Introduction to Software Engineering

Software & Software Engineering

What is Software?

- *The product that software professionals build and then support over the long term.*
- *Software consists of:*
  - *(1) instructions (computer programs) that when executed provide desired features, function, and performance;*
  - *(2) data structures that enable the programs to adequately store and manipulate information and*
  - *(3) documentation that describes the operation and use of the programs.*

Software products

- Generic products
  - Stand-alone systems that are marketed and sold to **any customer** who wishes to buy them.
  - Examples – PC software such as editing, graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
- Customized products
  - Software that is commissioned by **a specific customer** to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.

Software Engineering Definition

The seminal definition:

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

The IEEE definition:

The application and study of approaches of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software is Software Engineering

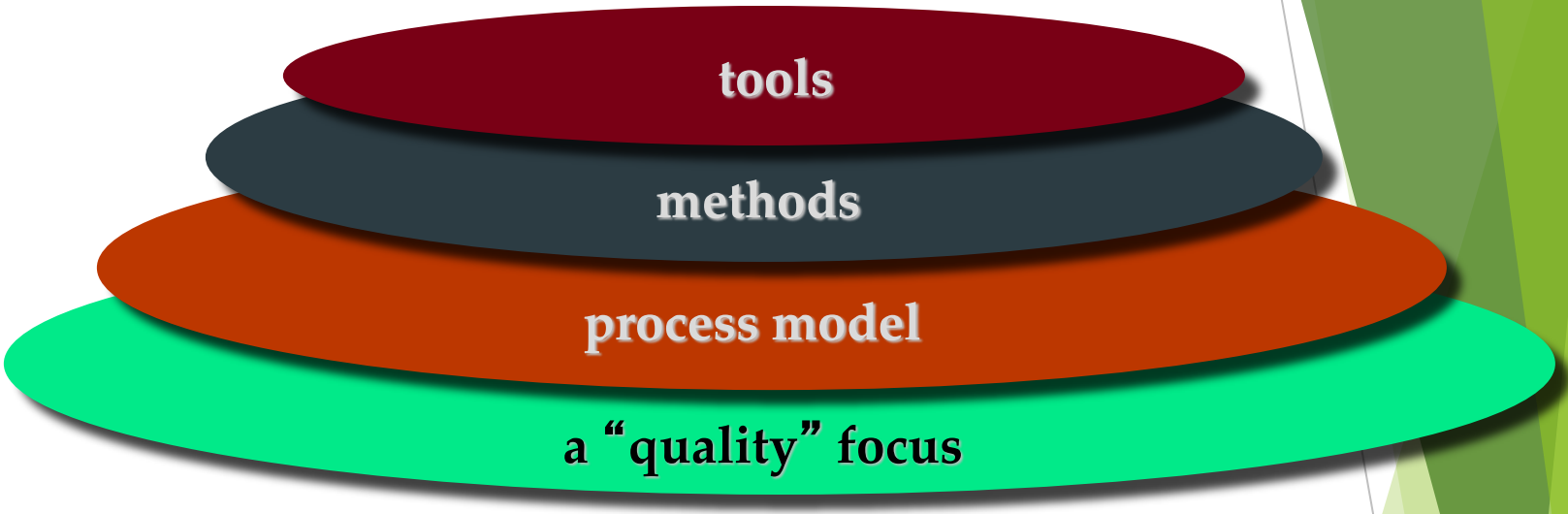
Importance of Software Engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

FAQ about software engineering

Question	Answer
What is software?	Computer programs, data structures 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 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.

A Layered Technology



- Any engineering approach must rest on organizational commitment to **quality** which fosters a continuous process improvement culture.
- **Process** layer as the foundation defines a framework with activities for effective delivery of software engineering technology. Establish the context where products (model, data, report, and forms) are produced, milestone are established, quality is ensured and change is managed.
- **Method** provides technical how-to's for building software. It encompasses many tasks including communication, requirement analysis, design modeling, program construction, testing and support.
- **Tools** provide automated or semi-automated support for the process and methods.

