the risks, the team got the support and help needed from management. Senior management tends to be more accepting of surprises from teams doing a good job of risk management, than from those teams hiding or ignoring the risks to their project. These projects, where executive management was made aware of all risks, trained the entire project team and practised risk management as a team activity. These teams discussed risk using the most details, since they had nothing to hide from executive management. Teams have done significantly better at practising risk management than individual team leaders – if only one person tries to implement risk management, it will probably fail since it should be a team effort to succeed.

We therefore conclude that if executive management is not supportive, risk management at the project level is an uphill battle that will in all probability fall short of achieving the desired results. We need a solid management infrastructure, uniting management in the goals of effective risk management, in order to bring about the necessary changes for effective risk management.

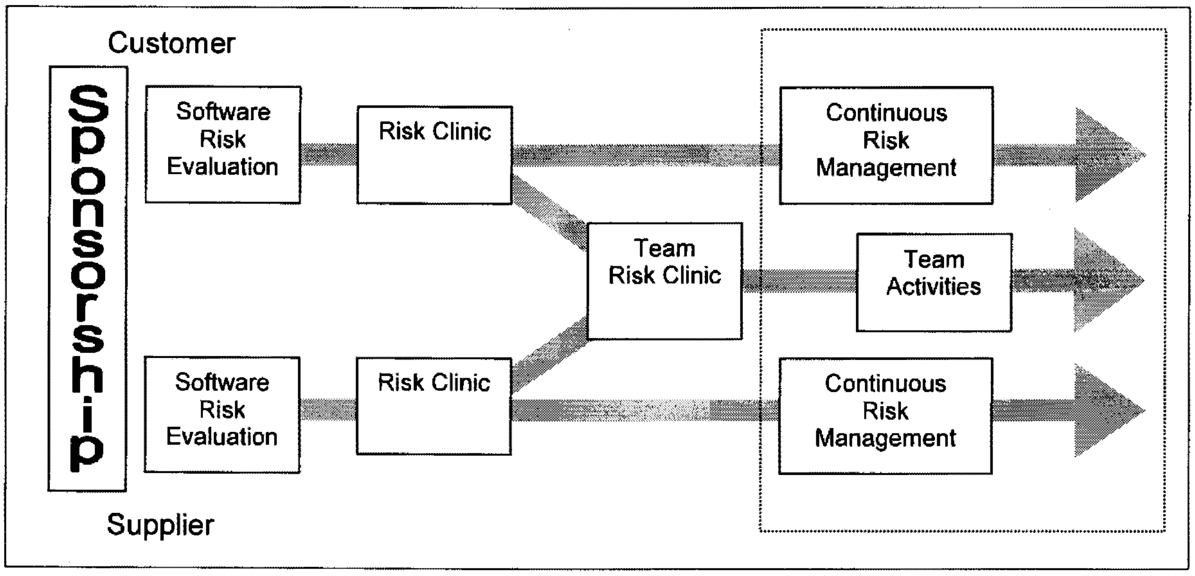
6.3.2.4 Inconsistent risk management methods between client and supplier

The DoD observed risk management to be so important that they set policies in place emphasising system and software risk management. However, these policies were not sufficient to achieve successful system and software risk management. Conrow [Conrow, 1997] observed some risk management deficiencies in several DoD development programs – these deficiencies were also often found in civilian development projects.

The first deficiency observed is that the client and supplier often have weakly structured or "ad hoc" risk management processes with no single set of guidelines as to how the risk management process should be implemented. The problems caused by inconsistent risk management methods between client and supplier include that they do not have a common understanding of the risks to the project. Furthermore, if both parties have their own risk analysis and risk handling methods, they may actually be trying to handle the same risk simultaneously, working against each other instead of together.

The SEI developed a road map to address this specific problem of inconsistent risk management processes between client and supplier. Figure 6.1 shows the road map used by the SEI for a complete installation of SEI risk management in a project that includes both a customer and supplier organisation.

Figure 6.1: The risk management road map [Williams, 1997].



Any organisation can follow this road map, even if it does not use the SEI methodologies or tools. Even if projects are mostly in-house development, this road map can still be followed in the sense that the customer will be the user, while the supplier will be the development team. In such a case, the users and the development team can work closely together in the risk management effort. This road map highlights the fact that the development team should listen to the concerns of the users (customer), since the users may identify risks that may have been overlooked by the development team. The roadmap furthermore stresses the importance of having a project sponsor – the sponsor should be visibly involved throughout the project life cycle.

Following the SEI Road map, the first step is to conduct a risk clinic with individual members of the customer and supplier work groups. This is an extended workshop in which the group's key leaders determine the best way to adapt and install methods and tools to support each phase of the risk management paradigm that they will be adopting. The CRM then continuously refines this risk management process. Step 2 is the team risk clinic. This clinic is similar to the workgroup risk clinic – the goal of the team risk clinic is to establish methods and tools for inter-organisational risk management between the customer and supplier, and among all organisations on which the project's success depends such as users, subcontractors and vendors [Williams, 1997].

Perspectives may differ in the larger team environment of the team risk clinic. The basic goal of the TRM is that those committed to the project's success must have the same understanding of all relevant perspectives, and manage risks accordingly [Williams, 1997].

6.3.2.5 Inadequate documentation

Another deficiency observed in the risk management process of DoD projects, is that risk analysis is often too subjective and not adequately documented [Conrow, 1997]. Risk analysis should be based on specific guidelines making the process transparent, so that anybody can understand how and why a specific outcome was reached. This process of analysing risks should be documented thoroughly, explaining exactly how and why certain assumptions and decisions were reached.

