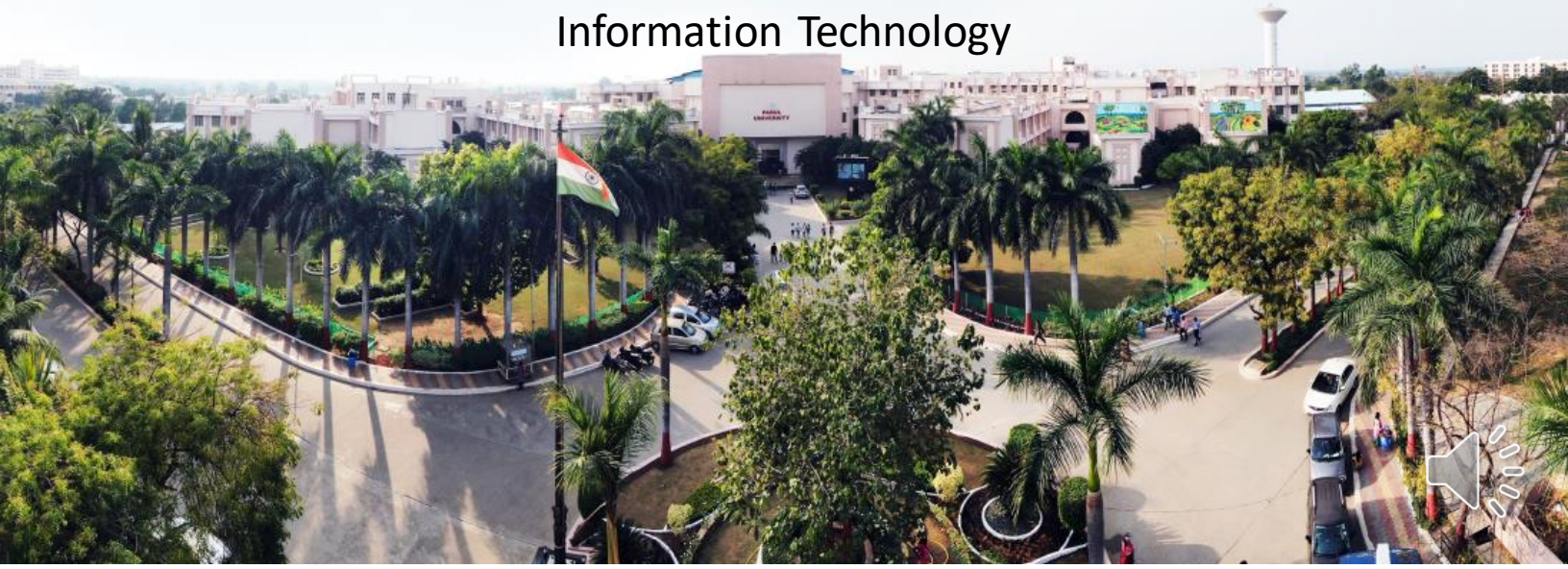
Parul® University

Software Engineering (03105401)

PIET/PIT CSE-IT Team

Computer Science & Engineering

Information Technology



Text Books

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

Reference Books:

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

CHAPTER-1

Introduction

- Study of Different Models, Software Characteristics, Components, Applications, Layered Technologies, Processes, Methods and Tools, Generic View Of Software Engineering
- Process Models- Waterfall model, Incremental, Evolutionary process models- Prototype, Spiral And Concurrent Development Model

Ariane 5 - One bug, one crash

•It took the European Space Agency 10 years and **$7 billion** to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

•Issue was a small **computer program** trying to **stuff a 64-bit** number into a **16-bit space**.

Image source : Google

One bug, one crash.
Of all the careless lines of code recorded
in the history of computer science

Y2K bug (millennium bug)

- The Y2K bug was a computer flaw, or bug, that may have caused problems when **dealing with dates** beyond December 31, 1999.
- The flaw, faced by computer programmers and users all over the world on January 1, 2000, is also known as the **"millennium bug..**
- Computer engineers used a ***two-digit code for the year***. Instead of a date reading 1970, it read 70.

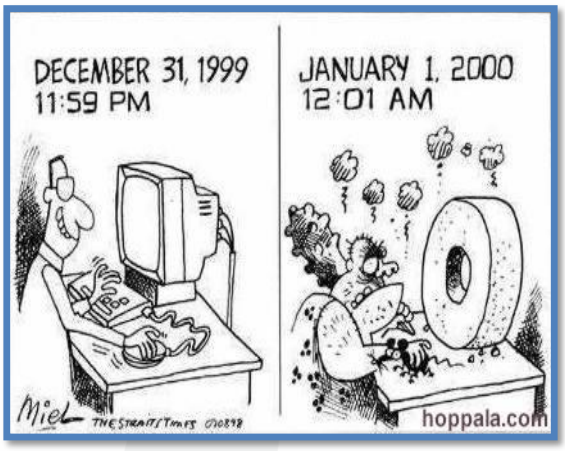


Image source : Google

As the year 2000 approached,
computer programmers realized that computers might not interpret **00** as **2000**, but as **1900**

SOFTWARE ENGINEERING

Definition of Engineering

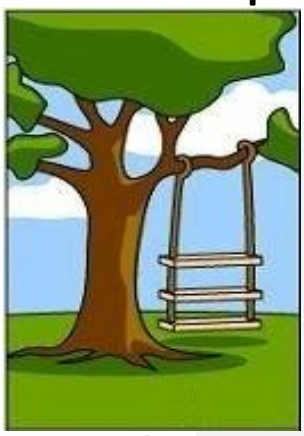
- Application of science, tools and methods to find cost effective solution to problems

Definition of Software Engineering

- Software Engineering is defined as systematic, disciplined and quantifiable approach for the development, operation and maintenance of software

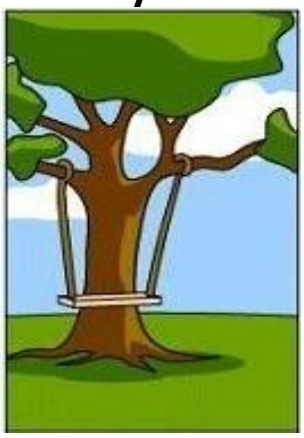
Why to Study Software Engineering?

