DOI: https://doi.org/10.53555/nnmce.v1i1.380 **Publication URL: http://nnpub.org/index.php/MCE/article/view/380**

RISK MANAGEMENT IN SOFTWARE PROJECTS WITH MICROCONTROLLER

Nilesh Kumar[1]*, Deepak Shaharan[2], Nishant Kumar[3]

Corresponding Author: -

Abstract: -
The main motive to present this paper is to provide a secure management in software with the help of sense-controlled device microcontroller This paper reviews the basic concepts, terminology, and techniques of Software Risk Management. It teaches readers how to identify and analyse software risks on their projects. Readers then learn techniques for planning indicting to mitigate risks so that the overall impact of those risks on their projects is minimized.

1. INTRODUCTION

Risk management is an action that helps a software development team to understand what kinds of risks are there in software development. Risk is always concern with today's and yesterday's uncertainty. It is a potential problem. So, it might happen, it might not. It is better to identify its probability of occurrence.

1.defining software risk management

There are many risks involved in creating high quality software on time and within budget. However, in order for it to be worthwhile to take these risks, they must be compensated for by a perceived reward. The greater the risk, the greater the reward must be to make it worthwhile to take the chance. In software development, the possibility of reward is high, but so is the potential for disaster. The need for software risk management is illustrated in Gilb's risk principle. "If you don't actively attack the risks, they will actively attack you" [Gilb-88]. In order to successfully manage a software project and reap our rewards, we must learn to identify, analyze, and control these risks. This paper focuses on the basic concepts, processes, and techniques of software risk management.

There are basic risks that are generic to almost all software projects. Although there is a basic component of risk management inherent in good project management, risk management differs from project management in the following.

Project Management	Risk Management
Designed to address general or generic risks	Designed to focus on risks unique to each
	project
Looks at the big picture and plans for details	Looks at potential problems and plans for contingencies
Plans what should happen and looks for ways to make it happen	Evaluates what could happen and looks for ways to minimize the damage
Plans for success	Plans to manage and mitigate potential causes of failure

3. Risk Management Process with microcontroller

This process starts with the identification of a list of potential risks. Each of these risks is then analyzed and prioritized. A risk management plan is created that identifies containment actions that will reduce the probability of the risk occurring and/or reduce the impact if the risk turns into a problem. The plan also includes contingency \actions that will be taken if the risk turns into a problem and the associated triggers (indicators that the risk is turning into a problem). The containment part of the plan is then implemented and actions are taken. The tracking step involves monitoring the status of known risks as well as the results of risk reduction actions. If a trigger indicates the onset of a problem, the corresponding contingency plans are implemented. As new status and information are obtained, the risk management plans are updated accordingly. Tracking may also result in the addition of newly identified risks or in the closure of known risks. Risk management process is on-going part of managing the software building. It si designed to be a continuous feedback loop where additional information and risk status are utilized to refine the project's risk list and risk management plans.

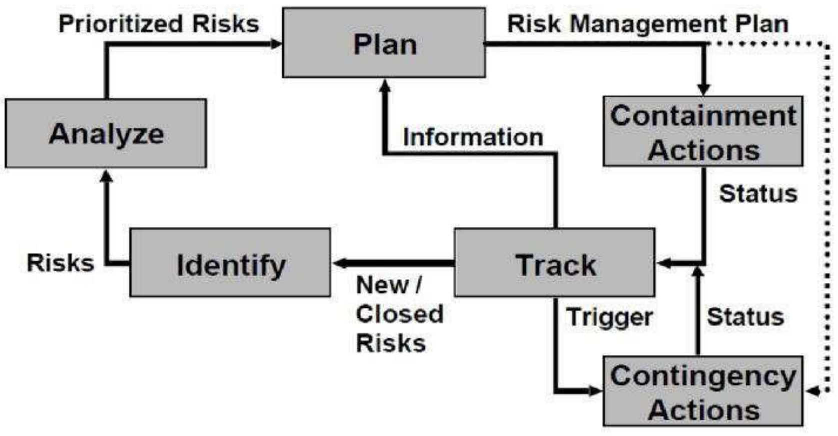


Figure 1 - Risk Management Process

A **microcontroller** (sometimes abbreviated **μC**, **uC** or **MCU**) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.
Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.
Some microcontrollers may use four-bit words and operate at clock rate frequencies as low as 4 kHz, for low power consumption (single-digit milliwatts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

Risk Identification
During the first step in the software risk management process, risks are identified and added to the list of known risks. The output of this step is a list of project-specific risks that have the potential of compromising the project's success. There are many techniques for identifying risks, including interviewing, reporting, decomposition, assumption analysis, critical path analysis, and utilization of risk taxonomies.

