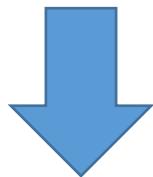


# SPM 1.0

# *What is Project?*

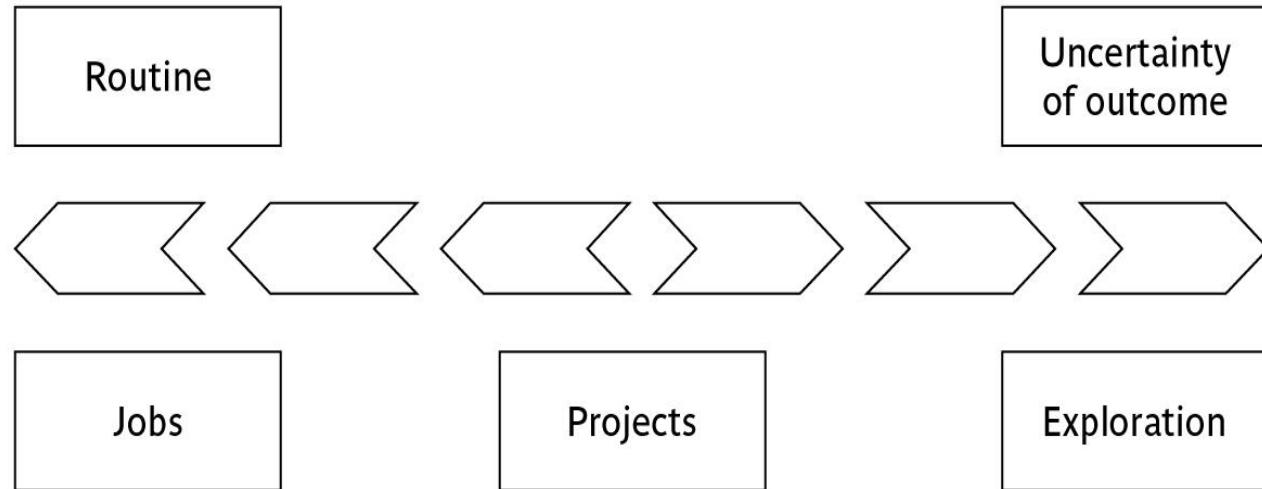


*A project is well-defined task,  
which is a collection of several  
operations done in order to achieve  
a goal (for example, software  
development and delivery).*

## A Project can be characterized as:

- ❖ Every project may have a unique and distinct goal.
- ❖ Project is not routine activity or day-to-day operations.
- ❖ Project comes with a start time and end time.
- ❖ Project ends when its goal is achieved hence it is a temporary phase in the lifetime of an organization.
- ❖ Project needs adequate resources in terms of time, manpower, finance, material and knowledge-bank.

# Jobs versus projects



**Jobs:** repetition of very well-defined and well understood tasks with very little uncertainty

**Exploration:** e.g. finding a cure for cancer: the outcome is very uncertain

**Projects:** In the middle between Jobs & Exploration

# Software Project

**A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.**

## Software Projects vs Other Projects

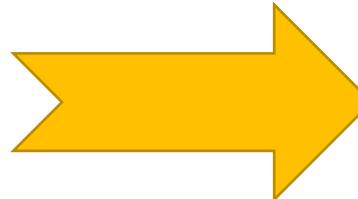


- **Invisibility:** With Software, progress is not immediately visible since work is logical; however, for physical artifacts like bridges, work progress can be seen from time to time. In Software Development, there is a level of uncertainty.
- **Complexity:** Software projects contain more complexity than other engineered artifacts. For example, in a bridge, there is a clear structural relationship between parts, whereas software component relationships are much more complicated. We can't measure the complexity of a software project until we work on it.
- **Conformity:** Physical systems are governed by consistent physical law, while Software developers have to conform to the requirements of human clients.
- **Flexibility:** Software systems are particularly subject to change. A bridge has to be built in a specific order, whereas we can make Software much more flexibly and restructure parts quite freely.

# What is software project management?

- Software project management is an art and discipline of planning and supervising software projects. It is a sub-discipline of software engineering in which software projects are planned, implemented, monitored and controlled.
- It is a procedure of managing, allocating and timing resources to develop computer software that fulfills requirements.
- In software Project Management, the client and the developers need to know the length, period and cost of the project.

# Need of Software Project Management



*It is an essential part of software organization  
to deliver quality product, keeping the cost  
within client's budget constrain and deliver the  
project as per scheduled.*

# **Project Manager:**

A project manager is an individual who has the overall responsibility for the planning, design, execution, monitoring, controlling and closure of a project.

## **Responsibilities of a Project Manager:**

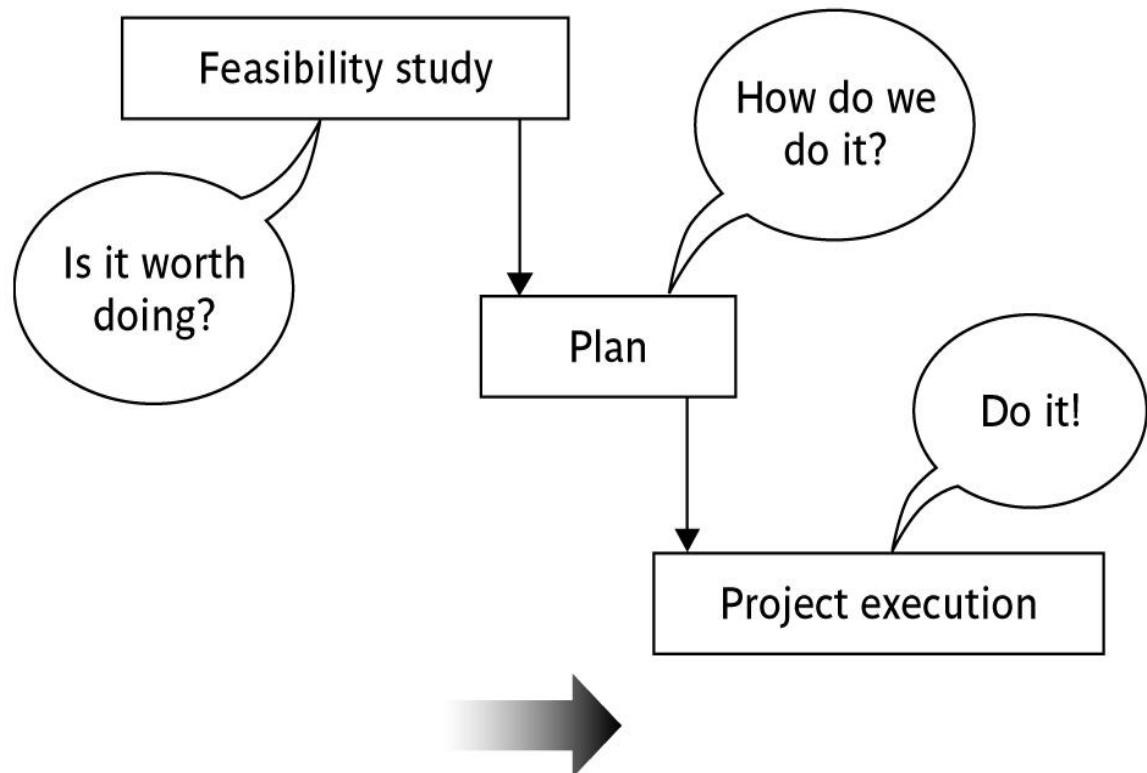
- ✓ Managing risks and issues.
- ✓ Create the project team and assigns tasks to several team members.
- ✓ Activity planning and sequencing.
- ✓ Monitoring and reporting progress.
- ✓ Modifies the project plan to deal with the situation.

# Project Management vs Contract Management

Project Management	Contract Management
Focuses on management of particular project till it gets completed successfully.	Focuses on management of contract between two companies, parties, or organizations.
Goal is to achieve desired project goal on time and within budget.	Goal is to achieve mutual satisfaction among parties or organizations or companies whose objectives are not same but are closely linked.
Project usually involves many parties from one business entity but mostly involves various related or unrelated entities.	Contract usually binds two different business entities but if there are more than two entities then there are separate sub-contracts.
It involves various processes such as planning, organizing, and managing efforts and tasks that are made and performed to complete a project successfully.	It involves various processes such as managing contracts, deliverables, guidelines, deadlines, execution, analysis
It mainly focuses on project constraints i.e. scope, time, budget, quality as per contract.	It mainly focuses on economic of project and manage claims and dispute against contract.
Objective is to predict problems or dangers as many as possible so that such problems can be removed on time and project can be completed in spite of all problems.	Objective is to create value for organization.

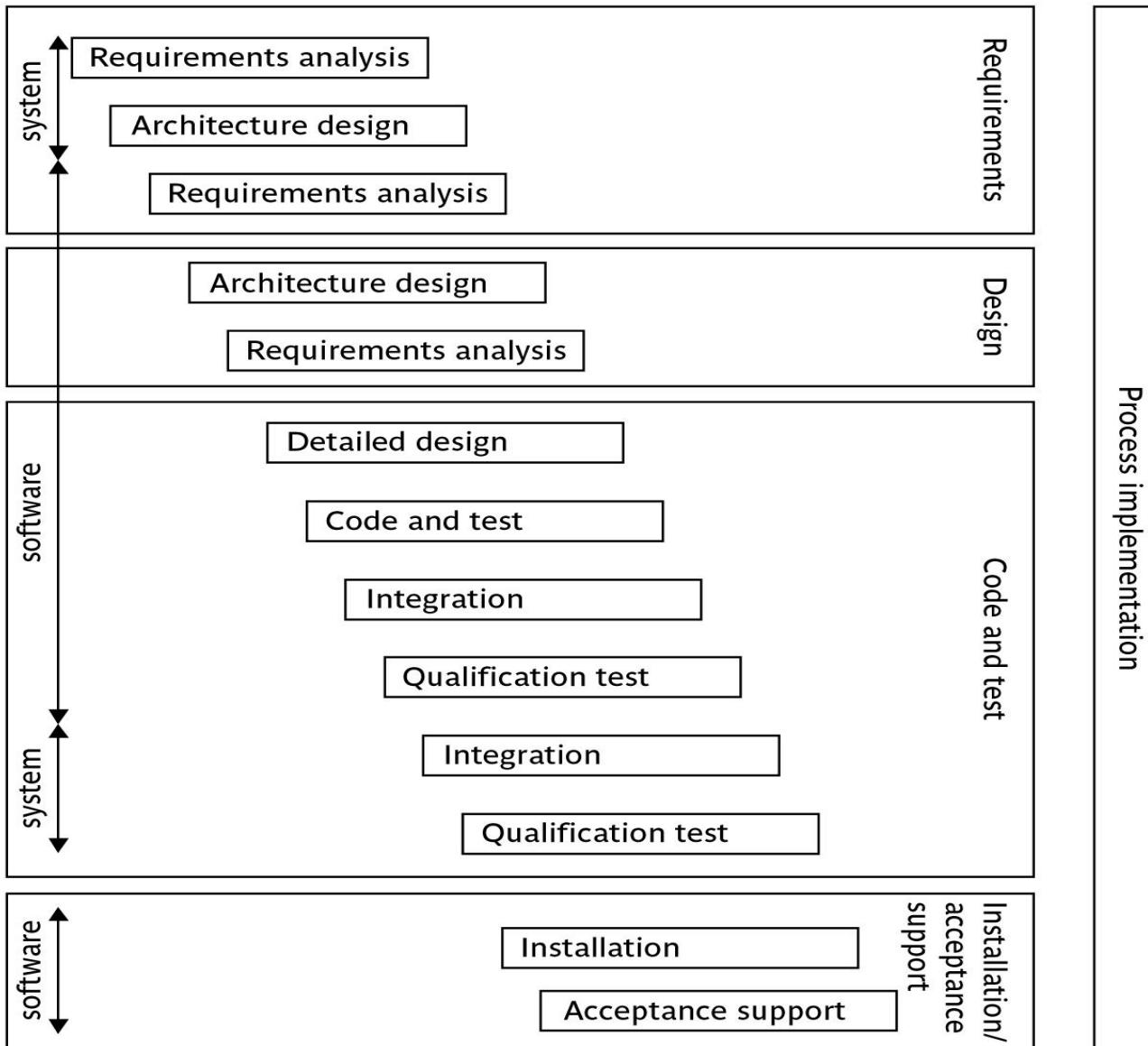
# Activities Covered by Project Management

- ❖ **Project Planning:** It is a set of multiple processes, or we can say that it a task that performed before the construction of the product starts.
- ❖ **Scope Management:** It describes the scope of the project. Scope management is important because it clearly defines what would do and what would not. Scope Management create the project to contain restricted and quantitative tasks, which may merely be documented and successively avoids price and time overrun.
- ❖ **Estimation management:** This is not only about cost estimation because whenever we start to develop software, but we also figure out their size(line of code), efforts, time as well as cost.
- ❖ **Scheduling Management:** Scheduling Management in software refers to all the activities to complete in the specified order and within time slotted to each activity. Project managers define multiple tasks and arrange them keeping various factors in mind.
- ❖ **Project Resource Management:** In software Development, all the elements are referred to as resources for the project. It can be a human resource, productive tools, and libraries.
- ❖ **Project Risk Management:** Risk management consists of all the activities like identification, analyzing and preparing the plan for predictable and unpredictable risk in the project.
- ❖ **Project Communication Management:** Communication is an essential factor in the success of the project. It is a bridge between client, organization, team members and as well as other stakeholders of the project such as hardware suppliers.
- ❖ **Project Configuration Management:** Configuration management is about to control the changes in software like requirements, design, and development of the product.



- **Feasibility study:** Is project technically feasible and worthwhile from a business point of view?
- **Planning:** Only done if project is feasible
- **Execution:** Implement plan, but plan may be changed as we go along

# Software Development Life-cycle

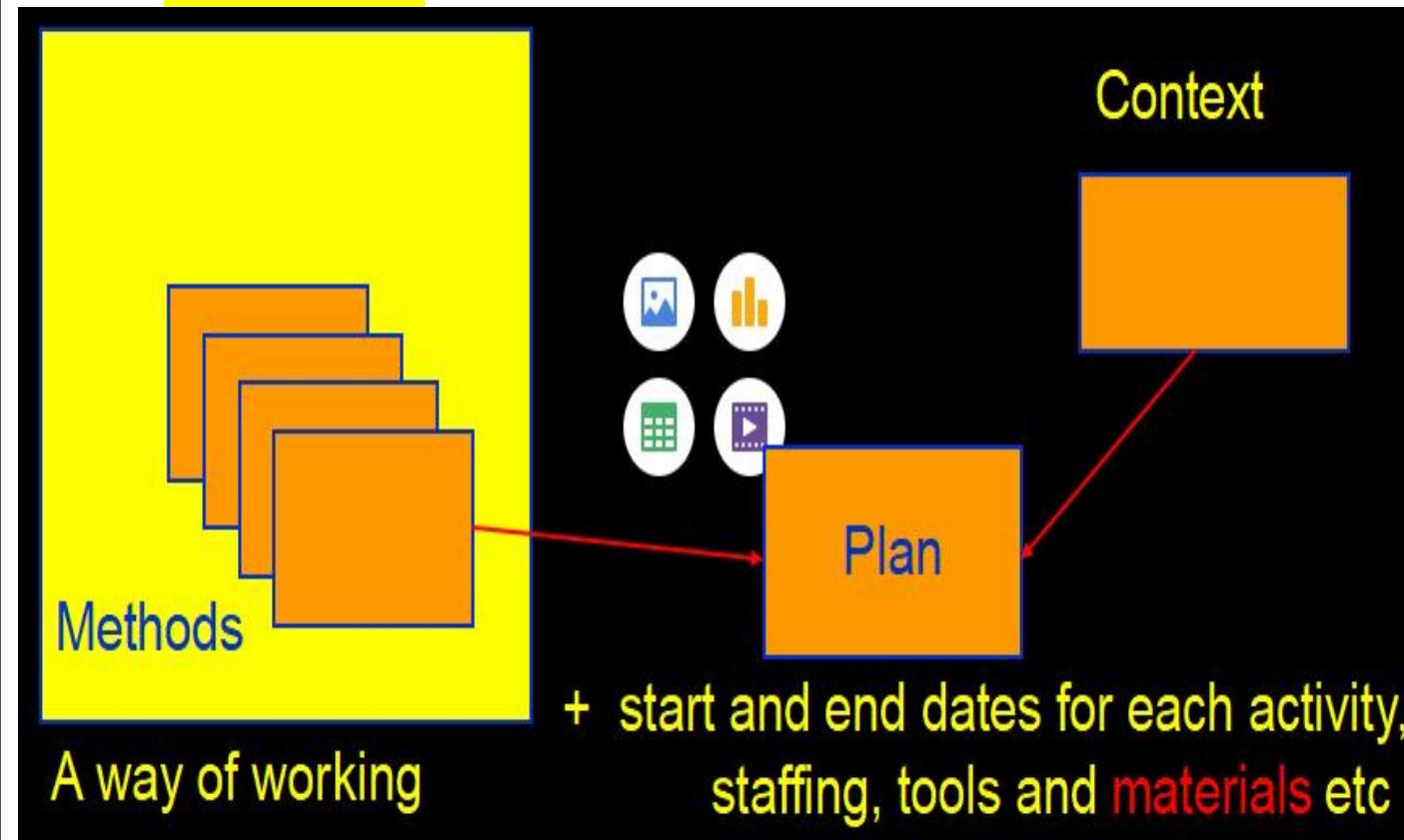


- **Requirements analysis** starts with requirements elicitation or requirements gathering which establishes what the potential users and their managers require of the new system.
- **Architecture design** The components of the new system that fulfil each requirement have to be identified. Existing components may be able to satisfy some requirements. In other cases, a new component will have to be made. These components are not only software: they could be new hardware or work processes.
- **Detailed design** Each software component is made up of a number of software units that can be separately coded and tested. The detailed design of these units is carried out separately.

- Code and test** refers to writing code for each software unit. Initial testing to debug individual software units would be carried out at this stage.
- Integration** The components are tested together to see if they meet the overall requirements. Integration could involve combining different software components, or combining and testing the software element of the system in conjunction with the hardware platforms and user interactions.
- Qualification testing** The system, including the software components, has to be tested carefully to ensure that all the requirements have been fulfilled.
- Installation** This include activities such as setting up standing data (for example, the details for employees in a payroll system), setting system parameters, installing the software onto the hardware platforms and user training.
- Acceptance support** This is the resolving of problems with the newly installed system, including the correction of any errors, and implementing agreed extensions and improvements. Software maintenance can be seen as a series of minor software projects.

# Plans, Methods and Methodologies

Methodology = a set  
of methods



- Method relates to a type of activity
- Plan takes that method and converts it to a real activities
  - ✓ Its start and end dates
  - ✓ Who will carry it out
  - ✓ What tools and materials (also information) will be needed

*The output from one method might be the input to another. Group of methods or techniques are often grouped into methodologies such as object-oriented design.*

# Some ways of Categorizing Projects

## 1. Compulsory Vs Voluntary systems (projects):

- Compulsory systems are the systems which the staff of an organisation have to use if they want to do a task.
- Voluntary systems are the systems which are voluntarily used by the users eg. computer gaming, school project, etc.

## 2. Information Vs Embedded systems (projects):

- Information systems are used by staff to carry out office processes and tasks eg. stock control system.
- Embedded systems are used to control machines eg. a system controlling equipment in a building.

## 3. Objective-based Vs Product-based systems (projects):

- Project whose requirement is to meet certain objectives which could be met in a number of ways, is objective-based project.
- Project whose requirement is to create a product, the details of which have been specified by the client, is product-based project.

# Stakeholders in a Software Project

- Internal to the project team This means that they will be under the direct managerial control of the project leader.
- External to the project team but within the same organization For example, the project leader might need the assistance of the users to carry out systems testing. Here the commitment of the people involved has to be negotiated.
- External to both the project team and the organization External stakeholders may be customers (or users) who will benefit from the system that the project implements. They may be contractors who will carry out work for the project.

# Setting objectives

Objectives focus on the desired outcomes of the project rather than tasks within it. They are the *post condition* of the project.

- Answering the question '*What do we have to do to have a success?*'
- Need for a *project authority*
  - ◆ Sets the project scope
  - ◆ Allocates/approves costs
- Could be one person - or a group
  - ◆ Project Board
  - ◆ Project Management Board
  - ◆ Steering committee

Informally, the objective of a project can be defined by completing the statement:

*The project will be regarded as a success  
if.....*

Rather like *post-conditions* for the project

Focus on *what* will be put in place, rather than *how* activities will be carried out

Objectives should be  
SMART



S – specific, that is, concrete and well-defined  
M – measurable, that is, satisfaction of the objective can be objectively judged  
A – achievable, that is, it is within the power of the individual or group concerned to meet the target  
R – relevant, the objective must relevant to the true purpose of the project  
T – time constrained: there is defined point in time by which the objective should be achieved

# Goals/sub-objectives

- These are steps along the way to achieving the objective.
- Informally, these can be defined by completing the sentence.
- To reach objective X, the following must be in place
  - A.....
  - B.....
  - C..... etc

Often a goal can be allocated to an individual.  
*Individuals might have the capability of achieving goal on their own, but not the overall objective e.g.*

*Overall objective* – user satisfaction with software product

*Analyst goal* – accurate requirements

*Developer goal* – reliable software

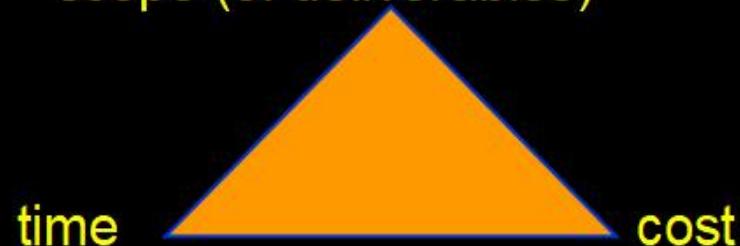
Any project plan must ensure that the business case is kept intact. For example:

- that development costs are not allowed to rise to a level which threatens to exceed the value of benefits;
- that the features of the system are not reduced to a level where the expected benefits cannot be realized;
- that the delivery date is not delayed so that there is an unacceptable loss of benefits.

# Project success/failure

- Degree to which objectives are met

scope (of deliverables)



In general if, for example, project is running out of time, this can be recovered for by reducing scope or increasing costs. Similarly costs and scope can be protected by adjusting other corners of the 'project triangle'.

## Other success criteria are.....

- These can relate to longer term, less directly tangible assets
- Improved skill and knowledge
- Creation of assets that can be used on future projects e.g. software libraries
- Improved customer relationships that lead to repeat business

## **Project objectives and Business objectives::::**

**A computer game could be delivered on time and within budget, but might then not sell.**

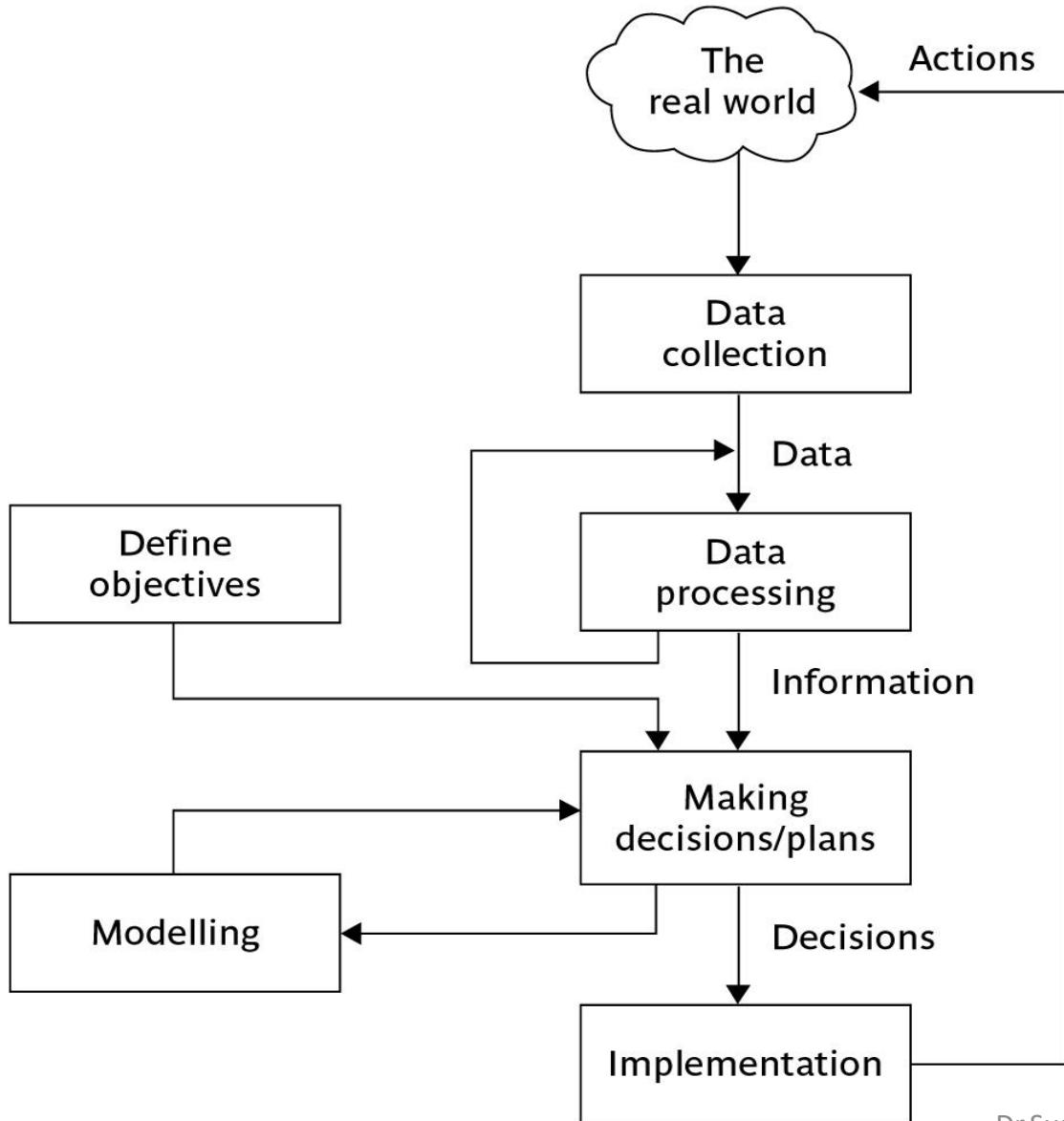
**A commercial website used for online sales could be created successfully, but customers might not use it to buy products, because they could buy the goods more cheaply elsewhere.**



# What is Management?

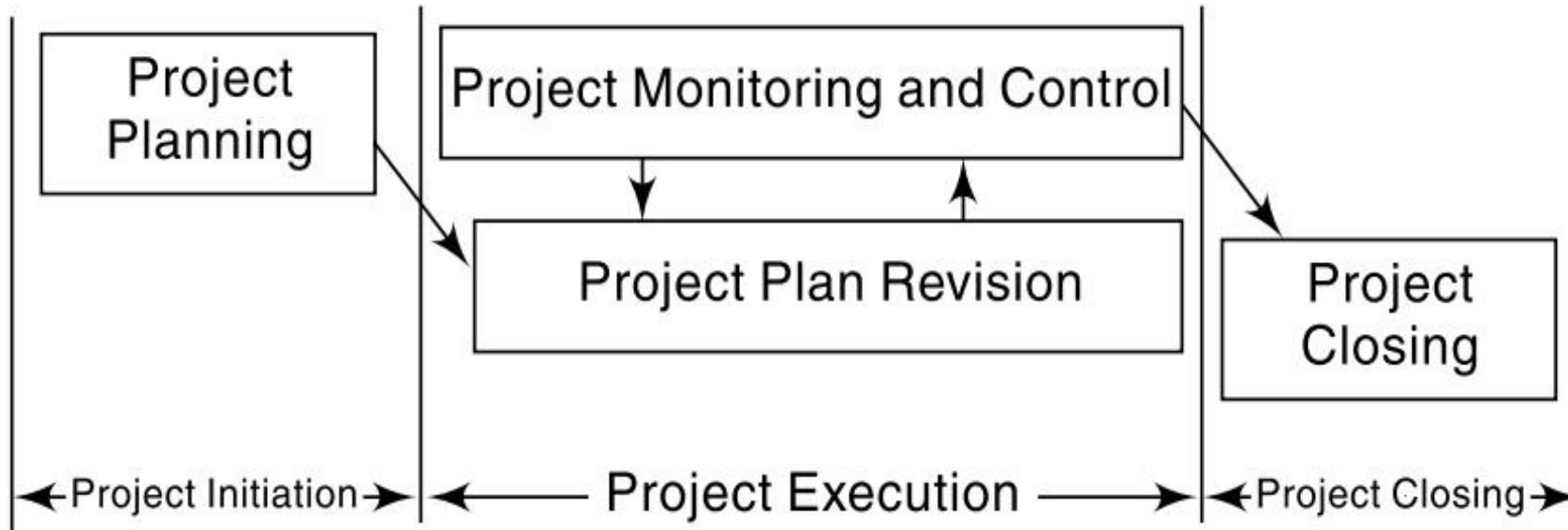
- Planning – deciding what is to be done
- Organizing – making arrangements
- Staffing – selecting the right people for the job
- Directing – giving instructions
- Monitoring – checking on progress
- Controlling – taking action to remedy hold-ups
- Innovating – coming up with solutions when problems emerge
- Representing – liaising with clients, users, developers and other stakeholders

# Management Control



- ❖ **Data:** the raw details  
e.g. '*6,000 documents processed at location X*'
- ❖ **Information:** the data is processed to produce something that is meaningful and useful  
e.g. '*productivity is 100 documents a day*'
- ❖ **Comparison with objectives/goals:**  
e.g. *we will not meet target of processing all documents by 31<sup>st</sup> March*
- ❖ **Modelling:** working out the probable outcomes of various decisions  
e.g. if we employ two more staff at location X how quickly can we get the documents processed?
- ❖ **Implementation:** carrying out the remedial actions that have been decided upon

*Much of the project manager's time is spent on only three of the eight identified activities, viz., project planning, monitoring, and control.*



# Traditional versus Modern Project Management

Characteristics	Modern approach	Traditional approach
Organizational structure	Iterative	Linear
Scale of projects	Small and medium scale	Large scale
User requirements	Interactive input	Clearly defined before implementation
Involvement of clients	High	Low
Development model	Evolutionary delivery	Life cycle
Customer involvement	Customers are involved from the time work is being performed	Customers get involved early in the project but not once the execution has started
Escalation management	When problems occur, the entire team works together to resolve it	Escalation to managers when problem arise
Test documentation	Tests are planned one sprint at a time	Comprehensive test planning

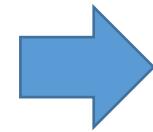
# END

# SPM 2.0

# Project Evaluation and Programme Management



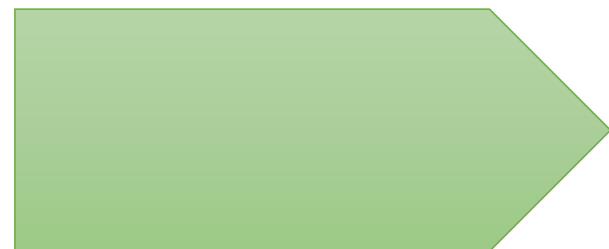
## A Business Case



*Its objective is to provide a rationale for the project by showing that the benefits of the project outcomes will exceed the costs of development, implementation and operation (or production).*

### A business case document might contain:

1. Introduction and background to the proposal
2. The proposed project
3. The market
4. Organizational and operational infrastructure
5. The benefits
6. Outline implementation plan
7. Costs
8. The financial case
9. Risks
10. Management plan



**Introduction and background:** This is a description of the current environment of the proposed project. A problem to be solved is identified.

**The proposed project:** A brief outline of the proposed project is provided.

**The market:** This is needed when the project is to create new product or a new service capability.

**Organizational and operational infrastructure:** This describes how the structure of the organization will be affected by the implementation of the project.

**Benefits:** Where possible, a financial value should be put on the benefits of the implemented project.

**Outline implementation plan:** In addition to the ICT aspects of the project, activities such as marketing, promotion and operational and maintenance infrastructures need to be considered.

**Costs:** Having outlined the steps needed to set up the operations needed by the proposal, a schedule of expected costs associated with the planned approach can now be presented.

**The financial case:** There are a number of ways in which the information on income and costs can be analysed.

**Risks:** Many estimates of costs and, more particularly, benefits of the project will be speculative at this stage and the section on risk should take account of this.

# Project Portfolio Management (PPM)

- When there are many projects run by an organization, it is vital for the organization to manage their project portfolio. This helps the organization to categorize the projects and align the projects with their organizational goals.
- Portfolio project management provides an overview of all the projects that an organization is undertaking or is considering. It prioritizes the allocation of resources to projects and decides which new projects should be accepted and which existing ones should be dropped.

## Objectives::::

- Identifying which project proposals are worth implementation
- Assessing the amount of risk of failure that a potential project has
- Deciding how to share limited resources, including staff time and finance, between projects
- Being aware of the dependencies between projects, especially where several projects need to be completed for an organization to reap benefits;
- Ensuring that projects do not duplicate work;
- Ensuring that necessary developments have not been inadvertently been missed.

# Three elements to PPM

Project  
portfolio  
definition

Project  
portfolio  
management

Project  
portfolio  
optimization

---Create a central record of all projects within an organization  
---Must decide whether to have ALL projects in the repository or, say, only ICT projects

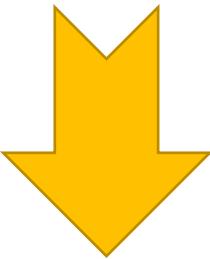
---

---Actual costing and performance of projects can be recorded and assessed.  
---The value that managers hope will be generated by each project can also be recorded. Actual performance of projects on these performance indicators can then be tracked.

---

---A better balance of projects may be achieved. Some projects could potentially be very profitable but could also be risky.  
---In the case of an e-commerce site, for example, sales may not be as great as hoped because established competitors reduce prices.  
---Other projects could have modest benefits, such as those cutting costs by automating processes, but have fewer risks. The portfolio ought to have a carefully thought-out balance between the two types of project.

## Some problems with project portfolio management



- ❖ An important role of project portfolio management is sharing resources between projects. There can be problems because while apparently full-time staff are allocated to a project, they may effectively be part-time because they still have routine work to do.
- ❖ The official project portfolio may not accurately reflect organizational activity if some projects are excluded.
- ❖ The ‘below the line’ projects could in fact consume substantial staff effort and bleed away effort from the official projects.

## Evaluation of Individual Projects:::

- It is important to look at how the feasibility of an individual project can be evaluated.
- Technical assessment of a proposed system consists of evaluating whether the required functionality can be achieved with current affordable technologies.
- The costs of the technology adopted must be taken into account in the cost–benefit analysis.

# Cost benefit analysis (CBA)

- ❖ A process of comparing the projected or estimated costs and benefits associated with a project decision to determine whether it makes sense from a business perspective.
- ❖ If the projected benefits outweigh the costs, you could argue that the decision is a good one to make. If, on the other hand, the costs outweigh the benefits, then a company may want to rethink the decision or project.
- ❖ Even where the estimated benefits will exceed the estimated costs, it is often necessary to decide if the proposed project is the best of several options.
- ❖ Not all projects can be undertaken at any one time and, in any case, the most valuable projects should get most resources.

## **2-Steps of Cost–benefit analysis::::**

**#1. Identifying all of the costs and benefits of carrying out the project and operating the delivered application -**

These include the development costs, the operating costs and the benefits expected from the new system. A new sales order processing system, for example, could only claim to benefit an organization by the increase in sales due to the use of the new system.

**#2. Expressing these costs and benefits in common units -**

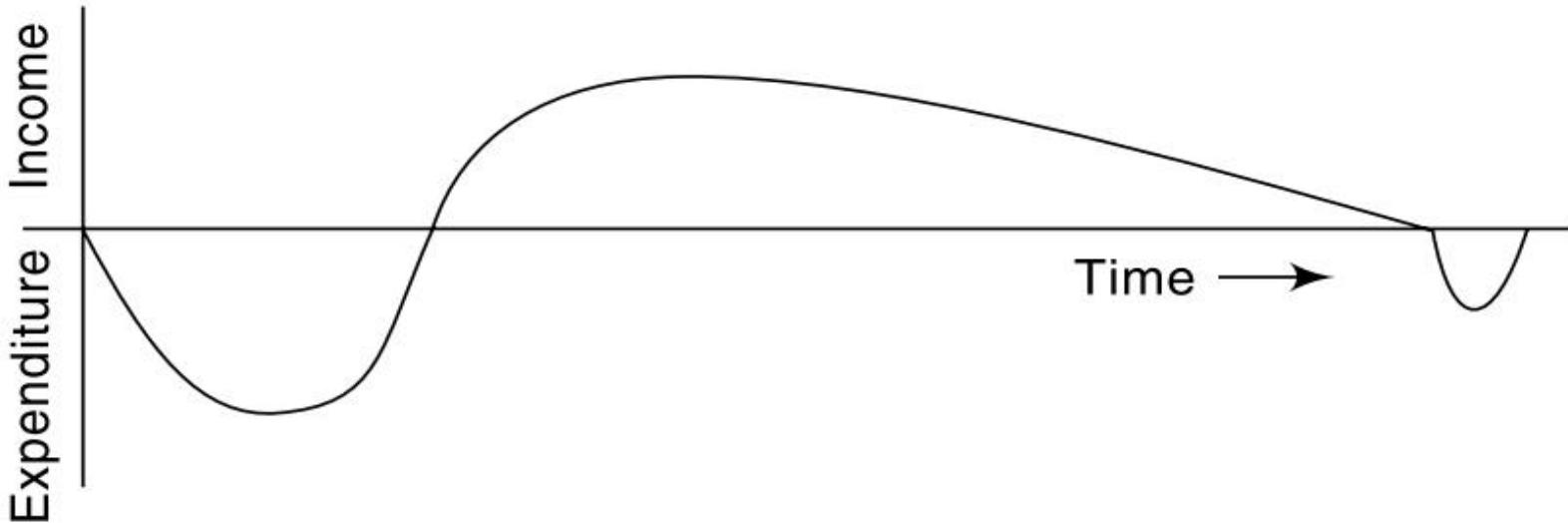
We must express each cost and benefit and the net benefit which is the difference between the two – in money.

**Most direct costs are easy to quantify in monetary terms and can be categorized as:**

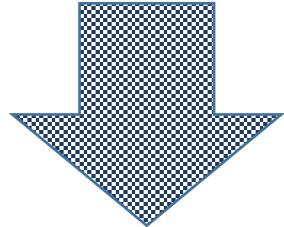
- Development costs, including development staff costs;
- Setup costs, consisting of the costs of putting the system into place, mainly of any new hardware but also including the costs of file conversion, recruitment and staff training;
- Operational costs relating to operating the system after installation

*As important as estimating the overall costs and benefits of a project is producing a **cash flow forecast** which indicates when expenditure and income will take place.*

*Accurate cash flow forecasting is difficult, as it is done early in the project's life cycle (at least before any significant expenditure is committed) and many items to be estimated (particularly the benefits of using software) might be some years in the future.*



## ***Cost–benefit Evaluation Techniques***



*We now take a look at some methods  
for comparing projects on the basis of  
their cash flow forecasts*

*Table illustrates cash flow forecasts for four projects. In each case it is assumed that the cash flows take place at the end of each year. Negative values represent expenditure and positive values income.*

### Four project cash flow projections – figures are end of year totals (£)

Year	Project 1	Project 2	Project 3	Project 4
0	-100,000	-1,000,000	-100,000	-120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000
Net profit	50,000	100,000	50,000	75,000

## Net profit

The net profit of a project is the difference between the total costs and the total income over the life of the project.

- ❖ Project 2 in Table shows the greatest net profit but this is at the expense of a large investment.
- ❖ Moreover, the simple net profit takes no account of the timing of the cash flows. Projects 1 and 3 each have a net profit of £50,000 and therefore, according to this selection criterion, would be equally preferable.
- ❖ The bulk of the income occurs late in the life of project 1, whereas project 3 returns a steady income throughout its life.

## Payback period

The payback period is the time taken to break even or pay back the initial investment. Normally, the project with the shortest payback period will be chosen

- ❖ Its disadvantage as a selection technique is that it ignores the overall profitability of the project – in fact, it totally ignores any income (or expenditure) once the project has broken even.
- ❖ Thus the fact that projects 2 and 4 are, overall, more profitable than project 3 is ignored.

<b>Year</b>	<b>P1</b>	<b>P2</b>	<b>P3</b>	<b>P4</b>	<b>P5</b>
0	-50,000	-100,000	-30,000	-50,000	-10,000
1	20,000	60,000	10,000	5,000	-10,000
2	20,000	50,000	-20,000	30,000	30,000
3	-20,000	5,000	15,000	25,000	20,000
4	10,000	-10,000	1,20,000	-10,000	-5000
5	15,000	10,000	-20,000	20,000	60,000

Determine the Project with:

- ✓ Optimal Net Profit.
- ✓ Least Payback Period

## Return on investment

The return on investment (ROI), also known as the accounting rate of return (ARR), provides a way of comparing the net profitability to the investment required

$$\text{ROI} = \frac{\text{average annual profit}}{\text{total investment}} \times 100$$

- ❖ The return on investment provides a simple, easy-to-calculate measure of return on capital.
- ❖ It suffers from two severe disadvantages. Like the net profitability, it takes no account of the timing of the cash flows. More importantly, this rate of return bears no relationship to the interest rates offered or charged by banks since it takes no account of the timing of the cash flows.

## Net present value

Net present value is a project evaluation technique that takes into account the profitability of a project and the timing of the cash flows that are produced.

- ❖ The present value of any future cash flow may be obtained by applying the following formula:

$$\text{Present value} = \frac{\text{value in year } t}{(1 + r)^t}$$

where  $r$  is the discount rate, expressed as a decimal value, and  $t$  is the number of years into the future that the cash flow occurs

- ❖ The annual rate by which we discount future earnings is known as the discount rate.

## NPV discount factors

Year	Discount rate (%)					
	5	6	8	10	12	15
1	0.9524	0.9434	0.9259	0.9091	0.8929	0.8696
2	0.9070	0.8900	0.8573	0.8264	0.7972	0.7561
3	0.8638	0.8396	0.7938	0.7513	0.7118	0.6575
4	0.8227	0.7921	0.7350	0.6830	0.6355	0.5718
5	0.7835	0.7473	0.6806	0.6209	0.5674	0.4972
6	0.7462	0.7050	0.6302	0.5645	0.5066	0.4323
7	0.7107	0.6651	0.5835	0.5132	0.4523	0.3759
8	0.6768	0.6274	0.5403	0.4665	0.4039	0.3269
9	0.6446	0.5919	0.5002	0.4241	0.3606	0.2843
10	0.6139	0.5584	0.4632	0.3855	0.3220	0.2472
15	0.4810	0.4173	0.3152	0.2394	0.1827	0.1229
20	0.3769	0.3118	0.2145	0.1486	0.1037	0.0611
25	0.2953	0.2330	0.1460	0.0923	0.0588	0.0304

## Applying the discount factors to project 1

<b>Year</b>	<b>Project 1 cash flow (£)</b>	<b>Discount factor @ 10 %</b>	<b>Discounted cash flow (£)</b>
0	-100,000	1.0000	-100,000
1	10,000	0.9091	9,091
2	10,000	0.8264	8,264
3	10,000	0.7513	7,513
4	20,000	0.6830	13,660
5	100,000	0.6209	62,090
Net Profit:	£50,000	NPV: £618	

*It is interesting to note that the net present values for projects 1 and 3 are significantly different – even though they both yield the same net profit and both have the same return on investment. The difference in NPV reflects the fact that, with project 1, we must wait longer for the bulk of the income.*

**Calculate the net present value for each of the projects A, B and C shown in Table using each of the discount rates 8%, 10% and 12%.**

**For each of the discount rates, decide which is the best project.**

Year	Project A (£)	Project B (£)	Project C (£)
0	-8,000	-8,000	-10,000
1	4,000	1,000	2,000
2	4,000	2,000	2,000
3	2,000	4,000	6,000
4	1,000	3,000	2,000
5	500	9,000	2,000
6	500	-6,000	2,000
Net Profit	4,000	5,000	6,000



*Dealing with uncertainty:  
Risk evaluation*

**Every project involves risk of some form. When assessing and planning a project, we are concerned with the risk of the project's not meeting its objectives.**

**A project risk assessment** is a process that aims to gain a deeper understanding of which project tasks, deliverables, or events could influence its success. Through the assessment process, you identify potential threats to your project and analyze consequences in case they occur.

