Parul® University

NAAC A++

Study of Different Models

Chapter-1: Introduction

Prof. Arpita Vaidya
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

Content

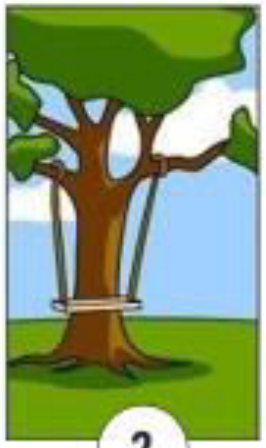
INDEX

Why to Study Software Engineering?

Software Development Life Cycle without Software Engineering

1

How the Customer Explains Requirement



2

How the Project Leader understand it

3

How the System Analyst design it

4

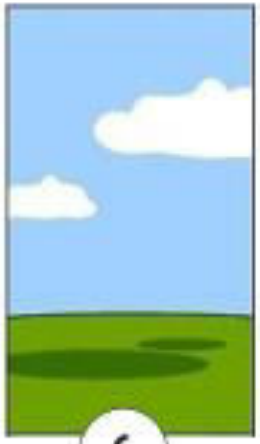
How the Programmer Works on it

5

How the Business Consultant describe it

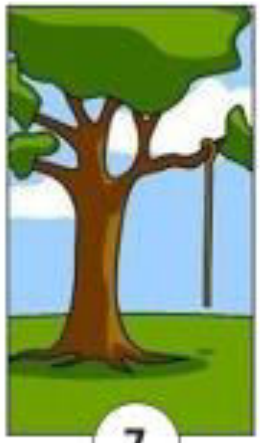
Why to Study Software Engineering?

Software Development Life Cycle without Software Engineering



6

How the Project documented

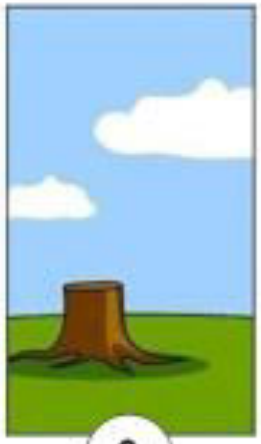


7

What Operations Installed

8

How the Customer billed



9

How it was supported

10

What the customer really needed

SDLC without Software Engineering

Customer Requirement	Solution
• Have one trunk • Have four legs • Should carry load both passenger & cargo • Black in color • Should be herbivorous	• Have one trunk • Have four legs • Should carry load both passenger & cargo • Black in color • Should be herbivorous
	 Our value added, also gives milk

The software development **process** needs to be **engineered** to avoid the **communication gap** & to **meet the actual requirements** of customer within **stipulated budget** & **time**

Software Engineering

	Design	Build	Product
Engineering			
Software Engineering			

What is Software?

Software is :

1) **Computer program** that when executed provide desired features, function & performance
2) **Data Structure** that enable programs to easily manipulate information
3) **Descriptive information** in both hard and soft copy that describes the operation and use of programs

Computer Program + **Data Structure** + **Documents Soft & Hard**

What is Engineering ?

Definition:

- **Software Engineering** is an engineering branch related to the evolution of software product using well-defined **scientific principles**, **techniques**, and **procedures**. The result of software engineering is an effective and reliable software product.

What is Software Engineering?

- The term is made of two words - **Software** and **Engineering.**

  ➢ The **software** is a collection of integrated programs.

- Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.

  ➢ **Engineering** is the application of **scientific** and **practical** knowledge to **invent, design, build, maintain**, and **improve frameworks, processes, etc**.

- **Goal** of software engineering is to produce software that is efficient, easy to use, and easy to maintain

What is Software Engineering?

A branch of engineering that deals with software development
Combines principles of computer science and engineering
Focus on:

- Systematic development
- Maintenance
- Testing and validation

Why Is It Important?
Software is part of everything: phones, cars, health, education
Demand for high-quality, secure, scalable systems
Drives innovation and automation

Requirements

- Software Engineering is required due to the following reasons:

❖ To manage Large software

❖ For more Scalability

❖ Cost Management

❖ To manage the dynamic nature of software

❖ To decrease time

Software Characteristics

1. Software is developed or engineered
2. not manufactured
3. Software doesn't "wear out."
4. Although the industry is moving toward component-based assembly, most software continues to be custom built.

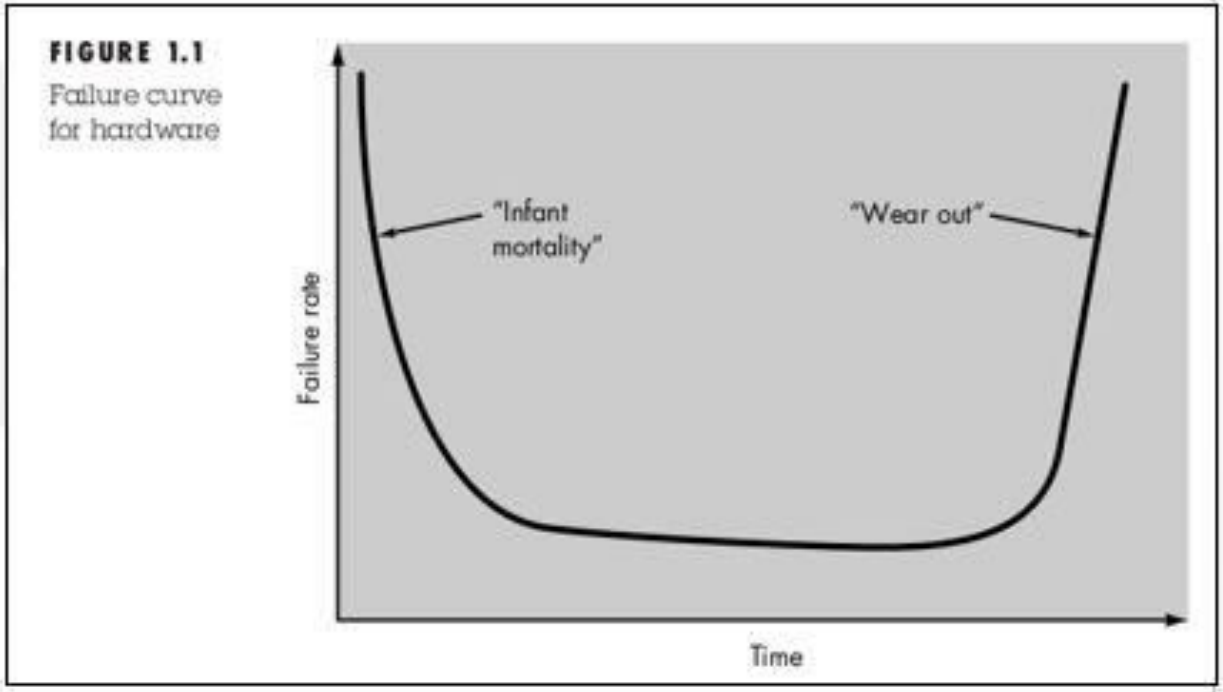
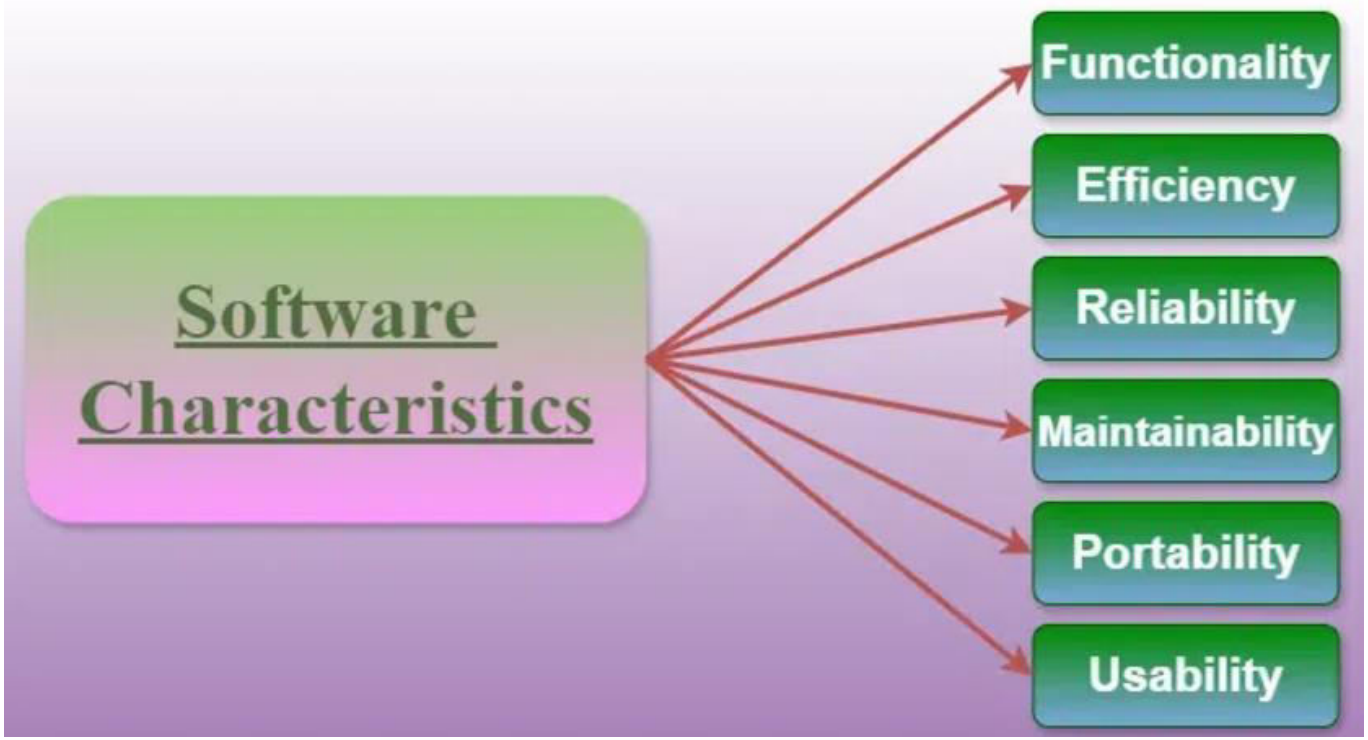


Image source : Presuman, R. S. (1987). *Software engineering: A practitioner's approach*. New York: McGraw-Hill.