- Software Development Life Cycle without Software Engineering



1

How the Customer Explains Requirement

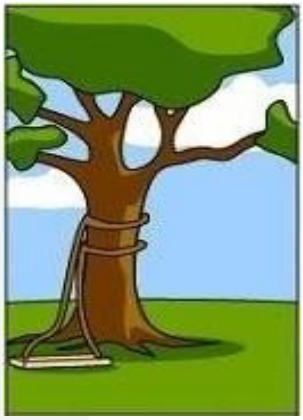


2

How the Project Leader understand it

3

How the System Analyst design it



4

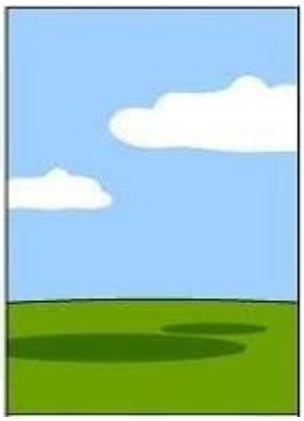
How the Programmer Works on it

Why to Study Software Engineering?

- Software Development Life Cycle without Software Engineering

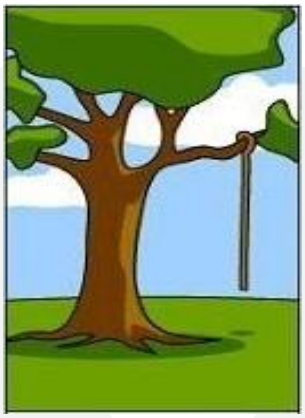
5

How the Business Consultant describe it



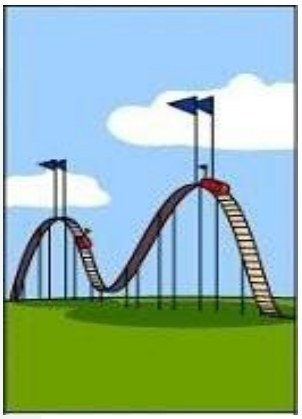
6

How the Project was documented



7

What Operations Installed

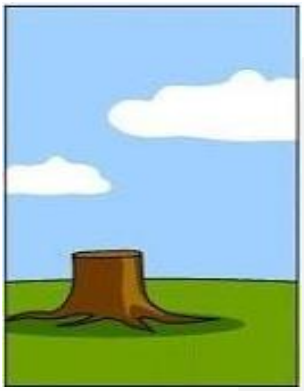


8

How the Customer was billed

Why to Study Software Engineering?

- **Software Development Life Cycle without Software Engineering**



9

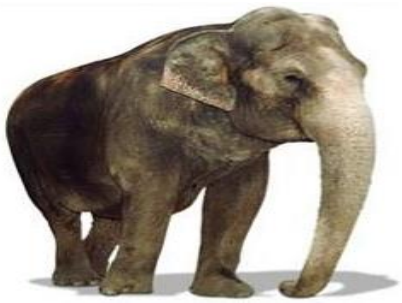
How it was supported

10

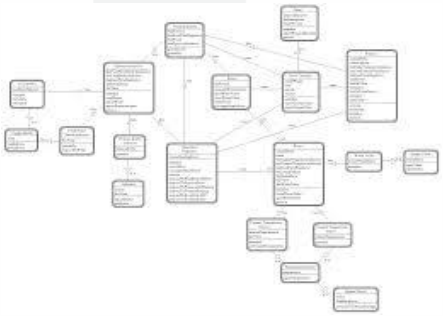
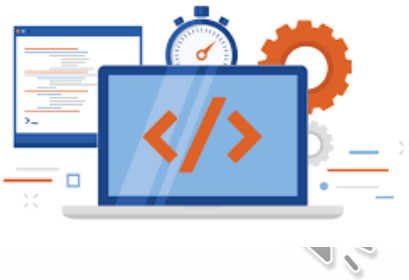
What the customer really needed

Software development **Process** needs to be **engineered** to avoid the **communication gape** & to **meet the actual requirement** of customer within stipulated budget & time

SDLC without Software Engineering

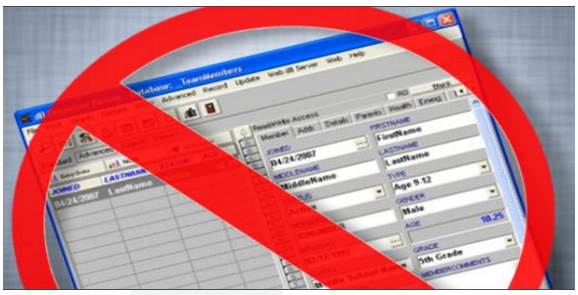
Customer Requirement	Solution
• Have one trunk • Have four legs • Should carry load both passenger & cargo • Black in color • Should be herbivorous	• Have one trunk • Have four legs • Should carry load both passenger & cargo • Black in color • Should be herbivorous
	 Our value added Also gives milk

SOFTWARE DESIGNED AS

Engineering			
	Design	Build	Product
Software Engineering			

Software is dead…..!

- The **old School view of Software**
  - You buy it
  - You own it &
  - It's your job to manage it
  - That is coming to an end



- Because of **web 2.0** & extensive **computing power**, there is a different generation of software
  - It is delivered via Internet
  - It looks exactly like it's residing on each user's computing device
  - Actually it reside on far away server