Software Process

- A process is a collection of activities, actions and tasks that are performed when some work product is to be created. It is **not a rigid prescription** for how to build computer software. Rather, it is an adaptable approach that enables the people doing the work to pick and choose the **appropriate set of work actions** and tasks.
- Purpose of process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

Five Activities of a Generic Process framework

- Communication: communicate with customer to understand objectives and gather requirements
- Planning: creates a “map” defines the work by describing the tasks, risks and resources, work products and work schedule.
- Modeling: Create a “sketch”, what it looks like architecturally, how the constituent parts fit together and other characteristics.
- Construction: code generation and the testing.
- Deployment: Delivered to the customer who evaluates the products and provides feedback based on the evaluation.

Umbrella Activities

Complement the five process framework activities and help team manage and control progress, quality, change, and risk.

- Software project tracking and control: assess progress against the plan and take actions to maintain the schedule.

- Risk management: assesses risks that may affect the outcome and quality.
  - Risk matrix
  - Decision tree

- Software quality assurance: defines and conduct activities to ensure quality.
  - ISO 9126 quality factors
  - ISO 9000 quality standards
  - Mc Call's Quality Factors
  - Six Sigma Principles
- Technical reviews: assesses work products to uncover and remove errors before going to the next activity.
- Measurement: define and collects process, project, and product measures to ensure stakeholder's needs are met.

- Software configuration management: manage the effects of change throughout the software process.
  - Software Configuration Management Process

Software Configuration Tool	Description
IBM Rational ClearCase	Used to provide complete software configuration capability and enables software team to implement a common, consistent process for *submitting*, *assigning*, *resolving*, and *verifying changes* across the software development.
Surround SCM	Used to manage source code changes with ease. It maintains complete and accurate records of *who*, *what* and *when* of changes.
Serena ChangeMan Version Manager	Used for software change and configuration management. It comprises locking, synchronization, automatically reconciling and merging multiple changes, which are made concurrently through distributed or parallel programming.
CCC/ Harvest	Used for process control and supports change package. This tool is suitable for simple, well-defined projects.
PVCS	Used for projects having requirements for version control. It is particularly suited for organizations, which require a mixed range of hardware and software environments.
Razor	Used for projects, which utilize single repository with development and maintenance processes where problem tracking and change management are essential.
TRUEchange	Used for managing the flow of changes for producing applications in large organizations. It provides good software configuration management.
Serena TeamTrack	Used as an approach to manage changes in the IT environment.