COMPONENTS OF SOFTWARE CHARACTERISTICS

- There are 6 components of Software Characteristics-



Software Characteristics

- **Complexity**: Software systems can be highly complex, with numerous interacting components.
- **Invisibility**: Software is intangible and cannot be seen or touched.
- **Changeability**: Software is often changed and updated to adapt to new requirements or fix issues.
- **Conformity**: Software must confirm to external constraints like hardware, system interfaces, and other software.
- **Ease of Replication**: Once developed, software can be easily replicated and distributed with minimal cost.

Software Characteristics

Functionality:

Functionality refers to the set of features and capabilities that a software program or system provides to its users.

functionality in software include:

1. Data storage and retrieval
2. Data processing and manipulation
3. User interface and navigation
4. Communication and networking
5. Security and access control
6. Reporting and visualization
7. Automation and scripting

Software Characteristics

Reliability:

Its ability to perform its functions correctly and consistently over time.

Factors that can affect the reliability of software include:

1. Bugs and errors in the code
2. Lack of testing and validation
3. Poorly designed algorithms and data structures
4. Inadequate error handling and recovery
5. Incompatibilities with other software or hardware

Software Characteristics

Reliability:

To improve the reliability of software, various techniques, and methodologies can be used, such as testing and validation, formal verification, and fault tolerance.

Required functions are:

- Recoverable
- Fault tolerance
- Maturity

Software Characteristics

Efficiency:

- It refers to the ability of the software to use system resources (such as memory, processing power, and network bandwidth) in the most effective and efficient manner.
- Factors that can affect the efficiency of software include:
  - Poorly designed algorithms and data structures
  - Inefficient use of memory and processing power
  - High network latency or bandwidth usage
  - Unnecessary processing or computation and Optimized code

Software Characteristics

Efficiency:

- To improve the efficiency of software, various techniques, and methodologies can be used, such as performance analysis, optimization, and profiling.
- Required functions are:
  - Times
  - Performance

Software Characteristics

Usability:

- It refers to the extent to which the software can be used with ease the amount of effort or time required to learn how to use the software.
- Required functions are:
  - Understandability
  - Learnability
  - Operatability

Software Characteristics

Maintainability:

- It refers to the ease with which modifications can be made in a software system to extend its functionality, improve its performance or correct errors.
- Required functions are:
  - Testability
  - Stability
  - Changeability

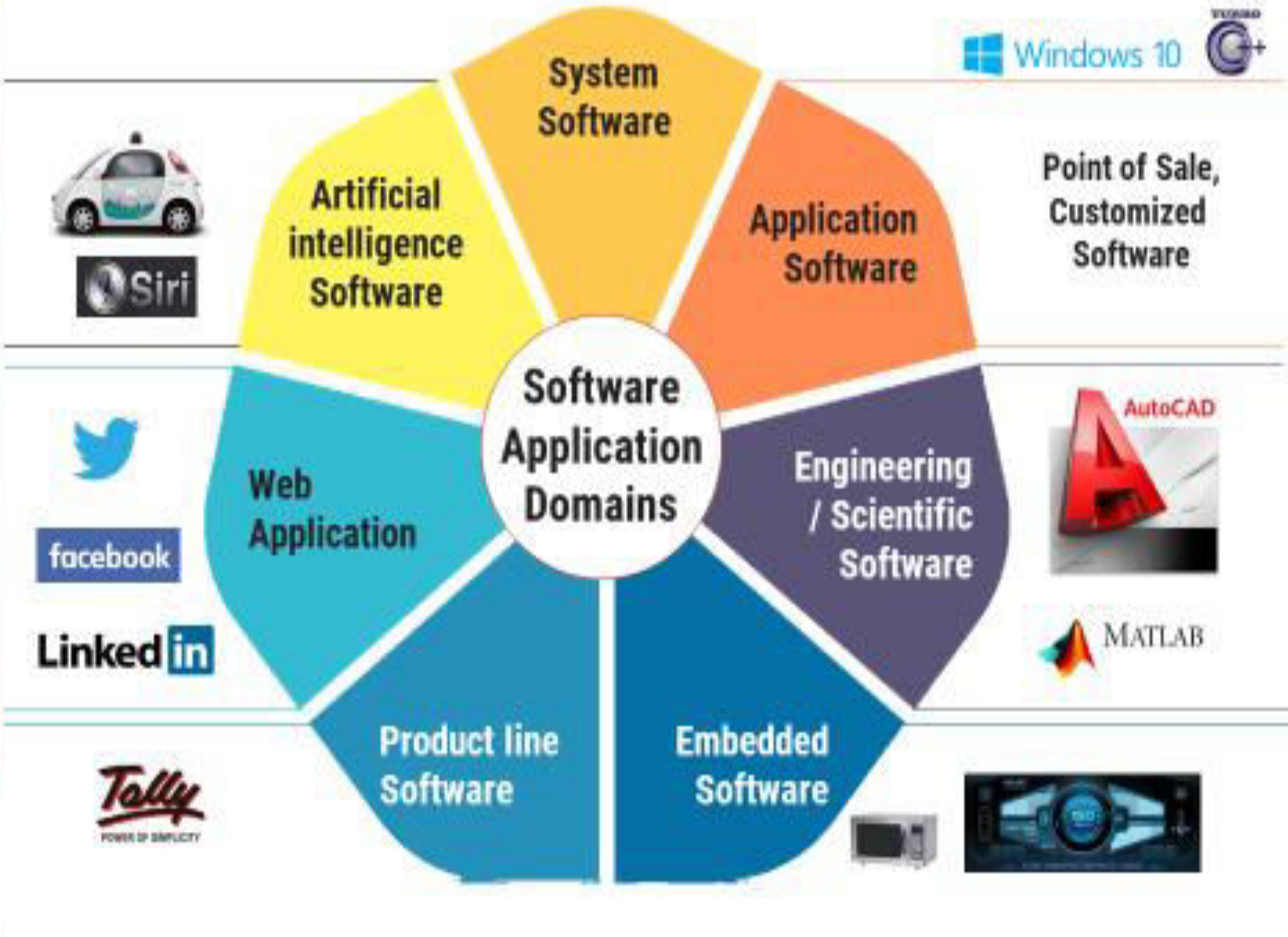
Software Characteristics

Portability:

- A set of attributes that bears on the ability of software to be transferred from one environment to another, without minimum changes.
- Required functions are:
  - Adaptability
  - Install-ability
  - Replace-ability

Software Application Domains

- System Software
- Application Software
- Engineering / Scientific Software
- Embedded Software
- Product line Software
- Web Application
- Artificial intelligence Software



Software Applications

1. **System Software**

Operating Systems (Windows, macOS, Linux, etc.), Utilities (antivirus, disk cleanup, backup and recovery, etc.)

2. **Real Time Software**

Aircraft navigation programs, Multimedia broadcasts, Multi-player video games, Data analysis programs and Stock-trading applications

3. **Business Software**

CRM (customer relationship management), ERP (Enterprise Resource Planning) software, Microsoft Dynamics 365

Software Applications

4. **Engineering & Scientific Software**

AutoCAD (for design), **MATLAB** (for data analysis and modeling), **SolidWorks** (for 3D design), **ANSYS** (for simulation), **SPSS** Statistics (for statistical analysis), and **LabVIEW** (for data acquisition and control systems)

5. **Embedded Software**

Calculators, Image processing systems in medical imaging equipment, Traffic control systems in traffic lights, Control functions of a video game controller

6. **Personal Computer Software**

Operating systems, Internet browsers, Spreadsheets, Bitmap graphics editors

Software Applications

7. Web Based Software

Google Docs, Gmail, Facebook, Twitter, Amazon, and Netflix

8. Artificial Intelligence Software

ChatGPT, Google Assistant, Chatbots, Grammerly, Amazon Alexa,

Face Recognition

PPT Content Resources Reference :

1. **Book Reference**

   Software Engineering: A Practitioner's Approach, by R.S.Pressman published by TMH.

2. **Reference Books:**

3. Software Engineering, 8th Edition by Sommerville, Pearson.
4. Software Engineering 3rd Edition by Rajiv Mall, PHI.
5. An Integrated Approach to Software Engineering by Pankaj Jalote Wiley India, 2009.

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/

Parul® University

NAAC A++

Study of Different Models

Chapter-1: Introduction

Prof. Arpita Vaidya
Assistant Professor
Department of Computer Science and Engineering

Parul® University

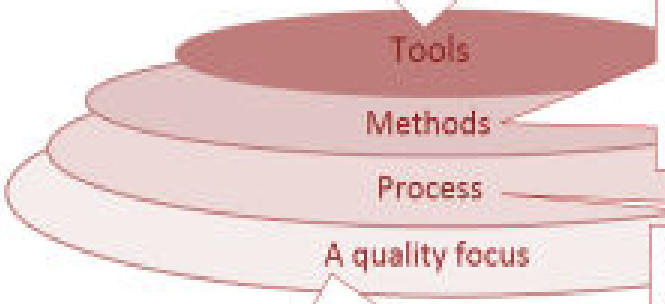
NAAC A++

Content

INDEX

Software Engineering Layered Approach

Software Engineering Tools **allows automation of activities** which helps to perform systematic activities. A system for the support of software development, called **computer-aided software engineering** (CASE). **Examples:** Testing Tools, Bug/Issue Tracking Tools etc…



It provides **technical how-to's** for building software, it encompasses **many tasks** including communication, requirement analysis, design modeling, program construction, testing and support

It is a foundation of Software Engineering, It is the **glue** the holds the **technology layers**, It **defines** a **framework activities**

Defines continuous **process improvement principles**

Software Engineering Layered Approach Cont.

