Risk Management

The proactive management of risks throughout the software development lifecycle is important for project success. In this chapter, we will explain the following:

- the risk management practice, which involves risk identification, analysis, prioritization, planning, mitigation, monitoring, and communication
- software development risks that seem to reoccur in educational and industrial projects
- a risk-driven process for selecting a software development model

> *Risk in itself is not bad; risk is essential to progress, and failure is often a key part of learning. But we must learn to balance the possible negative consequences of risk against the potential benefits of its associated opportunity.* (Van Scoy, 1992)

A *risk* is a *potential future harm that may arise from some present action* (Wikipedia, 2004), such as, a schedule slip or a cost overrun. The loss is often considered in terms of direct financial loss, but also can be a loss in terms of credibility, future business, and loss of property or life.

This chapter is about doing proactive planning for your software projects via risk management. *Risk management* is a *series of steps whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful software operation or a major source of expensive rework*. (Boehm, 1989) The software industry is fraught with failed and delayed projects, most of which far exceed their original budget. The Standish Group reported that only 28 percent of software projects are completed on time and on budget. Over 23 percent of software projects are cancelled before they ever get completed, and 49 percent of projects cost 145 percent of their original estimates. (Standish, 1995) In hindsight, many of these companies indicated that their problems could have been avoided or strongly reduced if there had been an explicit early warning of the high-risk elements of the project. Many projects fail either because simple problems were reported too late or because the wrong problem was addressed. (Bruegge and Dutoit, 2000)

Problems happen. Teams can choose to be reactive or proactive about these problems. *Reactive* teams fly into action to correct the problem rapidly in a crisis-driven, fire-fighting mode. Without proper planning, problems often occur late in the schedule. At this point, resolving any serious problems can require extensive modification, leading to big delays. *Proactive* teams begin thinking about risks even before technical work is initiated. Their objective is to be able to avoid risk whenever possible, to solve problems before they manifest themselves and to respond to problems that do happen in a controlled and effective manner. This chapter is about being proactive.

1 The Risk Management Practice

The risk management process can be broken down into two interrelated phases, risk assessment and risk control, as outlined in Figure 1. These phases are further broken

down. Risk assessment involves risk identification, risk analysis, and risk prioritization. Risk control involves risk planning, risk mitigation, and risk monitoring.(Boehm, 1989) Each of these will be discussed in this section. It is essential that risk management be done iteratively, throughout the project, as a part of the team's project management routine.

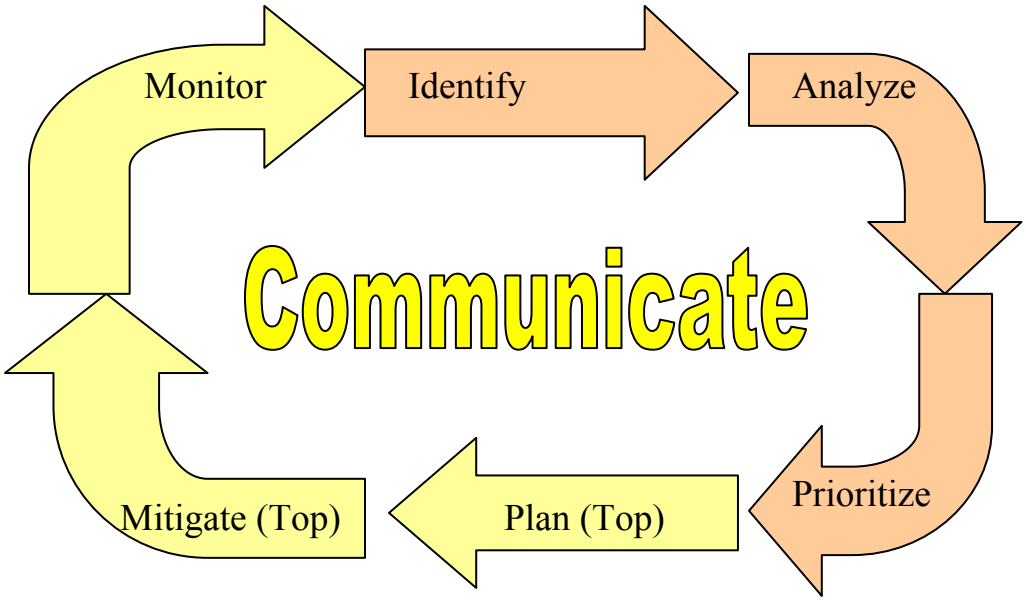

Figure 1: The Risk Management Cycle.

1.1 Risk Identification

In the risk identification step, the team systematically enumerates as many project risks as possible to make them explicit before they become problems. There are several ways to look at the kinds of software project risks, as shown in Table 1. It is helpful to understand the different types of risk so that a team can explore the possibilities of each of them. Each of these types of risk is described below.

Table 1: General Categories of Risk

Generic Risks		**Product-Specific Risks**
Project Risks	**Product Risks**	**Business Risks**
Factors to consider: People, size, process, technology, tools, organizational, managerial, customer, estimation, sales, support		

Generic risks are potential threats to *every* software project. Some examples of generic risks are changing requirements, losing key personnel, or bankruptcy of the software company or of the customer. It is advisable for a development organization to keep a checklist of these types of risks. Teams can then assess the extent to which these risks are a factor for their project based upon the known set of programmers, managers, customers,

and policies. *Product-specific risks* can be distinguished from generic risks because they can only be identified by those with a clear understanding of the technology, the people, and the environment of the specific product. An example of a product-specific risk is the availability of a complex network necessary for testing.