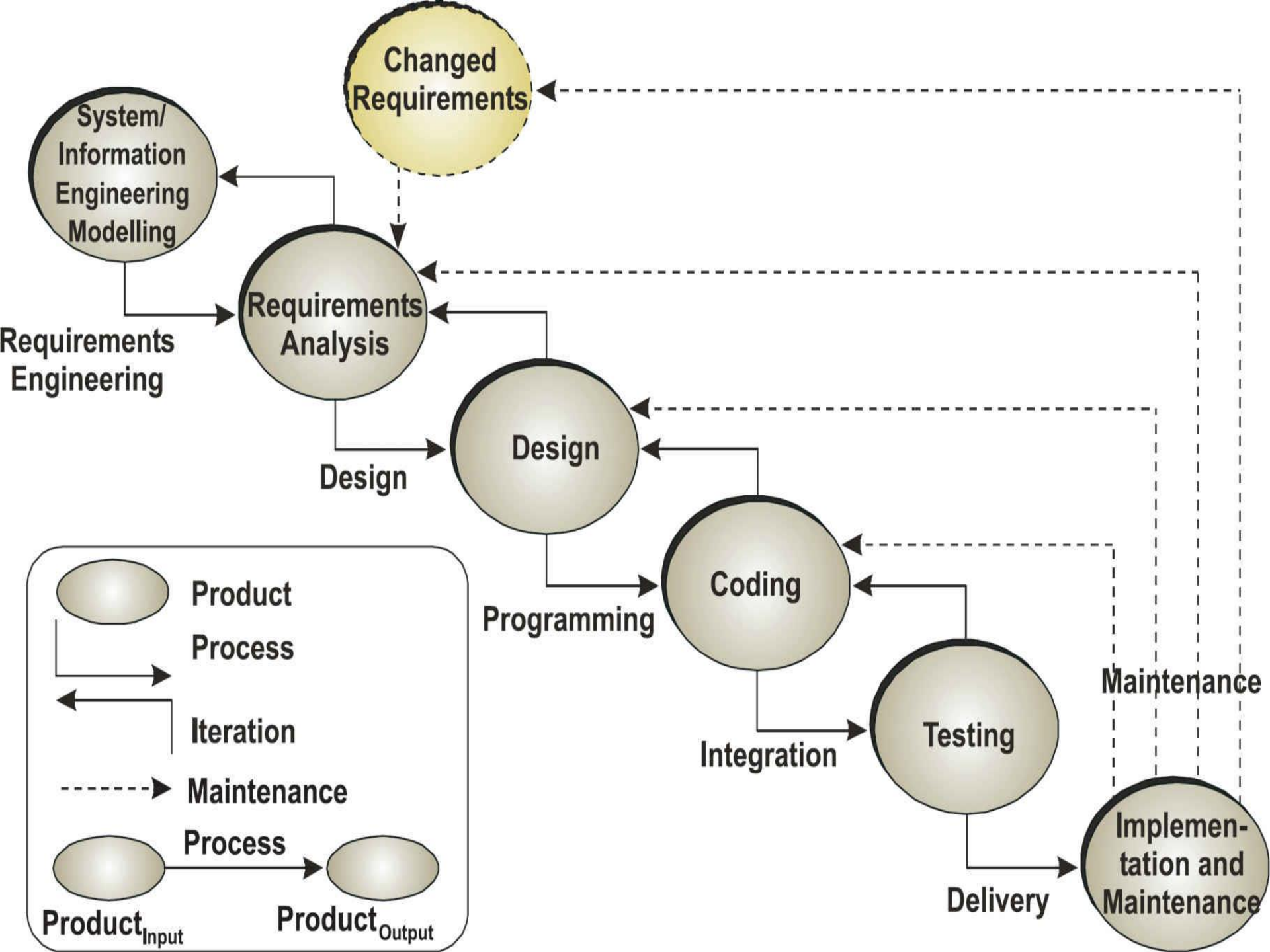
- Reusability management: defines criteria for work product reuse and establishes mechanism to achieve reusable components.
- Work product preparation and production: create work products such as models, documents, logs, forms and lists

Software Life Cycle models

- Process model depends on the nature and application of the software
- Different process models are as follows:
  - Waterfall model
  - Prototyping model
  - Spiral model
  - Incremental model
  - V model

The Waterfall model

- In the waterfall model (also known as the classical life cycle model), the development of software proceeds linearly and sequentially from requirements analysis to design, coding, testing, integration, implementation and maintenance.
- Thus, this model is also known as the linear sequential model.
- This model is simple to understand and represents processes which are easy to manage and measure.
- It comprises different phases and each phase has its distinct goal.


Changed Requirements
System/ Information Engineering Modelling
Requirements Engineering
Requirements Analysis
Design
Design
Programming
Coding
Integration
Testing
Maintenance
Delivery
Implemen- tation and Maintenance
Product
Process
Iteration
Maintenance
Process
Product Input
Product Output

Input to Phase	*Process/Phase*	*Output of Phase*
Requirements defined through communication	Requirements analysis	Software requirements specification document
Software requirements specification document	Design	Design specification document
Design specification document	Coding	Executable software modules
Executable software modules	Testing	Integrated product
Integrated product	Implementation	Delivered software
Delivered software	Maintenance	Changed requirements