Quality

- Main principle of Software Engineering is Quality Focus.
- An **engineering approach** must have a **focus on quality**.
- Total Quality Management **(TQM)**, **Six Sigma**, **ISO** 9001, ISO 9000-3, CAPABILITY MATURITY MODEL **(CMM)**, **CMMI** & similar approaches encourages a continuous process improvement culture

Process Layer

- It is a foundation of Software Engineering, It is the glue the holds the technology layers together and enables logical and timely development of computer software.
- It **defines** a **framework** with activities for effective delivery of software engineering technology
- It establish the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.

Software Engineering Layered Approach Cont.

Method

- It provides **technical how-to's** for building software
- It **encompasses many tasks** including communication, requirement analysis, design modeling, program construction, testing and support

Tools

- Software engineering tools provide automated or semiautomated support for the process and the methods
- Computer-aided software engineering (**CASE**) is the scientific application of a **set of tools** and **methods** to a software system which is meant to **result in high-quality, defect-free, and maintainable software products**.
- CASE tools automate many of the activities involved in various life cycle phases.

PPT Content Resources Reference :

1. **Book Reference**

   Software Engineering: A Practitioner's Approach, by R.S.Pressman published by TMH.

2. **Reference Books:**

3. Software Engineering, 8th Edition by Sommerville, Pearson.
4. Software Engineering 3rd Edition by Rajiv Mall, PHI.
5. An Integrated Approach to Software Engineering by Pankaj Jalote Wiley India, 2009.

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/

Parul® University

NAAC A++

Study of Different Models

Chapter-1: Introduction

Prof. Arpita Vaidya
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

Content

INDEX

Software Process

- A **process** is a collection of **activities**, **actions** and **tasks** that are performed when some work product is to be created
- A process is not a **rigid prescription** for how to build the software, rather it is **adaptable approach** that enables the people doing the work to **pick** and **choose** the **appropriate set of work actions** and tasks
- An **activity** try to **achieve** a **broad objective** (e.g., communication with stakeholders)
- An **activity** is **applied** regardless of the **application domain**, **size of the project**, **complexity of the effort**, or **degree of accuracy** with which software engineering is to be applied.
- An **action** (e.g., architectural design) **encompasses** a **set of tasks** that **produce** a major **work product** (e.g., an architectural design model).
- A **task** **focuses** on a **small, but well-defined objective** (e.g., conducting a unit test) that **produces** a **noticeable outcome**.
- **Each of these** activities, actions & tasks **reside within** a **framework** or model

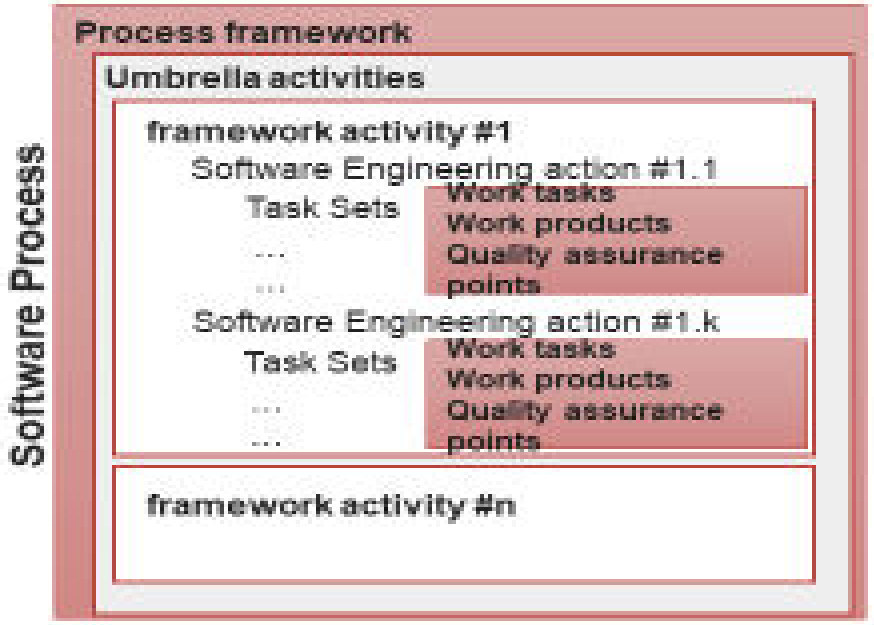
Software Process

- Figure represents "The Software Process"
- Each framework **activity is populated** by **set of** software engineering **actions**
- Each software engineering **action is defined by** a **task set** that identifies work to be completed, product to be produced, quality assurance points & milestones to indicate progress

The **purpose** of software process is

- to **deliver** software in **timely** manner and
- within sufficient **quality to satisfy** those
  - Who has given proposal for software development and
  - Those who will use software

Software Process Framework



Process Framework Activities (CPMCD)

A process framework establishes the foundation for complete software engineering process, it encompasses five activities

Communication 	Communication with Customers / stockholders to understand project requirements for defining software features	**Planning** 	Software Project Plan which defines workflow that is to follow. It describes technical task, risks, resources, product to be produced & work schedule
Modeling 	Creating models to understand requirements and shows design of software to achieve requirements	**Construction** 	Code Generation (manual or automated) & Testing (to uncover errors in the code)
Deployment 	Deliver Software to Customer Collect feedback from customer based on evaluation Software Support		

Process activities are listed below:-

i. **Communication:** It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.

ii. **Planning:** It basically means drawing a map for reduced the complication of development.

iii. **Modeling:** In this process, a model is created according to the client for better understanding.

iv. **Construction:** It includes the coding and testing of the problem.

v. **Deployment:-** It includes the delivery of software to the client for evaluation and feedback.

Software development life cycle

- SDLC consists of a precise plan that describes how to develop, maintain, replace, and enhance specific software. The life cycle defines a method for improving the quality of software and the all-around development process.
- Without using an exact life cycle model, the development of a software product would not be in a systematic and disciplined manner.

Software development life cycle

Software development life cycle

- Also known as **Software development life cycle (SDLC)** or Application development life cycle Models
- Process models **prescribe** a distinct set of **activities, actions, tasks and milestones (deliverables)** required to engineer high quality software.
- Process **models are not perfect,** but **provide roadmap** for software engineering work.
- Software models provide stability, control and organization to a process that if not managed can easily get out of control.
- Software process models are adapted (adjusted) to meet the needs of software engineers and managers for a specific project.

Generic View Of Software Engineering

- The work associated with software engineering can be categorized into three generic phases, regardless of application area, project size, or complexity.
- The definition phase: focuses on what, identify what information is to be processed, what function and performance are desired, what system behavior can be expected, what interfaces are to be established, what design constraints exist, and what validation criteria are required to define a successful system.
- The key requirements of the system and the softwar major tasks
- 1. system or information engineering
- 2. software project planning
- 3. requirements analysis

Think about MIS

Image source : Google

PPT Content Resources Reference :

1. **Book Reference**

   Software Engineering: A Practitioner's Approach, by R.S.Pressman published by TMH.

2. **Reference Books:**

3. Software Engineering, 8th Edition by Sommerville, Pearson.
4. Software Engineering 3rd Edition by Rajiv Mall, PHI.
5. An Integrated Approach to Software Engineering by Pankaj Jalote Wiley India, 2009.

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/

Parul® University

NAAC A++

Study of Different Models

Chapter-1: Introduction

Prof. Arpita Vaidya
Assistant Professor
Department of Computer Science and Engineering

