**Unit 5: Risk Analysis and Management**

REACTIVE VS. PROACTIVE RISK STRATEGIES

Reactive risk strategies - worrying about problems when they happen

Proactive risk strategies - begin long before technical work is initiated

Potential risks are identiﬁed

their probability and impact are assessed they are ranked by importance

Not all risks can be avoided

Team works to develop a contingency plan that will enable it to respond in a controlled and effective manner

### SOFTWARE RISKS

Risk always involves two characteristics:

*Uncertainty—*the risk may or may not happen; that is, there are no 100% probable risks

*Loss—*if the risk becomes a reality, unwanted consequences or losses will occur

Must quantify:

level of uncertainty

degree of loss associated with each risk

## Kinds of Risks:

*Project risks* threaten the project plan.

Identify potential budgetary, schedule, personnel (staﬃng and organization), resource, customer, and requirements problems and their impact on a software project

*Technical risks* threaten the quality and timeliness of the software to be produced

Identify potential design, implementation, interface, veriﬁcation, and maintenance problems.

Speciﬁcation ambiguity, technical uncertainty, technical obsolescence, and "leading-edge" technology are risk factors. Technical risks occur because the problem is harder to solve than we thought it would be.

*Business risks* threaten the viability of the software to be built.

Top ﬁve business risks :

1. building an excellent product or system that no one really wants (market risk)
2. building a product that no longer ﬁts into the overall business strategy for the company (strategic risk)
3. building a product that the sales force doesn't understand how to sell
4. losing the support of senior management due to a change in focus or a change in people (management risk)
5. losing budgetary or personnel commitment (budget risks).

*Known risks* - uncovered after evaluation: project plan

business and technical environment in which the project is being developed

other reliable information sources (e.g., unrealistic delivery date, lack of documented requirements or software scope, poor development environment).

*Predictable risks -* extrapolated from past project experience e.g.,

* + staff turnover
  + poor communication with the customer
  + dilution of staff effort as ongoing maintenance requests are serviced

*Unpredictable risks* - occur, but diﬃcult to identify in advance.

### RISK IDENTIFICATION

* Attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.)
* Two types of risks:
  + *Generic risks* - potential threat to every software project.
  + *Product-speciﬁc risks* - identiﬁed by: those with a clear understanding of the technology, the people, and the environment that is speciﬁc to the project at hand.

§ Identify by looking at the project plan and the software statement of scope

§ Ask question - "What special characteristics of this product may threaten our project plan?"

## Risk item checklist

*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 technical and project experience of the software engineers who will do the work.

#### Short List

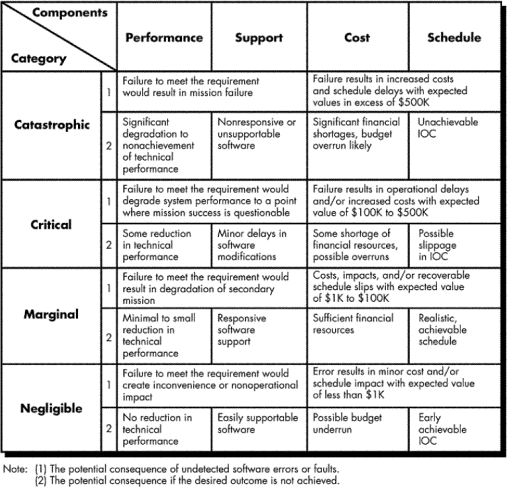
1. Have top software and customer managers formally committed to support the project?
2. Are end-users enthusiastically committed to the project and the system/product to be built?
3. Are requirements fully understood by the software engineering team and their customers?
4. Have customers been involved fully in the deﬁnition of requirements?
5. Do end-users have realistic expectations?
6. Is 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?

#### [Sample Risk Checklist](http://csis.pace.edu/~marchese/SE616/L10/APMrisktemplate.htm) (Long)

**Risk Components and Drivers**

* U.S. Air Force pamphlet contains guidelines for software risk identiﬁcation and abatement.
* Air Force approach requires that the project manager identify the risk drivers that affect software risk components—performance, cost, support, and schedule.
* Risk components:
  1. *Performance risk—*the degree of uncertainty that the product will meet its requirements and be ﬁt for its intended use.
  2. *Cost risk—*the degree of uncertainty that the project budget will be maintained.
  3. *Support risk—*the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
  4. *Schedule risk—*the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.
* Impact of each risk driver on the risk component is divided into one of four impact categories—negligible, marginal, critical, or catastrophic

#### Figure 1 - Risk Characterization Table





RISK PROJECTION (AKA RISK ESTIMATION)

Attempts to rate each risk in two ways

* The probability that the risk is real
* The consequences of the problems associated with the risk, should it occur.

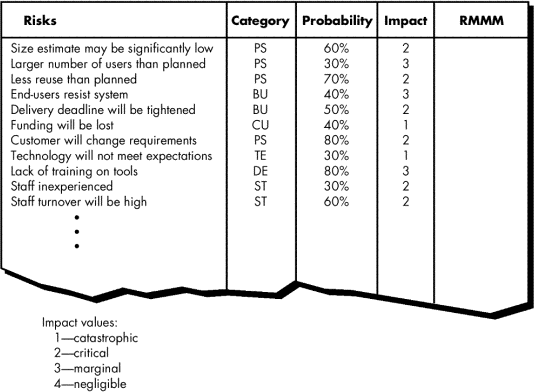
Project planner, along with other managers and technical staff, performs four risk projection activities:

1. establish a measure 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
4. note the overall accuracy of the risk projection so that there will be no misunderstandings.

