CHAPTER 1

Software Project Management Concepts

SOFTWARE:

Software encompasses programs that execute within a computer of any size and architecture, content that is presented as the computer programs executed and documents that encompasses all form of electronic media.

-Pressman

Computer software in general is used to define a collection of computer programs, procedures and documentation that performs certain tasks on computer system.

-Princeton University

Software is the instruction that when executed provide different features, functions and performance. It is the data structure that enables the program to adequately manipulate information.

Types of Software

1) **System software**: It helps to run the computer hardware and the computer system. Includes operating system, servers, device drivers etc.
2) **Programming software**: It provides tools to assist a programmer in writing programs and the software codes using different languages in easy way. Eg: text editors, compilers, debuggers etc.
3) **Application software**: It allows end users to accomplish one or more specific tasks. Eg: databases, computer games, medical softwares.

Even though there are different perspectives software can be divided into two ways:

1) **Generic products**: Produced by a development organization and sold to open market to any customer able to buy them.

2) **Bespoke product**: They are commissionable to particular customer, developed by software contractor.

SOFTWARE PROJECT MANAGEMENT (SPM)

It is an effective management of software development project is one of the vital aspects leading to the overall success. The main objective of SPM is to steers the software engineers towards the successful completion of the software project.

The requirements of SPM are:

- Managing the people , process *and* problem *during the project work*
- Software development team to produce reliable estimates of the effort cost and project duration
- Proper work distribution among technical and non-technical staffs
- Adequate risk assessment
- Software Quality Assurance (SQA) and Software Configuration Management (SCM) Formal Technical Review
- Decision making capabilities
- Good communication skills (Interpersonal , verbal and written)

SOFTWARE CRISIS

It is described as the impact of rapid increase in computer power and the complexity of the problems which could be better tackled. It refers to the difficulty of writing correct, understandable and verified programs.

It causes software complexity, expectations and change. The term software crisis was coined by F.L.Bauer at first NATO SE conference in 1968 at Germany.

“Silver Bullet” in the term used to solve the crisis but it doesn’t exist as crisis can’t be solved within time. Crisis occurred due to sudden introduction of third generation.

SOFTWARE MYTHS:

Myths appeared to be reasonable facts having initiative feeling and are often declared by experienced practitioners who knows the score but they are not always and at all true.

It's a belief about software and the process used to build the software in the earlier days. Myths were categorized in two aspects

- Management Myths
- Customer Myths
- Practitioners Myths

1) **Management Myths:**

Myth:

We already have a book that's full of standards and procedures for building software. Won't that provide my people with everything they need to know?

Reality:

The book of standards may very well exist, but is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice? Is it complete? Is it adaptable?

Myth:

If we get behind schedule, we can add more programmers and catch up.

Reality:

Software development is not mechanistic process like manufacturing. According to Brooks; "Adding people to late software project makes it later"

Myth:

If I decide to outsource the software project to third party, I can just relax and let that firm build it.

Reality:

If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsource software projects.

2) **Customer Myths**

Myth:

A general statement of objectives is sufficient to begin writing the programs; we can fill the details later.

Reality:

Although a comprehensive and stable statement of requirements is not always possible, an ambiguous statement of objectives is a recipe for disaster. Precise requirements are developed only through effective and continuous communication between customer and developer.

3) **Practitioners Myths**

Myth:

Once we write the program and get it to work, our work is done.

Reality:

Industry data indicate that between 60 to 80 percent of all effort expanded on software will be expanded after it is delivered to the customer for the first time.

Myth:

The only deliverable work product for a successful project is the working program.

Reality:

A working program is only one part of a software configuration that includes many elements. Documentation provides a foundation for successful engineering and, more importantly, guidance for software supports.

SOFTWARE PROCESS

When you work to build a products or system, it's important to go through a series of predictable steps-a road map that helps to create a timely high-quality result. Hence the road map that you follow is called a software process.

Software engineers and their managers adapt the process to their needs and then follow it.

The people who have requested the software have a role to play in the process of defining, building, and testing it.

Fundamental activities common to all software processes are

- Software specifications
- Software design and implementation
- Software validation
- Software evolution

SOFTWARE PROCESS MODELS

A software process model is an abstract representation of a software process. Process models defines a distinct set of activities, actions, tasks, milestones and work products that are required to engineer high quality software.

