Chapter 3: Literature Reviews

Risk management was introduced not as a discipline in software development. The use of risk analysis can be traced back to the Babylonians in 3200 BC. Bernoulli's paper "Exposition of a New Theory on the Measurement of Risk" was published in 1738. It discussed many risk situations that occur in games of chance.

The insurance business is successful because of its ability to assess and deal with risk. Insurance companies have risks management departments which has the responsibility to assess corporate risks and to establish appropriate risk management programs that involve various kinds of insurance, contracts and policies. "The Economic Theory of Risk and Insurance" by Allan Willett was first published in 1901.

However, application of risk management to software development environments requires a good deal of tailoring. The process is still in its infancy. There is no large volume of software risk management literature available to date. The major contributor to the field of software risk management is Barry W. Boehm of United States Air Force Defense System Group. He wrote a groundbreaking book on the topic, Software Risk Management, IEEE Computer Society. I will first discuss his approach to software project risk management.

Boehm has divided risk management into two primary steps: risk assessment and risk control, each with three subsidiary steps, and offer a short list of techniques for each of the step, as depicted in Figure 3.1 [10].

The first primary step, risk assessment, involves risk identification, risk analysis, and risk prioritization:

- ***Risk identification*** produces lists of the project-specific risk items likely to compromise a project's success. The techniques include risk identification checklists, decision driver analysis, comparison with experience (assumption analysis) and decomposition.

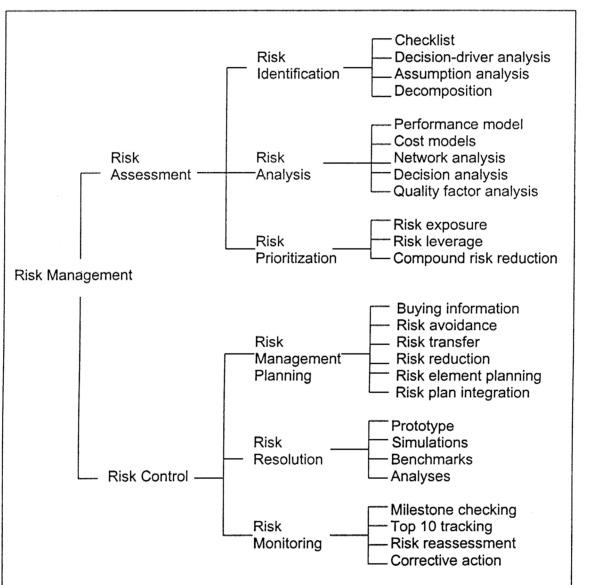

Figure 3.1: Software Risk Management Step

- ***Risk identification checklists*** (Figure 3.2) is derived based on a survey of experienced project managers, of the top-10 risk items most likely to compromise a software project's success. It also presents the managers' identification of the techniques most likely to resolve the risk items.

- In a ***decision driver analysis***, if factors other than technical and management achievability have driven a decision, then it will frequently be the source of a critical software risk item. Some of the most common classes of examples are:
  - ***Politically driven decisions***: Choice of equipment, choice of subcontractor, schedule and budget and allocation of responsibilities.
  - ***Marketing-driven decisions***: Gold plating, choice of equipment, schedule and budget.
  - ***Solution-driven versus Problem-driven decisions***: In-house components and tools, artificial intelligence and product champions.
  - ***Short-term versus long-term decisions***: Staffing, software reuse, and premature reviews.

- ***Assumption analysis***: Quite often major risk items are hidden behind optimistic assumptions. These arise most frequently from either ignorance about critical issues reflected in non-software people's decisions (accountants, hardware engineers, and user organizations), or from the tendency for software people to avoid conflict. Thus, when pressed to commit to ambitious budgets, schedules or performance requirements, software people will tempt to say, "We might be able to .....if we get all of the best people, if the vendor is delivered on time....".Analyzing these assumptions is an effective way of identifying risks.

Risk Item	Risk management technique
1. Personnel shortfalls	Staffing with top talent, job matching, team building, morale building, key-personnel agreements, cross-training, pre-scheduling key people.
2. Unrealistic schedules and budgets	Detailed multisource cost and schedule estimation, designing to cost, incremental development, software reuse, requirement scrubbing.
3. Developing the wrong software function and properties	Organization analysis, mission analysis, ops-concept formulation, user surveys and participation, prototyping, early users' manuals, off-nominal performance analysis, quality-factor analysis.
4. Developing the wrong user interface	Task analysis, prototyping, scenario, user characterization (functionality, style, and workload).
5. Gold plating	Requirements scrubbing, prototyping, cost-benefit analysis, designing to cost.
6. Continuing stream of requirement changes	High change threshold, information hiding, incremental development (defer changes to later increments).
7. Shortfalls in externally furnished components	Benchmarking, inspections, reference checking, compatibility analysis.
8. Shortfalls in externally performed tasks	Reference checking, preaward audits, award-fee contracts, competitive design or prototyping, team building.
9. Real-time performance shortfalls	Simulation, benchmarking, modeling, prototyping, instrumentation, tuning.
10. Straining computer-science capabilities	Technical analysis, cost-benefit analysis, prototyping, reference checking.

