UNIT –V

RISK MANAGEMENT

RISK MANAGEMENT

- Risk in software engineering ,is the **uncertainty of a potential problem that could negatively impact a project's success**, leading to losses like cost overruns, schedule delays, poor quality, or technical failures.

- Risk analysis and management are actions that help a software team to understand and manage that uncertainty.

- RISK may or may not occur in a project, it is just a prediction.

- Risk management is done by entire software team including managers, developers, testers and support team.

- The work product of risk management is **RMMM (Mitigation, Monitoring , management) plan**.

REACTIVE VERSUS PROACTIVE RISK STRATEGIES

- **Reactive** - When the team reacts after the risk event had occurred it is reactive risk strategy.
- Team does nothing until a risk is encountered and suddenly they start ***Fire-fighting*** once risk occurs.
- If that risk is failed to recover **Crisis management** will come into picture.
- Reactive strategy will cause a potential loss to the team.
- **Proactive** – It is an intelligent way to handle risk.
- Team will maintain a **PLAN B for every action** , well in advance before the commencement of project work

REACTIVE VERSUS PROACTIVE RISK STRATEGIES

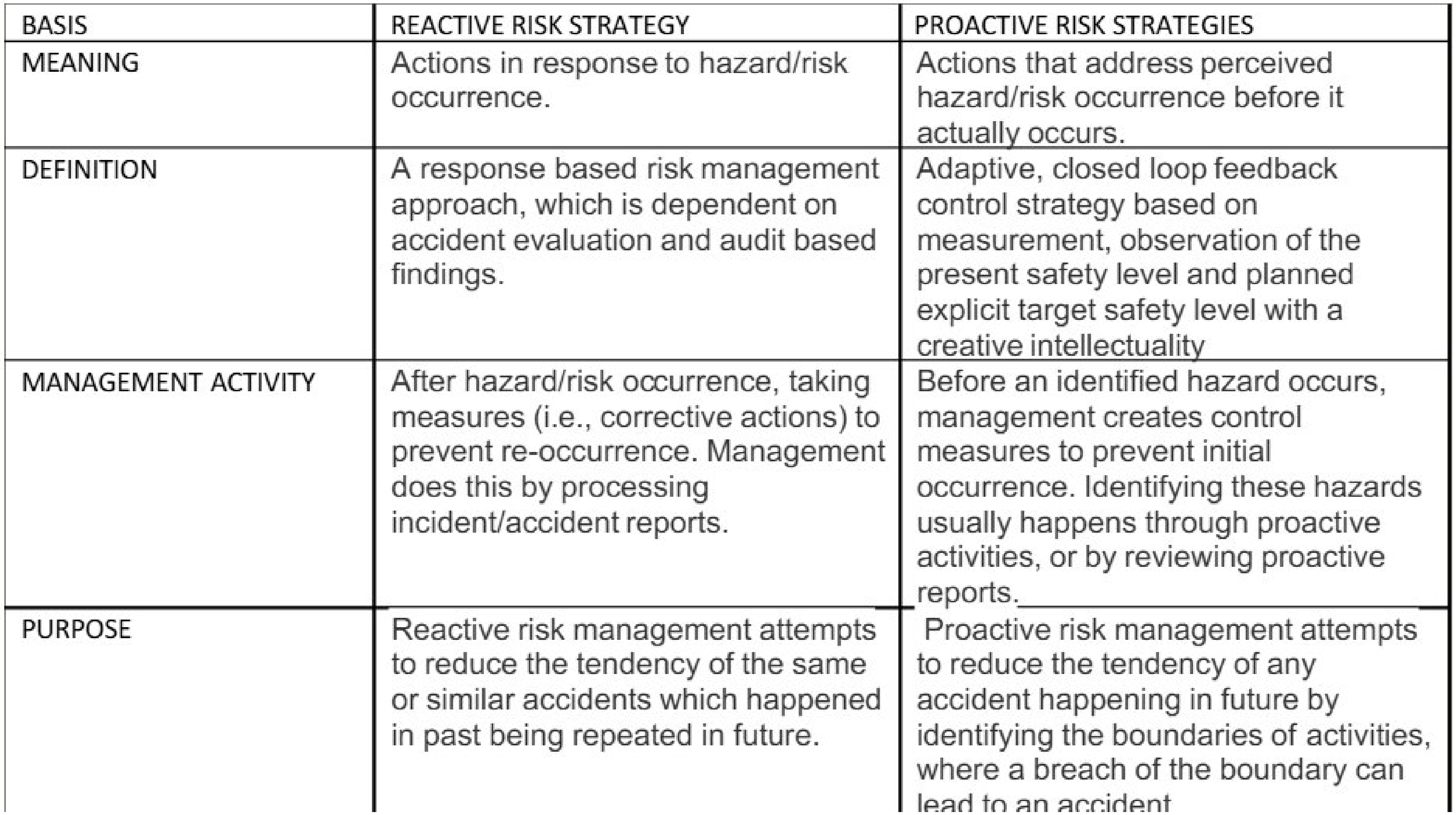
- In Proactive strategy,
  - risks are identified well in advance.
  - their probability and impact are analysed, and
  - risks are ranked by importance.

