# Introduction to Software Engineering

Courtesy:

Roger Pressman, Ian Sommerville &

Prof Rajib Mall

## What is Software Engineering?

- Software engineering encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.
- A systematic collection of good program development practices and techniques.
- An ***engineering approach*** to develop software.
- Systematic collection of past experience:
  - Techniques,
  - Methodologies,
  - Guidelines.

## Definition:

***IEEE Definition:***

"Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software."

***Other definitions:***

"Software engineering is the systematic application of engineering approaches to the development of software."

"Software engineering is the systematic, disciplined and cost-effective approaches to the development of software."

## Definition:

- is a field of computer science dealing with software systems large and complex, often undergo changes

- is multi-person construction, multi-version software (Parnas 1978)

- is an application of a systematic, disciplined, quantifiable approach to the development, operation, maintenance of software (IEEE 1990)

- SE is a discipline whose aim is the production of fault-free software, delivered on time & within budget, that satisfies the user’s needs and that is easy to modify.

## Frequently asked questions about software engineering:

| Question | Answer |
|---|---|
| What is software? | Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market. |
| What are the attributes of good software? | Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable. |
| What is software engineering? | Software engineering is an engineering discipline that is concerned with all aspects of software production. |
| What are the fundamental software engineering activities? | Software specification, software development, software validation and software evolution. |
| What is the difference between software engineering and computer science? | Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software. |
| What is the difference between software engineering and system engineering? | System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process. |

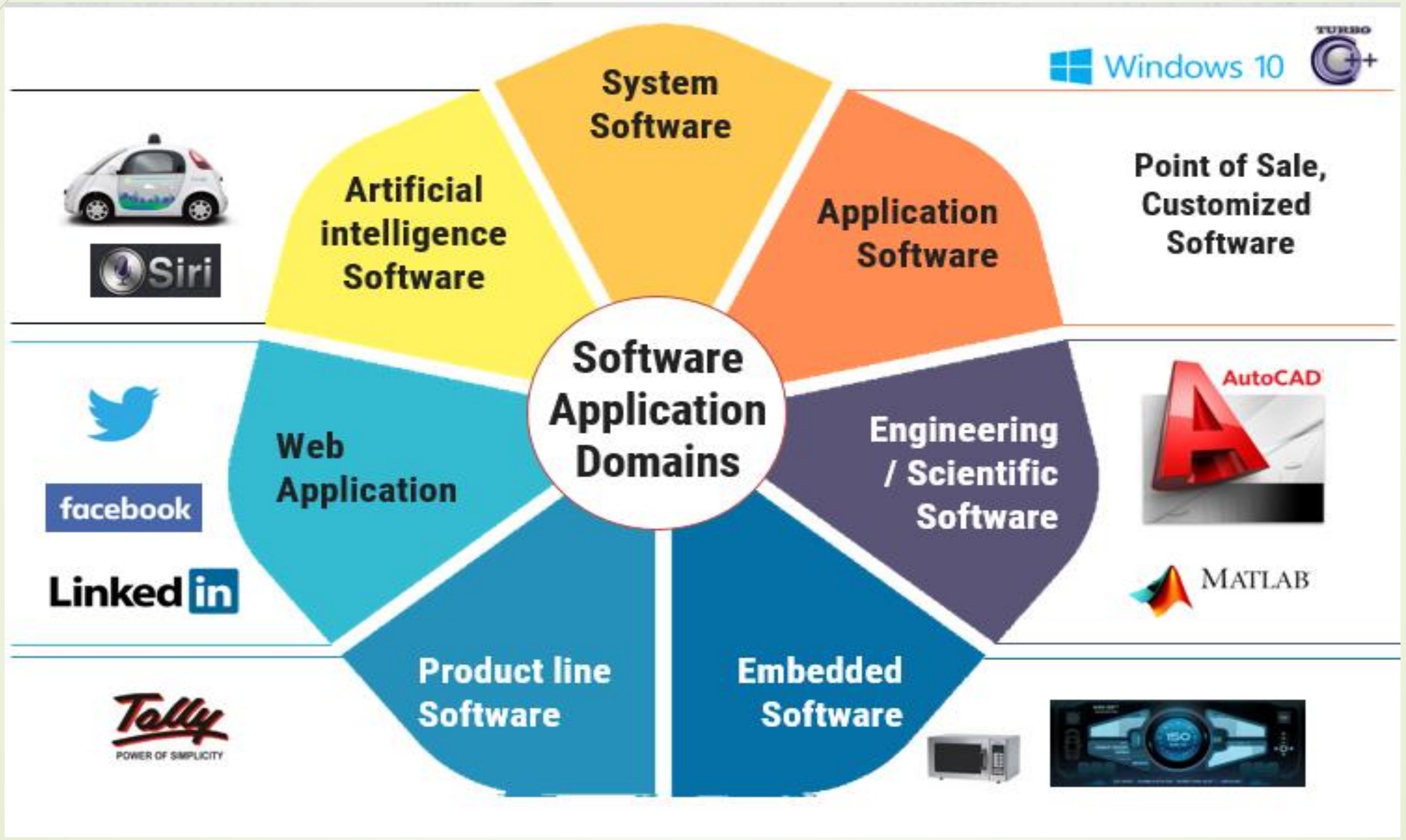
## Frequently asked questions about software engineering:

| Question | Answer |
|---|---|
| What are the key challenges facing software engineering? | Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software. |
| What are the costs of software engineering? | Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs. |
| What are the best software engineering techniques and methods? | While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another. |
| What differences has the web made to software engineering? | The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse. |

## Software Classification:

- **System Software** - OS, compilers, editors and drivers etc.
- **Networking and Web Applications Software** - www, HTML etc
- **Embedded Software** - microwaves, washing machines
- **Business Software** - inventory management, accounts, banking
- **Entertainment Software** - computer games, educational games, translation software
- **Artificial Intelligence Software** - decision support systems, pattern recognition
- **Scientific Software** - MATLAB, AUTOCAD
- **Utilities Software** - anti-virus software, voice recognition software.