**Once the risk has been identified, project managers need to come up with a mitigation plan or any other solution to counter attack the risk.**

**Managers can plan their strategy based on four steps of risk management which prevails in an organization. Following are the steps to manage risks effectively in an organization:**

- ✓ *Risk Identification*
- ✓ *Risk Quantification*
- ✓ *Risk Response*
- ✓ *Risk Monitoring and Control*

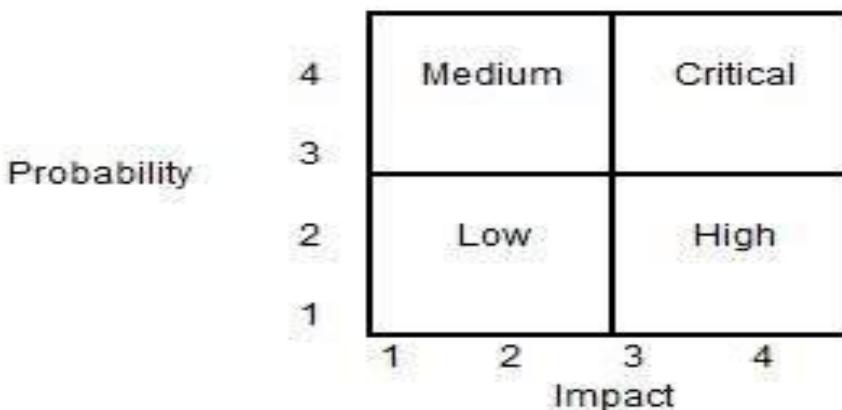
***Project risks prevent the project from being completed successfully, while Business risks denotes that the delivered products are not profitable.***

### **Step 1 - Risk Identification**

- Managers face many difficulties when it comes to identifying and naming the risks that occur when undertaking projects.
- Risks that often impact a project are supplier risk, resource risk and budget risk.
- Supplier risk would refer to risks that can occur in case the supplier is not meeting the timeline to supply the resources required. Resource risk occurs when the human resource used in the project is not enough or not skilled enough. Budget risk would refer to risks that can occur if the costs are more than what was budgeted.

## Step 2 - Risk Quantification

- Risks can be evaluated based on quantity. Project managers need to analyze the likely chances of a risk occurring with the help of a 'project risk matrix'.
- Using the matrix, the project manager can categorize the risk into four categories as Low, Medium, High and Critical. The probability of occurrence and the impact on the project are the two parameters used for placing the risk in the matrix categories.
- As an example, if a risk occurrence is low (probability = 2) and it has the highest impact (impact = 4), the risk can be categorized as 'High'.



## Step 3 - Risk Response

- When it comes to risk management, it depends on the project manager to choose strategies that will reduce the risk to minimal. Project managers can choose between the four risk response strategies, which are outlined below.
- Risks can be avoided
  - Pass on the risk
  - Take corrective measures to reduce the impact of risks
  - Acknowledge the risk

## Step 4 - Risk Monitoring and Control

- Risks can be monitored on a continuous basis to check if any change is made. New risks can be identified through the constant monitoring and assessing mechanisms.

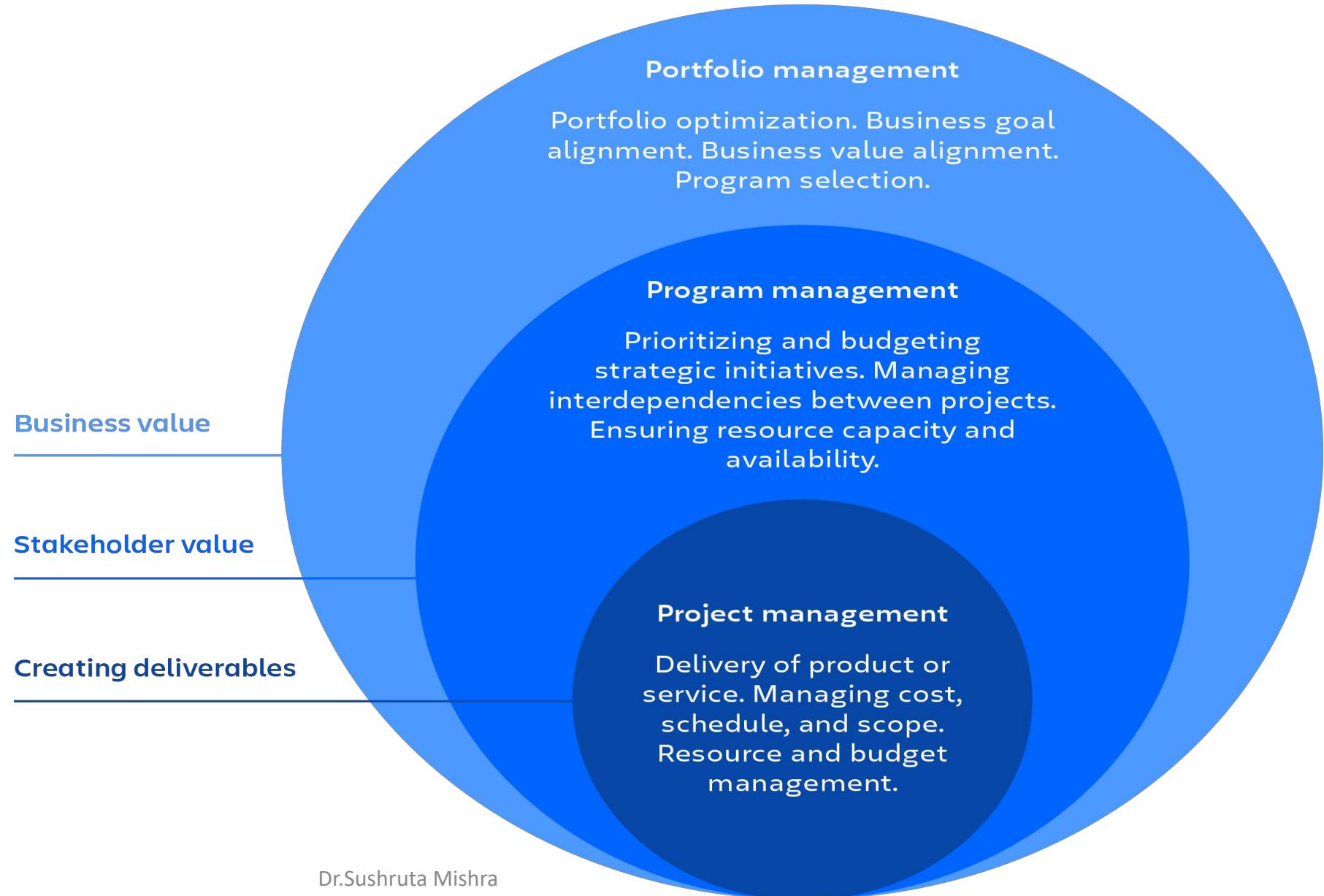
- The project risk matrix may be used as a way of evaluating projects (those with high risks being less favoured) or as a means of identifying and ranking the risks for a specific project.

A fragment of a basic project/business risk matrix for an e-commerce application

Risk	Importance	Likelihood
Client rejects proposed look and feel of site	H	—
Competitors undercut prices	H	M
Warehouse unable to deal with increased demand	M	L
Online payment has security problems	M	M
Maintenance costs higher than estimated	L	L
Response times deter purchasers	M	M

# Programme vs Project Management

**Program management** is the process of managing programs mapped to business objectives that improve organizational performance. Program managers oversee and coordinate the various projects and other strategic initiatives throughout an organization.

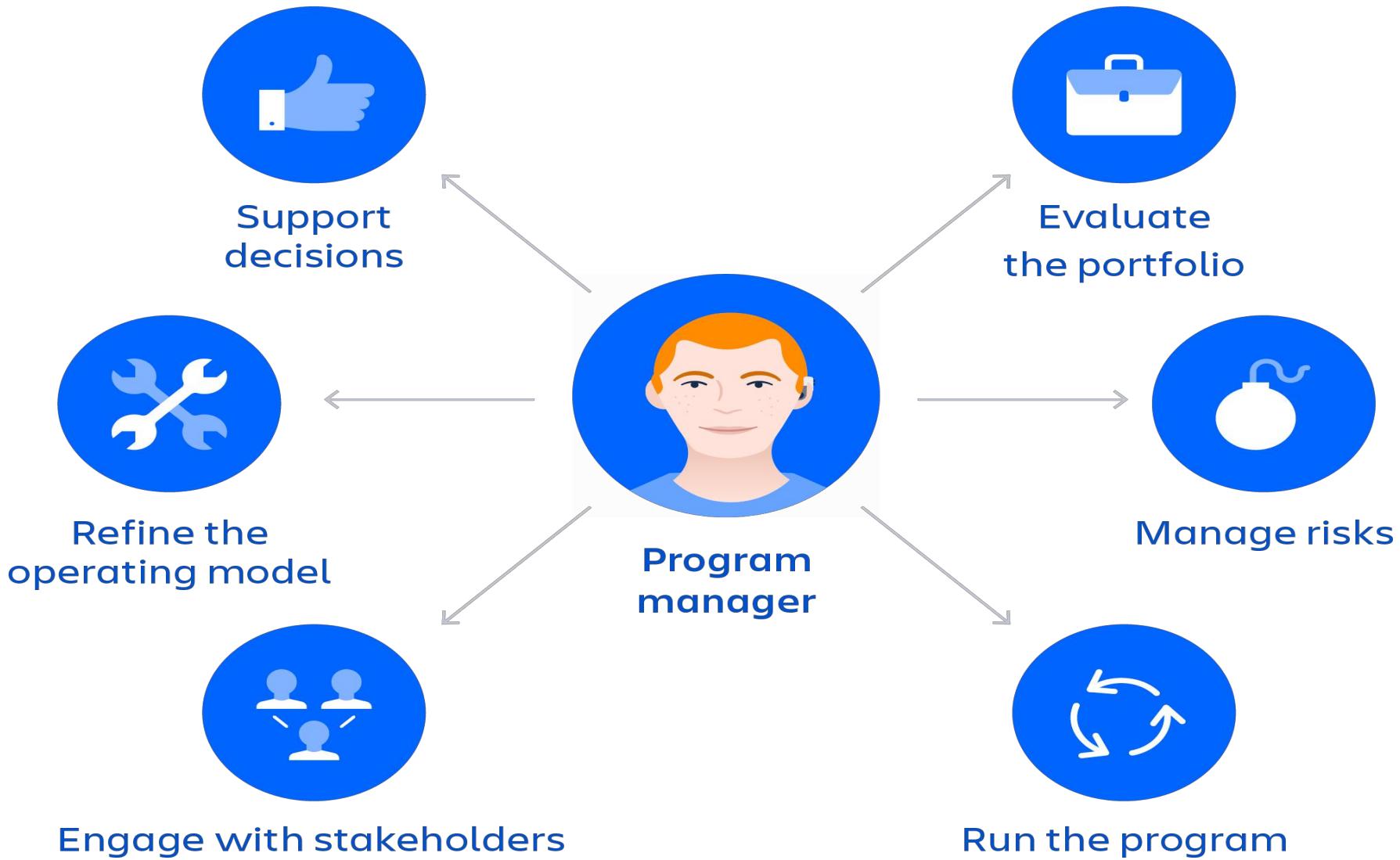


## Projects have:

- A set of tasks with a clear deliverable & a deadline for completion.
- Relates to creating, updating, or reviewing a particular document, process, outcome, or another single unit of work.
- A predefined scope that is limited to a specific output.
- Improves quality, efficiency, cost management, or customer satisfaction in a specific and predetermined way.

## Programs have:

- Unknown deadlines due to the large scope and impact of the work that must be done continuously over a long period of time.
- Multiple deliverables with inter-related dependencies that may continue to evolve based on changing business needs.
- A series of deliverables completed together to increase efficiency, accuracy, reliability, or other business needs.
- The work enables the company to achieve a long-term business goal or initiative that will run in perpetuity.

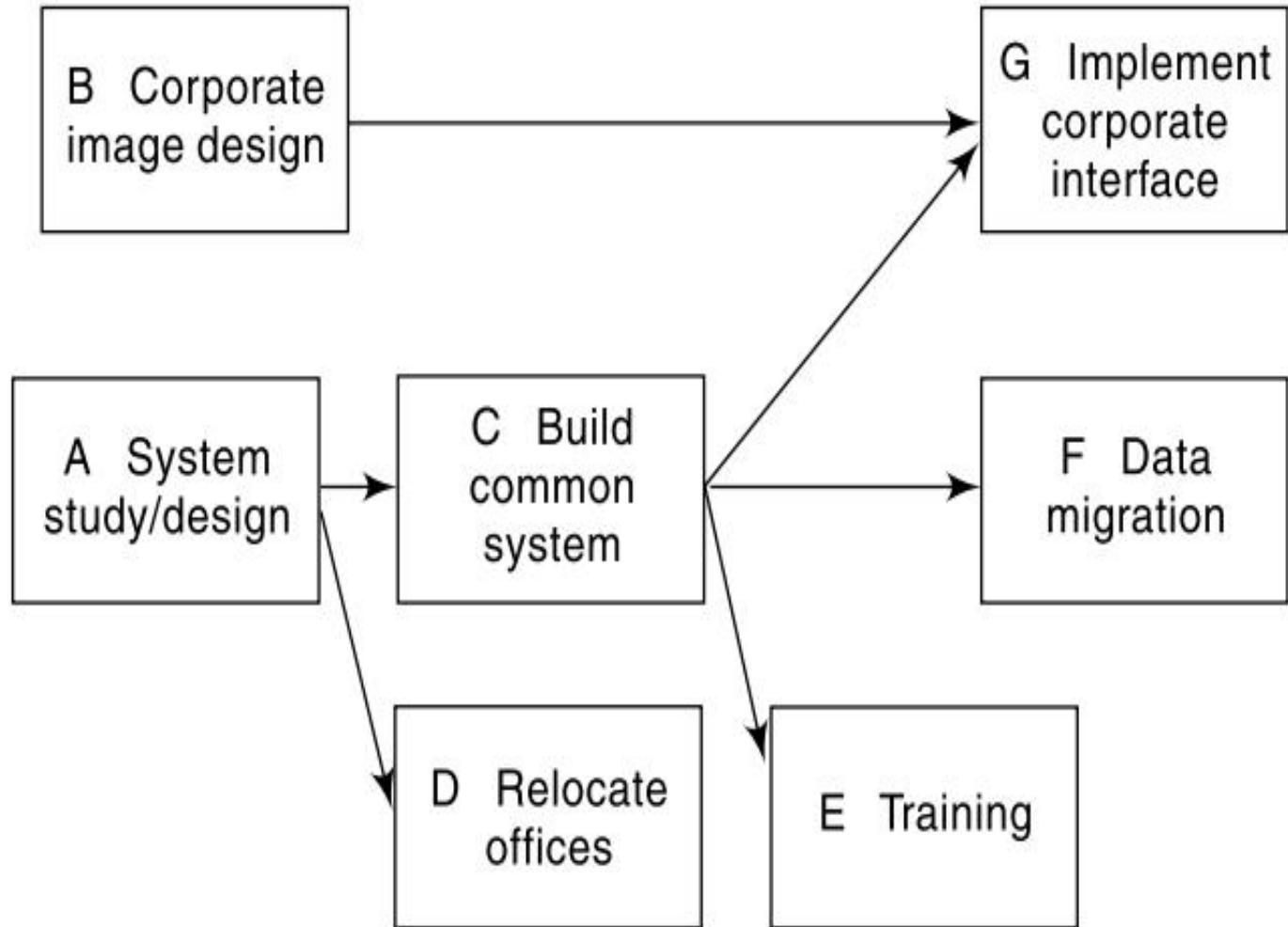


***Project managers lead individual projects to completion, while program managers are in charge of ensuring groups of projects are carried out effectively.***

Program manager	Project manager
Plans strategies	Plans projects
Provides advice to stakeholders	Tracks progress of projects
Review and advise on projects	Allocates resources
Offers audits and QA	Manage risks
Mentorship to project teams	Communicate

# Dependency diagrams

- There will often be physical and technical dependencies between projects. For example, a project to relocate staff from one building to another cannot start until the project to construct the new building has been completed.
- Dependency diagrams at project level, can show these dependencies. However, where projects run concurrently in a programme and products interchange, the dependency diagrams could become quite complicated.



Dependency diagram for a programme to merge two organizations

- A. Systems study/design** - A project is carried out which examines the various existing IT applications in the two old organizations, analyses their functionality, and makes recommendations about how they are to be combined.
- B. Corporate image design** - Independently of Project A, this project is designing the corporate image for the new organization. This would include design of the new logo to be put on company documents.
- C. Build common systems** - Once Project A has been completed, work can be triggered on the construction of the new common ICT applications.
- D. Relocate offices** - This is the project that plans and carries out the physical co-location of the staff in the two former organizations.
- E. Training** - Once staff have been brought together, perhaps with some staff being made redundant, training in the use of the new systems can begin.
- F. Data migration** - When the new, joint, applications have been developed and staff have been trained in their use, data can be migrated from existing databases to the new consolidated database.
- G. Implement corporate interface** - Before the new applications can 'go live', the interfaces, including the documentation generated for external customers, must be modified to conform to the new company image.

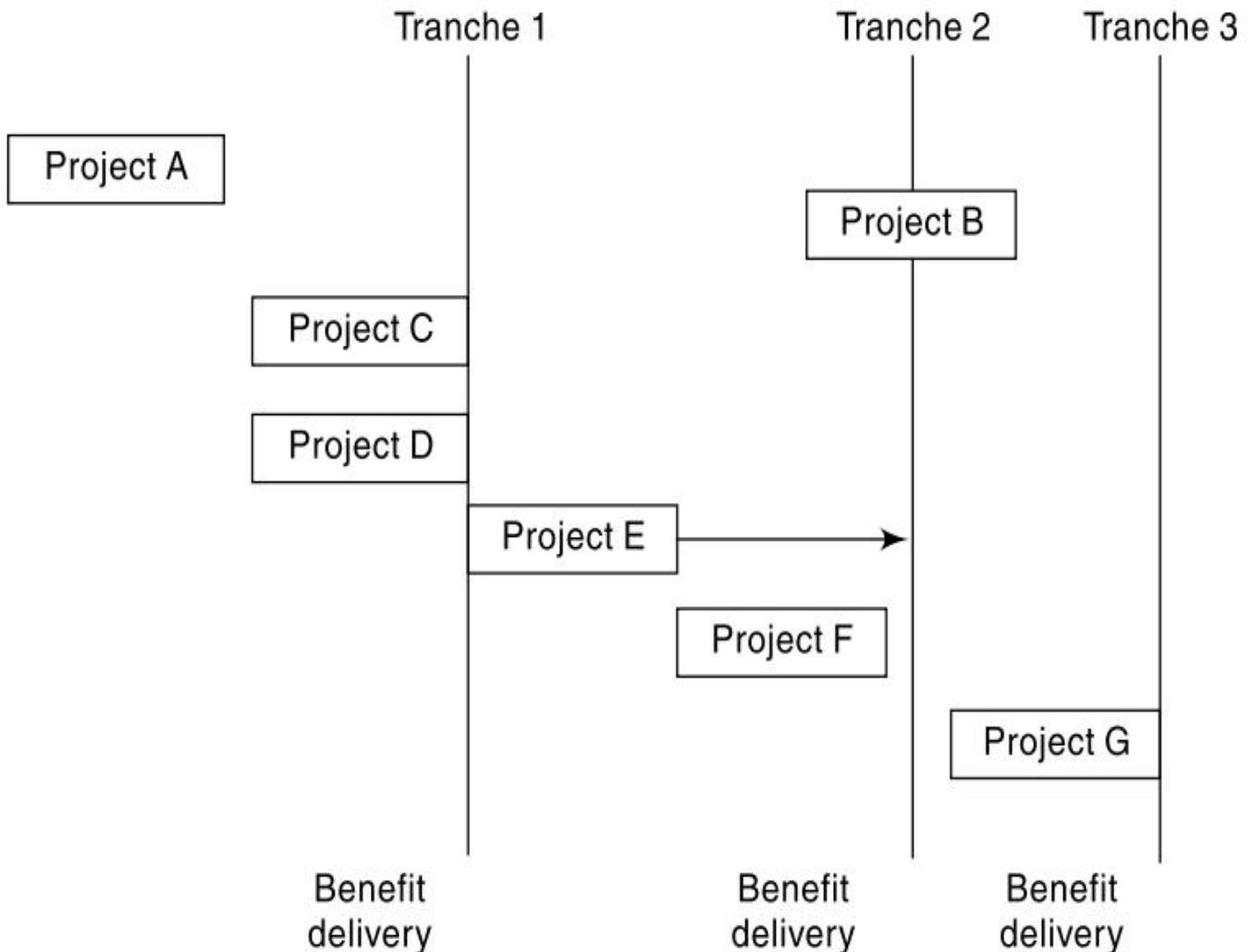
# Delivery planning

*The creation of a delivery dependency diagram would typically lead to the definition of tranches of projects.*

**A tranche** is a group of projects that will deliver their products as one step in the programme.

*The projects in a tranche should combine to provide a coherent new capability or set of benefits for the client.*

*Figure shows how the programme's portfolio of projects can be organized into tranches, each of which delivers some tangible benefits to the user.*



# Resource Allocation in Project Management

- There are many resources which have to be allocated when managing a project, beginning from budget to equipment and tools, to data and the project's plan.
- Resource allocation helps project managers schedule the best team for the job, and optimize how work is assigned. It allows to plan and prepare for the project's implementation or achieving goals. It is also possible to analyze existing threats and risks to the project.
- Effective resource allocation is the process of assigning tasks to the right team members to deliver projects on time and within budget.

## *How to Allocate Resources When Managing a Project?*

- ❖ **Know the scope** – to know what is your project about, what you will need to achieve it, and to be able to properly allocate resources;
- ❖ **Identify resources** – to know which tools, equipment, etc. you will need it completing the project;
- ❖ **Track time** – to have a deep analysis of the progress and current situation as well as be able to control it in the real-time;
- ❖ **Don't look only at the big picture** – the process of working on a project is not done with task allocation. Once you allocate resources you have to keep track of all of them. If you lose at least one tiny detail, your project may fail;
- ❖ **Don't over-allocate** – because your team will experience burnout and their productivity will significantly drop.

# Benefits Management

- ❖ Anytime someone mentions project management, we begin to think of budget development and scheduling development activities. These are definitely important aspects of project management.
- ❖ However, more often than not, the benefits management activities tied to project management are often overlooked but should be a critical aspect of successful project management.
- ❖ This is when a project manager takes time just to make sure that the final project outcome will provide some value to the organization.

## ***What does a benefits management plan include?***

- A benefits management plan should explain, in detail, what the overall benefit of a project is, and what the benefits are at individual milestones. For each benefit, the plan should include the following:
  - **Summary:** Clearly explain what the benefit is and why it's considered good for your organization.
  - **Schedule:** Include a timeline that the benefit can be measured against, as well as an expected date it should be achieved by.
  - **Ownership:** Add a simple statement explaining who benefits from achieving the goal, and why.
  - **Metrics:** Provide a clear explanation of how the benefit will be measured, including its baseline and how it was developed.
  - **Risk Analysis:** Outline potential risks the project faces. Determine who will assume said risks, and how they should be handled.

# *How to create a benefits management plan ???*



**STEP 1 -** Identify and list your project's potential benefits. Include information about who benefits, specifically how they benefit, and why these benefits are good for your organization.

**STEP 2 -** Create SMART goals that are intertwined with the benefits you've listed. For example, if a benefit is that your project will improve your company's market share, include specific details about how that's going to happen, when it's going to happen, and how you'll measure it along the way.

**STEP 3 -** Create a list of potential risks that might deter your project team from realizing the benefits you've written out in step one. Determine how these risks might affect progress, who is accountable for monitoring them, and what to do if they occur.

**STEP 4 -** Use a templated benefits management plan to create documentation for each benefit listed.

**Tangible benefits** can be measured using reliable metrics. These include cost, sales, and other benefits that have a direct impact on business success. They result in an increase in revenue, decreased costs, or increased productivity.

**Intangible benefits** are more difficult to identify and measure. These include benefits such as employee satisfaction, client satisfaction, and awareness of your brand among the general public.

## To carry out benefits management, you must:

- Define expected benefits
- Analyse balance between costs and benefits
- Plan how benefits will be achieved
- Allocate responsibilities for their achievement
- Monitor achievement of benefits

### These benefits might include:

- Mandatory requirement
- Improved quality of service
- Increased productivity
- More motivated workforce
- Internal management benefits
- Risk reduction
- Economies
- Revenue enhancement/acceleration
- Strategic fit

### ***Quantifying benefits:::::***

- Benefits can be Quantified and valued e.g. a reduction of x staff saving Rs.y
- Benefits can be Quantified but not valued e.g. a decrease in customer complaints by x%

Brightmouth College is considering the replacement of the existing payroll service, operated by a third party, with a tailored, off-the-shelf computer-based system. List some of the costs it might consider under the headings of:

- Development costs
- Setup costs
- Operational costs

List some of the benefits under the headings:

- Quantified and valued benefits
- Quantified but not valued
- Identified but not easily valued

For each cost or benefit, explain how, in principle, it might be measured in monetary terms.

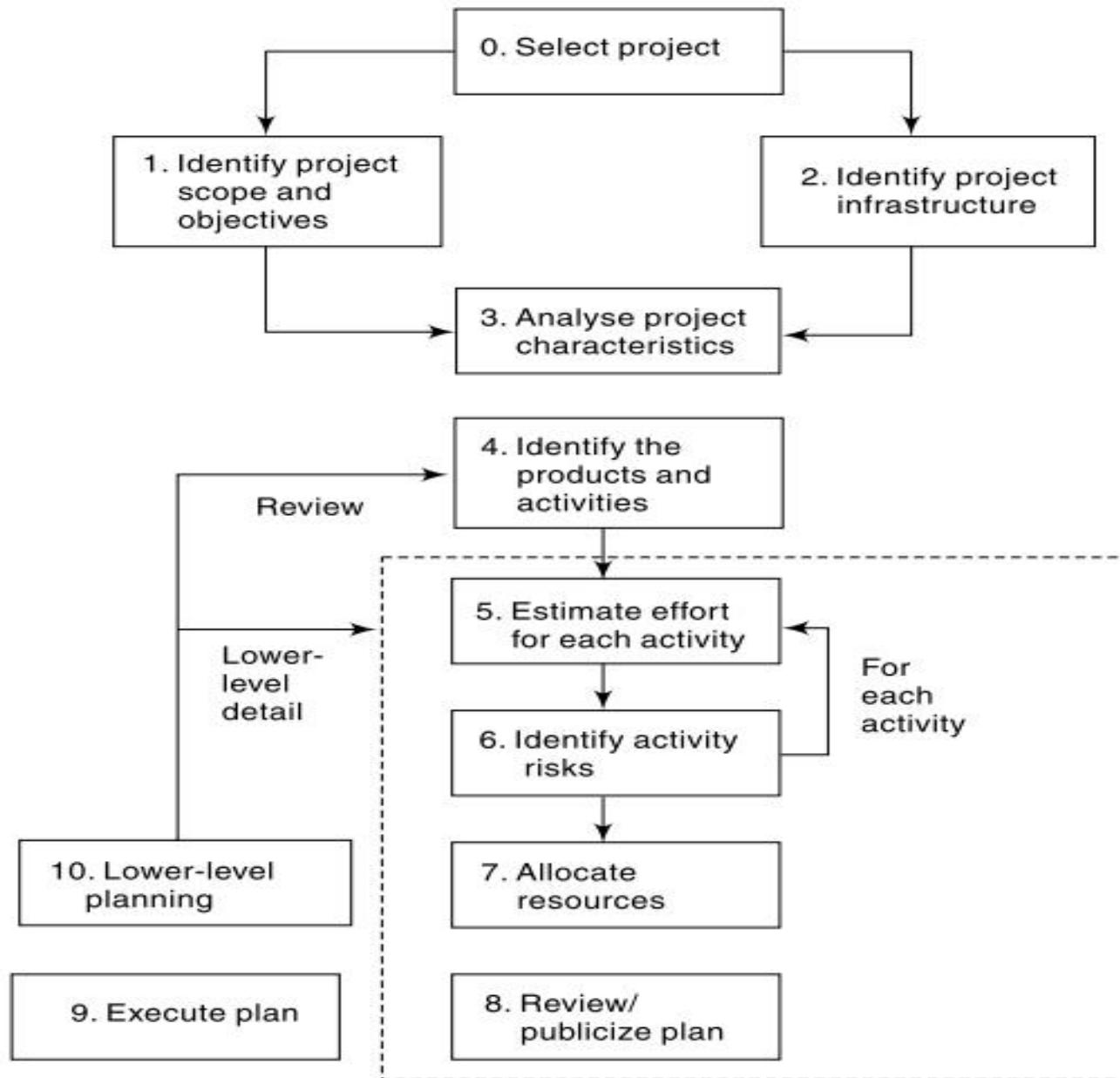
# END

# SPM 3.0

# Project Planning



- In project management, proceeding without a project plan leaves you in danger of overrunning available resources and failing to achieve the client's goals.
- Project management plan, is the document that describes how the project will be executed, monitored, and controlled, and closed. This outlines the objectives and scope of the project.
- Project constraints such as time, scope, and costs are discussed in the project planning process, and mitigation plans are developed after the identification of potential risks.
- By comparing the actual progress with the project plan, you can also monitor the performance of your team and take the necessary steps to improve it.



Step	Activities within step
0	Select project
1	Identify project scope and objectives <ul style="list-style-type: none"> <li>1.1 Identify objectives and measures of effectiveness in meeting them</li> <li>1.2 Establish a project authority</li> <li>1.3 Identify stakeholders</li> <li>1.4 Modify objectives in the light of stakeholder analysis</li> <li>1.5 Establish methods of communication with all parties</li> </ul>
2	Identify project infrastructure <ul style="list-style-type: none"> <li>2.1 Establish relationship between project and strategic planning</li> <li>2.2 Identify installation standards and procedures</li> <li>2.3 Identify project team organization</li> </ul>
3	Analyse project characteristics <ul style="list-style-type: none"> <li>3.1 Distinguish the project as either objective- or product-driven</li> <li>3.2 Analyse other project characteristics</li> <li>3.3 Identify high-level project risks</li> <li>3.4 Take into account user requirements concerning implementation</li> <li>3.5 Select general life-cycle approach</li> <li>3.6 Review overall resource estimates</li> </ul>
4	Identify project products and activities <ul style="list-style-type: none"> <li>4.1 Identify and describe project products (including quality criteria)</li> <li>4.2 Document generic product flows</li> <li>4.3 Recognize product instances</li> <li>4.4 Produce ideal activity network</li> <li>4.5 Modify ideal to take into account need for stages and checkpoints</li> </ul>
5	Estimate effort for each activity <ul style="list-style-type: none"> <li>5.1 Carry out bottom-up estimates</li> <li>5.2 Revise plan to create controllable activities</li> </ul>
6	Identify activity risks <ul style="list-style-type: none"> <li>6.1 Identify and quantify activity-based risks</li> <li>6.2 Plan risk reduction and contingency measures where appropriate</li> <li>6.3 Adjust plans and estimates to take account of risks</li> </ul>
7	Allocate resources <ul style="list-style-type: none"> <li>7.1 Identify and allocate resources</li> <li>7.2 Revise plans and estimates to take account of resource constraints</li> </ul>
8	Review/publicize plan <ul style="list-style-type: none"> <li>8.1 Review quality aspects of project plan</li> <li>8.2 Document plans and obtain agreement</li> </ul>
9/10	Execute plan/lower levels of planning <p>This may require the reiteration of the planning process at a lower level</p>

## Step 1::: Establish project scope and objectives

- ✓ Identify objectives and measures of effectiveness  
**'how do we know if we have succeeded?'**
- ✓ Establish a project authority  
**'who is the boss?'**
- ✓ Identify all stakeholders in the project and their interests  
**'who will be affected/involved in the project?'**
- ✓ Modify objectives in the light of stakeholder analysis  
**'do we need to do things to win over stakeholders?'**
- ✓ Establish methods of communication with all parties  
**'how do we keep in contact?'**

## Step 2::: Establish project infrastructure

- ✓ Establish link between project and any strategic plan  
**'why did they want the project?'**
  
- ✓ Identify installation standards and procedures  
**'what standards do we have to follow?'**  
  
**(change control and configuration management standards,  
quality standards and procedure manuals, measurement)**
  
- ✓ Identify project team organization  
**'where do I fit in?'**  
**(SW developers, business analyst, business-to-customer  
web application group, database group)**

## Step 3::: Analysis of project characteristics

- ✓ Distinguish the project as either objective or product-based.  
Is there more than one way of achieving success?  
**(tends to be more product driven and the underlying objectives always remain and must be respected)**
- ✓ Analyze other project characteristics (including quality based ones)  
what is different about this project?  
**(information system, process control system, safety critical)**
- ✓ Identify high level project risks  
**'what could go wrong?'**  
**'what can we do to stop it?'**
- ✓ Take into account user requirements concerning implementation
- ✓ Select general life cycle approach  
**waterfall? Increments? Prototypes?**
- ✓ Review overall resource estimates  
**'does all this increase the cost?'**

## Step 4::: Identify project products and activities

- ✓ **Identify and describe project products - ‘what do we have to produce?’**
- ❖ Identifying all things the project is to create helps us to ensure that all the activities we need to carry out are accounted for. Some of these products will be handed over to the client at the end of the project – these are deliverables.
- ❖ Other products might not be in the final configuration, but are needed as intermediate products used in the process of creating the deliverables.
- ❖ The products will form a hierarchy. The main products will have sets of component products which in turn may have sub-component products, and so on. These relationships can be documented in a Product Breakdown Structure (PBS)

### Products::::::::::

- The result of an activity could be (among other things)
  - *physical thing ('installed pc')*,
  - *a document ('logical data structure')*
  - *a person ('trained user')*
  - *a new version of an old product ('updated software')*
- The following are NOT normally products:
  - *activities (e.g. 'training')*
  - *events (e.g. 'interviews completed')*
  - *resources and actors (e.g. 'software developer')*
- Products CAN BE deliverable or intermediate

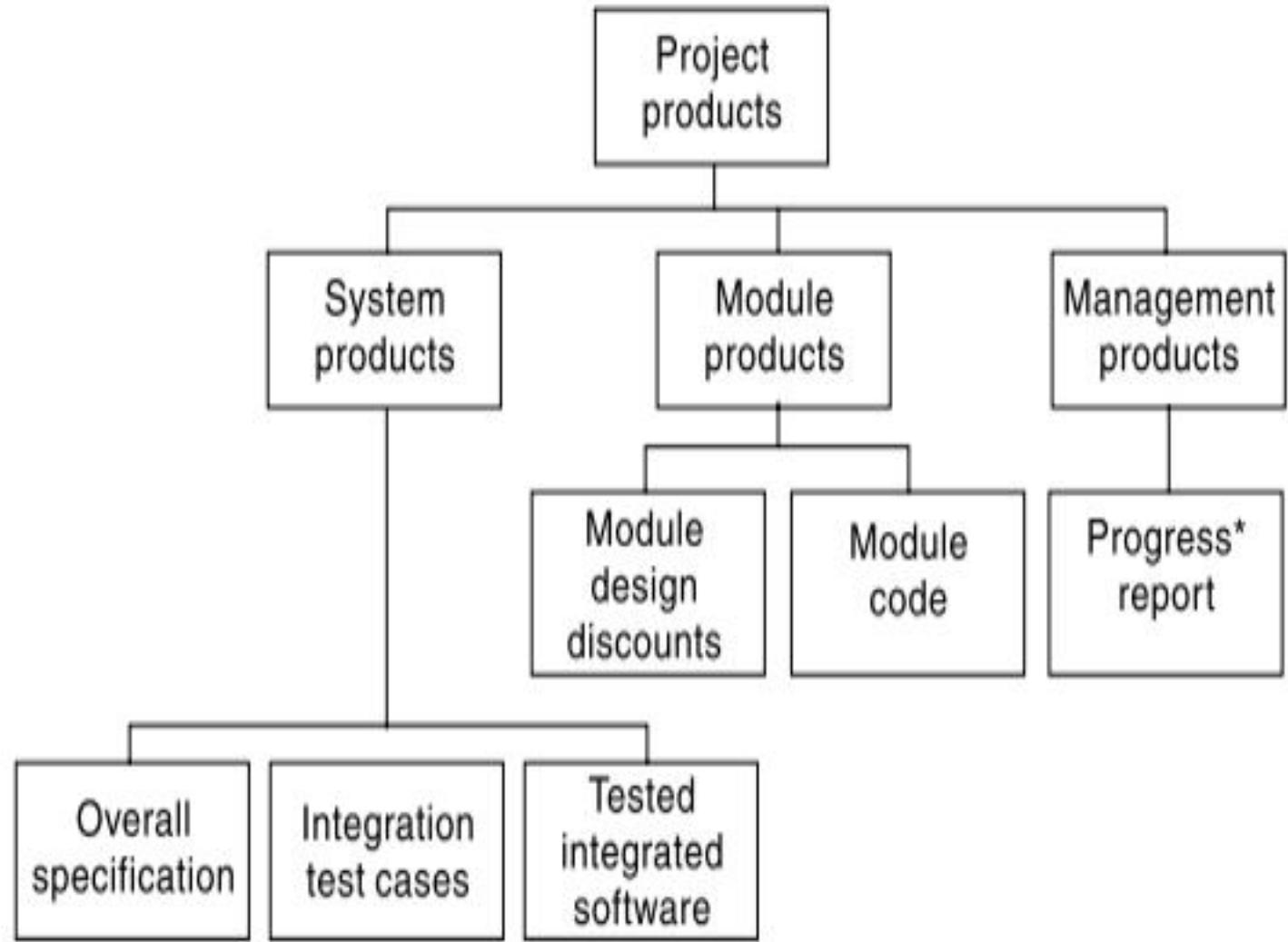
### Product description (PD)::::::::::

- Product identity
- Description - what is it?
- Derivation - what is it based on?
- Composition - what does it contain?
- Relevant standards
- Quality criteria whether the product is acceptable

**Some products are created from scratch, for example new software components. A product could quite easily be a document, such as a software design document.**

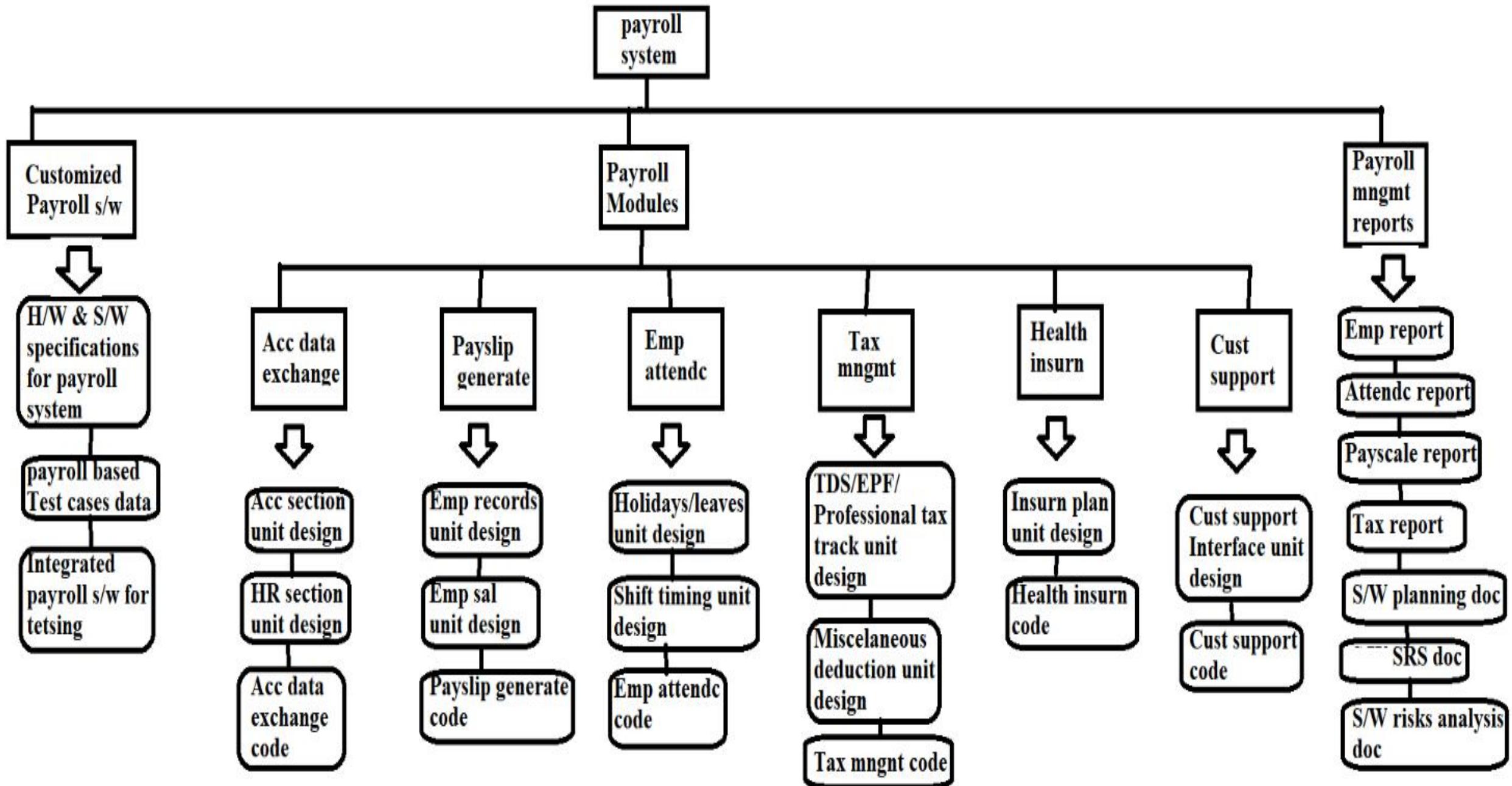
**It might be a modified version of something that already exists, such as an amended piece of code. A product could even be a person, such as a ‘trained user’, a product of the process of training.**

**Always remember that a product is the result of an activity. A common error is to identify as products things that are really activities, such as ‘training’, ‘design’ and ‘testing’.**



A fragment of a Product Breakdown Structure for a system development task

**What would be the product breakdown structure of  
the deliverables of the vendor who would develop the  
Brightmouth College payroll software by customizing  
one of its existing products?**



✓ Document generic product flows

A program design must be created before the program can be written and program specification must exist.

The relationships can be portrayed in a Product Flow Diagram (PFD).

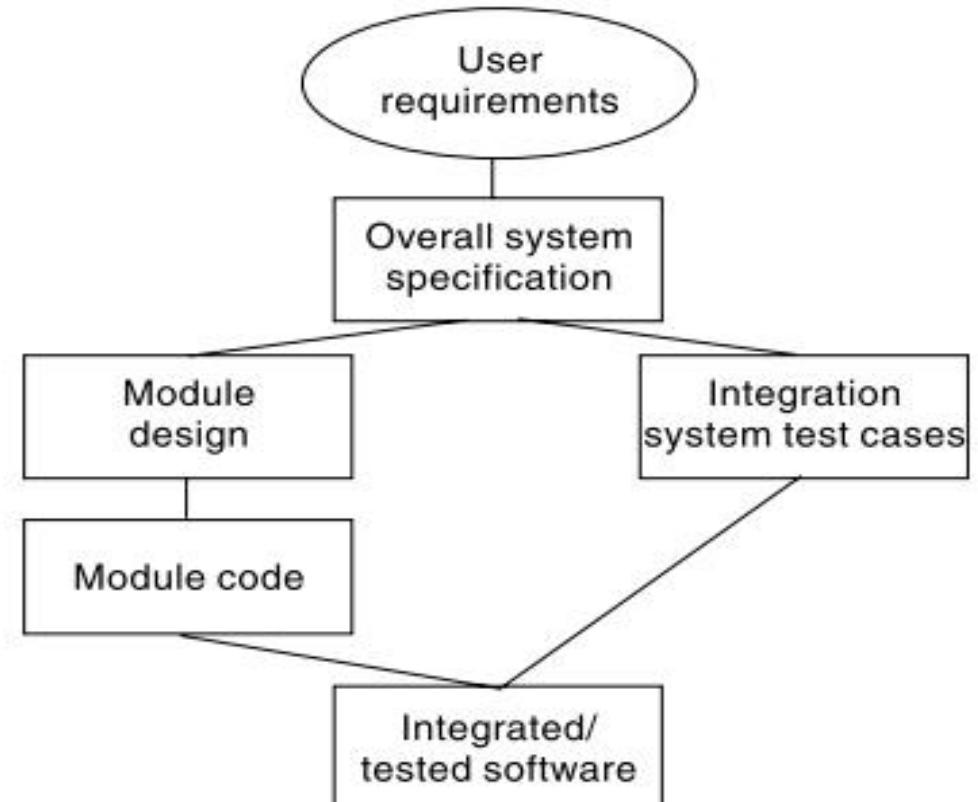
Flow on the diagram to be from top to bottom and left to right.

User requirement is an oval.

Note that the 'flow' in the diagram is assumed to be from top to bottom and left to right.

In the example, 'user requirements' is in an oval which means that it is used by the project but is not created by it.

It is often convenient to identify an overall product at the bottom of the diagram, in this case 'integrated/tested software', into which all the other products feed.



PFD for a software development task

✓ Recognize product instances

The PBS and PFD will probably have identified generic products e.g. 'software modules'

It might be possible to identify specific instances e.g. 'module A', 'module B' ...