Advantages

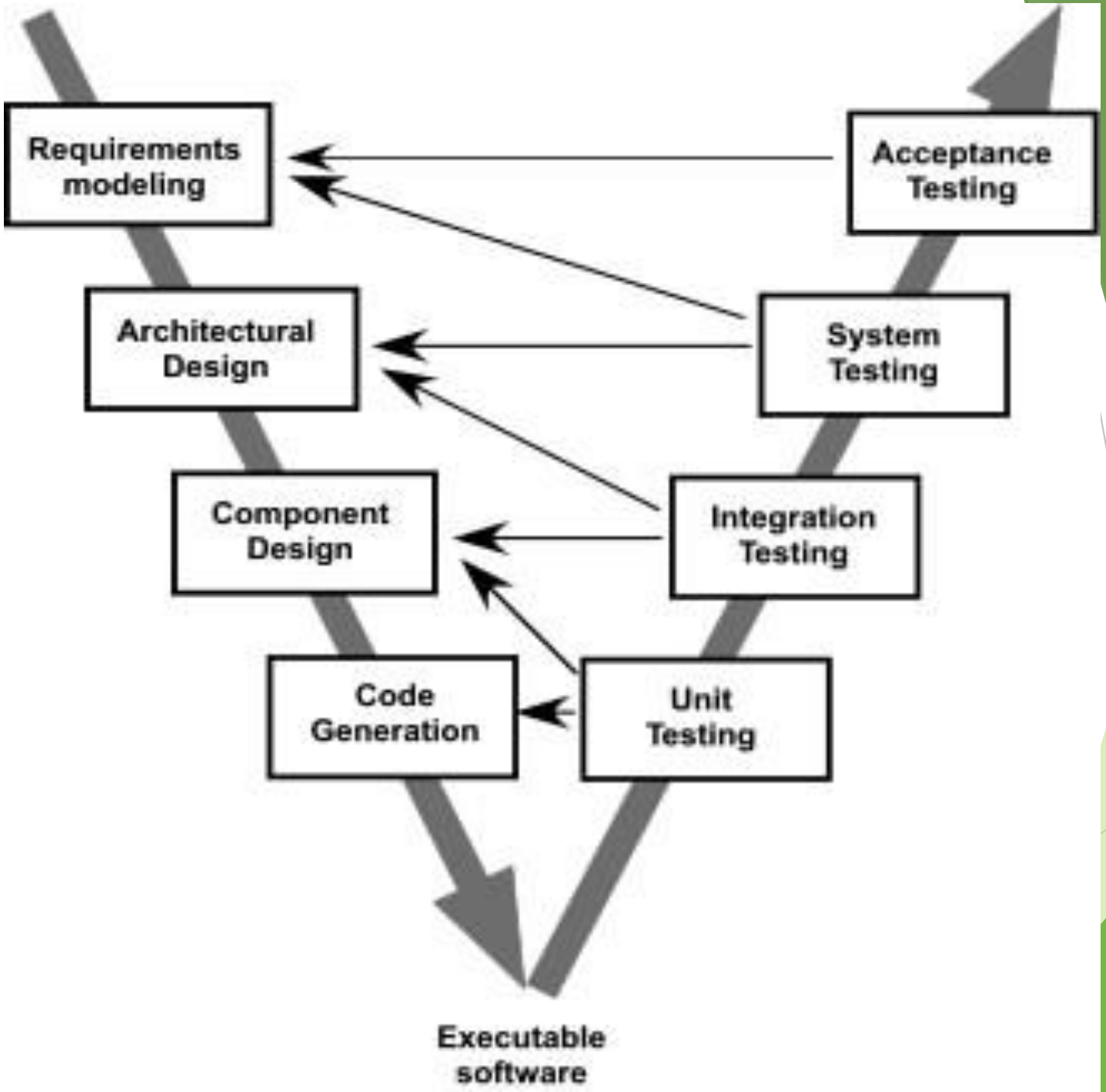
- This model is very simple and is easy to understand.
- Phases in this model are processed one at a time.
- Each stage in the model is clearly defined.
- This model has very clear and well understood milestones.
- Process, actions and results are very well documented.
- Reinforces good habits: define-before- design, design-before-code.
- This model works well for smaller projects and projects where requirements are well understood.

