**RISK IDENTIFICATION**

Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.). 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 each of the categories that have been presented generic risks and product-specific risks. Generic risks are a potential threat to every software project. Product-specific risks can be identified only by those with a clear understanding of the technology, the people, and the environment that is specific to the project at hand. To identify product-specific risks, the project plan and the software statement of scope are examined and an answer to the following question is developed: "What special characteristics of this product may threaten our project plan?"

One method for identifying risks is to create a risk item checklist. The checklist can be used for risk identification and focuses on some subset of known and predictable risks in the following generic subcategories:

•Product size—risks associated with the overall size of the software to be built or modified.

• Business impact—risks associated with constraints imposed by management or the marketplace.

• Customer characteristics—risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.

• Process definition—risks associated with the degree to which the software process has been defined and is followed by the development organization.

• Development environment—risks associated with the availability and quality of the tools to be used to build the product.

• Technology to be built—risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.

• Staff size and experience—risks associated with the overall tech.

### **Risk Assessment :**

Risk assessment simply means to describe the overall process or method to identify risk and problem factors that might cause harm. It is actually a systematic examination of a task or project that you perform to simply identify significant risks, problems, hazards, and then to find out control measures that you will take to reduce risk. The best approach is to prepare a set of questions that can be answered by project managers in order to assess overall project risks.

These questions are shown below:

* Will the project get proper support from the customer manager?
* Are end-users committed to software that has been produced?
* Is there a clear understanding of the requirements?
* Is there an active involvement of customers in the requirement definition?
* Are the expectations set for the product are realistic?
* Is project scope stable?
* Are there team members with the required skills?
* Are project requirements stable?
* Does technology used for software is known to developers?
* Is the size of the team sufficient to develop the required product?
* Is that all customers know the importance of the product/requirements of the system to be built?

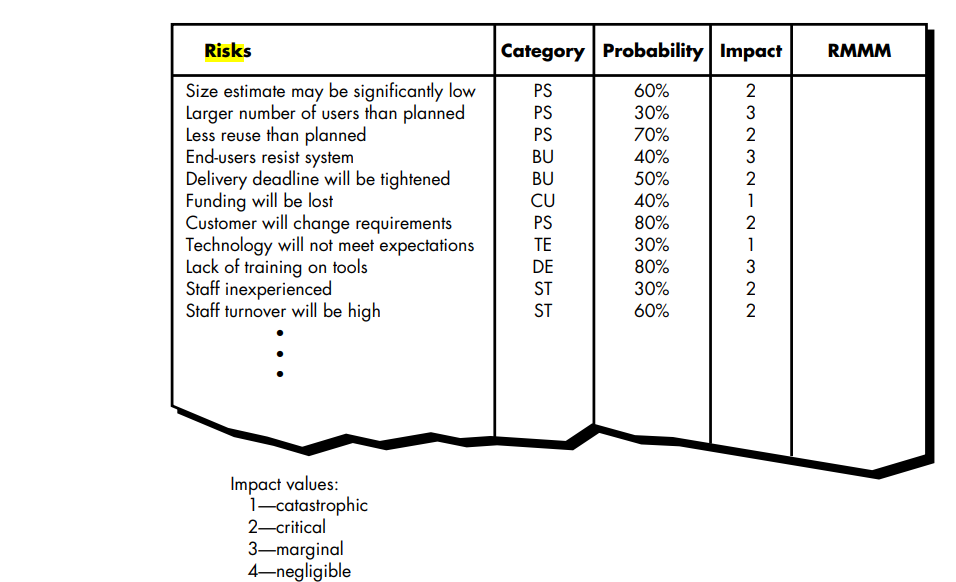
Thus, the number of negative answers to these questions represents the severity of the impact of risk on the overall project. It is not about creating or making a large number of work papers, but rather simply identify and find out measures to control risks in your workplace.

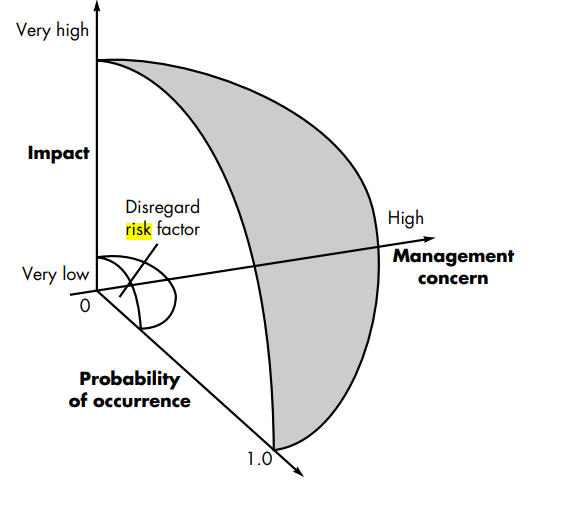
**RISK PROJECTION**

Risk projection, also called risk estimation, attempts to rate each risk in two ways—the likelihood or probability that the risk is real and the consequences of the problems associated with the risk, should it occur. The project planner, along with other managers and technical staff, performs four risk projection activities: (1) establish a scale that reflects the perceived likelihood of a risk, (2) delineate the consequences of the risk, (3) estimate the impact of the risk on the project and the product, and (4) note the overall accuracy of the risk projection so that there will be no misunderstandings.