Generic and product-specific risks can be further divided into project, product, and business risks. *Project risks* are those that affect the project schedule or the resources (personnel or budgets) dedicated to the project. *Product risks* are those that affect the quality or performance of the software being developed. Finally, *business risks* are those that threaten the viability of the software, such as building an excellent product no one wants or building a product that no longer fits into the overall business strategy of the company.

There are some specific factors to consider when examining project, product, and business risks. Some examples of these factors are listed here, although this list is meant to stimulate your thinking rather than to be an all-inclusive list.

- *People risks* are associated with the availability, skill level, and retention of the people on the development team.
- *Size risks* are associated with the magnitude of the product and the product team. Larger products are generally more complex with more interactions. Larger teams are harder to coordinate.
- *Process risks* are related to whether the team uses a defined, appropriate software development process and to whether the team members actually follow the process.
- *Technology risks* are derived from the software or hardware technologies that are being used as part of the system being developed. Using new or emerging or complex technology increases the overall risk.
- *Tools risks*, similar to technology risks, relate to the use, availability, and reliability of support software used by the development team, such as development environments and other Computer-Aided Software Engineering (CASE) tools.
- *Organizational and managerial risks* are derived from the environment where the software is being developed. Some examples are the financial stability of the company and threats of company reorganization and the potential of the resultant loss of support by management due to a change in focus or a change in people.
- *Customer risks* are derived from changes to the customer requirements, customers' lack of understanding of the impact of these changes, the process of managing these requirements changes, and the ability of the customer to communicate effectively with the team and to accurately convey the attributes of the desired product.
- *Estimation risks* are derived from inaccuracies in estimating the resources and the time required to build the product properly.
- *Sales and support risks* involve the chances that the team builds a product that the sales force does not understand how to sell or that is difficult to correct, adapt, or enhance.

Spontaneous and sporadic risk identification is usually not sufficient. There are various risk elicitation techniques the team can use to systematically and proactively surface risks:

- *Meeting*. The team, including the development team and the marketing and customer representatives if possible, gathers together. The group brainstorms; each participant spontaneously contributes as many risks as they can possibly think of.
- *Checklists/Taxonomy*. The risk elicitors are aided in their risk identification by the use of checklists and/or taxonomies (in other words, a defined, orderly classification of potential risks) that focuses on some subset of known and predictable risks. Checklists and taxonomies based upon past projects are especially beneficial. These artifacts should be used to interview project participants, such as the client, the developers, and the manager.
- *Comparison with past projects*. The risk elicitors examine the risk management artifacts of previous projects. They consider whether these same risks are present in the new project.
- *Decomposition*. Large, unwieldy, unmanageable risks that are identified are further broken down into small risks that are more likely to be managed. Additionally, by decomposing the development process into small pieces, you may be able to identify other potential problems.

Project participants can be reluctant to communicate potential failures or shortcomings and can be too optimistic about the future. It is essential that all participants are encouraged to report risks so they can be monitored and managed. Participants should be rewarded for identifying risks and problems as early as possible.

It is recommended that risks should be stated using the condition-transition-consequence (CTC) format (Gluch, 1994):

Given that <condition> then there is a concern that (possibly) <transition> <consequence>.

- Condition is a description of the current conditions prompting concern.
- Transition is the part that involves change (time).
- Consequence is a description of the potential outcome.

For example, *given that no one in our team has ever developed a product in Prolog, then there is a concern that (possibly) the project will take two months longer than has been estimated*.

1.2 Analyze

After risks have been identified and enumerated, the next step is risk analysis. Through *risk analysis,* we transform the risks that were identified into decision-making information. In turn, each risk is considered and a judgment made about the probability and the seriousness of the risk. For each risk, the team must do the following:

- Assess the *probability* of a loss occurring. Some risks are very likely to occur. Others are very unlikely. Establish and utilize a scale that reflects the perceived likelihood of a risk. Depending upon the degree of detail desired and/or possible, the scale can be numeric, based on a percentage scale, such as “10 percent likely to lose a key team

member" or based on categories, such as: very improbable, improbable, probable, or frequent. In the case that a categorical assignment is used, the team should establish a set numerical probability for each qualitative value (e.g. very improbable= 10 percent, improbable = 25 percent).

- Assess the *impact* of the loss if the loss were to occur. Delineate the consequences of the risk, and estimate the impact of the risk on the project and the product. Similar to the probability discussion above, the team can choose to assign numerical monetary values to the magnitude of loss, such as $10,000 for a two-week delay in schedule. Alternately, categories may be used and assigned values, such as 1=negligible, 2=marginal, 3=critical, or 4=catastrophic.

Determining the probability and the magnitude of the risk can be difficult and can seem to be arbitrarily chosen. One means of determining the risk probability is for each team member to estimate each of these values individually. Then, the input of individual team members is collected in a round robin fashion and reported to the group. Sometimes the collection and reporting is done anonymously. Team members debate the logic behind the submitted estimates. The individuals then re-estimate and iterate on the estimate until assessment of risk probability and impact begins to converge. This means of converging on the probability and estimate is called the Delphi Technique (Gupta and Clarke, 1996). The Delphi Technique is a group consensus method that is often used when the factors under consideration are subjective.

The analyzed risks are organized into a risk table. The template for a risk table is shown in Table 2. In Sections 2 and 3, we show you some completed sample risk tables. The information that is to be provided in each of the columns is now explained.