Figure 3.2: A prioritized top-10 software risk items

- ***Comparison with experience***: Compare software project assumptions with previous experience.
- ***Murphy's Law***: Useful in identifying risks that the software product or project is over optimized around the nominal or fault-free case.

➢ ***Decomposition***: Look within any big, poorly-described blobs within the project's plans and specifications, decompose these blobs into their constituent elements to identify risk items.

- ***Pareto 80-20 phenomena***: 80 percent of the contribution comes from 20 percent of the contributors. Examples: 20 percent of the modules contribute 80 percent of the cost, 20 percent of the modules contribute 80 percent of the errors, 20 percent of the errors cause 80 percent of the down time, 20 percent of the errors consume 80 percent of the cost to fix, and 20 percent of the modules consume 80 percent of the execution time. It is likely that, a large, poorly-described blob will be one of the 20 percent that will cause 80 percent of the project's problems.
- ***Task dependencies***: A major source of software project schedule risk is the existence of unidentified task dependencies within poorly-described complex tasks. Decomposing these complex tasks into a PERT chart or task-dependency network will identify many of these schedule risk items.
- ***Uncertainty areas***: mission requirements, life-cycle concept of operation, system performance drivers, user interface characteristics, and interfacing system characteristics.

♦ ***Risk analysis*** assesses the loss probability and loss magnitude associated with each of the identified risk items, and it assesses compound risks in risk-item interaction. Techniques include performance models, cost models network analysis,

statistical decision analysis, decision trees and quality factor (like reliability, security, and availability) analysis.

- ***Decision analysis***: The decision tree structures risk situations in terms of the possible decisions one can make, and in terms of the risk exposure factors associated with each decision option. The various options are characterized by their possible outcomes, with their possibilities of occurrence and the resulting cost (or benefit) of each outcome.
- ***Network analysis***: In the schedule risk area, one of the best techniques for analyzing and reducing risk is to break down each large schedule blob into an activity network or PERT of its constituent tasks. By doing so, one can analyze various aspects of the chart for high-risk features. Examples: the presence of high fan-in or high fan-out nodes, multiple critical-path situations and the presence of highly overlapped paths.
- ***Cost risk analysis***:
  - ***Cost model and cost driver analysis***: Most software cost models use a series of parameters called cost drivers that describe aspects of the project significantly affect its cost. Examples of cost drivers are product size, personnel experience and capability, hardware constraints, and use of tolls or modern programming practices.
  - ***Schedule risk analysis***: Most models such as COCOMO[III] have a development schedule estimating capability that can be used for risk analysis as well. The COCOMO equation for estimating the project's development time, TDEV in months is

$$TDEV = 2.5(MM)^{0.32}$$ where MM is man-months.

[III] COCOMO is an acronym formed from the first two letters of each word in COnstructive COst MOdel.

- ***Performance risk analysis***: Techniques such as simulation, benchmarking, modeling, prototyping, instrumentation, and tuning are effective.
- ***Quality factors analysis***: For reliability, availability and safety analysis, techniques such as checklists, hierarchy design, fault tree analysis, failure mode and effect analysis, and static and dynamic analysis of specifications and programs are effective. For ease of use, prototyping techniques and human-computer interface treatments are useful.

♦ ***Risk prioritization*** produces a ranked ordering of the risk items identified and analyzed. Typical techniques include risk exposure analysis, risk reduction leverage analysis (involving cost-benefit analysis), and Delphi or group-consensus techniques.

➢ ***Risk exposure (RE)***: Sometimes also called risk impact, is defined by

$$RE = Prob(UO) * Loss(UO) \quad [10]$$

where Prob(UO) is the probability of an unsatisfactory outcome and Loss(UO) is the loss to the parties affected if the outcome is unsatisfactory.

➢ ***Risk Reduction Leverage(RRL)***: RRL is defined as

$$RRL = \frac{RE_{before} - RE_{after}}{\text{Risk Reduction Cost}} \quad [10]$$

where RE_{before} is the RE before initiating the risk reduction effort and RE_{after} is the RE afterwards. RRL is a measure of the relative cost-benefit of performing various candidate risk reduction activities.

The second primary step, risk control, follows the classic closed-loop control approach shown in Figure 3.3. It involves risk management planning, risk resolution and risk monitoring.