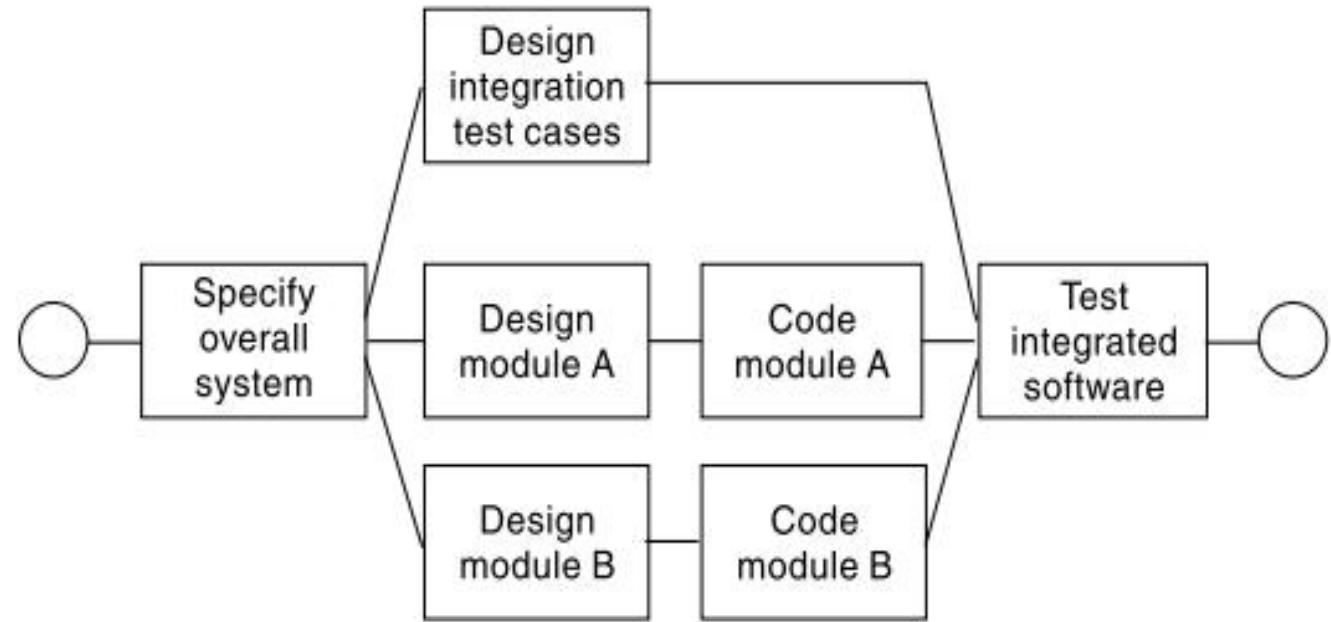
But in many cases this will have to be left to later, more detailed, planning

✓ Produce ideal activity network

Identify the activities needed to create each product in the PFD

More than one activity might be needed to create a single product

Draw up activity network



An example of an activity network

✓ Add check-points if needed

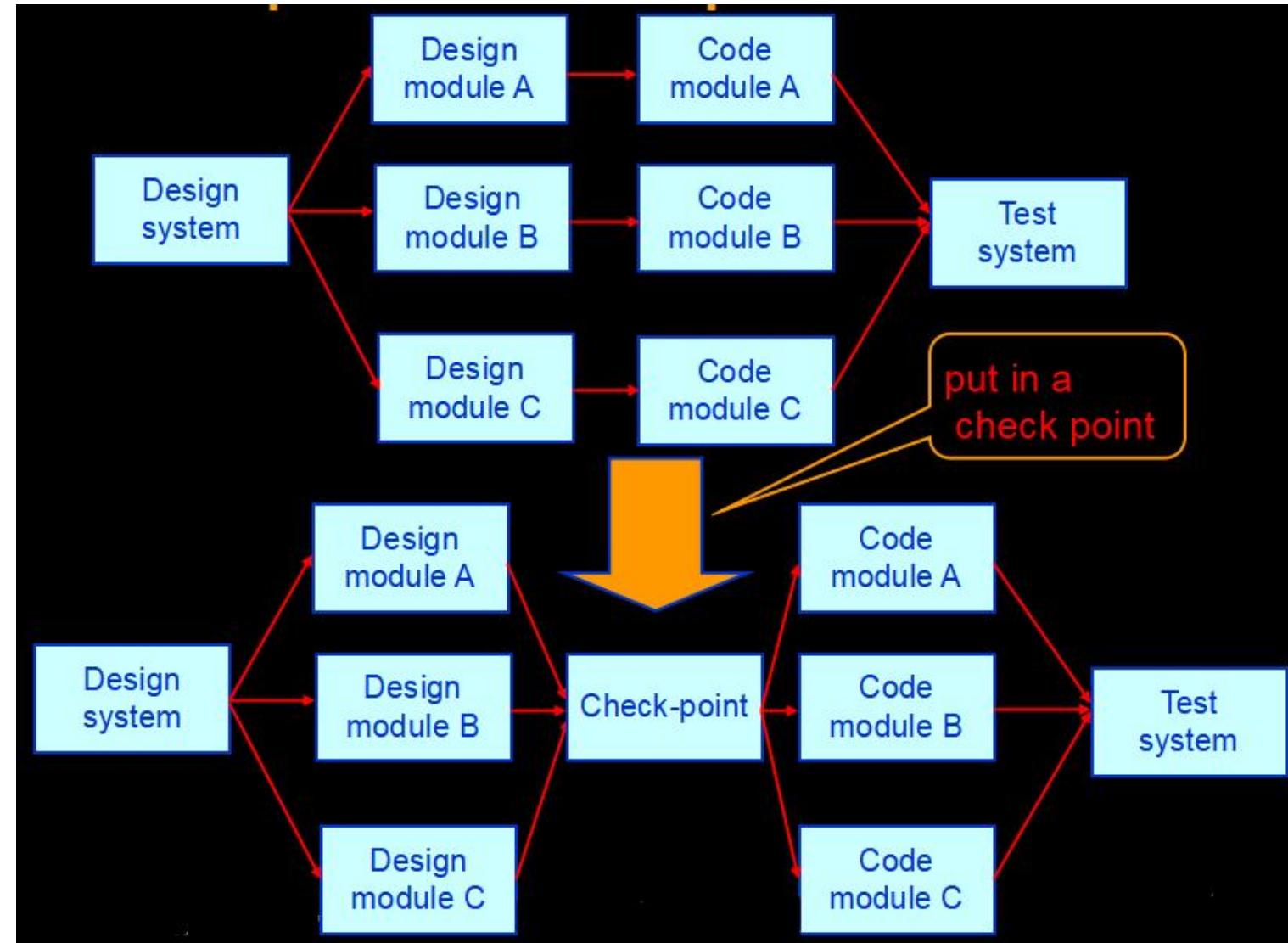
An activity will start as soon as the preceding ones are completed.

Sometimes as per need, divide the project into stages and introduce a check point activities.

Check point ensures the preceding activities together are complete and compatible.

This may delay work on some elements of the project.

There should be a trade-off between efficiency and quality



## Step 5::: Estimate effort for each activity

- ✓ Carry out bottom-up estimates

At this point, estimates of the staff effort required, the probable elapsed time and the non-staff resources needed for each activity will need to be produced.

Effort is the amount of work that needs to be done. If a task requires three members of staff to work for two full days each, the effort expended is six days.

Elapsed time is the time between the start and end of a task. In our example above, if the three members of staff start and finish at the same time then the elapsed time for the activity would be two days.

- ✓ Revise plan to create controllable activities

(If an activity involving system testing is to take 12 weeks, it is difficult after six weeks to judge whether 50% work is completed.)

Break up very long activities into a series of smaller ones

Bundle up very short activities (create check lists?)

## Step 6::: Identify activity risks

- ✓ Identify and quantify risks for activities

A project plan will be based on a huge number of assumptions, and so some way of picking out the risks that are most important is needed. The damage that each risk could cause and the likelihood of it occurring have to be gauged.

- ✓ Plan risk reduction and contingency measures

It may be possible to avoid or at least reduce some of the identified risks. On the other hand, contingency plans specify action that is to be taken if a risk materializes.

For example, a contingency plan could be to use contract staff if a member of the project team is unavailable at a key time because of serious illness.

- ✓ Adjust overall plans and estimates to take account of risks

e.g. add new activities which reduce risks associated with other activities e.g. training, pilot trials, information gathering

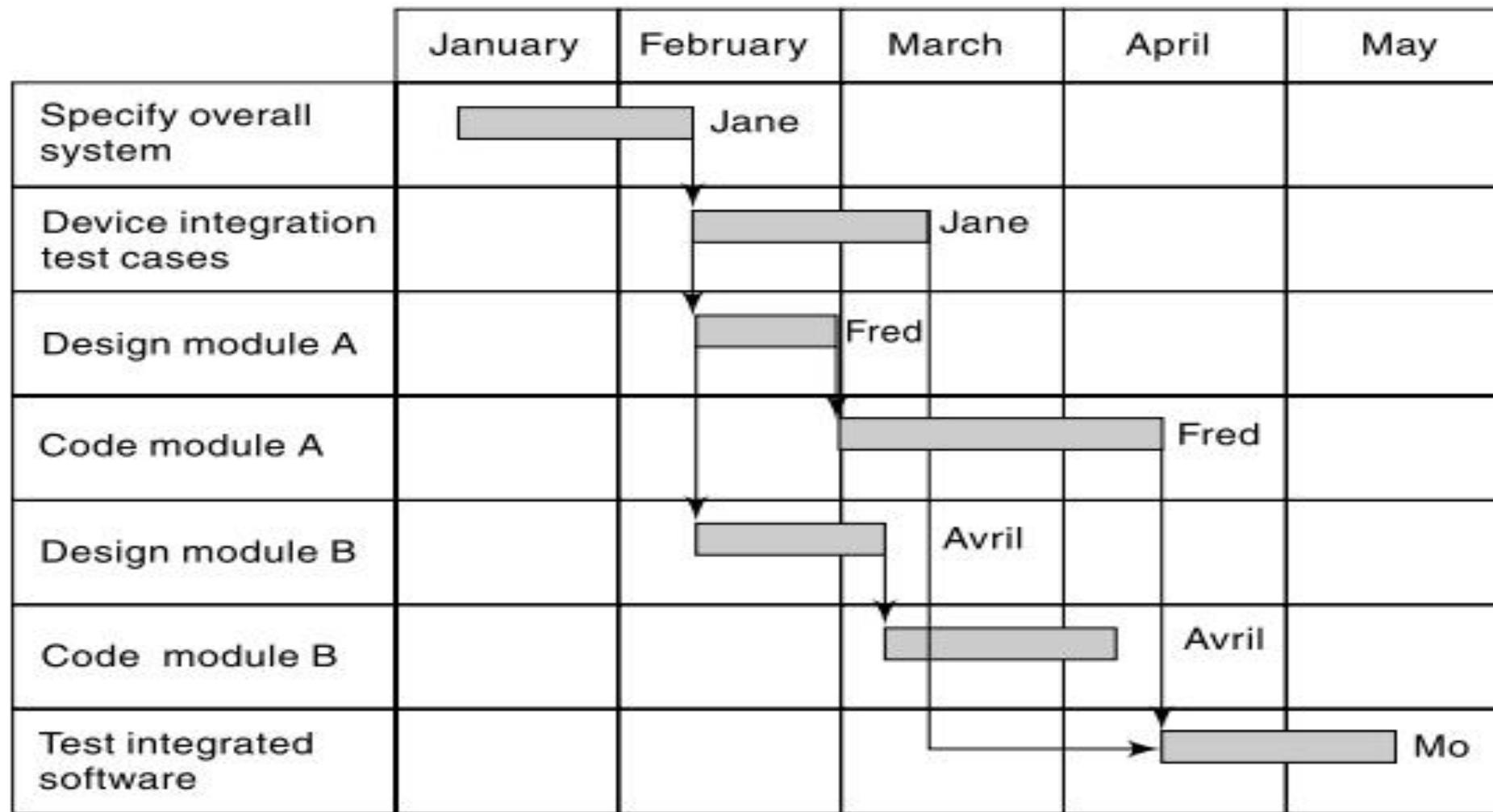
## Step 7::: Allocate resources

- ✓ Identify and allocate resources to activities

The type of staff needed for each activity is recorded. The staff available for the project are identified and are provisionally allocated to tasks.

- ✓ Revise plans and estimates to take into account

Resource constraints e.g. staff not being available until a later date non-project activities



Gantt chart showing when staff will be carrying out tasks

## Step 8::: Review/publicize plan

- ✓ Review quality aspects of project plan  
**(ensure the quality criteria has been ensured before completion of the project)**
  
- ✓ Document plan and obtain agreement  
**It is important that the plans be carefully documented and that all the parties to the project understand and agree to the commitments required of them in the plan.**

## **Step 9 and 10:::::: Execute plan and create lower level plans**

**Once the project is under way, plans will need to be drawn up in greater detail for each activity as it becomes due.**

**Detailed planning of the later stages will need to be delayed because more information will be available nearer the start of the stage.**

# Key points

- ❖ Establish your objectives
- ❖ Think about the characteristics of the project
- ❖ Discover/set up the infrastructure to support the project (including standards)
- ❖ Identify products to be created and the activities that will create them
- ❖ Allocate resources
- ❖ Set up quality processes

# END

# SPM 4.0

# **Selection of an appropriate project approach**



vs



Build

Buy

*When it comes to developing a new software system, one of the first decisions to be made is whether to build the system in-house or buy a pre-existing system.*

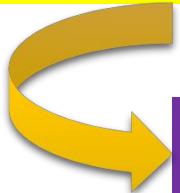
*With the build option, the organization constructs the IT capability needed from scratch. The organization acquires an already-existing IT capability from a software vendor or other external sources with the buy option.*

**The development of software in-house usually means that:**

- The developers and the users belong to the same organization;
- The application will slot into a portfolio of existing computer-based systems;
- The methodologies and technologies are largely dictated by organizational standards and policies, including the existing enterprise architecture.

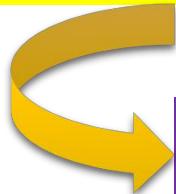
*They would need to review the methodologies and technologies to be used for each individual project. This decision-making process has been called technical planning by some, although here we use the term project analysis.*

# Pros and cons of Building Software in-house Solution

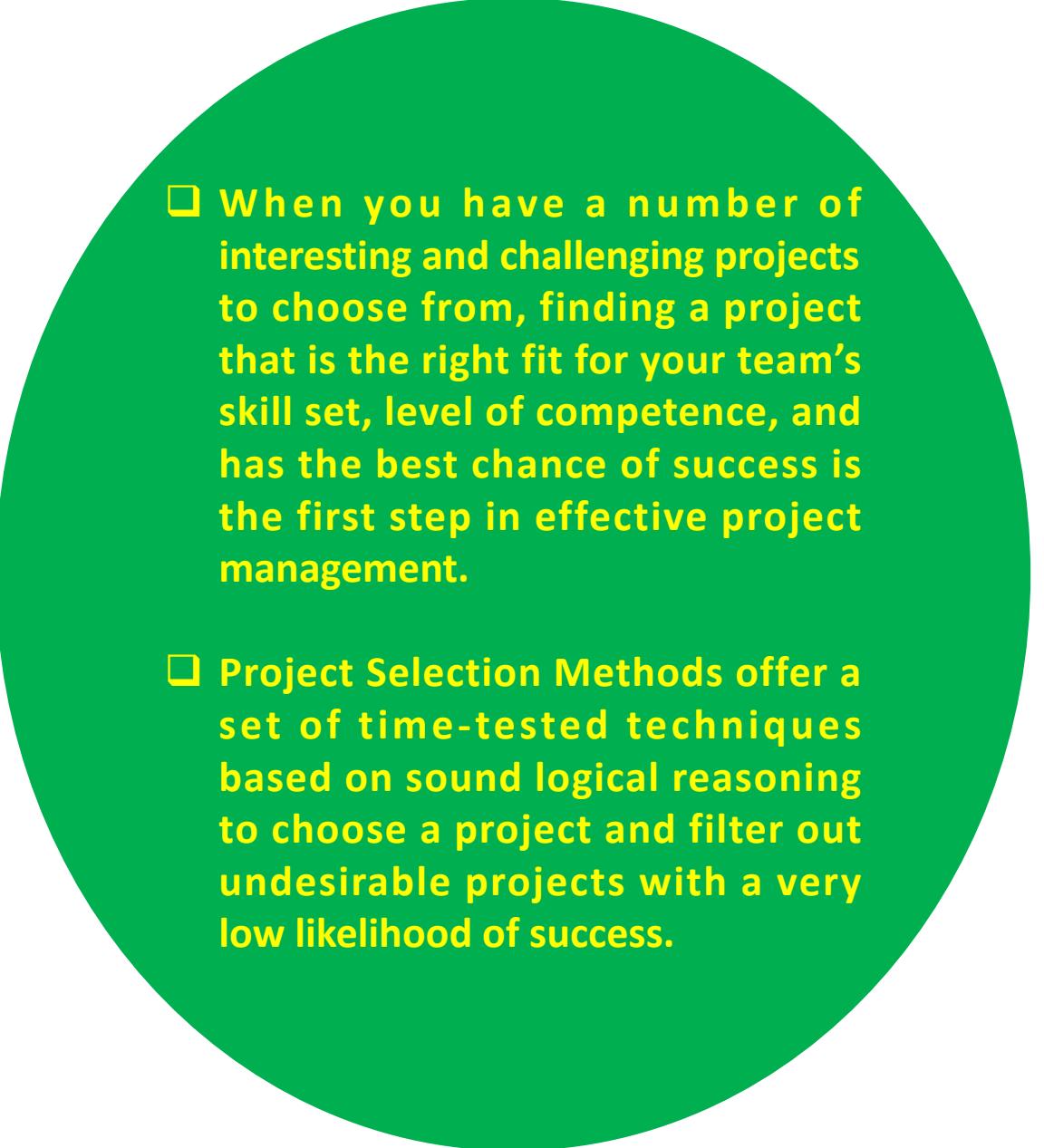


- ❖ Building software in-house generally requires less upfront investment.
- ❖ Custom-built software also offers greater flexibility than packaged solutions.
- ❖ Building custom software in-house also gives you more control over security and compliance.
- ❖ On the downside, building software in-house generally takes longer and requires more internal resources than purchasing an off-the-shelf product. You will have to arrange time and money to design, develop, test, and deploy your software. And you will need to staff your own in-house team, which can be costly and time-consuming.

# Pros and cons of Buying (off-the-shelf) Software Solution



- ❖ Purchasing an off-the-shelf software package generally requires less time and money than building software in-house.
- ❖ The off-the-shelf custom solution also offers more features than custom-built software.
- ❖ it's usually easier to use than custom-built software. Packaged software is typically designed with a user-friendly interface that is easy to navigate.
- ❖ One disadvantage is that you may not be able to get the exact software that you want or need. Packaged software is designed for a wide range of users, so it may not include all the features that you require. Another disadvantage is that custom software may not be as secure as software that you build in-house.

- 
- When you have a number of interesting and challenging projects to choose from, finding a project that is the right fit for your team's skill set, level of competence, and has the best chance of success is the first step in effective project management.
  - Project Selection Methods offer a set of time-tested techniques based on sound logical reasoning to choose a project and filter out undesirable projects with a very low likelihood of success.
- ❖ Look at risks and uncertainties e.g.
    - are requirement well understood?
    - are technologies to be used well understood?
  - ❖ Look at the type of application being built e.g.
    - information system? embedded system?
    - criticality? differences between target and development environments?
  - ❖ Clients' own requirements
    - need to use a particular method

## **Analysis of critical project characteristics:::::**

**Data-oriented or Process-oriented system to be implemented?** Data-oriented systems generally mean information systems that will have a substantial database. Process-oriented systems refer to embedded control systems.

**Will the software be a general tool or application specific?** An example of a general tool would be a spreadsheet or a word processing package. An application-specific package could be, for example, an airline seat reservation system.

**Are there specific tools available for implementing the particular type of application? For example:-**

----- does it involve concurrent processing?

----- will the system to be created be knowledge-based?

----- will the system to be produced make heavy use of computer graphics?

**Is the system to be created safety critical?** For instance, could a malfunction in the system endanger human life? If so, among other things, testing would become very important.

**Is the system designed to carry out predefined services or to be engaging and entertaining?** With software designed for entertainment, design and evaluation will need to be carried out differently from more conventional software products.

**Hardware/software environment in which system will operate?** The environment in which the final software will operate could be different from that in which it is to be developed.

How would you categorize each of the following systems according to the classification above?

- (a) a payroll system;
- (b) a system to control a bottling plant;
- (c) a system which holds details of the plans of plant used by a water company to supply water to consumers;
- (d) a software package to support project managers;
- (e) a system used by lawyers to access case law relating to company taxation.

**A payroll system:** Data-oriented, application specific, generic tools, safety critical, predefined services, hardware/software environment identical.

**A Bottling plant control:** Process-oriented, application specific, specific features like (high bph, automatic, scalability, durable etc), safety critical, predefined services, hardware/software environment identical.

**A Water supplier system:** Both Data-oriented & Process-oriented, application specific, specific features like sustainability, scalability, fault tolerance etc), safety critical, predefined services, hardware/software environment identical.

**A project management software:** Data-oriented, generic, specific features like Time tracking, Reporting, budgeting, Billing & quotes), safety critical voluntary, engaging to various needs, diverse hardware/software environment.

## Identify high-level project risks

Until we have analysed the users' requirements in detail we cannot estimate the effort needed to build a system to meet those requirements.

The greater the uncertainties at the beginning, the greater the risk that the project will be unsuccessful.

Uncertainty can be associated with the products, processes, or resources of a project.

**Product uncertainty:** How well are the requirements understood? The users themselves could be uncertain about what a proposed information system is to do. The government, say, might introduce a new form of taxation but its detailed operation might not be known until case law has been built up.

**Process uncertainty:** The project under consideration might be the first where an organization is using an approach like extreme programming (XP) or a new application-building tool. Any change in the way that the systems are developed introduces uncertainty.

**Resource uncertainty:** The main area of uncertainty here is likely to be the availability of staff of the right ability and experience. The larger the number of resources needed or the longer the duration of the project, the more inherently risky it will be.

*Some factors such as continually changing requirements increase uncertainty, while others for instance, software size also increase complexity. Different strategies are needed to deal with the two distinct types of risks.*

## Select general life-cycle approach

- Control systems: A real-time system will need to be implemented using an appropriate methodology.
- Availability of users: Where the software is for the general market rather than application and user specific, then a methodology which assumes that identifiable users exist who can be quizzed about their needs would have to be thought about with caution.
- Specialized techniques: For example, expert system shells and logic-based programming languages have been invented to expedite the development of knowledge-based systems.
- Hardware environment: The environment in which the system is to operate could put constraints on the way it is to be implemented.
- Safety-critical systems: Where safety and reliability are essential, this might justify the additional expense of a formal specification.
- Imprecise requirements: Uncertainties or a novel hardware/software platform mean that a prototyping approach should be considered.

# Software Processes and Process Models

- ❖ A software product development process usually starts when a request for the product is received from the customer. For a generic product, the marketing department of the company is usually considered as the customer. This expression of need for the product is called product inception.
- ❖ From the inception stage, a product undergoes a series of transformations through a few identifiable stages until it is fully developed and released to the customer. After release, the product is used by the customer and during this time the product needs to be maintained for fixing bugs and enhancing functionalities. This stage is called the maintenance stage.
- ❖ When the product is no longer useful to the customer, it is retired. This set of identifiable stages through which a product transits from inception to retirement form the life cycle of the product. The software life cycle is also commonly referred to as Software Development Life Cycle (SDLC) and software process.
- ❖ A life cycle model (also called a process model) of a software product is a graphical or textual representation of its life cycle. Additionally, a process model may describe the details of various types of activities carried out during the different phases and the documents produced.

# Selection Process Parameters for a Software Process Model::::::

- Requirements characteristics

- Reliability of Requirements*

- Types & Number of requirements*

- Can the requirements be defined at an early stage*

- Development team

- Team size & Environment*

- Experience of developers on similar type of projects*

- Domain knowledge of developers*

- Availability of training*

- User involvement in the project

- Expertise of user in project*

- Involvement of user in all phases of the project*

- Experience of user in similar project in the past*

- Project type and associated risk

- Stability of funds*

- Tightness of project schedule*

- Availability of resources*

- Type & Size of project*

- Expected duration for the completion of project*

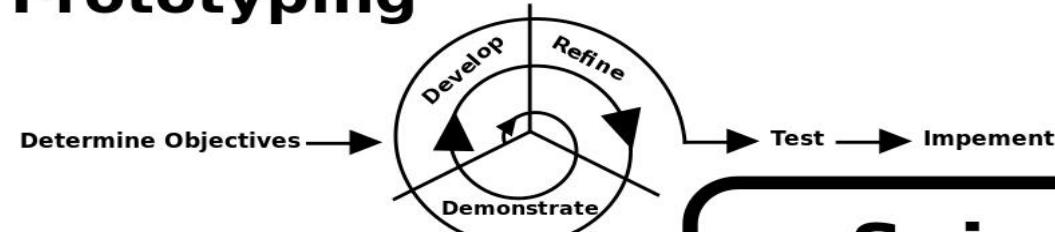
- Complexity of the project*

- Associated risks involved*

# Choice of Process Models

- 'Waterfall' also known as 'one-shot', 'once-through'
- Incremental delivery
- Evolutionary development
- Also use of 'agile methods' e.g. extreme programming

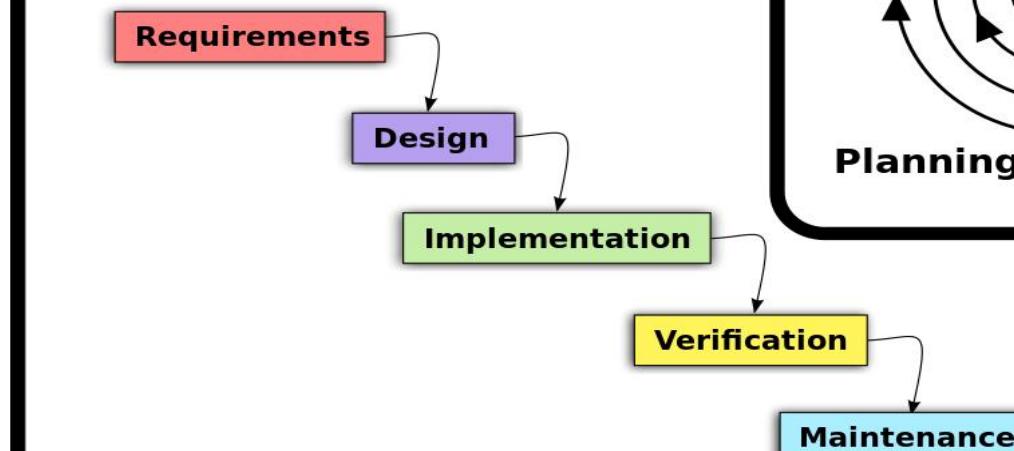
## Prototyping



## Spiral



## Waterfall



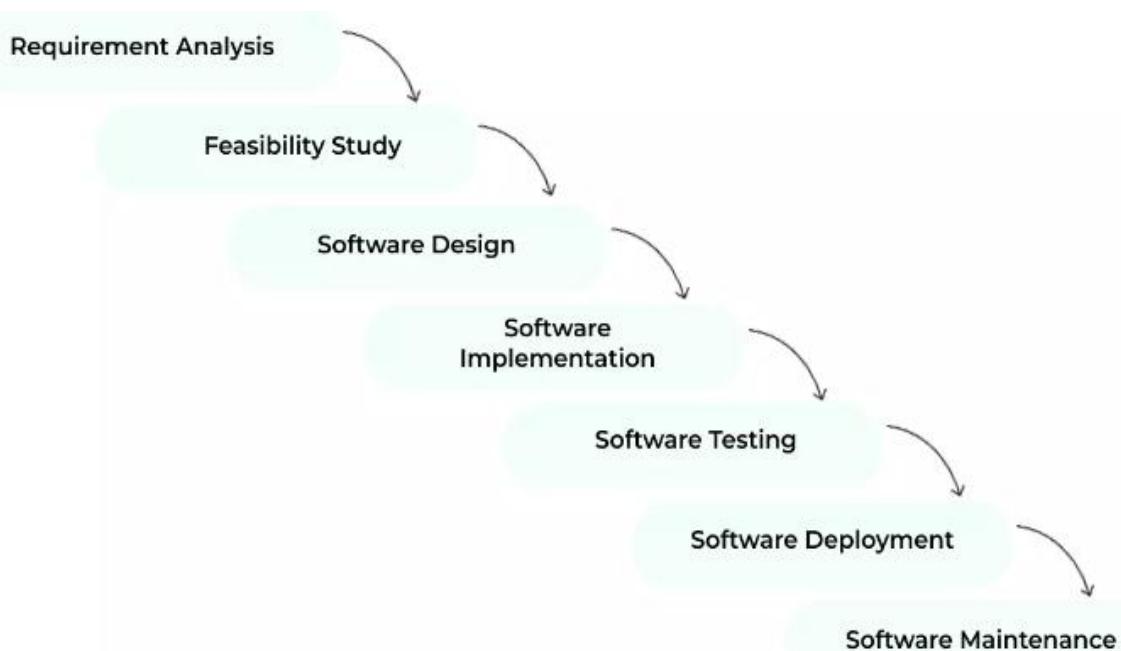
## NEED FOR A SOFTWARE LIFE CYCLE MODEL

- ❖ Without using of a particular life cycle model the development of a software product would not be in a systematic and disciplined manner.
- ❖ When a software product is being developed by a team there must be a clear understanding among team members about when and what to do. Otherwise it would lead to chaos and project failure.
- ❖ Suppose a software development problem is divided into several parts and the parts are assigned to the team members. From then on, suppose the team members are allowed the freedom to develop the parts assigned to them in whatever way they like. It is possible that one member might start writing the code for his part, another might decide to prepare the test documents first, and some other engineer might begin with the design phase of the parts assigned to him.
- ❖ So without software life cycle model the entry and exit criteria for a phase cannot be recognized. Without software life cycle models it becomes difficult for software project managers to monitor the progress of the project.

**Commonly used life cycle models are as follows:**

- *Classical Waterfall Model*
- *Iterative Waterfall Model*
- *Prototyping Model*
- *Evolutionary Model*
- *Spiral Model*

# Classical waterfall model

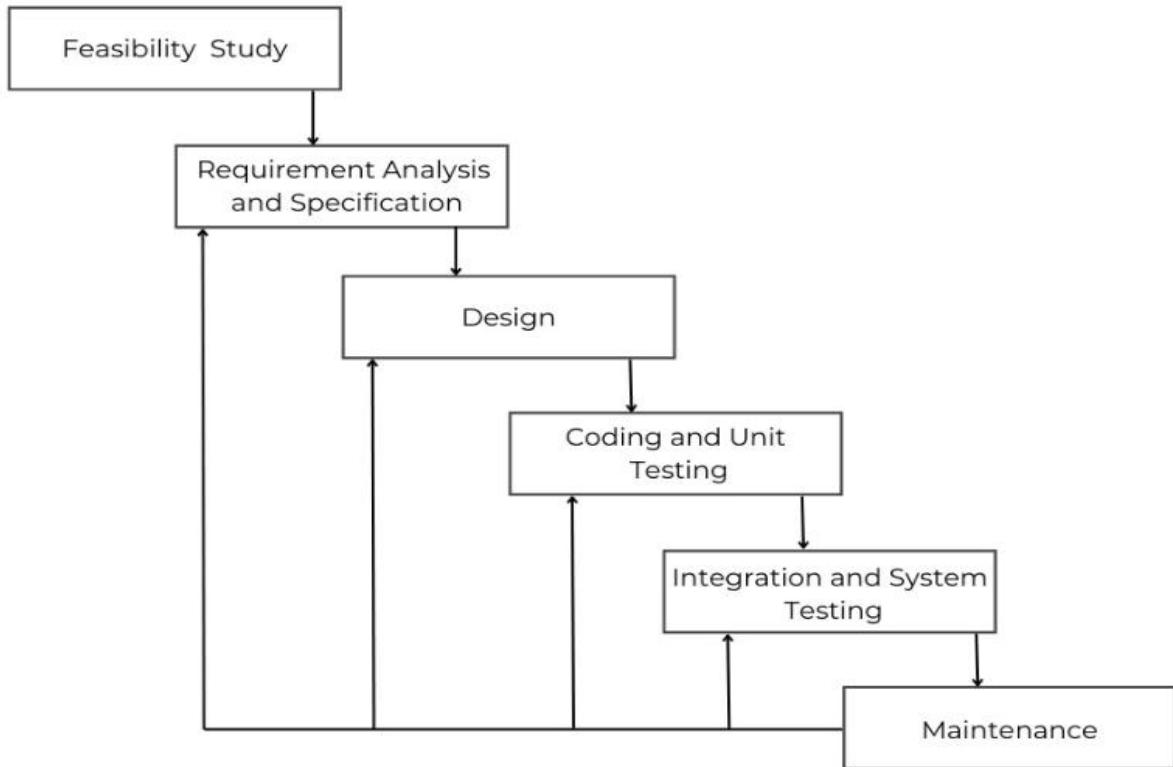


- The classical waterfall model is intuitively the most obvious way to develop software. Though the classical waterfall model is elegant and intuitively obvious, it is not a practical model in the sense that it cannot be used in actual software development projects.
- Thus, this model can be considered to be a theoretical way of developing software. But all other life cycle models are essentially derived from the classical waterfall model.

## Shortcoming.....

- The classical waterfall model is an idealistic one since it assumes that no development error is ever committed by the engineers during any of the life cycle phases. However, in practical development environments, the engineers do commit a large number of errors in almost every phase of the life cycle.
- For example, a design defect might go unnoticed till we reach the coding or testing phase. Once a defect is detected, the engineers need to go back to the phase where the defect had occurred and redo some of the work done during that phase and the subsequent phases to correct the defect and its effect on the later phases.

# Iterative Waterfall Model

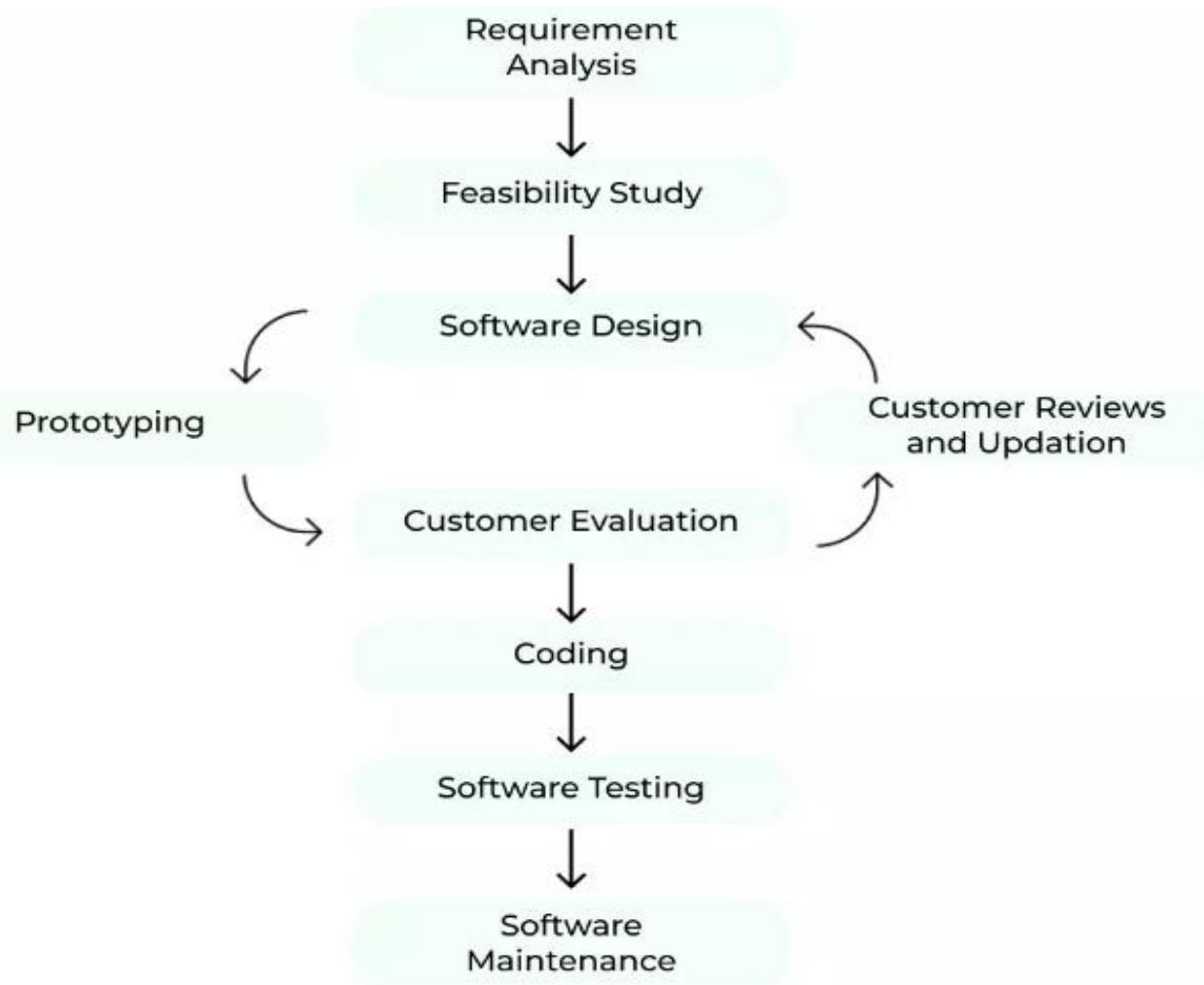


- Here, we provide feedback paths for error correction as & when detected later in a phase. Though errors are inevitable, but it is desirable to detect them in the same phase in which they occur. If so, this can reduce the effort to correct the bug.
- The advantage of this model is that there is a working model of the system at a very early stage of development which makes it easier to find functional or design flaws.

## Shortcoming::::::

- The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

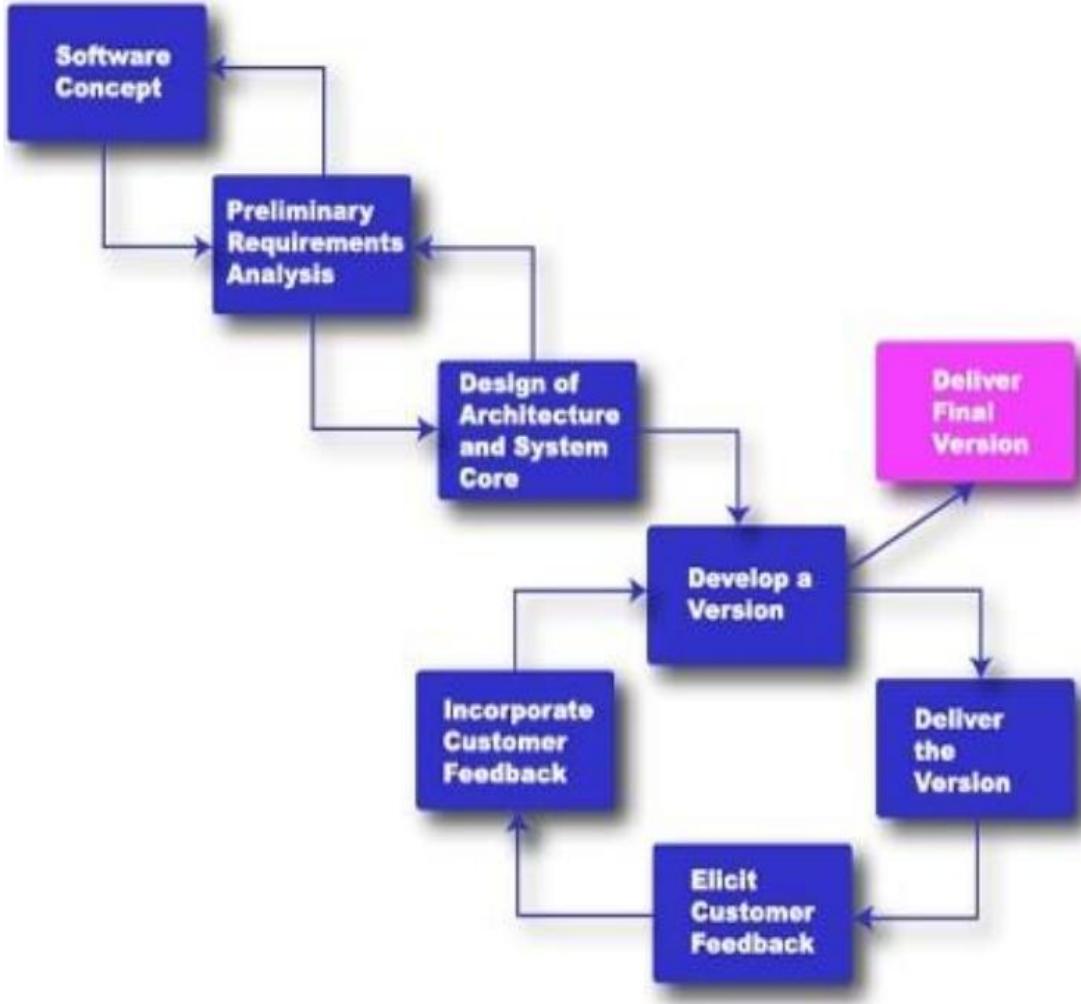
# Prototype Model



- A prototype is a toy implementation of the system. A prototype usually exhibits limited functional capabilities, low reliability, and inefficient performance compared to the actual software.
- An important purpose is to illustrate the input data formats, messages, reports, and the interactive dialogues to the customer. Another reason for developing a prototype is that it is impossible to get the perfect product in the first attempt.
- A prototype of the actual product is preferred in situations such as User requirements are not complete and technical issues are not clear

# Evolutionary Model

- It is also called incremental model. At first, a simple working model is built. Subsequently it undergoes functional improvements & we keep on adding new functions till the desired system is built.



## Applications:

Large projects where you can easily find modules for incremental implementation. Often used when the customer wants to start using the core features rather than waiting for the full software. Also used in object oriented software development because the system can be easily portioned into units in terms of objects.

## Advantages:

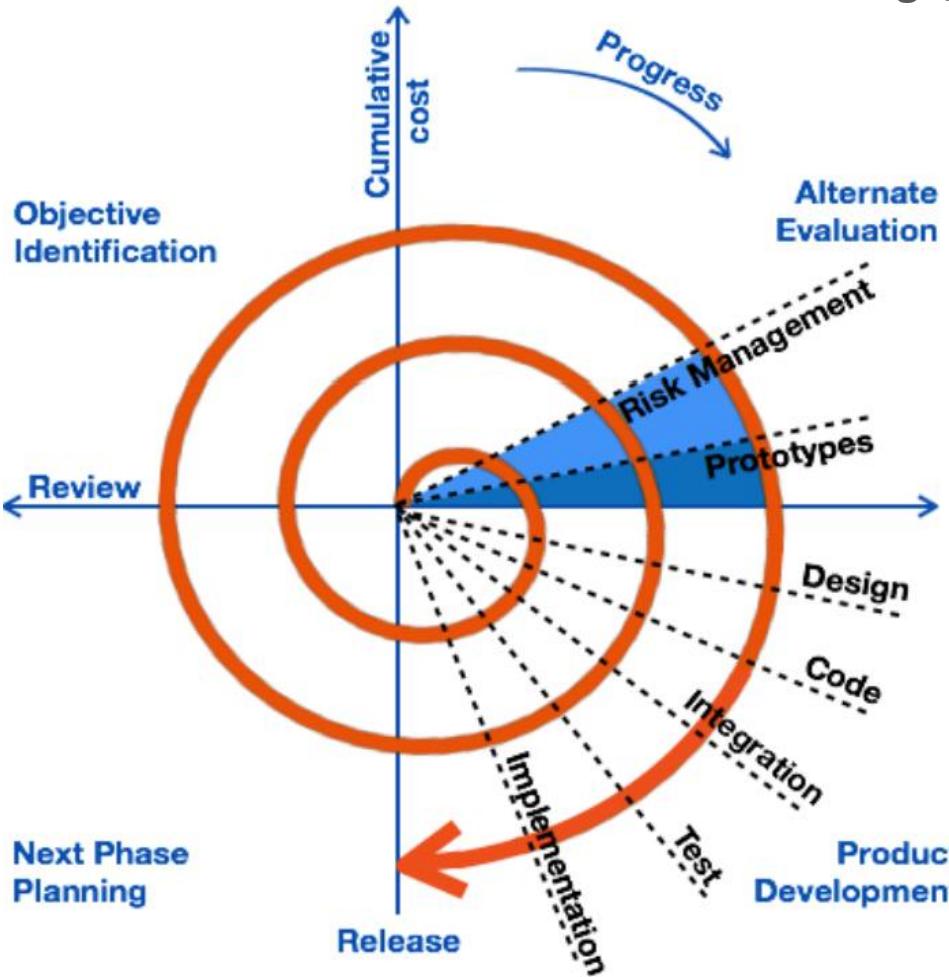
User gets a chance to experiment partially developed system  
Reduce the error because the core modules get tested thoroughly.

## Disadvantages:

It is difficult to divide the problem into several versions that would be acceptable to the customer which can be incrementally implemented & delivered.

# Spiral Model

- Each loop of the spiral represents a phase of the software process. For example, the innermost loop might be concerned with feasibility study, the next loop with requirements specification, the next one with design, and so on. Each phase in this model is split into four quadrants.



