Software Development

1. Process framework activities
2. Software myths
3. Prescriptive process models
4. Product and process
5. Difference b/w functional and non functional
6. Requirements Engineering
7. Agility principles
8. Software requirements specification

**1. Process framework Activities**

* Communication
* Planning
* Modeling
  + Analysis of requirements
  + Design
* Construction
  + Code generation
  + Testing
* Deployment
* **Communication:** communicate with customer to understand objectives and gather requirements
* **Planning**: creates a “map” that defines the work by describing tasks, risks and resources, work products and work schedule.
* **Modeling:** Create a “sketch”, what it looks like architecturally, how the essential parts fit together and other characteristics.
* **Construction**: code generation and the testing.
* **Deployment:** Delivered to the customer who evaluates the products & provides feedback based on the evaluation.
* These five framework activities can be used to all software development, regardless of the application domain, size of the project, complexity of the efforts etc.
* though the details will be different in each case.
* For many software projects, these framework activities are applied **iteratively** as a project progresses. Each iteration produces a software increment that provides a subset of overall software features and functionality.

**\* Hooker’s General Principles**

* 1: The Reason It All Exists
* 2: KISS (Keep It Simple, Stupid!)
* 3: Maintain the Vision
* 4: What You Produce, Others Will Consume
* 5: Be Open to the Future
* 6: Plan Ahead for Reuse
* 7: Think!

2. **Software 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

**1.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 …”  
   (a.k.a. “**The Mongolian Horde concept**”)
* “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 …”

**2. 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 …”

**3. 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 is baloney(nonsense.). It makes us create tons of paperwork, only to slow us down …”

3. Prescriptive process models:

The name 'prescriptive' is given because the model prescribes a set of activities, actions, tasks, quality assurance and change the mechanism for every project.

What is the prescriptive process model?A prescriptive process model is a model that describes "how to do" according to a certain software process system. A prescriptive model prescribes how a new software system should be developedWaterfall Model.Incremental Process Model.Evolutionary Process Model.Concurrent model

**Waterfall Model**Oldest software lifecycle model & best understood by upper management  
Used when requirements are well understood and risk is low  
Work flow is in a linear fashion (i.e., sequential) Used often with well-defined adaptations or enhancements to current software