Image source : Google

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

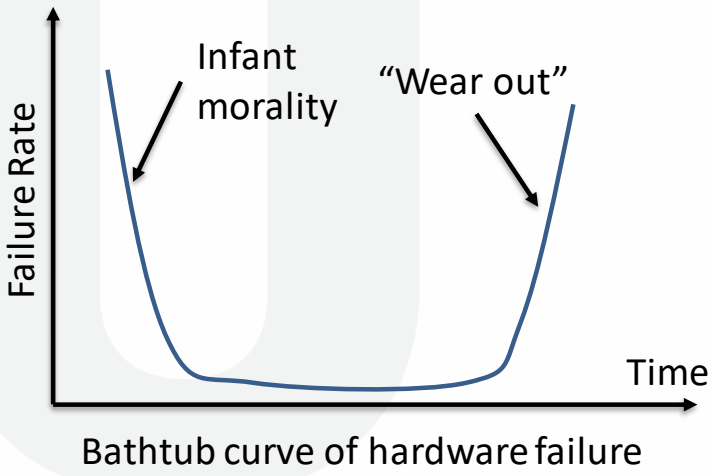
Computer Program

Data Structure

Documents Soft & Hard

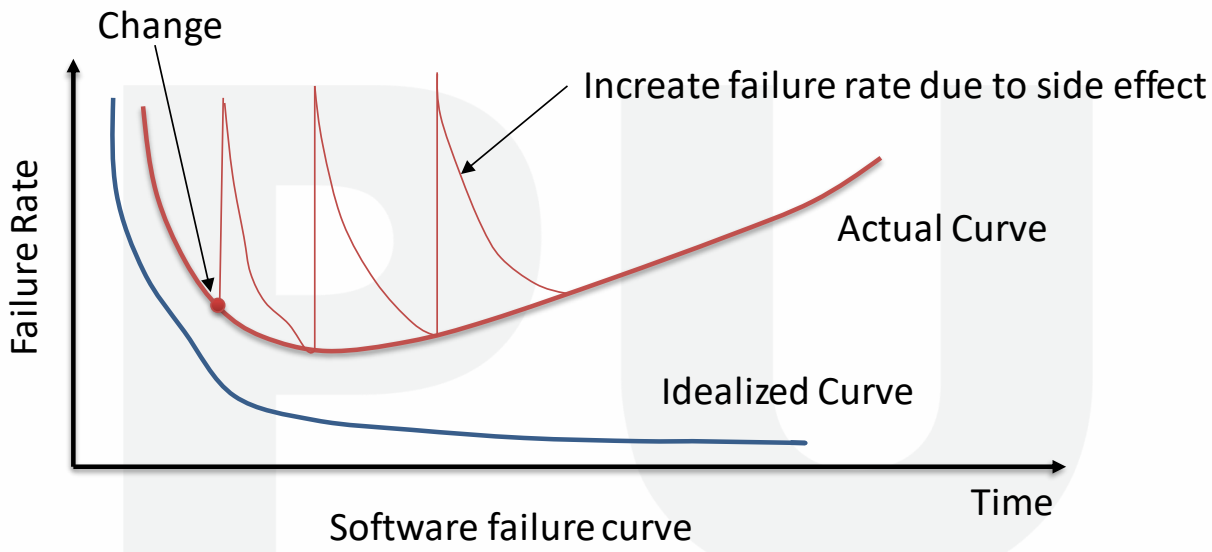
Characteristics of Software

- **Software is developed or engineered**
  - It is not manufactured like hardware
    - Manufacturing phase can introduce quality problem that are nonexistent (or easily corrected) for software
    - Both requires construction of "product" but approaches are different
- **Software doesn't "wear-out"**



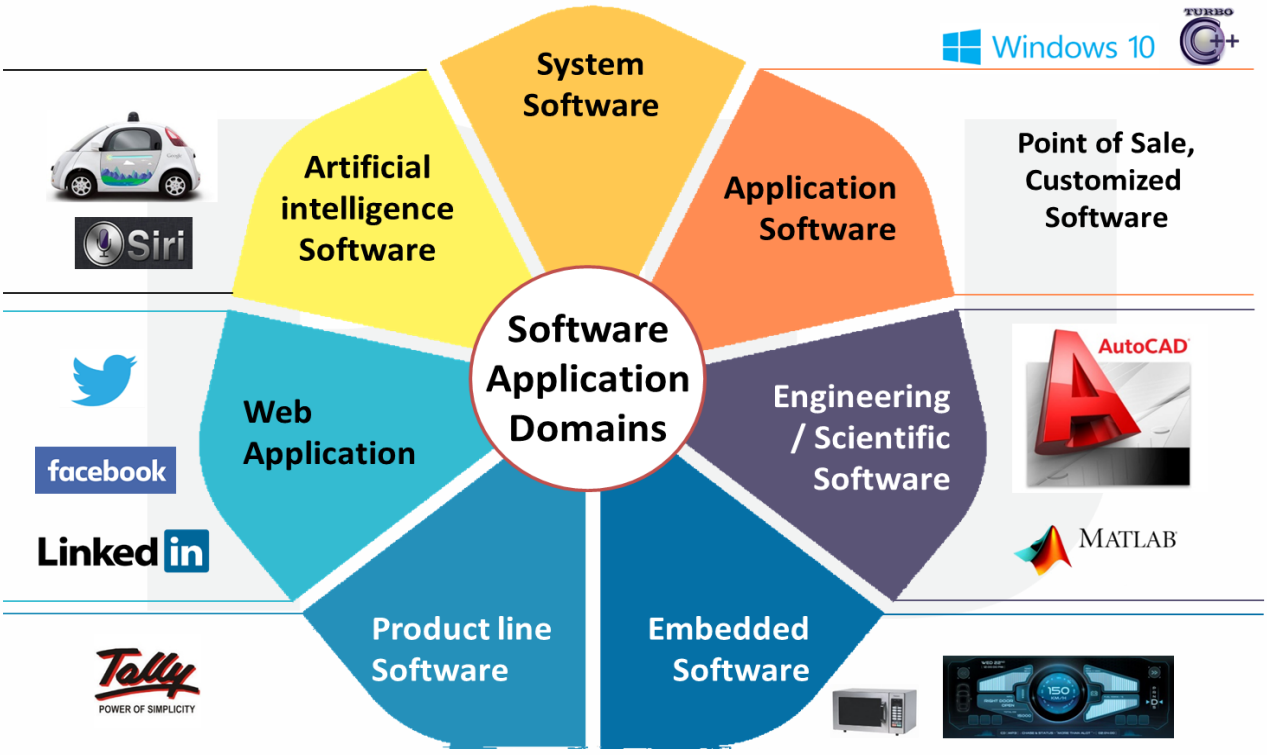
Bathtub curve of hardware failure

Characteristics of Software



- Although the industry is **moving toward component based construction, most software** continues to be **custom built**

Software Application Domains



THE CHANGING NATURE OF SOFTWARE

Seven Broad Categories of software are challenges for software engineers:-

- System software
- Application software
- Engineering and scientific software
- Embedded software
- Product-line software
- Web-applications
- Artificial intelligence software

Challenges faced by software engineers

- What “sound engineering principles” can be applied to computer software development?
- How do we “economically” build software so that it is “reliable”?
- What is required to create computer programs that work “efficiently” on not one but many different “real machines”?

PROCESS

- **What is it?**
- A series of predictable steps
- A road map that helps you create for timely, high-quality result.
- A software process is a framework for the tasks that are required to build high-quality software
- **Who does it?**
- Software engineers, project managers and customers
- **Why is it important?**
- It provides stability, control, and organization to an activity that can, if left uncontrolled, become quite chaotic.

PROCESS

- **What are the steps?**
- The process that you adopt depends on the software you're building.
- **What is the work product?**
- The work products are the programs, documents, and data produced as a consequence of the
- **How do I ensure that i've done it right?**
- The quality, timeliness, and long-term viability of the product you build are the best indicators of the efficacy of the process that you use.
- A number of software process assessment mechanisms enable organizations to determine the "maturity" of a software process.