*The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks.*

## First quadrant (Objective Setting)

- During the first quadrant, it is needed to identify the objectives of the phase & Examine the risks associated with these objectives.

## Second Quadrant (Risk Assessment and Reduction)

- A detailed analysis is carried out for each identified project risk. Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

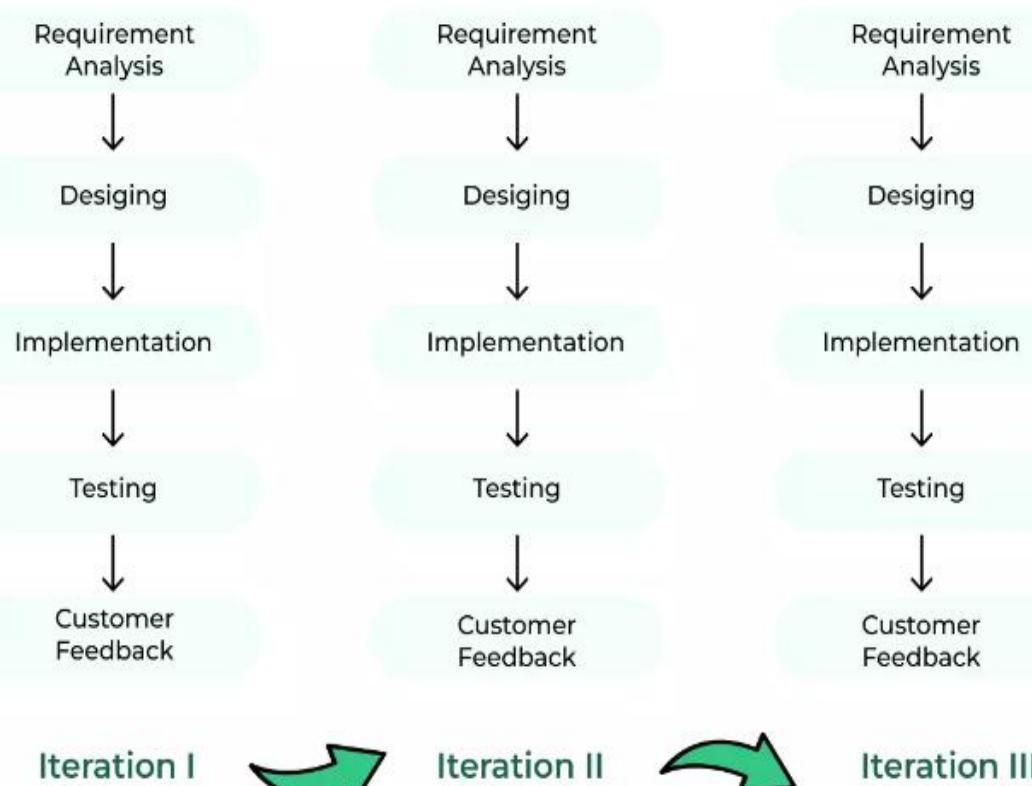
## Third Quadrant (Development and Validation)

- Develop and validate the next level of the product after resolving the identified risks.

## Fourth Quadrant (Review and Planning)

- Review the results achieved so far with the customer and plan the next iteration around the spiral.
- Progressively more complete version of the software gets built with each iteration around the spiral

# Agile Model



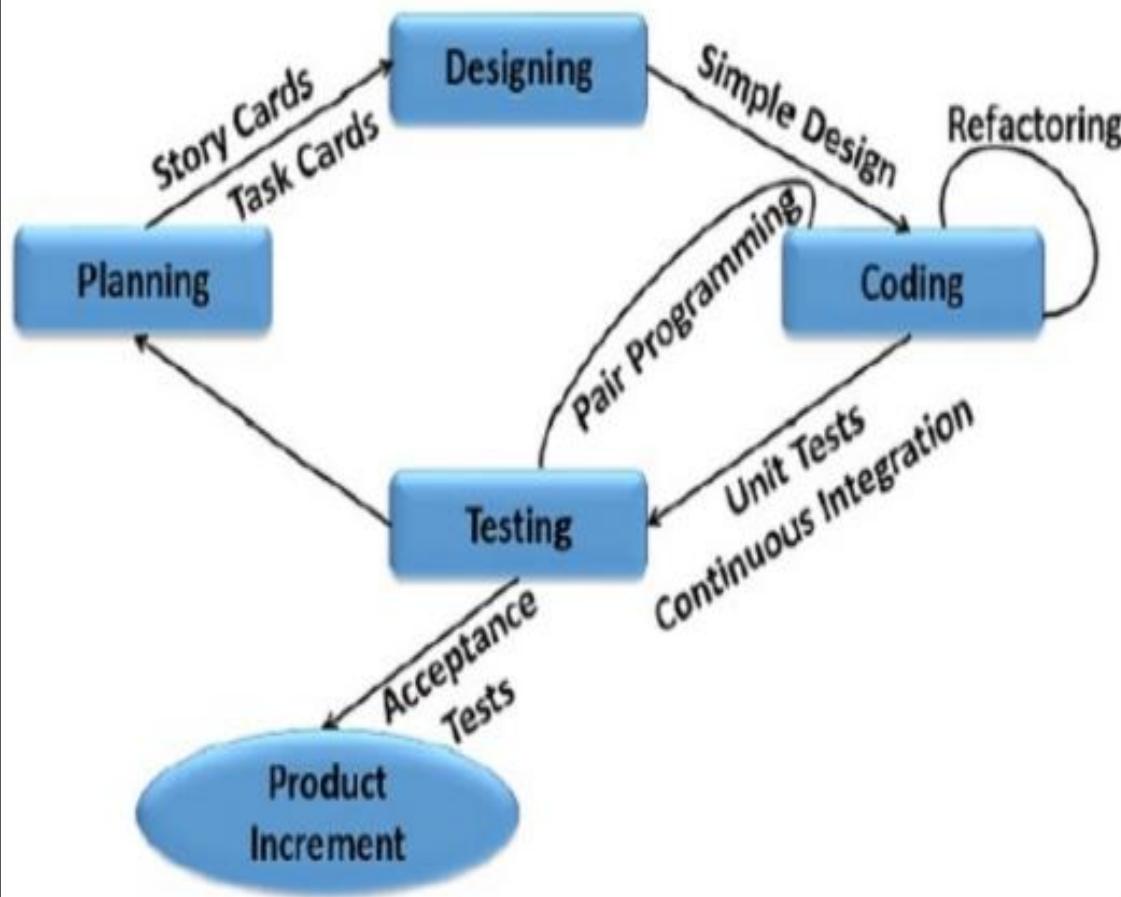
*Agile model emphasizes collaboration, flexibility, and rapid delivery of small increments of functionality. It is well-suited for projects that require rapid development or where requirements may change frequently.*

*One of the key characteristics of Agile is the emphasis on collaboration and frequent communication between team members, as well as with the customer or end user. This helps to ensure that the final product meets the needs of the user and is delivered in a timely manner.*

## Shortcomings:::

- ❖ Lack of upfront planning
- ❖ Limited documentation
- ❖ Communication challenges
- ❖ Limited ability to handle large projects

## Extreme Programming Agile development



## Extreme Programming involves –

- Writing unit tests before programming and keeping all of the tests running at all times. The unit tests are automated and eliminates defects early, thus reducing the costs.
- Starting with a simple design just enough to code the features at hand and redesigning when required.
- Programming in pairs (called pair programming), with two programmers at one screen, taking turns to use the keyboard. While one of them is at the keyboard, the other constantly reviews and provides inputs.
- Integrating and testing the whole system several times a day.
- Putting a minimal working system into the production quickly and upgrading it whenever required.
- Keeping the customer involved all the time and obtaining constant feedback.

## Why is it called “Extreme?”

- Extreme Programming takes the effective principles and practices to extreme levels.
- Code reviews are effective as the code is reviewed all the time.
- Testing is effective as there is continuous regression and testing.
- Design is effective as everybody needs to do refactoring daily.
- Integration testing is important as integrate and test several times a day.

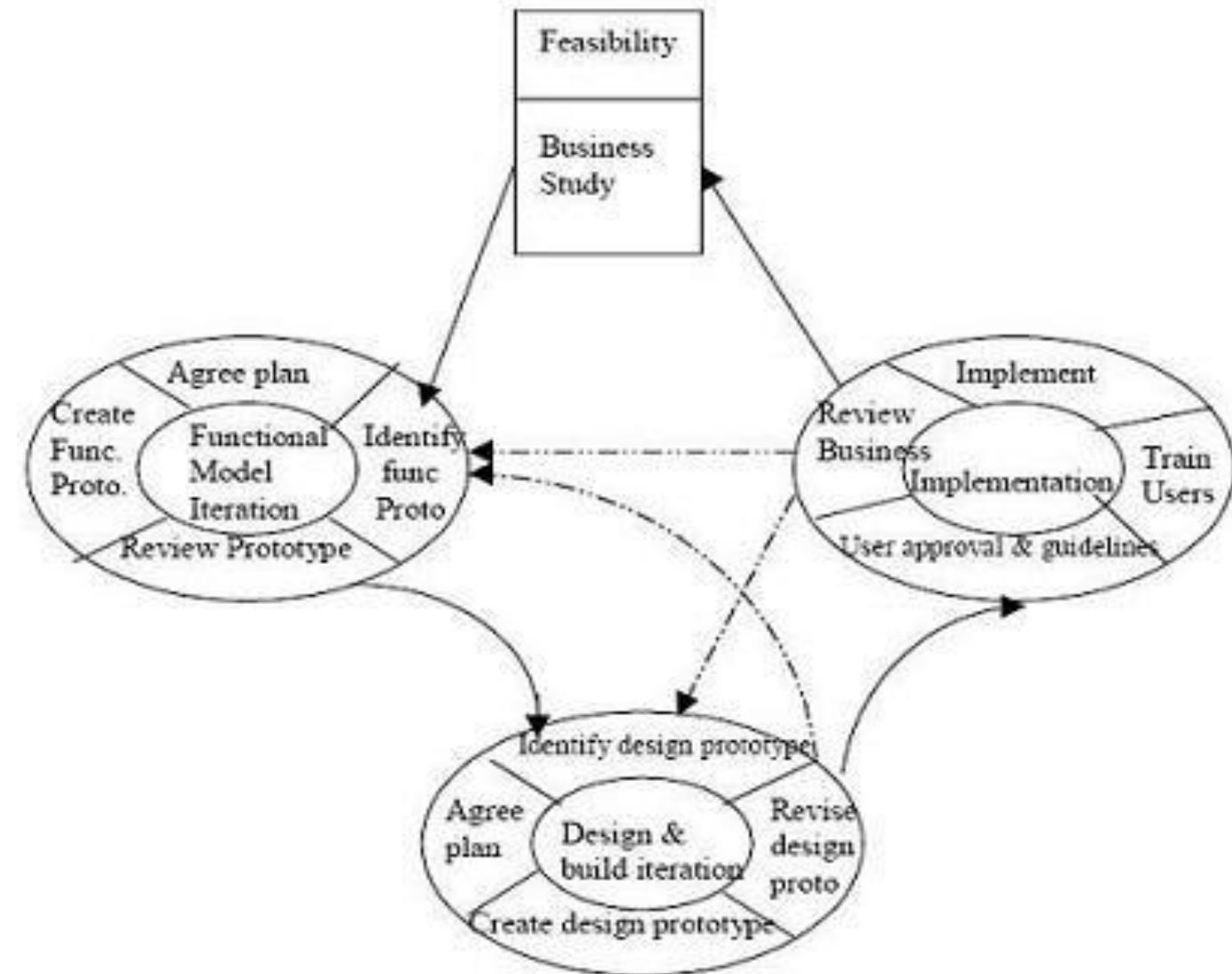
# Dynamic System Development Method (DSDM)

DSDM is an agile framework that follows an iterative approach to system development, which, as the name suggests, develops the system dynamically.

It uses incremental prototyping. This method is particularly useful for the systems to be developed in short time span and where the requirements cannot be frozen at the start of the application building.

Whatever requirements are known at a time, design for them is prepared and incorporated into system. In DSDM, analysis, design and development phase can overlap. Like at one time some people will be working on some new requirements while some will be developing something for the system.

In Dynamic System Development Method (DSDM), requirements evolve with time.



- **Feasibility Study:** In this phase the problem is defined and the technical feasibility of the desired application is verified. Apart from these routine tasks, it is also checked whether the application is suitable for Rapid Application Development (RAD) approach or not.
- **Business Study:** In this phase the overall business study of the desired system is done. The business requirements are specified at a high level and the information requirements out of the system are identified. Once this is done, the basic architectural framework of the desired system is prepared.
- **Functional Model Iteration:** The main focus in this phase is on building the prototype iteratively and getting it reviewed from the users to bring out the requirements of the desired system. The prototype is improved through demonstration to the user, taking the feedback and incorporating the changes. This cycle is repeated until a part of functional model is agreed upon.
- **Design and Build Iteration:** This phase stresses upon ensuring that the prototypes are satisfactorily and properly engineered to suit their operational environment. The software components designed during the functional modeling are further refined till they achieve a satisfactory standard. The product of this phase is a tested system ready for implementation.
- **Implementation:** Implementation is the last and final development stage in this methodology. In this phase the users are trained and the system is actually put into the operational environment.

## **DSDM Model Limitations::::**

- It is a relatively new model. It is not very common. So it is difficult to understand.

## **DSDM Model Advantages::::**

- Active user participation throughout the project and iterative nature of development improves quality of the product.
- DSDM ensures rapid deliveries.
- Both of the above factors result in reduced project costs

**'rules of thumb'  
about approach  
to be used**



**IF uncertainty is high  
THEN use *Spiral approach***

**IF complexity is high but uncertainty is not  
THEN use *Incremental approach***

**IF uncertainty and complexity both low  
THEN use *Waterfall approach***

**IF schedule is tight  
THEN use *Agile approach***

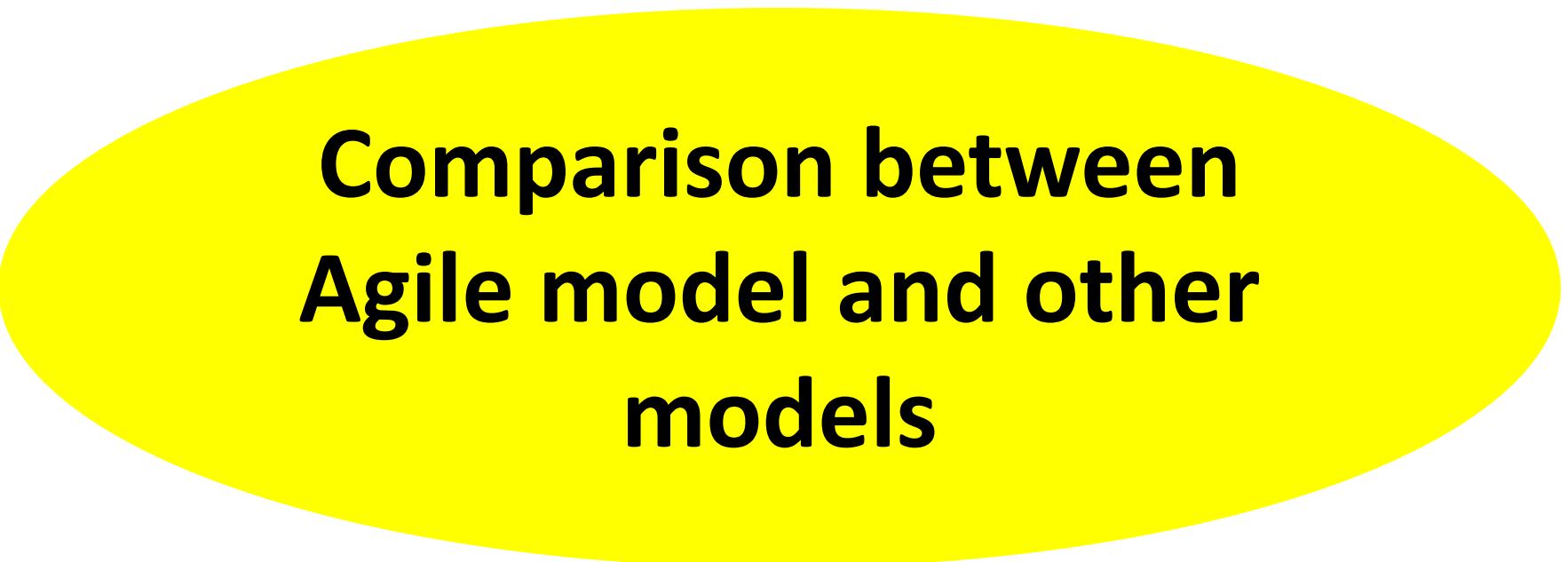
# **Comparison of different life cycle models**

- **Classical Waterfall Model:** The Classical Waterfall model can be considered as the basic model and all other life cycle models are based on this model. It is an ideal model. However, the Classical Waterfall model cannot be used in practical project development, since this model does not support any mechanism to correct the errors that are committed during any of the phases but detected at a later phase. This problem is overcome by the Iterative Waterfall model through the inclusion of feedback paths.
- **Iterative Waterfall Model:** The Iterative Waterfall model is probably the most used software development model. This model is simple to use and understand. But this model is suitable only for well-understood problems and is not suitable for the development of very large projects and projects that suffer from a large number of risks.
- **Prototyping Model:** The Prototyping model is suitable for projects, which either the customer requirements or the technical solutions are not well understood. This risks must be identified before the project starts. This model is especially popular for the development of the user interface part of the project.

- **Evolutionary Model:** The Evolutionary model is suitable for large projects which can be decomposed into a set of modules for incremental development and delivery. This model is widely used in object-oriented development projects. This model is only used if incremental delivery of the system is acceptable to the customer.
- **Spiral Model:** The Spiral model is considered as a meta-model as it includes all other life cycle models. Flexibility and risk handling are the main characteristics of this model. The spiral model is suitable for the development of technically challenging and large software that is prone to various risks that are difficult to anticipate at the start of the project. But this model is more complex than the other models.
- **Agile Model:** The Agile model was designed to incorporate change requests quickly. In this model, requirements are decomposed into small parts that can be incrementally developed. But the main principle of the Agile model is to deliver an increment to the customer after each Time-box. The end date of an iteration is fixed, it can't be extended. This agility is achieved by removing unnecessary activities that waste time and effort.

# END

# SPM 5.0



**Comparison between  
Agile model and other  
models**

<b>Agile model</b>	<b>Waterfall model</b>
The agile model is an incremental delivery process where each incremental delivered part is developed through an iteration after each timebox.	The waterfall model is highly structured and systematically steps through requirements gathering, analysis, SRS document preparation, design, coding, and testing in a planned manner. These phases of the Waterfall model follow a sequential order.
While using an agile model, progress is measured in terms of the developed and delivered functionalities.	In the Waterfall model, progress is generally measured in terms of the number of completed and reviewed artifacts such as requirement specifications, design documents, test plans, code reviews, etc. for which review is complete.
With the agile model, even if a project is canceled midway, it still leaves the customer with some worthwhile code, that might possibly have already been put into live operation.	If a project is developed using the waterfall model is canceled midway during development, then there is nothing to show from the abandoned project beyond several documents.
The agile model allows to change the requirements after the development process starts, so it is more flexible.	The waterfall model is rigid, it does not allow to change requirements after the development process starts.
Customer interaction is very high. After each iteration, an incremental version is deployed to the customer.	Customer interaction is very less. The product is delivered to the customer after the overall development is completed.
The lack of proper formal documentation leaves ample scope for confusion and important decisions taken during various phases can be misinterpreted in fewer later phases.	In the Waterfall model, proper documentation is very important, which gives a clear idea of what should be done to complete the project and it also serves as an agreement between the customer and the development team.
The agile team consists less members (5 to 9 people), but they coordinate and interact with others very frequently.	In the Waterfall model, teams may consist of more members but the interaction between them is limited.

Agile model	RAD model
The Agile model does not recommend developing prototypes but emphasizes the systematic development of each incremental feature at the end of each iteration.	The central theme of RAD is based on designing quick and dirty prototypes, which are then refined into production-quality code.
Agile projects logically break down the solution into features that are incrementally developed and delivered.	The developers using the RAD model focus on developing all the features of an application by first doing it badly and then successively improving the code over time.
The Agile team only demonstrates completed work to the customer after each iteration.	Whereas RAD teams demonstrate to customers screen mocks up and prototypes, that may be based on simplifications such as table lookup rather than actual computations.
The agile model is not suitable for small projects as it is difficult to divide the project into small parts that can be incrementally developed.	When the company has not developed an almost similar type of project, then it is hard to use the RAD model as it is unable to reuse the existing code.

Agile model	Incremental development model
<p>The agile model is an incremental delivery process where each incremental delivered part is developed through an iteration after each time box. The main principle of the Agile model is to achieve agility by removing unnecessary activities that waste time and effort.</p>	<p>The requirements of the software are divided into several modules that can be incrementally developed and delivered. The core features are developed first and the whole software is developed by adding new features in successive versions.</p>
<p>In the Agile model, the end date for an iteration is fixed, it cannot be changed. The development team may have to decide to reduce the delivered functionality to complete that iteration on time.</p>	<p>In the Incremental development model, there is no fixed time to complete the next iteration.</p>

Agile model	Spiral model
The main principle of the Agile model is to achieve agility by removing unnecessary activities that waste time and effort.	The main principle of the Spiral model is risk handling.
The Agile model focuses on the delivery of an increment to the customer after each Time-box, so customer interaction is more frequent.	The spiral model mainly deals with various kinds of unanticipated risks but customer interaction is less.
The agile model is suitable for large projects that are easy to divide into small parts that can be easily developed incrementally over each iteration.	The Spiral model is suitable for those projects that are prone to various kinds of risks that are difficult to anticipate at the beginning of the project.
The agile model does not rely on documentation.	Proper documentation is required for the Spiral model.

# END

# SPM 6.0

# **Software effort estimation**

**S o f t w a r e   E f f o r t**  
**e s t i m a t i o n** is a process in which project managers evaluate how much time and money they need for completing a project. This technique is common in software development, where technology professionals define the resources and schedule for developing a new application or releasing an update. These forecasts help create accurate estimates that often require approval before work on a project begins.

Estimates are done during the project planning stage.

## Why is effort estimation important?

**Understanding a project:** When estimating the effort for development, it can help team members to understand each task they complete, along with any dependencies.

**Improving collaboration:** Collaboration is a key concept in Agile framework, and teams meet regularly to discuss priorities and status to accomplish this. During estimation meetings, each team member can express their opinions on efforts, priorities and resource needs.

**Creating an accurate budget:** When you evaluate the effort to complete a project, you can determine what team members, tools and time you need. This can help you create an accurate project budget by estimating wages and the cost of each resource.

**Making an accurate schedule:** Organizations often rely on accurate delivery schedules and provide this information to employees and customers. Using the various estimation techniques can help ensure you build an accurate schedule based on expert input and historical data.

**Managing resources and assignments:** By evaluating the resources you need for a project and the tasks to complete, you can estimate what resources you need.

**Improving risk management:** Estimating effort at the start of a project can help you identify and manage risks, like breaching a budget or schedule.

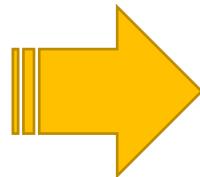
# Problems with estimating

- ❖ Subjective nature of much of estimating  
Under estimating the difficulties of small task and over estimating large projects.
- ❖ Political pressures  
Managers may wish to reduce estimated costs in order to win support for acceptance of a project proposal (over estimate to create a comfort zone)
- ❖ Changing technologies  
these bring uncertainties, especially in the early days when there is a 'learning curve'. Difficult to use the experience of previous projects.
- ❖ Projects differ -lack of homogeneity of project experience  
Experience on one project may not be applicable to another

**Calculate the productivity (i.e. SLOC/work month) of each of the projects in Table next slide and also for the organization as a whole. If the project leaders for projects a and d had correctly estimated the source number of lines of code (SLOC) and then used the average productivity of the organization to calculate the effort needed to complete the projects, how far out would their estimate have been from the actual one.**



## Under-estimation or Over-estimation?



*For any project task, it is unlikely that your estimate of how long it will take will be perfect. So, is it worse to overestimate the time it will take to complete a task or underestimate it?*

**Cost of overestimation:**  
Overestimation creates the problem that the task takes longer than it would have done with a more accurate estimate in place. There are two ideas behind this linked to how people behave. Firstly, Student's Syndrome states that people often won't start working until very close to a deadline. Secondly, Parkinson's Law states that work expands to fill the time available. Therefore, if you have a task with overestimated length, the impact is the task might take longer than it 'should' do.

**Costs of underestimation:**  
If a task is assumed to take long time than it actually needs, one of two things will happen. Either the task gets done at lower quality, or the task doesn't get done on time and any tasks dependent on it are pushed out.

## Which is worse???

Whilst obviously accurate estimates are the best outcome, over-estimation is less bad than underestimation. Underestimation can impact dependencies and the overall quality of the project. Overestimation may be wasteful for the resources on a particular task, but it is less likely to impact other tasks or overall quality.

Of course, a third option following the critical chain methodologies is to consider adding buffers to the schedule to allow for some underestimation at the individual task level.

## Basis for successful estimating

- ❑ Information about past projects

Need to collect performance details about past project: how big were they?

How much effort/time did they need?

- ❑ Need to be able to measure the amount of work involved

Traditional size measurement for software is 'lines of code' (LOC) – but this can have problems

FP measure corrects to a great extent.

## Parameters to be Estimated

- Size is a fundamental measure of work
- Based on the estimated size, two parameters are estimated:
  - *Effort*
  - *Duration*
- Effort is measured in person-months:  
One person-month is the effort an individual can typically put in a month.

## Measure of Work

- Project size is a measure of the problem complexity in terms of effort and time required to develop the product.
- Two metrics are used to measure project size:
  - *Source Lines of Code (SLOC)*
  - *Function point (FP)*
- FP is now-a-days favored over SLOC:

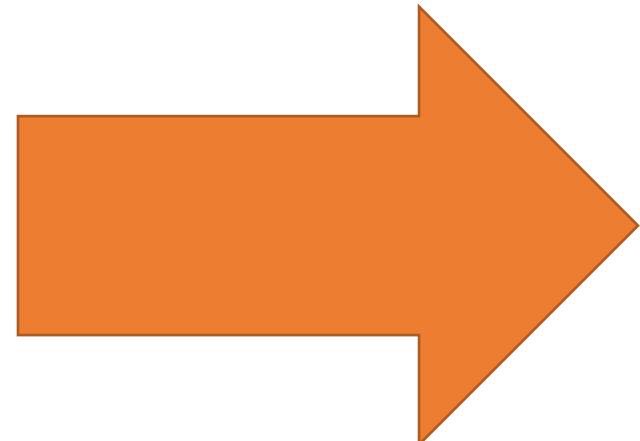
Because of the many **Shortcomings of SLOC**

- ❖ No precise definition (e.g. comment line, data declaration line to be included or not?)
- ❖ Difficult to estimate at start of a project
- ❖ Only a code measure
- ❖ Programmer-dependent
- ❖ Does not consider code complexity

# Taxonomy of estimating methods::::::

Understanding the size and effort of a software project early on is a difficult problem. different methods exist, but no method is perfect. Here, we present an overview of the four methods most mentioned in literature:

- ❖ Expert opinion-based
- ❖ Top-down estimation
- ❖ Bottom-up estimation
- ❖ Parametric or algorithmic based



## **Expert estimation:**

- Expert estimation means that an expert estimates how much effort a project requires. The advantages of asking somebody else than the project manager to estimate a project, is that some experts have deep knowledge about the problem at hand.
- Many experts will use intuition and/or previous experience to estimate the project, which is usually more time-efficient than any of the other methods.
- The problem with expert estimates is that their reliability is typically unknown (unless the organisation tracks the performance of their estimators) and that their estimates are not objective. This last aspect is important when discussions arise.

## **Top down estimation:**

- Top-down estimation methods use experience from the past to make estimates for the future. They require examples of completed IT projects to base the new estimates upon. The easiest way is finding three previous projects, and take the average of these projects. A better, more advanced way is to use more information for selecting the most similar projects.
- An estimate for a new project is derived by comparing the new project to all old projects and selecting the (effort of the) project that is most similar. The advantage is that their basis is more objective and repeatable than expert estimation.

## Bottom-up estimation:

- Bottom-up estimation methods take a project definition & breaks up the project activities or deliverables into smaller sub activities or partial deliverables until each sub activity of partial deliverable requires less time & effort. After this project decomposition, in which the project is broken into smaller bits, each individual part is estimated. These estimates are then combined into an overall estimate.
- The advantage is that their estimates are more understandable than an expert's holistic estimate. Also, since estimation errors tend to cancel each other out, it is more reliable than an expert's single estimate of an entire project.

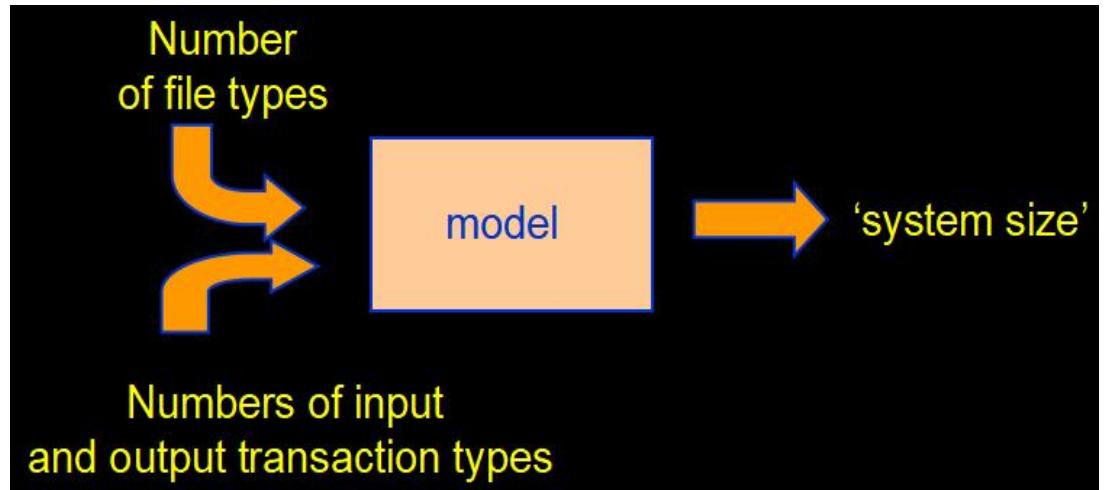
## Parametric estimation:

- Parametric estimation methods use a model or algorithm that takes as input some aspects of the project (such as the required functionality and the quality that is expected). The model (a formula) or algorithm (computational steps) then produce an estimate based on those inputs alone.
- The advantage is that they tend to be more objective as their counterparts. When properly used and with sufficient calibration these methods can produce highly reliable estimates. Unfortunately their use is more complex and often more time-consuming
- COCOMO (lines of code) and function points examples.

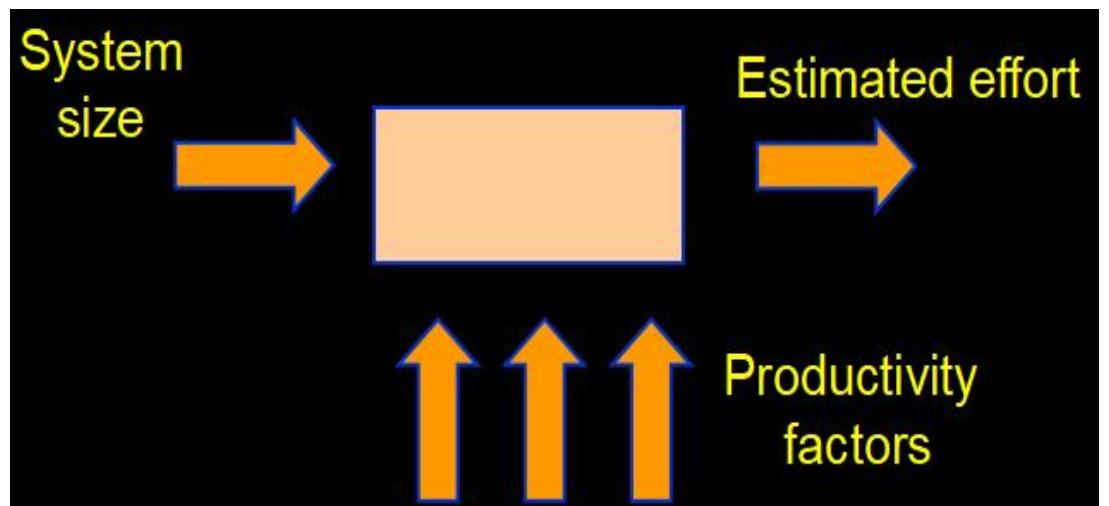
# Top-Down vs Bottom-Up Estimates

Top-Down	Bottom-Up
Feasible for overview estimates	Feasible for detailed estimates
Tasks are not defined clearly	Tasks are defined clearly
Volatile environment, not viable to develop well-defined schedules & budget	Stable environment, viable to develop well-defined schedules & budget
Less reliable duration & costs estimates	More reliable duration & costs estimates

- Some models focus on task or system size  
e.g. Function Points
- FPs originally used to estimate Lines of Code, rather than effort



- Other models like COCOMO focus on productivity.
- Lines of code (or FPs etc) an input



# END

# SPM 7.0

# Parametric models

## Parametric models - the need for historical data

- ✓ Simplistic model for an estimate

estimated effort = (system size) / (productivity)

e.g.

system size = lines of code

productivity = lines of code per day

- ✓ Productivity = (system size) / effort

based on past projects

Software size = 2 KLOC

If A (expert) can take 40 days per KLOC    80 days

If B (novice) takes 55 days per KLOC    110 days

- ✓ KLOC is a size driver, experience influences productivity

## **We are now looking more closely at four parametric models:**

- ❖ **Albrecht/IFPUG function points**
- ❖ **Symons/Mark II function points**
- ❖ **COSMIC function points**
- ❖ **cocomo81 and COCOMO II**

COSMIC- common software measurement consortium

COCOMO- cost constructive model

IFPUG- international function point user group

# Albrecht/IFPUG function points

- Albrecht worked at IBM and needed a way of measuring the relative productivity of different programming languages.
- Needed some way of measuring the size of an application without counting lines of code.
- Identified five types of component or functionality in an information system
- Counted occurrences of each type of functionality in order to get an indication of the size of an information system

### Five function types

1. **Logical internal file (LIF) types** – equates roughly to a data store in systems analysis terms. Created and accessed by the target system (it refers to a group data items that is usually accessed together i.e. one or more record types) PURCHASE-ORDER and PURCHASE-ORDER-ITEM
2. **External interface file types (EIF)** – where data is retrieved from a data store which is actually maintained by a different application.
3. **External input (EI) types** – input transactions which update internal computer files
4. **External output (EO) types** – transactions which extract and display data from internal computer files. Generally involves creating reports.
5. **External inquiry (EQ) types** – user initiated transactions which provide information but do not update computer files. Normally the user inputs some data that guides the system to the information the user needs.

Table-1

External user types	Low complexity	Medium complexity	High complexity
EI External input type	3	4	6
EO External output type	4	5	7
EQ External inquiry type	3	4	6
LIF Logical internal file type	7	10	15
EIF External interface file type	5	7	10



With FPs originally defined by Albrecht, the external user type is of high, low or average complexity is intuitive. Ex: in the case of **logical internal files** and **external interface files**, the boundaries shown in table below are used to decide the complexity level.

Table-2

Number of record types	Number of data types		
	< 20	20 – 50	> 50
1	Low	Low	Average
2 to 5	Low	Average	High
> 5	Average	High	High

## Example

A **logical internal file** might contain data about purchase orders. These purchase orders might be organized into two separate record types: the main PURCHASE-ORDER details, namely purchase order number, supplier reference and purchase order date.

The details of PURCHASE-ORDER-ITEM specified in the order, namely the product code, the unit price and number ordered.

- The number of record types for this will be 2
- The number of data types will be 6
- According to the previous table-2 file type will be rated as 'Low'
- According to the previous table-1 the FP count is 7

# Examples

Payroll application has:

1. Transaction to input, amend and delete employee details – an EI that is rated of medium complexity
2. A transaction that calculates pay details from timesheet data that is input – an EI of high complexity
3. A transaction of medium complexity that prints out pay-to-date details for each employee – EO
4. A file of payroll details for each employee – assessed as of medium complexity LIF
5. A personnel file maintained by another system is accessed for name and address details – a simple EIF

What would be the FP counts for these?



## FP counts

Refer Table-1

Medium EI	4 FPs
High complexity EI	6 FPs
Medium complexity EO	5 FPs
Medium complexity LIF	10 FPs
Simple EIF	5 FPs
<b>Total</b>	<b>30 FPs</b>

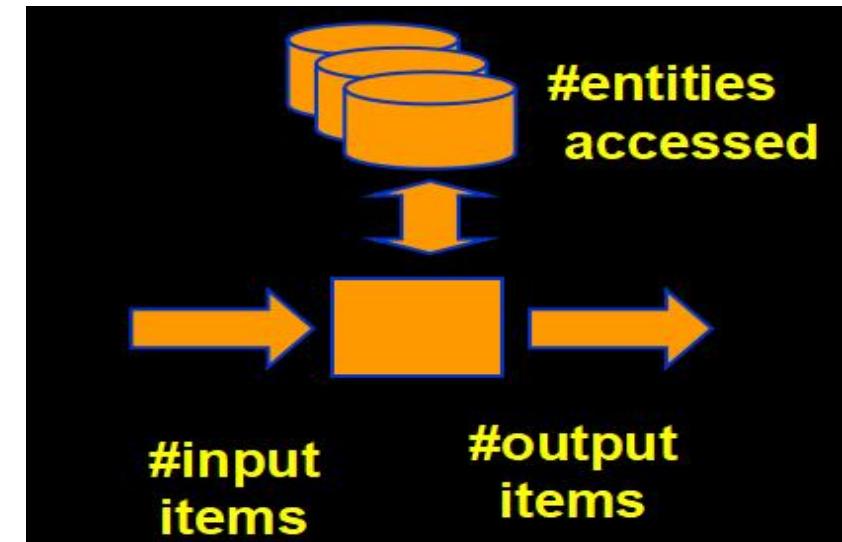
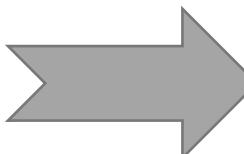
If previous projects delivered 5 FPs a day, implementing the above should take  $30/5 = 6$  days

# Function points Mark II

- Developed by Charles R. Symons  
Software sizing and estimating - Mk II FPA', Wiley & Sons, 1991.
- Builds on work by Albrecht
- Work originally for CCTA:  
should be compatible with SSADM; mainly used in UK
- Has developed in parallel to IFPUG FPs

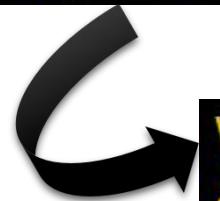
For each transaction,  
count

- ↳ data items input ( $N_i$ )
- ↳ data items output ( $N_o$ )
- ↳ entity types accessed  
( $N_e$ )



$$\text{FP count} = N_i * 0.58 + N_e * 1.66 + N_o * 0.26$$

- UFP – Unadjusted Function Point – Albrecht  
(information processing size is measured)
- TCA – Technical Complexity Adjustment  
(the assumption is, an information system comprises transactions which have the basic structures, as shown in previous slide)
- For each transaction the UFPs are calculated:  
 $W_i \times (\text{number of input data element types}) +$   
 $W_e \times (\text{number of entity types referenced}) +$   
 $W_o \times (\text{number of output data element types})$



$W_i, W_e, W_o$  are weightings derived by asking developers the proportions of effort spent in previous projects developing the code dealing with input, accessing stored data and outputs.

$W_i = 0.58, W_e = 1.66, W_o = 0.26$  (industry average)

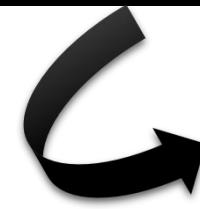
## **Exercise:**

A cash receipt transaction in an accounts subsystem accesses two entity types INVOICE and CASH-RECEIPT.

The data inputs are:

- Invoice number
- Date received
- Cash received

If an INVOICE record is not found for the invoice number then an error message is issued. If the invoice number is found then a CASH-RECEIPT record is created. The error message is the only output of the transaction. Calculate the unadjusted function points, using industry average weightings, for this transaction.



$$(0.58 \times 3) + (1.66 \times 2) + (0.26 \times 1) = 5.32$$

## Exercise:

In an annual maintenance contract subsystem is having a transaction which sets up details of new annual maintenance contract customers.

- |                            |                  |
|----------------------------|------------------|
| 1. Customer account number | 2. Customer name |
| 3. Address                 | 4. Postcode      |
| 5. Customer type           | 6. Renewal date  |

All this information will be set up in a CUSTOMER record on the system's database. If a CUSTOMER account already exists for the account number that has been input, an error message will be displayed to the operator.

Calculate the number of unadjusted Mark II function points for the transaction described above using the industry average.

Answer:

The function types are:

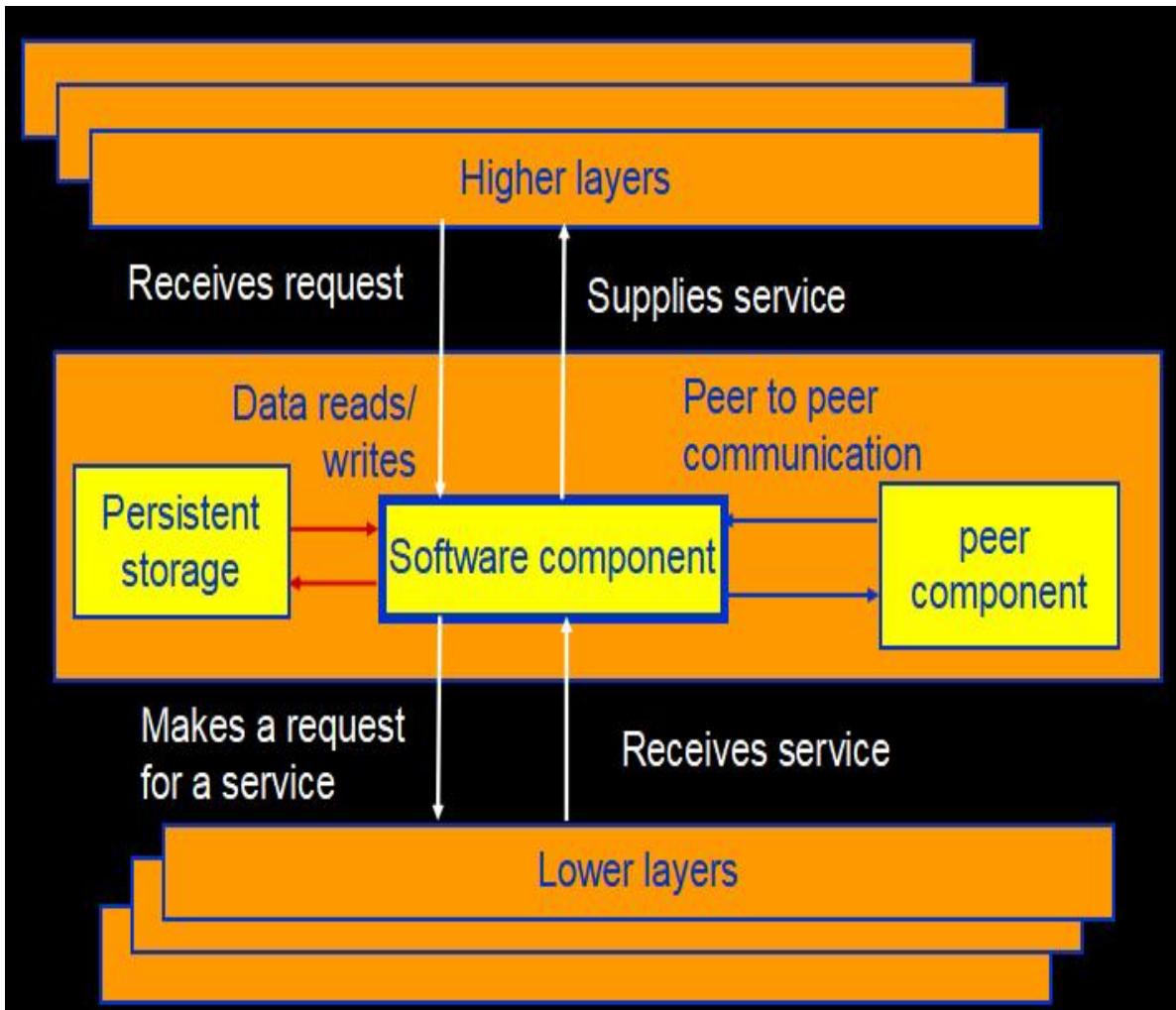
Input data types	6
Entities accessed	1
Output data types	1

$$\begin{aligned} \text{UFP} &= \text{Unadjusted function points} \\ &= (0.58 \times 6) + (1.66 \times 1) + (0.26 \times 1) \\ &= 5.4 \end{aligned}$$

# COSMIC function points

- ✓ Mark II function points, IFPUG function points were designed for information systems environments
- ✓ They are not helpful for sizing real-time or embedded systems
- ✓ COSMIC-FFPs (common software measurement consortium-full function point) attempt to extend concept to embedded systems or real-time systems
- ✓ FFP method origins the work of two interlinked groups in Quebec, Canada
- ✓ Embedded software seen as being in a particular ‘layer’ in the system
- ✓ Communicates with other layers and also other components at same level
- ✓ The argument is existing function point method is effective in assessing the work content of an information system.
- ✓ Size of the internal procedures mirrors the external features.
- ✓ in real-time or embedded system, the features are hidden because the software’s user will probably not be human beings but a hardware device.
- ✓ COSMIC deals with by decomposing the system architecture into a hierarchy of software layers.
- ✓ The software component to be sized can receive requests the service from layers above and can request services from those below.
- ✓ There may be separate software components engage in peer-to-peer communication.
- ✓ Inputs and outputs are aggregated into data groups, where each data group brings together data items related to the same objects.