#### Developing a Risk Table

Risk table provides a project manager with a simple technique for risk projection

#### Figure 2 - Risk Table



**Steps in Setting up Risk Table**

1. Project team begins by listing all risks in the ﬁrst column of the table.

Accomplished with the help of the risk item checklists.

1. Each risk is categorized in the second column

(e.g., PS implies a project size risk, BU implies a business risk).

1. 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.

1. Individual team members are polled in round-robin fashion until their assessment of risk probability begins to converge.

#### Assessing Impact of Each Risk

1. Each risk component is assessed using the Risk Charcterization Table (Figure 1) and impact category is determined.
2. Categories for each of the four risk components—performance, support, cost, and schedule—are averaged to determine an overall impact value.
   1. A risk that is 100 percent probable is a constraint on the software project.
   2. The risk table should be implemented as a spreadsheet model. This enables easy manipulation and sorting of the entries.
   3. A weighted average can be used if one risk component has more signiﬁcance for the project.
3. 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.

1. Project manager studies the resultant sorted table and deﬁnes a cutoff line.

* *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 below the line are re-evaluated to accomplish second-order prioritization.
* Risk impact and probability have a distinct inﬂuence on management concern.
  + Risk factor with a high impact but a very low probability of occurrence should not absorb a signiﬁcant amount of management time.
  + High-impact risks with moderate to high probability and low-impact risks with high probability should be carried forward into the risk analysis steps that follow.
* 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*

#### Assessing Risk Impact

* Three factors determine the consequences if a risk occurs:
  + ***Nature*** of the risk - the problems that are likely if it occurs.

o e.g., a poorly deﬁned external interface to customer hardware (a technical risk) will preclude early design and testing and will likely lead to system integration problems late in a project.

* + ***Scope*** of a risk - combines the severity with its overall distribution (how much of the project will be affected or how many customers are harmed?).
  + ***Timing*** of a risk - when and how long the impact will be felt.
* Steps recommended to determine the overall consequences of a risk:
  1. Determine the average probability of occurrence value for each risk component.
  2. Using Figure 1, determine the impact for each component based on the criteria shown.
  3. Complete the risk table and analyze the results as described in the preceding sections.

Overall ***risk exposure****,* **RE**, determined using:

**RE = *P* x *C***

***P*** is the probability of occurrence for a risk

***C*** is the the cost to the project should the risk occur.

#### Example

**Assume the software team deﬁnes a project risk in the following manner:**

**Risk identiﬁcation.**

* Only 70 percent of the software components scheduled for reuse will be integrated into the application.
* The remaining functionality will have to be custom developed.

**Risk probability.** 80% (likely).

#### Risk impact.

* 60 reusable software components were planned.
* If only 70 percent can be used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development).
* Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00, the overall cost (impact) to develop the components would be 18 x 100 x 14 = $25,200.

**Risk exposure.** *RE* = 0.80 x 25,200 ~ $20,200.

#### Risk Assessment

* Have established a set of triplets of the form:

**[*ri*, *li*, *xi*]**

where

***r****i* is risk

***li*** is the likelihood (probability) of the risk

***xi*** is the impact of the risk.

* During **risk assessment** :
  + further examine the accuracy of the estimates that were made during risk projection
  + attempt to rank the risks that have been uncovered
  + begin thinking about ways to control and/or avert risks that are likely to occur.
* Must Deﬁne a ***risk referent level***
  + performance, cost, support, and schedule represent risk referent levels.
  + There is a level for performance degradation, cost overrun, support diﬃculty, or schedule slippage (or any combination of the four) that will cause the project to be terminated.
  + A risk referent level has a single point, called the ***referent point*** or ***break point****,* at which the decision to proceed with the project or terminate it are equally weighted.
* Referent level rarely represented as a smooth line on a graph.
* Most cases - a region in which there are areas of uncertainty
* Therefore, during risk assessment, perform the following steps:

1. Deﬁne the risk referent levels for the project.
2. Attempt to develop a relationship between each (*ri, li, xi*) and each of the referent levels.
3. Predict the set of referent points that deﬁne a region of termination, bounded by a curve or areas of uncertainty.
4. Try to predict how compound combinations of risks will affect a referent level.

RISK REFINEMENT

* A risk may be stated generally during early stages of project planning.
* With time, more is learned about the project and the risk
  + may be possible to reﬁne the risk into a set of more detailed risks
* Represent risk in ***condition-transition-consequence*** (CTC) format.
  + Stated in the following form:

#### Given that <condition> then there is concern that (possibly)

**<consequence>**

* Using CTC format for the reuse we can write:

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.



RISK MITIGATION, MONITORING, AND MANAGEMENT

Effective strategy must consider three issues: risk avoidance

risk monitoring

risk management and contingency planning

* Proactive approach to risk - avoidance strategy.
* Develop *risk mitigation* plan*.*
* e.g. assume high staff turnover is noted as a project risk, ***r****1.*
* Based on past history
  + the likelihood, ***l****1,* of high turnover is estimated to be 0.70
  + the impact, ***x****1,* is projected at level 2.
  + So… high turnover will have a critical impact on project cost and schedule.
* Develop a strategy to mitigate this risk for reducing turnover.
* Possible steps to be taken

Meet with 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 be sure that documents are developed in a timely manner.

Conduct peer reviews of all work (so that more than one person is "up to speed").

Assign a backup staff member for every critical technologist.