**Communication**

Project initiation

Requirement gathering

**Planning**

Estimating

Scheduling

Tracking

**Modeling**

Analysis

Design

**Construction**

Code

Test

**Deployment**

Delivery

Support

Feedback

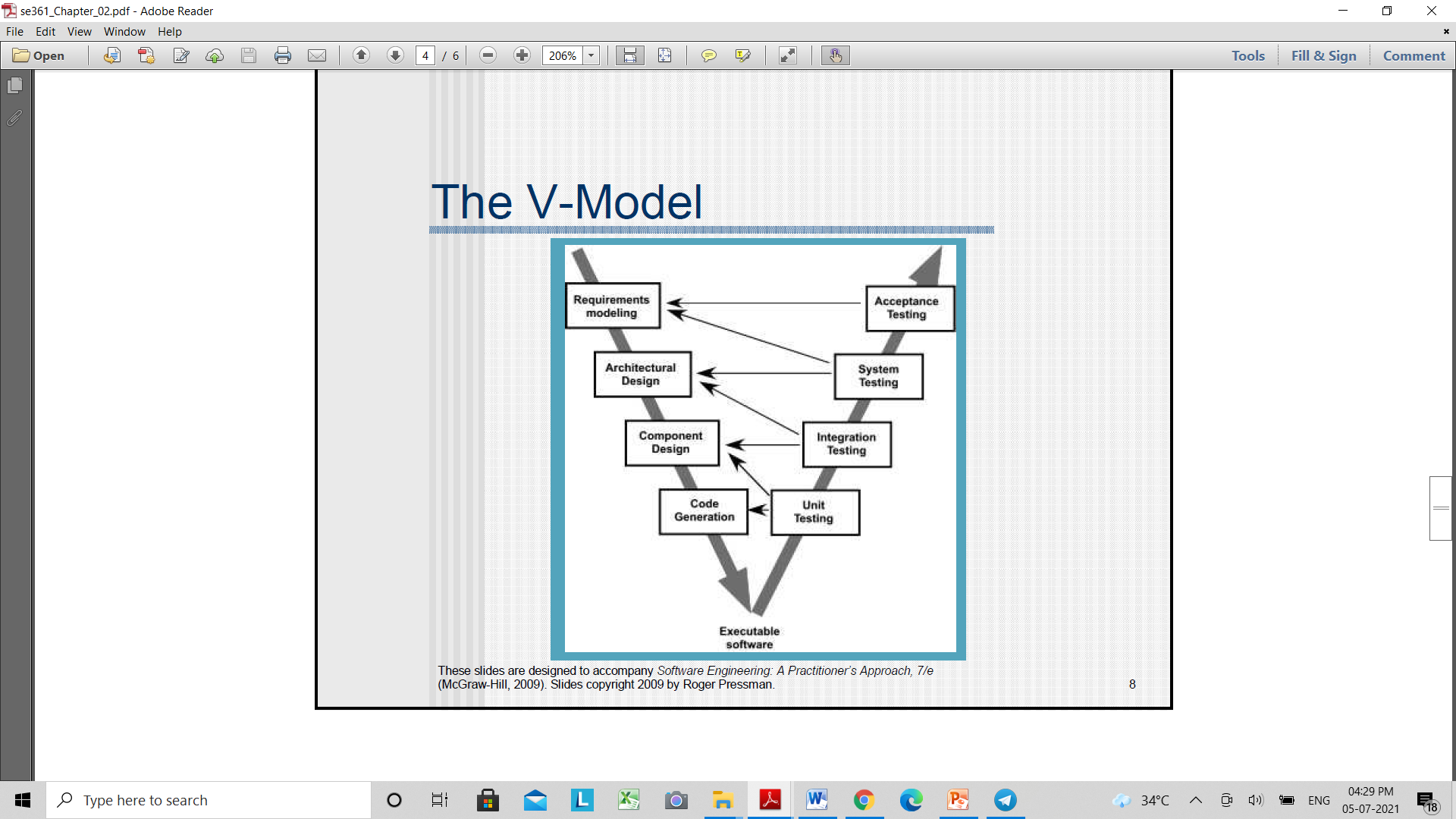
**Waterfall Model Problems:**

Real projects rarely follow sequential flow.Difficult for customers to state all requirements explicitly and up frontRequires customer patience because a working version of the program doesn't occur until the final phaseDoesn't support iteration, so changes can cause confusion and Leads to “blocking states”.UsefulWhere requirements are fixed and is to proceed to completion in a linear manner.

**Cons in waterfall model:**

Real projects rarely follow the sequential flow that the model proposesIt is often difficult for the customer to state all requirements explicitlyThe customer must have patience. A working version of the program(s) will not be available until late in the project time span

**The V-Model:**



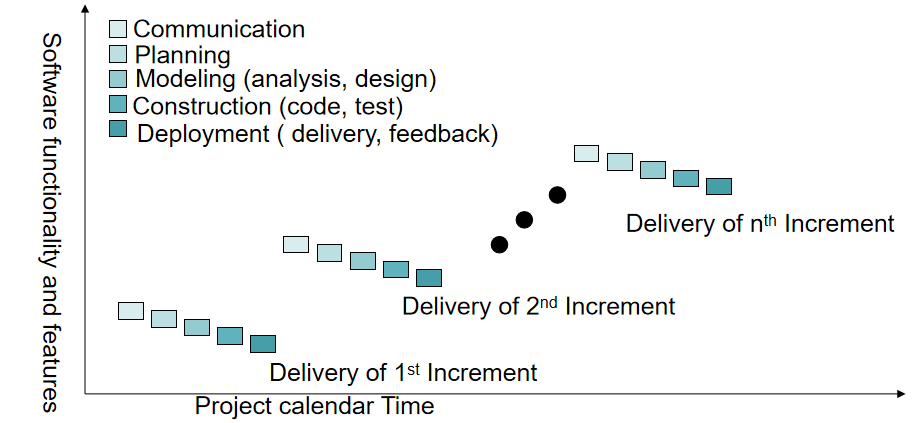
The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering workV- model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins**Advantages of V-model:**Simple and easy to use.Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.Defects are found at early stage.Works well for small projects where requirements are easily understood.