## Layered software



The following are counted:

(Data groups can be moved in four ways)

- Entries (E): movement of data into software component from a higher layer or a peer component
- Exits (X): movements of data out to a user outside its boundary
- Reads (R): data movement from persistent storage
- Writes (W): data movement to persistent storage

Each counts as 1 'COSMIC functional size unit' (Cfsu). The overall FFP count is derived by simply adding up the counts for each of the four types of data movement.

### **Exercise:**

A small computer system controls the entry of vehicles to a car park. Each time a vehicle pulls up before an entry barrier, a sensor notifies the computer system of the vehicle's presence. The system examines a count that it maintains the number of vehicles currently in the car park. This count is kept on the backing storage so that it will still be available if the system is temporarily shut down, for example because of a power cut. If the count does not exceed the maximum allowed then the barrier is lifted and count is incremented. When the vehicle leaves the car park, a sensor detects the exit and reduce the count of vehicles.

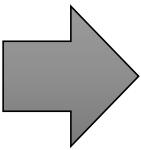
Identify the entries, exits, reads and writes in this application.

Data movement	Type
Incoming vehicles sensed	E
Access vehicle count	R
Signal barrier to be lifted	X
Increment vehicle count	W
Outgoing vehicle sensed	E
Decrement vehicle count	W
New maximum input	E
Set new maximum	W
Adjust current vehicle count	E
Record adjusted vehicle count	W

**Note:** different interpretations of the requirements could lead to different counts. The description in the exercise does not specify to give a message that the car park is full or has spaces.

# COCOMO81 & COCOMO II

## COCOMO81



- Based on industry productivity standards - database is constantly updated
- Allows an organization to benchmark its software development productivity
- **Basic model**  
$$\text{effort} = c \times \text{size}^k$$
- C and k depend on the type of system: organic, semi-detached, embedded
- Size is measured in 'kloc' ie. Thousands of lines of code



System type	c	k
Organic (broadly, information systems, small team, highly familiar in-house environment)	2.4	1.05
Semi-detached (combined characteristics between organic and embedded modes)	3.0	1.12
Embedded (broadly, real-time, products developed has to operate within very tight constraints and changes to system is very costly)	3.6	1.20

$$\text{effort} = c (\text{size})^k$$

effort – pm (person month) – 152 working hours

size – KLOC – kdsi (thousands of delivered source code instruction)

c and k – constants depending on whether the system is organic, semi-detached or embedded.

Organic : Effort = 2.4(KLOC)<sup>1.05</sup> PM

Semidetached : Effort = 3.0(KLOC)<sup>1.12</sup> PM

Embedded : Effort = 3.6(KLOC)<sup>1.20</sup> PM

Estimation of development time

$$T_{\text{dev}} = a \times (\text{Effort})^b$$

Organic : 2.5(Effort)<sup>0.38</sup>

Semidetached : 2.5(Effort)<sup>0.35</sup>

Embedded : 2.5(Effort)<sup>0.32</sup>

### **Exercise:**

Assume that the size of an organic type software product is estimated to be 32,000 lines of source code. Assume that the average salary of a software developer is Rs.50,000 per month. Determine the effort required to develop the software product, the nominal development time, and the staff cost to develop the product.

$$\text{Effort} = 2.4 \times 32^{1.05} = 91 \text{ pm}$$

$$\text{Nominal development time} = 2.5 \times 91^{0.38} = 14 \text{ months}$$

Staff cost required to develop the product

$$91 \times \text{Rs. } 50,000 = \text{Rs. } 45,50,000$$

*Ex-Two software managers separately estimated a given product to be of 10,000 and 15,000 lines of code respectively. Bring out the effort and schedule time implications of their estimation using COCOMO. For the effort estimation, use a coefficient value of 3.2 and exponent value of 1.05. For the schedule time estimation, the similar values are 2.5 and 0.38 respectively. Assume all adjustment multipliers to be equal to unity.*

For 10,000 LOC

$$\text{Effort} = 3.2 \times 10^{1.05} = 35.90 \text{ PM}$$

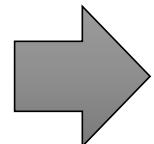
$$\text{Schedule Time} = T_{\text{dev}} = 2.5 \times 35.90^{0.38} = 9.75 \text{ months}$$

For 15,000 LOC

$$\text{Effort} = 3.2 \times 15^{1.05} = 54.96 \text{ PM}$$

$$\text{Schedule Time} = T_{\text{dev}} = 2.5 \times 54.96^{0.38} = 11.46 \text{ months}$$

## COCOMO II



An updated version of COCOMO:

- There are different COCOMO II models for estimating at the 'early design' stage and the 'post architecture' stage when the final system is implemented. We'll look specifically at the first.
- The core model is:

$$pm = A(\text{size})^{sf} \times (em_1) \times (em_2) \times (em_3) \dots$$

where **pm** = person months, **A** is 2.94, **size** is number of thousands of lines of code, **sf** is the scale factor, and **em** is an effort multiplier

$$sf = B + 0.01 \times \sum (\textit{exponent driver ratings})$$

## COCOMO II Scale factor

Boehm et al. have refined a family of cost estimation models. The key one is COCOMO II. It uses multipliers and exponent values. Based on five factors which appear to be particularly sensitive to system size.

Precedentedness (**PREC**). Degree to which there are past examples that can be consulted, else uncertainty

Development flexibility (**FLEX**). Degree of flexibility that exists when implementing the project

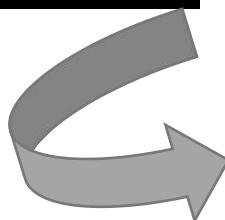
Architecture/risk resolution (**RESL**). Degree of uncertainty about requirements, liable to change

Team cohesion (**TEAM**). Large dispersed

Process maturity (**PMAT**) could be assessed by CMMI, more structured less uncertainty

## COCOMO II Scale factor values

Driver	Very low	Low	Nominal	High	Very high	Extra high
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00



### Example of scale factor

- A software development team is developing an application which is very similar to previous ones it has developed.
- A very precise software engineering document lays down very strict requirements. PREC is very high (score 1.24).
- FLEX is very low (score 5.07).
- The good news is that these tight requirements are unlikely to change (RESL is high with a score 2.83).
- The team is tightly knit (TEAM has high score of 2.19), but processes are informal (so PMAT is low and scores 6.24)

## Scale factor calculation

The formula for sf is

$$sf = B + 0.01 \times \Sigma \text{ scale factor values}$$

$$\begin{aligned} \text{i.e. } sf &= 0.91 + 0.01 \times (1.24 + 5.07 + 2.83 + 2.19 + 6.24) \\ &= 1.0857 \end{aligned}$$

If system contained 10 kloc then estimate would be *effort*

$$= c (\text{size})^k = 2.94 \times 10^{1.0857} = 35.8 \text{ person-months}$$

Using exponentiation ('to the power of') adds disproportionately more to the estimates for larger applications

$$B = 0.91(\text{constant}), \quad c = 2.94 \text{ (average)}$$

### Exercise:

A new project has ‘average’ novelty for the software supplier that is going to execute it and thus given a nominal rating on this account for precedentedness. Development flexibility is high, requirements may change radically and so risk resolution exponent is rated very low. The development team are all located in the same office and this leads to team cohesion being rated as very high, but the software house as a whole tends to be very informal in its standards and procedures and the process maturity driver has therefore been given a rating of ‘low’.

- (i) What would be the scale factor ( $sf$ ) in this case?
- (ii) What would the estimate effort if the size of the application was estimated as in the region of 2000 lines of code?

### Assessing the scale factors

Factor	Rating	Value
PREC	nominal	3.72
FLEX	high	2.03
RESL	very low	7.07
TEAM	very high	1.10
PMAT	low	6.24

- (i) The overall scale factor =  $sf = B + 0.01 \times \sum \text{ (exponent factors)}$
- $$= 0.91 + 0.01 \times (3.72 + 2.03 + 7.07 + 1.10 + 6.24)$$
- $$= 0.91 + 0.01 \times 20.16 = 1.112$$
- (ii) The estimated effort =  $c \times (\text{size})^k = 2.94 \times 2^{1.112} = 6.35 \text{ staff-months}$

## COCOMO II Effort multipliers

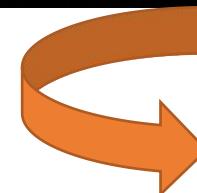
As well as the scale factor effort multipliers are also assessed:

RCPX	Product reliability and complexity
RUSE	Reuse required
PDIF	Platform difficulty
PERS	Personnel capability
PREX	Personnel experience
FCIL	Facilities available
SCED	Schedule pressure

	Extra low	Very low	Low	Nominal	High	Very high	Extra high
RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE			0.95	1.00	1.07	1.15	1.24
PDIF			0.87	1.00	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	1.59	1.33	1.12	1.00	0.87	0.74	0.62
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED		1.43	1.14	1.00	1.00	1.00	

# Example

- Say that a new project is similar in most characteristics to those that an organization has been dealing for some time
- **except**
  - the software to be produced is exceptionally complex and will be used in a safety critical system.
  - The software will interface with a new operating system that is currently in beta status.
  - To deal with this the team allocated to the job are regarded as exceptionally good, but do not have a lot of experience on this type of software.



RCPX	very high	1.91
PDIF	very high	1.81
PERS	extra high	0.50
PREX	nominal	1.00

All other factors are nominal

Say estimate is 35.8 person months

With effort multipliers this becomes  $35.8 \times 1.91 \times 1.81 \times 0.5 = 61.9$  person months

### **Exercise:**

A software supplier has to produce an application that controls a piece of equipment in a factory. A high degree of reliability is needed as a malfunction could injure the operators. The algorithms to control the equipment are also complex. The product reliability and complexity are therefore rated as very high. The company would like to take opportunity to exploit fully the investment that they made in the project by reusing the control system, with suitable modifications, on future contracts. The reusability requirement is therefore rated as very high. Developers are familiar with the platform and the possibility of potential problems in that respect is regarded as low. The current staff are generally very capable and are rated as very high, but the project is in a somewhat novel application domain for them so experience is rated as nominal. The toolsets available to the developers are judged to be typical for the size of company and are rated nominal, as it is the degree of schedule pressure to meet a deadline.

- What would be the value for each of the effort multipliers?
- What would be the impact of all the effort multipliers on a project estimated as taking 45 pm effort?

Factor	Description	Rating	Effort multiplier
RCPX	Product reliability and complexity	Very high	1.91
RUSE	Reuse	Very high	1.15
PDIF	Platform difficulty	Low	0.87
PERS	Personnel capability	Very high	0.63
PREX	Personnel experience	Nominal	1.00
FCIL	Facilities available	Nominal	1.00
SCED	Required development schedule	nominal	1.00

## New development effort multipliers (dem)

According to COCOMO, the major productivity drivers include:

**Product attributes:** required reliability, database size, product complexity

**Computer attributes:** execution time constraints, storage constraints, virtual machine (VM) volatility

**Personnel attributes:** analyst capability, application experience, VM experience, programming language experience

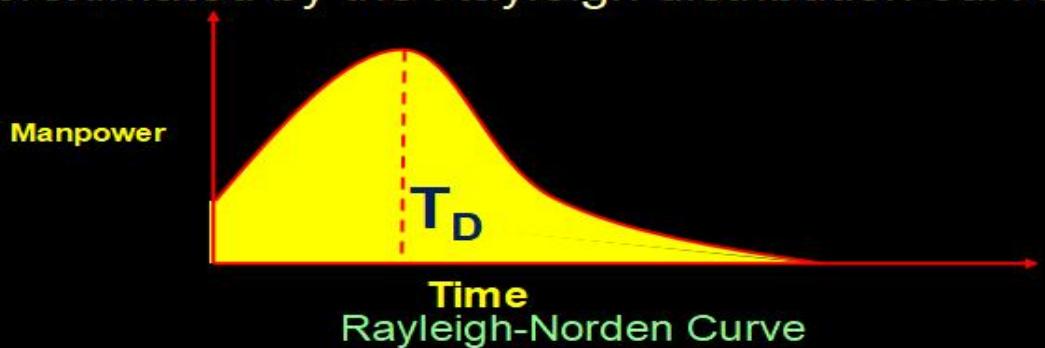
**Project attributes:** modern programming practices, software tools, schedule constraints

## COCOMO II Post architecture effort multipliers

<b>Modifier type</b>	<b>Code</b>	<b>Effort multiplier</b>
Product attributes	RELY	Required software reliability
	DATA	Database size
	DOCU	Documentation match to life-cycle needs
	CPLX	Product complexity
	REUSE	Required reusability
Platform attributes	TIME	Execution time constraint
	STOR	Main storage constraint
	PVOL	Platform volatility
Personnel attributes	ACAP	Analyst capability
	AEXP	Application experience
	PCAP	Programmer capabilities
	PEXP	Platform experience
	LEXP	Programming language experience
Project attributes	PCON	Personnel continuity
	TOOL	Use of software tools
	SITE	Multisite development
	SCED	Schedule pressure

# Staffing

- Norden was one of the first to investigate staffing pattern:
  - ↳ Considered general research and development (R&D) type of projects for efficient utilization of manpower.
- Norden concluded:
  - ↳ Staffing pattern for any R&D project can be approximated by the Rayleigh distribution curve



## Putnam's Work

- Putnam adapted the Rayleigh-Norden curve:
  - ↳ Related the number of delivered lines of code to the effort and the time required to develop the product.
  - ↳ Studied the **effect of schedule compression**:

$$pm_{new} = pm_{org} \times \left( \frac{td_{org}}{td_{new}} \right)^4$$

### Exercise:

The nominal effort and duration of a project is estimated to be 1000 pm and 15 months. This project is negotiated to be £200,000. This needs the product to be developed and delivered in 12 months time. What is the new cost that needs to be negotiated.

The project can be classified as a large project. Therefore the new cost to be negotiated can be given by Putnam's formula as

$$\text{New Cost} = \text{£200,000} \times (15/12)^4 = \text{£488,281}$$

## Boehm's Result

- There is a limit beyond which a software project cannot reduce its schedule by buying any more personnel or equipment.
  - This limit occurs roughly at 75% of the nominal time estimate for small and medium sized projects
  - If a project manager accepts a customer demand to compress the development schedule of a project (small or medium) by more than 25% , he is very unlikely to succeed.
  - The reason is, every project has a limited amount of activities which can be carried out in parallel and the sequential work can not be speeded up by hiring more number of additional developers.

# Capers Jones' Estimating Rules of Thumb

- Empirical rules: (IEEE journal – 1996)
  - ↳ Formulated based on observations
  - ↳ No scientific basis
- Because of their simplicity:
  - ↳ These rules are handy to use for making off-hand estimates. Not expected to yield very accurate estimates.
  - ↳ Give an insight into many aspects of a project for which no formal methodologies exist yet.

- Rule 1: SLOC-function point equivalence:
  - ↳ One function point = 125 SLOC for C programs.
- Rule 2: Project duration estimation:
  - ↳ Function points raised to the power 0.4 predicts the approximate development time in calendar months.
- Rule 3: Rate of requirements creep:
  - ↳ User requirements creep in at an average rate of 2% per month from the design through coding phases.

#### Illustration:

Size of a project is estimated to be 150 function points.

Rule-1:  $150 \times 125 = 18,750$  SLOC

Rule-2: Development time =  $150^{0.4} = 7.42 \approx 8$  months

Rule-3: The original requirement will grow by 2% per month i.e 2% of 150 is 3 FPs per month.

- If the duration of requirements specification and testing is 5 months out of total development time of 8 months, the total requirements creep will be roughly  $3 \times 5 = 15$  function points.
- The total size of the project considering the creep will be  $150 + 15 = 165$  function points and the manager need to plan on 165 function points.

## Rule 4: Defect removal efficiency:

- Each software review, inspection, or test step will find and remove 30% of the bugs that are present.  
(Companies use a series of defect removal steps like requirement review, code inspection, code walk-through followed by unit, integration and system testing. A series of ten consecutive defect removal operations must be utilized to achieve good product reliability.)

## Rule 5: Project manpower estimation:

- The size of the software (in function points) divided by 150 predicts the approximate number of personnel required for developing the application. (For a project size of 500 FPs the number of development personnel will be  $500/125 = 4$ , without considering other aspects like use of CASE tools, project complexity and programming languages.)

## Rule 6: Software development effort estimation:

- The approximate number of staff months of effort required to develop a software is given by the software development time multiplied with the number of personnel required. (using rule 2 and 5 the effort estimation for the project size of 150 FPs is  $8 \times 1 = 8$  person-months)

## Rule 7: Number of personnel for maintenance

- *Function points divided by 500 predicts the approximate number of personnel required for regular maintenance activities.* (as per Rule-1, 500 function points is equivalent to about 62,500 SLOC of C program, the maintenance personnel would be required to carry out minor fixing, functionality adaptation **ONE.**)

# END

# SPM 8.0

# **Activity planning in Software Project Management**

- In earlier study, we looked at methods for forecasting the effort required for a project – both for the project as a whole and for individual activities.
  - A detailed plan for the project, however, must also include a schedule indicating the start and completion times for each activity. This will enable us to:
    - *ensure that the appropriate resources will be available precisely when required;*
    - *avoid different activities competing for the same resources at the same time;*
    - *produce a detailed schedule showing which staff carry out each activity;*
    - *produce a detailed plan against which actual achievement may be measured;*
    - *produce a timed cash flow forecast;*
    - *replan the project during its life to correct drift from the target.*
- 
- A plan must be stated as a set of targets, the achievement or non-achievement of which can be unambiguously measured.
  - The activity plan does this by providing a target start and completion date for each activity. The starts and completions of activities must be clearly visible and this is one of the reasons why it is advisable to ensure that each and every project activity produces some ‘deliverable’

## Objectives of Activity Planning:::

- Feasibility assessment
- Resource allocation
- Detailed costing
- Motivation
- Coordination

*Activity planning and scheduling techniques place an emphasis on completing the project in a minimum time at an acceptable cost or, alternatively, meeting a set target date at minimum cost.*

## Project Schedules

**STEP 1-** Decide what activities need to be carried out and in what order they are to be done. From this we can construct an ideal activity plan – that is, a plan of when each activity would ideally be undertaken were resources not a constraint.

**STEP 2 -** The ideal activity plan will then be the subject of an activity risk analysis, aimed at identifying potential problems. This might suggest alterations to the ideal activity plan and will almost certainly have implications for resource allocation.

**STEP 3 -** The third step is resource allocation. The expected availability of resources might place constraints on when certain activities can be carried out, and our ideal plan might need to be adapted to take account of this.

**STEP 4 -** The final step is schedule production. Once resources have been allocated to each activity, we will be in a position to draw up and publish a project schedule, which indicates planned start and completion dates and a resource requirements statement for each activity.

## **Projects and Activities::::**

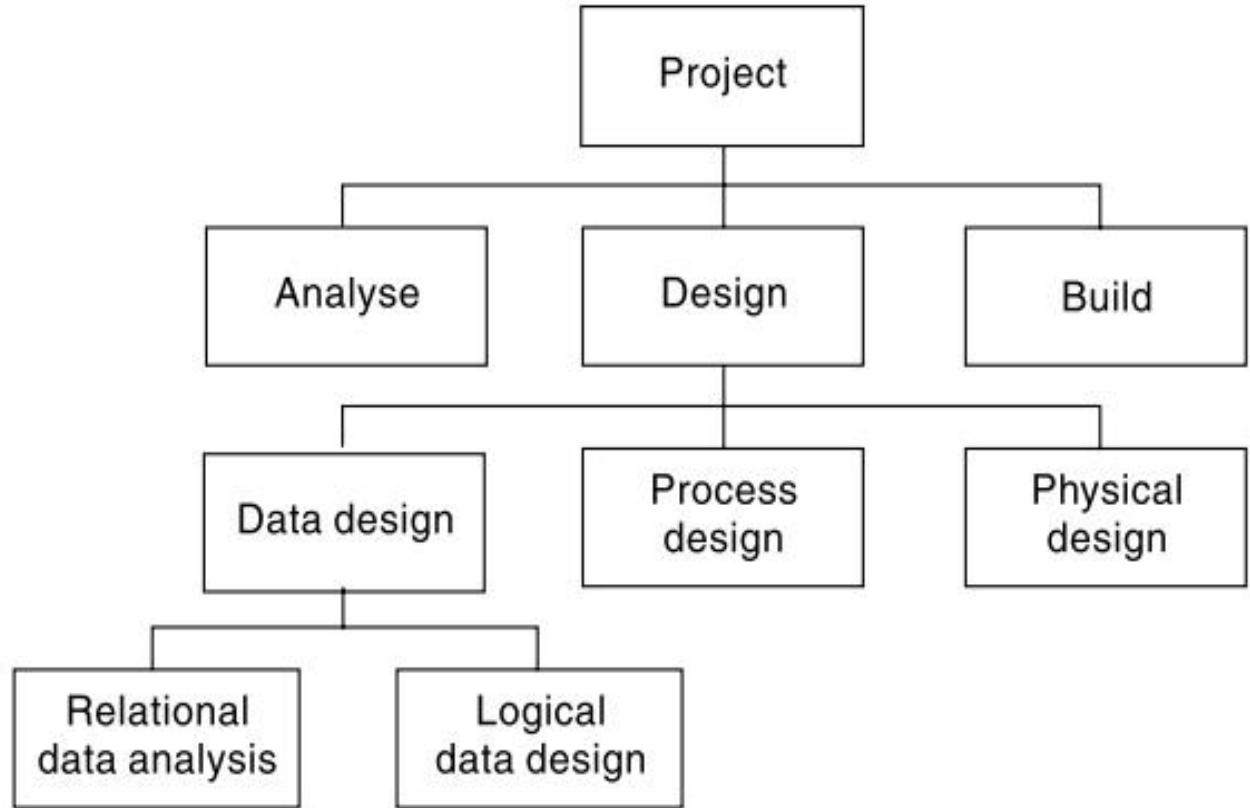
- ✓ A project is composed of a number of interrelated activities.
- ✓ A project may start when at least one of its activities is ready to start.
- ✓ A project will be completed when all of the activities it encompasses have been completed
- ✓ An activity must have a clearly defined start and a clearly defined end-point, normally marked by the production of a tangible deliverable.
- ✓ If an activity requires a resource (as most do) then that resource requirement must be forecastable and is assumed to be required at a constant level throughout the duration of the activity.
- ✓ The duration of an activity must be forecastable – assuming normal circumstances, and the reasonable availability of resources.
- ✓ Some activities might require that others are completed before they can begin

Essentially there are three approaches to identifying the activities or tasks that make up a project

- ❖ *Activity-based approach*
- ❖ *Product-based approach*
- ❖ *Hybrid approach*

## Activity-based approach

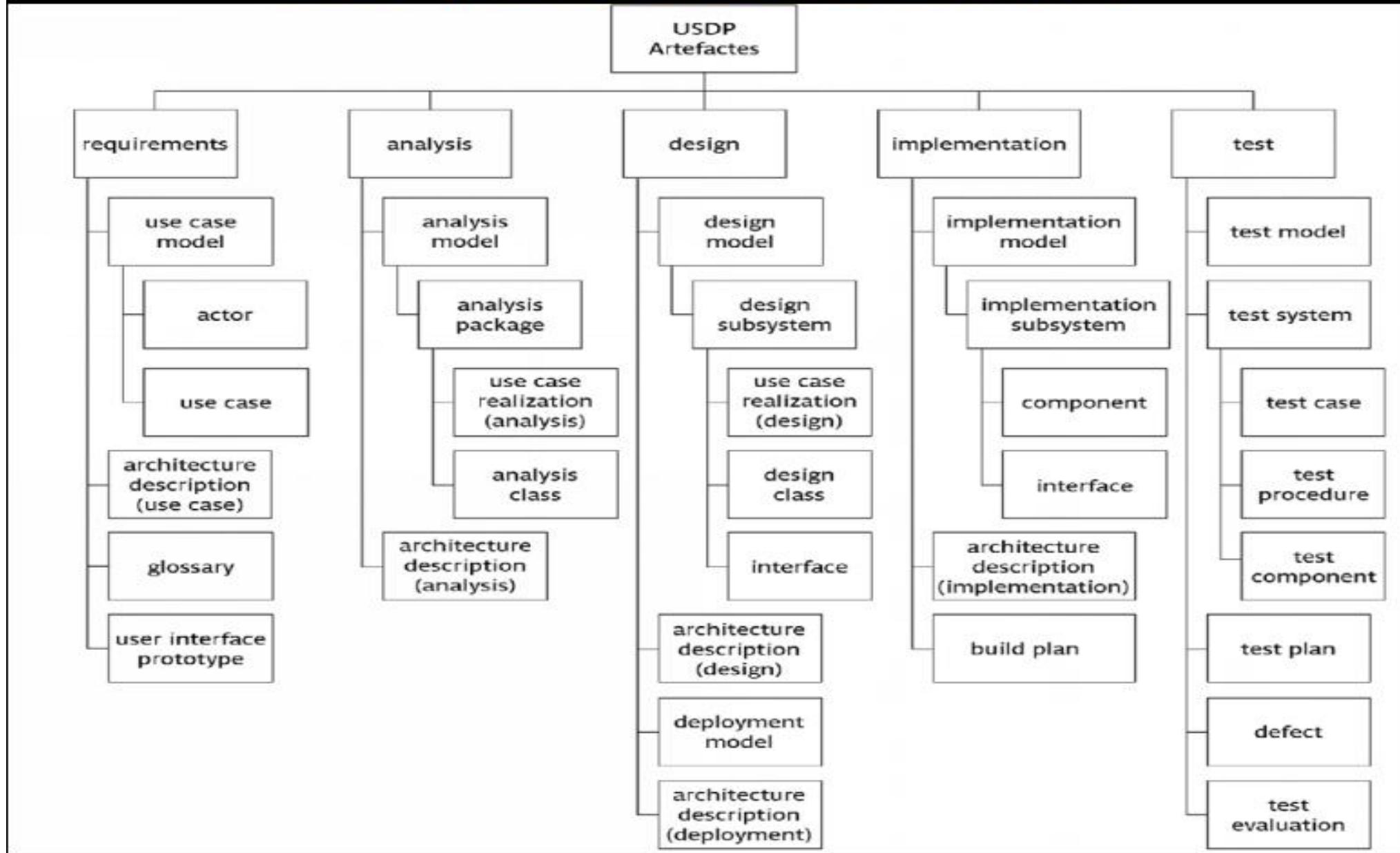
- It consists of creating an activities list that the project is thought to involve. When listing activities, particularly for a large project, it might be helpful to subdivide the project into the main life-cycle stages and consider each of these separately.
- This involves identifying the main (or high-level) tasks required to complete a project and then breaking each of these down into a set of lower-level tasks.
- Figure shows a fragment where the design task has been broken down into three tasks and one of these has been further decomposed into activities that are added to a branch in the structure if they contribute directly to the task immediately above.



## Product-based approach

- It consists of producing a Product Breakdown Structure and a Product Flow Diagram.
- It lists the deliverable and intermediate products of project – product breakdown structure (PBS)
- It Identify the order in which products have to be created & work out the activities needed to create the products
- Here, products are referred to as artifacts and the sequence of activities needed to create them is called a workflow

# USDP product breakdown structure



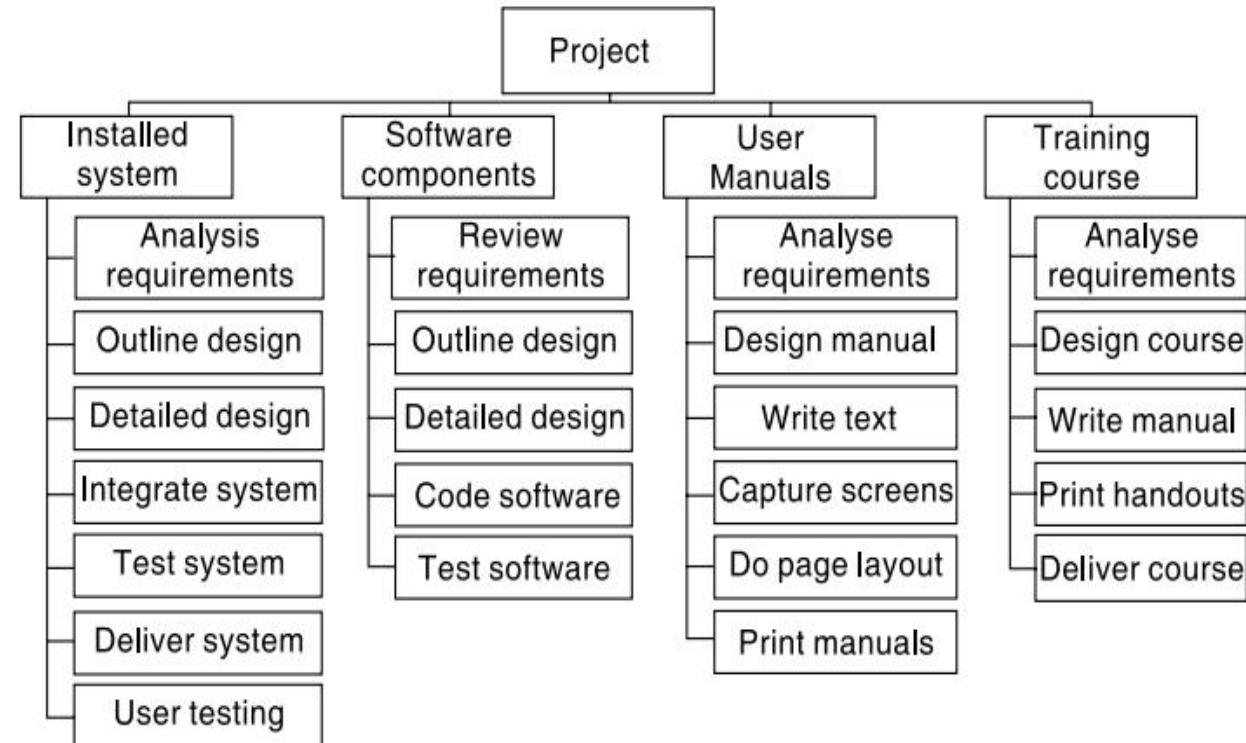
## Hybrid-based approach

In a project, it would be beneficial to introduce additional levels – structuring both products and activities. The degree to which the structuring is product-based or activity-based might be influenced by the nature of the project and the particular development method adopted.

A framework dictating the number of levels and the nature of each level in the structure may be imposed on a WBS.

The following five levels should be used in a WBS:

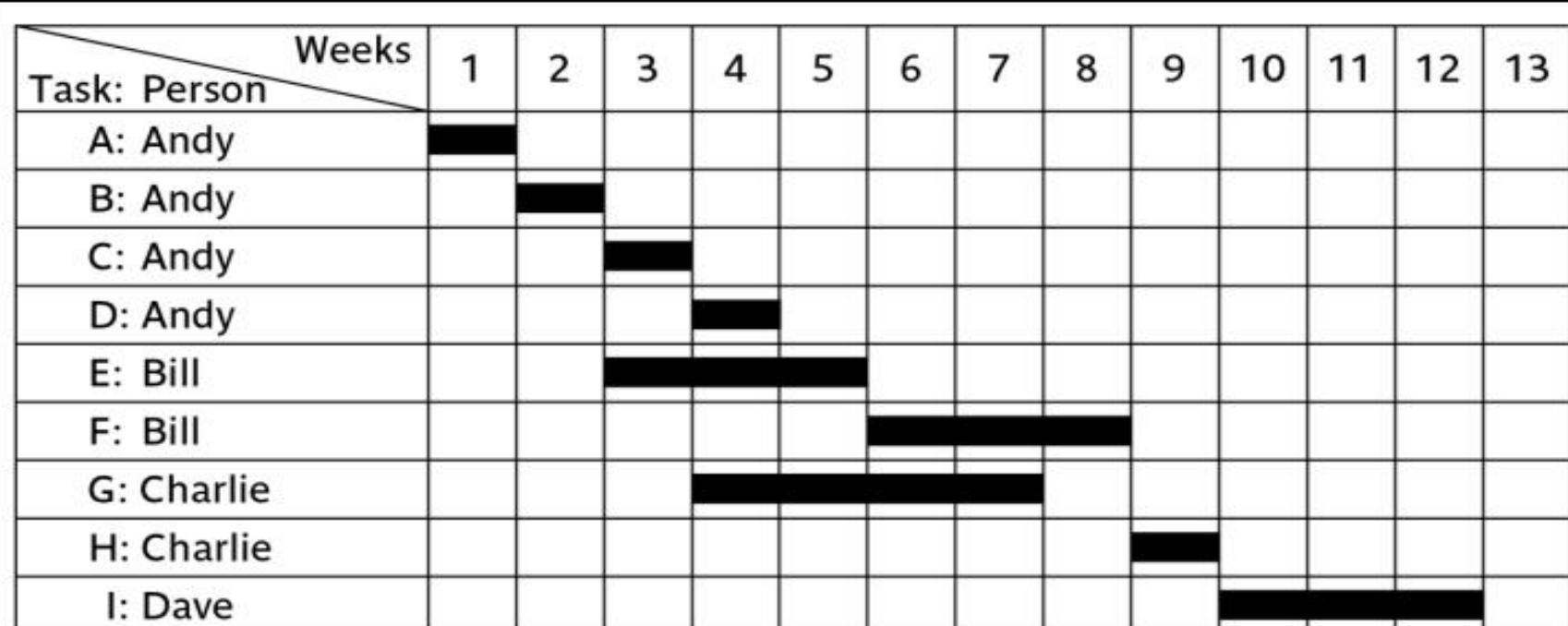
- **Level 1: Project.**
- **Level 2: Deliverables such as software, manuals and training courses.**
- **Level 3: Components, which are the key work items needed to produce deliverables, such as the modules and tests required to produce the system software.**
- **Level 4: Work-packages, which are major work items, or collections of related tasks, required to produce a component.**
- **Level 5: Tasks, which are tasks that will normally be the responsibility of a single person.**



## Sequencing and Scheduling Activities:::

# Final outcome of planning process

## A project plan as a bar chart



Activity key

A: Overall design

B: Specify module 1

C: Specify module 2

D: Specify module 3

E: Code module 1

F : Code module 3

G: Code module 2

H: Integration testing

I: System tesing

## Activity networks

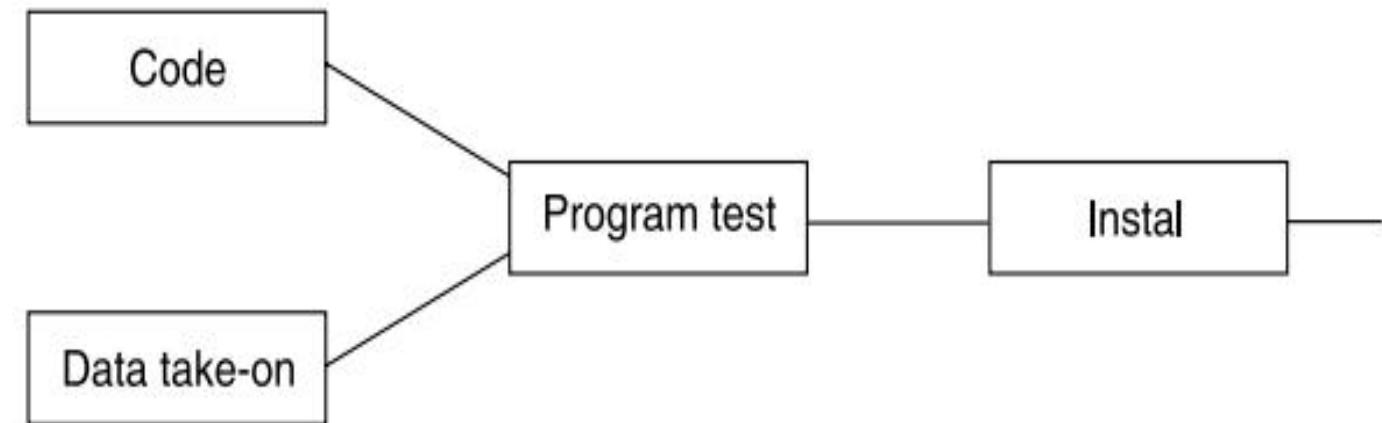
- Ensure that the appropriate resources will be available precisely when required.
- Avoid different activities competing for the same resources at the same time.
- Calculate when costs will be incurred.
- Produce a timed cash flow forecast.
- Produce a detailed schedule showing which staff carry out each activity.
- Produce a detailed plan against which actual achievement may be measured.
- Plan the project during its life to correct drift from the target.
- Activity plan will provide a target start and completion date for each activity.
- The start & completion of activities must be clearly visible & ensure each activity to produce some tangible product or 'deliverable'.
- The project will progress according to the plan.
- In case of deviation, identify the cause and plan to mitigate its effects.
- Activity plan provides a means of evaluating the consequences of not meeting the activity target dates and guide to bring the project back to target.

## Rules for Formulating a Network Model

- A project network should have only one start node
- A project network should have only one end node
- A node has duration
- Links normally have no duration
- Precedents are the immediate preceding activities
- Time moves from left to right
- A network may not contain loops
- A network should not contain dangles

'Program test' cannot start until both 'Code' and 'Data take-on' have been completed and activity 'Instal' cannot start until 'Program test' has finished.

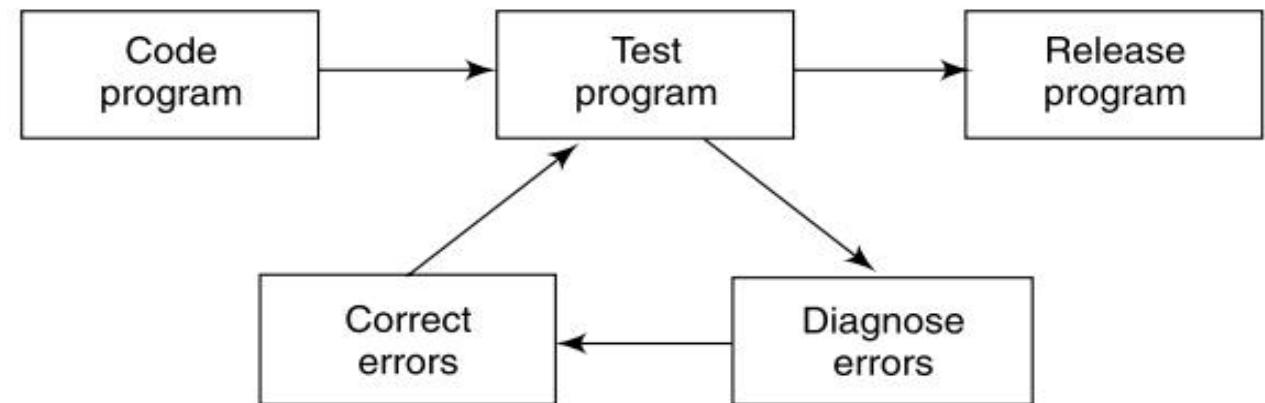
'Code' and 'Data take-on' can therefore be said to be precedents of 'Program test', and 'Program test' is a precedent of Instal'.



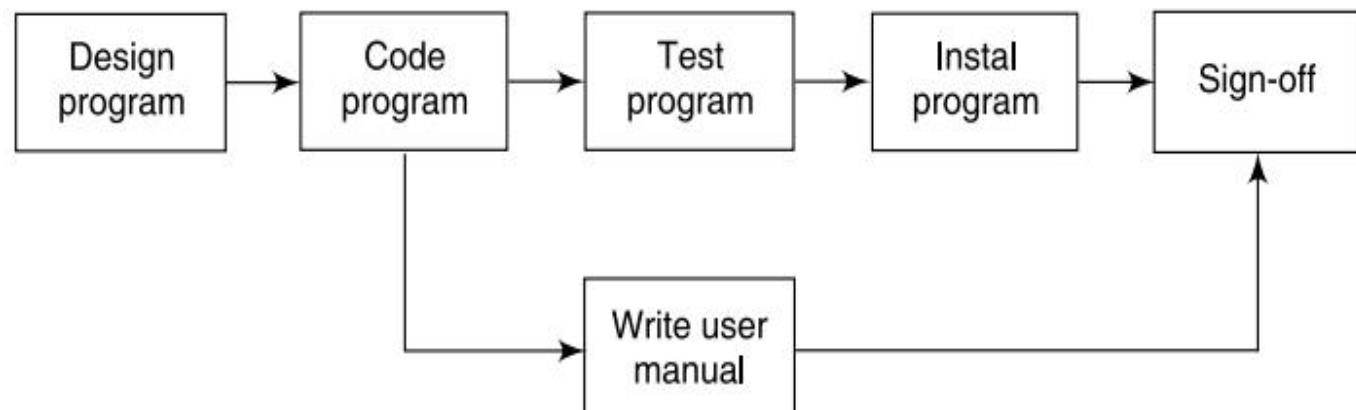
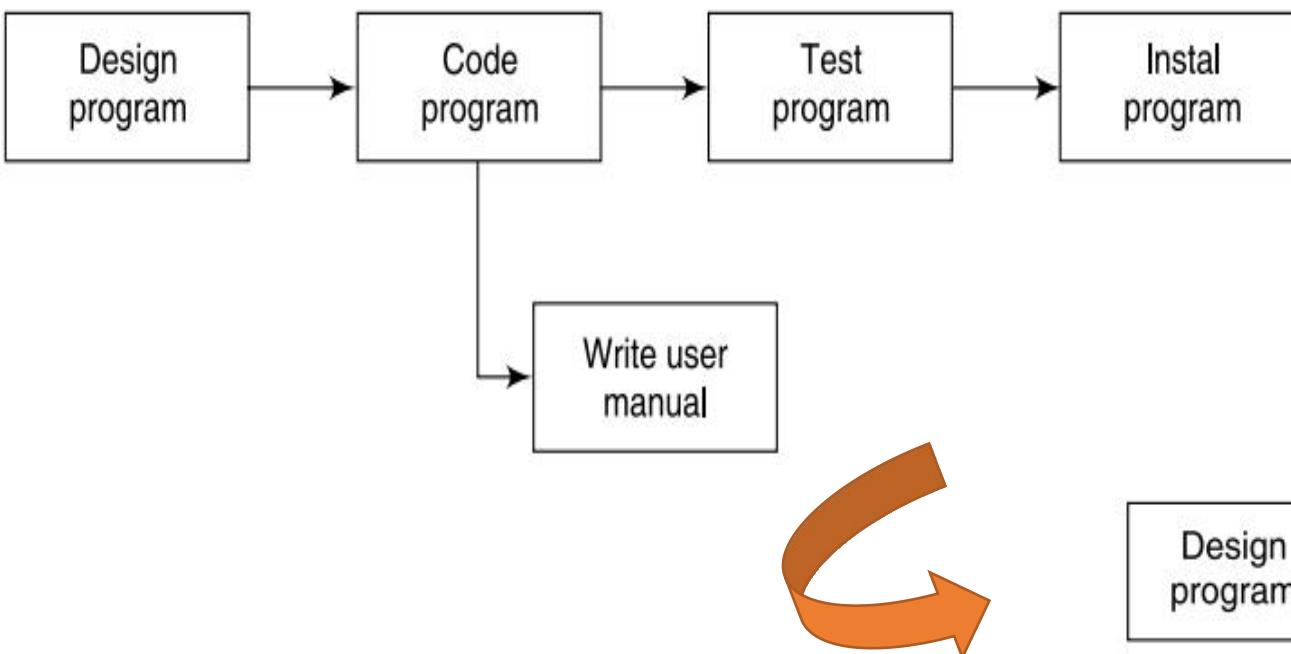
A dashed blue horizontal bar is present above this text.

A loop is an error in that it represents a situation that cannot occur in practice. While loops, in the sense of iteration, may occur in practice, they cannot be directly represented in a project network.

Note that program testing cannot start until the errors have been corrected.



- ❖ A dangling activity such as ‘Write user manual’ should not exist as it is likely to lead to errors in subsequent analysis. Indeed, in many cases dangling activities indicate errors in logic when activities are added as an afterthought.
- ❖ If, we mean to indicate that the project is complete once the software has been installed and the user manual written then we should redraw the network with a final completion activity – which, at least in this case, is probably a more accurate representation of what should happen.