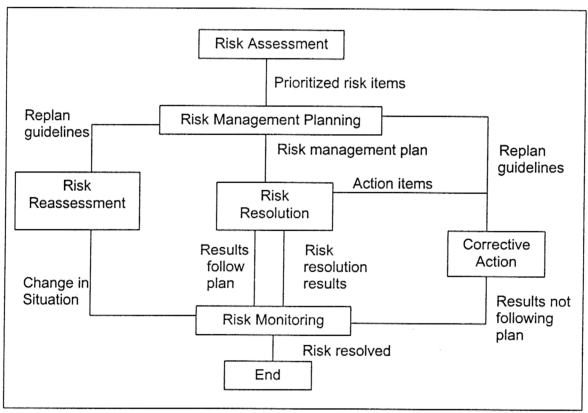

Figure 3.3: Risk management control loop

- ***Risk management planning*** produces plans for addressing each risk item, including the coordination of the individual risk-item plans with each other and with the overall project plan. Depending on the project's size and criticality, the project risk management plan may consist of a set of briefing charts, a section of the project plan, or a standalone document. Or, it may consist of a series of risk management plans, one for each of the project's major phases. The most important thing is that the risk management plan should be used to drive the budget and schedule

parameters of the overall project plan, particularly in the early phase, so that there is enough time and effort available to resolve risk items early before they become major project crises. Risk management planning techniques include checklists of risk-resolution techniques, cost-benefit analysis, statistical decision analysis of the relative cost and effectiveness of alternative risk-resolution approaches, and standard risk management plan outlines, forms and elements.

- Risk management planning process as depicted in Figure 3.4.
- Risk management plan formulation and coordination. An example of plan outline is presented in Figure 3.5.

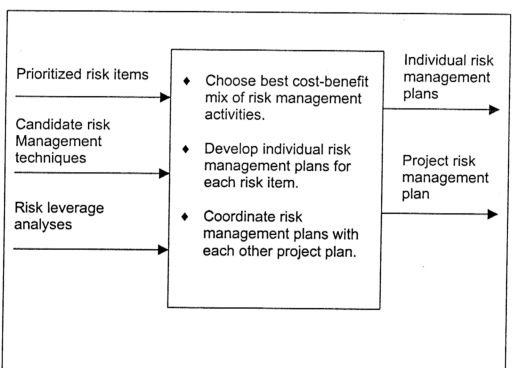

Figure 3.4: Risk management planning process

For each risk item, answer the following questions

1. Why?
   Risk item importance, relation to project objectives
2. What, when?
   Risk resolution deliverables, milestones, activity nets
3. Who, where?
   Responsibilities, organization
4. How?
   Approach (prototypes, surveys, models.....)
5. How much?
   Resources (budget, schedule, and key personnel)

Figure 3.5: Risk management plan outline

- ***Risk resolution*** produces a situation in which the risk items are eliminated or otherwise resolved (e.g. risk avoidance via relaxation of requirements). Techniques include prototypes, simulations, benchmarks, mission analysis, key-personnel agreements, design-to-cost approaches and incremental development.

- ***Risk monitoring*** involves tracking the project's progress toward resolving its risk items and taking corrective appropriate actions indicated in Figure 3.3 to ensure that risk control is a closed-loop process. Typical techniques include milestone tracking and a top-10 risk-item list (Figure 3.2) that is highlighted at each weekly, monthly, or milestone project reviews and follow up with reassessment of the risk item or corrective action.

Richard Fairley of IEEE, has also greatly contributed to the techniques of risk management. He followed up on Boehm's techniques and demonstrated their application on a telecommunication software project. He made significant use of the COCOMO cost estimation model to estimate the impact of risk factors on budget and schedule, and demonstrated the use of statistical distributions to provide a likely range of probable outcomes. He then developed contingency plans based on this information and documented what actions project management actually takes on the project. He also created a seven-step process for risk management that can be applied to all types of software projects:

1. ***Identify risk factors***: A regression-based cost model is used to identify and assess the impact of risk factors on estimated project effort. This is to surface risks before they become problems and adversely affect a project.
2. ***Assess risk probabilities and effects on the project***: Probability of a risk is assessed by computing probability distributions of code size and complexity. Monte Carlo simulation is then used to compute the distribution of estimated project effort as a function of size, complexity, timing and memory using regression-based modeling.
3. ***Develop strategies to mitigate identified risks***. Risk mitigation involves two types of strategies: Action planning and contingency planning. Action planning addresses risks that can be mitigated by immediate response; contingency planning, on the other hand, addresses risks that require monitoring for some future response should the need arise. Contingency planning involves preparing a contingency plan, a crisis-management plan, and a crisis-recovery procedure. A contingency plan addresses the risks not addressed in the action plans. A crisis-management plan is the backup plan to be used if the contingency plan

fails to solve a problem within a specified time. A crisis-recovery procedure is invoked when the crisis is over, whether the outcome is positive or negative.