## Software Classification:



## Essential attributes of good software

| Product characteristic | Description |
|---|---|
| Maintainability | Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment. |
| Dependability and security | Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system. |
| Efficiency | Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc. |
| Acceptability | Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use. |

## Successful Software System

- Software development projects have not always been successful
- When do we consider a software application successful?
  - ✓ Development completed
  - ✓ It is useful
  - ✓ It is usable
  - ✓ It is used
- Cost-effectiveness, maintainability implied

## Reason for failure

- Ad hoc software development results in such problems
  - No planning of development work
  - Poor understanding of user requirements
  - No control or review
  - Technical incompetence of developers
  - Poor understanding of cost and effort by both developer and user

## Software Development Myths:

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- Are believable because they often have elements of truth,

*but …*

- Invariably lead to bad decisions,

*therefore …*

- Insist on reality as you navigate your way through software engineering

## Management Myths:

- “ We already have a book of standards and procedures for building software. It does provide my people with everything they need to know …”
- “If my project is behind the schedule, I always can add more programmers to it and catch up …”
- “If I decide to outsource the software project to a third party, I can just relax: Let them build it, and I will just pocket my profits …”

## Customer Myths:

- “A general statement of objectives is sufficient to begin writing programs - we can fill in the details later …”
- “Project requirements continually change but this change can easily be accommodated because software is flexible …”

## Practitioner's Myths:

- "Let's start coding ASAP, because once we write the program and get it to work, our job is done …"
- "Until I get the program running, I have no way of assessing its quality …"
- "The only deliverable work product for a successful project is the working program …"
- "Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down…"

## Software Processes and Models

## Software Engineering:

- Software engineering encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high-quality computer software.

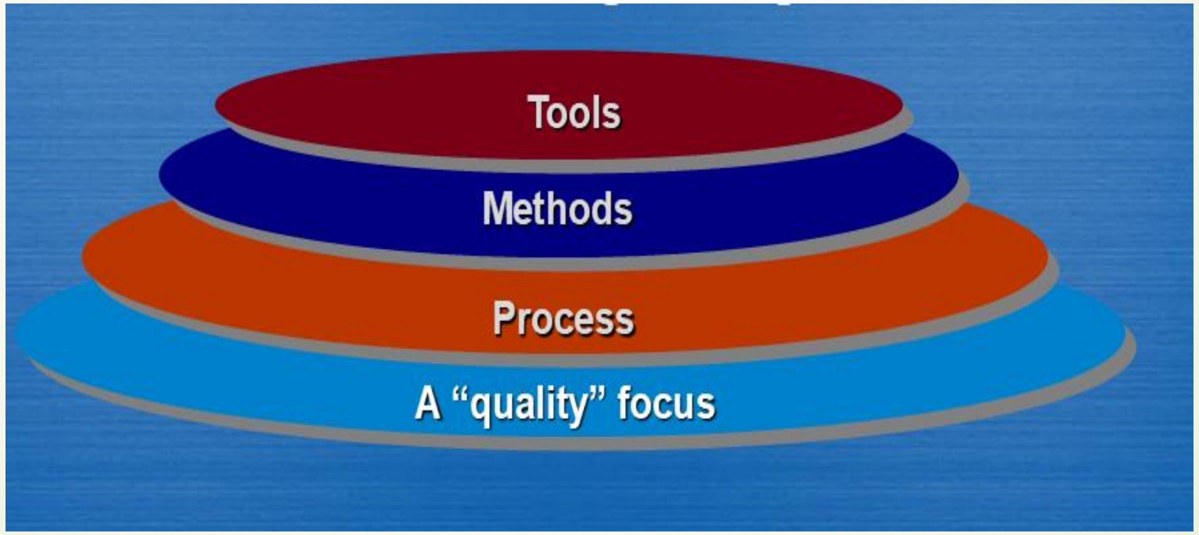


**Figure: Software Engineering Layers**

## Process:

- A software process is define as a framework for the task that are required to build high quality software.
- A process covers all activities starting from product inception through delivery and retirement.
- The Software process establishes context in which technical methods are applied, work products (models, documents, data, reports, etc) are produced, milestones are established, quality is ensured, and change is properly managed

## Software Processes:

- A set of activities whose goal is the development or evolution of software
- def: a framework of tasks required to build high-quality software.
- A software process defines an approach taken
  - while SE involves not only the approach but technology used.
  - thus, process is a glue that holds the technology layers.