Interviewing/Brainstorming: One technique for identifying risks is interviewing or brainstorming with project personnel, customers, and vendors. Open-ended questions such as the following can help identify potential areas of risk.
What new or improved technologies does this project implement?
What interfaces issues still need to be defined?
What requirements exist that we aren't sure how to implement?
What concerns do we have about our ability to meet the required quality and performance levels?

Voluntary Reporting: Another risk identification technique is voluntary reporting, where any individual who identifies a risk is encouraged and rewarded for bringing that risk to management's attention. This requires the complete elimination of the "shoot the messenger" syndrome. It avoids the temptation to assign risk reduction actions to the person who identified the risk. Risks can also be identified through required reporting mechanisms such as status reports or project reviews.

Decomposition: As the product is being decomposed during the requirements and design phases, another opportunity exists for risk identifications. Every TBD ("To Be Done/Determined") is a potential risk. As Ould states, "The most important thing about planning is writing down what you *don't know*, because what you don't know is what you must find out" [Ould-90]. Decomposition in the form of work breakdown structures during project planning can also help identify areas of uncertainty that may need to be recorded as risks.

Assumption Analysis: Process and product assumptions must be analyzed. For example, we might assume the hardware would be available by the system test date or three additional experienced C++ programmers will be hired by the time coding starts. If these assumptions prove to be false, we could have major problems.

Critical Path Analysis: As we perform critical path analysis for our project plan, we must remain on the alert to identify risks. Any possibility of schedule slippage on the critical path must be considered a risk because it directly impacts our ability to meet schedule.

Risk Taxonomies: Risk taxonomies are lists of problems that have occurred on other projects and can be used as checklists to help ensure all potential risks have been considered. An example of a risk taxonomy can be found in the

Risk Analysis

During the risk analysis step, each risk is assessed to determine: Likelihood: the probability that the risk will result in a loss

Impact: the size or cost of that loss if the risk turns into a problem

Timeframe: when the risk needs to be addressed (i.e., risk associated with activities in the near future would have a higher priority than similar risks in later activities) Additionally, the interrelationships between risks are assessed to determine if compounding risk conditions magnify losses.

Risk Projection

Risk identification is also known as risk estimation, attempts to rate each risk in two ways –

(1) Probability that the risk is real.

(2) The consequences of the problem associated with the risk, should it occur. You always work along with other managers and technical staff to perform four