4. ***Monitor risk factors***. Risk monitoring ensures that there is a closed-loop process by tracking risk-reduction progress and applying whatever corrective actions necessary to keep the risk-resolution process on track.
5. ***Invoke a contingency plan***. A contingency plan is invoked when a quantitative risk indicator crosses a predetermined threshold.
6. ***Manage the crisis***. Despite a team's best efforts, the contingency plan may fail, in which case the project enters crisis mode. There must be some plan for seeing a project through this situation, including allocating sufficient resources and specifying a drop-dead date, at which time the management must reevaluate the project for more drastic corrective action (possibly major redirection or cancellation of the project). The elements of crisis management are to:
   - Announce and generally publicize the problem.
   - Assign responsibilities and authorities: responsible parties stop all other work to concentrate on the problem.
   - Update status frequently.
   - Relax resource constraints: management dedicated all needed resources to solve the problem.
   - Have project personnel operate in burnout mode: the crisis team works as many hours as possible.
   - Establish a drop-dead date: set a drop-dead date because no one can sustain this kind of effort indefinitely.
   - Release unessential personnel: all personnel not assigned to the team continue with normal work activities.

7. ***Recover from the crisis***. After a crisis, certain actions are required, such as rewarding personnel who have worked in burnout mode for an extended period and reevaluating cost and schedule in light of the drain on resources from managing the crisis.
    - ***Conduct a crisis postmortem***: to fix any systemic problems that may have precipitated the crisis and to document any lesson learned.
    - ***Calculate cost to complete the project***: to know how the crisis has affected the project's budget and schedule.
    - ***Update plans, schedules and work assignments***: Time and resources have been expended on the contingency plan and crisis management, so original project budget and schedule are likely to be invalid.
    - ***Compensate workers for extraordinary efforts***: Bonuses time off and overtime pay are appropriate forms of compensation.
    - ***Formally recognize outstanding performers and their families***: This may include formal letters of commendation, accelerated promotions, and letters to the families of those who worked around the clock. Free dinners and weekend vacations are other ideas.

The well known Software Engineering Institute of Carnegie Mellon University[IV], Pittsburgh had also contributed much to our knowledge of risk in software environments. Their 1993 paper, *Taxonomy-Based Risk Identification* [17], includes a compilation of questions that bring out risks common to software development projects. The taxonomy[V] organizes software development risks into 3 levels – classes, element and attribute.

[IV] The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

[V] Taxonomy is a scheme that partitions a body of knowledge and defines the relationships among the pieces. It is used for classifying and understanding the body of knowledge. See IEEE software Engineering Standards Collections, IEEE-STD-610.12, 1990 for more information.

These taxonomic classes are further divided into elements and each element is characterized by its attributes. The three major classes are:

1. ***Product Engineering***: The technical aspects of the work to be accomplished.
   - ***Requirements***: The definition of what the software product is to do, the needs it must meet, how it is to behave, and how it will be used. This element also addresses the feasibility of developing the product and the scale of the effort. It attributes are:
     - Stability
     - Completeness
     - Clarity
     - Validity
     - Feasibility
     - Precedent
     - Scale
   - ***Design***: The translation of requirements into an effective design within a project and its operational constraints. Its attributes are:
     - Functionality
     - Difficulty
     - Interfaces
     - Performance
     - Testability
     - Hardware constraints
     - Non-development-software
   - ***Code and unit test***: The translation of software designs into code that satisfies the requirements allocated to individual units. Its attributes are:

- Feasibility
- Testing
- Coding/implementation

- ***Integration and test***: The integration of units into a working system and the validation that the software product performs as required. Its attributes are:
  - Environment
  - Product
  - System

- ***Engineering specialties***: Product requirements or development activities that may need specialized expertise. Its attributes are:
  - Maintainability
  - Reliability
  - Safety
  - Security
  - Human factors
  - Specifications

2. ***Development Environment***: The methods, procedures, and tools used to produce the product.
   - ***Development process***: The definition, planning, documentation, suitability, enforcement, and communication of the methods and procedures used to develop the product. Its attributes are:
     - Formality
     - Suitability
     - Process control

- Familiarity
- Product control

- ***Development system***: The tools and supporting equipment used in product development, such as CASE[VI] tools, simulators, compilers, and host computer systems. Its attributes are:
  - Capacity
  - Suitability
  - Usability
  - Familiarity
  - Reliability
  - System support
  - Deliverabilty

- ***Management process***: The planning, monitoring, and controlling of budgets and schedules; controlling factors involved in defining, implementing, and testing the product; the project manager's experience in software development, management, and product domain; and the manager's expertise in dealing with external organizations including customers, senior management, matrix management, and other contractors. Its attributes are:
  - Planning
  - Project organization
  - Management experience
  - Program interfaces

[VI] Computer-Aided Software Engineering tools that provide automated support for some portion of the systems development process. They generally lead to improvements in software productivity.