We should always be able to go back on our steps in the risk management process to understand why we made certain decisions and what we should change the next time to improve the risk management process. In the discussion about risk-averse culture (section 6.3.2.1), we mentioned that project teams often have a tendency to discuss serious issues without reaching consensus or a plan of action. These discussions are usually not documented, and therefore the project teams tend to waste time in discussions that have no positive outcome or solutions to the issues being discussed.

We have stressed the importance of adequate documentation throughout this dissertation. The documentation should show exactly why a certain decision was made, outlining all assumptions and information on which the decision was based. Documenting this process ensures that discussions reach a point where the participating parties reach consensus – if nothing is documented, discussions may go on for quite a while without any specific decisions being made.

6.3.2.6 Probability and impact

Conrow [Conrow, 1997] found that risk assessment processes generally emphasise the probability associated with a specific event, with little attention afforded to its consequence. However, the RE of a risk is a combination of an event's probability and its consequence – both factors must therefore be analysed and tracked over time.

As discussed in section 5.5.1 (chapter 5) in the context of an impact model, it is often easier and more useful to focus on the impact of a risk rather than the probability of the risk materialising. It can be quite difficult to determine the probability of an event occurring, especially since we do not always have historical data on which to base statistical analysis. It may prove easier to determine the impact of the risk by asking "What if...?" questions, looking at the worst-case scenario of what will happen if a risk materialises. If the impact of the risk is insignificant or can be rectified without much trouble, it may be worthwhile to ignore this risk. However, if the risk may have a significant impact and there exists a reasonably probability of it materialising, this risk must be mitigated. The probability and impact should be considered when analysing a risk; otherwise the analysis may not give a true indication of the RE of the risk.

6.3.2.7 No continuous risk management

Project risk assessments and risk-handling plans in DoD projects were found to be often unlinked [Conrow, 1997]. This is a mistake that organisations often make when first implementing risk management. Risk management is often perceived as only identifying and analysing the risks. Risk handling plans are then prepared on an as-needed basis, with limited tracking against the key project milestones.

To be effective, risk management cannot be implemented as ad hoc actions on a need-be basis. Although risk management must be seen as a specific activity of the software development process, it should be integrated closely into the software development process. Events in the software development process should therefore trigger certain actions in the risk management process. Risk management should be a continuous effort of identifying, analysing and handling risks. This loop is triggered every time a new risk to the project is identified, or whenever a risk previously managed becomes a potential problem again.

The lack of an infrastructure to support continuous risk management makes it at best a one-shot activity – Carr [Carr, 1997] terms this the "risk management season". This typically happens in an organisation at maturity level 1 of the CMM when either an action of a customer or a crisis within the project makes the project manager realise that too many things are going wrong. The project manager will then decide that something must be done to bring the situation under control. The effect is usually that the project manager sees risks as problems needing attention – thus, the top 10 problems will be managed rather than looking at the future to determine the risks potentially faced by the project.

If nothing is done to mitigate a risk, it will eventually become one of the top 10 problems. In Carr's [Carr, 1997] application of the SRE method, he saw many instances where the project managers foresaw a risk and took mitigating steps, only to be diverted by a current crisis and never to return to the original mitigation strategy. It happens quite often that people are allocated to risk mitigation until a crisis occurs, never to return to the risk mitigation effort, until the risk that was being mitigated becomes a crisis. This problem will persist until an infrastructure of continuous risk management to ensure risk handling is established. If the project was supported by continuous risk management from the start, the first crisis might have been averted, and therefore the interference with the risk mitigation efforts too!

6.3.2.8 Lack of training and coaching

Another mistake that organisations often make when first implementing risk management is to assume that all employees will be able to follow guidelines on their own and implement risk management effectively. Risk management is not an exact science, but requires participants to acquire certain skills to become better at managing risks. Team members will acquire these skills sooner under the guidance of an experienced person.

Risk management training can be given to participants in the form of case studies where team members can participate in a "mock" risk management exercise on a project for which good documentation is available in terms of its risk management. All participants should become aware of terms such as "issues" and "problems" that may point to possible risks to a project. The team members should also be coached as to how to prioritise risks. The importance of continuous risk management should be brought up again to make the team aware that changing the priorities of the risks too often can only lead to risks moving in and out of the top 10 (n) risks. This will cause mitigation proceedings to stop, start, stop, ... and thus not be effective in the sense that the risk cannot be handled and permanently eliminated from the project once and for all.

The aim of the coaching and training should be to make team members aware of risks generally associated to SDPs, as well as for the team members to become familiar with the whole risk management process before actually implementing it. Once the team members implement risk management in their projects, they will continually learn from their peers as well as their own experience, becoming coaches to other, less experienced team members.

6.3.2.9 Lack of communication

Open communications concerning risks within the project team, and between the project team and executive management is extremely important. Open communications forms the backbone of the SEI risk management framework (fig. 3.6, chapter 3). Teams working together are much better equipped to manage risks than an individual – by considering risks from the different viewpoints of the team members, aspects of the risks may be considered that would have gone unnoticed if it was up to one person to analyse the risk. In a risk-averse culture, information is often perceived as "power" and therefore not shared. Everybody needs to realise that they have a better chance of dealing with risks if they work together.

Charette [Charette, 1997] found that many large-scale, extremely complex and unprecedented DoD projects that had been perceived to be very risky were declared to be "low-risk" in order to obtain future funding. The project team felt they had to hide the risks from executive management in order to obtain the funding they needed. Given this mindset, it is difficult to make a case for performing risk management. However, if executive management (in this case the US government) had made it clear that they want to be made aware of all risks, not necessarily to cancel a project, but to assist in managing the risks, the project team might have acted differently. It is evident from this example that risk management is doomed if open communications between all parties concerned do not exist.