Developing a Risk Table

A risk table provides a project manager with a simple technique for risk projection.





A project team begins by listing all risks (no matter how remote) in the first column of the table. This can be accomplished with the help of the risk item checklists. Each risk is categorized in the second column (e.g., PS implies a project size risk, BU implies a business risk). The probability of occurrence of each risk is entered in the next column of the table. The probability value for each risk can be estimated by team members individually. Individual team members are polled in round-robin fashion until their assessment of risk probability begins to converge. Next, the impact of each risk is assessed. Each risk component is assessed using the characterization, and an impact category is determined. The categories for each of the four risk components—performance, support, cost, and schedule—are averaged to determine an overall impact value. Once the first four columns of the risk table have been completed, the table is sorted by probability and by impact. High-probability, high-impact risks percolate to the top of the table, and low-probability risks drop to the bottom. This accomplishes first-order risk prioritization. The project manager studies the resultant sorted table and defines a cutoff line. The cutoff line (drawn horizontally at some point in the table) implies that only risks that lie above the line will be given further attention. Risks that fall below the line are re-evaluated to accomplish second-order prioritization.

All risks that lie above the cutoff line must be managed. The column labeled RMMM contains a pointer into a Risk Mitigation, Monitoring and Management Plan or alternatively, a collection of risk information sheets developed for all risks that lie above the cutoff.

**RISK REFINEMENT**

During early stages of project planning, a risk may be stated quite generally. As time passes and more is learned about the project and the risk, it may be possible to refine the risk into a set of more detailed risks, each somewhat easier to mitigate, monitor, and manage.

One way to do this is to represent the risk in condition-transition-consequence (CTC) format. That is, the risk is stated in the following form: Given that then there is concern that (possibly).

Given that all reusable software components must conform to specific design standards and that some do not conform, then there is concern that (possibly) only 70 percent of the planned reusable modules may actually be integrated into the as-built system, resulting in the need to custom engineer the remaining 30 percent of components.

This general condition can be refined in the following manner:

Subcondition 1. Certain reusable components were developed by a third party with no knowledge of internal design standards.

Subcondition 2. The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.

Subcondition 3. Certain reusable components have been implemented in a language that is not supported on the target environment. The consequences associated with these refined subconditions remains the same (i.e., 30 percent of software components must be customer engineered), but the refinement helps to isolate the underlying risks and might lead to easier analysis and response.

**Risk Mitigation, Monitoring, and Management (RMMM) plan**

[RMMM Plan :](https://www.geeksforgeeks.org/short-note-on-risk-assessment-and-risk-mitigation/)   
A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM). In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan.

In some software teams, risk is documented with the help of a Risk Information Sheet (RIS). This RIS is controlled by using a database system for easier management of information i.e creation, priority ordering, searching, and other analysis. After documentation of RMMM and start of a project, risk mitigation and monitoring steps will start.

**Risk Mitigation :**   
It is an activity used to avoid problems (Risk Avoidance).   
Steps for mitigating the risks as follows.

1. Finding out the risk.
2. Removing causes that are the reason for risk creation.
3. Controlling the corresponding documents from time to time.
4. Conducting timely reviews to speed up the work.

**Risk Monitoring :**   
It is an activity used for project tracking.   
It has the following primary objectives as follows.

1. To check if predicted risks occur or not.
2. To ensure proper application of risk aversion steps defined for risk.
3. To collect data for future risk analysis.
4. To allocate what problems are caused by which risks throughout the project.

**Risk Management and planning :**   
It assumes that the mitigation activity failed and the risk is a reality. This task is done by Project manager when risk becomes reality and causes severe problems. If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows that the response that will be taken for each risk by a manager. The main objective of the risk management plan is the risk register. This risk register describes and focuses on the predicted threats to a software project.

**Example:**

Let us understand RMMM with the help of an example of high staff turnover.

**Risk Mitigation:**

To mitigate this risk, project management must develop a strategy for reducing turnover. The possible steps to be taken are:

* Meet the current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market).
* Mitigate those causes that are under our control before the project starts.
* Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
* Organize project teams so that information about each development activity is widely dispersed.
* Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner.
* Assign a backup staff member for every critical technologist.