- ***Management methods***: The methods, tools and supporting equipment that will be used to manage and control the product development. Its attributes are:
  - Monitoring
  - Personnel management
  - Quality assurance
  - Configuration management
- ***Work environment***: The general environment within which the work will be performed, including the attitudes of people and the levels of cooperation, communication, and morale. Its attributes are:
  - Quality attitude
  - Cooperation
  - Communication
  - Morale

3. ***Program Constraints***: The contractual, organizational, and operational factors within which the software is developed but which are generally outside of the direct control of the local management.
   - ***Resources***: The external constraints imposed on schedule, staff, budget, or facilities. Its attributes are:
     - Schedule
     - Staff
     - Budget
     - Facilities
   - ***Contract:*** The terms and conditions of the project contract. Its attributes are:
     - Type of contract
     - Restrictions

- Dependencies

- ***Program interfaces***: The external interfaces to customers, other contractors, corporate management, and vendors. Its attributes are:
  - Customer
  - Associate contractors
  - Subcontractors
  - Prime contractor
  - Corporate management
  - Vendors
  - Politics

SEI had also designed and developed the ***Software Risk Evaluation Method*** [93] as one of the methods of software risk management that can meet the specific needs of decision-makers who are responsible for managing software-intensive programs or projects.

The Software Risk Evaluation (SRE) is a diagnostic and decision making tool that enables the identification, analysis, tracking, mitigation, and communication of risks in software-intensive programs.

SRE provides a program manager with a mechanism to anticipate and address program risks. It introduces a set of activities that, when initiated, begins the process of managing risk. These activities can be integrated with existing methods and tools to enhance program management practices:

- Diagnostic – are the risks acceptable for starting a program?
- Baselining- the SRE identifies a critical set of risks before they become problems so they can be managed on a continuous basis.

- Preparing for a critical milestone.
- "Recovering from crisis" – The SRE provides a way to reset a baseline for a program.

The SEI emphasizes on the continuous aspect of risk management. It employs processes, methods, data, and tools for managing risks as an integrated function of the engineering and management activities in a program. Fundamental to this practice are the activities of communicating, identifying, analyzing, planning, tracking and controlling risks. In addition, risk management practice requires various infrastructure and support elements essential for operation in a dynamic environment.

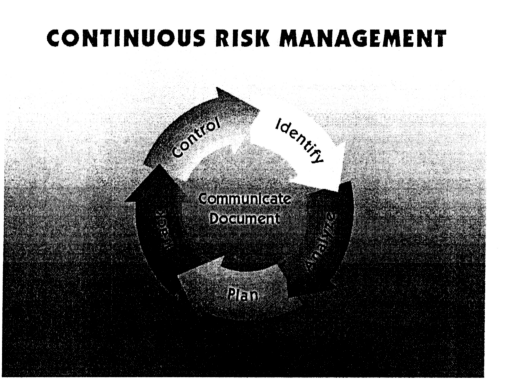

Figure 3.6: SEI Risk Management Paradigm

The model used to represent the *SEI Risk Management Paradigm* is a circular set of activities depicted above. The paradigm illustrates a set of functions that are identified as

continuous activities throughout the life cycle. The arrows within the model represent the logical path and the sequence in flow of information within the activities in risk management. Communication is placed in the center of the paradigm because it is both the conduit through which all information flows and, often, is the major obstacle to risk management.

Function	Description
Identify	Search for and locate software-related risks before they become problems that can adversely affect the program or project.
Analyze	Transform risk data into decision-making information. Evaluate impact, probability, and timeframe, classify risks, and prioritize risks.
Plan	Translate risk information into decisions and mitigating actions (both present and future) and implement those actions.
Track	Monitor risk indicators and mitigation actions.
Control	Correct for deviations from the risk mitigation plans by using existing program or project management control functions.
Communicate	Providing and exchanging risk management information among the functions and at all levels of the organization. This activity is represented in the center of the model to emphasize its pervasiveness and criticality for implementing the other activities in the paradigm.

Figure 3.7: The function of Continuous Risk Management

The table above shows the functions of continuous risk management. Each risk nominally goes through these functions sequentially, but the activity occurs continuously, concurrently (e.g., risks are tracked in parallel while new risks are identified and analyzed), and iteratively (e.g., the mitigation plan for one risk may yield another risk) throughout the project life cycle.