## Methods:

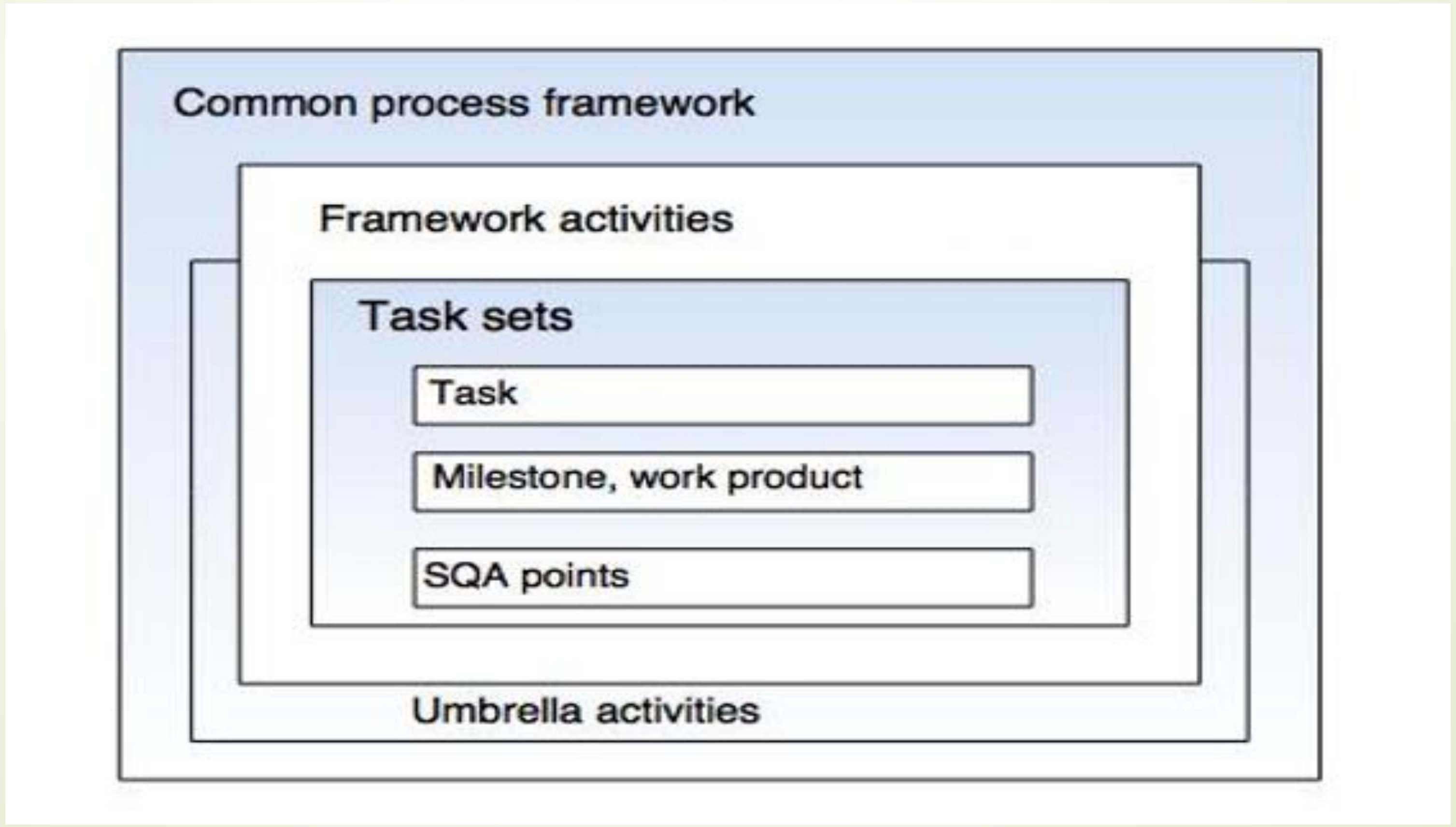
- Software engineering *methods* provide solution to the technical how's for building software.
- Methods encompass a broad array of tasks that include communication, requirements analysis, design modelling, program construction, testing, and support.
- Include modeling activities and other descriptive techniques.

## Tools:

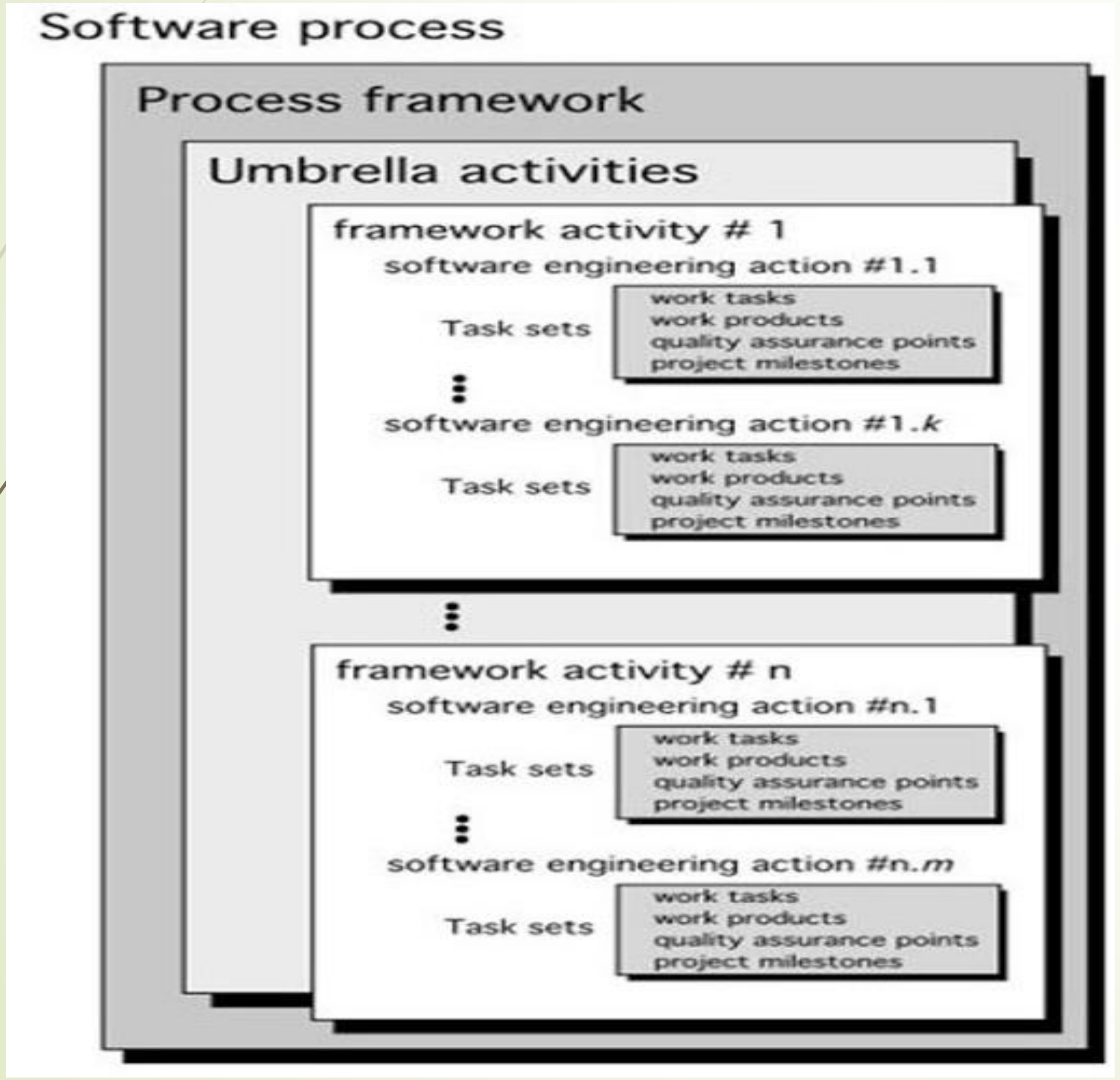
- Software engineering *tools* provide automated or semi-automated support for the process and the methods.
- Tools are integrated so that information created by one tool can be used by another.