- *Rank* will be discussed in section 1.3.
- *Risk* is the description of the risk itself, preferably stated in CTC format.
- *Probability* is the likelihood of the risk occurring, using either a numeric or categorical scale, as discussed in the last section.
- *Impact* is the magnitude of the loss if the risk were to occur, using either a numeric or a categorical scale.
- *Rank last week* and the number of *weeks on list* are documented so the team can monitor changes in priority, to determine if actions are being taken that cause changes in the stature of the risk.
- *Action* documents what the team is doing to manage the risk, as will be discussed in sections 1.4-1.5. The action field is often not completed until the risks have been prioritized, as will be discussed in the next section.

Table 2: Risk Table Template

Rank	Risk	Probability	Impact	Rank Last Week/ Weeks on list	Action

1.3 Prioritize

After the risks have been organized into a risk table, such as Table 4.2, the team prioritizes the risks by ranking them. It is too costly and perhaps even unnecessary to take action on *every* identified risk. Some of them have a very low impact or a very low probability of occurring – or both. Through the prioritization process, the team determines which risks it will take action on.

The team sorts the list so that the high probability, high impact risks percolate to the top of the table and the low-probability, low impact risks drop to the bottom. If the team used categorical values for probability (e.g. very improbable, improbable, probable, or frequent) and/or impact (e.g. negligible, marginal, critical, or catastrophic), group consensus techniques may need to be used to produce the risk ranking. We will show you an example of this type of ranking in Section 2.

If numerical values were given for probability (percentage) and impact (monetary), the risk exposure can be calculated. Risk exposure is calculated as follows (Boehm, 1989):

$$\textit{Risk Exposure (RE)} = P \times C$$

where P = probability of occurrence for a risk and C is the impact of the loss to the product should the risk occur. For example, if the probability of a risk is 10 percent and the impact of the risk is \$10,000, the risk exposure = (0.1)(\$10,000) = \$1,000. If RE is calculated for each risk, the prioritization is based upon a numerical ranking of the risk exposures. We will show you an example of this type of ranking in Section 3.

After the risks are prioritized, the team, led by the project manager, defines a cut off line so that only the risks above the line are given further attention. The activities of this "further attention" are to plan, mitigate, monitor, and communicate – as is discussed in the following sections. The lower ranked risks stay on the table for the time being with no action other than monitoring.

1.4 Plan

Risk management plans should be developed for each of the "above the line" prioritized risks so that proactive action can take place. These actions are documented in the Action column of the Risk Table (Table 2). Following are some examples of the kinds of risk planning actions that can take place:

- *Information buying*. Perceived risk can be reduced by obtaining more information through investigation. For example, in a project in which the use of a new technology has created risk, the team can invest some money to learn about the technology. Throw-away prototypes can be developed using the new technology to educate some of the staff on the new technology and to assess the fit of the new technology for the product.

- *Contingency plans*. A contingency plan is a plan that describes what to do if certain risks materialize. By planning ahead with such a plan, you are prepared and have a strategy in place do deal with the issue.
- *Risk reduction*. For example, if the team is concerned that the use of a new programming language may cause a schedule delay, the budget might contain a line item entitled "potential schedule" to cover a potential schedule slip. Because the budget already covers the potential slip, the financial risk to the organization is reduced. Alternately, the team can plan to employ inspections to reduce the risk of quality problems.
- *Risk acceptance*. Sometimes the organization consciously chooses to live with the consequences of the risk (Hall, 1998) and the results of the potential loss. In this case, no action is planned.

1.5 Mitigate

Related to risk planning, through risk mitigation, the team develops strategies to reduce the possibility or the loss impact of a risk. Risk mitigation produces a situation in which the risk items are eliminated or otherwise resolved. These actions are documented in the Action column of the Risk Table (Table 2). Some examples of risk mitigation strategies follow:

- *Risk avoidance*. When a lose-lose strategy is likely (Hall, 1998)[1], the team can opt to eliminate the risk An example of a risk avoidance strategy is the team opting not to develop a product or a particularly risky feature.
- *Risk protection*. The organization can buy insurance to cover any financial loss should the risk become a reality. Alternately, a team can employ fault-tolerance strategies, such as parallel processors, to provide reliability insurance.

Risk planning and risk mitigation actions often come with an associated cost. The team must do a cost/benefit analysis to decide whether the benefits accrued by the risk management steps outweigh the costs associated with implementing them. This calculation can involve the calculation of risk leverage (Pfleeger, 1998).

Risk Leverage =
(risk exposure before reduction – risk exposure after reduction)/cost of risk reduction

If risk leverage value, *rl*, is ≤ 1, clearly the benefit of applying risk reduction is not worth its cost. If *rl* is only slightly > 1, still the benefit is very questionable, because these computations are based on probabilistic estimates and not on actual data. Therefore, *rl* is usually multiplied by a *risk discount factor* $\rho < 1$. If $\rho\ rl > 1$, then the benefit of applying risk reduction is considered worth its cost. If the discounted leveraged valued is not high enough to justify the action, the team should look for other, less costly or more effective, reduction techniques.

[1] In the lose-lose strategy, everyone gives something up, in the sense that neither side gets what they want, but everyone can live with the decision.

1.6 Monitor

After risks are identified, analyzed, and prioritized, and actions are established, it is essential that the team regularly monitor the progress of the product and the resolution of the risk items, taking corrective action when necessary. This monitoring can be done as part of the team project management activities or via explicit risk management activities. Often teams regularly monitor their "Top 10 risks."

Risks need to be revisited at regular intervals for the team to reevaluate each risk to determine when new circumstances caused its probability and/or impact to change. At each interval, some risks may be added to the list and others taken away. Risks need to be reprioritized to see which are moved "above the line" and need to have action plans and which move "below the line" and no longer need action plans. A key to successful risk management is that proactive actions are owned by individuals and are monitored. (Larman, 2004)

As time passes and more is learned about the project, the information gained over time may alter the risk profile considerably. Additionally, time may make it possible to refine the risk into a set of more detailed risks. These refined risks may be easier to mitigate, monitor, and manage.