There are seven principles that provide a framework to accomplish effective risk management:

Principle	Effective risk management requirements
Global perspective	• Viewing software development within the context of the larger systems-level definition, design and development. • Recognizing both the potential values of opportunity and the potential impact of adverse effects.
Forward-looking view	• Thinking toward tomorrow, identifying uncertainties, anticipating potential outcomes. • Managing project resources and activities while anticipating uncertainties.
Open communication	• Encouraging free-flowing information at and between all project levels. • Enabling formal, informal and impromptu communication. • Using processes that value the individual voice (bringing unique knowledge and insight to identifying and managing risk).

Principle	Effective risk management requirements
Integrated management	• Making risk management an integral and vital part of project management. • Adapting risk management methods and tools to a project's infrastructure and culture.
Continuous process	• Sustaining constant vigilance. • Identifying and managing risks routinely throughout all phases of the project's life cycle.
Shared product vision	• Mutual product vision based on common purpose, shared ownership, and collective communication. • Focusing on results.
Teamwork	• Working cooperatively to achieve common goal. • Pooling talents, skills, and knowledge.

Figure 3.8: The Seven Principle of effective Software Risk Management.

NASA Lewis Research Center had also came out with its own ***Risk Management Cycle*** [57]. The cycle phases are:

- ***Identify***: identify that a risk exits and give it a meaningful name.
- ***Analyze***: Determine the severity of the risk. If the risk is negligible (low to medium severity, low likelihood of occurrence), stop here. However, if the risk could cause damage to the system or the system's users, continue.

- ***Plan*:** Decide how to combat the risk based on the risk's severity and likelihood of occurrence.
- ***Mitigate*:** Follow the plan formulated in the previous phase as closely as possible to combat the risk. If this approach does not work, return to the previous phase and make a new plan. If the plan does work, continue analyzing the risk to determine whether it has been reduced to an acceptable severity level.
- ***Track*:** Once the risk has been mitigated to an acceptable severity level, the risk should be tracked to ensure the continued control of the risk. If at any time the risk seems to resurface, the risk management cycle should begin again, starting with the analysis phase.

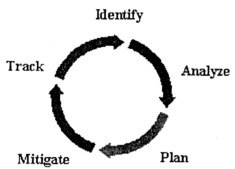

Figure 3.9: NASA Risk Management Cycle

Another newer method for Software Risk Management, called *Riskit* [59] is from University of Maryland, USA.

The Riskit method covers risk management from risk identification to selection and planning of risk mitigating action. A high-level view of the main aspects of the method is represented in Figure 3.10. As the figure implies, implementing the risk mitigating actions is considered as part of the project management.

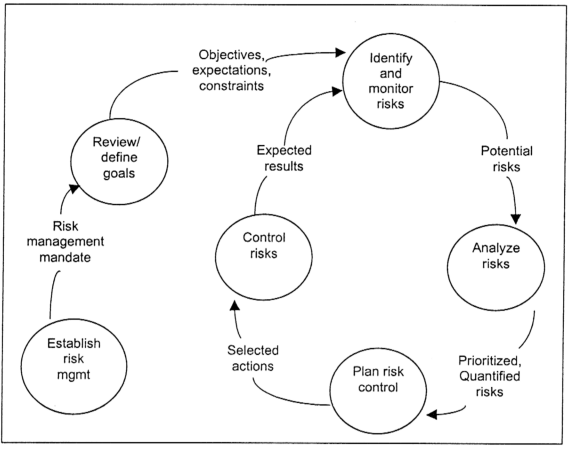

Figure 3.10: The Riskit process

The following list gives an overview of the steps in the method:

- ***Establish risk management***
  - Create risk management infrastructure for the organization.
  - Define risk management mandate for the project.
- ***Review and Define Goals***
  - Review stated goals.
  - Identify and define implied expectations.
  - Identify and define stakeholders and their association with goals.
- ***Identify and monitor***

- Identify potential risks using multiple identification methods.
- Monitor risk situation.

- ***Risk Analysis***
  - Classify identified risks into risk elements: factors and events and document their relationships.
  - For each risk event, describe plausible reactions, resulting in a risk scenario.
  - For each scenario, estimate the effect on goals that are affected.
  - Rank the scenarios based on their impact on the utility loss by stakeholders.
  - Estimate probabilities for each scenario.

- ***Risk Control Planning***
  - Select the highest risk scenarios and propose controlling actions for them.
  - Prioritize controlling actions based on the impact on reducing risks.

- ***Controlling of risks***
  - Select controlling actions and implement them.

Each of the steps in the method is supported by some techniques or tools.

The method takes advantage of many existing risk management approaches, such as risk management checklists [10], [17] and risk exposure [9]. The method can be applied by an individual project manager or a dedicated risk management team.