- The primary objective is to avoid risk.

- As part of Proactive strategies the below are the steps to be followed in Risk Management

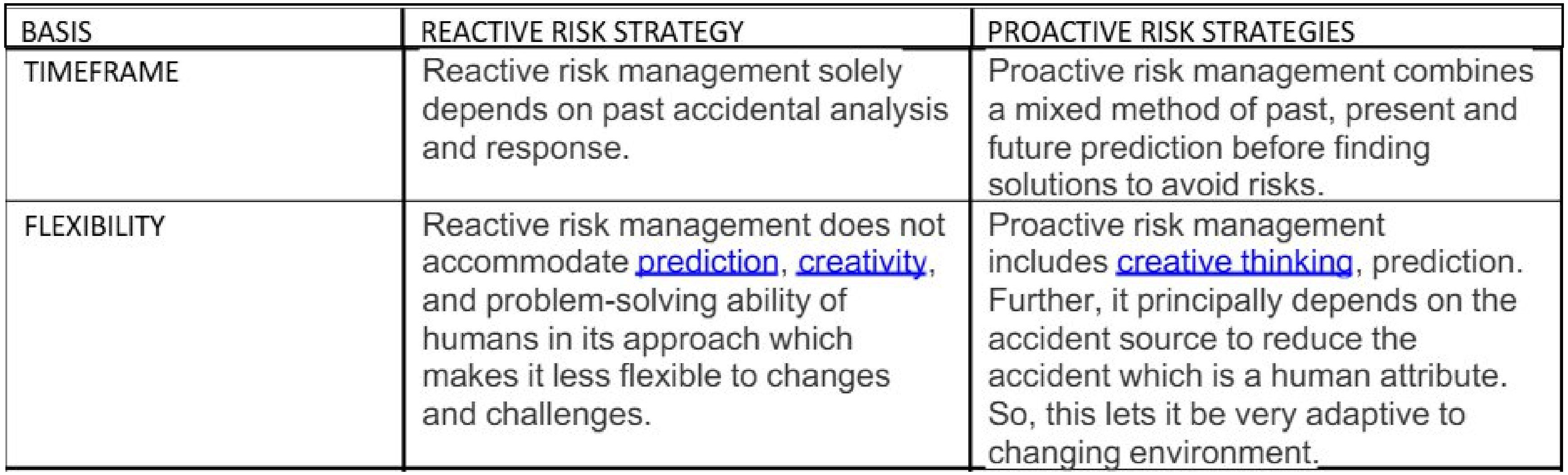
  - Risk Identification
  - Risk Projection
  - Risk Refinement
  - Risk Mitigation, Monitoring, Management