6.3.2.10 Project scope

It is wrong to assume large-scale projects are just like small ones, only bigger...! [Charette, 1997]. The problems experienced with software development actually increased as technology advanced – projects became bigger and more involved, requiring more sophisticated SE tools. However, the SE tools usually only evolved after problems were experienced in software development. The waterfall model actually worked quite well with projects of limited scope, but as projects became more involved, the success rate of software projects implemented using the waterfall model deteriorated. Accurately predicting the cost or schedule at the start of a large-scale project that is long-lived, is highly unlikely. The data available is usually insufficient, with the environment prone to change.

We cannot always handle all risks in a software project at the same time – we prioritise the risks in our projects and handle those risks with the highest priorities first. Williams [Williams, 1997] found that project teams are usually only able to effectively handle eight risks at a time. However, what if there are more than eight risks, or more risks than we can handle, that may have a significant negative impact on our project if not handle immediately? If this is the case, we need to look at an iterative SDLC model, as opposed to traditional linear models such as the waterfall or prototyping models.

The spiral model (section 3.3.1, chapter 3) was the first SDLC model to attempt to divide large-scale projects into smaller iterations, focussing on a smaller portion of the project at one time and reducing the number of risks that needed to be managed at a specific time. The spiral model, as well as other iterative SDLC models such as the RUP, allows abstraction – the development team can focus on issues important at a specific moment in time while ignoring issues that will not immediately affect the outcome of the software development process. The RUP actually bases the iterations and prioritisation of the iterations on the risks in the SDP. Those iterations with the highest risk have the highest priority and are implemented first in order to reduce the risks to the project as early as possible in the SDLC.

Large-scale projects do not always have a definite end. By using an iterative SDLC model, milestones can be declared in terms of the products of iterations – this gives the project team short-term goals instead of having to wait until finalising the whole project before seeing tangible results. As shown in figure 3.4 (chapter 3) earlier, each iteration of the RUP results in an executable release.

It is important that, when we implement risk management in a large-scale SDP, we divide our project in such a way that we are able to handle the risks of every iteration at a specific time.

6.3.2.11 Lack of commitment from key stakeholders

A stakeholder is a person or representative of an organisation who has a stake – a vested interest – in the outcome of a project. A stakeholder can be an end user, a purchaser, a contractor, a developer, a project manager, or anyone else who cares enough about, or whose needs must be met by the project [Kruchten, 1998].

We saw that the two most important factors playing a role in project success are user involvement and the involvement of executive management (figure 4.1). Just as is the case in SDPs, the commitment of the users, in this case the software development teams, together with the commitment of executive management can make or break the risk management implementation effort. It is important that executive management supports and oversees the implementation of risk management, pointing out its importance to the organisation. However, if the software development teams do not support risk management and do not follow the guidelines for effective risk management, the implementation of risk management in the organisation will just be another failed project.

Without the commitment of the people that need to make it happen, risk management will not be implemented successfully.

6.3.3 Planning the implementation of risk management

During the planning phase of the SDLC, we determine the tasks that need to be executed to implement a project, as well as who will be responsible, and the due dates for the specific tasks.

As specified in section 6.3.1, the requirements for risk management are as follows:

- A repeatable risk management process that is visible and measurable, which allows it to be repeated and improved on.
- Widespread access to adequate risk knowledge and information.
- Specific functional behaviour that deals with human interactions, motivations and incentives, perceptions and perspectives, communication and consensus, and decision-making and risk tolerance.

We need to determine what we have to do to satisfy the requirements.

6.3.3.1 Establishing a repeatable risk management process

As per definition of the CMM (section 5.2.1, chapter 5), organisations at maturity level 1 (the initial level of the CMM) do not have sound SE practices in place. At this level, organisations also lack proper project planning, with crises-management typically ruling the day. Such a software development process is almost impossible to predict, and totally staff-dependent.

Before we can establish a repeatable risk management process, we need to have basic repeatable software development and project management practices in place. When implementing risk management, we therefore first have to determine whether the organisation is at level 2 of the CMM. If this is not the case, the first objective of our effort to implement risk management should be to reach maturity level 2.

6.3.3.1.1 Improving the maturity of the software development process

It is typical of an organisation at level 1 of the CMM to have ad-hoc software development processes, dependent on the resources. Each resource or project team will therefore follow their own way of developing software since there is no defined method of software development prescribed by the organisation.

The organisation must first adopt a software development methodology. The organisation may also opt to buy CASE tools that support the chosen methodology – however, it is important to keep in mind that tools only support the methodology – it is up to the software development teams to follow the process, using the tools to support the processes of the methodology. It is the responsibility of executive management of Information Technology to assign the responsibility of choosing and establishing a software development methodology. Some organisations have an Architecture department that must look into issues like standards and methodologies – the function of selecting a methodology will typically be allocated to such a department.

6.3.3.1.2 Establish a repeatable risk management process

Just as the software development processes in an organisation can mature (improving the level of the CMM), the risk management process followed by an organisation will improve as experience is gained and deficiencies in the process identified and rectified. It is very important to first establish a basic risk management process – this process can be followed and improved on with time.

We discussed a general framework for a risk management methodology in detail in chapter 4. In order to have a repeatable risk management process, we need to determine which tools or methods we are going to use for each of the phases of risk management, namely risk assessment, analysis, handling and experience gained.

Executive management of Information Technology should assign the responsibility of compiling a risk management process to a specific team. We suggest that an expert in risk management is assigned the position of team leader.