The Riskit method uses a graphical formalism, called Riskit analysis graph as a tool to formalize risks and analyze their relationships. The Riskit analysis graph is filled in gradually during the risk management cycle and it can be used to communicate about each risk to each participant. Each path in the Riskit analysis graph represents a scenario that can be evaluated for its impact and probability. The Riskit analysis graph can be edited with a drawing tool tailored for the Riskit method.

Each risk scenario is evaluated against the relevant project and product characteristics, e.g., a risk scenario's effect on schedule, cost, and company reputation. The scenarios are compared with each other in terms of their utility loss, i.e., the "pain" they cause to each stakeholder. This allows different scenarios to remain comparable with each other. The probabilities of each scenario are also compared and the highest risks are addressed in the risk mitigation planning step.

Another two contributors are worth mentioning here; they are:

- Defend System Management College – "***Risk Assessment Techniques***" [28].

  It deals with general methods of risk assessment, but the techniques apply well to software-specific risk. The primary quantitative methods of risks analysis is summarized below:

  - ***Network methods***: represent the project tasks dependencies as a network, and support the analysis of schedule and budget risks as function of the probabilities of overruns in each individual task.
  - ***Decision analysis methods***: represent the outcomes of major program decision in terms of decision trees, using estimates of the probabilities and costs associated with the possible decision outcomes.
  - ***Method of moments***: This is a generalized case of the PERT[VII] sizing risk analysis method frequently used in software cost estimation. Practical use of this generality is infrequent.
  - ***Work breakdown structure simulation methods***: consists of breaking down a large project into a hierarchy of Work Breakdown Structure (WBS) elements.

[VII] PERT stands for Program Evaluation and Review Technique. It is a diagramming technique that depicts project activities and their inter-relationships.

Each of the bottom-level WBS elements are then described in terms of a range or probability distribution of costs or other additive resource requirements. These can then be combined statistically to produce overall system risk ranges.

- ***Graphic methods***: graphing and combining risk probabilities on 'normal probability paper'. This method is rarely used.
- ***Estimating relationship methods***: use probability distributions of individual project cost drivers or Cost Estimating Relationships (CERs) to derive the cumulative effects over several CERs.
- ***Risk factor***: various WBS components of the system are compared with respect to their levels of risk (by using expert judgement, group consensus, or other methods), to determine a set of risk factors for the components.

❖ Humphrey – "***A Method of Assessing the Software Engineering Capability of Contractors***" [39].

There are 101 graded questions for determining the relative maturity level of a software organization and five maturity levels (CMM[VIII]) associated with the organization's process of developing software:

1. ***Initial***: ill-defined procedures and controls.
2. ***Repeatable***: uses basis standard and methods.
3. ***Defined***: uses thorough, consistent standards and methods.
4. ***Managed***: uses quantitative, closed-loop management methods.
5. ***Optimized***: methods optimized with respect to quantitative experience base.

[VIII] CMM is developed by the Software Engineering Institute (SEI). It is a five-point grading scheme that used to determine the compliance of an organization to software process maturity level.

These maturity levels correlate strongly with the level of risk that the software organization will run out of control, particularly when developing an unfamiliar product within a tight budget and schedule.

3.1 Formal/Traditional Risk Management Services

There are currently several companies and institutions providing formal risk management services to software organizations. Examples are Software Engineering Institute of Carnegie Mellon University [116] and Kulik & Lazarus Consulting, Inc. [110]. Due to the high cost and long duration of these type of services, there are only cost-justified for software intensive program and large scale software project. In general, these services have the following characteristics:

- Risk experts or risk consultants are used;
- Involves a series of field visits/studies and interviews with clients;
- The cost of these services ranges from US$10,000 to US$50,000 per project;
- Usually takes 4 to 16 weeks to complete;
- Typically costs 1% to 2% of project budget;
- Significantly impact on staff time, typically more than 2 hours for each staff member;
- The outcome of these services is long reports concerning risks to the clients.

3.2 Automated Risk Management Aids

There are a number of software packages available that calculate schedule, cost, and technical risk quantities based on the user's estimates of the probabilities and criticality of the project's constituent elements.

Schedule risk calculations generally assume a PERT network model, accept probability distributions for the duration of each activity in the network, and use a Monte Carlo[IX] approach to run a sequence of possible critical-path schedule outcomes to approximate the probability distribution of the overall project schedule.

The cost and technical risk calculations use a similar Monte Carlo approach, but also need the user to define how the component cost and technical risk quantities combine together (additively, multiplicatively, maximum of several candidates, etc.).