It provides stability, control and organization to an activity that can, if left uncontrolled becomes quite chaotic.

Types of software process models

1) **Linear Sequential Model (Water Fall Model)**

The waterfall model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of analysis, design, coding, testing and maintenance phase. Steps that are involved in waterfall model are:

a. **Software Requirement Analysis**
Requirement gathering process is intensified and focused specially on software
There is the study of information domain.
Requirements for system and software must be determined and reviewed with customer

b. **Design**
It is a multi step process to address various aspects to be implemented such as data structures , software architecture, interface representation , procedural (algorithmic details) etc It translates requirements into representation of the software which can be assessed before the code generation.

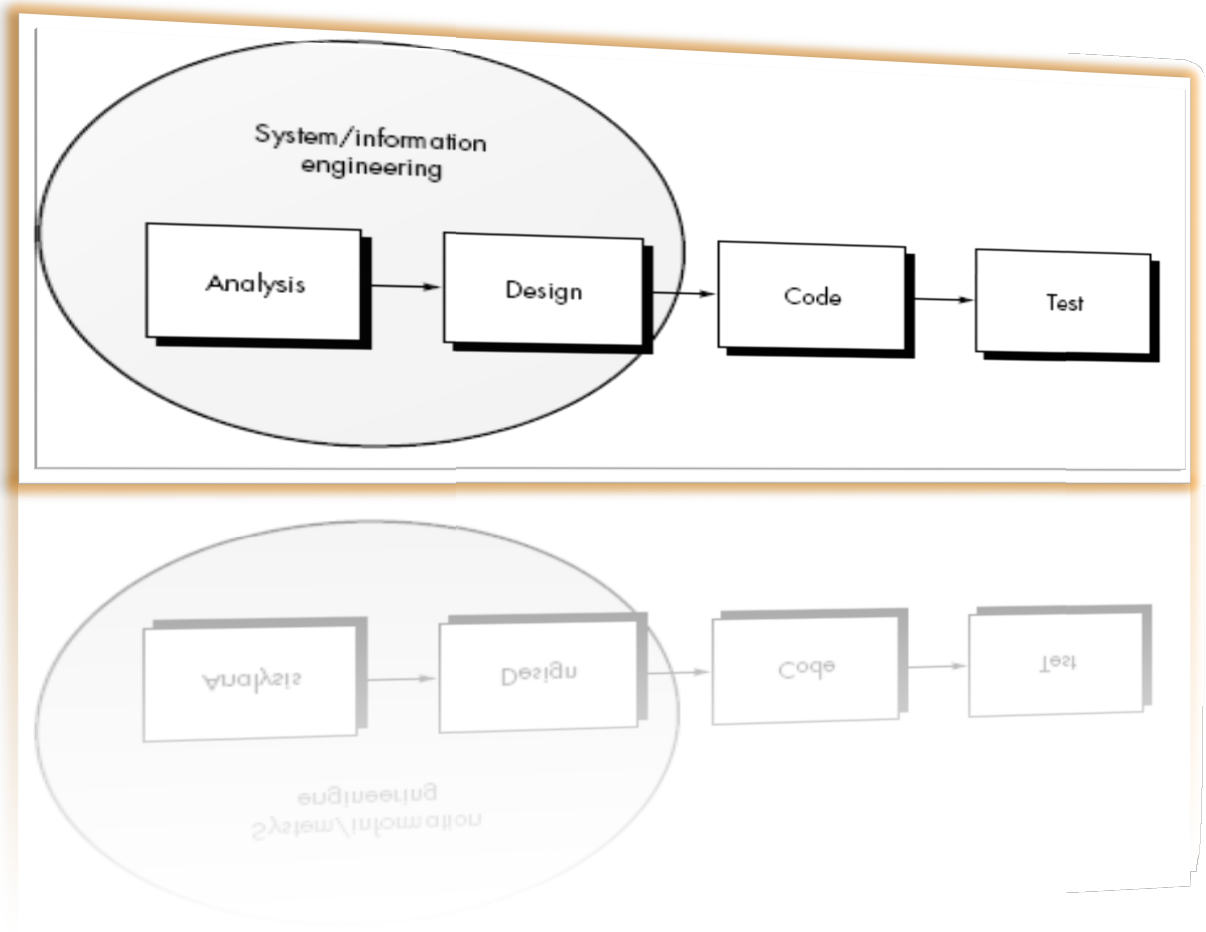
c. **Code generation**
Design translated into a machine readable form.

d. **Testing**
It focuses on the logical intervals of the software (all statements) and functional externals of the software, tests to uncover errors

Basic objective: Defined input should produce desired output.