6.3.3.2 Widespread access to adequate risk knowledge and information

Establishing a knowledge base of risk information and allowing all parties that may benefit from this information to access and update this database is imperative if we want to be able to utilise our experiences to improve our risk management skills and processes.

Whether we decide to buy a risk management product that incorporates such a database or implement our own database, a project team needs to be assigned to this task. In addition to implementing the risk management knowledge base, the team will be responsible for drawing up the operational procedures for the database, such as who is responsible for backing up the data, issuing user accounts, etc.

Once we have made the decisions regarding tools and methods to be used for the four phases of the risk management methodology, we need to implement the tools and methods we decided on.

6.3.3.3 Establishing the necessary functional behaviour

Functional behaviour deals with human interactions, motivations and incentives, perceptions and perspectives, communication and consensus, decision-making and risk tolerance [Gemmer, 1997]. In order to establish the correct functional behaviour that is required for effective risk management, the employees of an organisation have to be convinced that risk management will improve the success rate of software development within the organisation, making software development easier in the long run. It is the responsibility of executive management to induce a risk-aware culture in the organisation. The implementation of risk management in an organisation is bound to fail if the culture remains risk-averse.

Executive management needs to realise that a paradigm shift is necessary from traditional project management to risk management. In traditional project management, project teams focussed on the tasks at hand, dealing with problems as they occurred. However, in risk management, project teams must focus on the risks associated with the tasks at hand, as well as the tasks to be done in future, trying to circumvent the possible problems. Project teams must move away from reactive problem solving to become proactive in their efforts of finding solutions to problems.

6.3.4 Design and implementation of risk management

In the previous section, we focussed on the planning phase of the risk management implementation effort. In this section, we consider possible implementations for the identified tasks.

6.3.4.1 Improving the maturity of the software development process

As mentioned before, an organisation at the initial level of the CMM does not have a repeatable software development process in place. To rectify this situation, we suggest that organisations adopt an iterative software development process such as the RUP discussed in chapters 2 and 3. This SDLC model is an iterative model aimed at reducing risk to a project as early as possible in the SDLC by implementing those iterations with the highest risk association first. The RUP defines the inputs and outputs of each of the life cycle phases, guiding organisations in terms of what is required to implement the RUP. The RUP also provides guidelines for documentation, with documentation templates. Since documentation is extremely important in establishing a repeatable software development process, the documentation templates ease the task of project teams since these templates guide teams in the correct implementation of the RUP.

Once an organisation has established the use of the RUP or another SDLC methodology, the organisation should be on maturity level 2 of the CMM. We can now commence the implementation of a repeatable risk management process.

6.3.4.2 Repeatable risk management process

We need to choose and implement specific methods and tools for the different phases of the risk management methodology as discussed in chapter 4. In this section, we consider possible options for the implementation of a repeatable risk management process. It is important to keep in mind that the risk management process can and should be improved as we gain experience. It is therefore not required to establish a "perfect" risk management process the first time we start to implement risk management. The risk management implementation process should be iterated through continuously as we gain skills and experience, improving the risk management process with time.

6.3.4.2.1 Risk assessment

For the risk identification phase, we suggest adopting a risk questionnaire that will guide project teams in identifying the risks to their projects. However, it is imperative that the adopted risk questionnaire be evaluated to ensure that it contains all the risk areas that have been identified to play a role in our projects. The categories and risks discussed in section 4.2.3 (chapter 4) can be used as an example against which existing risk questionnaires may be evaluated. This evaluation model should also be extended over time, as more risks are identified through experience.

The organisation should define guidelines determining who should be involved in the risk assessment effort. This identification of risks to a project should be a joint effort between the executive sponsor of the project and the development team and the users, with a risk management expert guiding the process if possible. The same people should be involved in the risk analysis effort.

6.3.4.2.2 Risk analysis

For a first time implementation of a basic repeatable risk management process, we suggest that organisations use a simple, easy-to-understand ordinal risk assessment model such as either the impact model discussed in section 5.5.1 (chapter 5) or Covey's third habit as discussed in section 4.3.3 (chapter 4). To implement numerical risk assessment usually requires more expertise and statistical information.

6.3.4.2.3 Risk handling

Risks can be handled by either reducing the probability of the risk occurring, or by reducing the impact that the risk will have on the project. A defined method for risk handling does not exist – the risk handling method depends on the characteristics of the specific risk. However, as experience is gained, information will be kept on how risks were handled in the past. Should the same or a similar risk therefore occur later in the same or in another project, the information pertaining to how the risk was handled in the past could give an indication of the best way to handle the risk.

6.3.4.2.4 Experience gained

We can only learn from our mistakes and experience if we have information available pertaining to our past experiences. We must therefore make provision for an electronic database in which we can capture this information. We can either buy a commercially available product that supports this function such as RAMP (section 4.6.2), or we can develop and build our own knowledge base. Procedures must also be set in place which encourage project teams to refer to this information, as well as to capture their own experiences with as much detail as necessary.

6.3.4.3 Establishing the necessary functional behaviour

Executive management must establish the functional behaviour required for effective risk management by making the employees of the organisation aware of what the required behaviour is. This can be done at risk management workshops or discussion groups. Sending documentation around and expecting employees to read through and understand it has proven to be ineffective.

The biggest change required in functional behaviour is to become proactive as opposed to reactive in problem solving. Executive management can make this change clear by changing the reward system immediately. In a risk-averse culture, heroism was associated with crises-management. The person who solved the crisis was usually rewarded as the one who saved the day. However, by managing risks, crises should be averted instead of handled. The people who identify and prevent risks from occurring should therefore be awarded instead – it is better to prevent a crisis than to solve it!