**Risk Monitoring:**

As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of high staff turnover, the following factors can be monitored:

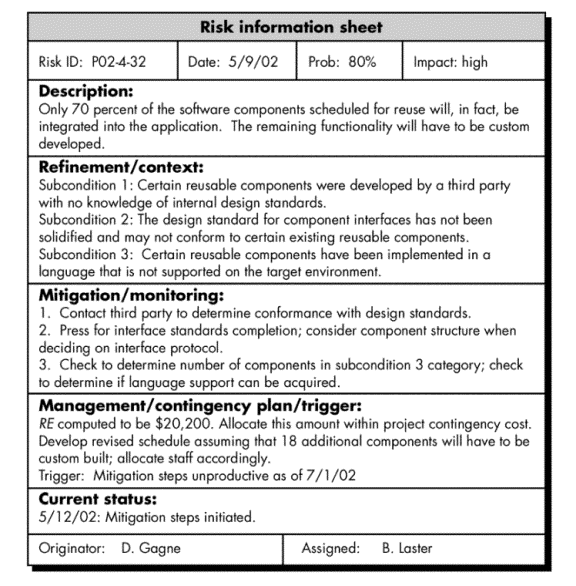
* General attitude of team members based on project pressures.
* Interpersonal relationships among team members.
* Potential problems with compensation and benefits.
* The availability of jobs within the company and outside it.

**Risk Management:**

Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality. Continuing the example, the project is well underway, and a number of people announce that they will be leaving. If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team. In addition, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to “get up to the speed**“.**

**Drawbacks of RMMM:**

* It incurs additional project costs.
* It takes additional time.
* For larger projects, implementing an RMMM may itself turn out to be another tedious project.
* RMMM does not guarantee a risk-free project, infact, risks may also come up after the project is delivered.



# Software Configuration Management

When we develop software, the product (software) undergoes many changes in their maintenance phase; we need to handle these changes effectively.

Several individuals (programs) works together to achieve these common goals. This individual produces several work product (SC Items) e.g., Intermediate version of modules or test data used during debugging, parts of the final product.

The elements that comprise all information produced as a part of the software process are collectively called a software configuration.

As software development progresses, the number of Software Configuration elements (SCI's) grow rapidly.

**These are handled and controlled by SCM. This is where we require software configuration management.**

A configuration of the product refers not only to the product's constituent but also to a particular version of the component.

Therefore, SCM is the discipline which

* Identify change
* Monitor and control change
* Ensure the proper implementation of change made to the item.
* Auditing and reporting on the change made.

Configuration Management (CM) is a technic of identifying, organizing, and controlling modification to software being built by a programming team.

**The objective is to maximize productivity by minimizing mistakes (errors).**

CM is used to essential due to the inventory management, library management, and updation management of the items essential for the project.

## **Why do we need Configuration Management?**

Multiple people are working on software which is consistently updating. It may be a method where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently. It changes in user requirements, and policy, budget, schedules need to be accommodated.

## **Importance of SCM**

It is practical in controlling and managing the access to various SCIs e.g., by preventing the two members of a team for checking out the same component for modification at the same time.

**It provides the tool to ensure that changes are being properly implemented.**

It has the capability of describing and storing the various constituent of software.

SCM is used in keeping a system in a consistent state by automatically producing derived version upon modification of the same component.

# SCM Process

It uses the tools which keep that the necessary change has been implemented adequately to the appropriate component. The SCM process defines a number of tasks:

* Identification of objects in the software configuration
* Version Control
* Change Control
* Configuration Audit
* Status Reporting

**Identification**

**Basic Object:** Unit of Text created by a software engineer during analysis, design, code, or test.

**Aggregate Object:** A collection of essential objects and other aggregate objects. Design Specification is an aggregate object.

Each object has a set of distinct characteristics that identify it uniquely: a name, a description, a list of resources, and a "realization."

**The interrelationships between configuration objects can be described with a Module Interconnection Language (MIL).**

**Version Control**

Version Control combines procedures and tools to handle different version of configuration objects that are generated during the software process.

**Clemm defines version control in the context of SCM:** Configuration management allows a user to specify the alternative configuration of the software system through the selection of appropriate versions. This is supported by associating attributes with each software version, and then allowing a configuration to be specified [and constructed] by describing the set of desired attributes.

**Change Control**

James Bach describes change control in the context of SCM is: Change Control is Vital. But the forces that make it essential also make it annoying.

We worry about change because a small confusion in the code can create a big failure in the product. But it can also fix a significant failure or enable incredible new capabilities.

We worry about change because a single rogue developer could sink the project, yet brilliant ideas originate in the mind of those rogues, and

A burdensome change control process could effectively discourage them from doing creative work.