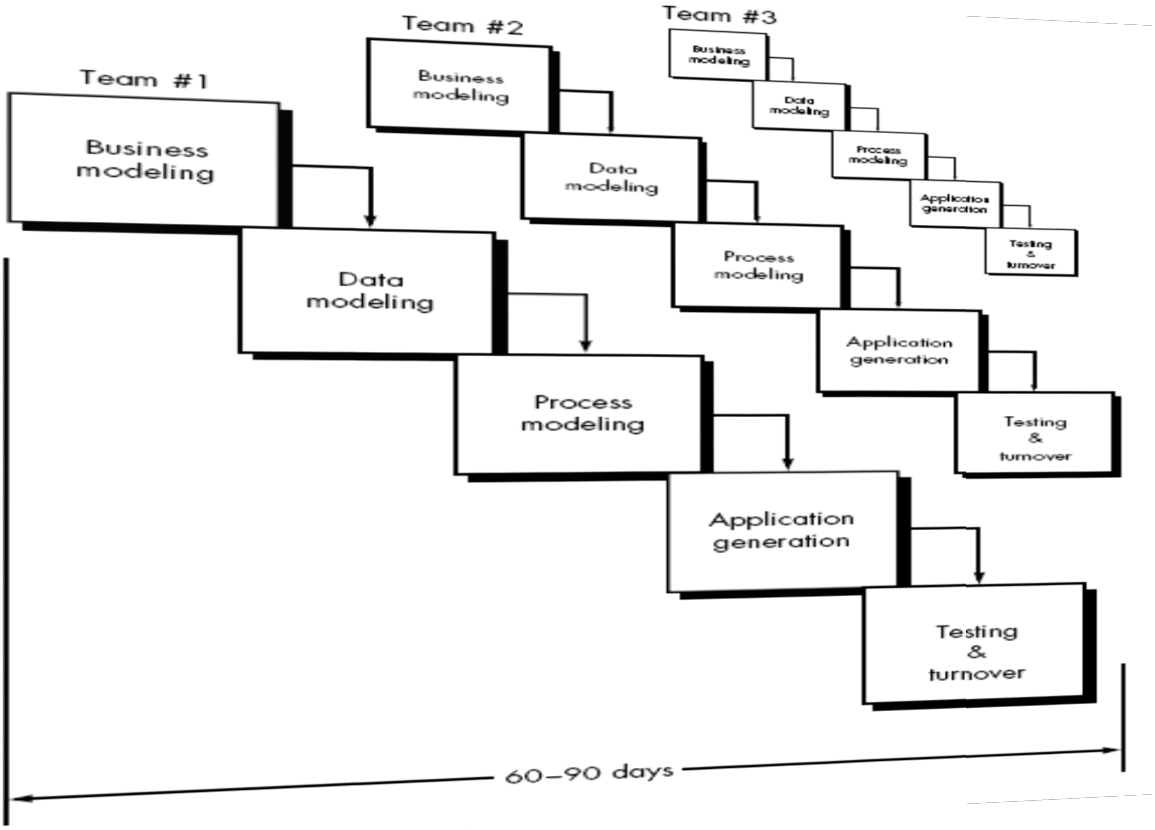
Drawback: Difficulty of accommodating change after the process is underway