Through focussing on behaviour, Gemmer [Gemmer, 1997] observed that risk management is not separate from real work; rather, it is how we think about, talk about and do real work. We tend to acknowledge that we may possibly encounter problems when executing the tasks at hand. By making a conscious effort to handle these problems or risks, we give ourselves the opportunity to prevent disasters from occurring.

Gemmer [Gemmer, 1997] identified the following desired functional behaviours:

- Risk should be managed as an asset.
- Treat decision-making as a skill that can be taught, practised, and constantly improved.
- Create a pull for risk information – actively seek this information, conduct meaningful discussions about the risks, and then act on them.
- Seek diversity in perspectives and information sources – listen for and learn from divergent viewpoints.
- Minimise the uncertainty in time, control, and information – systematically search for uncertainty – this search is very important for a learning organisation.
- Recognise and minimise the bias in perceiving risk – make decisions based on sound information derived from adequate analysis of the situation.
- Plan for multiple futures – include the best case, worst case as well as the most likely scenario.
- Be proactive – act before things go wrong. Attack the root causes.
- Make timely, well-informed decisions and commitments – understand when decisions must be made – manage the risks and understand the chances of success before making commitments.
- Reward those who identify and manage risks early, even if the risks become problems. "Heroes are not just problem solvers; they are also problem avoiders".

6.3.5 Subsequent iterations

Initially, we must establish a basic repeatable risk management process. After the implementation of this initial repeatable risk management process, we need to iterate through the implementation process to continually improve the process.

Improvements may include opting for another risk identification questionnaire, or extending the existing questionnaire. It may become necessary to discard the ordinal risk analysis method in favour of a numerical method. We may also improve our documentation process.

Risk management is fast becoming a mature discipline. Boehm [Boehm, 1997] feels that to achieve the promise of fully effective software risk management, the software industry must address several continuing challenges. These challenges can be addressed in iterations subsequent to the initial risk management process. Issues to be addressed include:

- Achieve commitment of all key stakeholders (developers, customers, users, maintainers, and others) to a risk management approach.
- Establish evolving knowledge base of risk management experience and expertise, organised for easy and collaborative use by all stakeholders.
- Define and propagate mature guidelines on when and how to avoid, prevent, transfer, or accept and manage risk.
- Develop metrics and tools for reasoning about risk management's ROI issues, including guidelines for deciding how much of a risk reduction activity is enough. Tools that might be used here include risk-focused prototyping, specifying, testing, formal verification and validation, configuration management and quality assurance.

6.4 POTENTIAL BENEFITS WHEN PRACTISING RISK MANAGEMENT

The main goal of risk management and other SE practices is to improve the success rate of SDPs. Risk management helps project teams and management to focus on the essentials – the key concepts of risk management can help software managers assess problem situations and formulate proactive solutions [Boehm, 1997]. One of the main benefits achieved with proper risk management, is that an organisation can focus its efforts on avoiding future problems rather than solving current ones – therefore solving problems before they become critical. People can recognise and deal with potential problems daily, before they occur, and produce the finest product they can within budget and schedule constraints.

A well-defined and disciplined risk management process can also increase the level of communication both vertically and horizontally [Conrow, 1997]. We saw in figure 3.4, the risk management framework of the SEI, that communication forms the backbone of the risk management process. By having open communications between the project team, management and the users, problems can be identified earlier and dealt with before they can cause crises. There are usually fewer surprises in projects of which the risks are managed. Proper risk management achieves a free flow of information at and among all program levels, co-ordinated by a centralised system to capture the identified risks and information about how they are analysed, planned, tracked and controlled. Workgroups throughout the project understand that they are building just one end product and have a shared vision of a successful outcome [Williams, 1997].

The organisation can routinely apply the experience gained to avoid future crises rather than fixing blame. By comparing the lessons learnt, an organisation can evaluate activities in the work plans for their effect on the overall project risk, schedule and cost. The organisation can structure its important meeting agendas to discuss risks and their effects before discussing the specifics of technical approach and current status.

Project performance becomes more predictable. Gemmer [Gemmer, 1997] compared some projects' risk management evaluations with their Schedule Performance Index (ratio of work completed to work planned) and found that the projects managing risks were more likely to perform to schedule. The performance improved because projects can perform to expectations in riskier situations. It was also found that strategic planning benefits from risk management. Some senior managers remarked that risk management helps them to do better contingency planning. Senior management usually holds a risk reserve at organisational level. They use the RE of projects to determine how much risk reserve to hold and when this reserve can be allocated to new work [Gemmer, 1997].

Formal risk management created a pull for process and knowledge. It was found that the implementation of improved risk management practices took between four and eight months to reach an acceptable level or practice. When this stage was reached, it became a matter of maintaining that performance level. At this point, projects typically started pulling for improved risk management methods and improved strategies [Gemmer, 1997].

6.5 THE IMPORTANCE OF RISK MANAGEMENT

Although not much has been published in terms of implementation details of risk management done by organisations, the importance of risk management has been acknowledged by the software development industry.

According to the Project Management Body of Knowledge [PMBOK, 1998], risk management is currently the leading discipline in project management practice. The main reason for the software development industry trying to improve traditional project management, is that traditional project management has been found to solve problems reactively, whereas risk management enables project managers to attack problems proactively, thus preventing crises.