* Project manager monitors for likelihood of risk
* For high staff turnover, the following factors can be monitored: General attitude of team members based on project pressures. The degree to which the team has jelled.

Interpersonal relationships among team members. Potential problems with compensation and beneﬁts.

The availability of jobs within the company and outside it.

* Project manager should monitor the effectiveness of risk mitigation steps.
* *Risk management and contingency planning* assumes that mitigation efforts have failed and that the risk has become a reality.

e.g., the project is underway and a number of people announce that they will be leaving.

* Mitigation strategy makes sure:

§ backup is available

§ information is documented

§ knowledge has been dispersed across the team.

* RMMM steps incur additional project cost

e.g. spending time to "backup" every critical technologist costs money.

* Large project - 30 or 40 risks.
* 80 percent of the overall project risk (i.e., 80 percent of the potential for project failure) can be accounted for by only 20 percent of the identiﬁed risks.
* Work performed during earlier risk analysis steps will help the planner to determine which of the risks reside in that 20 percent (e.g., risks that lead to the highest risk exposure).

### THE RMMM PLAN

* ***Risk Mitigation, Monitoring and Management Plan*** (RMMM) - documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan.
* Alternative to RMMM - ***risk information sheet*** (RIS)

RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily.

1. Risk monitoring is a project tracking activity
2. Three primary objectives:
   1. assess whether predicted risks do, in fact, occur
   2. ensure that risk aversion steps deﬁned for the risk are being properly applied
   3. collect information that can be used for future risk analysis. Problems that occur during a project can be traced to more than one risk.

Another job of risk monitoring is to attempt to allocate *origin*

(what risk(s) caused which problems throughout the project).

# Chapter 7: Project Scheduling and Tracking

### BASIC CONCEPTS

#### Causes of Software Lateness:

An unrealistic deadline established by someone outside the software development group and forced on managers and practitioners within the group.

Changing customer requirements that are not reﬂected in schedule changes.

An honest underestimate of the amount of effort and/or the number of resources that will be required to do the job.

Predictable and/or unpredictable risks that were not considered when the project commenced.

Technical diﬃculties that could not have been foreseen in advance. Human diﬃculties that could not have been foreseen in advance.

Miscommunication among project staff that results in delays.

A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem.

* ***Software project scheduling*** *-* activity that distributes estimated effort across the planned project duration by allocating the effort to speciﬁc software engineering tasks.
* Schedule evolves over time.
  + Early stages of planning - ***macroscopic schedule*** - identiﬁes all major software engineering activities and the product functions to which they are applied.
  + Project under way - ***detailed schedule*** *-* each entry on the macroscopic schedule is reﬁned*.* Here, s

§ speciﬁc software tasks are identiﬁed and scheduled.

* Two Points of view for Scheduling

1. End-date for release of a computer-based system has already (and irrevocably) been established.
2. Rough chronological bounds have been discussed but that the end- date is set by the software engineering organization. Effort is distributed to make best use of resources and an end-date is deﬁned after careful analysis of the software.

* Principles guiding software project scheduling:
  + **Compartmentalization.** The project must be compartmentalized into a number of manageable activities and tasks.
  + **Interdependency.** The interdependency of each compartmentalized activity or task must be determined. Some tasks must occur in sequence while others can occur in parallel.

#### Time allocation.

§ Each task to be scheduled allocated some number of work units

§ Each task assigned a start date and a completion date that are a function of the interdependencies and whether work will be conducted on a full-time or part-time basis.

#### Effort validation.

§ Every project has a deﬁned number of staff members.

§ As time allocation occurs, the project manager ensures that no more than the allocated number of people has been scheduled at any given time.

* + **Deﬁned responsibilities.** Every task that is scheduled should be assigned to a speciﬁc team member.
  + **Deﬁned outcomes.** Every task that is scheduled should have a deﬁned outcome. For software projects, the outcome is normally a work product or a part of a work product.
  + **Deﬁned milestones.** Every task or group of tasks should be associated with a project milestone. A milestone is accomplished when one or more work products has been reviewed for quality and has been approved.

### THE RELATIONSHIP BETWEEN PEOPLE AND EFFORT

* Small software development project - single person can analyze requirements, perform design, generate code, and conduct tests.
* Larger software development project - more people must become involved.
* Highly nonlinear relationship between chronological time to complete a project and human effort applied to the project.
* The number of delivered lines of code (source statements), ***L****,* is related to effort and development time by the equation:

***L* = *P* x *E* 1/3 *t* 4/3**

where

***E*** is development effort in person-months

***P*** is a productivity parameter that reﬂects a variety of factors that lead to high-quality software engineering work (typical values for ***P*** range between 2,000 and 12,000)

***t*** is the project duration in calendar months.

* Rearranging this software equation gives an expression for development effort ***E****:*

***E = L3/(P3t4)***

where

***E*** is the effort expended (in person-years) over the entire life cycle for software development and maintenance

***t*** is the development time in years.

Example:

* Real-time software project:

o 33,000 LOC

* + 12 person-years of effort.
* If eight people are assigned to the project team, the project can be completed in approximately 1.3 years.
* Extend the end-date to 1.75 years:

***E* = *L*3/( *P*3*t*4) ~** 3.8 person-years.

* Extending the end-date six months reduces the number of people from eight to four!
* ***40–20–40 rule*** - Recommended distribution of effort across the deﬁnition and development phases.
  + Forty percent of all effort is allocated to front-end analysis and design.
  + Forty percent is applied to back-end testing.
  + Twenty percent is applied to coding.