REACTIVE VERSUS PROACTIVE RISK STRATEGIES



BASIS	REACTIVE RISK STRATEGY	PROACTIVE RISK STRATEGIES
MEANING	Actions in response to hazard/risk occurrence.	Actions that address perceived hazard/risk occurrence before it actually occurs.
DEFINITION	A response based risk management approach, which is dependent on accident evaluation and audit based findings.	Adaptive, closed loop feedback control strategy based on measurement, observation of the present safety level and planned explicit target safety level with a creative intellectuality
MANAGEMENT ACTIVITY	After hazard/risk occurrence, taking measures (i.e., corrective actions) to prevent re-occurrence. Management does this by processing incident/accident reports.	Before an identified hazard occurs, management creates control measures to prevent initial occurrence. Identifying these hazards usually happens through proactive activities, or by reviewing proactive reports.
PURPOSE	Reactive risk management attempts to reduce the tendency of the same or similar accidents which happened in past being repeated in future.	Proactive risk management attempts to reduce the tendency of any accident happening in future by identifying the boundaries of activities, where a breach of the boundary can lead to an accident

REACTIVE VERSUS PROACTIVE RISK STRATEGIES



BASIS	REACTIVE RISK STRATEGY	PROACTIVE RISK STRATEGIES
TIMEFRAME	Reactive risk management solely depends on past accidental analysis and response.	Proactive risk management combines a mixed method of past, present and future prediction before finding solutions to avoid risks.
FLEXIBILITY	Reactive risk management does not accommodate prediction, creativity, and problem-solving ability of humans in its approach which makes it less flexible to changes and challenges.	Proactive risk management includes creative thinking, prediction. Further, it principally depends on the accident source to reduce the accident which is a human attribute. So, this lets it be very adaptive to changing environment.

SOFTWARE RISKS

- PROJECT RISKS
  - Affects project schedule, stakeholder trust, personnel issues
- TECHNICAL RISKS
  - Affects the quality of the software
  - Design, implementation, maintainence issues
- BUSSINESS RISKS
  - Building a software that no longer meets present technical requirements
  - Software which sales team are not aware of how to market.
  - Losing the support of senior management team
  - Losing budget or commitments
- KNOWN RISKS
- PREDICTABLE RISKS
- UNPREDICTABLE RISKS

RISK IDENTIFICATION

- Risk identification is a systematic process of identifying threats in a software process.
- By identifying **known and predictable risks**, the project manager takes a first step toward avoiding them when possible and controlling them when necessary.
- There are two distinct types of risks for every project
    - Generic risk
    - Product specific risk
- To identify product specific risk there is a checklist where system can be analysed.

RISK IDENTIFICATION

CHECKLIST FOR RISK IDENTIFICATION

- Product size
- Business impact
- Stakeholder characteristics
- Process definition
- Development environment
- Technology to be built
- Staff size and experience

- Questions relevant to each of the topics can be answered and documented for the software project. The answers to these questions allows us to estimate the **impact of risk.**

RISK IDENTIFICATION

Assessing Overall Project Risk

The following questions have been derived from risk data obtained by surveying experienced software project managers in different parts of the world.

1. Have top software and customer managers formally committed to support the project?
2. Are end users enthusiastically committed to the project and the system/product to be built?
3. Are requirements fully understood by the software engineering team and its customers?
4. Have customers been involved fully in the definition of requirements?