## The Process Framework:

- A process is a collection of activities, actions, and tasks that are performed when some work product is to be created.



## The Process Framework:



- Each framework activity is populated by a set of software engineering actions.
- Each software engineering action is defined by a task set that identifies
    - the work **tasks** that are to be completed,
    - the **work products** that will be **produced**,
    - the **quality assurance points** that will be required, and
    - the **milestones** that will be used to indicate progress.

## The Process Framework:

- The process framework identify a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.
  - Communication
  - Planning
  - Modeling
  - Construction
  - Deployment
- For many software projects, framework activities are applied iteratively as project progresses.

## Umbrella Activities:

- The umbrella activities are applied throughout a software project and help software team to manage and control progress, quality, change, and risk.
  - Software project tracking and control
  - Risk management
  - Software quality assurance
  - Technical Reviews
  - Measurement
  - Software configuration management
  - Reusability management
  - Work product preparation and production
- Umbrella activities occur throughout the software process and focus primarily on project management, tacking and control.

## Task Set:

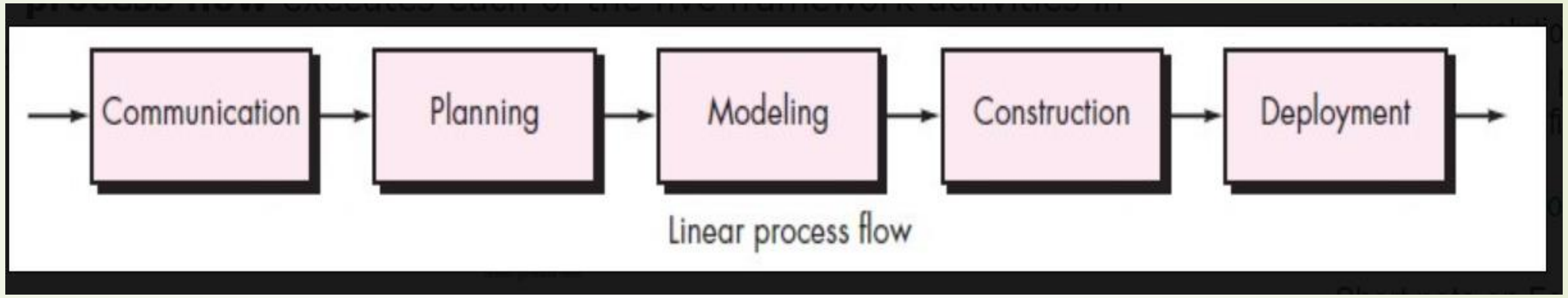
- A task set defines the actual work to be done to accomplish the objective of a software engineering action.
- Example: **Requirement gathering** is an important software engineering action that occurs during the **communication** activity.
- Different projects demand different task sets. **The software team chooses the task set based on problem and project characteristics.**
  - Single Stakeholder: *Phone conversation* is a way for requirement gathering.
  - Group of Stakeholders: Communication activity perform distinct actions like *inception, elicitation, elaboration, negotiation, specification, and validation.*