Risk management is especially important in mission critical systems, such as where human lives might be at stake. Many of the projects of the US DoD can be termed as such – therefore the US DoD enforces the management of risks in SDPs. The US DoD Directive 5000.2-R outlines the procedure mandatory for major projects and serves as a general model for all contractors. The following quote gives an indication of the importance associated with risk management – "Project managers and other acquisition managers shall continually assess program risks. Risks must be well understood, and risk management approaches developed, before decision authorities can authorise a program to proceed into the next phase of the acquisition process" [PMBOK, 1998]. The DoD has even gone so far as to specify in their directive 5000.2-R that contractors with superior risk management will have the advantage in procuring contracts.

The DoD designated formal risk management as a paramount practice under its Software Acquisition Best Practices Initiative. At the software level, the DoD's MIL-STD-498 directive states in section 5.19.1: "The developer shall perform risk management throughout the software development process. The developer shall identify, analyse, and prioritise the areas of the SDP that involve potential technical cost, or schedule risks; develop strategies for managing those risks; record the risks and strategies in the software development plan; and implement the strategies in accordance with the plan" [Conrow, 1997].

Although most of the software development industry agrees that risk management of SDPs is necessary, some contradictory beliefs exist too. Marvin Carr [Carr, 1997] is cautionary about risk management – according to his article "Counterpoint" in the IEEE Software Journal of May 1997, risk management can itself be risky, particularly if it is not done on an institutional basis. The point that he makes is that risk management, if practised incorrectly, can pose greater threat to SDPs than not practising risk management at all. However, Carr does acknowledge that risk management, when practised correctly on an institutional basis, can improve the success rate of SDPs.

By observing only these few opinions of leading software development organisations, it is clear that the software development industry has realised the importance of risk management.

6.6 CONCLUSION

In order to successfully implement risk management in software development, an organisation needs the following:

- A repeatable risk management process.
- Widespread access to adequate risk knowledge.
- The required functional behaviour by the people involved.

Risk management should be embraced by the entire organisation as opposed to the implementation of risk management only at project level. Executive management must realise their responsibility of establishing a risk-aware culture, making it known to all involved that risk is not something to ignore or hide, but to bring out into the open and to address.

In this chapter, we provided an overview of what is required to implement risk management. The success of the implementation is up to the organisation. However, implementing risk management is one project that cannot be allowed to fail!

7 CONCLUSION

7.1 RESEARCH OPPORTUNITIES

The field of risk management in software development is still fairly young, with a number of research opportunities that could add value to the effectiveness of risk management. In this dissertation, we strove to provide software development project managers with an overview of what they need to know about risk management, and how they can go about implementing it. Risk management in software development is still far from reaching a mature state where most questions have been answered. We now consider a number of issues that still need to be addressed.

7.1.1 Risk assessment

The risk assessment step of the risk management methodology provides for some interesting challenges. We evaluated a number of risk assessment questionnaires against Moynihan's evaluation framework (section 4.2.3, chapter 4) and from this noted that the questionnaires differed from each other in terms of the concepts they addressed or focussed on. Depending on the risk questionnaire used by an organisation to identify risks, different risks will therefore be identified. We need to ask the following questions with regard to the risk questionnaires used during the risk assessment step:

- Which risk questionnaire should an organisation use in the risk assessment effort?
- How can we evaluate existing risk questionnaires to establish whether they cover the risk areas we need to identify?
- What risk areas in software development need to be covered by a risk assessment questionnaire?

7.1.2 Risk analysis

During the risk analysis step, the risks identified in the assessment step are prioritised in terms of the urgency they need to be handled with. Risks are prioritised in terms of their risk exposure, namely the product of the probability of the risk occurring and the impact that the occurrence of the risk will have on the project. Questions that arise when we prioritise risks include:

- How can we attach an accurate value to the probability of a risk materialising? How accurate is the value that we attach to this probability?
- Should we use numerical or ordinal risk assessment scales when prioritising risks?
- Should we pay more attention to the probability of the risk materialising, or to the impact that the occurrence of the risk will have on the project when we determine the risk exposure of a risk?

7.1.3 Building up a risk information database

We discussed in this dissertation the fact that the risk handling capability of an organisation is directly related to the maturity of the software development processes used by the organisation (section 5.2, chapter 5). Central to the maturity of the software process is that the process used must be repeatable, i.e. that we reuse the process, learning from our mistakes each time we use it. The same principle applies to the risk management method we use – it needs to be a repeatable method and we need to be able to learn from past mistakes, preventing these mistakes from reoccurring. In order to utilise these experiences, we need to make the risk information available to project teams for reference purposes. Questions we may ask with regard to the risk information to store include:

- What information with regard to risks do we need to store?
- Who will be responsible for updating this database to ensure that the most current data is always available? Will this be the responsibility of one person, or will the responsibility be shared across project teams in the organisation?

- How will we categorise and reference this information? What type of searches will be supported?
- Will anyone have access to the information, or will we restrict all or portions of the data?
- What about the possibility of constructing an expert system that will learn from past experiences and make suggestions regarding how current risks should be handled?

7.2 OVERVIEW OF THIS RESEARCH

Every year, the software industry wastes an enormous amount of resources in time and money on software projects that eventually fail. Certain software best practices such as Software Engineering were established as an attempt to increase the success rate of software projects. However, even the implementation of these best practices (section 2.3, chapter 2) did not significantly improve the success rate of software development projects. It became evident that something else was needed to address the high rate of failure of software development projects.

In attempting to answer to the first question of our problem statement (chapter 1), namely

- Why do we need to manage the risks to SDPs?,

we came to the conclusion that the implementation of existing software best practices (chapter 2) has not improved the success rate of software development to the degree desired. These practices only cater for the tasks at hand, without taking into account that things may go wrong. By managing risks, we take into account that everything does not always go as planned, enabling us to plan proactively for possible deviancies from our initial plans. Managing risks enhance the chances of project success by allowing proactive planning as opposed to reactive crisis management. The software industry cannot continue to waste resources on projects that end in failure – we have to do everything in our power to ensure project success, and that requires managing the risks to projects.