## Start and finish times

Earliest start

activity

Latest  
finish

Latest start

Earliest finish

- Activity 'write report software'
- Earliest start (ES)
- Earliest finish (EF) = ES + duration
- Latest start(LS) = latest task can be completed without affecting project end Latest start = LF - duration

Earliest start	Duration	Earliest finish
Activity label, activity description		
Latest start	Float	Latest finish

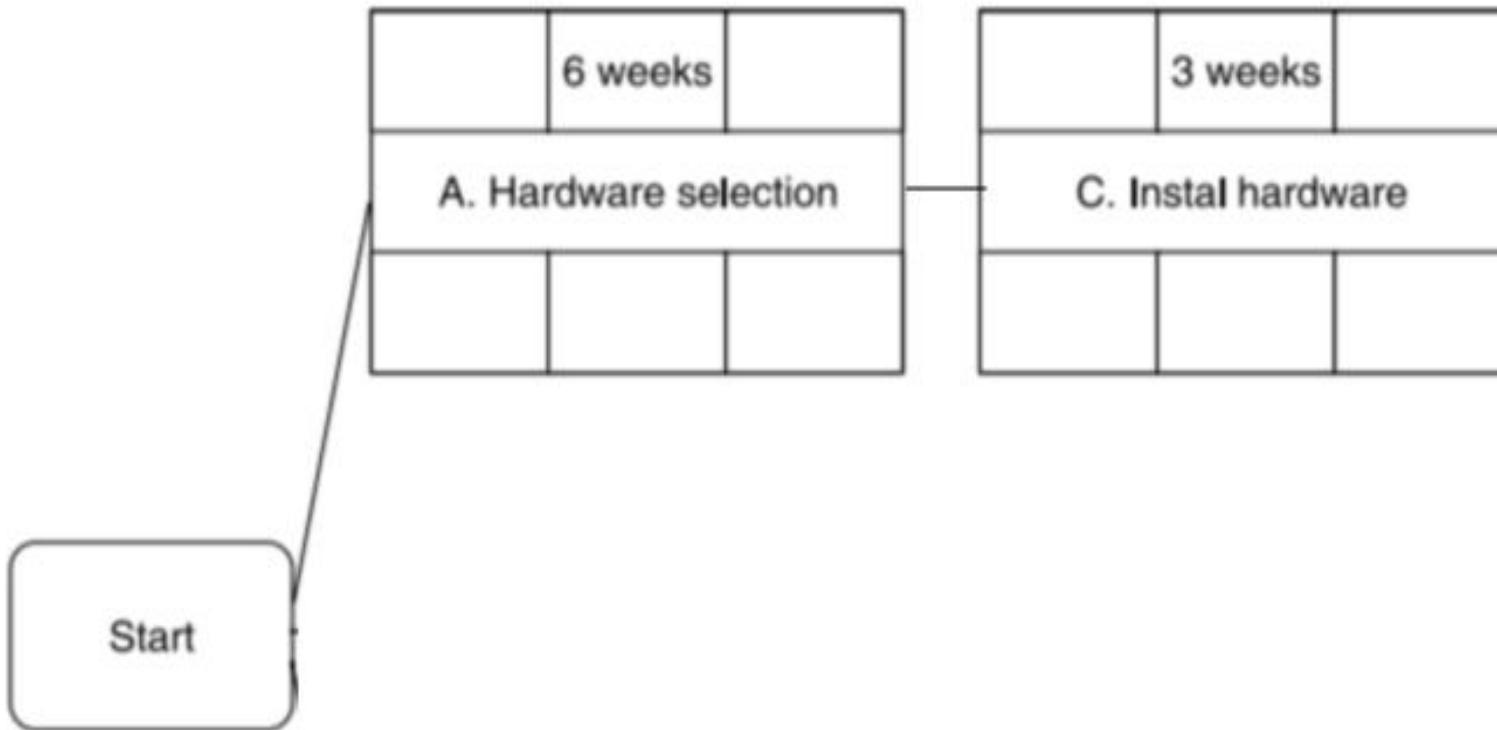
- ❖ Having created the logical network model indicating what needs to be done and the interrelationships between those activities, we are now ready to start thinking about when each activity should be undertaken.
- ❖ The critical path approach is concerned with two primary objectives: planning the project in such a way that it is completed as quickly as possible; and identifying those activities where a delay in their execution is likely to affect the overall end date of the project or later activities' start dates.
- ❖ The method requires that for each activity we have an estimate of its duration. The network is then analysed by carrying out a forward pass, to calculate the earliest dates at which activities may commence and the project be completed, and a backward pass, to calculate the latest start dates for activities and the critical path.

# The Forward Pass

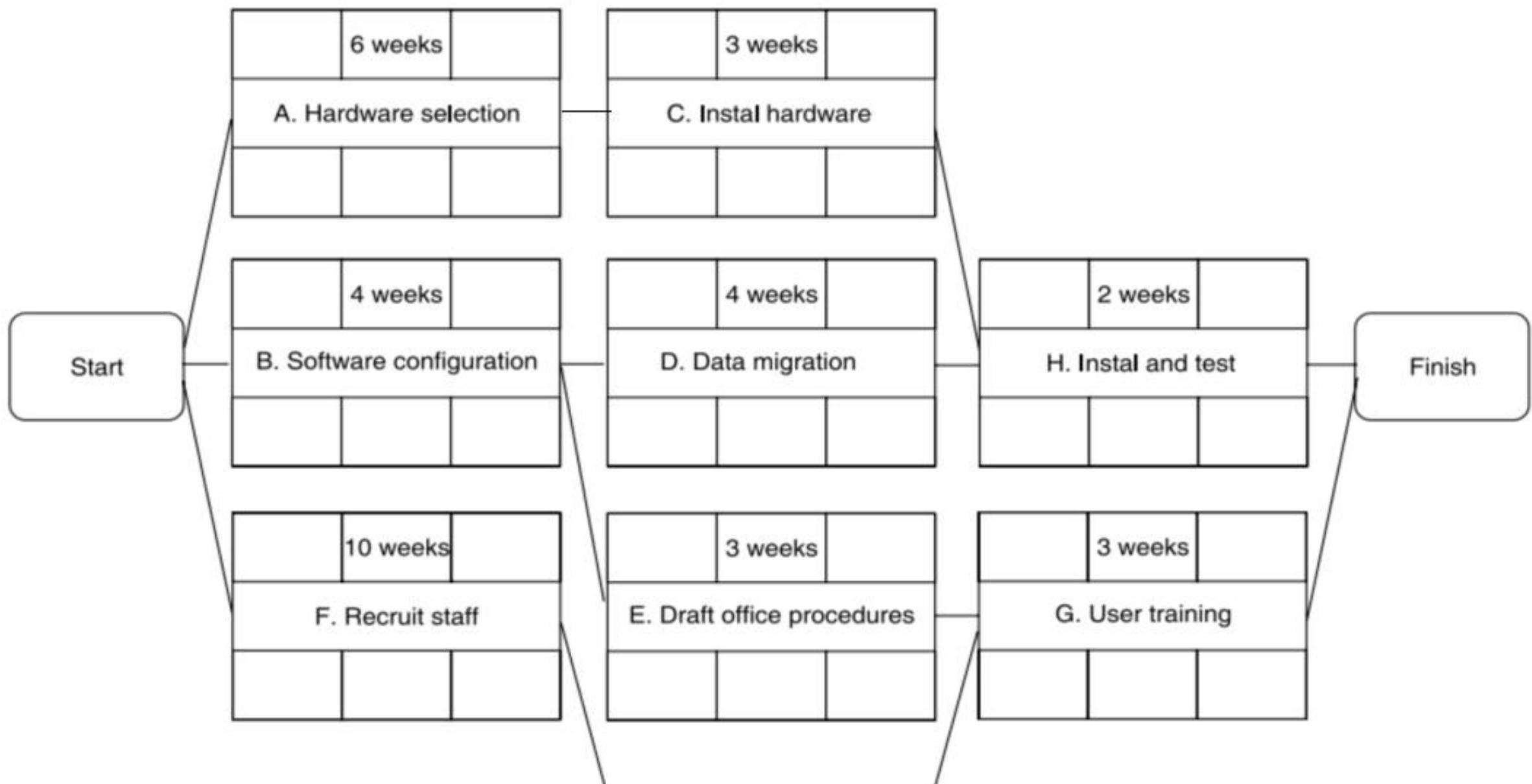
An example project specification with estimated activity durations and precedence requirements

<b>Activity</b>	<b>Duration (weeks)</b>	<b>Precedents</b>
A Hardware selection	6	
B System configuration	4	
C Instal hardware	3	A
D Data migration	4	B
E Draft office procedures	3	B
F Recruit staff	10	
G User training	3	E, F
H Instal and test system	2	C, D

## The Activity network for the previous example



## The Activity network for the previous example

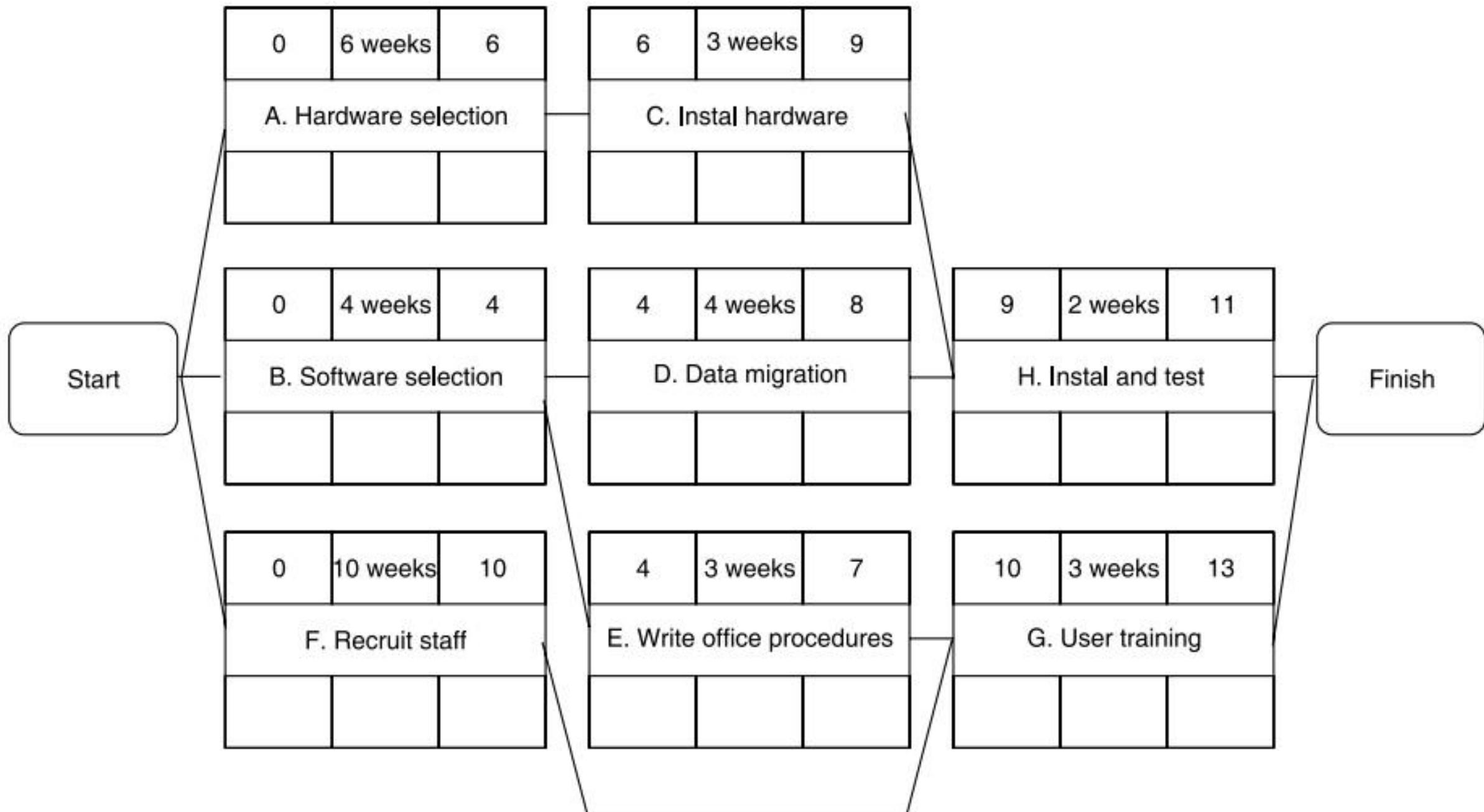


The forward pass and the calculation of earliest start dates are carried out according to the following reasoning.

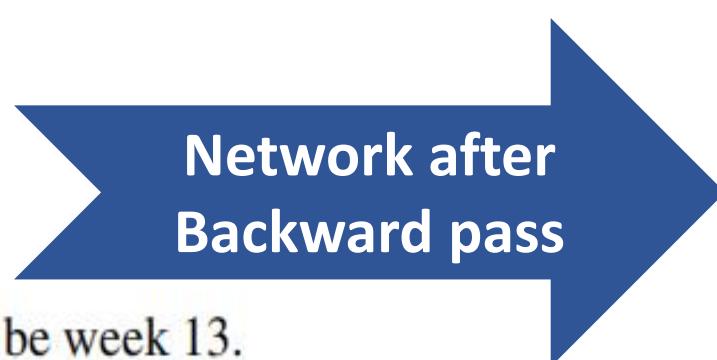
- Activities A, B and F may start immediately, so the earliest date for their start is zero.
- Activity A will take 6 weeks, so the earliest it can finish is week 6.
- Activity B will take 4 weeks, so the earliest it can finish is week 4.
- Activity F will take 10 weeks, so the earliest it can finish is week 10.
- Activity C can start as soon as A has finished so its earliest start date is week 6. It will take 3 weeks so the earliest it can finish is week 9.
- Activities D and E can start as soon as B is complete so the earliest they can each start is week 4. Activity D, which will take 4 weeks, can therefore finish by week 8 and activity E, which will take 3 weeks, can therefore finish by week 7.
- Activity G cannot start until both E and F have been completed. It cannot therefore start until week 10 – the later of weeks 7 (for activity E) and 10 (for activity F). It takes 3 weeks and finishes in week 13.
- Similarly, Activity H cannot start until week 9 – the later of the two earliest finish dates for the preceding activities C and D.
- The project will be complete when both activities H and G have been completed. Thus the earliest project completion date will be the later of weeks 11 and 13 – that is, week 13.



Network after  
Forward pass



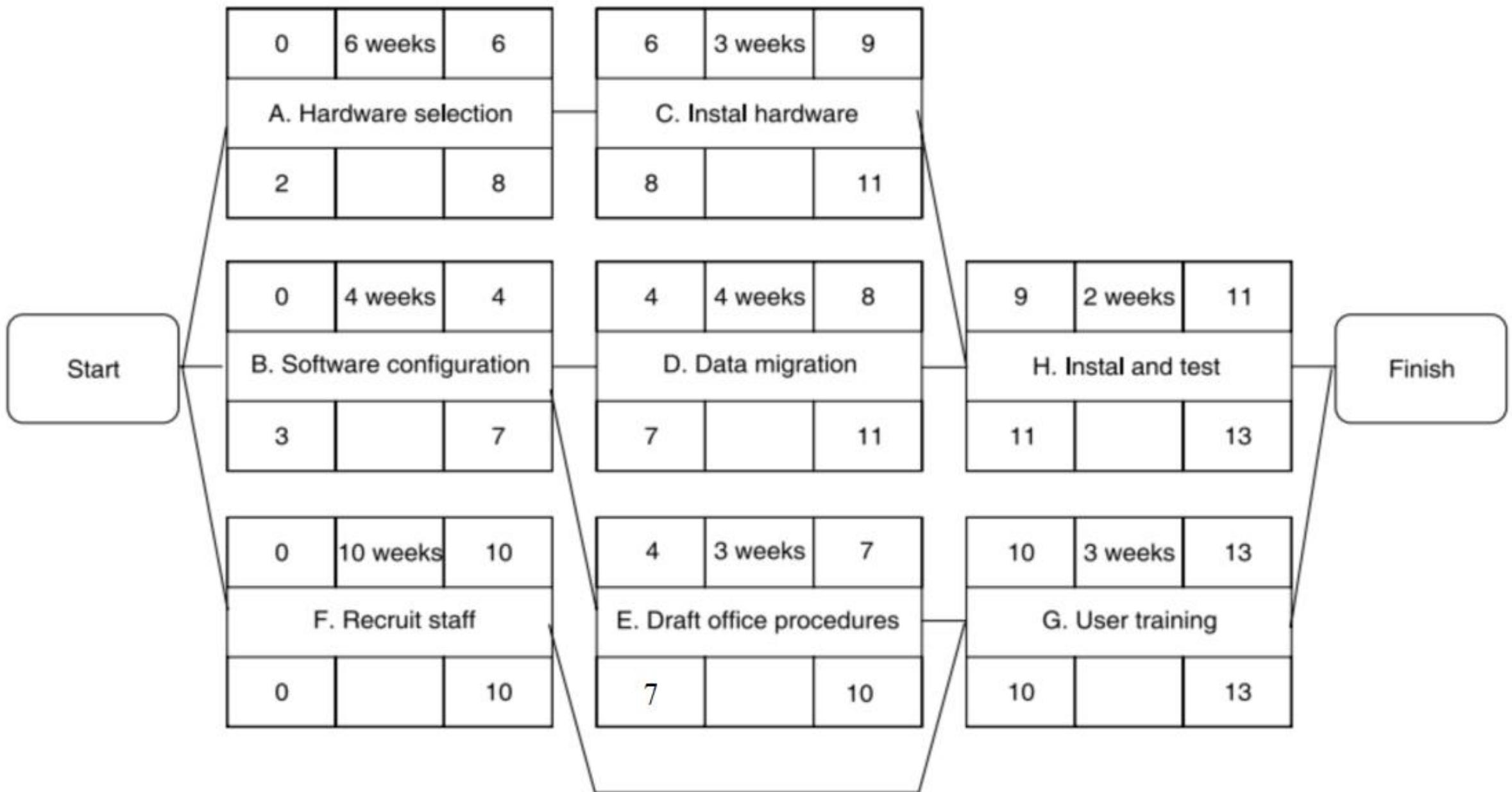
# The Backward Pass



## Network after Backward pass

The latest activity dates are calculated as follows.

- The latest completion date for activities G and H is assumed to be week 13.
- Activity H must therefore start at week 11 at the latest ( $13 - 2$ ) and the latest start date for activity G is week 10 ( $13 - 3$ ).
- The latest completion date for activities C and D is the latest date at which activity H must start – that is, week 11. They therefore have latest start dates of week 8 ( $11 - 3$ ) and week 7 ( $11 - 4$ ) respectively.
- Activities E and F must be completed by week 10 so their earliest start dates are weeks 7 ( $10 - 3$ ) and 0 ( $10 - 10$ ) respectively.
- Activity B must be completed by week 7 (the latest start date for both activities D and E) so its latest start is week 3 ( $7 - 4$ ).
- Activity A must be completed by week 8 (the latest start date for activity C) so its latest start is week 2 ( $8 - 6$ ).
- The latest start date for the project start is the earliest of the latest start dates for activities A, B and F. This is week zero. This is, of course, not very surprising since it tells us that if the project does not start on time it won't finish on time.



## Identifying the Critical Path

The path through network with zero floats. (**Float = Latest finish - Earliest start - Duration**)

The path through the network (that is, one set of successive activities) that defines the duration of the project. This is known as the critical path.

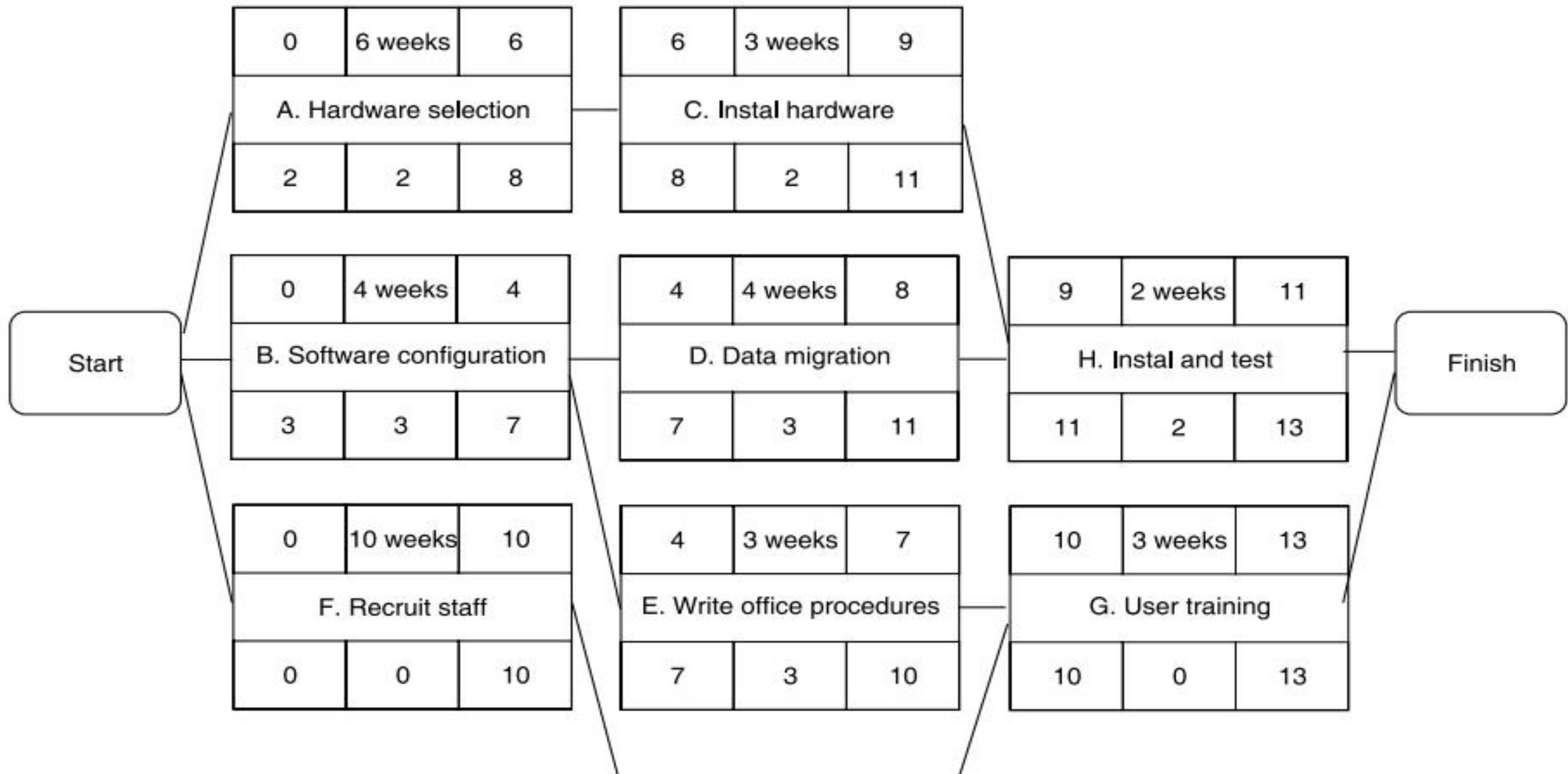
Any delay to any activity on this critical path will delay the completion of the project.



Critical  
Path

*The difference between an activity's earliest start date and its latest start date (or, equally, the difference between its earliest and latest finish dates) is known as the activity's float – it is a measure of how much the start or completion of an activity may be delayed without affecting the end date of the project.*

*Any activity with a float of zero is critical in the sense that any delay in carrying out the activity will delay the completion date of the project as a whole.*



## Activity Float

**Free float:** the time by which an activity may be delayed without affecting any subsequent activity. It is calculated as the difference between the earliest completion date for the activity and the earliest start date of the succeeding activity.

**Interfering float:** the difference between total float and free float. This is quite commonly used, particularly in association with the free float. Once the free float has been used (or if it is zero), the interfering float tells us by how much the activity may be delayed without delaying the project end date – even though it will delay the start of subsequent activities.

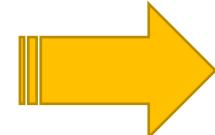
# END

# SPM 9.0

# RISK MANAGEMENT



## What is Risk?



*A problem that could cause some loss or threaten the progress of the project, but which has not happened yet. These potential issues might harm cost, schedule or technical success of the project and the quality of our software device, or project team morale.*

### Two characteristics of risk

- **Uncertainty**- the risk may or may not happen that means there are no 100% risks.
- **Loss** – If the risk occurs in reality , undesirable result or losses will occur.

## Risk Management



- ❑ Risk Management is the system of identifying, addressing and eliminating these problems before they can damage the project.
- ❑ It is a sequence of steps that help a software team to understand , analyze and manage uncertainty.
- ❑ In order to be identify the significant risks, it is essential to classify risks into different classes.

## Categories of risk:::

- ❖ **Project risks:** Project risks concern different forms of budgetary, schedule, personnel, resource, and customer-related problems. Since the software is intangible, it is very tough to monitor and control a software project. It is very tough to control something which cannot be identified. For any manufacturing program, such as the manufacturing of cars, the plan executive can recognize the product taking shape.
- ❖ **Technical risks:** Technical risks concern potential method, implementation, interfacing, testing, and maintenance issue. It also consists of an ambiguous specification, changing specification, technical uncertainty, and technical obsolescence. Most technical risks appear due to the development team's insufficient knowledge about the project.
- ❖ **Business risks:** This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.

**Example:** Let us consider a satellite based mobile communication project. The project manager can identify several risks in this project. Let us classify them appropriately.

- ✓ What if the project cost escalates and overshoots what was estimated? – Project Risk
- ✓ What if the mobile phones that are developed become too bulky in size to conveniently carry? Business Risk
- ✓ What if call hand-off between satellites becomes too difficult to implement? Technical Risk

## **The planning for risk includes these steps:**

- Risk identification – what risks might there be?
- Risk analysis and prioritization – which are the most serious risks?
- Risk planning – what are we going to do about them?
- Risk monitoring – what is the current state of the risk?



## Risk Management Activities

## Risk Assessment:

- The objective of risk assessment is to divide the risks in the condition of their loss, causing potential.
- For risk assessment, first, every risk should be rated in two methods:
  1. The possibility of a risk coming true (denoted as r).
  2. The consequence of the issues relates to that risk (denoted as s).
- Based on these two methods, the priority (risk exposure of each risk can be estimated as::::::::::  $p = r * s$

*where p is the priority with which the risk must be controlled, r is the probability of the risk becoming true and s is the severity of loss caused due to the risk becoming true.*

**Risk Identification:** The project organizer needs to anticipate the risk in the project as early as possible so that the impact of risk can be reduced by making effective risk management planning.

**Technology risks:** Risks that assume from the software or hardware technologies that are used to develop the system.

**People risks:** Risks that are connected with the person in the development team.

**Organizational risks:** Risks that assume from the organizational environment where the software is being developed.

**Tools risks:** Risks that assume from the software tools and other support software used to create the system.

**Requirement risks:** Risks that assume from the changes to the customer requirement and the process of managing the requirements change.

**Estimation risks:** Risks that assume from the management estimates of the resources required to build the system

**Risk Analysis:** During the risk analysis process, you have to consider every identified risk and make a perception of the probability and seriousness of that risk.

It is not possible to make an exact, the numerical estimate of the probability and seriousness of each risk. Instead, you should authorize the risk to one of several bands:

1. The probability of the risk might be determined as very low (0-10%), low (10-25%), moderate (25-50%), high (50-75%) or very high (+75%).
2. The effect of the risk might be determined as catastrophic (threaten the survival of the plan), serious (would cause significant delays), tolerable (delays are within allowed contingency), or insignificant.

## Risk Control:

- It is the process of managing risks to achieve desired outcomes. After all the identified risks of a plan are determined; the project must be made to include the most harmful and the most likely risks. Different risks need different containment methods.

There are three main methods to plan for risk management:

1. **Avoid the risk:** This may take several ways such as discussing with the client to change the requirements to decrease the scope of the work, giving incentives to the engineers to avoid the risk of human resources turnover, etc.
2. **Transfer the risk:** This method involves getting the risky element developed by a third party, buying insurance cover, etc.
3. **Risk reduction:** This means planning method to include the loss due to risk. For instance, if there is a risk that some key personnel might leave, new recruitment can be planned.

**Risk Leverage:** To choose between the various methods of handling risk, the project plan must consider the amount of controlling the risk and the corresponding reduction of risk. For this, the risk leverage of the various risks can be estimated.

$$\text{Risk leverage} = (\text{risk exposure before reduction} - \text{risk exposure after reduction}) / (\text{cost of reduction})$$

**Risk planning:** The risk planning method considers each of the key risks that have been identified and develop ways to maintain these risks.

For each of the risks, the action to minimize the disruption to the plan if the issue identified in the risk occurs should be considered. Again, it rely on the judgment and experience of the project manager.

**Risk Monitoring:** Risk monitoring is the method that your assumption about the product, process, and business risks has not changed.

Ref	Hazard	Likelihood	Impact	Risk
R1	Changes to requirements specification during coding	8	8	64
R2	Specification takes longer than expected	3	7	21
R3	Significant staff sickness affecting critical path activities	5	7	35
R4	Significant staff sickness affecting non-critical activities	10	3	30
R5	Module coding takes longer than expected	4	5	20
R6	Module testing demonstrates errors or deficiencies in design	4	8	32

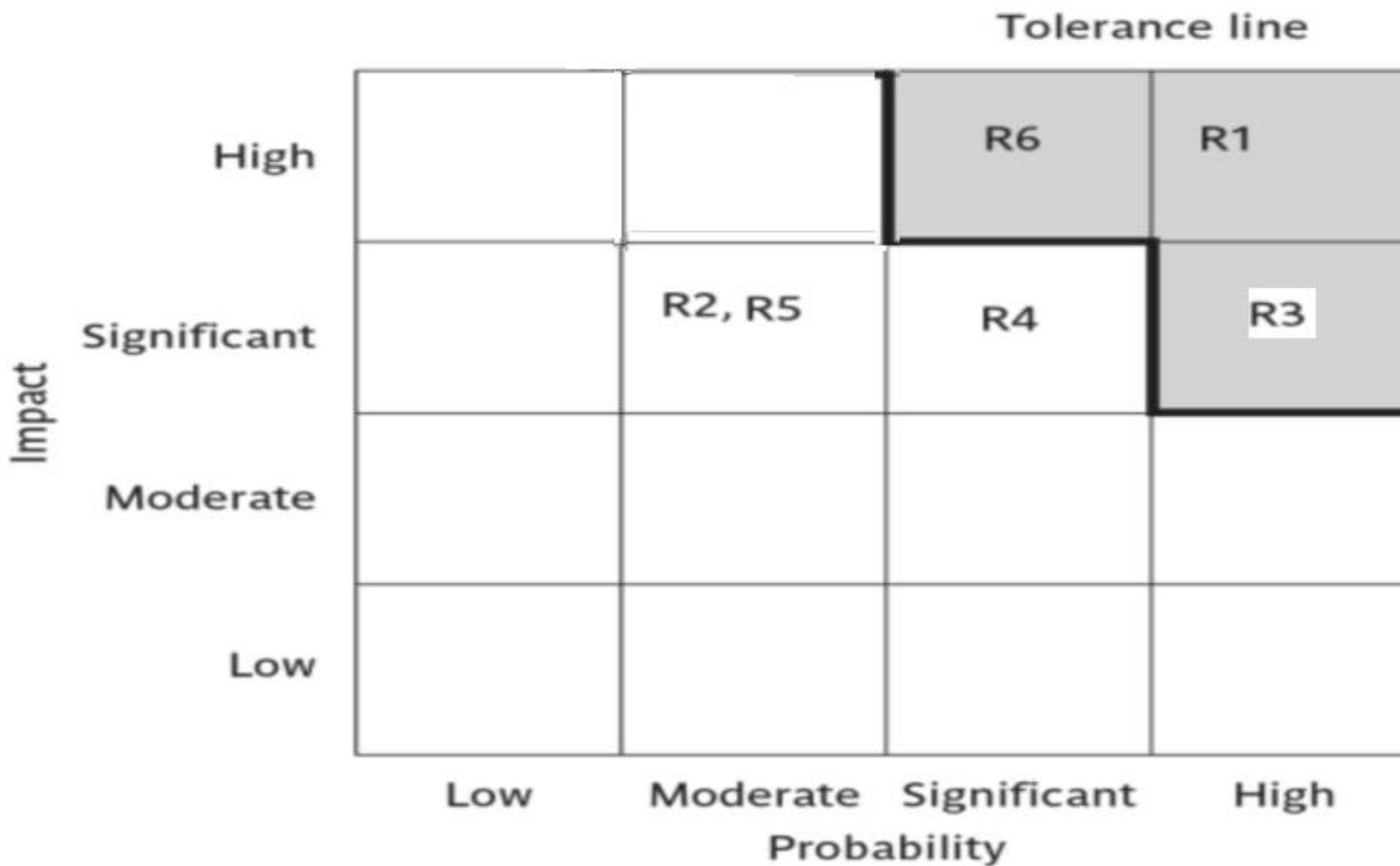
## Qualitative descriptors of risk probability and associated range values

Probability level	Range
High	Greater than 50% chance of happening
Significant	30–50% chance of happening
Moderate	10–29% chance of happening
Low	Less than 10% chance of happening

## Qualitative descriptors of impact on cost and associated range values

Impact level	Range
High	More than 30% above budgeted expenditure
Significant	20 to 29% above budgeted expenditure
Moderate	10 to 19% above budgeted expenditure
Low	Within 10% of budgeted expenditure.

## Probability impact matrix



Risks can be dealt with by:

- Risk acceptance
- Risk avoidance
- Risk reduction
- Risk transfer
- Risk mitigation/contingency measures

**Consider a fire control system where there is 1% chance of a fire causing 200K damage. It is further estimated that a fire alarm costing 500 INR can reduce the probability of fire damage to 0.5%.**

**Calculate the Risk Leverage Level. Also check if the system is worth implementing.**

## Risk reduction leverage =

$$(RE_{\text{before}} - RE_{\text{after}}) / (\text{cost of risk reduction})$$

$RE_{\text{before}}$  is risk exposure before risk reduction e.g. 1% chance of a fire causing £200k damage

$RE_{\text{after}}$  is risk exposure after risk reduction e.g. fire alarm costing Rs. 500 reduces probability of fire damage to 0.5%

$$\text{RRL} = (1\% \text{ of Rs. } 200\text{k}) - (0.5\% \text{ of Rs. } 200\text{k}) / \text{Rs. } 500 = 2$$

$\text{RRL} > 1.00$  therefore worth doing

## Evaluating the effects of uncertainty

Three estimates are produced for each activity

- *Most likely time (m)*
  - *Optimistic time (a)*
  - *Pessimistic (b)*
- 
- *Most likely time*: the time we would expect the task to take under normal circumstances. We shall identify this by the letter *m*.
  - *Optimistic time*: the shortest time in which we could expect to complete the activity, barring outright miracles. We shall use the letter *a* for this.
  - *Pessimistic time*: the worst possible time, allowing for all reasonable eventualities but excluding ‘acts of God and warfare’ (as they say in most insurance exclusion clauses). We shall call this *b*.



**Expected time:::**

$$t_e = (a + 4m + b) / 6$$

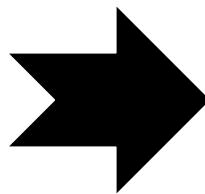
**Activity standard deviation::::**

$$S = (b-a)/6$$

Task	a	m	b	$t_e$	s
A	10	12	16	?	?
B	8	10	14	?	?
C	20	24	38	?	?



<b>Task</b>	<b>a</b>	<b>m</b>	<b>b</b>	<b>t<sub>e</sub></b>	<b>s</b>
<b>A</b>	10	12	16	?	?
<b>B</b>	8	10	14	?	?
<b>C</b>	20	24	38	?	?



**What would be the expected duration of the chain A + B + C?**

**Answer:  $12.66 + 10.33 + 25.66$  i.e. 48.65**

**What would be the standard deviation for A + B+ C?**

**Answer: square root of  $(1^2 + 1^2 + 3^2)$  i.e. 3.32**

<b>Activity</b>	<b>Optimistic (<i>a</i>)</b>	<b>Activity durations (weeks). Most likely (<i>m</i>)</b>	<b>Pessimistic (<i>b</i>)</b>
A	5	6	8
B	3	4	5
C	2	3	3
D	3.5	4	5
E	1	3	4
F	8	10	15
G	2	3	4
H	2	2	2.5

Activity	Activity durations (weeks)				
	Optimistic ( <i>a</i> )	Most likely ( <i>m</i> )	Pessimistic ( <i>b</i> )	Expected ( <i>t<sub>e</sub></i> )	Standard deviation ( <i>s</i> )
A	5	6	8	6.17	0.50
B	3	4	5	4.00	0.33
C	2	3	3	2.83	0.17
D	3.5	4	5	4.08	0.25
E	1	3	4	2.83	0.50
F	8	10	15	10.50	1.17
G	2	3	4	3.00	0.33
H	2	2	2.5	2.08	0.08

# END

# **SPM 10.0**

# **Resource allocation in Software Project Management**

- ❖ A resource is any item or person required for the execution of the project.
- ❖ One of the essential components of an effective project management (PM) process is resource allocation.
- ❖ Assigning the right resources to the proper projects requires preliminary preparation, specific knowledge, and experience. Robust allocation processes affect successful project realization.

*Activity schedule* - indicating start and completion dates for each activity

*Resource schedule* - indicating dates when resources needed + level of resources

*Cost schedule* - showing accumulative expenditure

# Types of resources

- ❑ **Labor.** These are various professionals, employees, and team members with different skills needed to complete a project successfully.
- ❑ **Equipment.** It includes necessary tools for project completion: from software or hardware to hammer or drill, depending on a company's specifics.
- ❑ **Facilities.** It is all about an environment for work and project realization such as offices, open space, meeting rooms, etc.
- ❑ **Materials.** In other words, it is the stuff needed to produce outputs: from pens and paper to raw material for house building.
- ❑ **Budget.** Probably, the most important type that allows buying all we listed above



# What is the purpose of resource allocation?

- Forecast and avoid resource scarcity
- Prevent employees' burnout
- Make workflow more transparent
- Meet deadlines
- Avoid extra spending



## What are the challenges of resource allocation?

- Changes in project
- A lack of qualified resources
- Poor resource capacity measuring
- Changes in resource availability
- Not assessed risks
- Task delays
- Working with international teams

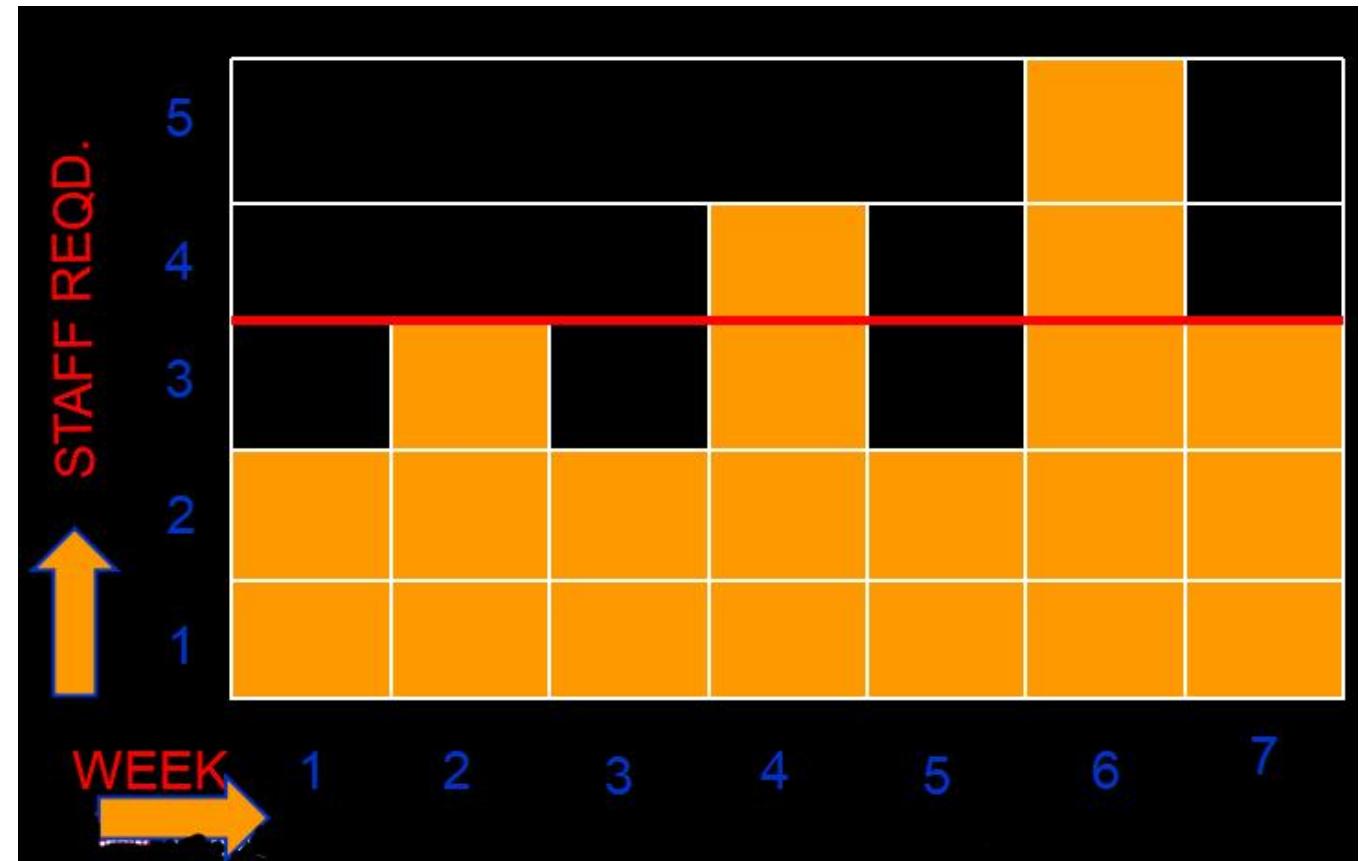


Identify the resources needed for each activity and create a *resource requirement list*

Identify *resource types* - individuals are interchangeable within the group

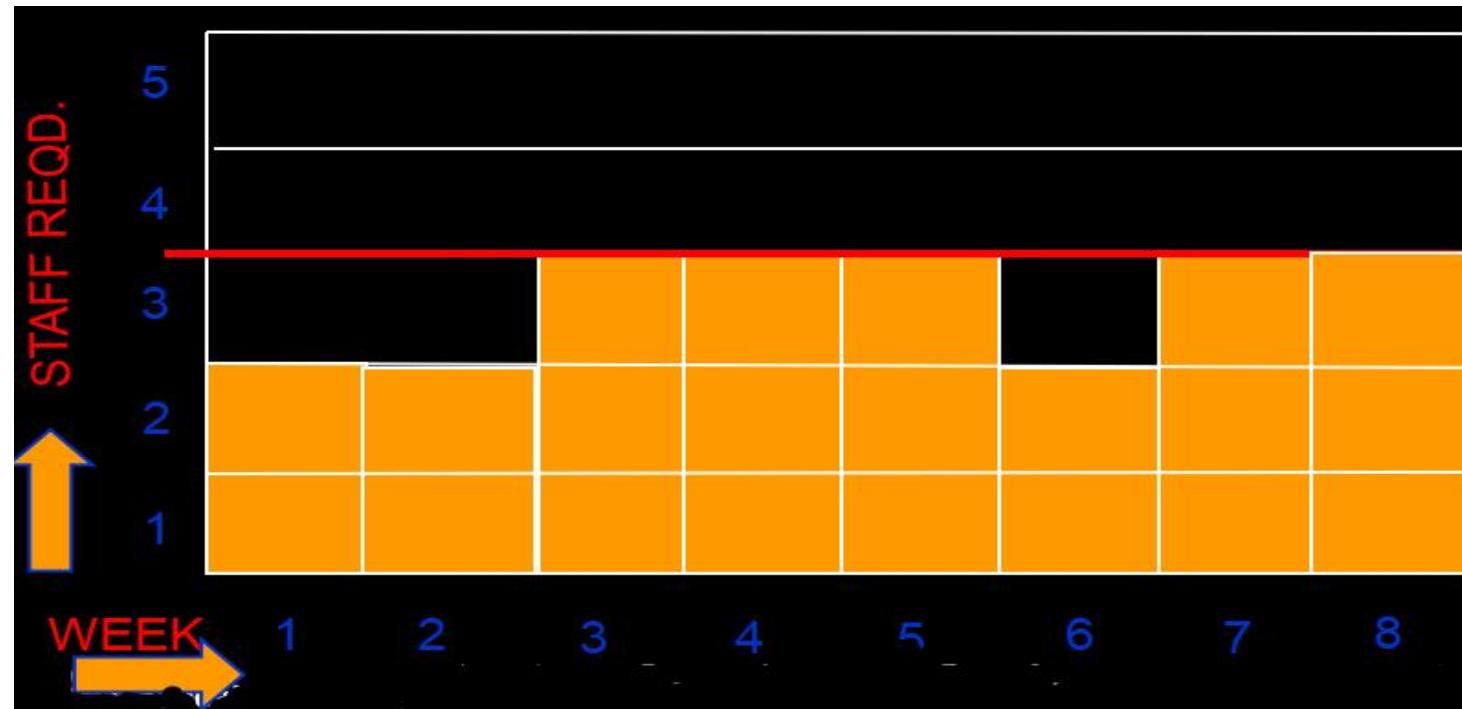
Allocate resource types to activities and examine the *resource histogram*

Resource histogram: Systems analysts



## Resource smoothing

- It is usually difficult to get specialist staff who will work odd days to fill in gaps – need for staff to learn about application etc
- Staff often have to be employed for a continuous block of time. Therefore desirable to employ a constant number of staff on a project – who as far as possible are fully employed
- Hence need for resource smoothing.
- Technique used to optimize resources to deliver a project without changing the delivery date. It might involve hiring new people to speed things up, or changing the scope of the project.
- The objective is to complete the work or activity within the required date and, at the same time, avoiding peaks and troughs of the resource demand.