RISK IDENTIFICATION

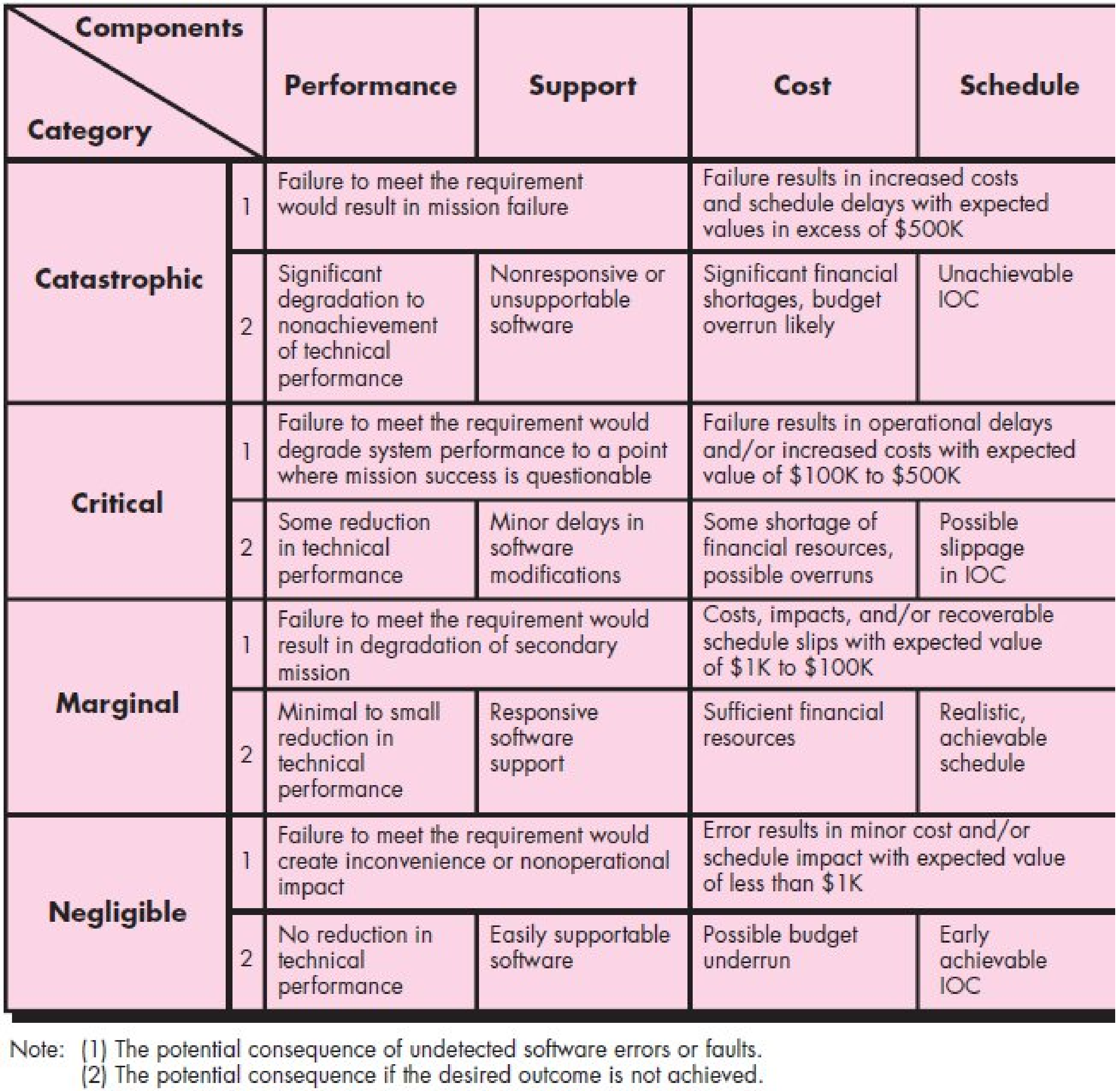
5. Do end users have realistic expectations?
6. Is the project scope stable?
7. Does the software engineering team have the right mix of skills?
8. Are project requirements stable?
9. Does the project team have experience with the technology to be implemented?
10. Is the number of people on the project team adequate to do the job?
11. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

RISK IDENTIFICATION

Risk Components and Drivers

- The U.S. Air Force has published a set of risk drivers that a project manager would identify that affect a set of components as below
  - ***Performance risk***—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
  - ***Cost risk***—the degree of uncertainty that the project budget will be maintained.
  - ***Support risk***—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
  - ***Schedule risk***—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