### DEFINING A TASK SET FOR THE SOFTWARE PROJECT

* ***Task set*** *-* collection of software engineering work tasks, milestones, and deliverables that must be accomplished to complete a particular project.
* Task sets are designed to accommodate different types of projects and different degrees of rigor.
* Most software organizations encounter the following projects:

1. *Concept development projects* - initiated to explore some new business concept or application of some new technology.
2. *New application development projects* - undertaken as a consequence of a speciﬁc customer request.
3. *Application enhancement projects* - occur when existing software undergoes major modiﬁcations to function, performance, or interfaces that are observable by the end-user.
4. *Application maintenance projects* - correct, adapt, or extend existing software in ways that may not be immediately obvious to the end- user.
5. *Reengineering projects* - undertaken with the intent of rebuilding an existing (legacy) system in whole or in part.

* ***Degree of rigor*** is a function of many project characteristics. Four different degrees of rigor can be deﬁned:

1. **Casual.** All process framework activities are applied, but only a minimum task set is required. In general, umbrella tasks will be minimized and documentation requirements will be reduced. All basic principles of software engineering are still applicable.
2. **Structured.** The process framework will be applied for this project. Framework activities and related tasks appropriate to the project type will be applied and umbrella activities necessary to ensure high quality will be applied. SQA, SCM, documentation, and measurement tasks will be conducted in a streamlined manner.
3. **Strict.** The full process will be applied for this project with a degree of discipline that will ensure high quality. All umbrella activities will be applied and robust work products will be produced.
4. **Quick reaction.** The process framework will be applied for this project, but because of an emergency situation only those tasks essential to maintaining good quality will be applied. "Back-ﬁlling" (i.e., developing a complete set of documentation, conducting additional reviews) will be accomplished after the application/product is delivered to the customer.

* Project manager must develop a systematic approach for selecting the degree of rigor that is appropriate for a particular project.
* To accomplish this, **project adaptation criteria** are deﬁned and a task set selector value is computed
* ***Adaptation criteria*** are used to determine the recommended degree of rigor with which the software process should be applied on a project.

Eleven adaptation criteria are deﬁned for software projects:

1. Size of the project
2. Number of potential users
3. Mission criticality
4. Application longevity
5. Stability of requirements
6. Ease of customer/developer communication
7. Maturity of applicable technology
8. Performance constraints
9. Embedded and nonembedded characteristics
10. Project staff
11. Reengineering factors

* Each adaptation criterion is assigned a grade that ranges between 1 and 5

**1** represents a project in which a small subset of process tasks are required and overall methodological and documentation requirements are minimal

**5** represents a project in which a complete set of process tasks should be applied and overall methodological and documentation requirements are substantial

**Task Selector Table**

* To select the appropriate task set for a project, the following steps should be conducted:
  1. Review each of the adaptation criteria and assign the appropriate grades (1 to 5) based on the characteristics of the project.
     + Enter grade into table.
  2. Review the weighting factors assigned to each of the criteria.
     + Value of a weighting factor ranges from 0.8 to 1.2 and provides an indication of the relative importance of a particular adaptation criterion to the types of software developed within the local environment.
  3. Multiply the grade entered in table by the ***weighting factor*** and by the ***entry point multiplier*** for the type of project to be undertaken.
     + Entry point multiplier takes on a value of 0 or 1 and indicates the relevance of the adaptation criterion to the project type.
     + Result of the product

#### grade x weighting factor x entry point multiplier

* + - Placed in the Product column of table for each adaptation criteria individually.
  1. Compute the average of all entries in the Product column and place the result in the space marked ***task set selector*** (TSS).
* This value will be used to help select the task set that is most appropriate for the project.

#### Interpreting the TSS Value and Selecting the Task Set

* Once the task set selector is computed, the following guidelines can be used to select the appropriate task set for a project:

**Task set selector value Degree of rigor**

TSS < 1.2 casual

* 1. < TSS < 3.0 structured

TSS > 2.4 strict

* Overlap in TSS values from one recommended task set to another is intended to illustrate that sharp boundaries are impossible to deﬁne when making task set selections.
* The task set selector value, past experience, and common sense must all be factored into the choice of the task set for a project.
* Table illustrates how TSS might be computed for a hypothetical project.
  + Project manager selects the grades shown in the Grade column.
  + Project type is *new application development.*
  + Therefore, entry point multipliers are selected from the NDev column. T
  + he entry in the Product column is computed using

#### Grade x Weight x NewDev entry point multiplier

* Value of TSS (computed as the average of all entries in the product column) is 2.8.
* The manager has the option of using either the structured or the strict task set.

SELECTING SOFTWARE ENGINEERING TASKS

* Development projects are approached by applying the following major tasks:
  + 1. **Concept scoping** determines the overall scope of the project.
    2. **Preliminary concept planning** establishes the organization's ability to undertake the work implied by the project scope.
    3. **Technology risk assessment** evaluates the risk associated with the technology to be implemented as part of project scope.
    4. **Proof of concept** demonstrates the viability of a new technology in the software context.
    5. **Concept implementation** implements the concept representation in a manner that can be reviewed by a customer and is used for "marketing" purposes when a concept must be sold to other customers or management.
    6. **Customer reaction to the concept** solicits feedback on a new technology concept and targets speciﬁc customer applications.
* Development framework activities are iterative in nature.
* Actual development project might approach these activities in a number of planned increments, each designed to produce a deliverable that can be evaluated by the customer.
* Model Selection:
  + Linear process model
  + Evolutionary model

### REFINEMENT OF MAJOR TASKS

* Major tasks may be used to deﬁne a macroscopic schedule for a project.
* Macroscopic schedule must be reﬁned to create a detailed project schedule.
* Reﬁnement begins by taking each major task and decomposing it into a set of subtasks (with related work products and milestones).