## Resource clashes

- 
- Where same resource needed in more than one place at the same time
  - can be resolved by:
    - ❖ delaying one of the activities
      - taking advantage of float to change start date
      - delaying start of one activity until finish of the other activity that resource is being used on - *puts back project completion*
    - ❖ moving resource from a non-critical activity
    - ❖ bringing in additional resource - *increases costs*

### Prioritizing activities

There are two main ways of doing this:

- *Total float priority* – those with the smallest float have the highest priority
- *Ordered list priority* – this takes account of the duration of the activity as well as the float

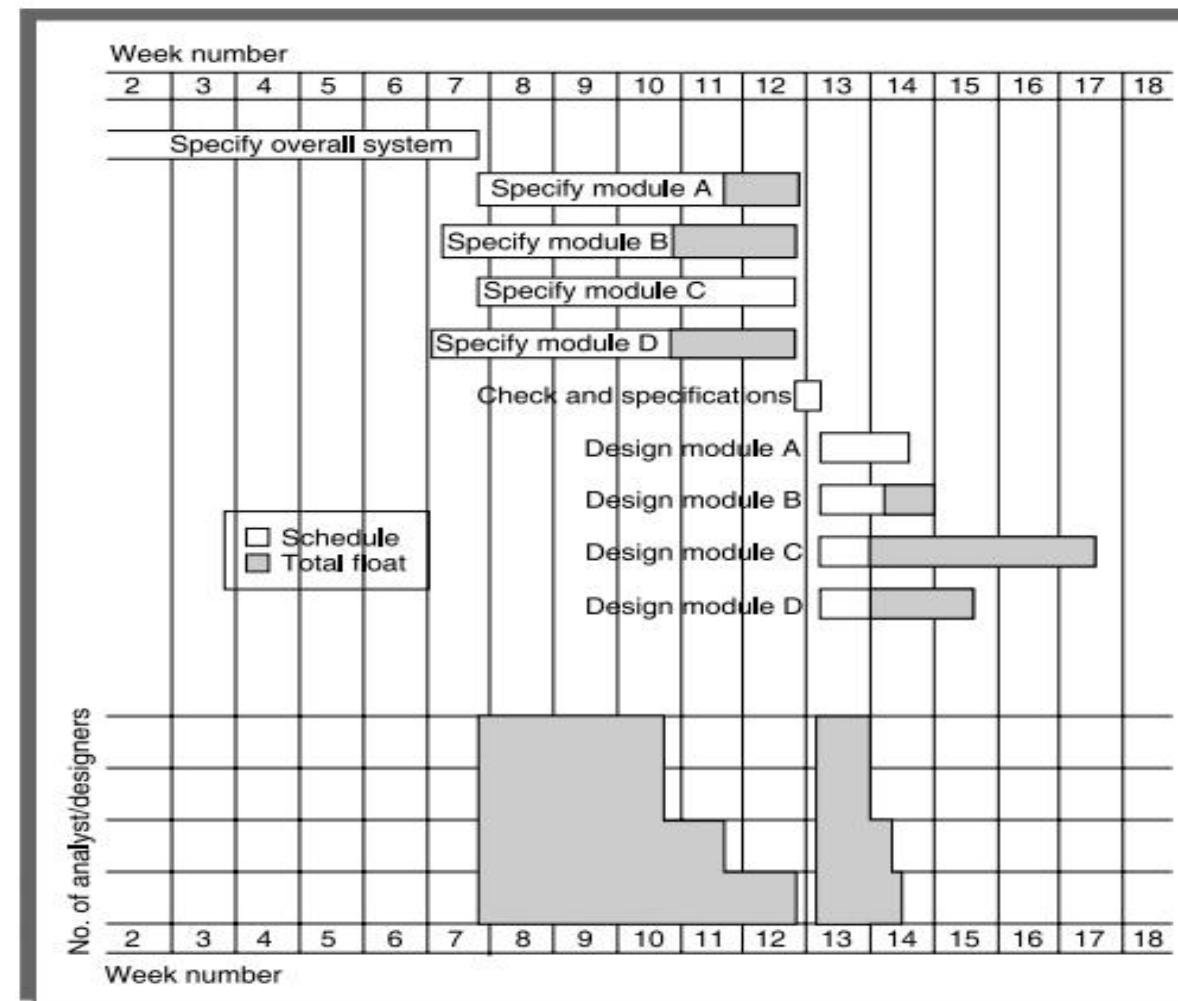
## Identifying Resource Requirements

- The first step in producing a resource allocation plan is to list the resources that will be required along with the expected level of demand. This will normally be done by considering each activity in turn and identifying the resources required.
- It is likely, however, that there will also be resources required that are not activity specific but are part of the project's infrastructure (such as the project manager) or required to support other resources (office space, for example, might be required to house contract software developers)

Stage	Activity	Resource	Days	Quantity	Notes
ALL	Project manager		104 F/T		
1	All	Workstation	–	34	Check software availability
	IoE/P/1	Senior analyst	34 F/T		
2	All	Workstation	–	3	One per person essential
	IoE/P/2	Analyst/designer	20 F/T		
	IoE/P/3	Analyst/designer	15 F/T		

## Scheduling Resources

- Having produced the resource requirements list, the next stage is to map this on to the activity plan to assess the distribution of resources required over the duration of the project. This is best done by representing the activity plan as a bar chart and using this to produce a resource histogram for each resource.



## Creating Critical Paths

- Scheduling resources can create new critical paths. Delaying the start of an activity because of lack of resources will cause that activity to become critical if this uses up its float.
- Furthermore, a delay in completing one activity can delay the availability of a resource required for a later activity. If the later one is already critical then the earlier one might now have been made critical by linking their resources.

## Counting the Cost

- The discussion so far has concentrated on trying to complete the project by the earliest completion date with the minimum number of staff. We have seen that doing this places constraints on when activities can be carried out and increases the risk of not meeting target dates.
- Alternatively, using additional staff or lengthening the overall duration of the project is also an option. The additional costs of employing extra staff would need to be compared to the costs of delayed delivery and the increased risk of not meeting the scheduled date.

## Allocating individuals to tasks

**Availability** - We need to know whether a particular individual will be available when required.

**Criticality** - Allocation of more experienced personnel to activities on the critical path often helps in shortening project durations or at least reduces the risk of overrun.

**Risk** - Identifying those activities posing the greatest risk, and knowing the factors influencing them, helps to allocate staff.

**Training** - It will benefit the organization if positive steps are taken to allocate junior staff to appropriate non-critical activities where there will be sufficient slack for them to train and develop skills. There can even be direct benefits to the particular project since some costs may be allocated to the training budget.

**Team building** - The selection of individuals must also take account of the final shape of the project team and the way they will work together.

## Publishing the Resource Schedule

- In allocating and scheduling resources we have used the activity plan, activity bar charts and resource histograms. Although good as planning tools, they are not the best way of publishing and communicating project schedules.
- For this we need some form of work plan. Work plans are commonly published as either lists or charts.

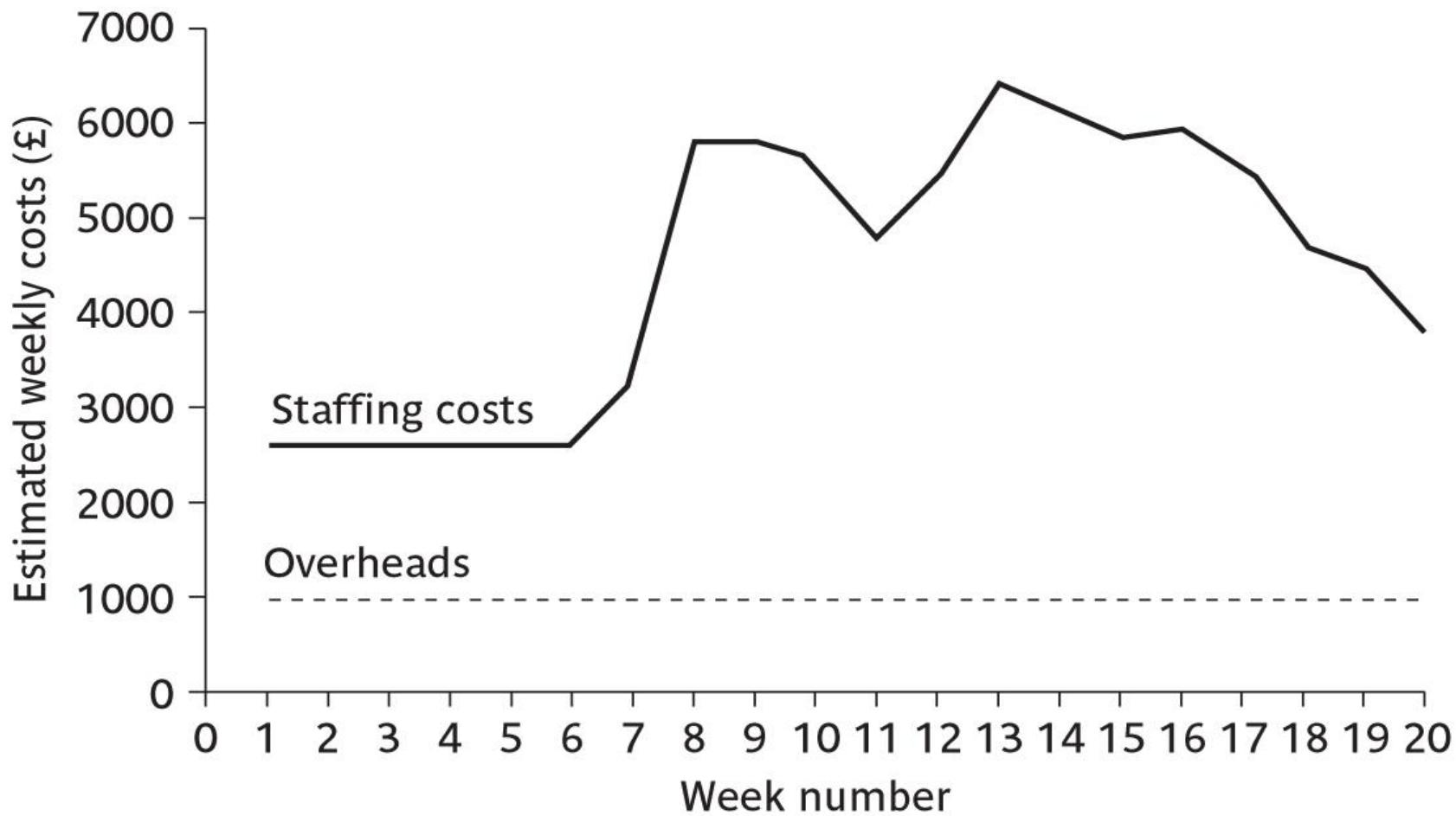
## Cost Schedules

- It is now time to produce a detailed cost schedule showing weekly or monthly costs over the life of the project. This will provide a more detailed and accurate estimate of costs and will serve as a plan against which project progress can be monitored.
- Calculating cost is straightforward where the organization has standard cost figures for staff and other resources. Where this is not the case, then the project manager will have to calculate the costs.

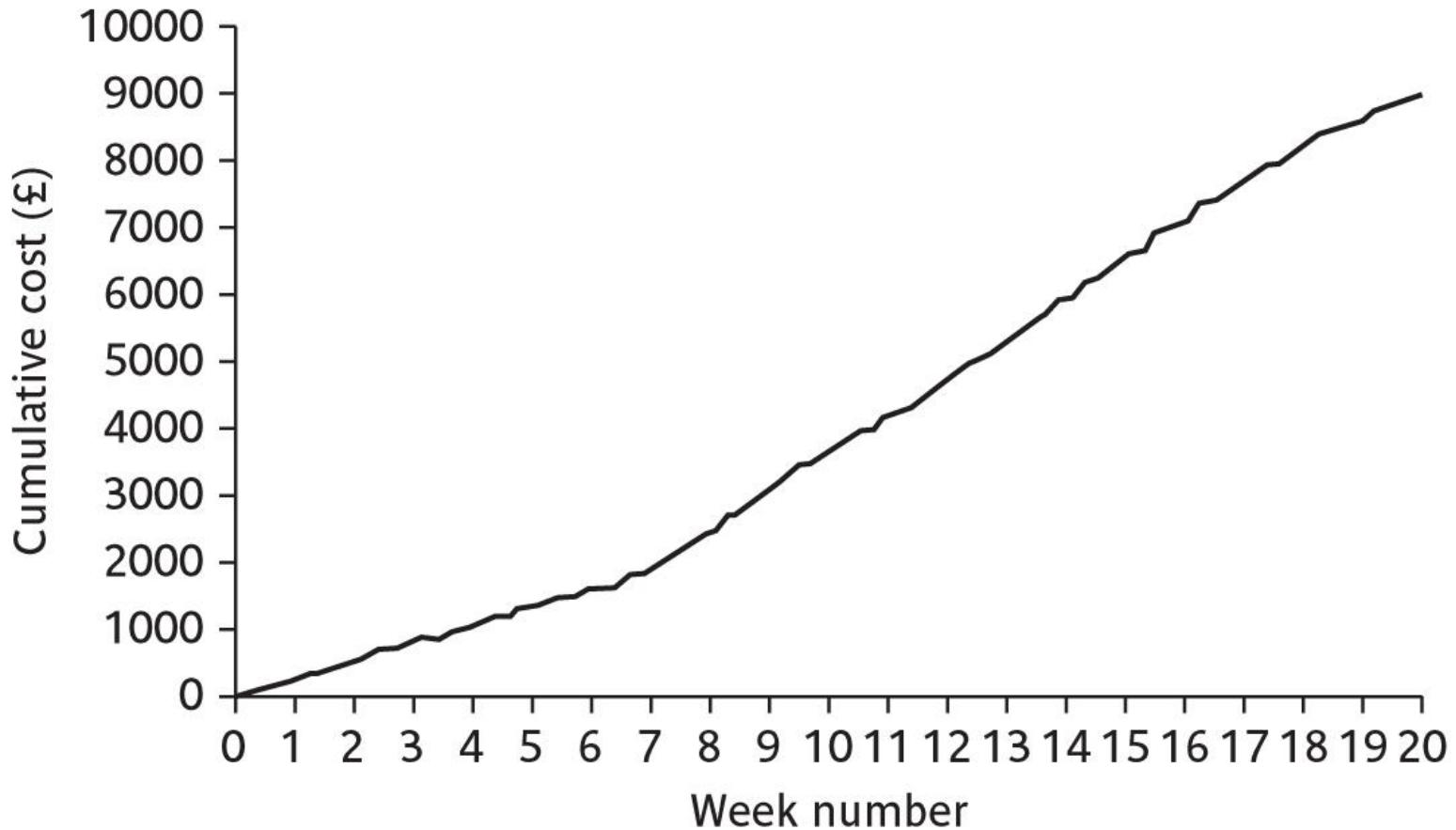
## In general, costs are categorized as.....

- ❖ Staff costs - These will include staff salaries as well as the other direct costs of employment such as the employer's contribution to social security funds, pension scheme contributions, holiday pay and sickness benefit.
- ❖ Overheads - Overheads represent expenditure that an organization incurs, which cannot be directly related to individual projects or jobs, including space rental, interest charges and the costs of service departments (such as human resources).
- ❖ Usage charges - In some organizations, projects are charged directly for use of resources such as computer time (rather than their cost being recovered as an overhead). This will normally be on an 'as used' basis.

## Cost profile



## Accumulative costs



## Scheduling Sequence

- Going from an ideal activity plan to a cost schedule can be represented as a sequence of steps.
- Activity plan and risk assessment would provide the basis for our resource allocation and schedule from which we would produce cost schedules.
- In practice, successful resource allocation often necessitates revisions to the activity plan, which, in turn, will affect our risk assessment.
- Similarly, the cost schedule might indicate the need or desirability to reallocate resources or revise activity plans – particularly where that schedule indicates a higher overall project cost than originally anticipated.

# END

# **SPM 11.0**

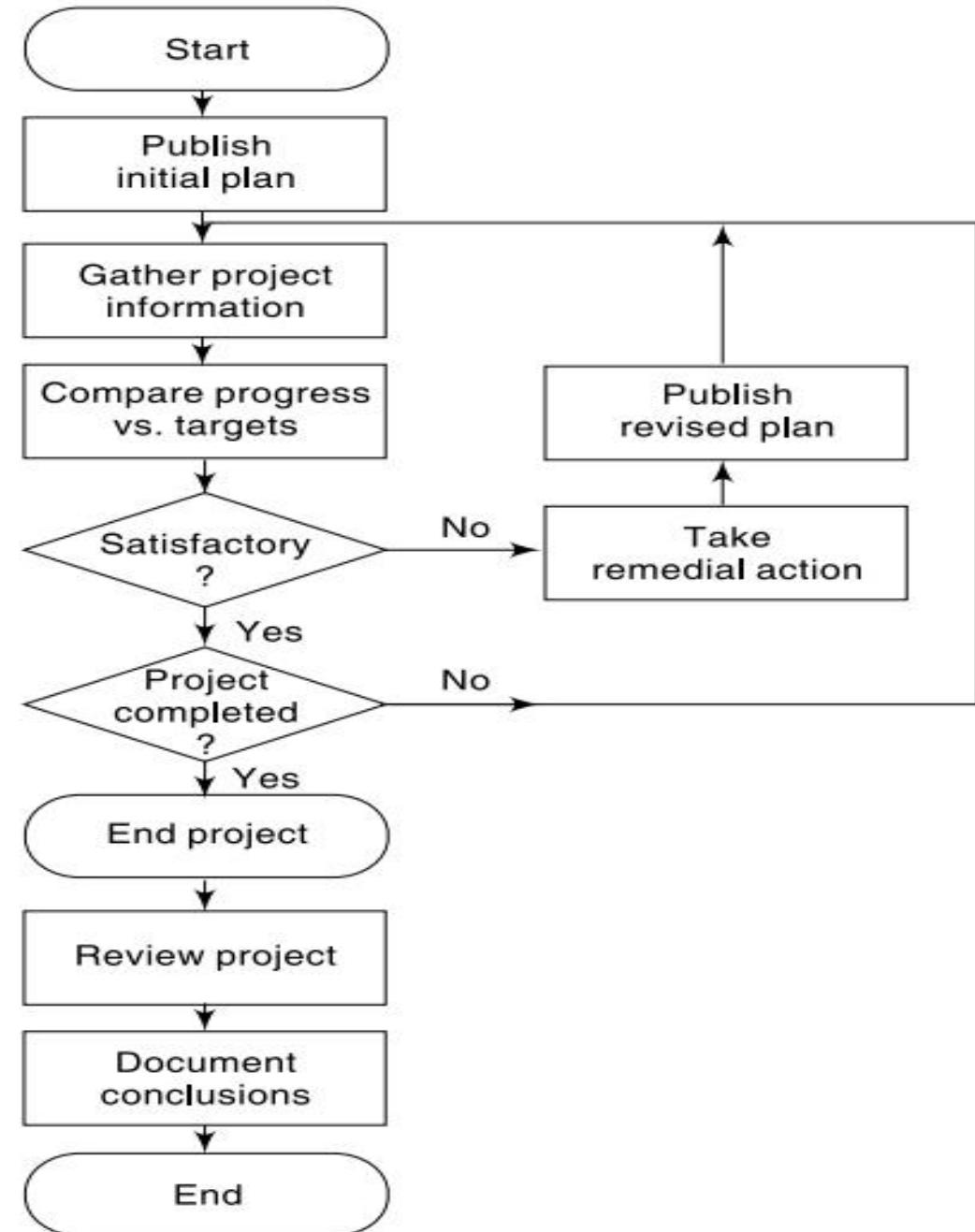
# **Monitoring and Control in Software Project Management**

*Once work schedules have been published and the project is started, attention must be focused on progress.*

*This requires monitoring of what is happening, comparison of actual achievement against the schedule and where necessary, revision of plans and schedules to bring the project as far as possible back on target.*

# The Project control cycle

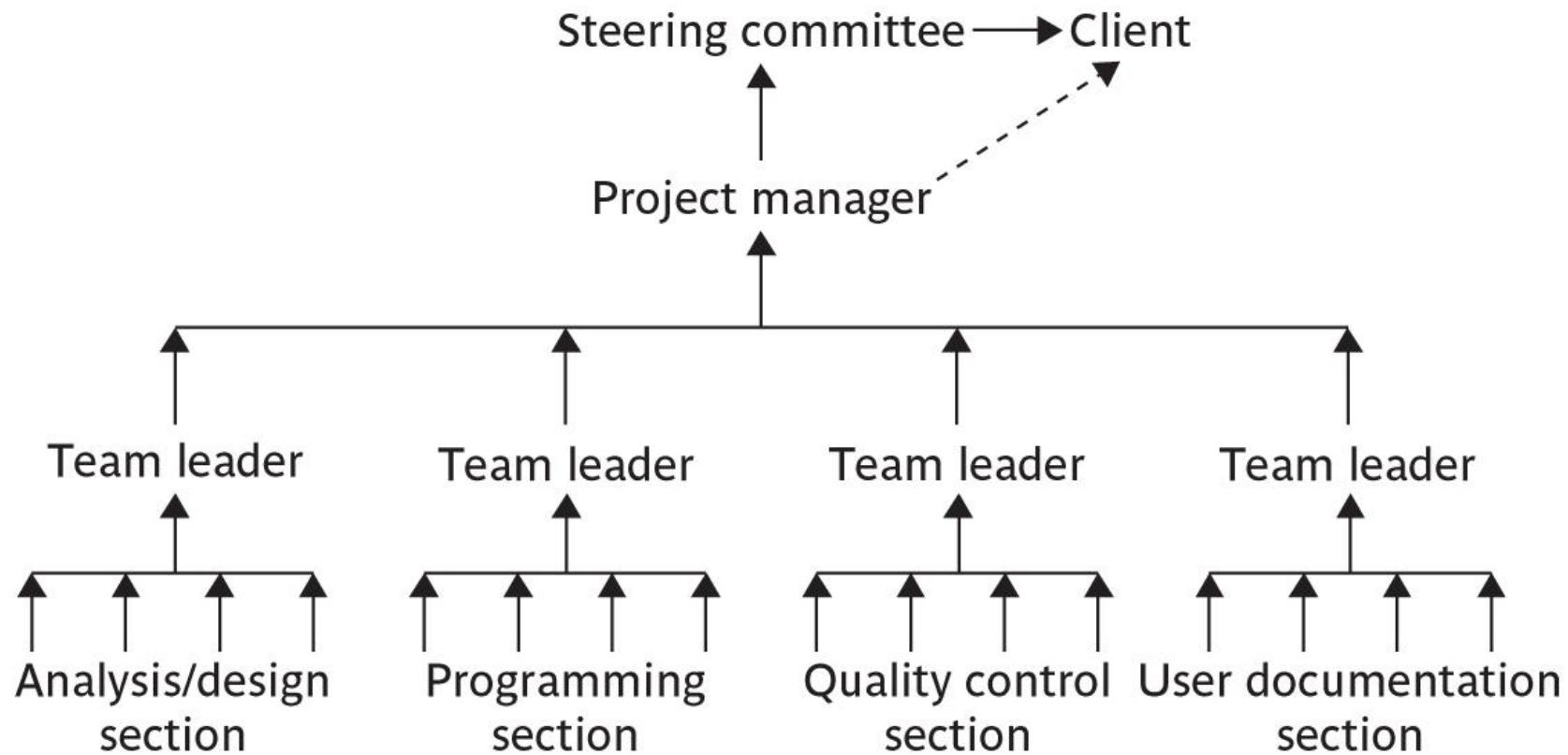
- Exercising control over a project and ensuring that targets are met is a matter of regular monitoring – finding out what is happening and comparing it with targets. There may be a mismatch between the planned outcomes.
- Figure illustrates a model of the project control cycle and shows how, once the initial project plan has been published, project control is a continual process of monitoring progress against that plan and, where necessary, revising the plan to take account of deviations.



# Project reporting structures

- ❖ The overall responsibility for ensuring satisfactory progress on a project is often the role of the project steering committee, project management board or Project Board.
- ❖ Day-to-day responsibility will rest with the project manager and, in all but the smallest of projects, aspects of this can be delegated to team leaders.

Figure illustrates the typical reporting structure found with medium and large projects. In most cases team leaders will collate reports on their section's progress and forward summaries to the project manager. These, in turn, will be incorporated into project-level reports for the steering committee and, via them or directly, progress reports for the client.



## Categories of reporting

Report type	Examples	Comment
Oral formal regular	Weekly or monthly progress meetings	While reports may be oral, formal written minutes should be kept
Oral formal ad hoc	End-of-stage review meetings	While largely oral, likely to receive and generate written reports
Written formal regular	Job sheets, progress reports	Normally weekly using forms
Written formal ad hoc	Exception reports, change reports	
Oral informal ad hoc	Canteen discussion, social interaction	Often provides early warning; must be backed up by formal reporting

## Assessing progress

Progress assessment will normally be made on the basis of information collected and collated at regular intervals or when specific events occur.

**Checkpoints:** Predetermined times when progress of a project is checked.

- Event driven: Check takes place when a particular event has been achieved
- Time driven: Date of the check is pre-determined

*Frequency of reporting depend upon the size and degree of risk of the project. Team leaders, for example, may want to assess progress daily (particularly when employing inexperienced staff) whereas project managers may find weekly or monthly reporting appropriate. In general, the higher the level, the less frequent and less detailed the reporting needs to be*

# Collecting progress details

Need to collect data about:

- ❖ Achievements
- ❖ Costs

A big problem: how to deal with *partial completions (99% completion syndrome)*

Possible solutions:

- Control of products, not activities
- Subdivide into lots of sub-activities

As a rule, managers will try to break down long activities into more controllable tasks of one or two weeks' duration. However, it will still be necessary to gather information about partially completed activities and, in particular, forecasts of how much work is left to be completed. It can be difficult to make such forecasts accurately.

Where there is a series of products, partial completion of activities is easier to estimate. Counting the number of record specifications or screen layouts produced, for example, can provide a reasonable measure of progress. In some cases, intermediate products can be used as in-activity milestones.

- Many organizations use standard accounting systems with weekly timesheets to charge staff time to individual jobs. The staff time booked to a project indicates the work carried out and the charges to the project. It does not, however, tell the project manager what has been produced or whether tasks are on schedule.

Staff	John Smith		Week ending	30/3/07		
<b>Rechargeable hours</b>						
Project	Activity code	Description	Hours this week	% complete	Scheduled completion	Estimated completion
P21	A243	Code mod A3	12	30	24/4/07	24/4/07
P34	B771	Document take-on	20	90	6/4/07	4/4/07

## Red/amber/green (RAG) reporting

- identify the key (first level) elements for assessment in a piece of work;
- Break these key elements into constituent elements (second level);
- Assess each of the second-level elements on the scale
  - ❖ **Green for 'on target',**
  - ❖ **Amber for 'not on target but recoverable',**
  - ❖ **Red for 'not on target and recoverable only with difficulty';**
- Review all the second-level assessments to arrive at first-level assessments
- Review first- and second-level assessments to produce an overall assessment.

Activity Assessment Sheet

Staff Justin

Ref: IoE/P/13

Activity: Code and test module C

Week number	13	14	15	16	17	18
Activity summary	G	A	A	R		

Component Comments

Screen handling procedures	G	A	A	G	
File update procedures	G	G	R	A	
Housekeeping procedures	G	G	G	A	
Compilation	G	G	G	R	
Test data runs	G	G	G	A	
Program documentation	G	G	A	R	

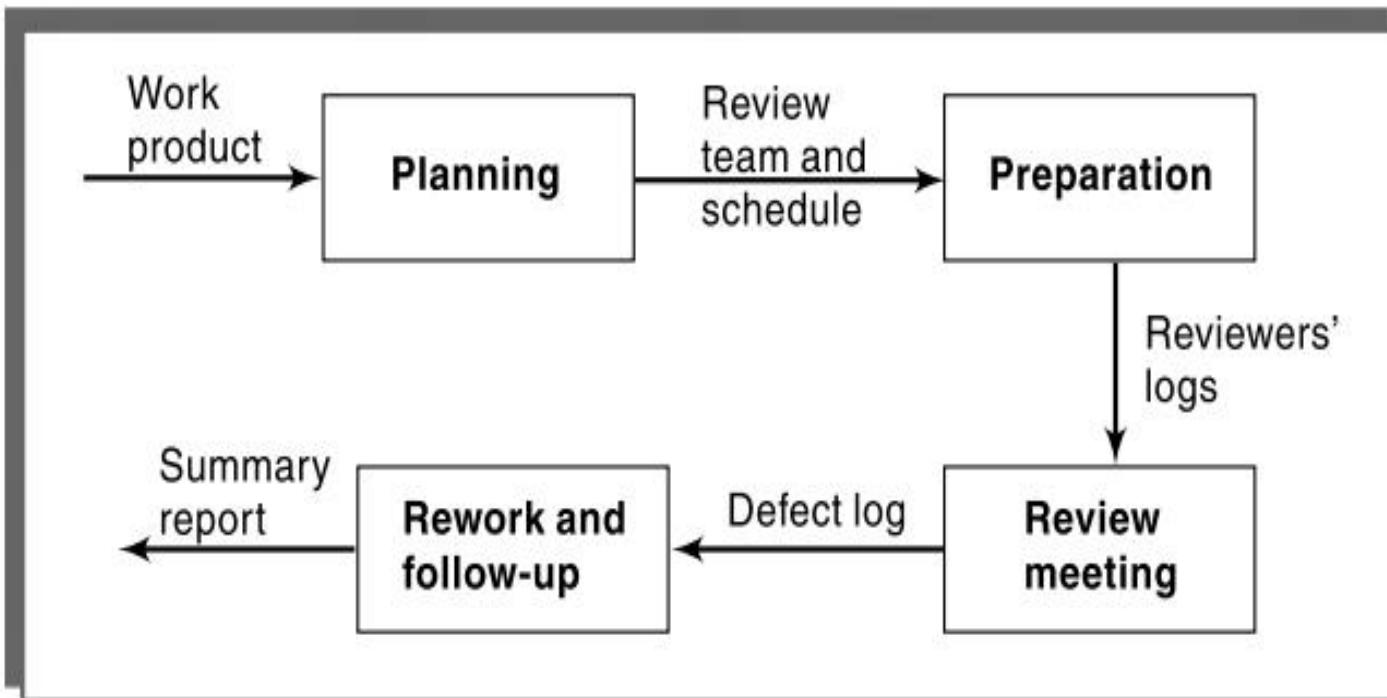
# Review

- Review of work products is an important mechanism for monitoring the progress of a project and ensuring the quality of the work products.
- Testing is an effective defect removal mechanism.
  - However, testing is applicable to only executable code.
  - Review is applicable to all work products.
- Review usually helps to identify any deviation from standards.
- Reviewers suggest ways to improve the work product
- A review meeting often provides learning opportunities to not only the author of a work product, but also the other participants of the review meeting.
- The review participants gain a good understanding of the work product under review, making it easier for them to interface or use the work product in their work.

## Review roles

- Role of the moderator include scheduling and convening meetings, distributing review materials, leading and moderating the review sessions, ensuring that the defects are tracked to closure.
- Role of the recorder is to record the defects found, the time, and effort data.
- The review team members review the work product and give specific suggestions to the author about the existing defects and also point out ways to improve the work product.

# Review process



**Planning:** The project manager nominates a moderator. Moderator selects rest of the team. Team can have the author of the preceding work product, member who would use the work, Peers of the author, authors of the work products interfaces.

**Preparation:** Moderator convenes a brief preparation meeting, author presents a brief overview of the work product, reviewers individually carry out review and record in review logs.

**Review Meeting:** Reviewers give their comments based on the logs they have prepared. Recorder scribes all the defects and points that the author agrees.

**Rework:** The author addresses all the issues raised by the reviewers and prepares a rejoinder. a final summary report of the review is prepared.

# Project Termination Review

- Project termination reviews provide important opportunities to learn from past mistakes as well as successes.
- Project termination need not necessarily mean project failure or premature abandonment. A project may be terminated on successful completion

## Reasons for Project Termination

- Project is completed successfully handed over to the customer.
- Incomplete requirements
- Lack of resources
- Some key technologies used in the project have become obsolete during project execution
- Economics of the project has changed, for example because many competing product may have become available in the market.

## Project Termination Process

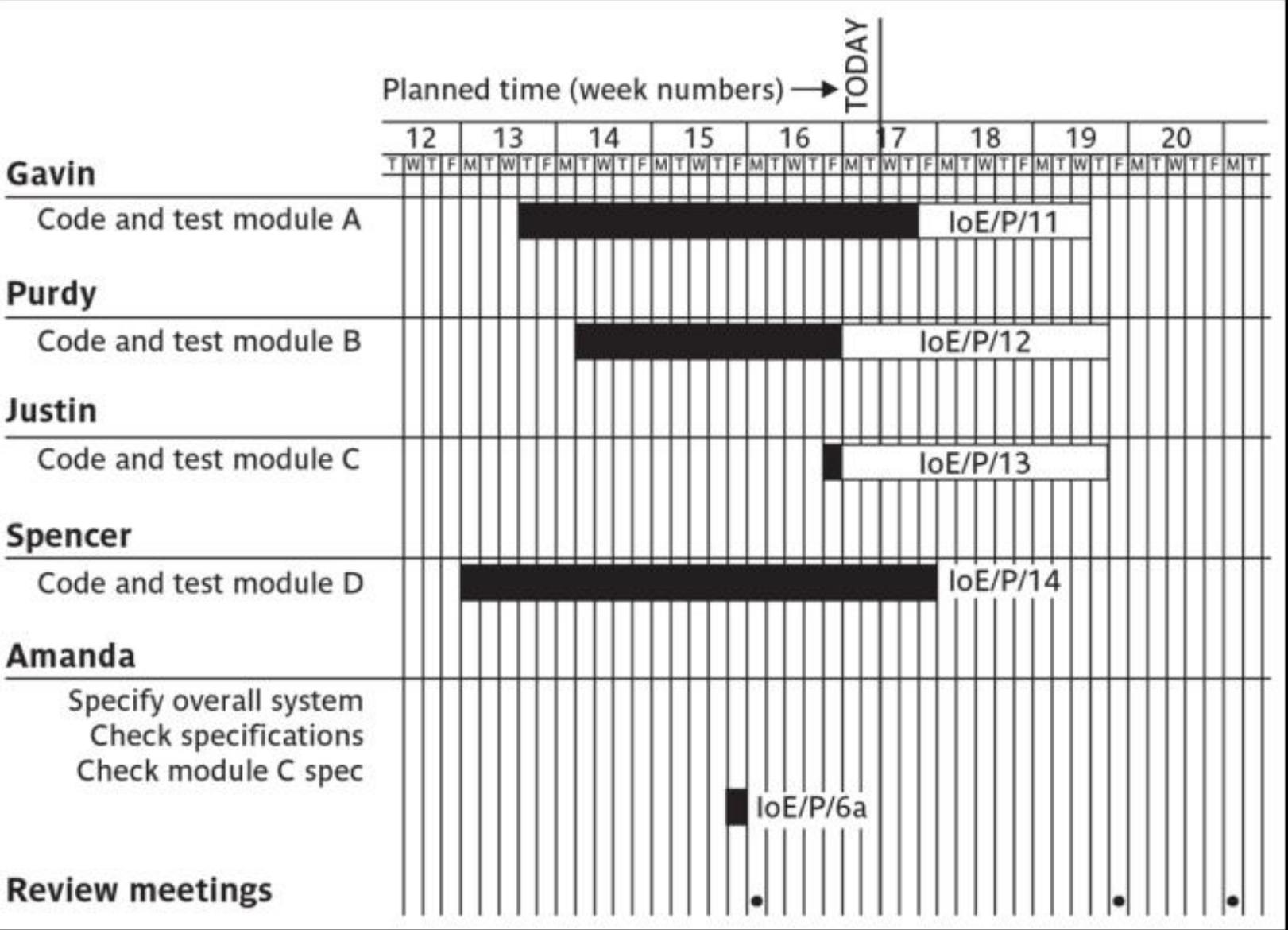
- ❖ Project survey
- ❖ Collection of objective information
- ❖ Debriefing meeting
- ❖ Final project review
- ❖ Result publication

## Visualization Progress:::

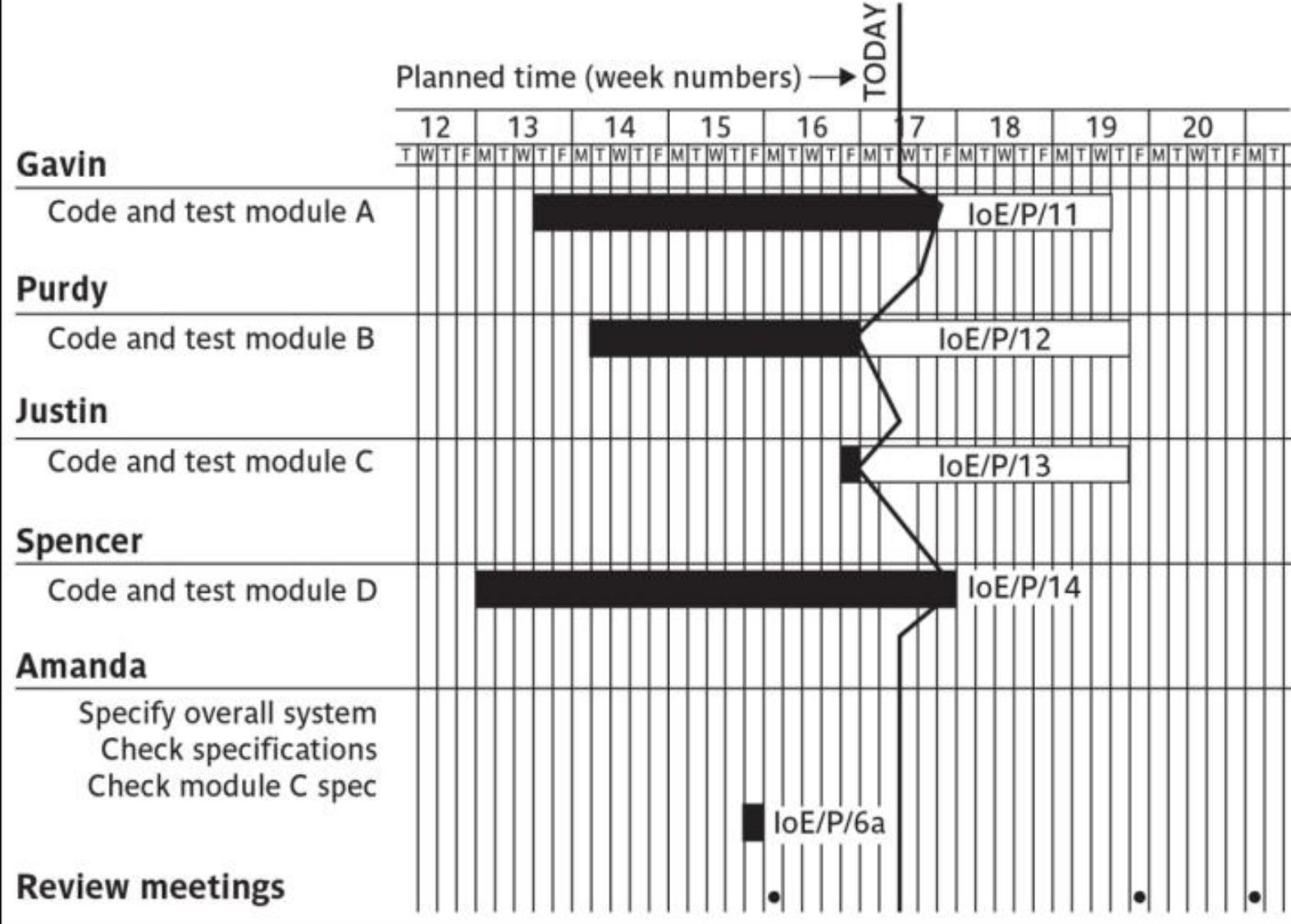
Having collected data about project progress, a manager needs some way of presenting that data to greatest effect.

Some of these methods (such as Gantt charts) provide a static picture, a single snap-shot, whereas others (such as time-line charts) try to show how the project has progressed and changed through time.

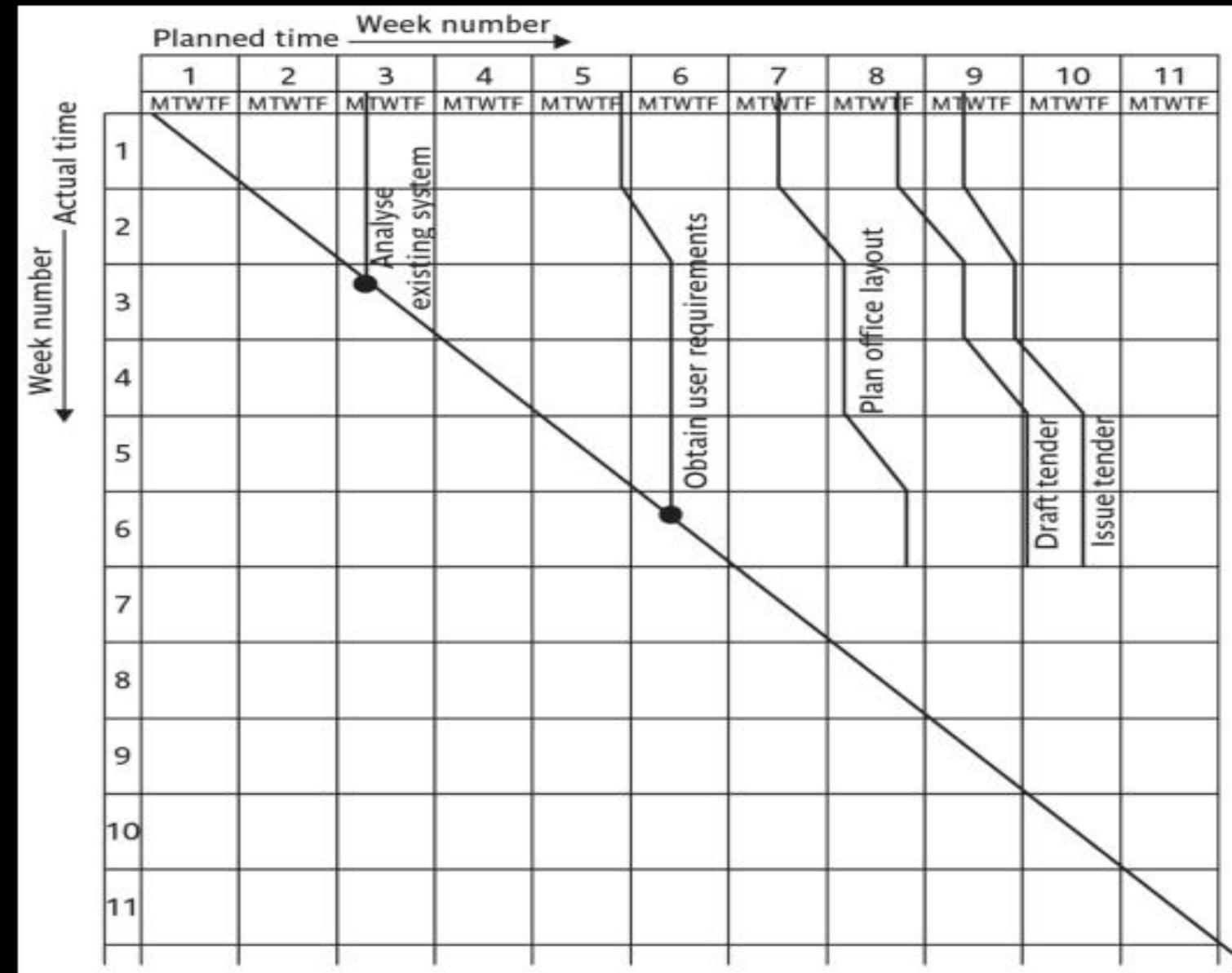
# Gantt charts



# Slip charts



## The timeline



## Cost monitoring

- A project could be late because the staff originally committed have not been deployed
- In this case the project will be *behind time* but *under budget*
- A project could be on time but only because additional resources have been added and so be *over budget*
- Need to monitor both achievements and costs

## ***Earned value analysis***

- *Planned value (PV)* or *Budgeted cost of work scheduled (BCWS)* – original estimate of the effort/cost to complete a task (compare with idea of a ‘price’)
- *Earned value (EV)* or *Budgeted cost of work performed (BCWP)* – total of PVs for the work completed at this time

# Earned value – an example

- Tasks
  - ◆ Specify module      5 days
  - ◆ Code module        8 days
  - ◆ Test module        6 days
- At the beginning of day 20, PV = 19 days
- If everything but testing completed EV = 13 days
- Schedule variance = EV-PV i.e.  $13-19 = -6$
- Schedule performance indicator (SPI) =  $13/19 = 0.68$
- SV negative or SPI <1.00, project behind schedule

## **Earned value analysis – actual cost**

- **Actual cost (AC)** is also known as **Actual cost of work performed (ACWP)**
- In previous example, if
  - ↳ ‘Specify module’ actually took 3 days
  - ↳ ‘Code module’ actually took 4 days
- **Actual cost = 7 days**
- **Cost variance (CV) = EV-AC i.e. 13-7 = 6 days**
- **Cost performance indicator = 13/7 = 1.86**
- **Positive CV or CPI > 1.00 means project within budget**

## **Earned value analysis – actual costs**

- CPI can be used to produce new cost estimate
- Budget at completion (BAC) – current budget allocated to total costs of project
- Estimate at completion (EAC) – updated estimate =  $BAC/CPI$ 
  - ↳ e.g. say budget at completion is £19,000 and CPI is 1.86
  - ↳  $EAC = BAC/CPI = £10,215$  (projected costs reduced because work being completed in less time)

Where tasks have been started but are not yet complete, some consistent method of assigning an earned value must be applied. Common methods in software projects are:

- **the 0/100 technique** Where a task is assigned a value of zero until such time that it is completed when it is given a value of 100% of the budgeted value;
- **the 50/50 technique** Where a task is assigned a value of 50% of its value as soon as it is started and then given a value of 100% once it is complete;
- **the milestone technique** Where a task is given a value based on the achievement of milestones that have been assigned values as part of the original budget plan.