1.7 Communicate

On-going and effective communication between management, the development team, marketing, and customer representatives about project risks is essential for effective risk management. This communication enables the sharing of all information and is the cornerstone of effective risk management.

1.8 The Stakeholders of Risk Management

The three stakeholders are involved in risk management.

- The developer must systematically and continually enumerate all the possible risks related to technical capability and making the schedule.
- The manager must lead the team to follow the risk management process to proactively manage the project risks. The manager must also allocate resources for proactive risks management.
- The customer must participate in the continual identification of risks.

None of these stakeholders is empowered to manage business risks, i.e. what we called organizational and managerial risks, and sales and support risks in the "Risk Identification " section above. This kind of risk must be managed by upper management and marketing department of the firm.

2 Risk Management in Educational Projects

Sometimes the need for risk management can seem far off for students. After all, you don't do anything close to buying insurance to reduce the risk for your class projects! However, consider that your success (your grade) in the class is at risk. In beginning computer science classes, your assignments were probably small, the requirements of

these assignments crisp and defined, and you worked alone. Your chances of being successful were well within your own control. As you advance in your academic career, course projects will likely become quite a bit longer, you will be working with at least one other person, and the requirements will be more ambiguous and even changeable. All of a sudden, things aren't nearly as under control. What can you do to improve your odds of getting a good grade? Employing risk management can help.

Table 3 shows the ranked "Top 10" risk items based upon the frequency with which they were identified during the six weeks of risk management by 24 student teams in an undergraduate software engineering class. The students worked in teams of four or five students on a project that lasted seven weeks. All project teams completed the same project. A graduate student performed the role of customer for the students. You should consider whether your own projects could encounter these same risks.

Table 3: Student Top 10 Risk Items

Risk Item	Risk Management Technique
Overriding other people's work, not having the latest versions of code	Use a configuration management tool effectively.
Lack of exposure to and/or experience with technologies	Take time to learn tools and technologies, seek help from teaching staff.
Being overwhelmed by work in other classes	Have a project management plan with deadlines and ownership, update the project management plan frequently.
Common meeting times	In the beginning of the project, determine all possible common times to meet based on class schedules and other commitments.
Requirements understanding	Meet with, e-mail, or phone customer.
Lack of communication	Set up a group Web page, group e-mail accounts, trade instant messaging IDs, meet regularly.
Project organization	Assign each team member a role, break down work in project management plan.
Loss of a team member	Assure files are uploaded and integrated consistently, use knowledge management strategies such as pair programming to understand each other's work.
Difficulty integrating work	Increase communication, integrate often.
Planning taking up too much time, not enough time to work on product	Don't get more detailed than necessary with the planning.

A sample student team risk management table from the class described above is shown below in Table 4; the team is in the fifth week of the project. Both the probability and the impact use categorical values, which is typical of a student project. Because of this, the student teams must use a group consensus technique to rank their risks. The method of

using categories for risk analysis and group consensus for risk prioritization is also used in industry.

Table 4: Sample Student Risk Table

Rank	Risk	Probability	Impact	Rank Last Week/ Weeks on list	Action
1	None of us knows how to use the technology.	frequent	critical	1/5	Read. Do tutorials.
2	Integration problems.	frequent	critical	2/5	Integrate all work Sunday nights.
3	Someone drops the class.	improb	critical	4/5	Pair programming for all work.
4	Team members missing important team meetings.	improb.	marginal	5/4	Person who misses meeting has to supply Sunday night pizza the next week.
5	Overriding each other's work	improb	marginal	3/5	Continue using CVS.

3 Risk Management in Industrial Projects

Industrial projects have many different types of risks than you would experience as a student. Some of the risks, such as changing requirements and losing team members are similar. Boehm developed a top 10 risk item for industrial projects by surveying several experienced managers. This list is shown below in Table 5.

Table 5: Industry Top 10 Software Risk Items, adapted from (Boehm, 1989; Boehm, January 1991)

Risk Item	Risk Management Technique
Personnel shortfall	Staffing with top talent, job matching, team building, key personnel agreements, cross training
Unrealistic schedules and budgets	Detailed milestone cost and schedule estimation, design to cost, incremental development, software reuse, requirements scrubbing
Developing the wrong functions and properties	Organizational analysis, mission analysis, operations-concept formulation, user surveys and user participation, prototyping, early users' manuals
Developing the wrong user interface	Prototyping, scenarios, task analysis, user participation
Gold-plating (e.g. implementing "neat features" not asked for by customer)	Requirements scrubbing, prototyping, cost-benefit analysis, designing to cost
Continuing stream of requirements changes	High change threshold information hiding, incremental development (deferring changes to later increments)
Shortfalls in externally-furnished components (e.g. component reuse)	Benchmarking, inspections, reference checking, compatibility analysis
Shortfalls in externally performed tasks (e.g. worked performed by a contractor)	Reference checking, pre-award audits, award-fee contracts, competitive design or prototyping, team building
Real-time performance shortfalls	Simulation, benchmarking, modeling, prototyping, instrumentation, tuning
Straining computer science capabilities	Technical analysis, cost-benefit analysis, prototyping, reference checking

Table 6 shows a sample risk table for an industrial team. The kinds of risk that rise to the top are different than in the student risk table. Additionally, while the student example used categories for probability and impact, the industrial team uses their best estimate of numerical probability and impact. As discussed earlier, using these numerical values, the risk exposure can be calculated (risk exposure = probability * impact). Risk exposure can then be used for ranking the risks.