**Disadvantages of V-model:**  
Very rigid and least flexible.Software is developed during the implementation phase, so no early prototypes of the software are produced.If any changes happen in midway, then the test documents along with requirement documents has to be updated.

**Incremental Process Model**:

Combines elements of linear and parallel process flows.It delivers a series of releases, called increments that provide progressively more functionality for the customer as each is deliveredThe first increment is often a core product.The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional feature and functionality.It focuses on the delivery of an operational product with each increment.

It is useful when staffing is unavailable for a complete implementation.Increments can be planned to manage technical risks.



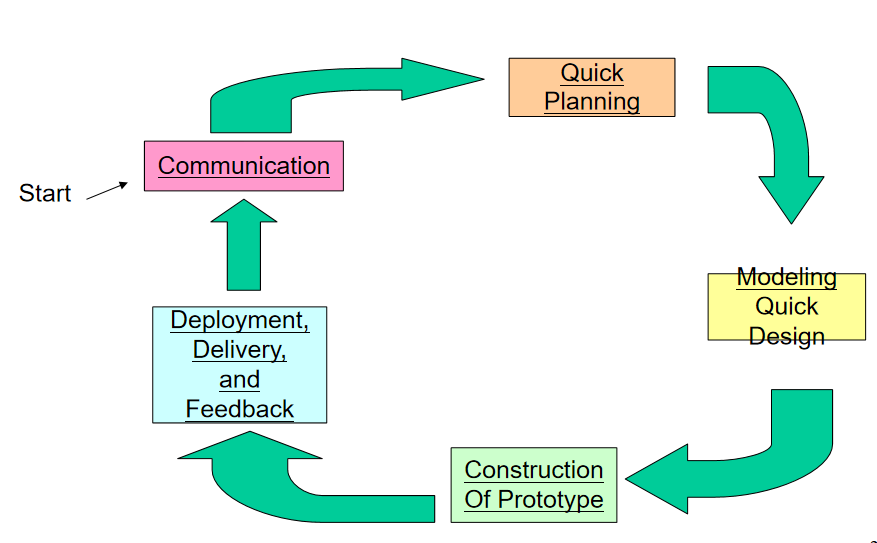
**Advantages of Incremental model:**Generates working software quickly and early during the software life cycle.This model is more flexible – less costly to change scope and requirements.It is easier to test and debug during a smaller iteration.In this model customer can respond to each built.Lowers initial delivery cost.Easier to manage risk because risky pieces are identified and handled during it’d iteration.

**Disadvantages of Incremental model:**Needs good planning and design.Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.Total cost is higher than waterfall.