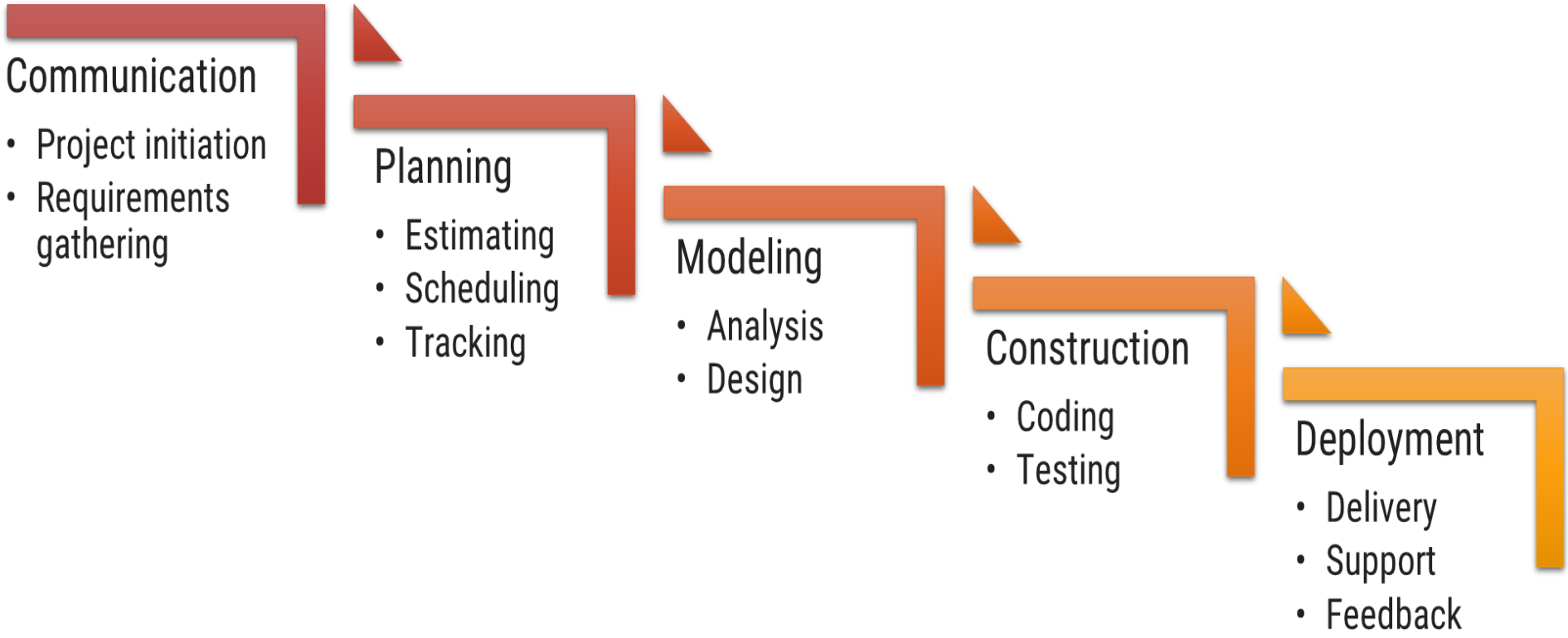
Parul® University

NAAC A++

INDEX

Content

Waterfall Model



When **requirements** for a problems are **well understood** then this model is used in which **work flow** from communication to deployment is **linear**

Waterfall Model

When to use ?

- Requirements are very well known, clear and fixed
- Product definition is stable
- Technology is understood
- There are no ambiguous (unclear) requirements
- Ample (sufficient) resources with required expertise are available freely
- The project is short

Advantages

- Simple to implement and manage

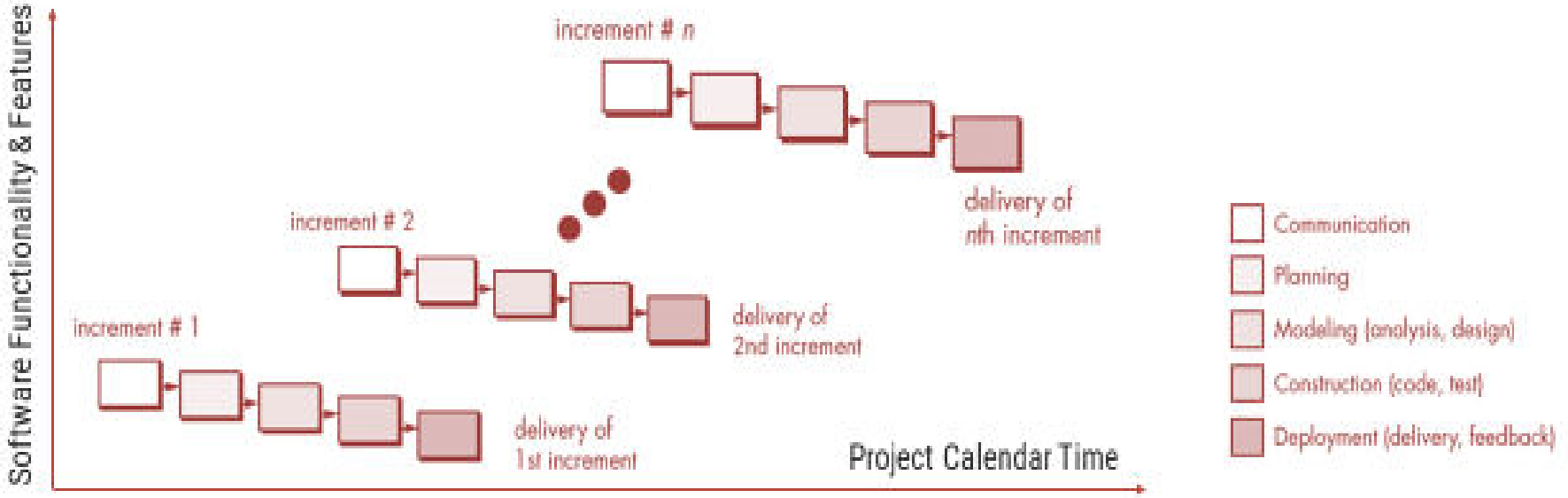
Drawbacks

- Unable to accommodate changes at **later stages**, that is required in most of the cases.
- Working version is not available during development. Which can lead the development with major mistakes.
- Deadlock can occur due to delay in any step.
- Not suitable for large projects.

Incremental Process Model

- There are many situations in which **initial software requirements** are reasonably **well defined**, but the **overall scope of the development** effort prevent a purely linear process.
- In addition, there may be a **compelling need** to provide a **limited set of software functionality** to users **quickly** and then **refine and expand on that functionality** in later software releases.
- In such cases, there is a need of a process model that is designed to produce the software in increments.

Incremental Process Model



- The incremental model **combines** elements of **linear** and **parallel** process flows.
- This model applies linear sequence in a iterative manner.
- Initially **core working product** is **delivered**.
- **Each** linear **sequence** produces deliverable **“increments”** of the software.

Incremental Process Model

- It might deliver basic file management, editing and document production functions in the first increment
- more sophisticated editing in the second increment;
- spelling and grammar checking in the third increment; and
- advanced page layout capability in the fourth increment.

When to use?

- When the **requirements** of the **complete** system are clearly **defined** and understood but **staffing is unavailable** for a **complete implementation** by the business deadline.

Advantages

- Generates **working software quickly** and early during the software life cycle.
- It is **easier to test** and debug during a smaller iteration.
- **Customer** can **respond** to each built.
- **Lowers initial** delivery **cost**.
- **Easier** to **manage risk** because risky pieces are identified and handled during iteration.

Evolutionary Software Process Model

- Evolutionary models are iterative. They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.
  - Incremental Model
  - Spiral Model
  - Concurrent Development Model

Image source : https://pixabay.com/illustrations/gps-map-phone-direction-smartphone-5137225/

Evolutionary Software Process Model

- When a set of **core product** or system requirements is **well understood** but the **details of product** or system extensions have **yet to be defined**.
- In this situation there is **a need of process model** which specially designed to accommodate **product** that **evolve with time**.
- **Evolutionary Process Models** are specially meant for that which produce an increasingly more complete version of the software with each iteration.
- Evolutionary Models are **iterative**.
- They are characterized in a manner that enables you to develop **increasingly more complete versions of the software**.
- Evolutionary models are
  - **Prototyping Model**
  - **Spiral Model**

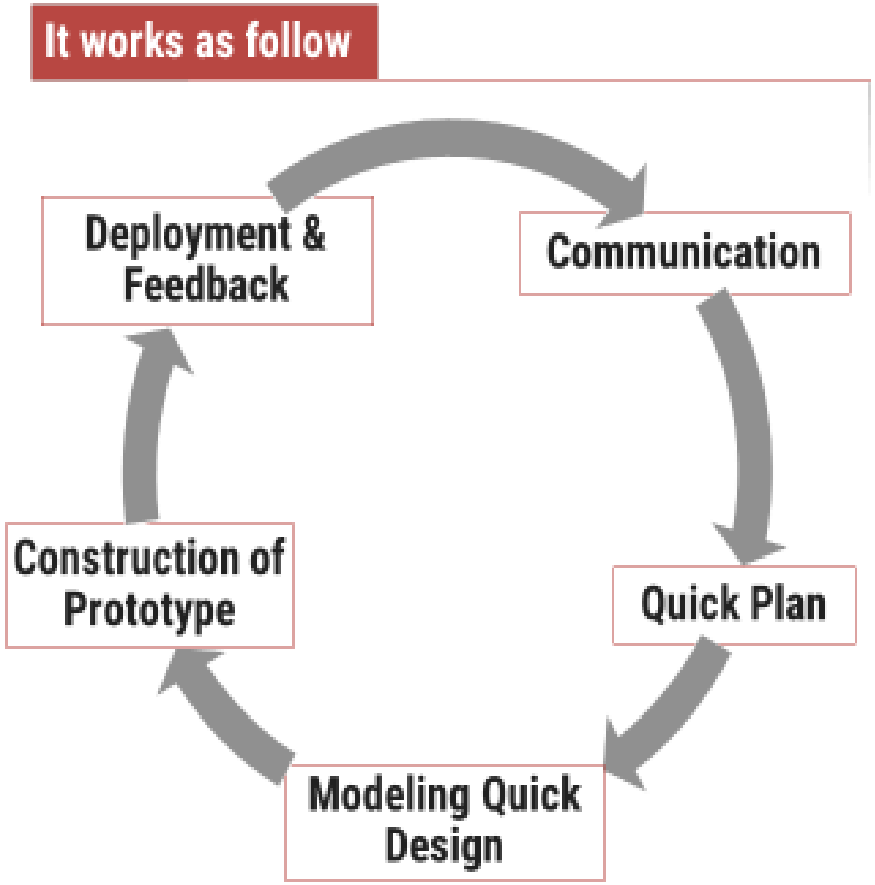
Prototype Model

When to use?

- **Customers have** general **objectives of software** but **do not have detailed requirements** for functions & features.
- **Developers** are **not sure** about **efficiency of an algorithm** & **technical feasibilities**.