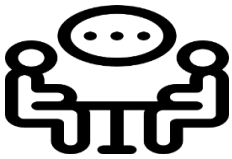
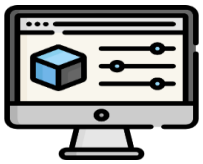
Software Process

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

A PROCESS FRAMEWORK

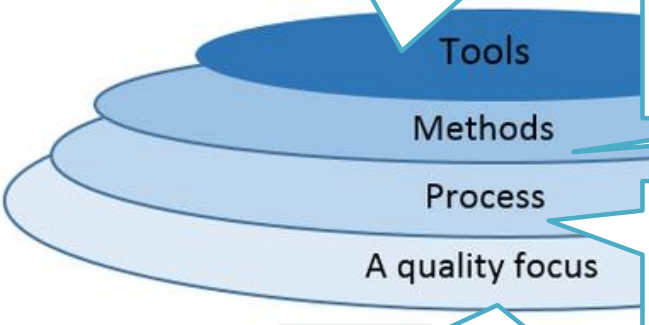
- **Used as a basis for the description of process models**
- **Generic process activities**
  - Communication
  - Planning
  - Modeling
  - Construction
  - Deployment

Process Framework 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		

Software Engineering – Layered Technology

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 Technology

- **Quality focus**
  - Bedrock that supports software Engineering.
  - Degree of Goodness
  - Maintainability
  - Correctness
  - Usability
- **Process**
  - Foundation for software Engineering
  - "What to do"
  - Deals with activities , actions & task
  - Comes out with "How" to questions.

Image source : Google

Software Engineering – Layered Technology

- **Methods**
  - Provide technical How-to's for building software
  - Deals with "How to" Implement
  - Communication
  - Requirement & design Modelling Analysis
  - Programming construction
  - Testing & Support
- **Tools**
  - Helping hand of process
  - Automated Support
  - Used for code, design, test & sell

Image source : Google

Software Engineering Cont.

- Software Engineering is a layered technology
- **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
  - It **defines** a **framework** with activities for effective delivery of software engineering technology

Software Engineering 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**
  - **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.

A 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.
  - *Definition phase*
  - *Development phase*
  - *Support phase*

A generic View of Software Engineering

- **Definition phase**
  - –The *definition phase focuses on "what"* the key requirements of the system and the software are identified.
- 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
- What validation criteria are required to define a successful system.
  - –Three major tasks will occur : system or information engineering, software project planning and requirements analysis

A generic View of Software Engineering

- **Development phase**

- The *development phase focuses on "how".*
  - How data are to be structured
  - How function is to be implemented within a software architecture,
  - How procedural details are to be implemented
  - How interfaces are to be characterized
  - How the design will be translated into a programming language and how testing will be performed.
- Three specific technical tasks will always occur in this phase: software design, code generation and software testing

A generic View of Software Engineering

- **Support phase**
- Focuses on *change*
- Reapplies the steps of the definition and development phases but does so in the context of existing software.
- Four types of change are encountered during the support phase:
  - **Correction**
    - Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software.
    - *Corrective maintenance changes* the software to correct defects.

A generic View of Software Engineering

– **Adaptation**

- Over time, the original environment (e.G., CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change.
- Adaptive maintenance results in modification to the software to accommodate changes to its external environment.

– **Enhancement**

- As software is used, the customer/user will recognize additional functions that will provide benefit.
- Perfective maintenance extends the software beyond its original functional requirements

A generic View of Software Engineering

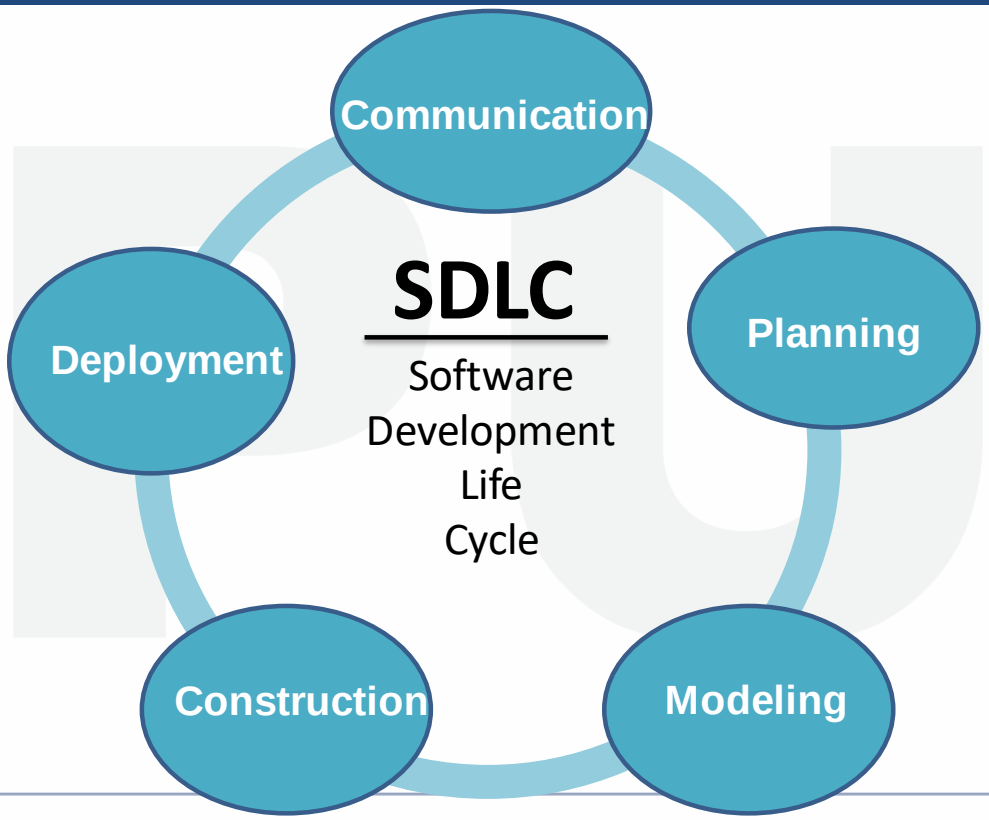
– **Prevention**

- Computer software deteriorates due to change, and because of this, preventive maintenance, often called software reengineering, must be conducted to enable the software to serve the needs of its end users.
- Preventive maintenance makes changes to computer programs so that they can be more easily corrected, adapted, and enhanced.