## Software quality: Quality attributes

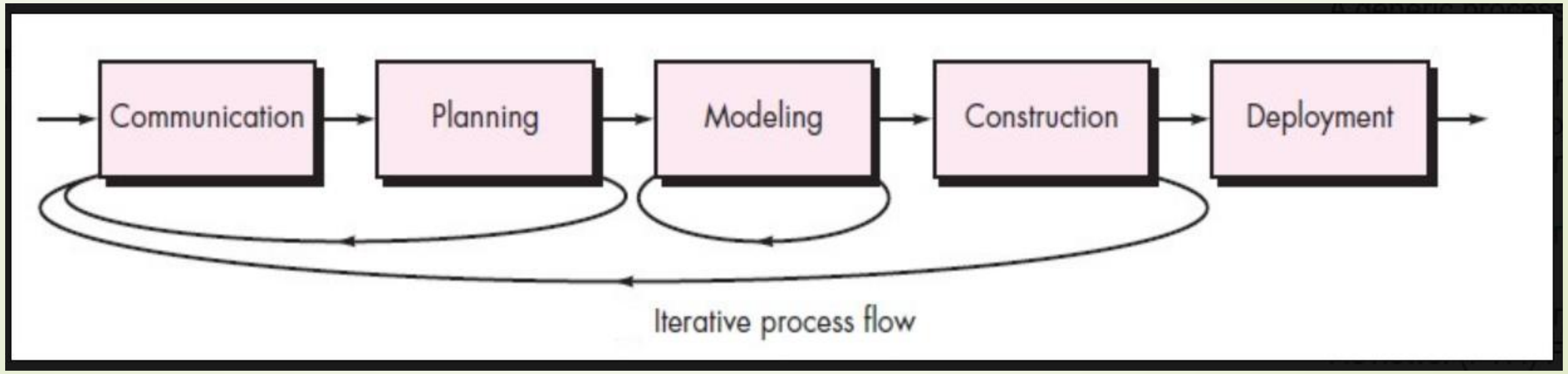
- **Reliability-** Reliability requirements deal with service failure.
- **Efficiency-** Deals with the hardware resources needed to perform the different functions of the software system.
- **Usability-** Deal with the staff resources needed to train a new employee.
- **Maintainability -** Bugs can be easily corrected, new tasks can be easily added to the product, functionalities of the product can be easily modified, etc.
- **Portability -** Adaptation of a software system to other environments.
- **Integrity -** Prevent access to unauthorized persons.
- **Correctness** – As per SRS

## Process Flow:

- **Linear Process Flow:** Executes activities in a sequence.

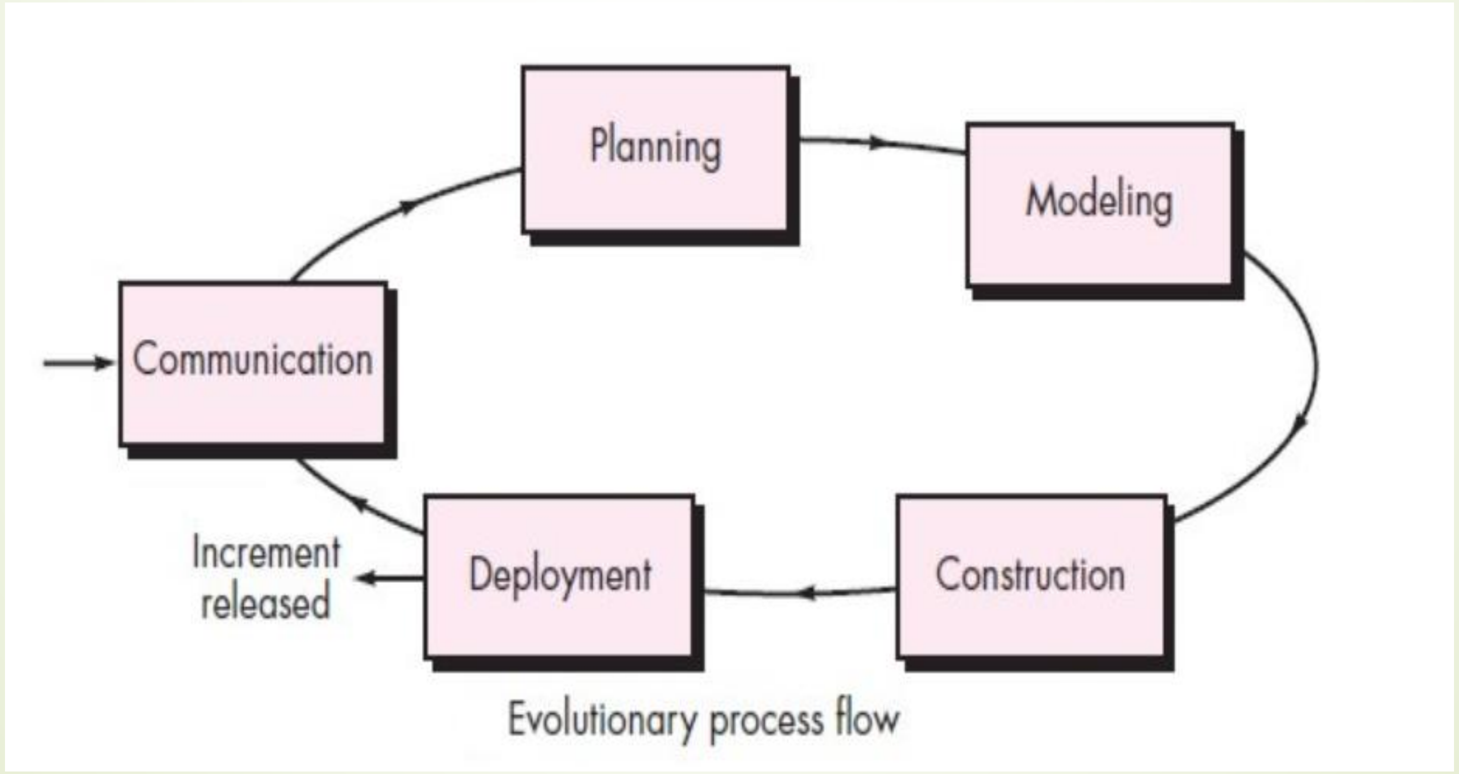


- **Iterative Process Flow:** Repeats one or more activities before proceeding further