Table 6 Sample Industrial Risk Table

Rank	Risk	Prob.	Impact	Risk Exp.	Rank Last Week/ Weeks on list	Action
1	Delay by Raleigh team to deliver toolkit	50%	$10,000	$5,000	3/10	Weekly status meeting, Possibility of interim releases.
2	Requirements changes	40%	$7,000	$2,800	1/12	Bi-weekly deliverables.
3	Aggressive performance requirements	30%	$9,000	$2,700	4/5	Prototyping, performance testing.
4	Lose team member	5%	$50,000	$2,500	8/12	Pair programming.
5	Unsure of desired graphical user interface	5%	$1,000	$50	6/12	Design with the Model-View-Controller pattern.

It can be difficult, even for an industrial team, to estimate numerical values for probability and loss. To overcome this, you can assess these two values on a relative scale of 0 to 10 rather than trying to estimate numerical values.

4 Risk Management for Software Development Model Selection

(with credit to Barry Boehm and Richard Turner)

One large and potentially risky decision for a software development team is the selection of the software development methodology and associated practices. We have introduced the plan-driven software development model and the agile software development model. Depending upon the type of project and team, one of these models or a hybrid of the two is best. This section of the chapter is very important for you to understand. As you proceed through the rest of the book, you will be presented with alternatives for many development practices (such as plan-driven requirements, agile requirements, plan-driven design, and agile design). It is important for you to understand that you need to choose the alternative that is appropriate for the project you are working on.

In this section, we explain a risk-driven approach to making the selection between an agile, a plan-driven, or a hybrid software development model. The five-step method was developed by Barry Boehm and Richard Turner (Boehm and Turner, 2003; Boehm and Turner, June 2003). Boehm and Turner developed the method so that software developers can enjoy the benefits of both agile and plan-driven methods, while mitigating many of their drawbacks. The guidance given by their method is important because every development practice has its situation-dependent shortcomings and its home ground (the situations for which each is best suited). Agile methodologies promise increased customer satisfaction, lower defect rates, faster development times, and a solution to

rapidly changing requirements. Agile methods are highly iterative in nature – meaning that partial working product is delivered to customers often. Iteration is a prudent risk mitigation strategy because the partial deliverables uncover risks while there is still time to alleviate them. Plan-driven approaches promise predictability, stability, and high assurance. It's all about picking the right model for the job depending upon the most important consideration of the project.

4.1 Personal Characteristics of Team

Some background is necessary before describing Boehm and Turner's method. To start, Boehm and Turner believe the personal characteristics of the people who make up the software development team are a key factor in determining whether to use an agile or plan-driven approach. Think about it. A team made up of very experienced team members is very different from a team that consists of all new people to the technology and the domain. The technology is the programming language, hardware platform, and so forth. The domain is the subject area of the program (for example, medical software or networking software). To classify individual skill level, Boehm and Turner adopted and then adapted the classification scheme of Alistair Cockburn (Cockburn, 2001), as shown in Table 7. In the table, the term *method* refers to a single (or set of) software development practice (such as eliciting requirements or automating tests).

Table 7: Levels of Software Method Understanding and Use (adapted from (Boehm and Turner, June 2003))

Level	Characteristics	Applicability
3	Able to revise a method, breaking its rules to fit an unprecedented new situation.	Can function well on any team.
2	Able to tailor a method to fit a precedented situation. With training, some can become Level 3.	Can function well in managing a small, precedented agile or plan-driven project but need the guidance of level 3s in unprecedented situations.
1A	With training, able to perform discretionary method steps such as providing resource estimates to decide which requirements should be included each release. With experience, can become Level 2.	Can function well on both agile and plan-driven teams that have enough Level 2 people to guide them.
1B	With training, able to perform procedural method steps such as coding a simple program, following coding standards, or running tests. With experience can master some Level 1A skills.	Function well in performing straightforward development in a stable situation. Would likely slow an agile team, particularly if a large percentage of the team was made up of 1B people.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.	Transfer to other work.

It is important to consider both technology and domain expertise when considering a person's skill level. A Level 3 expert in an object-oriented language such as Java developing software for the retail industry might temporarily revert back to being a Level 1B if moved to an assignment like developing a compiler in a functional language such as Haskell. This person's prior expertise enables him to fairly rapidly advance through the skill levels, most likely to the old Level 3. However, it is important to consider the person's current (not potential) skill level when considering the make up of the team relative to agile and plan-driven methods.

4.2 Agile and Plan-Driven Home Grounds

Boehm and Turner have observed projects succeed that have used purely an agile approach, they have observed projects succeed with purely plan-driven methods, and they have observed projects succeed with hybrid methods. Based on these experiences, they share the project characteristics of agile "home grounds" and plan-driven "home grounds" where home ground is defined as the situation for which each is best suited. These home grounds are summarized in Table 8.