Appropriate: When requirements are well understood



2) **Rapid Application Development (RAD) Model**

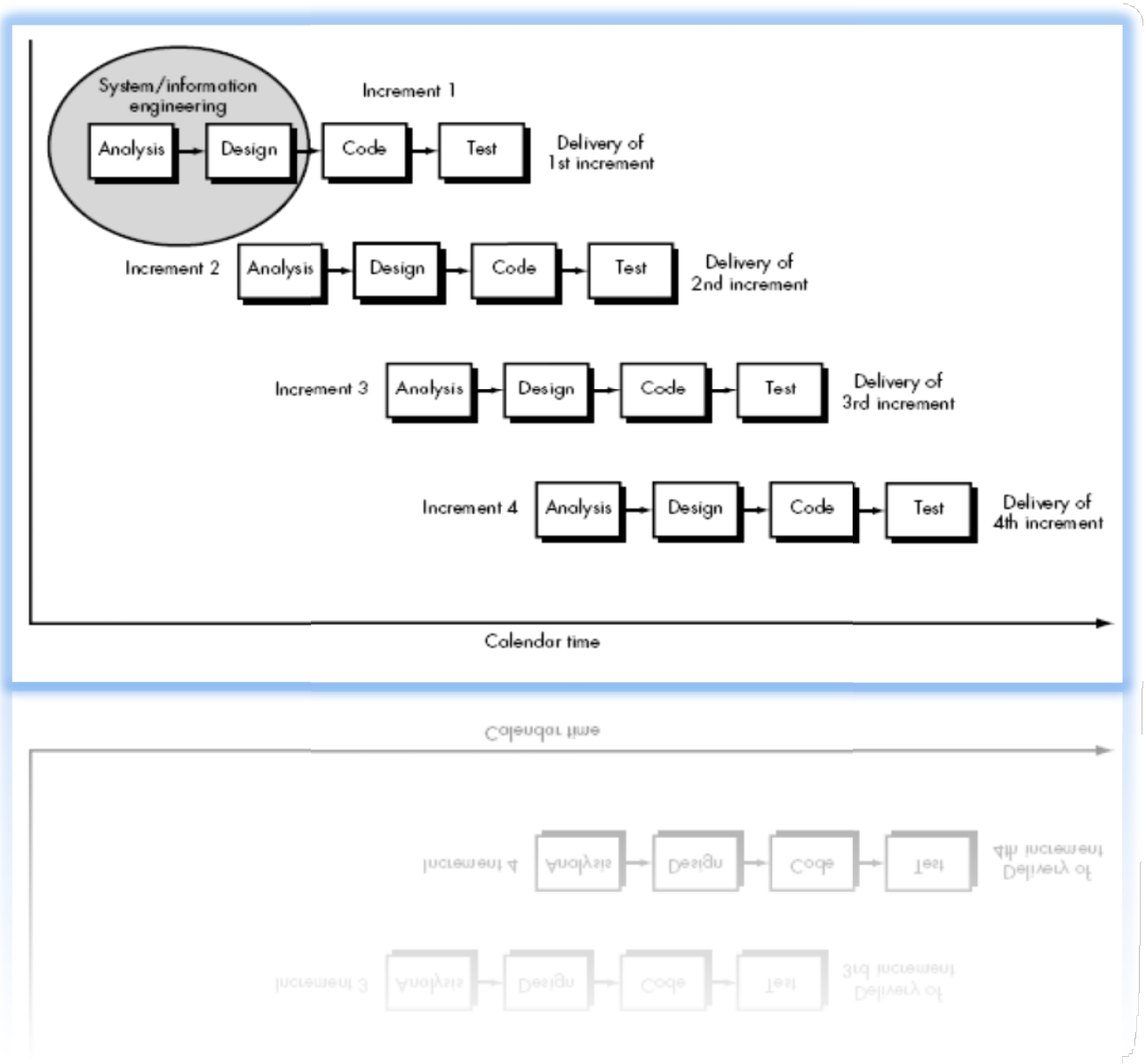
RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. If the requirements are well understood then the project can be finished within 60-90 days using RAD model.



3) **Incremental Model**

This model combines elements of the waterfall model in an iterative fashion. Each linear sequence produces deliverable “increments” of the software. For example; Word processing software developed using this model might deliver basic file management editing, and document production functions in the first increment; and more sophisticated editing, and document production capabilities in the second increment; spelling and grammar checking in the third increment; and so on. When this model is used, the first increment addresses the basic requirements but many supplementary features (known and unknown) remain undelivered.

This model focuses on the delivery of an operational product with each increment. It is useful when staffing is unavailable for a complete implementation by the business deadline that has been set for the project.