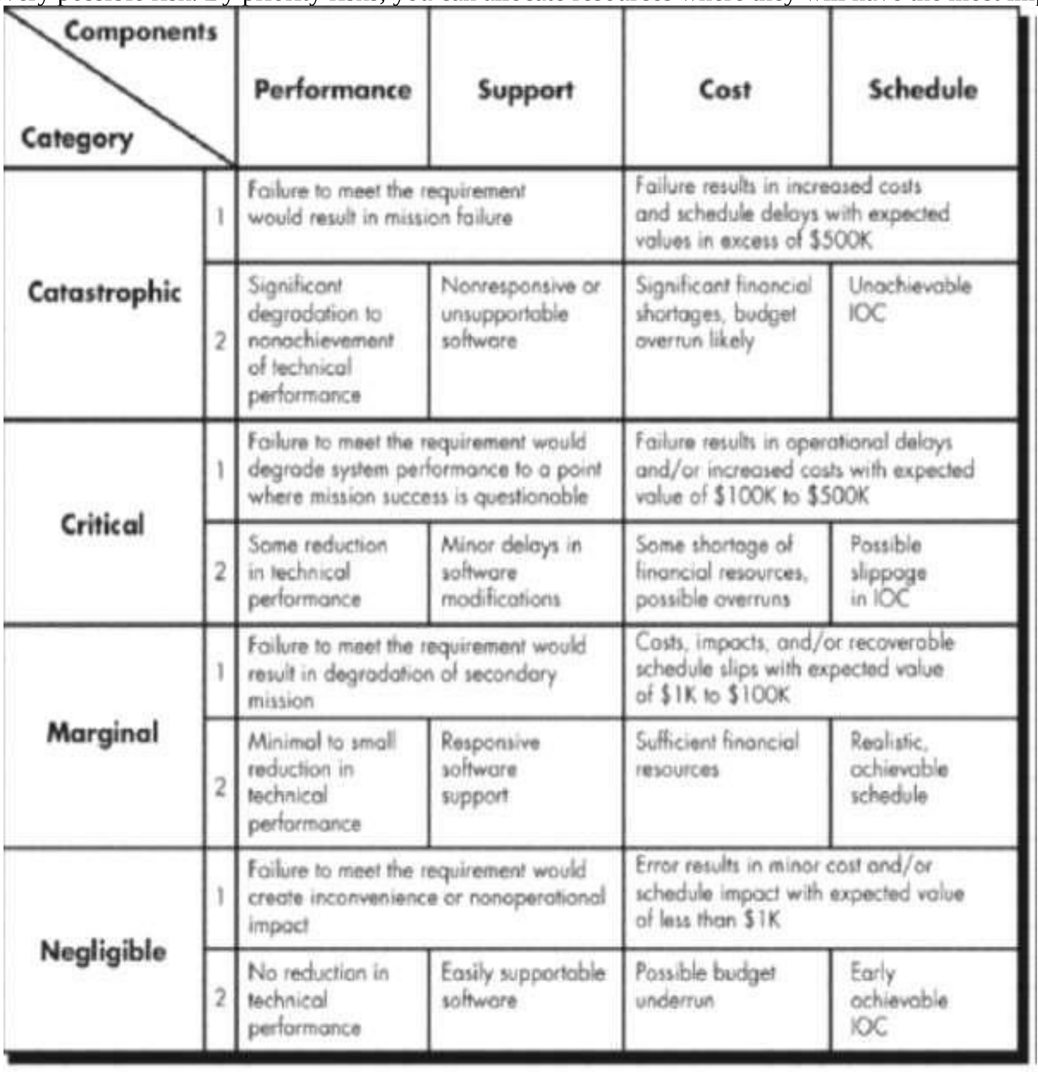
(3) risk projection steps:

Establish a scale that reflects the perceived probability of a risk. Delineate the consequences of the risk.

Estimate the impact of the risk on the projection and the product.

Assess the overall accuracy of the risk estimation so that there will be no misunderstanding.

The intention of these steps is to consider risks in a manner that leads to give priority. No software team has the resourses to address every possible risk. By priority risks, you can allocate resources where they will have the most impact.



Components / Category		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.

Conclusions

With ever-increasing complexity and increasing demand for bigger, better, and faster, the software industry is a highrisk business. When teams don't manage risk, they leave projects vulnerable to factors that can cause major rework,major cost or schedule over-runs, or complete project failure. Adopting a Software Risk Management Program is a step every software manager can take to more effectively manage software development initiatives. Risk management is an ongoing process that is implemented as part of the initial project planning activities and utilized throughout all of the phases of the software development lifecycle. Risk management requires a fear-free environment where risks can be identified and discussed openly. Based on a positive, proactive approach, risk management can greatly reduce or even eliminate the need for crisis management within our software projects.

References

[1].[Boehm-89] Barry W. Boehm, Tutorial: Software Risk Management, Les Alamitos, CA, IEEE Computer Society,1989.

[2].[Down-94] Alex Down, Michael Coleman, Peter Absolon, Risk Management for Software Projects, London,McGraw-Hill Book Company, 1994.

[3].[Ould-90] Martyn Ould, *Strategies For Software Engineering: The Management of Risk and Quality,*Chichester, England, John Wiley & Sons, 1990.

[4].[SEI-93] Marvin J. Carr, Suresh L. Konda, Ira Monarch, F. Carol Ulrich, Clay F. Walker, Taxonomy –Based Risk Identification, CMU/SEI-93-TR-006, Pittsburgh, PA, Software Engineering Institute, 1993.