A change request is submitted and calculated to assess technical merit; potential side effects, the overall impact on other configuration objects and system functions, and projected cost of the change.

The results of the evaluations are presented as a change report, which is used by a change control authority (CCA) - a person or a group who makes a final decision on the status and priority of the change.

The "check-in" and "check-out" process implements two necessary elements of change control-**access control** and **synchronization control**.

**Access Control** governs which software engineers have the authority to access and modify a particular configuration object.

**Synchronization Control** helps to ensure that parallel changes, performed by two different people, don't overwrite one another.

**Configuration Audit**

SCM audits to verify that the software product satisfies the baselines requirements and ensures that what is built and what is delivered.

SCM audits also ensure that traceability is maintained between all CIs and that all work requests are associated with one or more CI modification.

SCM audits are the "**watchdogs**" that ensures that the integrity of the project's scope is preserved.

**Status Reporting**

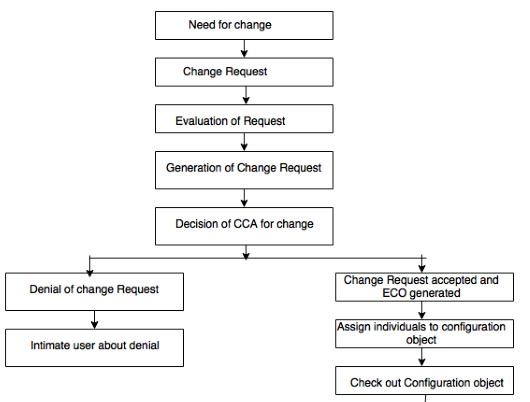
Configuration Status reporting (sometimes also called status accounting) providing accurate status and current configuration data to developers, testers, end users, customers and stakeholders through admin guides, user guides, FAQs, Release Notes, Installation Guide, Configuration Guide, etc.

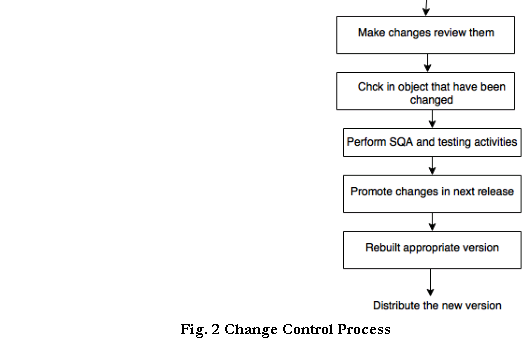
**Version Control**

* Version Control Combines procedures and tools to manage different version of configuration objects that are created during the software process.
* A version control system implements or is directly integrated with four major capabilities:
  1. A project database that stores all relevant configuration objects,
  2. A version management capability that stores all version of configuration object,
  3. A make facility that enables the software engineer to collect all relevant configuration objects, and
  4. Construct a specific version of the software.
* A number of version control systems establish a set – a collection of all changes (to some baseline configuration) that are required to create a specific version of the software.
* “Changes set” captures all changes to all files in the configuration along with reason for changes and details of who made the changes and when.
* A number of named change set can be identified for an application or system. This enables a software engineer to construct a version of the software by specifying the changes set (by name) that must be applied to the baseline configuration.
* To accomplish this, a system modelling approach is applied. The system model contains
  1. A template that include a component hierarchy and build order for the component that describe how the system must be constructed,
  2. Construction rules, and
  3. Verification rules.

**Change Control**

* Change control is manual step in software lifecycle. It combines human procedures and automated tools.
  + Change control process is illustrated in following figure 2.
  + Change request submitted and evaluated to assess technical merit, potential side effects, overall impact on other configuration object and system function, and project cost of change.
* The result of the evaluation are presented as a change report, which is used by the change control authority(CCA) – A person or group who make final decision on the status and priority of the change.
* An engineering change order (ECO) is generated for each approved change. The ECO describes the change order to be made, the constraints that must be respected, and the criteria for view and audit.
* The object to be changed can be placed in a directory that is controlled by software engineer making the change. As an alternative, the object to be changed can be “checked out” of the project database, change is made, and appropriate SQA activities are applied.
* The object are then “checked in” to the database and appropriate version control mechanism are used to create the next version of the software.
* Checked in and Checked out mechanism require two important elements
* Access Control
* Synchronization Control
  + The Access control mechanism gives the authority to the software engineer to access and modify the specific configuration object.
  + The Synchronization control mechanism allows to make parallel changes or the change made by two different people without overwriting each other’s work.





**Version Control** and **change control** system often implements an **issue tracking** (also called bug tracking) capability that enables the team to record and track the status of all outstanding issues associated with each configuration object.