Example of task decomposition - ***concept scoping*** for a development project:

### DEFINING A TASK NETWORK

* Individual tasks and subtasks have interdependencies based on their sequence.
* It is likely that development activities and tasks will be performed in parallel.
* Concurrent tasks must be coordinated so that they will be complete when later tasks require their work product(s).
* ***Task network ( activity network )*** is a graphic representation of the task ﬂow for a project.
  + The task network depicts major software engineering tasks.
* Parallel tasks occur asynchronously
  + Planner must determine intertask dependencies to ensure continuous progress toward completion.
  + Project manager should be aware of those tasks that must be completed on schedule if the project as a whole is to be completed on schedule.

### SCHEDULING

* Generalized project scheduling tools and techniques can be applied with little modiﬁcation to software projects.
* *Program evaluation and review technique* (PERT) and *critical path method* (CPM) are two project scheduling methods that can be applied to software development.
* Both techniques are driven by information already developed in earlier project planning activities:
  + Estimates of effort
  + A decomposition of the product function
  + The selection of the appropriate process model and task set
  + Decomposition of tasks
* Interdependencies among tasks may be deﬁned using a task network.
* Tasks (*work breakdown structure* (WBS)) are deﬁned for the product as a whole or for individual functions.
* PERT and CPM provide quantitative tools that allow the software planner to

1. determine the *critical path—*the chain of tasks that determines the duration of the project
2. establish "most likely" time estimates for individual tasks by applying statistical models
3. calculate "boundary times" that deﬁne a time "window" for a particular task.

* Important boundary times discerned from a PERT or CPM network:
  1. the earliest time that a task can begin when all preceding tasks are completed in the shortest possible time
  2. the latest time for task initiation before the minimum project completion time is delayed
  3. the earliest ﬁnish—the sum of the earliest start and the task duration
  4. the latest ﬁnish—the latest start time added to task duration
  5. the *total ﬂoat—*the amount of surplus time or leeway allowed in scheduling tasks so that the network critical path is maintained on schedule.
* Boundary time calculations lead to a determination of critical path and provide the manager with a quantitative method for evaluating progress as tasks are completed.
* PERT and CPM implemented in a variety of automated tools that are available for the personal computer
* Such tools are easy to use and make the scheduling methods described previously available to every software project manager

#### Timeline Charts

* Software project schedule - planner begins with a set of tasks
* Effort, duration, and start date are input for each task.
* Tasks may be assigned to speciﬁc individuals.
* ***Timeline chart****,* ( *Gantt chart)* generated.:

Depicts a part of a software project schedule that emphasizes the task for a new word-processing (WP) software product.

All project tasks are listed in the left-hand column. Horizontal bars indicate the duration of each task.

When multiple bars occur at the same time on the calendar, task concurrency is implied. The diamonds indicate milestones.

Software project scheduling tools produce *project tables—*a tabular listing of all:

project tasks

their planned and actual start- and end-dates, variety of related information

Used in conjunction with the timeline chart, project tables enable the project manager to track progress.

#### Tracking the Schedule

The project schedule provides a road map for a software project manager. Project schedule deﬁnes the tasks and milestones that must be tracked and controlled as the project proceeds.

Tracking can be accomplished in a number of different ways:

* Conducting periodic project status meetings in which each team member reports progress and problems.
* Evaluating the results of all reviews conducted throughout the software engineering process.
* Determining whether formal project milestones have been accomplished by the scheduled date.
* Comparing actual start-date to planned start-date for each project task listed in the resource table
* Meeting informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.

*Time-boxing* strategy recognizes that the complete product may not be deliverable by the predeﬁned deadline.

* An incremental software paradigm is chosen and a schedule is derived for each incremental delivery.
* Tasks associated with each increment are time-boxed.
* The schedule for each task is adjusted by working backward from the delivery date for the increment.
* A "box" is put around each task.
* When a task hits the boundary of its time box (plus or minus 10 percent), work stops and the next task begins.

### EARNED VALUE ANALYSIS

* *Earned value analysis* (EVA) - qualitative approaches to project tracking
  + Provides a common value scale for every software project task, regardless of the type of work being performed.
  + The total hours to do the whole project are estimated, and every task is given an earned value based on its estimated percentage of the total.
* Earned value is a measure of progress
  + Able to assess the "percent of completeness" of a project using quantitative analysis

Following steps performed:

1. The *budgeted cost of work scheduled* (BCWS) is determined for each work task represented in the schedule.
   1. The work (in person-hours or person-days) of each software engineering task is planned.
   2. BCWS***i*** is the effort planned for work task ***i****.*
   3. To determine progress at a given point along the project schedule, the value of BCWS is the sum of the **BCWS*i*** values for all work tasks that should have been completed by that point in time on the project schedule.
2. The BCWS values for all work tasks are summed to derive the budget at completion, BAC. Hence,

BAC = S (BCWS*k*) for all tasks ***k***

1. Next, the value for *budgeted cost of work performed* (BCWP) is computed.
   1. The value for BCWP is the sum of the BCWS values for all work tasks that have actually been completed by a point in time on the project schedule.