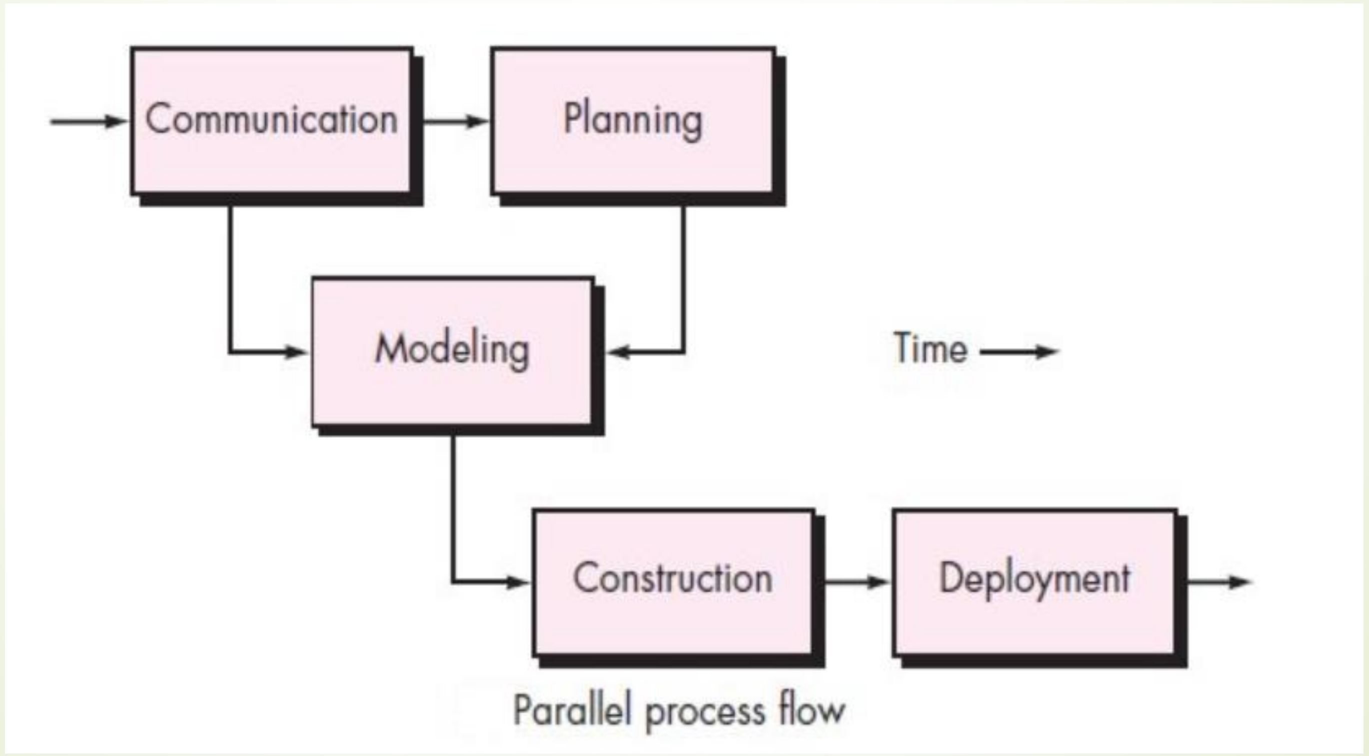
## Process Flow:

- **Evolutionary Process Flow:** Each circuit through the five activities leads to a more complete version of the software.



## Process Flow:

- **Parallel Process Flow:** Executes one or more activities in parallel with other activities.



## Software Process and Process Models

- A simplified representation of a software process
  - always presented from a specific perspective
  - defines a distinct set of activities, tasks, actions, milestones and work products
    - e.g specification, design, validation, evolution .
  - required to engineer high quality software with respect to the perspective
  - focuses on the construction tasks rather than the output products.

- Examples of process perspectives are
  - Workflow perspective  - sequence of activities
  - Data-flow perspective - information flow
  - Role/action perspective - who does what

## Software Process and Process Models

- Different organizations use different software processes. Why ?
  - different coding styles/rules/techniques
  - self-documenting or not
  - coding conventions/naming used or not.
  - iterative approach to design carried out or not.
  - modular approach followed or not
  - test data/cases prepared or not
  - suitable mechanism for effecting changes exists/not
  - software maintenance is done or not (e.g. academic environment)

## Why follow a software process ?

- Broadly, Software Development is an Intuitive Process
  - hence to serve as a template……
- Specifically, to exploit
  - Repeatability
  - Predictability
  - Improved quality through standardization
  - Continuous improvement
  - Training
- As a confidence building measure
- We tend to follow processes in everything!!

## Software Process and Models…

- Software Process
  - a set of partially ordered steps intended to reach a goal - in SE, the goal is to build a software product or enhance an existing one.
  - the systematic ways of developing and maintaining software systems
- Software Process Models
  - a software process model is an abstract representation of the software process
  - a software process is the implementation of a defined model.

## Software Process Models:

- A process model provides a specific roadmap for software engineering work and it defines the
  - flow of all activities, actions and tasks, the degree of iteration,
  - the work products, and the organization of the work that must be done.
- Attempt to organize the software life cycle by
  - defining activities involved in software production
  - order of activities and their relationships
- Goals of a software process
  - standardization, predictability, productivity, high product quality, ability to plan time and budget requirements

## Code and Fix:

- The earliest approach
  - Write code
  - Fix it to eliminate any errors that have been detected, to enhance existing functionality, or to add new features
- Source of difficulties and deficiencies
  - impossible to predict
  - impossible to manage