4) **Evolutionary Process Models**

Evolutionary models are inherently iterative in nature. It helps to develop increasingly more complete versions of the target software. Business and product requirement changes as the development proceeds, Straight path to end product becomes unrealistic. System requirements are well understood but details of product extensions or system extensions are not known.

Types of evolutionary process models:

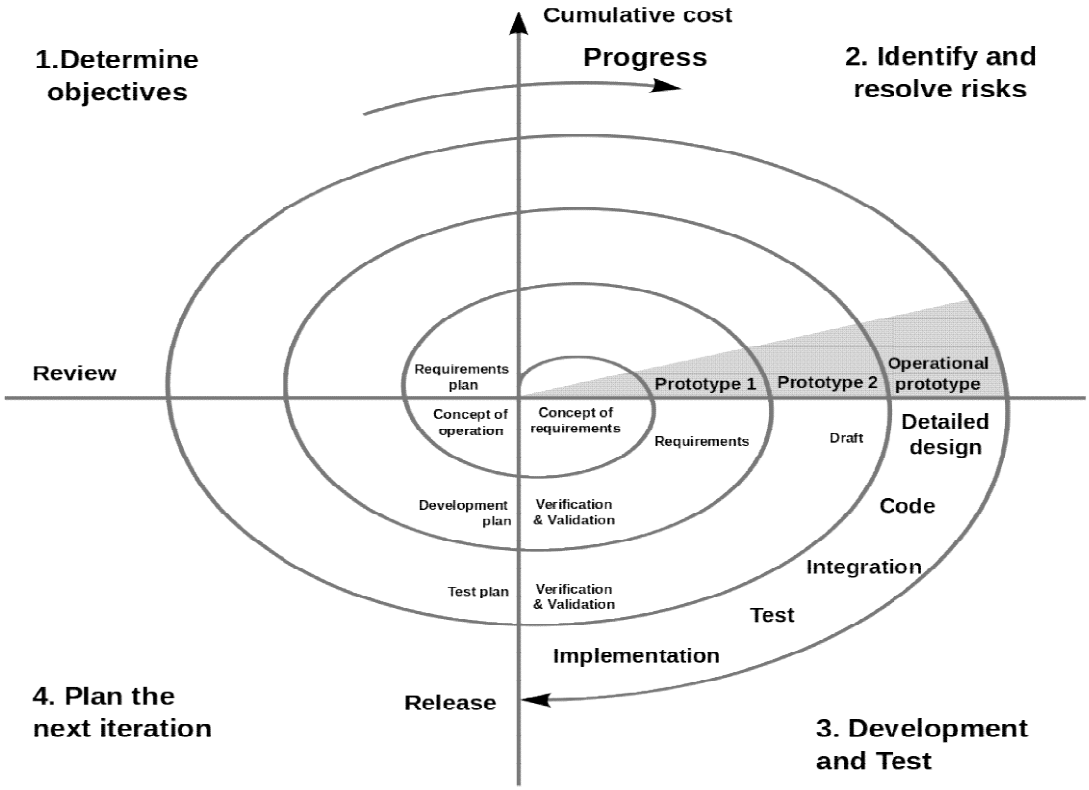
a. **Spiral Model**

The spiral model is a risk-driven process model generator for software projects. Based on the unique risk patterns of a given

project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.