- **The impact of each risk driver on the risk component is divided into one of four impact categories—negligible, marginal, critical, or catastrophic.**



Category \ Components		Performance	Support	Cost	Schedule
Catastrophic	1	Failure to meet the requirement would result in mission failure		Failure results in increased costs and schedule delays with expected values in excess of $500K	
	2	Significant degradation to nonachievement of technical performance	Nonresponsive or unsupportable software	Significant financial shortages, budget overrun likely	Unachievable IOC
Critical	1	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		Failure results in operational delays and/or increased costs with expected value of $100K to $500K	
	2	Some reduction in technical performance	Minor delays in software modifications	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts, and/or recoverable schedule slips with expected value of $1K to $100K	
	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic, achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or nonoperational impact		Error results in minor cost and/or schedule impact with expected value of less than $1K	
	2	No reduction in technical performance	Easily supportable software	Possible budget underrun	Early achievable IOC

Note: (1) The potential consequence of undetected software errors or faults.
(2) The potential consequence if the desired outcome is not achieved.

RISK PROJECTION

- *Risk projection*, also called ***risk estimation***, attempts to rate each risk in two ways
  - The likelihood or probability that the risk is real.
  - The consequences of the problems associated with the risk.

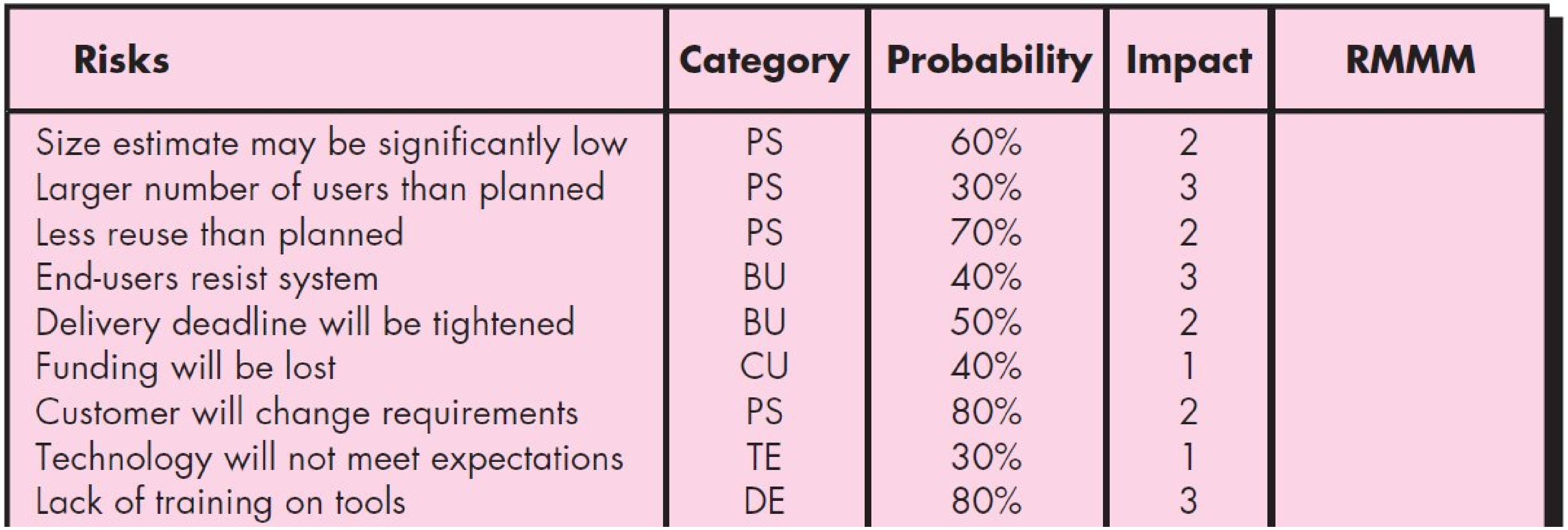
- **RISK PROJECTION STEPS**

  - Establish a scale that reflects the perceived likelihood of a risk
  - Delineate the consequences of the risk
  - Estimate the impact of the risk on the project and the product.
  - Assess the overall accuracy of the risk projection so that there will be no
  - misunderstandings

RISK PROJECTION

Developing a Risk Table

- It consists of columns Risks, category, probability, impact, RMMM.