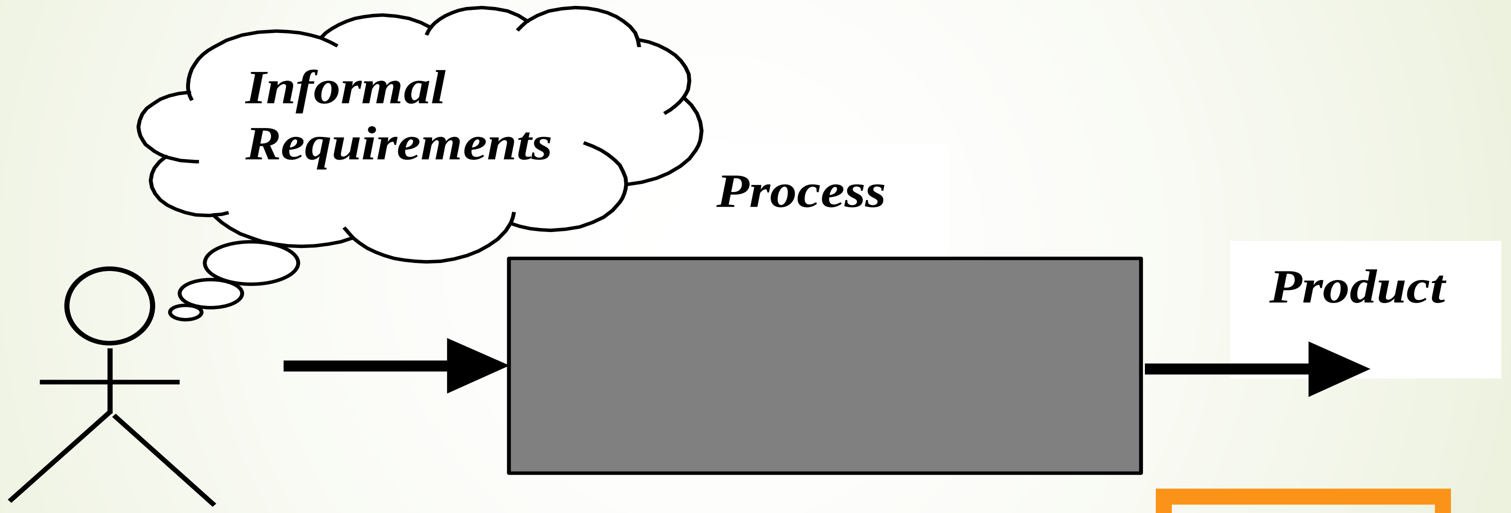
## Models are needed:

- Symptoms of inadequacy: the software crisis
  - scheduled time and cost exceeded
  - user expectations not met
  - poor quality
- The size and economic value of software applications required appropriate "process models"

## Process Model Goals (B. Boehm 1988):

- To determine the **order of stages** involved in software development and evolution, and to establish the **transition criteria** for progressing from one stage to the next.
- Transition criteria include completion criteria for the current stage plus choice criteria and entrance criteria for the next stage. Thus a process model addresses the following software project questions:
  - What shall we do next?
  - How long shall we continue to do it?"

## Process as a “black box”:



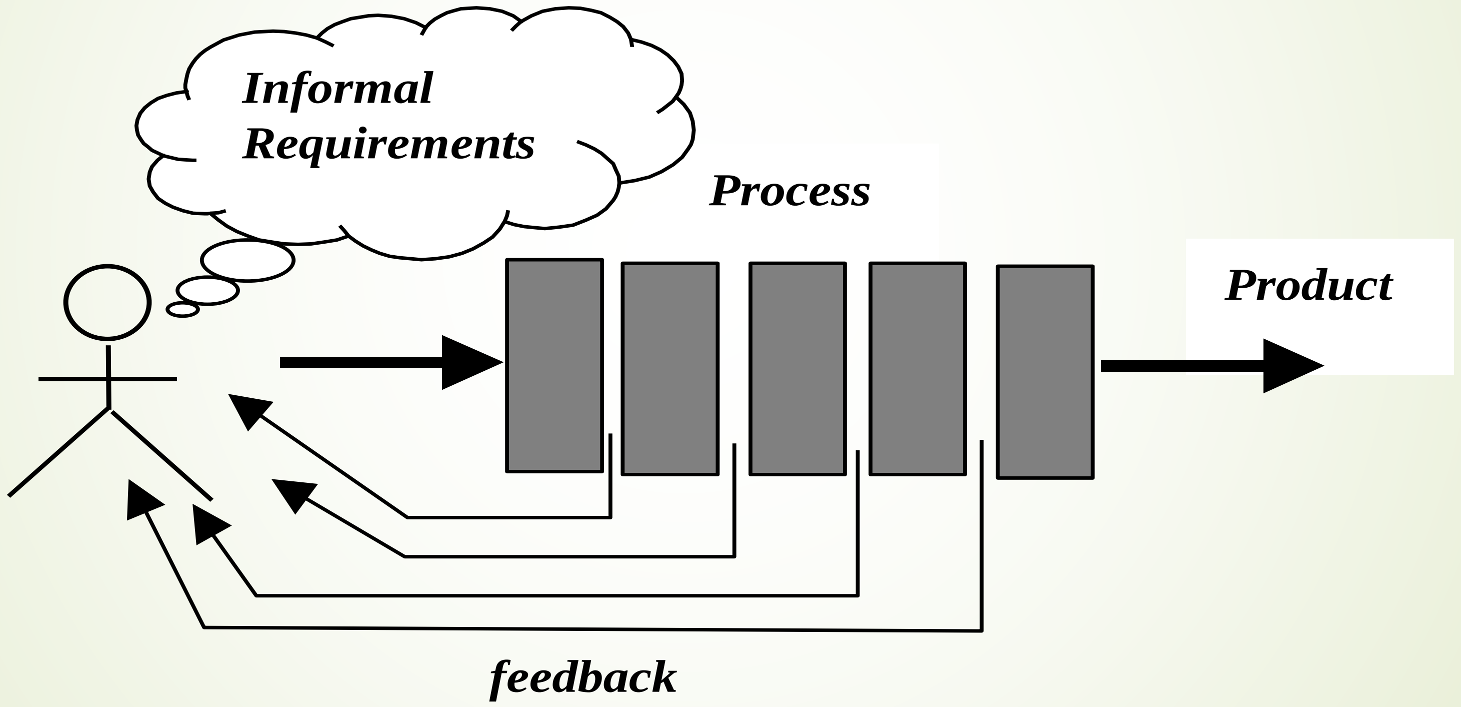
**Uncertain / Incomplete requirement In the beginning**