Drawbacks

- The model expects complete & accurate requirements early in the process, which is unrealistic
- It is difficult to define all requirements at the beginning of a project
- This model is not suitable for accommodating any change
- A working version of the system is not seen until late in the project's life
- It does not scale up well to large projects.
- Real projects are rarely sequential.

The V-Model(Verification and Validation model)

- A variation of waterfall model depicts the relationship of quality assurance actions to the actions associated with communication, modeling and early code construction activates.
- Team first moves down the left side of the V to refine the problem requirements. Once code is generated, the team moves up the right side of the V, performing a series of tests that validate each of the models created as the team moved down the left side.


Requirements modeling
Architectural Design
Component Design
Code Generation
Unit Testing
Integration Testing
System Testing
Acceptance Testing
Executable software

The V-Model(Verification and Validation model)

- **Verification:** It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.
- **Validation:** It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.
- So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.

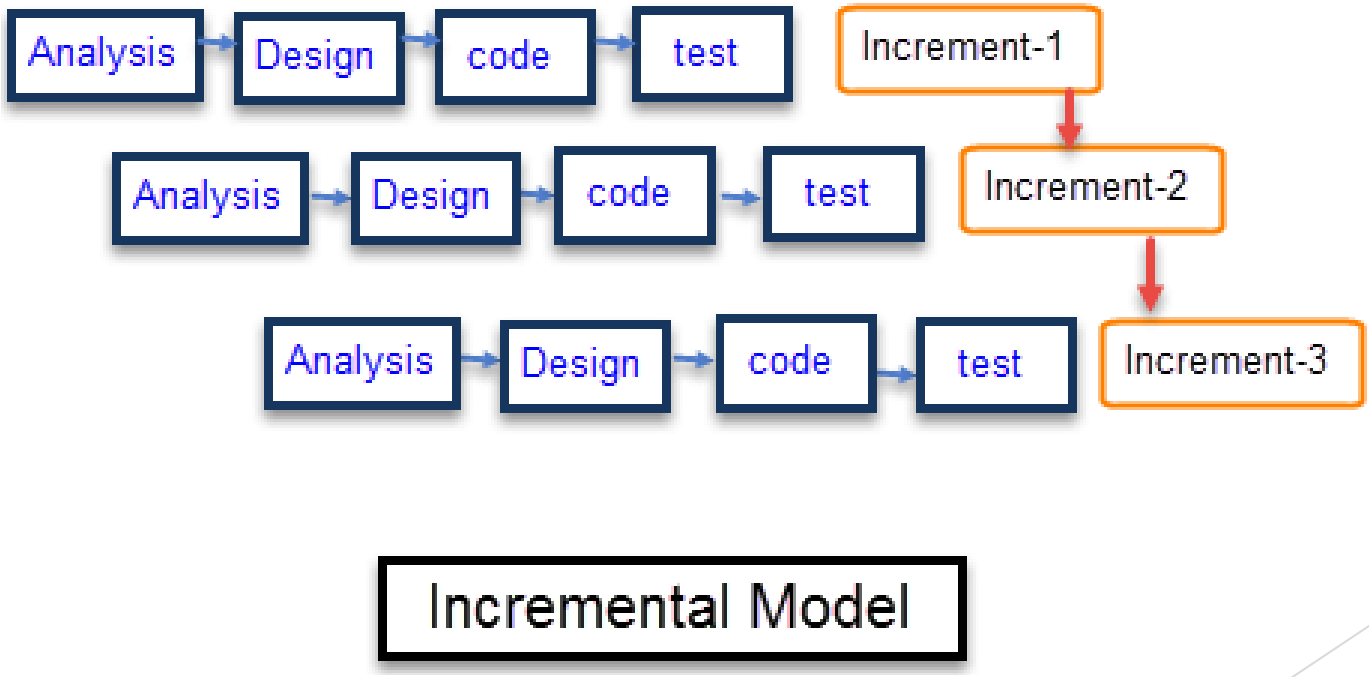
Advantages

- This is a highly disciplined model and Phases are completed one at a time.
- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.
- This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
- It enables project management to track progress accurately.