- It serves as a **mechanism** for **identifying software requirements**.
- Prototype can be serve as "**the first system**".
- Both stakeholders and software engineers like prototyping model
  - Users get feel for the actual system
  - Developers get to build something immediately

Prototype Model



- **Communicate** with stockholders & **define objective** of Software
- **Identify requirements** & design **quick plan**
- **Model** a quick **design** (focuses on visible part of software)
- **Construct Prototype** & deploy
- Stakeholders **evaluate** this **prototype** and provides **feedback**
- Iteration occurs and **prototype** is **tuned** based on **feedback**

Prototype Model

Problem Areas

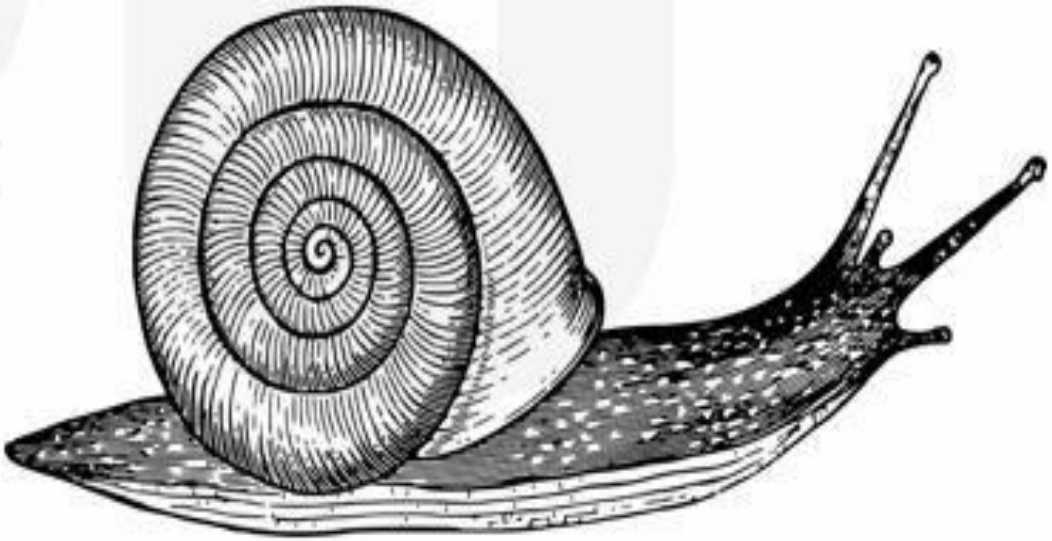
- **Customer demand** that "**a few fixes**" be applied to **make** the **prototype a working product**, due to that software quality suffers as a result
- **Developer** often makes **implementation** in order to get a prototype working quickly; **without considering other factors** in mind like OS, Programming language, etc.

Advantages

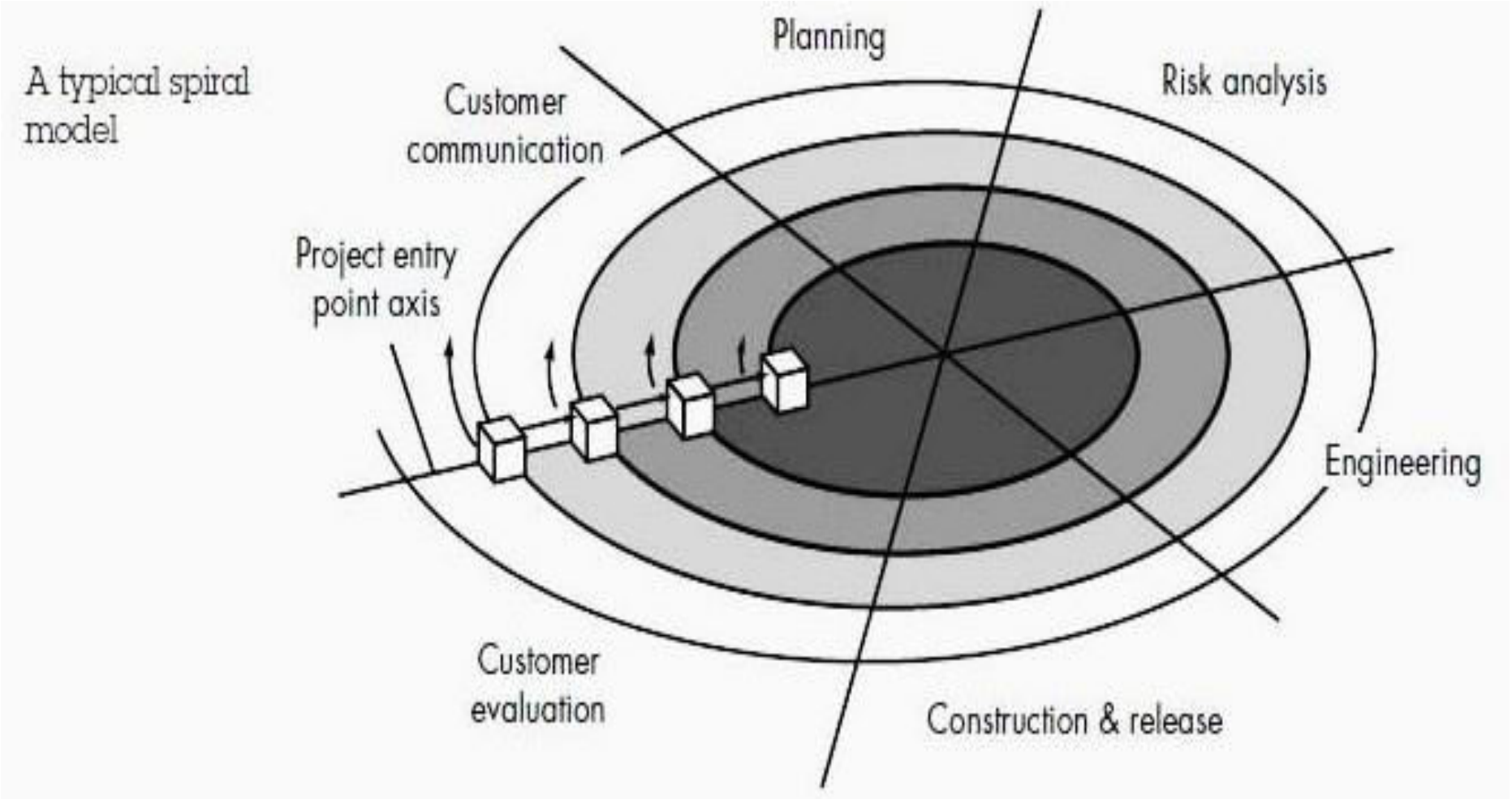
- **Users** are actively **involved** in the **development**
- Since in this methodology a working model of the system is provided, the **users get a better understanding** of the **system** being developed
- **Errors** can be **detected** much **earlier**

Spiral Model

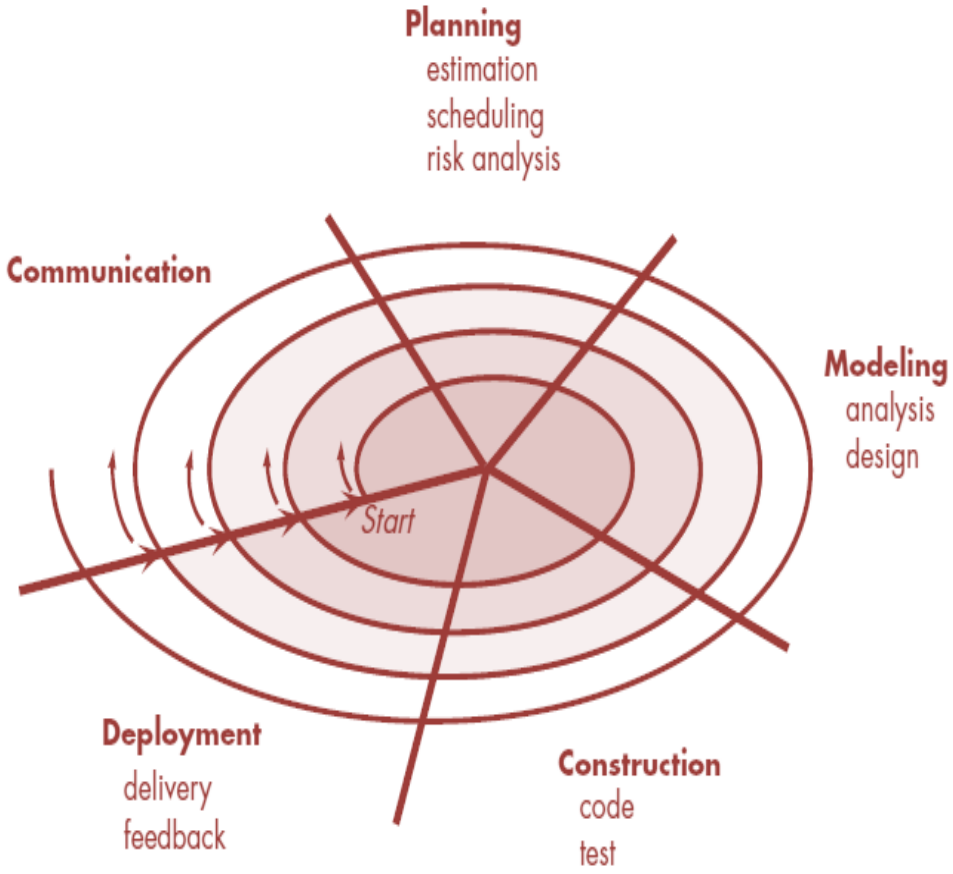
- Software is developed in a series of incremental releases
- During early iterations incremental release might be a prototype. During later iterations, increasingly more complete versions of the engineered system are produced.
- A spiral model is divided into a number of framework activities, also called
- task regions –
  - Customer communication
  - Planning
  - Risk analysis
  - Engineering
  - Construction and release
  - Customer evaluation



Spiral Model



Spiral Model



- It provides the **potential** for **rapid development**.
- Software is developed in a series of evolutionary releases.
- **Early iteration** release might be **prototype** but **later iterations** provides more **complete version of software**.
- It is divided into framework activities (C,P,M,C,D). Each activity represent one segment of the spiral
- **Each pass** through the **planning** region results in **adjustments** to
  - the **project plan**
  - **Cost** & **schedule** based on feedback

Spiral Model

When to use spiral Model ?