After establishing the necessity of managing risks, the following question arose:

- What do project managers need to know about risk management before they can start managing the risks in their SDPs?

It is imperative that a project manager, as well as the rest of the project team, realises the importance of managing the risks in software development. In order to realise this importance, the project manager needs to understand the shortcomings of the software best practices and traditional development life cycle models such as the waterfall model. The biggest problem with the waterfall model is that it represents a linear life cycle, with the risk being the highest later in the SDLC as illustrated in figure 3.1 (chapter 3). The spiral model was the first model to acknowledge the presence of risk in software development. From this model, other risk-driven methodologies such as the Rational Unified Process came into existence, addressing the risks in software development. The Rational Unified Process is a software development methodology that suggests an iterative development life cycle, with each iteration resulting in an executable release. The iterations are prioritised in terms of the risks associated with them; the iterations with the highest risk will be implemented first to reduce the risk as early as possible in the development life cycle.

The main difference between how project managers managed projects traditionally, and managing risks, is that project managers now need to take into account those things that may not go as planned. Traditionally, project managers only planned for the tasks necessary to finish a project, without taking into account that something may go wrong. The focus needs to shift from what needs to be done, to what may go wrong.

Project managers also need to be familiar with the concepts of risk management, such as risk, risk exposure, and risk handling capability. A risk is anything that may cause a project to fail, where the risk exposure indicates the seriousness of the risk in terms of the product of the probability and the impact of the risk. The risk handling capability of an organisation or project team is an indication of the risk exposure that the organisation is capable of handling. Projects with a higher risk exposure than the risk handling capability of the organisation should not be attempted, since these projects are bound to end in failure.

The process of risk management includes four steps, namely risk assessment, risk analysis, risk handling and experience gained. Risk assessment is the process of identifying all the risks that may possibly impede on the success of a project. Risk analysis takes the output of the assessment phase, namely the identified risks, and analyses these risks to determine the priority that should be attached to the specific risk. During the risk-handling phase, those risks with the highest priorities are attended to. There are different ways to handle risks, such as avoiding risks, mitigating risks, transferring risks, and so on. During this phase, it is decided how the risks with the highest priorities should be handled. The specific plan of action is then executed and the risks is once again reviewed to determine whether the risk-handling strategy was successful and whether further action and possibly a different risk-handling strategy is required.

Central to the whole risk management process is the decisions that governs which risks are handled, the priorities attached to the risks, how the risks are handled, and so on. The third question in our problem statement addresses the issue of decision making:

- How do we make decisions about risks?

In some very rare instances, we may be aware of the outcome of a certain decision. However, in practice this is rarely the case. Decisions have to be made under uncertain conditions with the outcome also being uncertain. To make the best possible decision under such circumstances, we need to consult all parties that may provide valuable insights and inputs into the circumstances governing the decision. It is important that the people making the decision agree on the final decision – i.e. it is important that consensus is reached on the final decisions. Equally important is that the whole decision-making process be documented. The rationale behind a certain decision should be documented so that it is always possible to determine why a certain decision was reached.

Finally, we asked the question:

- How do we go about implementing risk management in software development?

Risk management is not going to become part of the software development process without effort. Executive management of the organisation will have to see to the establishment of a risk-aware culture, moving away from a risk-averse culture that will cause problems to the implementation of risk management. The implementation of risk management should be handled as a specific task and an infrastructure conducive to risk management should be set in place. Project teams also need to be trained and guided in managing risks, and risk information should be documented so that people can start to benefit from the experiences of others.

With this dissertation, we provided an overview of risk management, including why it is necessary and how risks should be managed. We explained the risk management process and highlighted problems that could possibly be experienced when first implementing risk management. This dissertation provides project managers with the information they need to be aware of, before implementing risk management, as well as with guidelines for implementing risk management.

The software industry needs to realise that risks in software development need to be managed to increase the success rate of software development projects. As Tom Gilb [Gilb, 1988] so aptly put it:

"If you do not actively attack the risks in your project, they will actively attack you."

REFERENCES

[Action Design, 1995] ACTION DESIGN, *Organisational Learning In Action,* (Training Course) Action Design, Amherst, Massachusetts, 1995.

[Barki, 1993] H. BARKI, S. RIVARD, J. TALBOT, *"Toward an Assessment of Software Development Risk" in Management Information Systems,* Vol. 10, No. 2, 1993, pp 203-225.

[Bloom, 1956] B. BLOOM, *A Taxonomy of Educational Objectives: Cognitive Domain: Handbook 1,* David McKay, New York, 1956.

[Boehm, 1988] B.W. BOEHM, *"A Spiral Model of Software Development and Enhancement" in Computer,* May 1988.

[Boehm, 1989] B.W. BOEHM, *IEEE Tutorial on Software Risk Management*, IEEE Computer Society Press, Los Alamitos, California, 1989.

[Boehm, 1991] B.W. BOEHM, *"Software Risk Management: Principles and Practice" in IEEE Software,* Vol. 8, No. 1, 1991.

[Boehm, 1997] B.W. BOEHM, T. DEMARCO, *"Software Risk Management" in IEEE Software,* May 1997.

[Brooks, 1986] F.P. BROOKS, JR., *"No Silver bullet" in Information Processing 1986,* H.-J. Kugler (Editor), Elsevier North-Holland, New York, 1986.