Disadvantages

- High risk and uncertainty.
- It is not a good for complex and object-oriented projects.
- It is not suitable for projects where requirements are not clear and contains high risk of changing.
- This model does not support iteration of phases.
- It does not easily handle concurrent events.

Incremental model



Incremental Model

Incremental model

- Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle.
- Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.

- Each iteration passes through the requirements, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.
- The system is put into production when the first increment is delivered.
- The first increment is often a core product where the basic requirements are addressed, and supplementary features are added in the next increments.
- Once the core product is analyzed by the client, there is plan development for the next increment.

Characteristics of Incremental model

- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first
- Once the requirement is developed, requirement for that increment are frozen

Advantages

- The software will be generated quickly
- It is flexible and less expensive to change requirements and scope
- Throughout the development stages changes can be done
- This model is less costly compared to others
- A customer can respond to each building
- Errors are easy to be identified

Disadvantages

- It requires a good planning designing
- Problems might arise due to system architecture because not all requirements collected up front for the entire software lifecycle
- Each iteration phase is rigid and does not overlap each other
- Rectifying a problem in one unit requires correction in all the units and consumes a lot of time

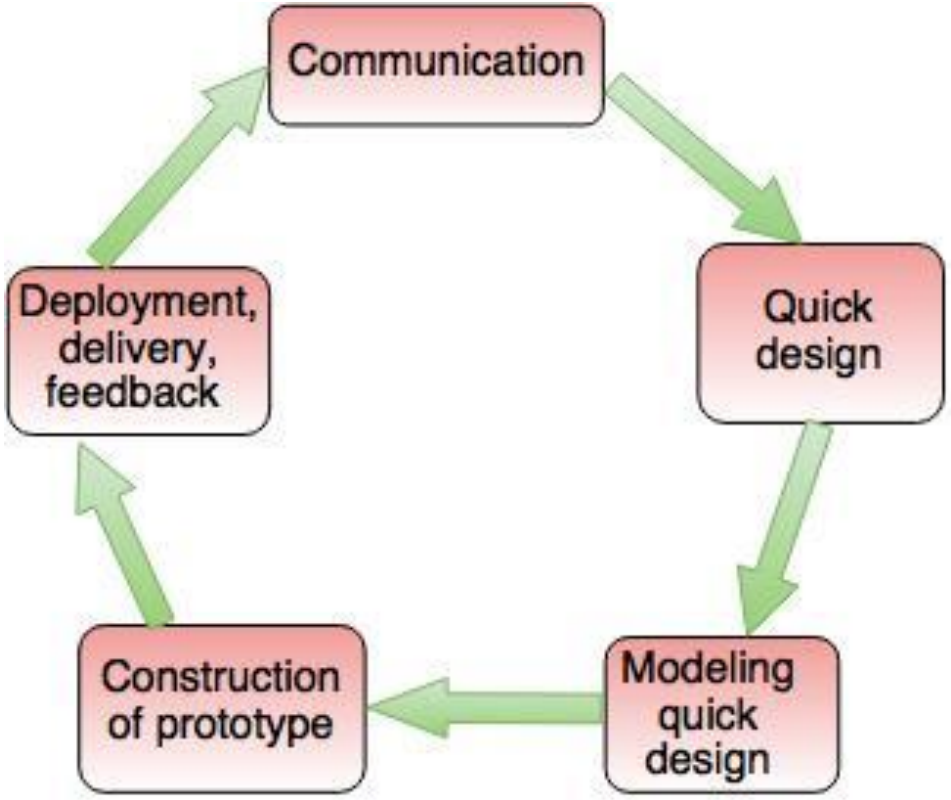
Incremental model can be used when…

- Requirements of the system are clearly understood
- demand for an early release of a product arises
- software engineering team are not very well skilled or trained
- high-risk features and goals are involved
- Development is web application and happens for product based companies