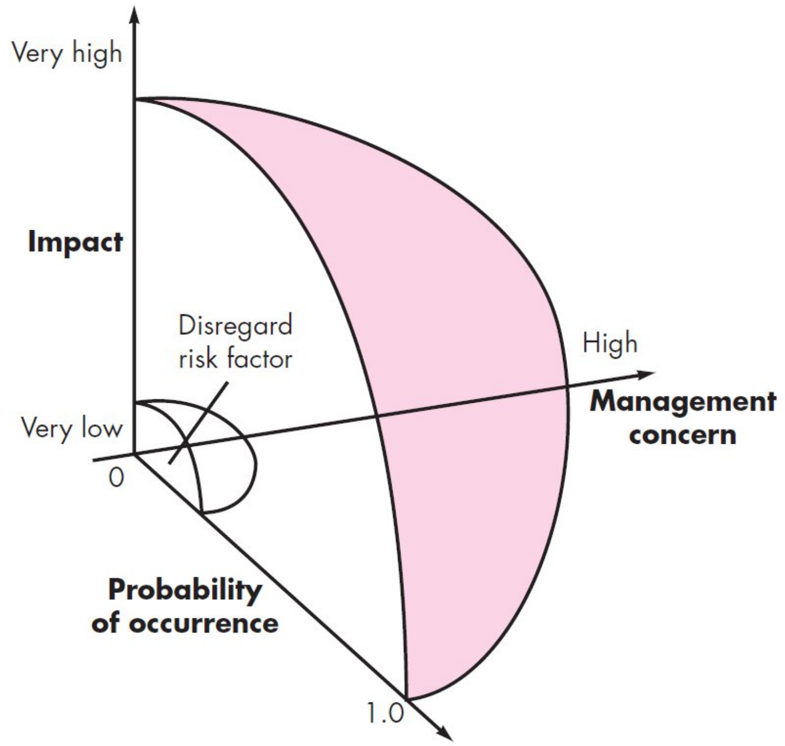
Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End-users resist system	BU	40%	3	
Delivery deadline will be tightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	

- PS – Project size risk , BU – business risk
- CU - Customer risk, TE – Technical risk, DE – Dev environment risk

Impact values:
1—catastrophic
2—critical
3—marginal
4—negligible

RISK PROJECTION

- Risk impact and probability can be plotted in a view along with Management concern as **RISK EXPOSURE GRAPH** below.



RISK PROJECTION

- **Assessing the Risk Impact**
- Three factors affect the consequences that are likely if a risk does occur: its nature, its scope, and its timing
- The following steps to determine the overall consequences of a risk:
  - determine the average probability of occurrence value for each risk component;
  - determine the impact for each component
  - complete the risk table and analyze the results

- The overall *risk exposure* RE is determined using the following relationship

$$RE = P \ x \ C$$

- where , *P* is the probability of occurrence for a risk, and
- *C* is the cost to the project if the risk occur.

RISK REFINEMENT

- It is a process of revisiting the risk and change its table as the project progresses.
- This can be done by representing the risk in *condition-transition-consequence* (CTC) format.
- Given that <condition> then there is concern that (possibly) <consequence>.

RISK MITIGATION, MONITORING, AND MANAGEMENT

UNIT V – PART II QUALITY MANAGEMENT

Software Quality - An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.