Software Process Models

- The **process model** is the abstract representation of process.
- 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.

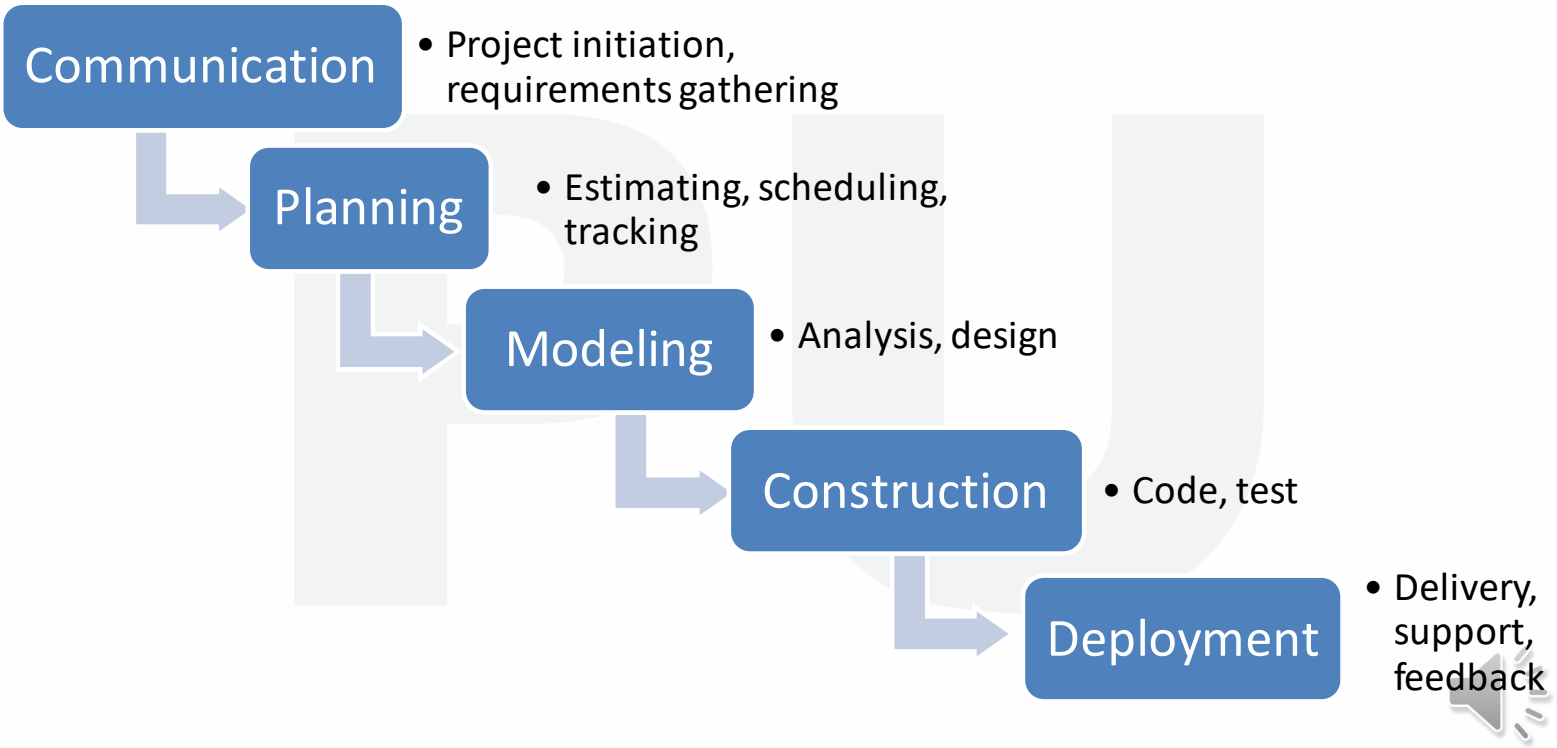
SDLC Phases


Communication
Planning
Modeling
Construction
Deployment
SDLC
Software
Development
Life
Cycle

PROCESS MODELS

- Linear Sequential Waterfall model
- Prototype Model
- RAD Model
- Evolutionary Software Process Model
  - Incremental Model
  - Spiral Model
  - Concurrent Development Process Model
  - Component based Assembly Model
- The Formal Methods Model
- Fourth Generation Techniques