- For development of **large scale / high-risk projects**.
- When costs and **risk evaluation is important**.
- Users are **unsure** of their **needs**.
- **Requirements** are **complex**.
- New product line.
- Significant (**considerable**) **changes** are expected.

Advantages

- High amount of risk analysis hence, **avoidance of Risk** is enhanced.
- **Strong approval** and **documentation** control.
- **Additional functionality** can be **added** at a later date.
- **Software** is **produced early** in the Software Life Cycle.

Disadvantages

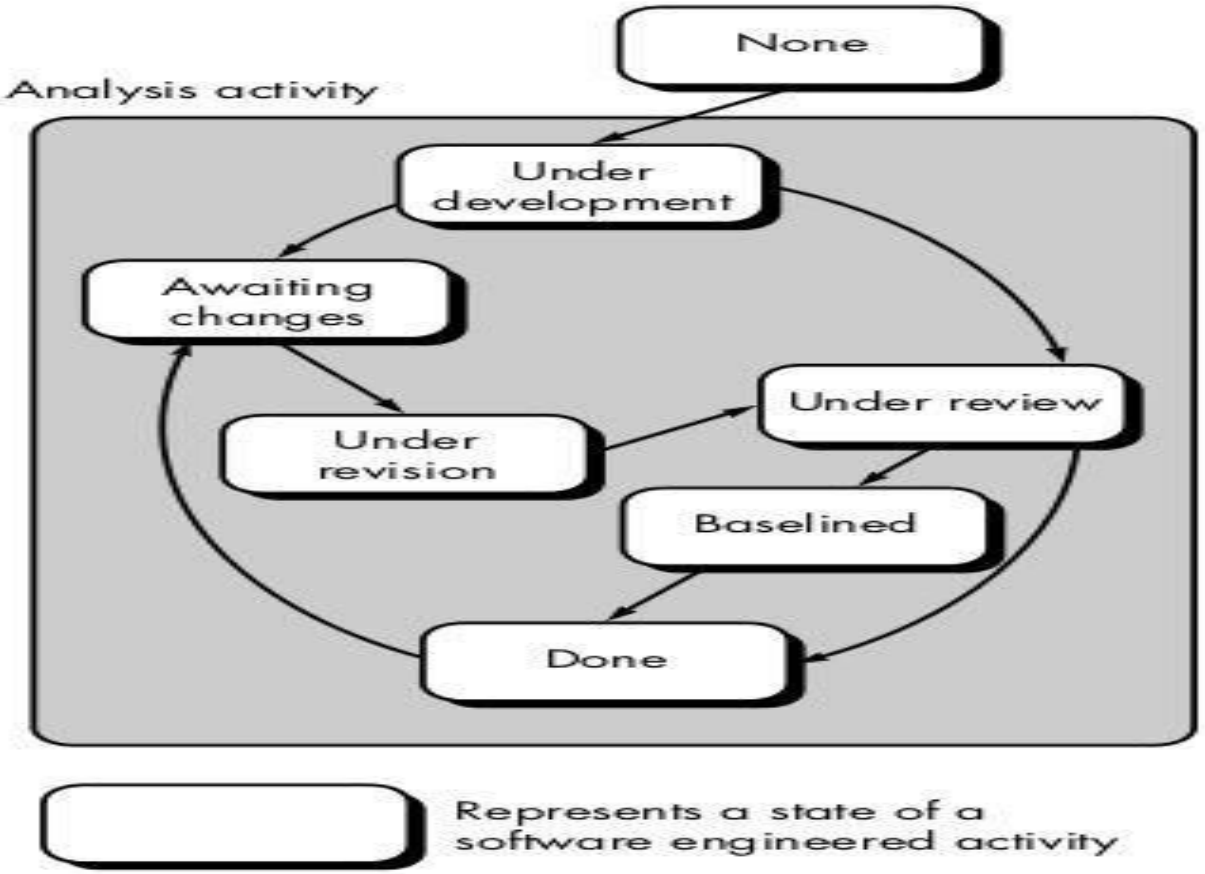
- Can be **a costly model** to use.
- Risk analysis **requires highly specific expertise**.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

Concurrent process

- Series of major technical activities
- Series of tasks & their associated states.
- All activities exist concurrently
- All activities reside in different states.
- Events that will trigger transitions from state to state for activities.

Concurrent process

FIGURE 2.10
One element of the concurrent process model



PPT Content Resources Reference :

1. **Book Reference**

   Software Engineering: A Practitioner's Approach, by R.S.Pressman published by TMH.

2. **Reference Books:**

3. Software Engineering, 8th Edition by Sommerville, Pearson.
4. Software Engineering 3rd Edition by Rajiv Mall, PHI.
5. An Integrated Approach to Software Engineering by Pankaj Jalote Wiley India, 2009.

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/

Parul® University

NAAC A++

Study of Different Models
Chapter-1:Agile Model

Prof. Arpita Vaidya
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

INDEX

Content

Agile Development

Agility

- Agility is ability to move quickly and easily.
- Property consisting of quickness, lightness & ease of movement
- Ability to create and respond to **change**
- Ability to quickly reprioritize use of resources on requirements, technology, and knowledge shift
- A very fast response to sudden market **changes**
- Respond to threats by intensive customer interaction
- Use of evolutionary, incremental, and iterative delivery to converge on an optimal customer solution
- Maximizing BUSINESS VALUE with right sized, just- enough, and just-in-time processes and documentation

Agile Methodology

- Best suited where requirements usually **change** rapidly during the development process
- **#IMP: The Manifesto for Agile Software Development :**
- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to **change** over following a plan

Change ! Change ! Change !

- Why there are changes in requirements? It is not necessary that it's a mistake. there are four fundamental sources of change [pressman]:
- New business or market conditions dictate changes in product requirements or business rules.

Change ! Change ! Change !

3. Reorganization or business growth / downsizing causes changes in project priorities or software engineering team structure.

4. Budgetary or scheduling constraints cause a redefinition of the system or product: money & time are most important factors to be taken care in any project.

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Where agile methodology not work

Project plan & requirements are clear & unlikely to change

Unclear understanding of Agile Approach among Teams

Where agile methodology does not work

Big Enterprises where team collaboration is tough

Agile Process Models

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method (DSDM)
- Feature Driven Development (FDD)
- Crystal
- Agile Modelling (AM)

Extreme Programming

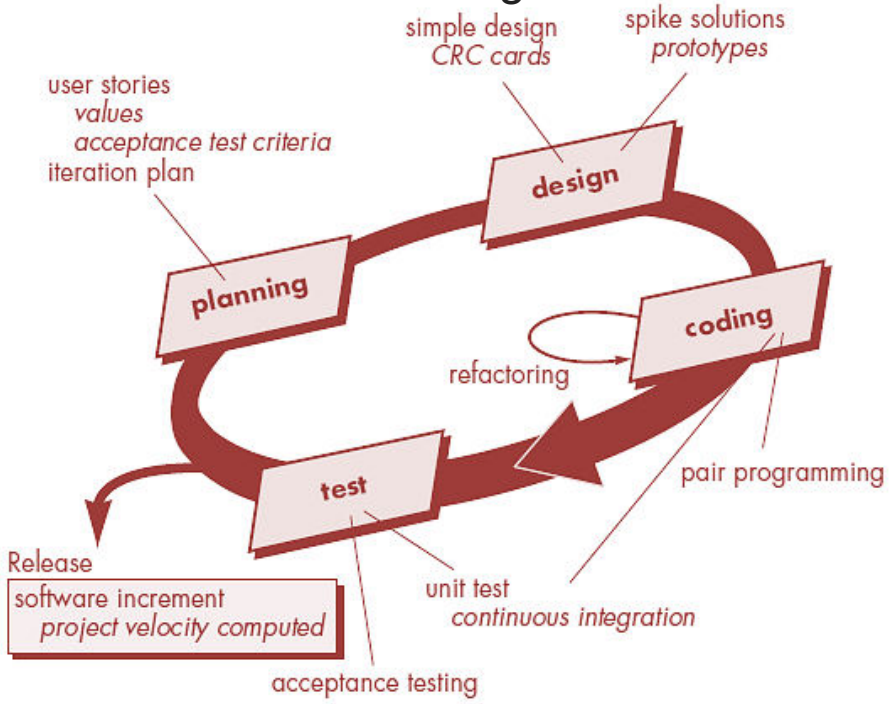
- The most widely used approach to agile software development
- A variant of XP called **Industrial XP (IXP)** has been proposed to target process for large organizations
- It uses **object oriented approach** as its preferred development model

XP Values

- **Communication:** To achieve effective communication, it **emphasized close & informal (verbal) collaboration** between customers and developers
- **Simplicity:** It restricts developers to **design** for **immediate needs not** for **future needs**
- **Feedback:** It is derived **from** three sources the **implemented software**, the **customer** and **other software team members**, it uses **Unit testing** as primary testing
- **Courage:** It demands courage (discipline), there is often significant pressure to design for future requirements, XP team **must have the discipline (courage) to design for today**
- **Respect:** XP team **respect** among **members**

XP Process

1. Planning □ **2.** Design □ **3.** Coding □ **4.** Testing



Planning

User Stories

- **Customers assigns value** (priority)
- **Developers assigns cost** (number of development weeks)

Project velocity

- Computed at the end of first release
- **Number of stories implemented in first release**
- Estimates for future release
- **Guard against over-commitment**

XP Process

Design

CRC card

Class Name	
Responsibilities	Collaborators

- **Keep-it-Simple** (Design of extra functionality is discouraged)
- **Preparation of CRC card** is work project
  - CRC cards identify and organize object oriented classes
- **Spike Solutions (**in case of difficult design problem is encountered**)**
  - Operational prototype intended to clear confusion
- Refactoring
- Modify internals of code, No observable change

Coding

- **Develops** a series of **Unit test** for stories included in current release
- **Complete code** perform **unit-test** to get immediate feedback
- XP recommend **pair-programming**, **"Two heads are better than one"**
- **Integrate code** with other team members, this **"continuous integration"** helps to avoid compatibility & interfacing problems, **"smoke testing"** environment to uncover errors early

Testing

- **Unit test** by **developers** & fix small problems
- **Acceptance tests** - Specified by **customer**
- This encourages regression testing strategy whenever code is modified

Pair Programming

- Observer + Programmer pair
- pairs discuss software before development
- fewer false starts and less rework.
- Less dependency on specific person
- Easy sharing of knowledge
- Reduced risk
- collective ownership
- Informal reviews
- Continuous code Refactoring

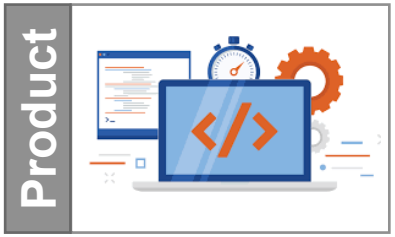
What is Scrum?