- Garvin's Quality Dimensions
- McCall's Quality Factors
- ISO 9126 Quality Factors
- Targeted Quality Factors

SOFTWARE QUALITY

Garvin’s Quality Dimensions

David Garvin suggests that quality should be considered by taking a multidimensional viewpoint that begins with an assessment of conformance and terminates with an aesthetic view

- Performance quality
- Feature quality.
- Reliability
- Conformance
- Durability.
- Serviceability.
- Aesthetics
- Perception

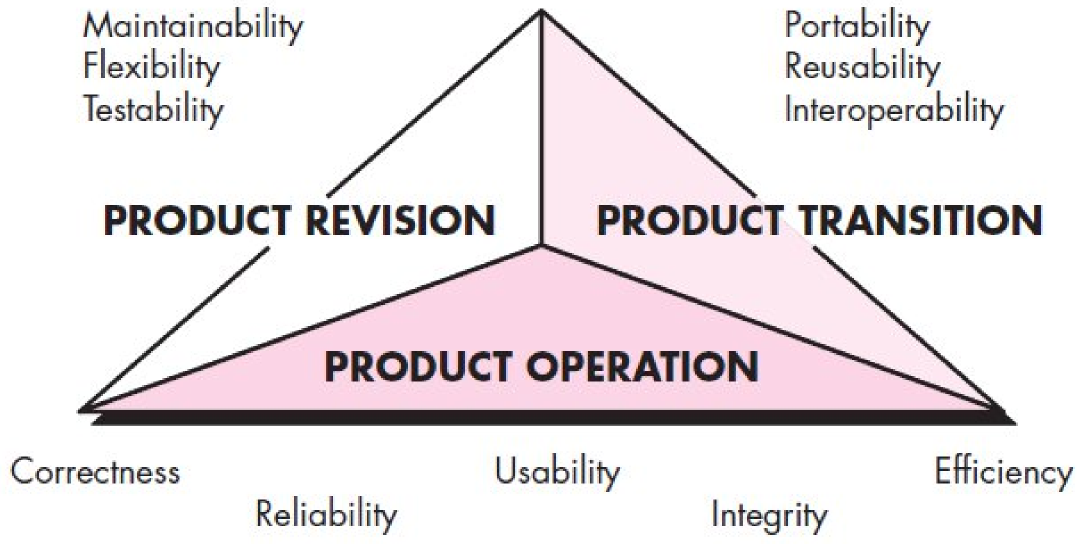
SOFTWARE QUALITY

McCall’s Quality Factors

McCall, Richards, and Walters propose a useful categorization of factors that affect software quality.

FIGURE 14.1

McCall’s software quality factors



SOFTWARE QUALITY

- The ISO 9126 standard was developed in an attempt to identify the key quality attributes for computer software.
- The standard identifies six key quality attributes:

1. Functionality
2. Reliability
3. Usability
4. Efficiency
5. Maintainability
6. Portability

SOFTWARE QUALITY

Targeted Quality Factors

- To conduct our assessment over any product, we need to address specific, measurable attributes of the interface.
- **Intuitiveness.** The degree to which the interface follows expected usage patterns so that even a novice can use it without significant training.
- **Efficiency.** The degree to which operations and information can be located or initiated.
- **Robustness.** The degree to which the software handles bad input data or inappropriate user interaction.
- **Richness.** The degree to which the interface provides a rich feature set.

SOFTWARE REVIEW

- A software review is a systematic examination of a software product to identify defects, ensure quality, and verify that it meets requirements.
- It is a crucial part of the software development process, involving different parties like developers, managers, and users who inspect software documents, code, and functionality.

INFORMAL REVIEWS

- Informal reviews include a simple desk check of a software engineering work product with a colleague.

- A simple *desk check* or a *casual meeting* conducted with a colleague is a review.