The Waterfall Model



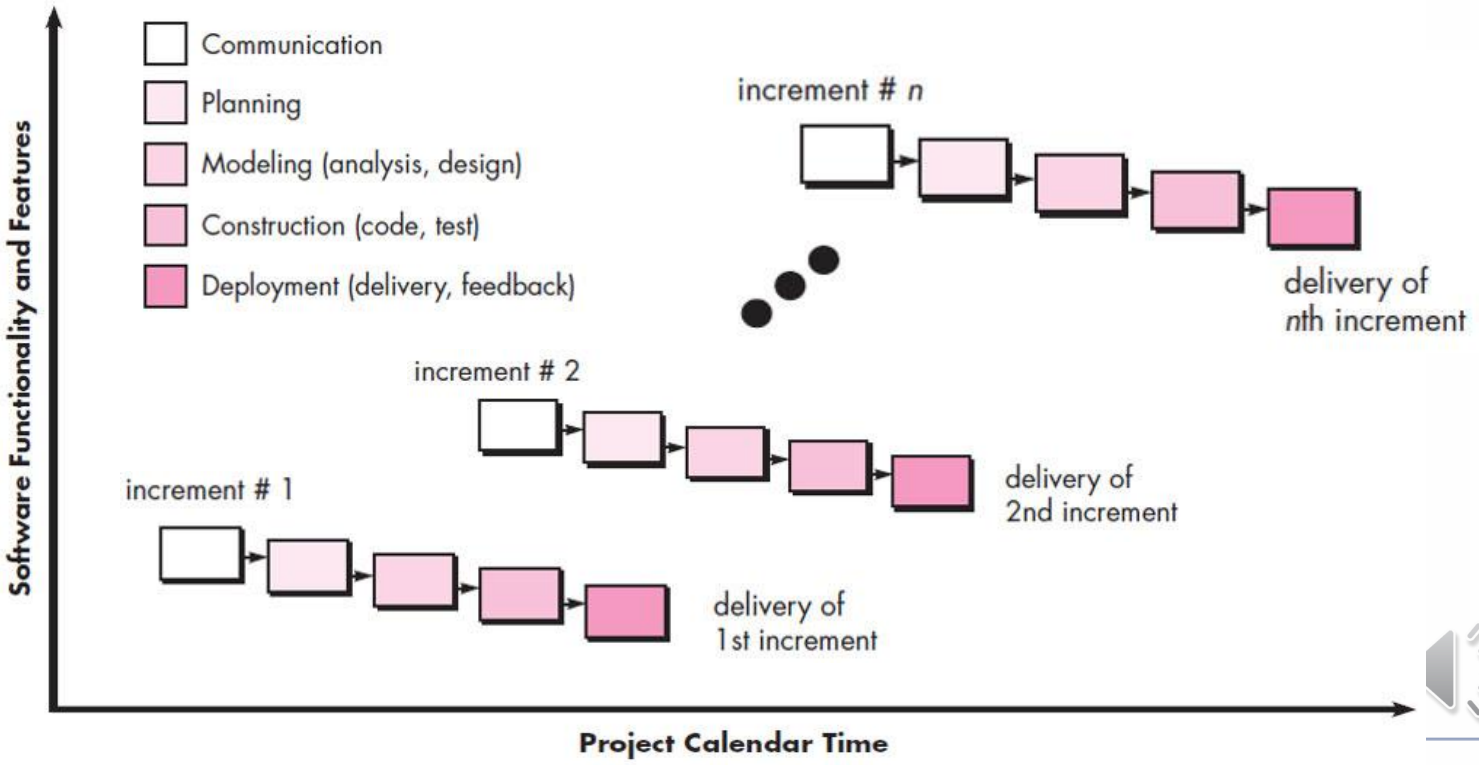
The Waterfall Model

- When **requirements** for a problems are **well understood** then this model is used in which **work flow** from communication to deployment is **linear**
- This Model also called as the **Classic life cycle** or **linear sequential 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

The Waterfall Model Cont…

- **Advantages**
- Simple & Easy
- Easy to manage
- Works well for smaller projects
- Results are well documented
- **Drawbacks**
  - Risk & Uncertainty
  - Not for projects where requirements are changing
  - **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.
  - Not for big & complex projects

Incremental Process Model



Incremental Process Model cont.

- 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.
- For example, word-processing software developed using the incremental 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.

Incremental Process Model cont.

- **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.

THE PROTOTYPING MODEL

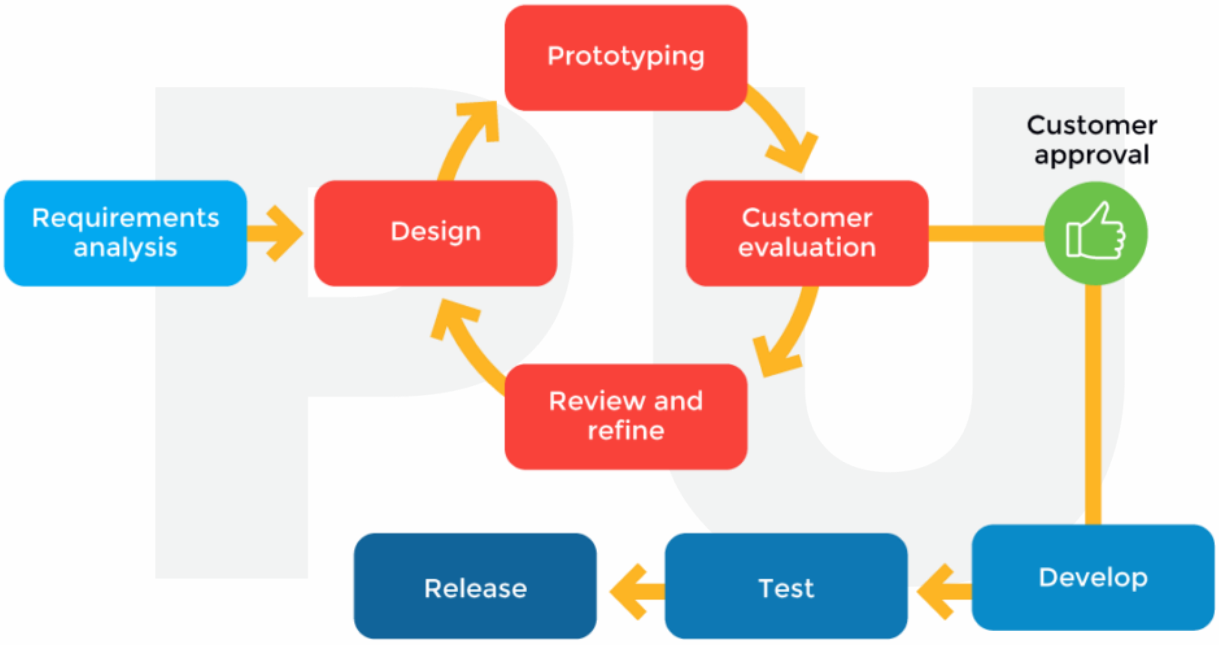


Image source : Google

Prototyping model cont.

- **It works as follow**
  - **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**
- **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, etc.

Prototyping model cont.