* The distinction between BCWS and BCWP is that the former represents the budget of the activities that were planned to be completed and the latter represents the budget of the activities that actually were completed.
* Given values for BCWS, BAC, and BCWP, progress indicators can be computed:

Schedule performance index, SPI = BCWP/BCWS Schedule variance, SV = BCWP - BCWS

* SPI is an indication of the eﬃciency with which the project is utilizing scheduled resources.
  + An SPI value close to 1.0 indicates eﬃcient execution of the project schedule.
* SV is an absolute indication of variance from the planned schedule.
* Percent scheduled for completion = BCWS/BAC
  + provides an indication of the percentage of work that should have been completed by time *t.*
* Percent complete = BCWP/BAC
  + provides a quantitative indication of the percent of completeness of the project at a given point in time, *t.*
* *Actual cost of work performed,* ACWP - the sum of the effort expended on work tasks that have been completed by a point in time on the project schedule.
  + It is then possible to compute

Cost performance index, CPI = BCWP/ACWP Cost variance, CV = BCWP - ACWP

* + CPI close to 1.0 provides a strong indication that the project is within its deﬁned budget.
  + CV is an absolute indication of cost savings (against planned costs) or shortfall at a particular stage of a project.

### ERROR TRACKING

* Throughout the software process, a project team errors associated with each work product.
* Error-related measures and resultant metrics are collected over many software projects, can be used as a baseline for comparison against error data collected in real time.
* Error tracking can be used as one means for assessing the status of a current project.

#### Defect removal eﬃciency

* Software team performs formal technical reviews (and, later, testing) to ﬁnd and correct errors, ***E****,* in work products produced during software engineering tasks.
* Any errors that are not uncovered (but found in later tasks) are defects,

***D****.*

* Defect removal eﬃciency is deﬁned as

**DRE = *E/*(*E* + *D*)**

* DRE is a process metric that provides a strong indication of the effectiveness of quality assurance activities
* DRE can also be used to assist a project manager in determining the progress that is being made as a software project moves through its scheduled work tasks.
  + e.g. a software organization has collected error and defect data over the past 24 months and has developed averages for the following metrics:

§ Errors per requirements speciﬁcation page, *E*req

§ Errors per component—design level, *E*design

§ Errors per component—code level, *E*code

§ DRE—requirements analysis

§ DRE—architectural design

§ DRE—component level design

§ DRE—coding

* As the project progresses through each software engineering step, the software team records and reports the number of errors found during requirements, design, and code reviews.
* The project manager calculates current values for *E*req, *E*design, and

*E*code.

* Compared to averages for past projects.
* If current results vary by more than 20% from the average, there may be cause for concern and there is certainly cause for investigation.
* For example, if *E*req = 2.1 for project X, yet the organizational average is 3.6, one of two scenarios is possible:

1. the software team has done an outstanding job of developing the requirements speciﬁcation or
2. the team has been lax in its review approach. If the second scenario appears likely, the project manager should take immediate steps to build additional design time into the schedule to accommodate the requirements defects that have likely been propagated into the design activity.

### THE PROJECT PLAN

* Each step in the software engineering process should produce a deliverable that can be reviewed and that can act as a foundation for the steps that follow.
* The *Software Project Plan* is produced at the culmination of the planning tasks.
* It provides baseline cost and scheduling information that will be used throughout the software process.
* The *Software Project Plan* must
  1. communicate scope and resources to software management, technical staff, and the customer
  2. deﬁne risks and suggest risk aversion techniques
  3. deﬁne cost and schedule for management review
  4. provide an overall approach to software development for all people associated with the project
  5. outline how quality will be ensured and change will be managed.

Software Maintenance Overviiew

Software maiintenance iis wiidelly accepted part of SDLC now a days..

IIt stands for allll

the

modiiﬁcatiions and updatiions done after the delliivery of software product.. There are number of reasons,, why modiiﬁcatiions are requiired,, some of them are briieﬂy mentiioned bellow::

**Market Condiitiions** -- Polliiciies,, whiich changes over the tiime,, such as taxatiion and newlly

iintroduced constraiints lliike,, modiiﬁcatiion..

how to maiintaiin bookkeepiing,,

may triigger need for

**Clliient Requiirements** -- Over the tiime,, customer may ask for new features or functiions iin the software..

**Host Modiiﬁcatiions** -- IIf any of the hardware and/or pllatform (such as operatiing system) of the target host changes,, software changes are needed to keep adaptabiilliity..

## Organiizatiion Changes

-- IIf there iis any busiiness llevell

change at clliient end,,

such as

reductiion of organiizatiion strength,, acquiiriing another company,, organiizatiion venturiing iinto new busiiness,, need to modiify iin the oriigiinall software may ariise..

# Types of maiintenance

IIn a software lliifetiime,, type of maiintenance may vary based on iits nature.. IIt may be jjust a routiine maiintenance tasks as some bug diiscovered by some user or iit may be a llarge event iin iitsellf based

on maiintenance siize or nature.. characteriistiics::

Follllowiing are some types of maiintenance based on theiir

**Correctiive Maiintenance** -- Thiis iinclludes modiiﬁcatiions and updatiions done iin order to

correct or ﬁx probllems,, whiich are eiither diiscovered by user or conclluded by user error reports..

**Adaptiive Maiintenance** -- Thiis iinclludes modiiﬁcatiions and updatiions applliied to keep the

software product up--to date and tuned to the ever changiing worlld of technollogy and busiiness enviironment..

**Perfectiive Maiintenance** -- Thiis iinclludes modiiﬁcatiions and updates done iin order to keep

the software usablle over llong periiod of tiime..