[Brown, 1992] A.W. BROWN, A.N. EARL, J.A. McDERMID, *Software Engineering Environments – Automated Support for Software Engineering,* McGraw-Hill, Berkshire, England, 1992.

[Carr, 1993] M.J. CARR et al, *"Taxonomy-Based Risk Identification",* Technical Report CMU/SEI-93-TR-006, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1993.

[Carr, 1997] M.J. CARR, *"Risk Management May not be for Everyone" in IEEE Software,* May 1997.

[Charette, 1989] R.N. CHARETTE, *Software Engineering Risk Analysis and Management*, McGraw-Hill, New York, 1989.

[Charette, 1990] R.N. CHARETTE, *Application Strategies for Risk Management,* McGraw-Hill, New York, 1990.

[Charette, 1991] R.N. CHARETTE, *"The Risks with Risk Analysis" in Communications of the ACM,* Vol. 34, No. 6, 1991.

[Charette, 1995] R.N. CHARETTE, *"On Becoming a Risk Entrepreneur" in American Programmer,* March 1995.

[Charette, 1997] R.N. CHARETTE, *"Large-Scale Project Management Is Risk Management" in IEEE Software,* May 1997.

[Conrow, 1997] E.H. CONROW, P.S. SHISHIDO, *"Implementing Risk Management on Software Intensive projects" in IEEE Software,* May 1997.

[Covey, 1989] S.COVEY, *The 7 habits of Highly Effective People: Restoring the Character Ethic,* Simon & Schuster, New York, 1989.

[Covey, 1990] S.COVEY, *The 7 habits of Highly Effective People: Powerful Lessons in Personal Change,* Simon & Schuster, New York, 1990.

[Dorofee, 1996] A.J. DOROFEE et al, *"Continuous Risk Management Guidebook",* Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1996.

[Fairly, 1994] R. FAIRLY, *"Risk Management for Software Projects" in IEEE Software,* Vol. 11, No. 3, 1994.

[Garvey, 1997] P.R. GARVEY, D.J. PHAIR, J.A. WILSON, *"An information architecture for Risk Assessment and Management" in IEEE Software,* May 1997.

[Gilb, 1988] T. GILB, *Principles of Software Engineering Management,* Addison-Wesley, 1988.

[Higuera, 1994] R.P. HIGUERA et al, *"Team Risk Management: A new model for Customer-Supplier Relationships",* Technical Report CMU/SEI-94-SR-5, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1994.

[Humphrey, 1994] W.S. HUMPHREY, *Managing the Software Process,* Addison-Wesley, Reading, MA, 1989.

[Jacobson, 1997] I. JACOBSON, M. GRISS, P. JONSSON, *Software Reuse – Architecture, Process and Organization for Business Success,* Addison-Wesley, 1997.

[Jones, 1995] C. JONES, *"Risks of Software System Failure or Disaster" in American Programmer,* March 1995.

[Kerzner, 1995] H. KERZNER, *Project Management – A Systems Approach to Planning, Scheduling, and Controlling,* Fifth Edition, Van Nostrand Reinhold, New York, 1995.

[Kruchten, 1998] P. KRUCHTEN, *The Rational Unified Process – An Introduction,* Addison-Wesley, 1998.

[Lister, 1997] T. LISTER, *"Risk Management is Project Management for Adults" in IEEE Software,* May 1997.

[Moynihan, 1997] T. MOYNIHAN, *"How Experienced Project Managers Assess Risk" in IEEE Software,* May 1997.

[Paulk, 1996] M.C. PAULK, *"Capability Maturity Model for Software Version 1.1",* Technical Report CMU/SEI-93-TR-24, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1996.

[PMBOK, 1998] http://www.pmforum.org/prof/specint2.htm *"Internet Directory of Project Management Resources",* Project Management Body of Knowledge (PMBOK), 1998.

[Quatrani, 1998] T. QUATRANI, *Visual Modeling with Rational Rose and UML,* Addison-Wesley, Massachusetts, 1998.

[RATL, 1999] http://www.rational.com/products/rs/index.jtmpl, *"Rational Suite Product Family",* Rational Software Corporation (Nasdaq: RATL), 1999.

[Rothfeder, 1988] J. ROTHFEDER, *"It's Late, Costly, and Incompetent – But Try Firing a Computer System" in Business Week,* November 7, 1988.

[Schach, 1993] S.R. SCHACH, *Software Engineering,* Second Edition, Aksen Associates, Boston, 1993.

[Sisti, 1994] F.J. SISTI, J. SUJOE, *"Software Risk Evaluation Version 1.0"*, Technical Report CMU/SEI-94-TR-19, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1994.

[SPMN, 1999] http://www.spmn.com *"Software Program Manager's Network best practices"*, Software Program Managers Network (SPMN), 1999.

[Standish, 1994] THE STANDISH GROUP INTERNATIONAL, *Charting the Seas of Information Technology,* Dennis, Mass., 1994.

[Standish, 1999] http://www.standishgroup.com/chaos.html *"Chaos"*, The Standish Group International, 1994.

[Transnet, 1996] TRANSNET NIP IT FORUM, *"Transnet Information Technology And Systems National Infrastructure Plan Investment Guidelines",* August 1996.

[TRW, 1999] http://www.trw.com/trw_profile/trw/ *"TRW Company Profile"*, TRW Inc, 1999.

[Williams, 1997] R.C. WILLIAMS, J.A. WALKER, A.J. DOROFEE, *"Putting Risk Management into Practice" in IEEE Software,* May 1997.

[Yourdon, 1995] E. YOURDON, *"Introduction to the March 1995 issue" in American Programmer,* March 1995.