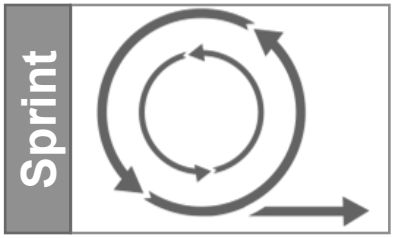
Scrum is an agile process model which is used for **developing** the **complex software** systems.

It is a **lightweight process framework**.

Lightweight means the **overhead of the process is kept as small** as possible in order to maximize the productivity.

A scrum is a method of restarting play in rugby that involves players packing closely together with their heads down and attempting to gain possession of the ball.





Scrum Frame work at Glance

Inputs from Customers, Team, Managers

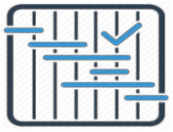
Team Selects starting at top as much as it can commit to deliver by end of sprint

Scrum Master

Daily Scrum Meetings

Product Owner

Product Backlo



Sprint Planning Meeting

Sprint Backlo g

Sprint Review

Finished Work

Sprint Retrospective

Prioritized list of what is required: features, bugs to fix...

Sprint end date and team deliverable do not change

Backlog

- It is a **prioritized list of project requirements** or features that must be provided to the customer.
- The **items can be included** in the backlog at **any time**.
- The **product manager analyses** this **list** and **updates** the **priorities** as per the requirements.

Sprint

- These are the **work units** that are needed **to achieve** the requirements mentioned in the backlogs.
- Typically the sprints have **fixed duration** or time box (of **2 to 4 weeks,**
- **Change** are **not introduced** during the **sprint**.
- Thus sprints allow the team **members** to **work** in **stable** and **short-term environment**

Backlog

- It is a **prioritized list of project requirements** or features that must be provided to the customer.
- The **items can be included** in the backlog at **any time**.
- The **product manager analyses** this **list** and **updates** the **priorities** as per the requirements.

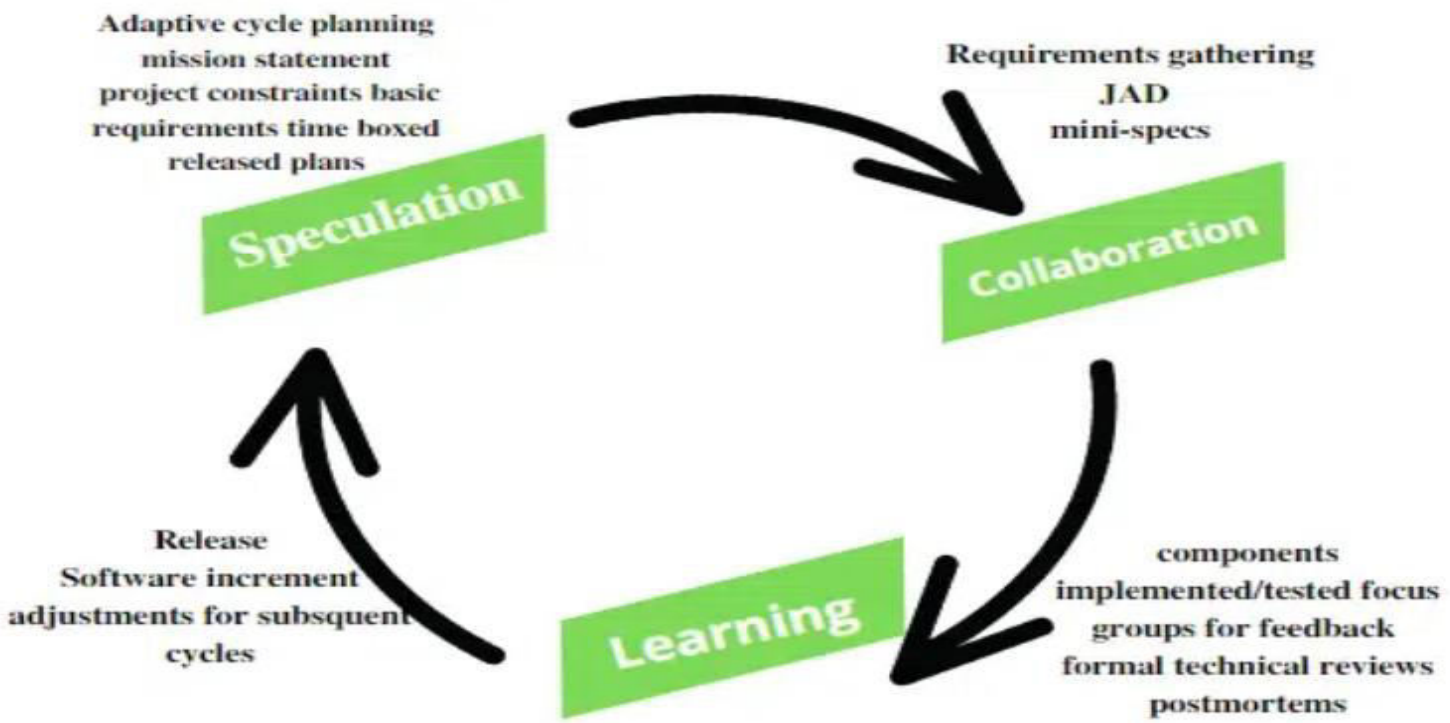
Sprint

- These are the **work units** that are needed **to achieve** the requirements mentioned in the backlogs.
- Typically the sprints have **fixed duration** or time box (of **2 to 4 weeks,**
- **Change** are **not introduced** during the **sprint**.
- Thus sprints allow the team **members** to **work** in **stable** and **short-term environment**

ADAPTIVE SOFTWARE DEVELOPMENT

- Emphasizes adaptability to respond to changing requirements and environments.
- It focuses on **continuous learning** and is considered a part of agile software development.
- **Iterative Development:** ASD breaks down the project into small, manageable iterations, delivering incremental values.
- **Risk Management:** ASD involves identifying and addressing risks early in the development process.
- **Continuous Testing:** It integrates testing throughout the development process to ensure high-quality outcomes.
- **Decentralized Control:** It encourages decision-making at the team level instead of relying on top-down directives.
- **Customer Satisfaction:** It prioritizes customer satisfaction by delivering products that meet their needs and expectations.