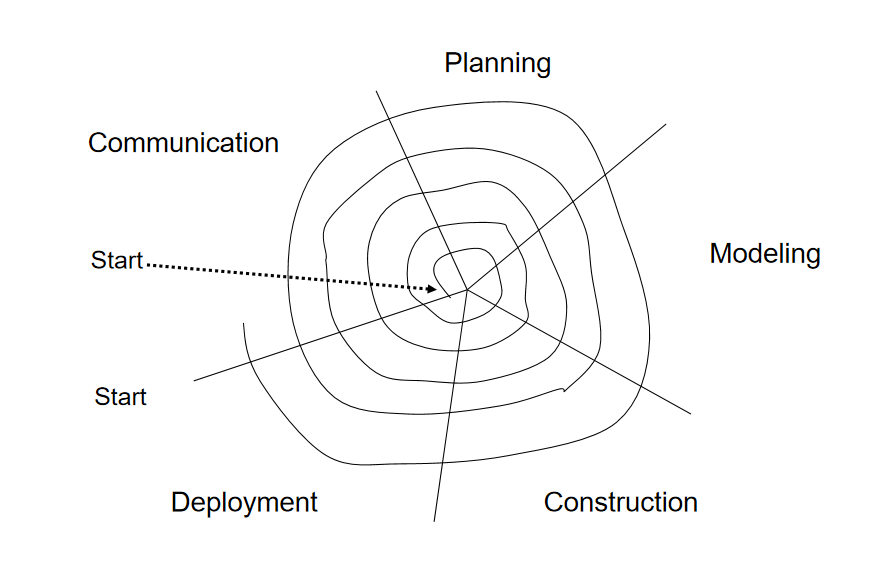
**Evolutionary Process Model**Software, like Complex systems evolve over a period of time

-Business and product requirements often change as development proceeds. - tight market deadlines make completion of a comprehensive software product impossible, but a limited version must be introduced to meet competitive or business pressure - a set of core product or system requirements is well understood, but the details of product or system extensions have yet to be defined.Evolutionary models are iterative.Evolutionary Process Model produce an increasingly more complete version of the software with each iteration.Specification, development and validation are interleaved.

Two types Evolutionary Process Model PrototypingSpiral ModelPrototyping model Diagram:



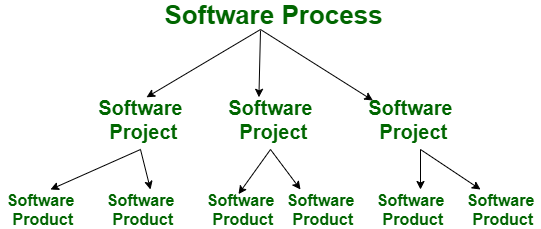
Spiral model:

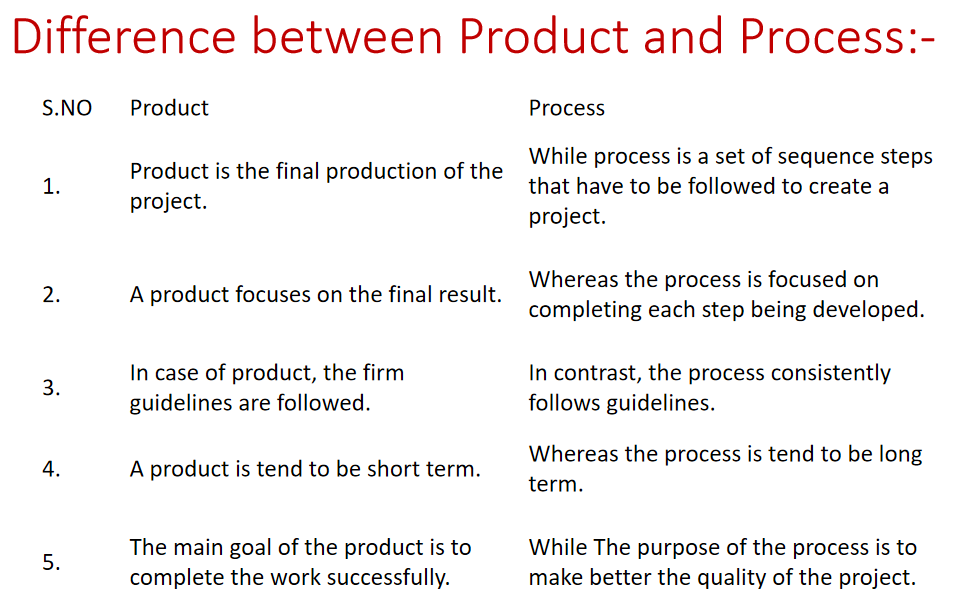


Concurrent Modeling:

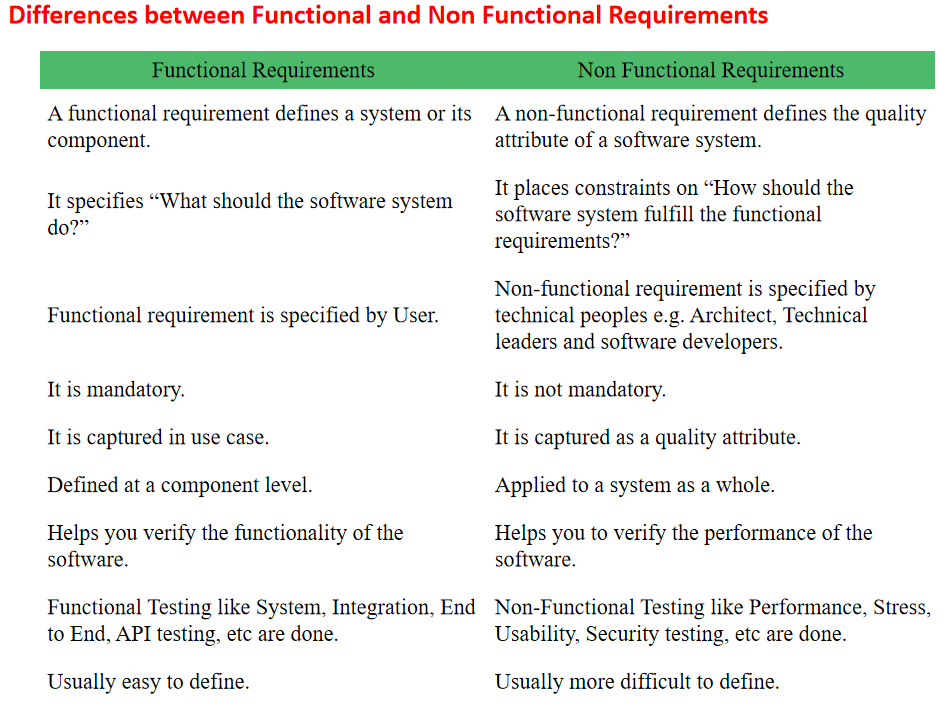
The concurrent development model sometimes called Concurrent Engineering.It allows team to represent iterative and concurrent elements of any of the process models.All software engineering activities exist concurrently but reside in different states.Concurrent modeling defines a series of events that will trigger transitions from state to state for each of the activities.Concurrent modeling is applicable to all types of software development and provide an accurate picture of the current state of a project.

4. **Product and Process:**Product includes any software manufactured based on the customer’s request. This can be a problem solving software or computer based system. It can also be said that this is the result of a project.**Process**Process is a set of sequence steps that have to be followed to create a project. The main purpose of a process is to improve the quality of the project. The process serves as a template that can be used through the creation of its examples and is used to direct the project.

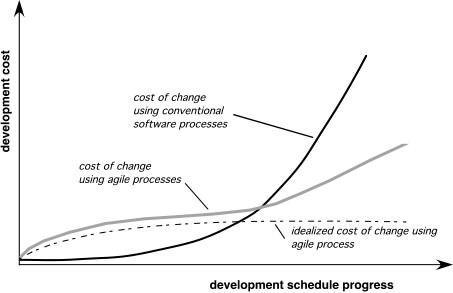




5. Functional and Non-functional



7. Agility Principles:



Agility and the cost of change.

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. 3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. 4. Business people and developers must work together daily throughout the project. 5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. 6. The most efficient and effective method of conveying information to and within a development team is face–to–face conversation.7. Working software is the primary measure of progress. 8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. 9. Continuous attention to technical excellence and good design enhances agility. 10. Simplicity – the art of maximizing the amount of work not done – is essential. 11. The best architectures, requirements, and designs emerge from self–organizing teams. 12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

8. Software requirement Specifications(SRS)

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.**Qualities of SRS:**CorrectUnambiguousCompleteConsistentRanked for importance and/or stabilityVerifiableModifiableTraceable