Table 8: Agile and Plan-driven Home Grounds (adapted from (Boehm and Turner, June 2003))

Project Characteristics	Agile Home Ground	Plan-Driven Home Ground
Application		
Primary goals	Rapid value, responding to change.	Predictability, stability, high assurance.
Size	Smaller teams and projects.	Larger teams and projects.
Environment	Turbulent, high change, project focused.	Stable, low change, project and organization focused.
Management		
Customer relations	Dedicated on-site customer, focused on prioritized product releases (increments).	As-needed customer interactions, focused on fulfilling a contract.
Planning and control	Team has an understanding of plans and monitors to this plan.	Documented plans and explicit monitoring to plans.
Communications	Passed from person to person (tacit, interpersonal).	Knowledge documented in team artifacts (explicit).
Technical		
Requirements	Prioritized, informal stories and test cases. Requirements are likely to change in unpredictable ways.	Formalized requirements. Requirements may change in predictable ways.
Development	Simple design, short increments	Extensive design, longer increments.
Test	Automated, executable test cases are used to further define the specifics of the requirements.	Documented test plans and procedures.
Personnel		
Customers	Dedicated, co-located CRACK* performers.	CRACK performers, not always co-located.
Developers (See Section 4.1)	At least 30% Level 2 and 3 experts; no level 1B or Level -1 personnel.	50% Level 3s early; 10% throughout; 30% Level 1B's workable; no Level -1s.
Culture	Team enjoys being empowered and having freedom (thriving on chaos).	Team is empowered via freedom embodied in policies and procedures (thriving on order).

* CRACK = Collaborative, Representative, Authorized, Committed, and Knowledgeable

4.3 Critical Factors and the Polar Chart

The analysis of the home grounds in Table 8 and the general characteristics of agile and plan-driven methods led Boehm and Turner to define five critical factors that can be used to describe a project environment and can be used to help determine the appropriate balance between agile and plan-driven methods. These five factors, intended to guide the choice of the right balance between flexibility and structure, are shown in Table 9.

Table 9: The Five Critical Agility and Plan-Driven Factors
(adapted from (Boehm and Turner, June 2003))

Factor	Agility discriminators	Plan-driven discriminators
Size (Number of people on team)	Well matched to small products and teams; reliance on person-to-person knowledge transfer and retention limits scalability.	Methods evolved to handle large projects and teams; hard to tailor down to small projects.
Criticality (The impact of a software defect in terms of comfort, money, and/or lives)	Untested on safety-critical products; potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products; hard to tailor down efficiently to low-criticality products.
Dynamism (The degree of requirements and technology change)	Simple design and continuous restructuring is excellent for highly dynamic environments, but present a source of potentially expensive rework for highly stable environments.	Detailed plans and "big design up front" excellent for highly stable environments, but a source of expensive rework for highly dynamic environments.
Personnel (Skill level of team)	Require continuous presence of a critical mass of scarce Level 2 or 3 experts; risky to use non-agile Level 1B people.	Need a critical mass of Level 2 and 3 experts during project definition, but can work with fewer later in the project. Can usually accommodate some Level 1B people.
Culture (Whether the individuals on the team prefer predictability/order or change)	Thrive in a culture where people feel comfortable and empowered by having many degrees of freedom; thrive on chaos.	Thrive in a culture where people feel comfortable and empowered by having their roles defined by clear policies and procedures; thrive on order.

Boehm and Turner have created a polar chart as a means for visually displaying a team's values for each of these criticality factors. An example of such a polar chart can be found in Figure 2. Each of the five factors has an axis. Each of the axes is labeled with carefully chosen values based on the authors' history. For each axis, the further from the graph's

center, the more conducive the method is toward plan-driven methods. Conversely, the more points lie toward the center of the chart, the more a project would likely benefit from agile methods.

Consider the black line joining the points of a sample project in Figure 4.2. Starting at the top of the chart, this team is comprised of a large number of novices and a small number of experts. Additionally, the requirements are not expected to change much throughout the project. The team members have a fairly strong preference for order and predictability. There are about 15 people on the team. The impact of a software defect is in essential funds. To clarify, an impact of "essential funds" indicates that a business could lose a large amount of money if there was a defect in the software. For example, our auction application could cause a loss of a large amount of money due to a software defect, as could software that ran a grocery store. Based on the shape of the polar chart for this particular application, the team would be best served by a plan-driven software development methodology.

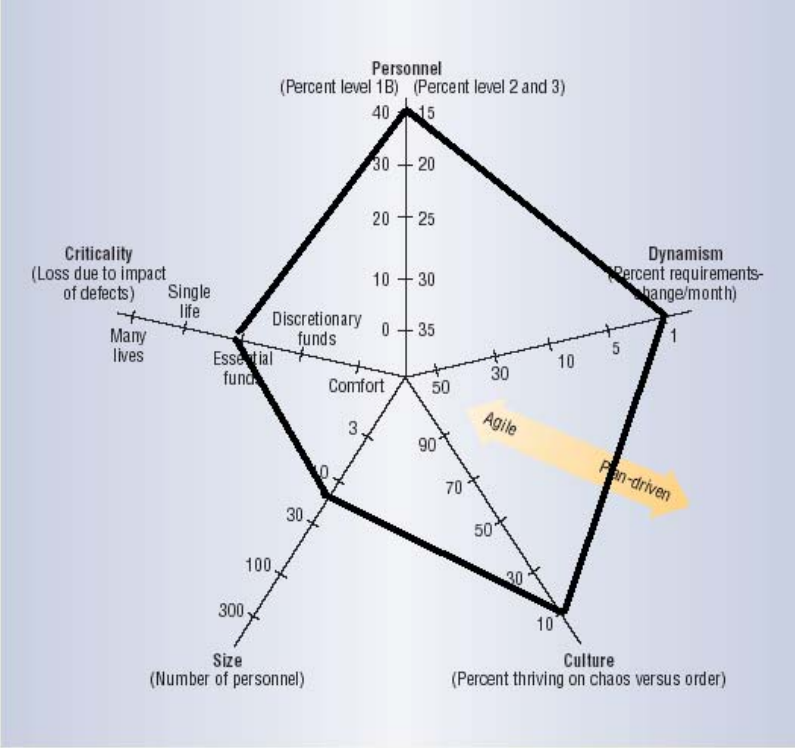

Figure 2: Example Polar Chart. (adapted from (Boehm and Turner, June 2003))

4.4 Risk-Driven Method for Balancing Agile and Plan-Driven Methods

With this background, you can now understand Boehm and Turner's five-step, risk-driven method for balancing agile and plan-driven methods. Each of the five steps will now be explained. The interactions between the steps are displayed in Figure 3.