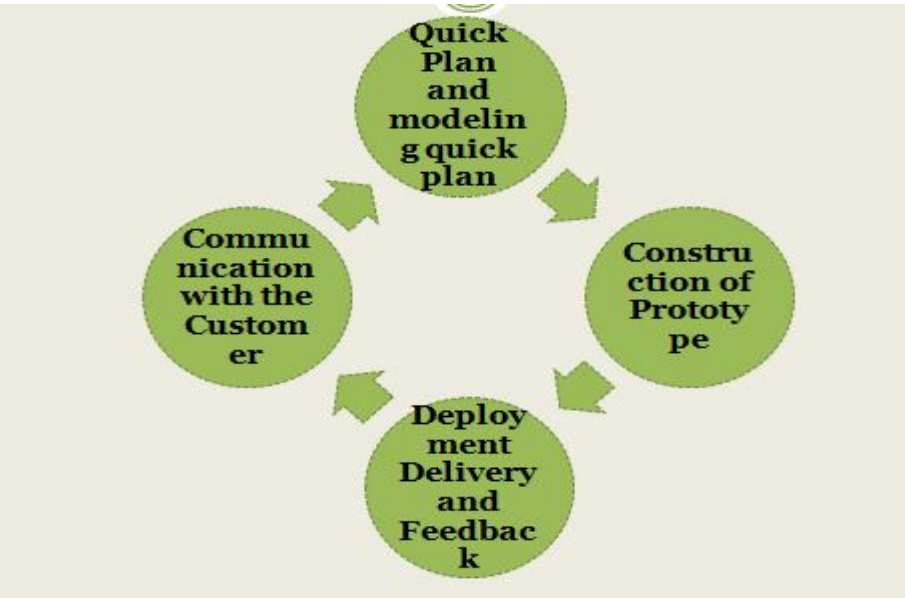
Phases of spiral model are:

- **Customer Communication**: Task for effective communication between customer and developer.
- **Planning**: Task is to define resources, timeline and other project related information.
- **Risk analysis**: Task is to assess technical and management of risks.
- **Engineering**: Task required building one or more representations of the application.
- **Construction and release**: Task to construct, test, install and provide support (documentation, training etc).
- **Customer evaluation**: Task is to obtain customer feedback or evaluation (engineering stage versus implementation stage)



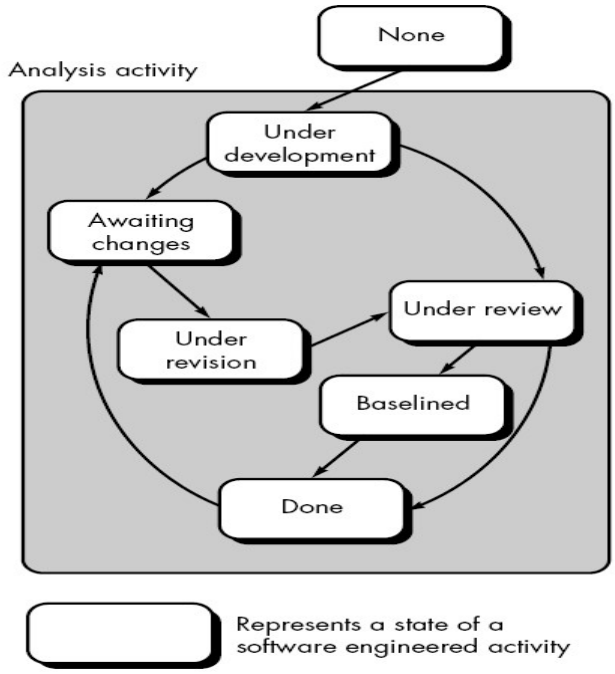
b. **Prototyping Model**

In this model a customer defines a set of general objectives for software, but does not identify detailed input, processing, or output requirements. And the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system. If this is the case, then prototyping paradigm may offer the best approach.



c. **Concurrent Development Model**

This model was proposed by Davis and Sitaram. Defines a series of events that triggers transition from state to state for each software engineering activity. Helps to figure out the actual picture of the state of the project. Instead of showing software engineering activities as a sequence of tasks it defines a network of activities existing simultaneous with other activities .Events generated in one activity may trigger a state transition of an activity. This model is applicable to all types of software development and provides an accurate picture of the current state of a project.



5) **Specialized Process Models**

a. **Component Based Development**

This model incorporates many of the characteristics of the spiral mode. Commercial-off-the Shelf (COTS) software components, developed by vendors who offer them as product, can be used when software is to be built. These targeted functionality with well-defined interfaces that enable the component to be integrated into the software. It is evolutionary in nature, demanding an iterative approach to the creation of software. Component based development model leads to software reuse, and reusability provides software engineers with a number of measurable benefits.

b. **Formal Methods Models**

When formal methods are used during development, they provide a mechanism for eliminating many problems that are difficult to overcome using other software engineering paradigms. Ambiguity,

incompleteness and inconsistency can be discovered and corrected more easily through the application of mathematical analysis. For the time being it is time consuming and expensive. Extensive training is required for necessary background of this model. Difficult to use for the technically unsophisticated customers.

c. **Aspect Oriented Software Development**

Aspect oriented software e development, often referred to as aspect-oriented programming(AOP), is a relatively new software engineering paradigm that provides a process and methodological approach for defining, specifying, designing, and constructing aspects. A distinct aspect- oriented process has not yet matured. However, it is likely that such a process will adopt characteristics of both the spiral and concurrent process models. The evolutionary nature of spiral is appropriate as aspects are identified and then constructed.