**Quality?**

## Problems:

- The assumption is that requirements can be fully understood prior to development
- Interaction with the customer occurs only at the beginning (requirements) and end (after delivery)
- Unfortunately the assumption almost never holds

## Process as a “white box”:



## Advantages

- Reduce risks by improving visibility
- Allow project changes as the project progresses
  - based on feedback from the customer

## The main activities of software production

- They must be performed independently of the model
- The model simply affects the flow among activities

## Characteristics of Good Model

- Should be precisely defined
  - No ambiguity about what is to be done, when, how etc..
- It must be predictable
  - Learn from feedback
  - Can be repeated in other projects with confidence about it's outcome
    - Project-A = done by 3 person in 4 months
    - Project-B = Similar in complexity should also take about same time or with some improvement it should take less time

## Software Process: By activities

- Requirements Identification
  - The problem to be solved
  - The features of the solution (functional and non-functional)
  - The business case (why solve the problem)
  - Acceptance criteria (a.k.a success criteria)
- Analysis
  - Understanding the domain (domain analysis)
  - Understanding the problem (problem analysis)
  - Understanding the solution (solution analysis)
- Architecture and Design
  - Overall structure
  - Detailed construction plan

$\boxed{\text{Phases}}$

## Software Process: By activities…

- Implementation
  - Building the system
  - Deployment
  - Putting the solution to work
- Maintenance
  - Keeping the system working and useful

$\left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\}$ Phases

## Software Process: by Timeline

- Inception Stage
  - Approximate vision, business case, scope, high-level estimates
- Elaboration Stage
  - Refined vision, iterative implementation of the core architecture, resolution of high risks, requirements identification, more realistic estimates
- Construction Stage
  - Iterative implementation of remaining features, preparation for deployment
- Transition Stage
  - Beta test, deployment

## Software Processes Categories

- Software processes may be broadly categorized as:
  - Structured (Waterfall, spiral, incremental, CMM, CMMI, RUPP)
  - Agile (XP, Scrum, Feature Driven Development)
- Questions to be answered
  - What is a structured process ?
  - What is an agile process ?

# Agile Vs Structured Process Model

| The Agile Process | The Structured/Planned Process |
|---|---|
| • Goal(s) - Comfort with change | • Goal(s) - Predictability and assurance to deal with mission criticality |
| • Tacit knowledge and communication | • Explicit, formal knowledge and communication |
| • Co-located, expert customer | • Contractual relationship with customer |
| • Small size skilled teams | • Large teams with a range of skills |
| • Dynamic team culture | • Comfort with stability |
| • Planning is a means to an end – but THERE IS LOTS OF PLANNING | • Process maturity and formal planning used to build trust |
| • Adds steps as needed | • Handles non-functional requirements better |
|  | • Subtracts steps if appropriate |

## Work Products

- What are they?
  - All the artifacts ("things") that need to be created for a software development project. Not necessarily only the software.
  - Each work-product captures a separate concern
- Where are they kept?
  - In a work-book
- Key concept – "Work-product orientation"
- Work-product orientation
  - Constraint-driven
    - Address the right "concern" at each time
  - Process is tailored to project

## Illustrative Work Products

- **Work Products for Requirements**
  - Problem statement
    - why – from a function point of view
  - Business case
    - why – from the business point of view
  - Storyboard
    - Narrative by users and other stakeholders
  - Use cases
    - <u>what</u> the system is supposed to do
  - Non-functional requirements
    - <u>how</u> the system is supposed to do it
  - Prioritized requirements
    - when
  - Acceptance plan
    - validation by the customer

# Illustrative Work Products…

- **Project management**
  - Intended development process
    - overview of how the process should be conducted
  - Configuration management plan
    - how the work-products should be managed
  - Resource plan
    - what resources – human as well as system – are available to the project
  - Project schedule
    - the timeline of the project
  - Release Plan
    - when features are given to the end-users
  - Iteration Plan
    - what will be done in each *development* cycle

## Illustrative Work Products…

- **Analysis**:
  - Guidelines
  - Scenarios, Sequence diagrams, Use case diagram, other UML diagrams
    - dynamic behavior
  - Class model and descriptions
    - static structure
- **User interface**
  - Guidelines
  - Screen flows
    - transition from screen to screen
  - Screen layouts
    - appearance

## Illustrative Work Products…

- **Design:**
  - Guidelines
  - System architecture
    - high-level view of how the system meets the functional and non-functional requirements
  - Target environment
    - a view of the System Architecture that shows what the system will be deployed on
  - Subsystem model
    - a view of the System Architecture that shows the partitioning of the system into smaller systems
  - Design object model and design class descriptions
    - detailed static structure of the system components
  - Design scenarios and sequence diagrams
    - detailed dynamic behavior of the system components
  - Design Alternatives

## Illustrative Work Products…

- **Implementation**
  - Guidelines
    - such as coding standards
  - Source code
- **Testing**
  - Guidelines
  - Unit test cases
  - System test cases

## Key Concepts: Structured Processes

- Scenario-driven
  - Discussion: What might be other drivers?

- Work-products and work-product orientation

- Management of risks
  - Incremental, Iterative
  - Scenario-driven – what might be others?
  - Separation of concerns:
    - Separable work-products. WHY?

## Key Concepts: Structured Processes…

- Traceability
  - Discussion: Why maintain a traceability chain?
    - Forward traceability and Reverse Traceability
- Verification
  - Are we developing the product right ? e.g. sorting
- Validation
  - Are we developing the right product ? e.g. sorting

Thank You.. ☺