- **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**

Rapid Application Development (RAD) Model

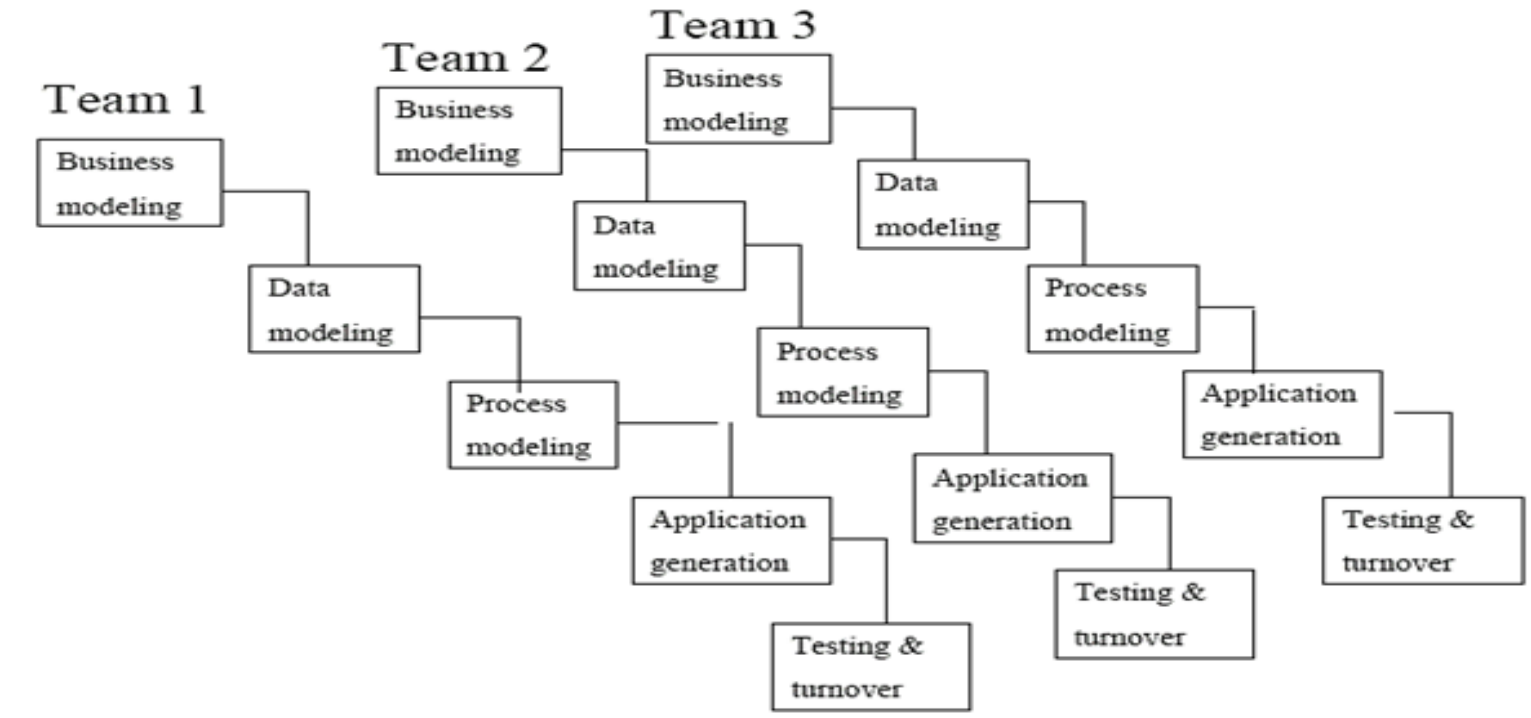


Image source : Google

Rapid Application Development (RAD) Model

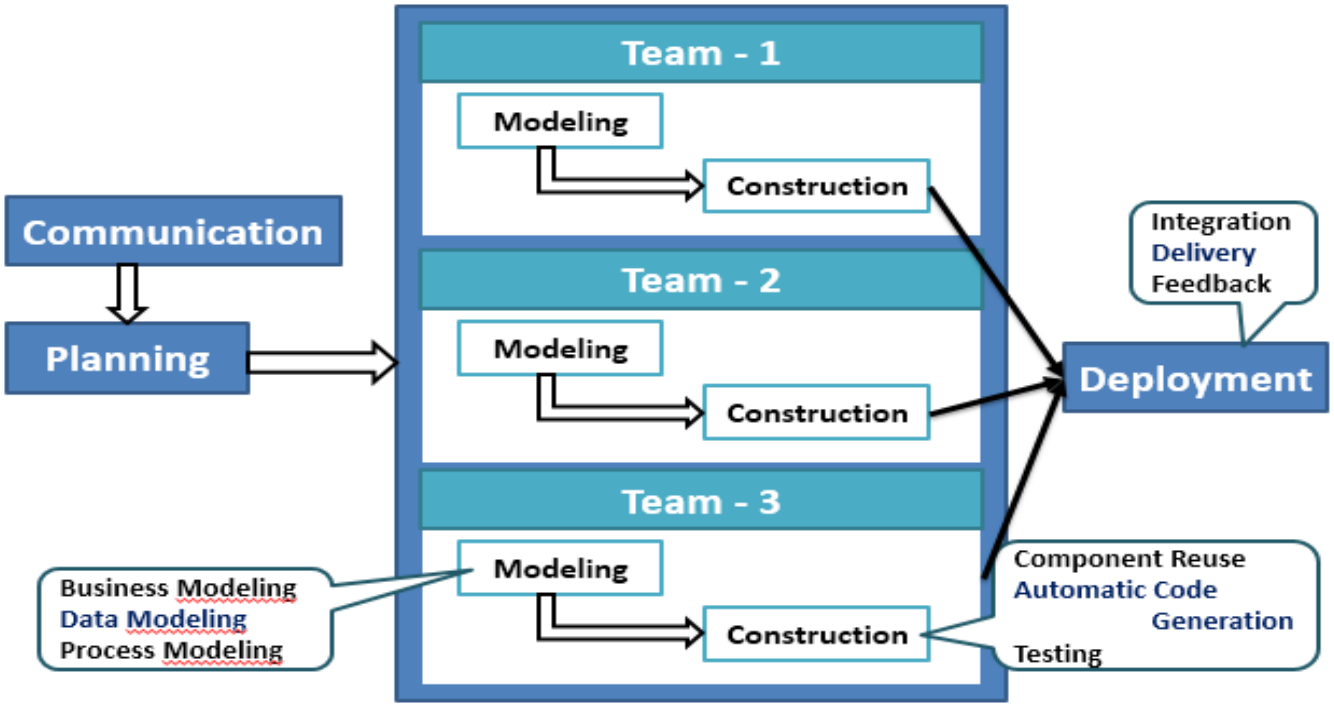


Image source : Google

Rapid Application Development (RAD) Model

- It is also know as **RAD** Model
- It is a type of **incremental model** in which; **components** or functions are **developed in parallel**.
- Rapid development is **achieved** by **component based construction**
- This can **quickly give** the customer **something to see** and use and to provide feedback.
- **Communication**
  - This phase is used to understand business problem.
- **Planning**
  - Multiple software teams work in parallel on different systems/modules.

Rapid Application Development (RAD) Model

- **Modeling**
  - **Business Modeling:** *Information flow* among the business.
    - Ex. What kind of information drives (moves)?
    - Who is going to generate information?
    - From where information comes and goes?
  - **Data Modeling:** Information refine into set of *data objects* that are *needed* to support business.
  - **Process Modeling:** *Data object* transforms **to** *information flow* necessary to implement business.
- **Construction**
  - It highlighting the *use of pre-existing software component*.
- **Deployment**
  - Deliver to customer basis on subsequent iteration.

Rapid Application Development (RAD) Model

- **When to Use ?**
  - There is a need to create a system that can be modularized in 2-3 months of time.
  - High availability of **designers** and budget for modeling along with the cost of automated code generating tools.
  - Resources with high business knowledge are available.
- **Advantages**
  - Reduced development time.
  - Increases reusability of components.
  - Quick initial reviews occur.
  - Encourages customer feedback.
  - Integration from very beginning solves a lot of integration issues.