IIt iinclludes new features,,

new user

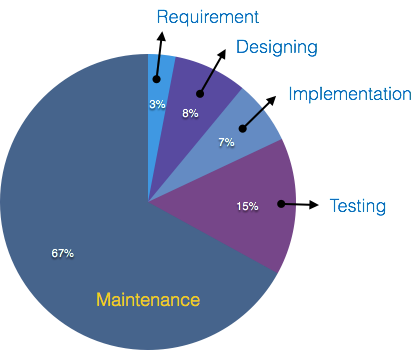
requiirements for reﬁniing the software and iimprove iits relliiabiilliity and performance..

**Preventiive Maiintenance** -- Thiis iinclludes modiiﬁcatiions and updatiions to prevent future

probllems of the software.. IIt aiims to attend probllems,, whiich are not siigniiﬁcant at thiis moment but may cause seriious iissues iin future..

# Cost of Maiintenance

Reports suggest that the cost of maiintenance iis hiigh.. A study on estiimatiing software maiintenance found that the cost of maiintenance iis as hiigh as 67% of the cost of entiire software process cyclle..



On an average,, the cost of software maiintenance iis more than 50% of allll SDLC phases.. There are variious factors,, whiich triigger maiintenance cost go hiigh,, such as::

## Reall-worlld factors affectiing Maiintenance Cost

The standard age of any software iis consiidered up to 10 to 15 years..

Ollder softwares,, whiich were meant to work on sllow machiines wiith lless memory and storage capaciity cannot keep themsellves challllengiing agaiinst newlly comiing enhanced softwares on modern hardware..

As technollogy advances,, iit becomes costlly to maiintaiin olld software..

Most maiintenance engiineers are newbiie and use triiall and error method to rectiify probllem..

Often,, changes made can easiilly hurt the oriigiinall structure of the software,, makiing iit hard for any subsequent changes..

Changes are often lleft undocumented whiich may cause more conﬂiicts iin future..

## Software-end factors affectiing Maiintenance Cost

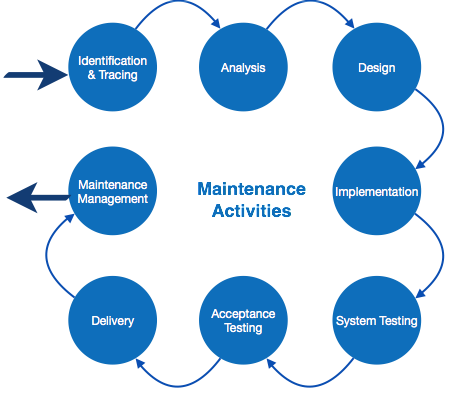
Structure of Software Program Programmiing Language

Dependence on externall enviironment

Staff relliiabiilliity and avaiillabiilliity

# Maiintenance Actiiviitiies

IIEEE proviides a framework for sequentiiall maiintenance process actiiviitiies.. IIt can be used iin iiteratiive manner and can be extended so that customiized iitems and processes can be iinclluded..



These actiiviitiies go hand--iin--hand wiith each of the follllowiing phase::

**IIdentiiﬁcatiion & Traciing** -- IIt iinvollves actiiviitiies pertaiiniing to iidentiiﬁcatiion of requiirement of modiiﬁcatiion or maiintenance.. IIt iis generated by user or system may iitsellf report viia llogs or error messages..Here,, the maiintenance type iis cllassiiﬁed allso..

**Anallysiis** -- The modiiﬁcatiion iis anallyzed for iits iimpact on the system iinclludiing safety and securiity iimplliicatiions.. IIf probablle iimpact iis severe,, allternatiive sollutiion iis llooked for.. A set of requiired modiiﬁcatiions iis then materiialliized iinto requiirement speciiﬁcatiions.. The cost of modiiﬁcatiion/maiintenance iis anallyzed and estiimatiion iis conclluded..

## Desiign

-- New modulles,,

whiich need to be repllaced or modiiﬁed,,

are desiigned agaiinst

requiirement speciiﬁcatiions set iin the previious stage.. Test cases are created for valliidatiion and veriiﬁcatiion..

**IImpllementatiion** -- The new modulles are coded wiith the hellp of structured desiign created iin the desiign step..Every programmer iis expected to do uniit testiing iin parallllell..

**System Testiing** -- IIntegratiion testiing iis done among newlly created modulles.. IIntegratiion testiing iis allso carriied out between new modulles and the system.. Fiinalllly the system iis tested as a wholle,, follllowiing regressiive testiing procedures..

**Acceptance Testiing** -- After testiing the system iinternalllly,, iit iis tested for acceptance wiith the hellp of users.. IIf at thiis state,, user compllaiints some iissues they are addressed or noted to address iin next iiteratiion..

**Delliivery** -- After acceptance test,, the system iis deplloyed allll over the organiizatiion eiither by smallll update package or fresh iinstallllatiion of the system.. The ﬁnall testiing takes pllace at clliient end after the software iis delliivered..

Traiiniing faciilliity iis proviided iif requiired,, iin addiitiion to the hard copy of user manuall..

**Maiintenance management** -- Conﬁguratiion management iis an essentiiall part of system

maiintenance.. IIt iis aiided wiith versiion controll toolls to controll versiions,, semii--versiion or patch management..