Evolutionary Models

- Software system evolves over time as requirements often change as development proceeds. Thus, a straight line to a complete end product is not possible. However, a limited version must be delivered to meet competitive pressure.
- Usually a set of core product or system requirements is well understood, but the details and extension have yet to be defined.
- You need a process model that has been explicitly designed to accommodate a product that evolved over time.
- It is iterative that enables you to develop increasingly more complete version of the software.
- Two types are introduced, namely Prototyping and Spiral models.

Prototyping model



The Prototype model

- Prototype is defined as first or preliminary form using which other forms are copied or derived.
- Prototype model is a set of general objectives for software.
- It does not identify the requirements like detailed input, output.
- It is software working model of limited functionality.
- In this model, working programs are quickly produced.

- **1. Communication**
  In this phase, developer and customer meet and discuss the overall objectives of the software.

  2. Quick design: Quick design is implemented when requirements are known.
- It includes only the important aspects like input and output format of the software.
- It focuses on those aspects which are visible to the user rather than the detailed plan.
- It helps to construct a prototype.

- **3. Modeling quick design:** This phase gives the clear idea about the development of software because the software is now built.
- It allows the developer to better understand the exact requirements.
- **4. Construction of prototype**
  The prototype is evaluated by the customer itself.

  5. Deployment, delivery, feedback: If the user is not satisfied with current prototype then it refines according to the requirements of the user.
- The process of refining the prototype is repeated until all the requirements of users are met.
- When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

Prototype model – When to use?

- **When to use:** Customer defines a set of general objectives but does not identify detailed requirements for functions and features. Or Developer may be unsure of the efficiency of an algorithm, the form that human computer interaction should take.
- Both stakeholders and software engineers like the prototyping paradigm. Users get a feel for the actual system, and developers get to build something immediately. However, engineers may make compromises in order to get a prototype working quickly. The less-than-ideal choice may be adopted forever after you get used to it.

Advantages

- Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
- In the development process of this model users are actively involved.
- The development process is the best platform to understand the system by the user.
- Errors are detected much earlier.
- Gives quick user feedback for better solutions.
- It identifies the missing functionality easily. It also identifies the confusing or difficult functions.

Disadvantages

- The client involvement is more and it is not always considered by the developer.
- It is a slow process because it takes more time for development.
- Many changes can disturb the rhythm of the development team.
- It is a thrown away prototype when the users are confused with it.
-