## Time variance

- **Time variance (TV) – difference between time when specified EV should have been reached and time it actually was**
- **For example say an EV of £19000 was supposed to have been reached on 1<sup>st</sup> April and it was actually reached on 1<sup>st</sup> July then TV = - 3 months**

## Prioritizing monitoring

We might focus more on monitoring certain types of activity e.g.

- Critical path activities
- Activities with no free float – if delayed later dependent activities are delayed
- Activities with less than a specified float
- High risk activities
- Activities using critical resources

## **Getting back on track: options**

- **Renegotiate the deadline – if not possible then**
- **Try to shorten critical path e.g.**
  - ◆ **Work overtime**
  - ◆ **Re-allocate staff from less pressing work**
  - ◆ **Buy in more staff**
- **Reconsider activity dependencies**
  - ◆ **Over-lap the activities so that the start of one activity does not have to wait for completion of another**
  - ◆ **Split activities**

# Exception planning

- Some changes could affect
  - ↳ **Users**
  - ↳ **The business case (e.g. costs increase reducing the potential profits of delivered software product)**
- These changes could be to
  - ↳ **Delivery date**
  - ↳ **Scope**
  - ↳ **Cost**
- In these cases an exception report is needed

- **First stage**
  - ↳ **Write an exception report for sponsors (perhaps through project board)**
    - **Explaining problems**
    - **Setting out options for resolution**
- **Second stage**
  - ↳ **Sponsor selects an option ( or identifies another option)**
  - ↳ **Project manager produces an exception plan implementing selected option**
  - ↳ **Exception plan is reviewed and accepted/rejected by sponsors/Project Board**

## Change control

- ❖ Identifying items that need to be subject to change control
- ❖ Management of a central repository of the master copies of software and documentation
- ❖ Administering change procedures
- ❖ Maintenance of access records

## Typical change control process

- Users perceive the need for a change. User management decide that the change is valid and worthwhile and pass it to development management.
- A developer is assigned to assess the practicality and cost of making the change.
- Development management report back to user management on the cost of the change; user management decide whether to go ahead. One or more developers are authorized to make copies of components to be modified.
- Copies modified. After initial testing, a test version might be released to users for acceptance testing
- When users are satisfied then operational release authorized – master configuration items updated

# Software Configuration Management (SCM)

- ❖ SCM is concerned with tracking and controlling changes to a software.
- ❖ Development and maintenance environment:
  - Various work products associated with the software continually change.
  - Unless a proper configuration management system is deployed, several problems can appear.

## Why Use SCM?

- Problems associated with concurrent access
- Undoing Changes
- System accounting
- Handling variants
- Accurate determination project status
- Preventing unauthorized access to the work products

# END

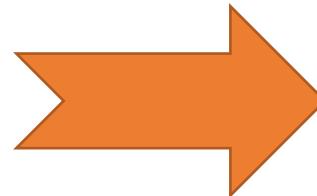
# **SPM 12.0**

# Contract Management

- A contract is any agreement between two or more parties where one party agrees to provide certain deliveries or services, and the other party agrees to pay for those deliveries or services.
- Contract management is a management that mainly focuses on management of contract between two or many parties and to ensure that all parties meet their respective objectives more effectively and efficiently.
- It involves various processes such as managing contracts, deliverables, guidelines, deadlines, execution, analysis

## Types Of Contracts

- Fixed price contracts
- Time and materials contracts
- Fixed price per delivered unit



## Fixed price contracts:::

Here a price is fixed when the contract is signed. The customer knows that, if there are no changes in the contract terms, this is the price they pay on completion.

## Time and materials contracts::::

Here, the customer is charged at a fixed rate per unit of effort, for example per staff-hour.

The supplier may provide an initial estimate of the cost based on their current understanding of the customer's requirements.

The supplier usually invoices the customer for work done at regular intervals, say each month.

## Fixed price per delivered unit::::

This is often associated with function point (FP) counting. The size of the system to be delivered is calculated or estimated at the outset of the project.

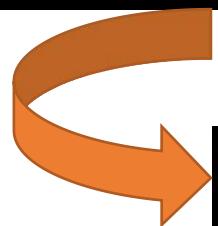
The size could be estimated in lines of code, but FPs can be more easily derived from requirements documents. A price per unit is also quoted. The final price is then the unit price multiplied by the number of units.

## A schedule of charges per function point

Function point count	Function design cost per FP	Implementation cost per FP	Total cost per FP
Up to 2,000	\$242	\$725	\$967
2,001–2,500	\$255	\$764	\$1,019
2,501–3,000	\$265	\$793	\$1,058
3,001–3,500	\$274	\$820	\$1,094
3,501–4,000	\$284	\$850	\$1,134

**A system shown above is to be implemented contains 2600 FPs. Determine the overall charge of contract.**

<i>FP count</i>	<i>Design cost/FP</i>	<i>implementation cost/FP</i>	<i>total cost/FP</i>
to 2,000	\$242	\$725	\$967
2,001- 2,500	\$255	\$764	\$1,019
2,501- 3,000	\$265	\$793	\$1,058
3,001- 3,500	\$274	\$820	\$1,094
3,501- 4,000	\$284	\$850	\$1,134



Given the Estimated system size is 2,600 FPs

$$\begin{aligned}
 \text{Price} &= 2000 \text{ FPs} \times \$967 + 500 \text{ FPs} \times \$1,019 \\
 &+ 100 \text{ FPs} \times \$1,058 = \$2,549,300
 \end{aligned}$$

**Contract may be placed to buy a completed software application. It can be::::**

**A contract for a completed software package may belong to the following category:::::**

- ❖ A bespoke system created specifically for one customer.
- ❖ An off-the-shelf package bought ‘as is’ – this is sometimes referred to as shrink wrapped software.
- ❖ A customized off-the-shelf (COTS) software – where a core system is modified to meet the needs of the client.

Another way of categorizing contracts, at least initially, is according to the approach that is used in contractor selection, namely

- *open*
- *restricted*
- *negotiated*

## The tendering process

### ✓ Open tendering

- Any supplier can bid to supply the goods and services. All bids compliant with the original conditions in the invitation to tender must be considered and evaluated in the same way.
- With a major project this evaluation process can be time consuming and expensive

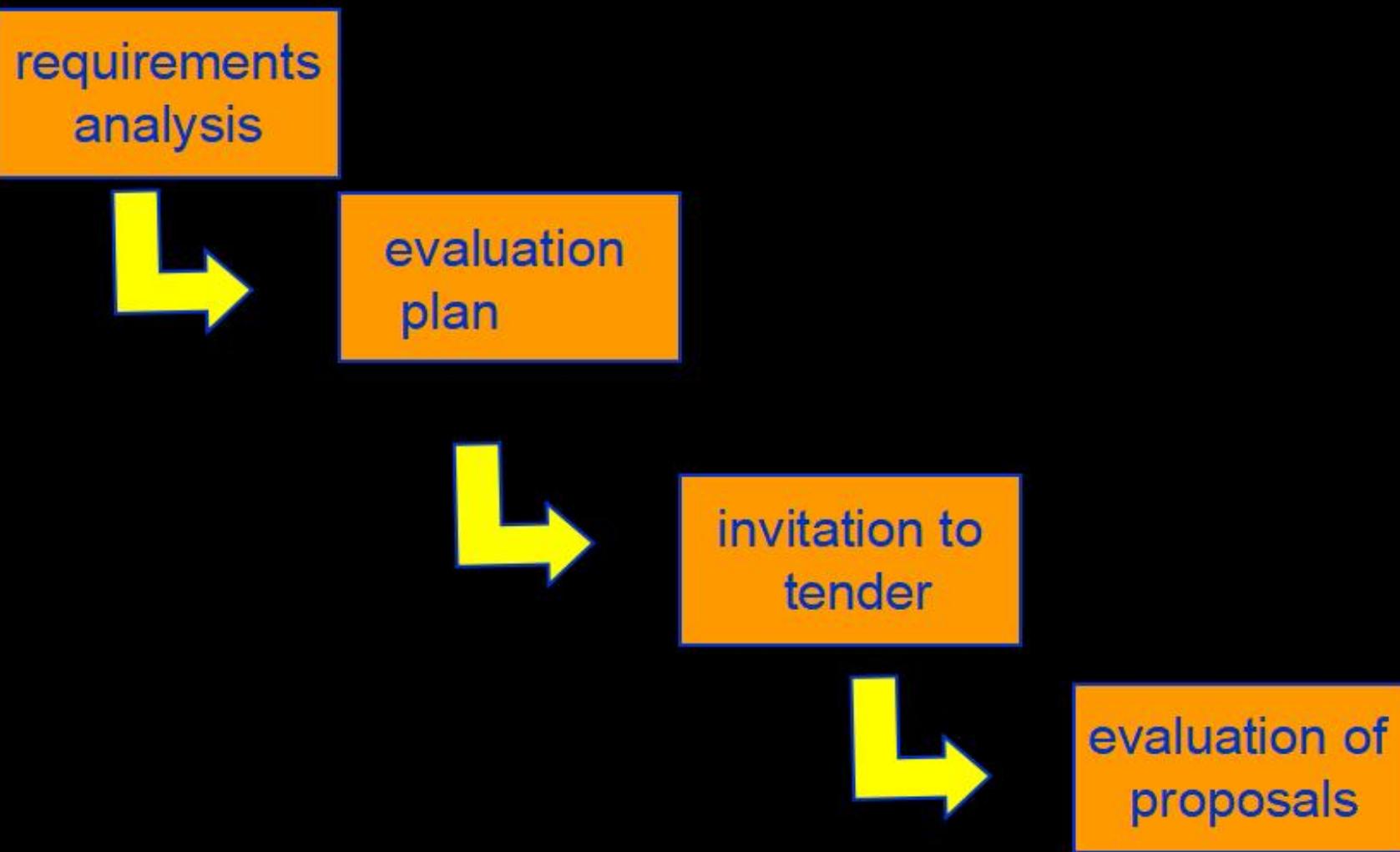
### ✓ Restricted tendering process

- There are bids only from suppliers who have been invited by the customer. Unlike the open tendering process, the customer may at any point reduce the number of potential suppliers being considered.
- This is usually the best approach to adopt.

### ✓ Negotiated procedure

- Negotiate with one supplier e.g. for extensions to software already supplied

# Stages in contract placement



# Requirements

These will include

- ↑ functions in software, with necessary inputs and outputs
- ↑ standards to be adhered to
- ↑ other applications with which software is to be compatible
- ↑ quality requirements e.g. response times

*Main sections in a requirements document*

---

*Section name*

---

- 1 Introduction
  - 2 A description of any existing systems and the current environment
  - 3 The customer's future strategy or plans
  - 4 System requirements
    - mandatory
    - desirable
  - 5 Deadlines
  - 6 Additional information required from potential suppliers
- 

## Evaluation plan

How are proposals to be evaluated?

Methods could include:

- ↑ reading proposals
- ↑ interviews
- ↑ demonstrations
- ↑ site visits
- ↑ practical tests

## Invitation to tender (ITT)

Note that bidder is making an *offer* in response to  
ITT

*acceptance* of offer creates a *contract*

Customer may need further information

Problem of different technical solutions to the  
same problem

## Memoranda of agreement (MoA)

- Customer asks for technical proposals
- Technical proposals are examined and discussed
- Agreed technical solution in MoA
- Tenders are then requested from suppliers based in MoA
- Tenders judged on price
- Fee could be paid for technical proposals by customer

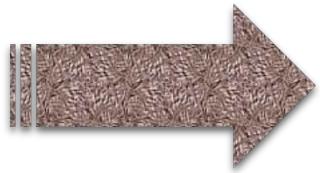
# Contracts

- A project manager cannot be expected to be a legal expert – needs advice
- BUT must ensure contract reflect true requirements and expectations of supplier and client

## Contract checklist

- Definitions – what words mean precisely e.g. ‘supplier’, ‘user’, ‘application’
- Form of agreement. For example, is this a contract for a sale or a lease, or a license to use a software application? Can the license be transferred?
- Goods and services to be supplied – this could include lengthy specifications
- Timetable of activities
- Payment arrangements – payments may be tied to completion of specific tasks
- Ownership of software
  - Can client sell software to others?
  - Can supplier sell software to others? Could specify that customer has ‘exclusive use’
  - Does supplier retain the copyright?
- Environment – for example, where equipment is to be installed, who is responsible for various aspects of site preparation e.g. electricity supply?
- Customer commitments – for example providing access, supplying information
- Standards to be met

## Contract management



**Some terms of contract will relate to management of contract, for example,**

- Progress reporting
- Decision points – could be linked to release of payments to the contractor
- Variations to the contract, i.e. how are changes to requirements dealt with?
- Acceptance criteria

**Contracts should include agreement about how customer/supplier relationship is to be managed e.g.**

- decision points - could be linked to payment
- quality reviews
- changes to requirements

# END

# **SPM 13.0**

# **Software Product Quality**

*Software quality product is defined in term of its fitness of purpose. That is, a quality product does precisely what the users want it to do.*

*Consider a functionally correct software product. That is, it performs all tasks as specified in the SRS document. But, has an almost unusable user interface. Even though it may be functionally right, we cannot consider it to be a quality product.*

## Importance of Software Quality:::

- ❖ **Increasing criticality of software** - The final customer or user is naturally anxious about the general quality of software, especially its reliability. This is increasingly the case as organizations become more dependent on their computer systems and software is used more and more in areas that are safety-critical.
- ❖ **The intangibility of software** - This makes it difficult to know that a particular task in a project has been completed satisfactorily. The results of these tasks can be made tangible by demanding that the developer produce 'deliverables' that can be examined for quality.
- ❖ **Accumulating errors during software development** - As computer system development is made up of a number of steps where the output from one step is the input to the next, the errors in the earlier deliverables will be added to those in the later steps leading to an accumulating detrimental effect.

## Software Quality Specifications:::

A quality specification with the following minimum details are as follows:

- **Definition/description:** definition of the quality characteristic
- **Scale:** the unit of measurement
- **Test:** the practical test of the extent to which the attribute quality exists
- **Minimally acceptable:** the worst value which might be acceptable if other characteristics compensated for it, and below which the product would have to be rejected out of hand
- **Target range:** the range of values within which it is planned the quality measurement value should lie
- **Now:** the value that applies currently

In the case of reliability, this might be measured in terms of:

- **Availability:** the percentage of a particular time interval that a system is usable;
- **Mean time between failures:** the total service time divided by the number of failures;
- **Failure on demand:** the probability that a system will not be available at the time required or the probability that a transaction will fail;
- **Support activity:** the number of fault reports that are generated and processed.

## ISO 9126

- The ISO 9126 standard was first introduced in 1991 to tackle the question of the definition of software quality.
- ISO 9126 identifies six major external software quality characteristics:
  - ***functionality***, which covers the functions that a software product provides to satisfy user needs;
  - ***reliability***, which relates to the capability of the software to maintain its level of performance;
  - ***usability***, which relates to the effort needed to use the software;
  - ***efficiency***, which relates to the physical resources used when the software is executed;
  - ***maintainability***, which relates to the effort needed to make changes to the software;
  - ***portability***, which relates to the ability of the software to be transferred to a different environment.

- ISO 9126 provides guidelines for the use of the quality characteristics.
- Once the requirements for the software product have been established, the following steps are suggested:
  - Judge the importance of each quality characteristic for the application
  - Select the external quality measurements within the ISO 9126 framework relevant to the qualities prioritized above
  - Map measurements onto ratings that reflect user satisfaction
  - Identify the relevant internal measurements and the intermediate products in which they appear
  - Overall assessment of product quality

Mapping measurements to user satisfaction

Response time (seconds)	Rating
<2	Exceeds expectation
2–5	Within the target range
6–10	Minimally acceptable
>10	Unacceptable

Mapping response times onto user satisfaction

Response time (seconds)	Quality score
<2	5
2–3	4
4–5	3
6–7	2
8–9	1
>9	0

### Weighted quality scores

Product quality	Importance rating (a)	Product A		Product B	
		Quality score (b)	Weighted score (a × b)	Quality score (c)	Weighted score (a × c)
Usability	3	1	3	3	9
Efficiency	4	2	8	2	8
Maintainability	2	3	6	1	2
Overall			17		19

## Product and Process Metrics

- ❖ Product metrics help measure the characteristics of a product being developed. A few examples of product metrics and the specific product characteristics that they measure are the following: the LOC and function point metrics are used to measure size, the PM (person-month) metric is used to measure the effort required to develop a product, and the time required to develop the product is measured in months.
- ❖ Process metrics help measure how a development process is performing. Examples of process metrics are review effectiveness, average number of defects found per hour of inspection, average defect correction time, productivity, average number of failures detected during testing per LOC, and the number of latent defects per line of code in the developed product.

*Product metrics help measure the characteristics of a product being developed, whereas process metrics help measure how a process is performing*

# Process Quality Management

Errors should therefore be eradicated by careful examination of the deliverables of each step before they are passed on. One way of doing this is by having the following process requirements for each step.

- **Entry requirements**, which have to be in place before an activity can start. An example would be that a comprehensive set of test data and expected results be prepared and approved before program testing can commence.
- **Implementation requirements**, which define how the process is to be conducted. In the testing phase, for example, it could be laid down that whenever an error is found and corrected, all test runs must be repeated, even those that have previously been found to run correctly.
- **Exit requirements**, which have to be fulfilled before an activity is deemed to have been completed. For example, for the testing phase to be recognized as being completed, all tests will have to have been run successfully with no outstanding errors

# Quality Management Systems

- ISO 9000 describes the fundamental features of a quality management system (QMS) and its terminology.
- ISO 9001 describes how a QMS can be applied to the creation of products and the provision of services.
- ISO 9004 applies to process improvement.

Principles are:

- ❖ understanding the requirements of customers so that they can be met, or even exceeded;
- ❖ leadership to provide the unity of purpose and direction needed to achieve quality objectives;
- ❖ involvement of staff at all levels;
- ❖ a focus on the individual processes which create intermediate or deliverable products and services;
- ❖ a focus on the systems of interrelated processes that create delivered products and services;
- ❖ continuous improvement of processes;
- ❖ decision making based on factual evidence;
- ❖ building mutually beneficial relationships with suppliers.

## Process Capability Models

- As compared to the product metrics, the process metrics are more meaningfully measured during product development. Consequently, to manage quality during development, process-based techniques are very important.
- SEI CMM, CMMI, ISO 15504, and Six Sigma, which are popular process capability models.

## **SEI capability maturity model (SEI CMM)**

- CMM is a reference model for apprising the software process maturity into different levels. This can be used to predict the most likely outcome to be expected from the next project that the organisation undertakes.
- It must be remembered that SEI CMM can be used in two ways— capability evaluation and software process assessment.
- Capability evaluation provides a way to assess the software process capability of an organisation. On the other hand, software process assessment is used by an organisation with the objective to improve its own process capability.
- SEI CMM classifies software development industries into the following five maturity levels.

<i>CMM Level</i>	<i>Focus</i>	<i>Key Process Areas (KPAs)</i>
Initial	Competent people	
Repeatable	Project management	Software project planning Software configuration management
Defined	Definition of processes	Process definition Training program Peer reviews
Managed	Product and process quality	Quantitative process metrics Software quality management
Optimising	Continuous process improvement	Defect prevention Process change management Technology change management

## Comparison Between ISO 9000 Certification and CMM

- ISO 9000 is awarded by an international standards body. Therefore, ISO 9000 certification can be quoted by an organisation in official documents, communication with external parties, and in tender quotations. However, SEI CMM assessment is purely for internal use.
- SEI CMM was developed specifically for software industry and therefore addresses many issues which are specific to software industry alone.
- SEI CMM goes beyond quality assurance and prepares an organisation to ultimately achieve TQM. In fact, ISO 9001 aims at level 3 of SEI CMM model.
- SEI CMM model provides a list of key process areas (KPAs) on which an organisation at any maturity level needs to concentrate to take it from one maturity level to the next. Thus, it provides a way for achieving gradual quality improvement. In contrast, an organisation adopting ISO 9000 either qualifies for it or does not qualify.

# ISO 15504 Process assessment

- Like CMMI the standard is designed to provide guidance on the assessment of software development processes.
- Processes are assessed on the basis of nine process attributes:::::

Level	Attribute	Comments			
0. Incomplete		The process is not implemented or is unsuccessful			
1. Performed process	1.1 Process performance	The process produces its defined outcomes	4. Predictable process	4.1 Process measurement	Quantitatively measurable targets are set for each sub-process and data collected to monitor performance
2. Managed process	2.1 Performance management	The process is properly planned and monitored		4.2 Process control	On the basis of the data collected by 4.1 corrective action is taken if there is unacceptable variation from the targets
	2.2 Work product management	Work products are properly defined and reviewed to ensure they meet requirements	5. Optimizing	5.1 Process innovation	As a result of the data collected by 4.1, opportunities for improving processes are identified
3. Established process	3.1 Process definition	The processes to be carried out are carefully defined		5.2 Process optimization	The opportunities for process improvement are properly evaluated and where appropriate are effectively implemented
	3.2 Process deployment	The processes defined above are properly executed by properly trained staff			

## Techniques to improve quality -Inspections

- When a piece of work is completed, copies are distributed to co-workers time is spent individually going through the work noting defects
- A meeting is held where the work is then discussed
- A list of defects requiring re-work is produced

### Inspections - advantages of approach

- An effective way of removing superficial errors from a piece of software
- Motivates the software developer to produce better structured and self-descriptive code
- Spreads good programming practice
- Enhances team-spirit
- *The main problem* maintaining the commitment of participants

## 'Clean-room' software development

### Three separate teams:

- ✓ *Specification team – documents user requirements and usage profiles (how much use each function will have)*
- ✓ *Development team – develops code but does not test it. Uses mathematical verification techniques*
- ✓ *Certification team – tests code. Statistical model used to decide when to stop*

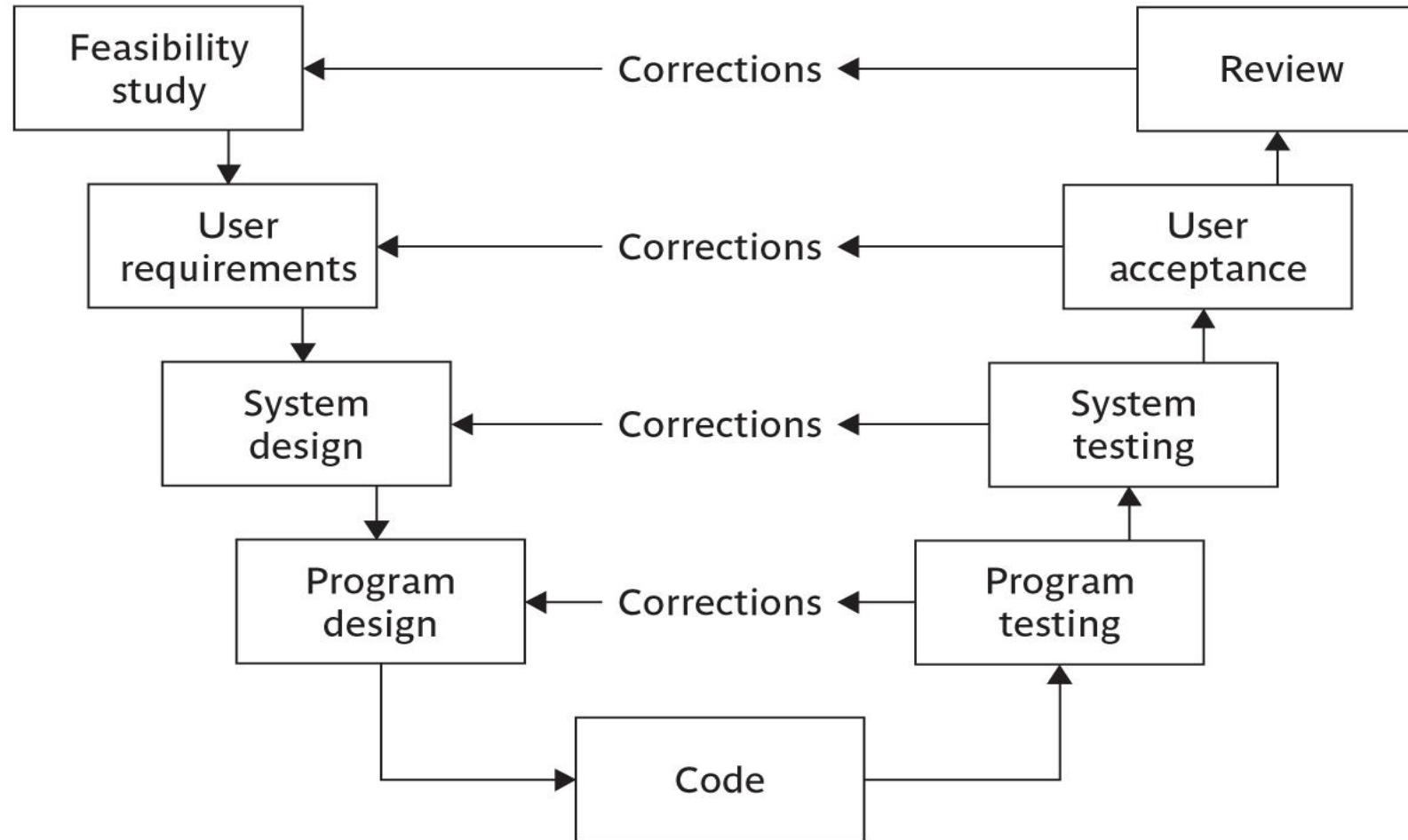
## Verification versus Validation

- ❑ Verification is the process of determining whether the output of one phase of software development conforms to that of its previous phase; whereas validation is the process of determining whether a fully developed software conforms to its requirements specification.
  
- ❑ Verification is carried out during the development process to check if the development activities are being carried out correctly, whereas validation is carried out towards the end of the development process to check if the right product as required by the customer has been developed.

## Testing: the V-process model

It is an extension of the waterfall approach. For each development stage there is a testing stage.

The testing associated with different stages serves different purposes e.g. system testing tests that components work together correctly, user acceptance testing that users can use system to carry out their work



## **Black box versus glass box test**

### **□ Glass box testing**

The tester is aware of the internal structure of the code; can test each path; can assess percentage test coverage of the tests e.g. proportion of code that has been executed

### **□ Black box testing**

The tester is not aware of internal structure; concerned with degree to which it meets user requirements

## Levels of testing

- Unit testing
- Integration testing
- System testing

## Testing activities

- *Test planning*
- *Test suite design*
- *Test case execution and result checking*
- *Test reporting:*
- *Debugging:*
- *Error correction:*
- *Defect retesting*
- *Test closure:*

## Test plans

- ❖ Specify test environment

In many cases, especially with software that controls equipment, a special test system will need to be set up
- ❖ Usage profile

failures in operational system more likely in the more heavily used components  
Faults in less used parts can lie hidden for a long time  
Testing heavily used components more thoroughly tends to reduce number of operational failures

# Management of testing

- The tester executes test cases and may as a result find discrepancies between actual results and expected results – issues
- Issue resolution – could be:
  - a mistake by tester
  - a fault – needs correction
  - a fault – may decide not to correct: off-specification
  - a change – software works as specified, but specification wrong: submit to change control

# Software reliability

- The reliability of a software product essentially denotes its *trustworthiness* or *dependability*.
- Reliability of a software product usually keeps on improving with time during the testing and operational phases as defects are identified and repaired.
- A reliability growth model (RGM) models how the reliability of a software product improves as failures are reported and bugs are corrected. An RGM can be used to determine when during the testing phase a given reliability level will be attained, so that testing can be stopped

# Quality plans

- quality standards and procedures should be documented in an organization's *quality manual*
- for each separate project, the quality needs should be assessed
- select the level of quality assurance needed for the project and document in a *quality plan*

## Components of a Quality plan::::

- ✓ scope of plan
- ✓ references to other documents
- ✓ quality management, including organization, tasks, and responsibilities
- ✓ documentation to be produced
- ✓ standards, practices and conventions
- ✓ reviews and audits

# END

# **SPM 14.0**

# Working in Teams

- This section will look enhancing communication between individual developers within teams and across teams. It will also look at how the efforts of individuals and teams can be coordinated through communication.
- By ‘teams’ we usually mean groups of people who are working together. Typically the individuals work in the same office, that is, are co-located. However, the term ‘project team’ is sometimes used to refer to all the people working on a project.

The collaborative nature of project work will have an influence on nearly all stages of the Step Wise project planning framework::::

- Identify project scope and objectives. Here stakeholders in the project are identified and communications channels are established.
- Identify project infrastructure. The organization structure within which the project team will exist is identified.
- Analyse project characteristics. Decisions made about how the project is to be executed – for example buying versus building software functionality – will affect the team structure needed.
- Estimate effort for each activity. Individual and group experience will have a key influence on developer productivity.
- Identify activity risks. Risks will include those that relate to staff such as continued availability.
- Allocate resources.
- Review/publicize plan. A communication plan could be produced at this point

## Becoming a Team

**It is suggested that teams go through five basic stages of development:**

- Forming** The members of the group get to know each other and try to set up some ground rules about behaviour.
- Storming** Conflicts arise as various members of the group try to exert leadership and the group's methods of operation are being established.
- Norming** Conflicts are largely settled and a feeling of group identity emerges.
- Performing** The emphasis is now on the tasks at hand.
- Adjourning** The group disbands.

**Belbin came to the conclusion that teams needed a balance of different types of people.**

- **The chair:** not necessarily brilliant leaders but they must be good at running meetings, being calm, strong but tolerant.
- **The plant:** someone who is essentially very good at generating ideas and potential solutions to problems.
- **The monitor-evaluator:** good at evaluating ideas and potential solutions and helping to select the best one.
- **The shaper:** rather a worrier, who helps to direct the team's attention to the important issues.
- **The team worker:** skilled at creating a good working environment, for example, by 'jollying people along'.
- **The resource investigator:** adept at finding resources in terms of both physical resources and information.
- **The completer-finisher:** concerned with completing tasks.
- **The company worker:** a good team player who is willing to undertake less attractive tasks if they are needed for team success

## Group performance

One way of categorizing group tasks is into:

- **Additive tasks;** Additive tasks mean that the efforts of each participant are added to get the final result,
- **Compensatory tasks;** With compensatory tasks the judgements of individual group members are pooled so that the errors of some are compensated for by the inputs from others.
- **Disjunctive tasks;** With disjunctive tasks there is only one correct answer. The effectiveness of the group depends on someone coming up with the right answer or the others recognizing it as being correct.
- **Conjunctive tasks;** Conjunctive tasks are where progress is governed by the rate of the slowest performer. Software production where different staff are responsible for different modules is a good example of this.

# Group decision making

Groups deal less effectively with poorly structured problems needing creative solutions. Brainstorming techniques can help groups in this situation but research shows that people often come up with more ideas individually than in a group.

## Obstacles to good group decision making::::

- It is time consuming; it can stir up conflicts within the group; and decisions can be unduly influenced by dominant personalities.

## Measures to reduce the disadvantages of group decision making::::

- the cooperation of a number of experts is enlisted;
- the problem is presented to the experts;
- the experts record their recommendations;
- these recommendations are collated and reproduced;
- the collected responses are recirculated;
- the experts comment on the ideas of others and modify their recommendations if so moved;
- if the leader detects a consensus then the process is stopped, otherwise the comments are recirculated to the experts.

## Organization and Team Structures

- Large software development companies are usually organized into departments based on several criteria such as staff specialization, product lines, categories of customers, or geographical location.
- Small companies do not have high-level departmentalization. Therefore, we can view a small company as having only a single department. Every department usually handles several projects at any time, and each project is assigned to a separate team of developers.
- The effectiveness of the developers in achieving the project objectives is significantly affected by how a department is organized into teams and how the individual teams are structured. In this context, two important issues that are critical to the effective functioning of every organization are:::::
  - Department Structure: How is a department organized into teams?*
  - Team Structure: How are project teams structured?*

## Functional formats

- ❖ In the functional format, the developers are divided into functional groups based on their specialization or experience. In other words, each functional group comprises developers having expertise in some specific task or functional area.
- ❖ For example, the different functional groups of an organization might specialize in areas such as database, networking, requirements analysis, design, testing, and so on.
- ❖ Every developer in an organization would belong to some functional group depending on his/her specialization. respective functional groups.
- ❖ A functional team working on a project does not physically meet the members of other functional teams who have carried out other parts of the project.
- ❖ A functional organization mandates production of good quality documentation.

## Project formats

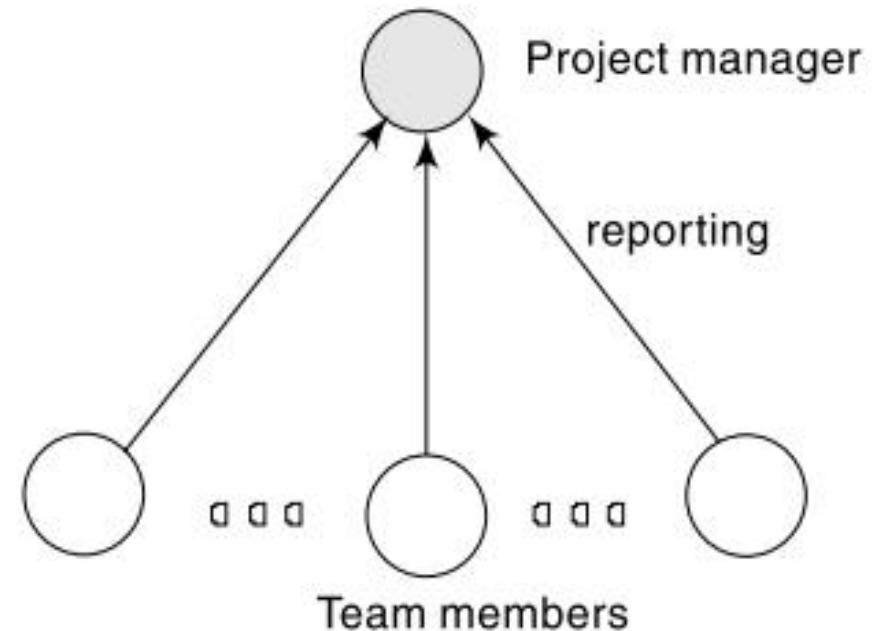
- ❖ The project format is designed for realizing task-oriented teams. In the project format, at the start of every project, a set of developers are assigned to it.
- ❖ It is assumed that, among them the assigned members can carry out various activities required for project completion. The developers remain with the project till the completion of the project. Thus, the same team carries out all the project activities.
- ❖ This is in contrast to a functional organization, where each developer belongs to a functional group and for completing a project activity, members of the corresponding functional area are assigned to the project temporarily, who are returned back to their respective functional area after completion of the activity.

## Matrix formats

- ❖ Matrix format is an extension of a functional format, and is intended to provide the advantages of both the functional and project structures. In a matrix organization, the pool of functional specialists is assigned to different projects as needed.
- ❖ Thus, in a matrix organization, the project manager needs to share the project responsibility with a number of individual functional managers.
- ❖ In a strong functional matrix, the functional managers have authority to assign workers to projects and project managers have to accept the assigned personnel. In a weak matrix, the project manager completely controls the project budget, can reject workers from functional groups, and can even decide to hire outside workers.

## Chief Programmer team

- ❖ In this team structure, a senior member provides the technical leadership and is designated as the chief programmer.
- ❖ The chief programmer defines the specification and constructs the high-level design; and then partitions the remaining tasks of detailed design, viz., coding, testing, documentation, etc., into many smaller tasks; and assigns them to the team members.
- ❖ He/she also verifies and integrates the work completed by different team members.
- ❖ A chief programmer team is more efficient than a democratic team for completing simple and small projects since the chief programmer can quickly work out a satisfactory design and assign work to the team members to code and test different modules of his design solution.

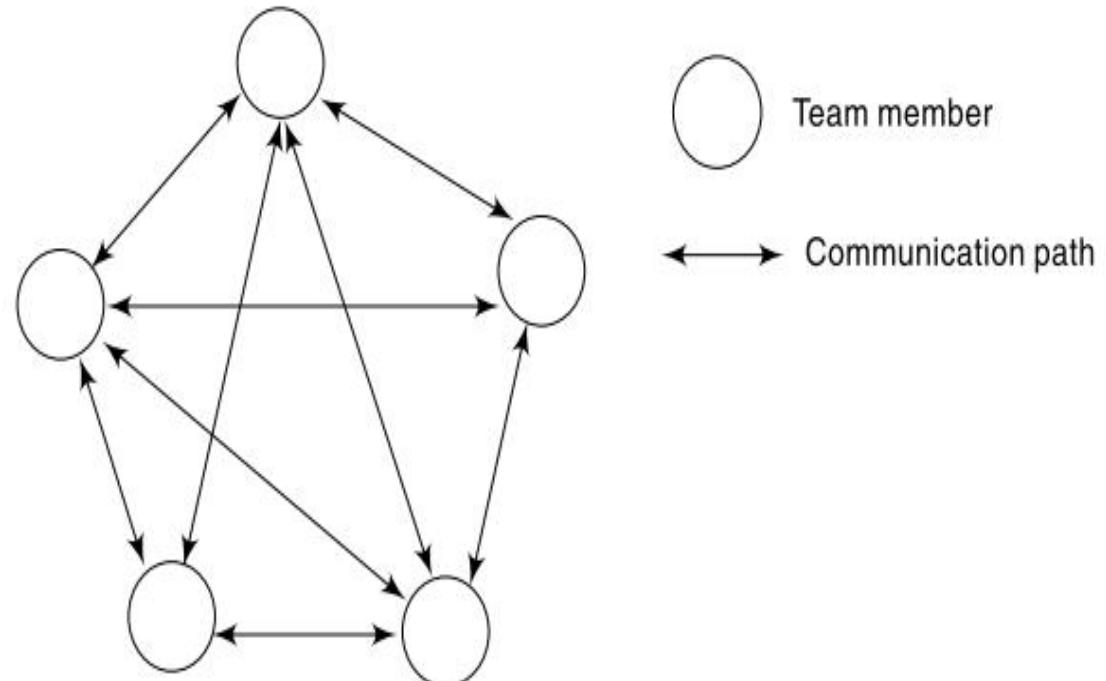


## Disadvantages of the chief programmer team

- The chief programmer is provided with an authority to assign work to the team members and to monitor their work. This however leads to lower team morale, as the team members work under the constant supervision of the chief programmer.
- The chief programmer team structure inhibits collective and original thinking by the team members and the chief programmer typically takes all important decisions by himself/herself.
- The chief programmer team is subject to single point failure since too much responsibility and authority is assigned to the chief programmer. That is, a project might get severely affected if the chief programmer either leaves the organization or becomes unavailable for some other reasons.
- A major problem with the chief programmer structure is getting hold of a really outstanding programmer for the role of the chief programmer. Since the chief programmer carries out many tasks individually, there is a danger of information overload on the chief programmer.

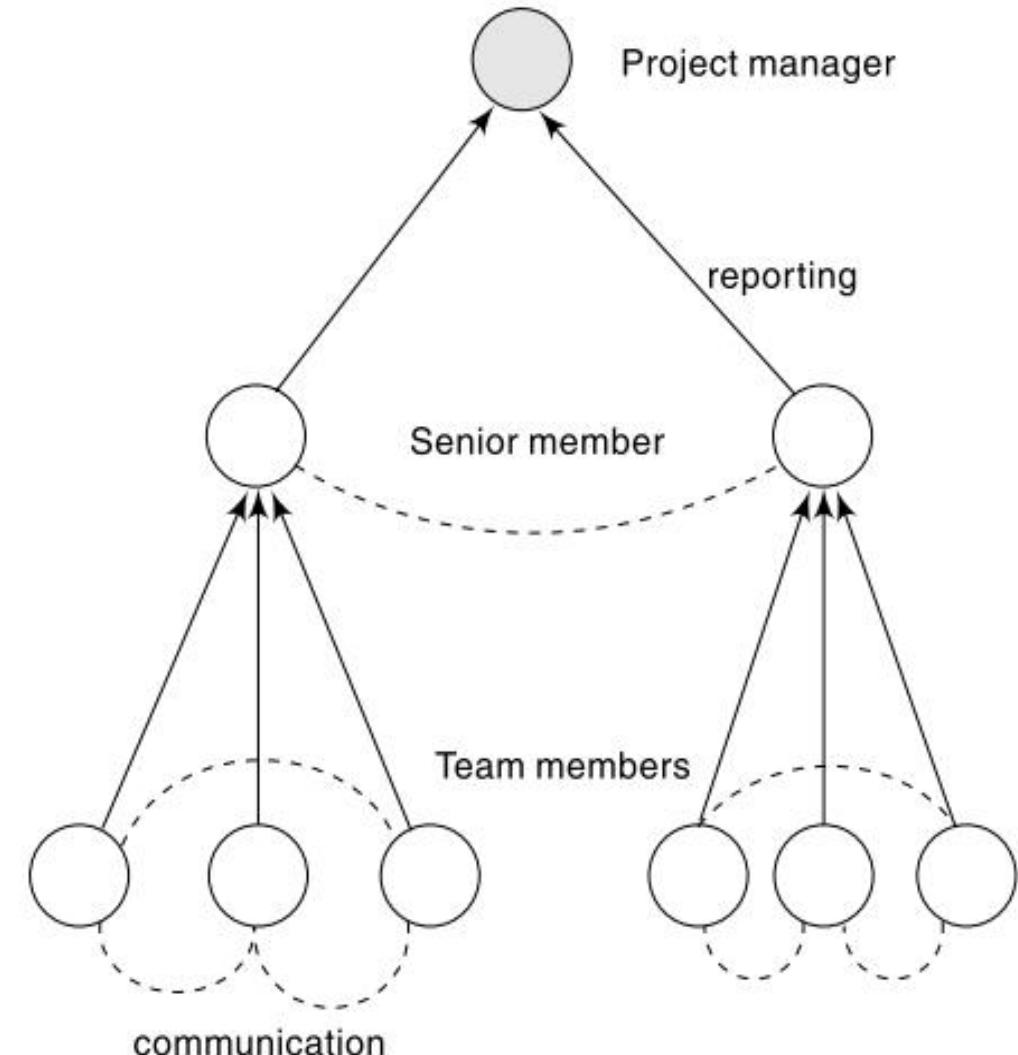
## Democratic team

- ❖ The democratic team structure, as the name implies, does not enforce any formal team hierarchy.
- ❖ Decisions are taken based on discussions, where any member is free to discuss with any other member.
- ❖ Typically, a manager provides the administrative leadership. At different times, different members of the team provide technical leadership.
- ❖ Since a lot of debate and discussions among the team members takes place, for large team sizes significant overhead is incurred on this count.



## Mixed control team structure

- ❖ The mixed team structure, as the name implies, draws ideas from both the democratic and chief-programmer team structures.
- ❖ Here the communication paths are shown as dashed lines and the reporting structure is shown using solid arrows. This team structure incorporates both hierarchical reporting and democratic set up.
- ❖ The democratic arrangement at the level of senior developers is used to decompose the problem into small parts. Democratic setup at the programmer level facilitates working out an effective solution to a single part. This team structure is extremely popular and is being used in many software development companies.
- ❖ The mixed control team organization is suitable for large team sizes.



# Coordination Dependencies

- Researchers and innovators in the area of computer supported cooperative work (CSCW) have been interested in identifying the types of coordination where computer tools could be of assistance.
  - A coordination theory has been developed which provides a useful classification of coordination dependencies that are likely to exist in any substantial organizational undertaking.
- ❖ Shared resources - Several projects need the services of particular types of scarce technical experts for certain parts of the project.
  - ❖ Producer–customer ('right time') relationships - A business analyst may need to produce an agreed requirements document before the development of software components can begin.
  - ❖ Task–subtask dependencies - In order to complete a task a sequence of subtasks have to be carried out.
  - ❖ Accessibility ('right place') dependencies - This type of dependency is of more relevance to activities that require movement over a large geographical area.
  - ❖ Usability ('right thing') dependencies - It relates to the general question of fitness for purpose, which includes the satisfaction of business requirements.
  - ❖ Fit requirements - This is ensuring that different system components work together effectively. Integration testing is one mechanism that ensures that these requirements are met.

# END