Step One: Risk Analysis
Three different areas of risk are analyzed: environmental, agile, and plan-driven. Each of these areas is now defined.

- *Environmental risks* – risks that result from the project's general environment, as discussed in the earlier sections of this chapter and enumerated in Table 1.
- *Agile risks* – risks that are specific to the use of agile methods. Some of these are issues related to the ability of agile methods to scale to larger teams and projects and to handle the reliability needs of critical projects. Additionally, there are agile risks associated with not thoroughly documenting prior to coding, with the potential of personnel turnover/churn, and with having enough skilled people.
- *Plan-driven risks* – risks that are specific to the use of plan-driven methods. Some of these issue relate to the ability of plan-driven methods to handle rapid technology and/or requirements change, the need to deliver rapid results, and/or having enough team members skilled in plan-driven methods.

If not enough information is known about any of these risks, some resources can be spent to obtain some information about the project's aspects until the team feels more confident about the project risks.

Step Two: Risk Comparison
After the risks are identified, the team assesses and compares them. If the plan-driven risks outweigh the agile risks (meaning the issues related to using a plan-driven methodology are more concerning), then the team should adopt an agile method and proceed to Step Four. If the agile risks outweigh the plan-driven risks, then the team should adopt a plan-driven method and proceed to Step Four. If neither dominates – and the project characteristics do not clearly lie in the agile or plan-driven home ground – then the team should proceed to Step Three.

Step Three: Architecture Analysis
The optional Step Three is done when the project characteristics do not clearly lie in either the agile or plan-driven home ground or when parts of the system lie in an agile home ground and other parts of the system lie in the plan-driven home ground. If possible, the team develops a system architecture so that the team is able to use agile methods on the parts of the system where their strengths can be best applied. The remainder of the system is developed via plan-driven methods.

Step Four: Tailor Life Cycle
A project strategy is developed to address the risks identified in Step One, as was discussed earlier in the chapter. The life-cycle process is tailored around the identified risk patterns.

Step Five: Execute and Monitor
Consistent with the need to consistently monitor risk items, as discussed earlier in the chapter – the team must consistently reassess the risks related to agile and plan-driven methods. If the risk profile changes, the team should consider their choice of process model.