ADAPTIVE SOFTWARE DEVELOPMENT PHASES



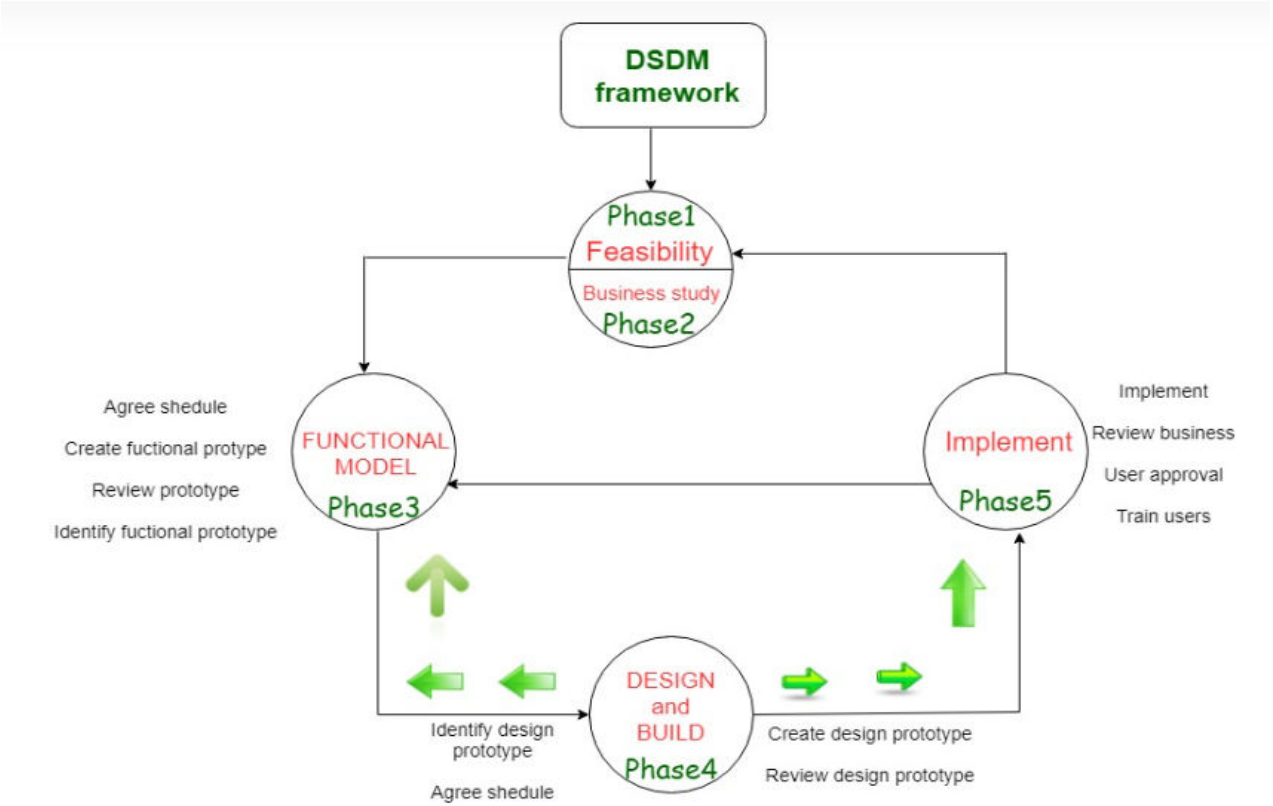
Phases of ASD

Dynamic System Development Methods

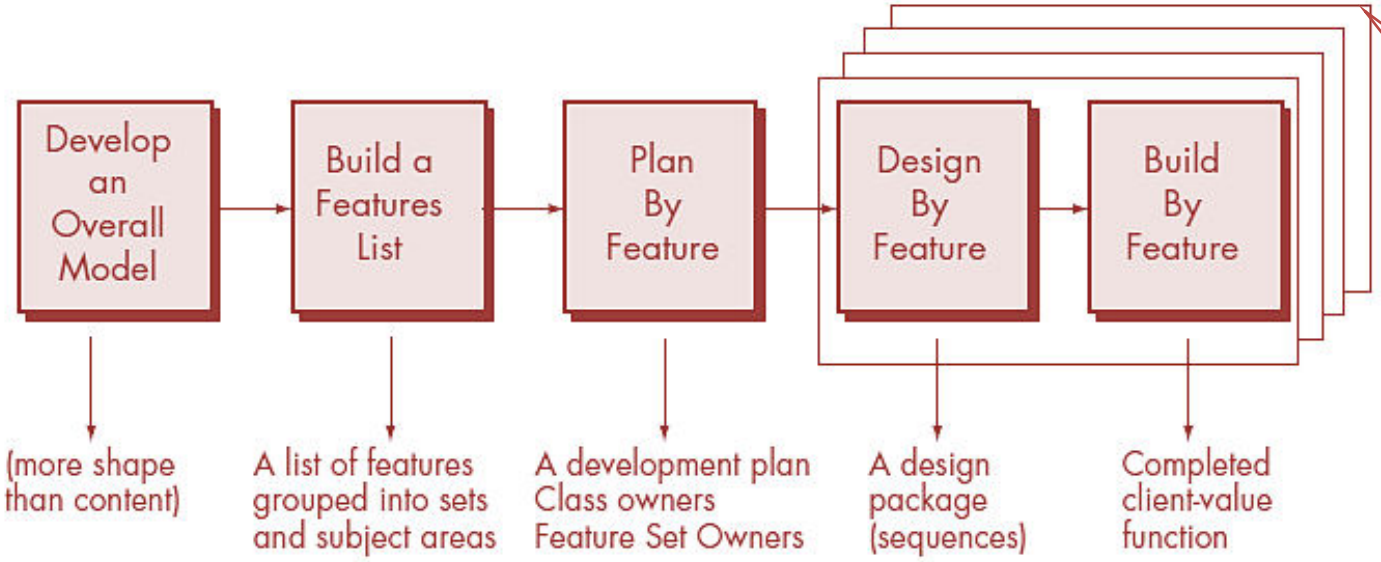
- **Feasibility study:** By analysing the business requirements and constraints the viability of the application is determined
- **Business study:** The functional and informational requirements are identified and then the business value of the application is determined
- **Functional model iteration:** The incremental approach is adopted for development
- **Design and build iteration:** If possible design and build activities can be carried out in parallel
- **Implementation:** The software increment is placed in the working environment

Dynamic System Development Methods

- Provides a framework for building and maintaining systems



Feature driven Development(FDD)



In FDD, the feature means client valued function.

It is practical process model for **object oriented software engineering**

Feature driven Development(FDD)

1. Develop overall model

- The high-level walkthrough of scope and detailed domain walkthrough are conducted to create overall models.

2. Build feature list

- List of features is created and expressed in the following form

<action> the **<result>** **<by for of to>** a(n) **<object>**
For Ex. "Display product-specifications of the product"

3. Plan by feature

- After completing the feature list the development plan is created

Design by feature

- For each feature the sequence diagram is created

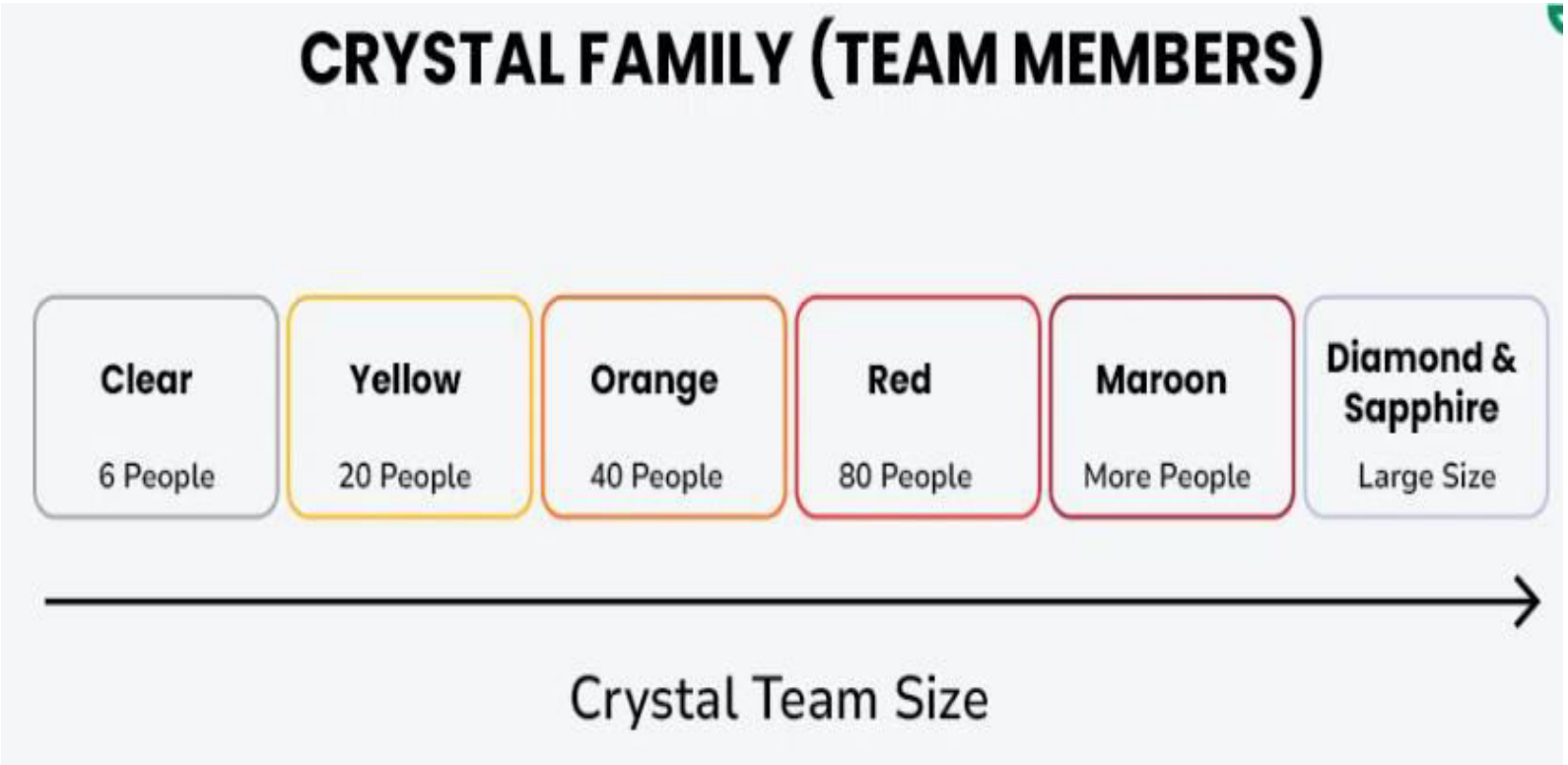
Build by feature

- Finally the class owner develop the actual code for their classes

Crystal

- Crystal methods are flexible approaches used in Agile software development to **manage projects** effectively.
- They adapt to the needs of the team and the project, promoting collaboration, communication, and adaptability for successful outcomes.
- The methods are **color-coded** to significant risk to human life. It is mainly for short-term projects by a team of developers working out of a single workspace

Crystal


CRYSTAL FAMILY (TEAM MEMBERS)
Clear
6 People
Yellow
20 People
Orange
40 People
Red
80 People
Maroon
More People
Diamond & Sapphire
Large Size
Crystal Team Size

Agile Modeling

- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.
- -The project scope and requirements are laid down at the beginning of the development process.
- -Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
- -Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements

Agile Modeling

Fig. Agile Model

PPT Content Resources Reference :

1. **Book Reference**

   Software Engineering: A Practitioner's Approach, by R.S.Pressman published by TMH.

2. **Reference Books:**

3. Software Engineering, 8th Edition by Sommerville, Pearson.
4. Software Engineering 3rd Edition by Rajiv Mall, PHI.
5. An Integrated Approach to Software Engineering by Pankaj Jalote Wiley India, 2009.

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/