PROCESS TECHNOLOGY

Process technology helps the development team by providing tools that helps in analysis of current process, organization of work tasks, to control and monitor progress and in managing technical quality.

Process technology or Process modeling tools or Process management tools are used to represent the key elements of a process so that it can be better understand the actions and work tasks that are required to perform it. It is used to plan, allocate, monitor or even control all software engineering tasks.

PRODUCT AND PROCESS

Process and the product depend on each other. If the process is better product comes better. Software industry has been changing its focus from product to process and process to product each 5-10 years.

THE MANAGEMENT SPECTRUM

Effective software project management focuses on four P's

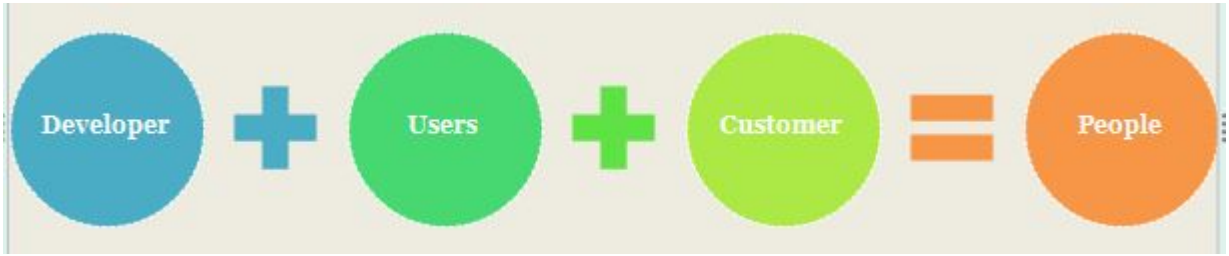
- PEOPLE
- PRODUCT
- PROCESS
- PROJECT

a. **People**:

All the people that are involved in software development directly or indirectly.

- **Senior Managers:** Define the business issues and the strategies which has significant influence on the project
- **Project (Technical Manager):** To plan, motivate, organize and control the technical team (Practitioners)
- **Practitioners:** To deliver the technical skill needed to engineer the software product
- **Customers**: To specify requirements of software product
- **End users:** The actual users who interact with the released software

Team leaders need to have skills to motivate the work force, organize the work so as to give the shape (product) to the concepts (requirements) and ideas and innovations *(model of leadership)*



b. Product

The working final software is the product. Product defines the requirements (input from the customer) for a visible output. It must decide what function transforms the input to output and what performance characteristic is to be considered. Product and problem are related. While starting for a product from specification and analysis to design, coding and testing and even deployment many problems are to be faced.

c. Process

It is needed to establish framework activities for the project depending upon the nature of the project. If the process is weak, the end product will undoubtedly suffer. Decomposition of process is dome by the team remembering constrains like flexibility, time, communication etc.

d. **Project**

A project is a planned undertaking of a piece of work. Need adequate planning and control to manage the complexities. For a successful project few factors are to be considered:

- Need of the user
- Scope of the project
- Management of changes in needs
- Management of technology/technological changes and enhancement.
- Deadlines
- User satisfaction
- User/financial stability
- Skill with technical persons
- Practices and patterns to be followed

THE W5HH PRINCIPLE

Bohem gave an organizing principle for a software process and called it 'WWWWWWHH' principle, which is now being referred as W5HH Principle.

W5HH principle is the series of questions that include 5 W's 2H's question such as,

- Why is the system being developed?
- What will be done?
- When will be done?
- Who is responsible for a function?
- Where they are organizationally located?
- How will the job be done technically and managerially?
- How much of each resource is needed?
- Boehm’s W5HH principle considered the size or complexity of a software project.
- These questions provide an excellent planning outline for the project manager and software team.