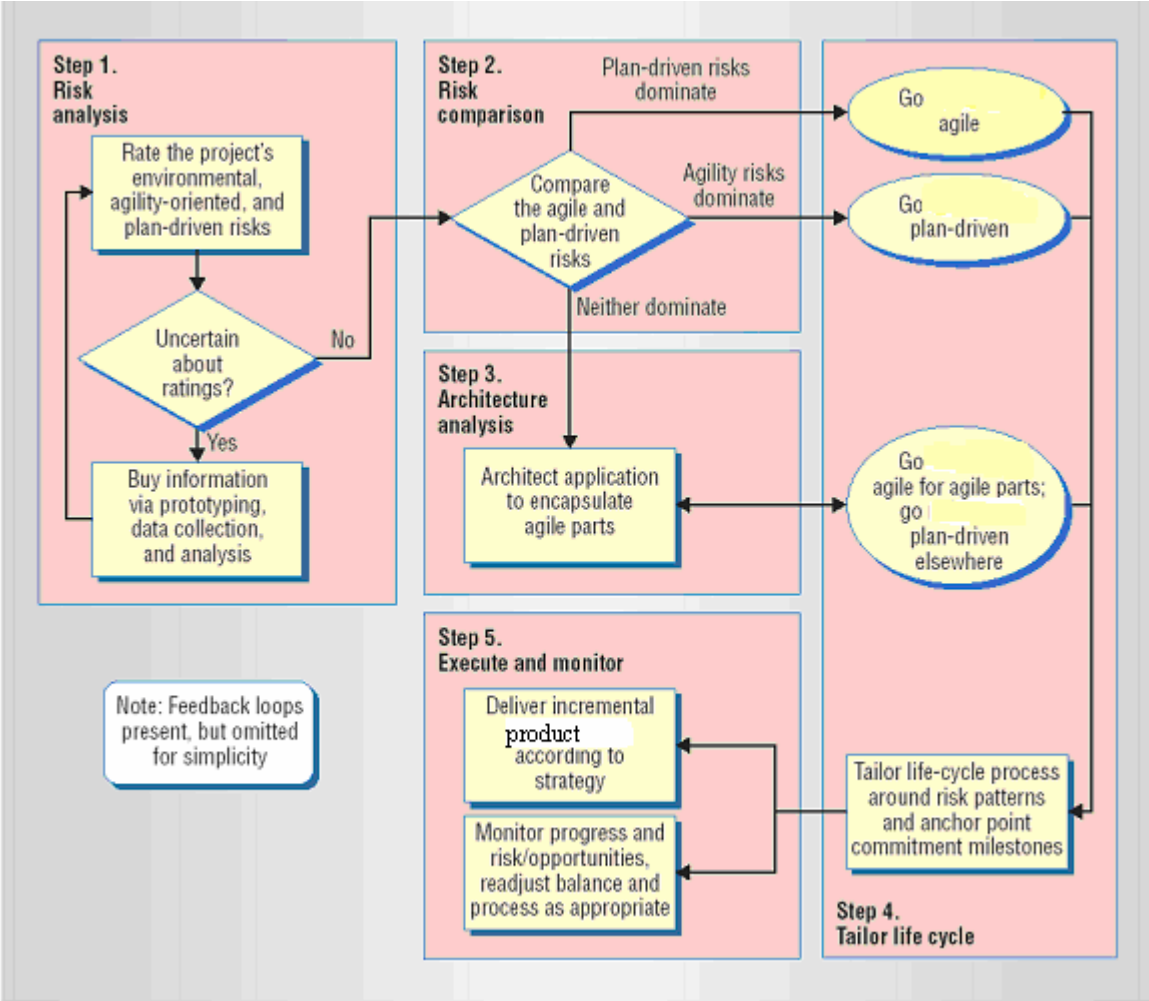

Figure 3: Boehm and Turner's Five Step Risk-Based method for balancing Agile and Plan-Driven methods. (adapted from (Boehm and Turner, 2003; Boehm and Turner, June 2003))

5 Summary

Several practical tips for risk management were presented throughout this chapter. The keys for successful risk management are summarized in Table 10.

Table 10 Key Ideas for Risk Management

	Be proactive about managing risk or you'll constantly be in crisis-driven, fire-fighting mode.
	Systematically surface risks by meeting with marketing and the customer, by using checklists and taxonomies, by comparing with past projects, and by decomposing large, unwieldy risks into smaller, more manageable risks.
	All the stakeholders must communicate about risks throughout the entire development cycle. Communication is at the center of the risk management process.
	Prioritize risks by computing the risk exposure of each risk. Sort the list of risks based upon the risk exposure and proactively manage those on the top of the list.
	Develop a "Top 10" risk list for your projects. It is likely that this "Top 10" list will contain risks that will appear on your next projects as well.
	Utilize a risk-driven process for choosing between an agile and a plan-driven process, or a hybrid of the two.

In the risk management cycle, product and project risks are identified, analyzed, and prioritized. The top-ranking risks are planned and mitigated. All risks are monitored. It is important for a project to focus on its critical success factors while keeping an eye on its risk factors. Risk management practices enable the team to find the opportunity in the risk items. Be proactive!

Glossary of Chapter Terms

Word	Definition	Source
Risk	potential future harm that may arise from some present action	(Wikipedia, 2004)
Risk Exposure	the product of the probability of a risk occurring multiplied by the magnitude of the loss if the risk did occur	(Boehm, 1989)
Risk Leverage	the quotient of the difference of the risk exposure before risk reduction minus the risk exposure after risk reduction, divided by the cost of risk reduction	(Pfleeger, 1998)
Risk Management	series of steps whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful software operation or a major source of expensive rework	(Boehm, 1989)

References

Boehm, B. (1989). Software Risk Management. Washington, DC, IEEE Computer Society Press.

Boehm, B. (January 1991). "Software Risk Management: Principles and Practices." IEEE Software: 32-41.

Boehm, B. and R. Turner (2003). Balancing Agility and Discipline: A Guide for the Perplexed. Boston, MA, Addison Wesley.

Boehm, B. and R. Turner (June 2003). "Using Risk to Balance Agile and Plan-Driven Methods." IEEE Computer **36**(6): 57-66.

Bruegge, B. and A. H. Dutoit (2000). Object-Oriented Software Engineering: Conquering Complex and Changing Systems. Upper Saddle River, NJ, Prentice Hall.

Cockburn, A. (2001). Agile Software Development. Reading, Massachusetts, Addison Wesley Longman.

Gluch, D. P., "A Construct for Describing Software Development Risks," Software Engineering Institute, Pittsburgh, PA CMU/SEI-94-TR-14.

Gupta, U. G. and R. E. Clarke (1996). "Theory and Applications of the Delphi Technique: A bibliography (1975-1994)." Technological Forecasting and Social Change **53**: 185-211.

Hall, E. M. (1998). Managing Risk: Methods for Software Systems Development, Addison Wesley.

Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Boston, Addison Wesley.

Pfleeger, S. L. (1998). Software Engineering: Theory and Practice. Upper Saddle River, NJ, Prentice Hall.

Standish (1995). "The Chaos Report."

Van Scoy, R. L., "Software Development Risk: Opportunity, Not Problem," Software Engineering Institute, Pittsburgh, PA CMU/SEI-92-TR-030.

Wikipedia (2004). Wikipedia, The Free Encyclopedia. http://www.wikipedia.org.

Chapter Questions

1. The Jones family has just moved in a new house. Mr. Jones did some research of this area and found out that the probability for a house to be flooded once in a six-month period is 0.5% and no house has flooded twice in a six-month period. Additionally, Mr. Jones evaluated that, in case a flood would happen, the property damage would be $4,000, on average. If Mr. Jones wants to buy flood insurance, what should he pay for a six-month policy based upon his research?

2. (Continued from Question 1) It is not possible for an insurance company to provide a rate quote as low as Mr. Jones likes it to be (or the insurance company wouldn't make any money!) After contacting several insurance companies, Mr. Jones found out that the lowest rate is $50 every six months. Compute the risk leverage if Mr. Jones buys the insurance. (Assume that the insurance company would pay $4,000 dollars if the flood occurs.)

3. Believe it or not, buying music CD can be a risky business. One day, your friend tells you that your favorite band has just released a new CD, and it is awesome (in her opinion, anyway). Complete a risk table for buying a new CD in order to minimize your risk.

4. A college student often has several assignments due each week. What are some of the factors a college student should think about in doing risk management for his or her

assignments?

5. Explain the five critical agile and plan-driven factors.