Following is a summary, in alphabetical order, of some of the leading risk management software packages:

Software Name	Origin	Functions/Features	Descriptions/comments
Open Plan	Welcom Software Technology	♦ Uses PERT network model to approximate project schedule.	♦ Powerful project management software for project planners and schedulers ♦ Never consider other factors contributed to project's risk.
PROMAP V	LOG/AN Inc.	Estimates technical risk as a function of the cost and schedule probability distributions.	♦ Operate on large mainframe. ♦ Difficult to use.

[IX] Monte Carlo Simulation is a sophisticated form of what-if analysis. It replaces points of estimate with fuzzy values and helps to characterize the range of potential outcomes. Probability of reaching specific targets is then assessed.

Software Name	Origin	Functions/Features	Descriptions/comments
PROSIM	Venture Analytical Associates	♦ Has network modeling capabilities and uses cost-schedule–technical relationships modeling approach.	♦ Follow-on to the Army VERT model. ♦ Risk identification software that concentrates on cost, schedule and technical risks only.
PSAK	Kulik & Lazarus Consulting Inc.	♦ Uses special staff surveys as the primary data source. ▪ Identifies project strengths and risk areas. ▪ Quantifies overall project risk and identifies high-confidence project completion dates. ▪ Develops action plans to reduce/ eliminate project risks.	♦ Self-administered project assessment tool. ♦ Based on the principle that project staff have considerable insight in a project's status. ♦ Areas assessed include requirements, plans and schedules, quality, project resources, technical complexity, project management infrastructures, teamwork, third-party involvement, and development processes. ♦ Add-on tool to Microsoft Excel.

Software Name	Origin	Functions/Features	Descriptions/comments
@ RISK	Palisade Corp., USA	♦ Develops a model. ♦ Identifies uncertainty with probability distribution. ♦ Analyzing the model with simulation to determine the range and probabilities of all possible outcomes. ♦ Making a decision based on the results.	♦ Risk Analysis and Simulation add-in for Microsoft® Project. ♦ Quantitative method that seeks to determine the outcomes of a decision as a probability distribution. ♦ High-resolution graphics are used to present the results.
RISK MASTER	Sphygmic Software Limited	♦ Addresses the quantitative aspects of risk management. ♦ Results are reported in a number of different formats to let user see what are the chances of completing a project within a specified time or cost. ♦ A 'Trigger' facility allowing users to model complex internal and external events including conditional branching along multiple paths, and correlation between related activities.	♦ An advanced graphical risk analysis tool. ♦ Applies Monte Carlo simulation techniques to derive the overall impact on a project of variances in cost, schedule and resources

Software Name	Origin	Functions/Features	Descriptions/comments
RISNET	John M. Cockerham Associates	♦ Have strong network logic capabilities.	♦ Its approach to technical risk estimation is similar to PROMAP V. ♦ Have both a mainframe and a PC capability.
SLAM	Pritsker and Associates	♦ Have powerful network capabilities. ♦ Applies cost-schedule-technical relationships.	♦ A risk analysis tool rather than a risk management tool.
SLIM	QSM Software Safari	♦ Identifies the best development strategy with the minimum risks based on inputs. ♦ Alternatives can be explored using GUI.	♦ Project estimation tool. ♦ Users need to input project assumptions and goals.
SLIM-CONTROL	QSM Software Safari	♦ Lets users forecast likely outcomes and allow them to explore alternative strategies for project completion. ♦ Uses the "core-metrics" recommended by the SEI.	♦ Project control tool. ♦ Performs monthly health checks to identify when a project is not performing as planned.

Software Name	Origin	Functions/Features	Descriptions/comments
SLIM-Metrics	QSM Software Safari	♦ Collects and analyzes software project data. ♦ Identifies development bottlenecks. ♦ Quantifies benefits of process improvements.	♦ A metric repository and process assessment tool.
wInsight 3.0	C/S Solution Inc.	♦ Identifies problem areas and analyzing "earned value" performance management data. ♦ Support both Integrated Product Development team and cost/schedule professionals	♦ For Windows 3.1, Windows 95, Windows NT, and Macintosh. ♦ Use to streamline the process of reporting and analyzing performance management data. ♦ Used by the (DSMC) to teach performance management techniques.

Software Name	Origin	Functions/Features	Descriptions/comments
SLIM-Metrics	QSM Software Safari	♦ Collects and analyzes software project data. ♦ Identifies development bottlenecks. ♦ Quantifies benefits of process improvements.	♦ A metric repository and process assessment tool.
wInsight 3.0	C/S Solution Inc.	♦ Identifies problem areas and analyzing "earned value" performance management data. ♦ Support both Integrated Product Development team and cost/schedule professionals	♦ For Windows 3.1, Windows 95, Windows NT, and Macintosh. ♦ Use to streamline the process of reporting and analyzing performance management data. ♦ Used by the (DSMC) to teach performance management techniques.