INFORMAL SOFTWARE REVIEW

- There is no advance planning or preparation, no agenda or meeting structure, and no follow-up on the errors that are uncovered.
- But a simple desk check can and does uncover errors that might otherwise propagate further into the software process.
- To improve the efficacy of a desk check review ,develop a set of simple review checklists for each major work product produced by the software team.

Example :

- Is the layout designed using standard conventions? Left to right? Top to bottom?
- Are all navigation choices clearly labeled?

FORMAL TECHNICAL REVIEWS

- A *formal technical review* (FTR) is a software quality control activity performed by software engineers.

- The objectives of an FTR are

- To uncover errors in function, logic, or implementation for any representation of the software.
- To verify that the software under review meets its requirements
- To ensure that the software has been represented according to predefined standards.
- To achieve software that is developed in a uniform manner.
- To make projects more manageable

FORMAL TECHNICAL REVIEWS

THE REVIEW MEETING

- **Guidelines:**
  - Three and five people should be involved in the review.
  - Advance preparation should occur but should require no more than two hours of work for each person.
  - The duration of the review meeting should be less than two hours.
- FTR focuses on a specific part of the overall software
- The main agenda of the FTR is on a work product.

- ***Producer*** *:* The individual who has developed the work product informs the project leader that the work product is complete and that a review is required.
- **Review Leader :** The project leader contacts a *review leader,* who evaluates the product for readiness, generates copies of product materials, and distributes them to two or three *reviewers* for advance preparation

FORMAL TECHNICAL REVIEWS

- Each reviewer is expected to spend between one and two hours reviewing the product, making notes, and otherwise becoming familiar with the work.

- Review leader, all reviewers, and the producer will attend review meeting.

- One of the reviewers takes on the role of a *recorder who will identify* ***MOM(minutes of meeting)***

- At the end of the review, all attendees of the FTR must decide whether to:
  - Accept the product without further modification
  - Reject the product due to severe errors
  - Accept the product provisionally
  - All FTR attendees complete a sign-off, indicating their participation in the review and their concurrence with the review team's findings.

FORMAL TECHNICAL REVIEWS

Review Guidelines

- Review the product, not the producer.
- Set an agenda and maintain it.
- Limit debate and rebuttal.
- Enunciate problem areas, but don’t attempt to solve every problem noted.
- Take written notes.
- Limit the number of participants and insist upon advance preparation
- Develop a checklist for each product that is likely to be reviewed.
- Allocate resources and schedule time for FTRs
- Conduct meaningful training for all reviewers
- Review your early reviews.

SOFTWARE QUALITY ASSURANCE

- SQA is an important to ensure the long life of any software in the industry

- The goal of SQA is to achieve quality in Requirements, Design, Code, Testing and deployment.

- The quality of a software can be measured in terms of Quality Attributes using many indices and vertices.

STATISTICAL SOFTWARE QUALITY ASSURANCE

To represent the quality of a software there are certain measures and metrics calculated statistically

STATISTICAL SOFTWARE QUALITY ASSURANCE

Statistical SQA implies the following steps.

- Information about software errors and defects is collected and categorized.
- Each error is traced and defect to its underlying cause is determined.
- Using the Pareto principle (80 percent of the defects can be traced to 20 percent of all possible causes), isolate the 20 percent (the *vital few*).
- Once the vital few causes have been identified, move to correct the problems that have caused the errors and defects.

STATISTICAL SOFTWARE QUALITY ASSURANCE

- All the errors that are encountered in a software can be mapped to one or more of the below causes.
  - Incomplete or erroneous specifications (IES)
  - Misinterpretation of customer communication (MCC)
  - Intentional deviation from specifications (IDS)
  - Violation of programming standards (VPS)
  - Error in data representation (EDR)
  - Inconsistent component interface (ICI)
  - Error in design logic (EDL)
  - Incomplete or erroneous testing (IET)
  - Inaccurate or incomplete documentation (IID)
  - Error in programming language translation of design (PLT)
  - Ambiguous or inconsistent human/computer interface (HCI)
  - Miscellaneous (MIS)