Rapid Application Development (RAD) Model

- **Drawback**
  - For **large** but scalable **projects**, RAD **requires sufficient human resources**.
  - Projects **fail if developers** and **customers** are **not committed** in a much shortened time-frame.
  - **Problematic** if system **can not be modularized**.
  - **Not appropriate when technical risks are high** (heavy use of new technology).

Evolutionary Process Models

- 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**.
- Evolutionary models are
  - **Prototype Model**
  - **Spiral Model**
  - **Concurrent Development Model**

The Spiral Model

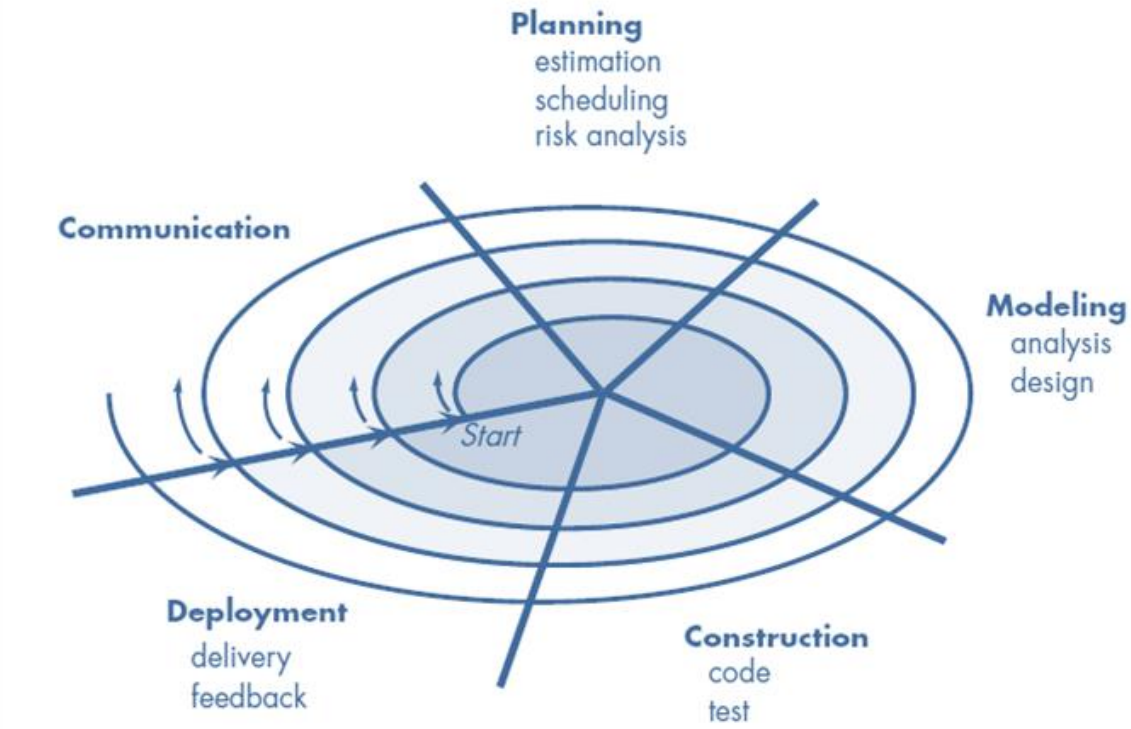


Image source : Google

The Spiral Model

- The Spiral model is an **evolutionary process model** that couples **iterative nature of prototyping** with the **controlled and systematic aspects of waterfall** 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

The 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.

The Spiral Model

- **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.

The Concurrent Development Model

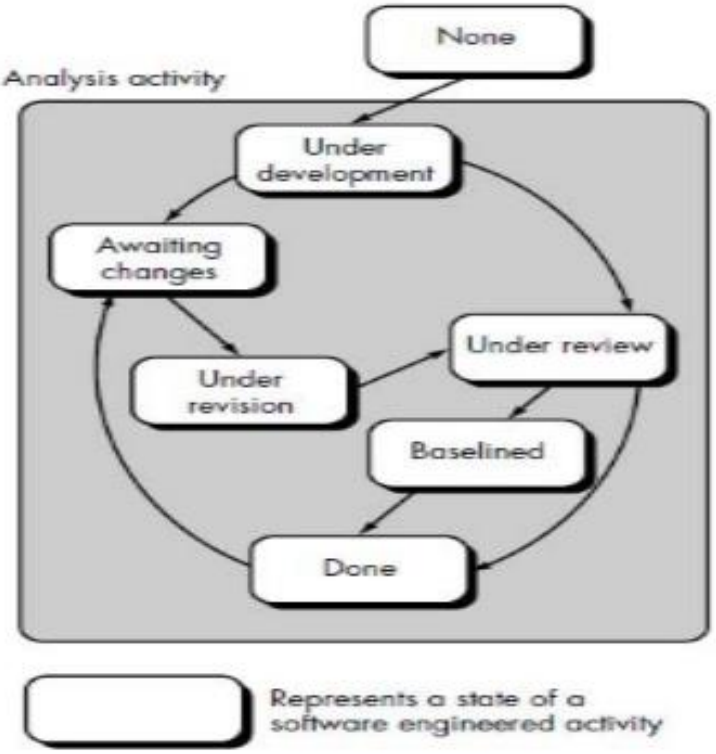
- The concurrent process model can be represented schematically as a series of major technical activities, tasks, and their associated states.
- All activities exist concurrently but reside in different states.
- The concurrent process model defines a series of events that will trigger transitions from state to state for each of the software engineering activities.
- **Example :**

Iteration	Activity	State
First	Customer Communication	Completed
	Analysis	None
Second	Analysis	Under Development
Third	Analysis	Awaiting changes

Image source : Google

The Concurrent Development Model

One element of the concurrent process model



Represents a state of a software engineered activity

REFERENCES

[1] Software Engineering (TextBook) R. Pressmen.
[2] Software Engineering ,Sommervilel
[3] Software Engineering, Rajiv Mall
[4] Software Engineering, PankajJalote
[5]https://www.geeksforgeeks.org/software-engineering/
[6]https://www.wisdomjobs.com/e-university/software-engineering-tutorial-338.html
[7]https://tutorialsinhand.com/tutorials/software-engineering-tutorial/software-engineering-introduction/software-engineering-home.aspx
Video Link:
[8] Youtube Channel - Ankit Chouhan
https://www.youtube.com/playlist?list=PLkKeulaggbtjT3Uf5pdosZ5req6dmX53Y

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in