# Software Re-engiineeriing

When we need to update the software to keep iit to the current market,, wiithout iimpactiing iits

functiionalliity,, iit iis calllled software re--engiineeriing.. IIt iis a thorough process software iis changed and programs are re--wriitten..

where the desiign of

Legacy software cannot keep tuniing wiith the llatest technollogy avaiillablle iin the market.. As the hardware become obsollete,, updatiing of software becomes a headache.. Even iif software grows olld wiith tiime,, iits functiionalliity does not..

For examplle,,

iiniitiialllly Uniix was develloped iin assemblly llanguage..

When llanguage C came iinto

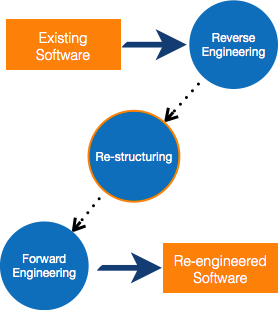
exiistence,, Uniix was re--engiineered iin C,, because workiing iin assemblly llanguage was diiﬃcullt..

Other than

thiis,,

sometiimes programmers notiice that few parts of software need more

maiintenance than others and they allso need re--engiineeriing..



## Re-Engiineeriing Process

**Deciide** what to re--engiineer.. IIs iit wholle software or a part of iit?

**Perform** Reverse Engiineeriing,, iin order to obtaiin speciiﬁcatiions of exiistiing software..

**Restructure Program** iif requiired.. For examplle,, changiing functiion--oriiented programs iinto objject--oriiented programs..

**Re-structure data** as requiired..

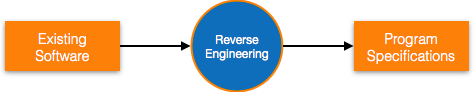
**Applly Forward engiineeriing** concepts iin order to get re--engiineered software..

There are few iimportant terms used iin Software re--engiineeriing

## Reverse Engiineeriing

IIt iis a process to achiieve system speciiﬁcatiion by thoroughlly anallyziing,, understandiing the exiistiing system.. Thiis process can be seen as reverse SDLC modell,, ii..e.. we try to get hiigher abstractiion llevell by anallyziing llower abstractiion llevells..

An exiistiing system iis previiouslly iimpllemented desiign,, about whiich we know nothiing.. Desiigners then do reverse engiineeriing by llookiing at the code and try to get the desiign.. Wiith desiign iin hand,, they try to concllude the speciiﬁcatiions.. Thus,, goiing iin reverse from code to system speciiﬁcatiion..



## Program Restructuriing

IIt iis a process to re--structure and re--construct the exiistiing software.. IIt iis allll about re--arrangiing the

source code,, eiither iin same programmiing llanguage or from one programmiing llanguage to a

diifferent one.. Restructuriing can have eiither source code--restructuriing and data--restructuriing or both..

Re--structuriing does not iimpact the functiionalliity of the software but enhance relliiabiilliity and

maiintaiinabiilliity..

Program

components,,

whiich cause errors very frequentlly can be changed,, or

updated wiith re--structuriing..

The dependabiilliity of software on obsollete hardware pllatform can be removed viia re--structuriing..

## Forward Engiineeriing

Forward engiineeriing iis a process of obtaiiniing desiired software from the speciiﬁcatiions iin hand whiich were brought down by means of reverse engiineeriing.. IIt assumes that there was some software engiineeriing allready done iin the past..

Forward engiineeriing iis same as software engiineeriing process wiith onlly one diifference – iit iis carriied out allways after reverse engiineeriing..

# Component reusabiilliity

A component iis a part of software program code,, whiich executes an iindependent task iin the system.. IIt can be a smallll modulle or sub--system iitsellf..

## Examplle

The llogiin procedures used on the web can be consiidered as components,, priintiing system iin software can be seen as a component of the software..

Components have hiigh

cohesiion of functiionalliity and llower rate of couplliing,,

ii..e..

they work

iindependentlly and can perform tasks wiithout dependiing on other modulles..

IIn OOP,, the objjects are desiigned are very speciiﬁc to theiir concern and have fewer chances to be used iin some other software..

IIn modullar programmiing,, the modulles are coded to perform speciiﬁc tasks whiich can be used across number of other software programs..

There iis a wholle new vertiicall,, whiich iis based on re--use of software component,, and iis known as Component Based Software Engiineeriing (CBSE)..

Re--use can be done at variious llevells

**Applliicatiion llevell** -- Where an entiire applliicatiion iis used as sub--system of new software..

**Component llevell** -- Where sub--system of an applliicatiion iis used..

**Modulles llevell** -- Where functiionall modulles are re--used..

Software components proviide iinterfaces,, whiich can be used to establliish communiicatiion among diifferent components..

## Reuse Process

Two kiinds of method can be adopted::

eiither by keepiing requiirements same and adjjustiing

components or by keepiing components same and modiifyiing requiirements..

**Requiirement Speciiﬁcatiion** -- The functiionall and non--functiionall requiirements are speciiﬁed,, whiich a software product must complly to,, wiith the hellp of exiistiing system,, user iinput or both..

**Desiign** -- Thiis iis allso a standard SDLC process step,, where requiirements are deﬁned iin terms of software parllance.. Basiic archiitecture of system as a wholle and iits sub--systems are created..

**Speciify Components** -- By studyiing the software desiign,, the desiigners segregate the entiire system iinto smallller components or sub--systems.. One compllete software desiign turns iinto a collllectiion of a huge set of components workiing together..

**Search Suiitablle Components** -- The software component reposiitory iis referred by

desiigners to search for the matchiing component,, iintended software requiirements....

on the basiis of functiionalliity and

**IIncorporate Components** -- Allll matched components are packed together to shape them as compllete software..