Evolutionary Models: The Spiral

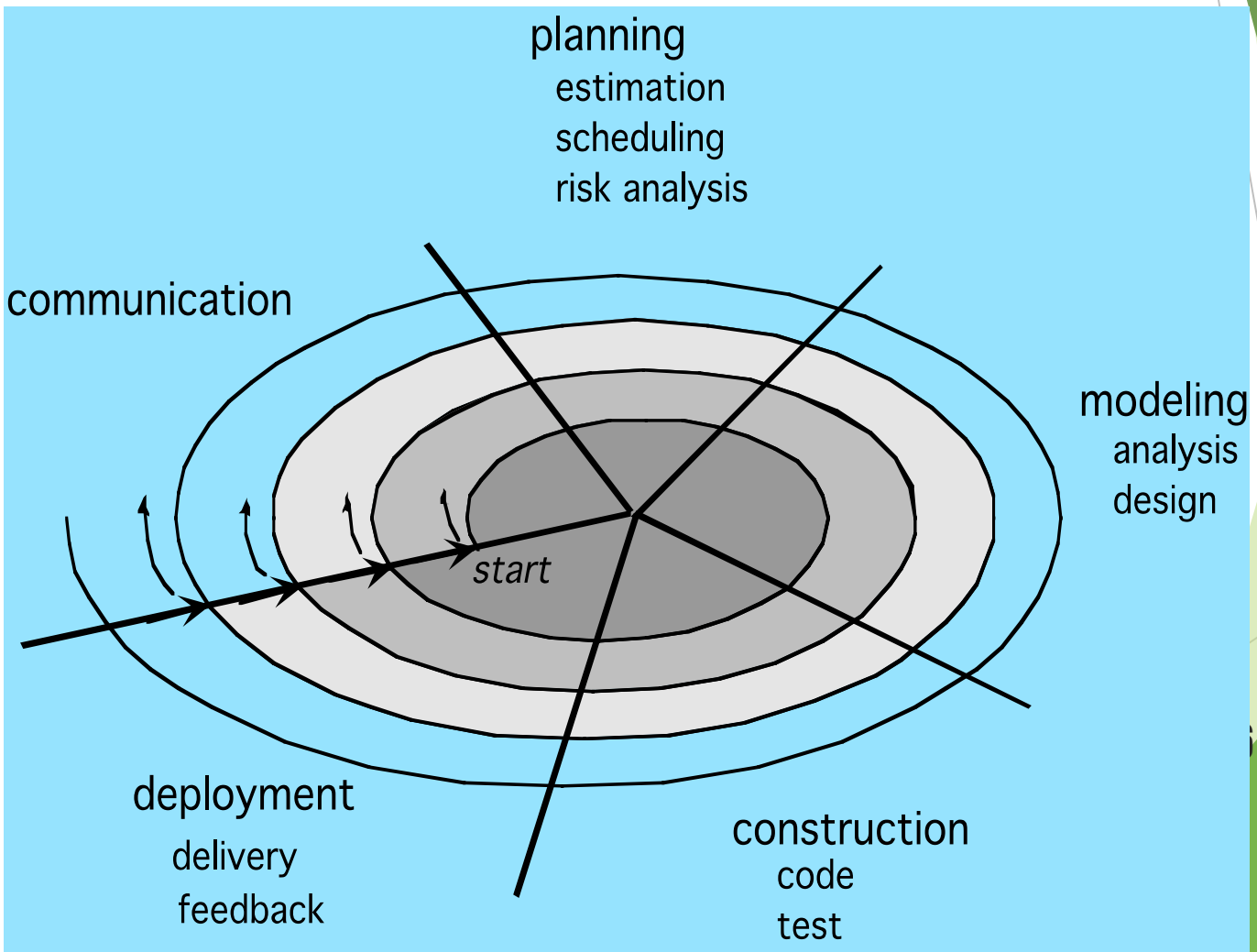
- Spiral Model is highly used in IT companies.
- This model involves strategies, which is a combination of incremental and prototype models.
- This Spiral Model is best to use for large projects which required more management and planning.

- A spiral model is a realistic approach to the development of large-scale software products because the software evolves as the process progresses.
- In addition, the developer and the client better understand and react to risks at each evolutionary level.
- The model uses prototyping as a risk reduction mechanism and allows for the development of prototypes at any stage of the evolutionary development.

- It maintains a systematic step wise approach, like the classic Life Cycle model but incorporates it into an iterative framework that more reflect the real world.
- If employed correctly, this model should reduce risks before they become problematic as consideration of technical risks are considered at all stages.

- A spiral model is divided into a set of framework activities
- Each of the framework activities represent one segment of the spiral path
- The first circuit around the spiral will result in the development of a product specification and progressively more sophisticated versions
- Unlike other development process, spiral model can be applied throughout the life of the software.

Evolutionary Models: The Spiral



Advantages

- Spiral Model mostly concentrates on risk analysis.
- Most useful for large and risky projects.
- Spiral Model can be used if requirement changes frequently.
- Focused model for all phases.
- Customer evaluation phase makes this model useful.

Disadvantages

- For risk, analysis phase requires an expert person to make an analysis.
- Not useful for small projects.
- Project duration and cost could be infinite because of the spiral feature.
- Documentation could become lengthy.

Concurrent Model

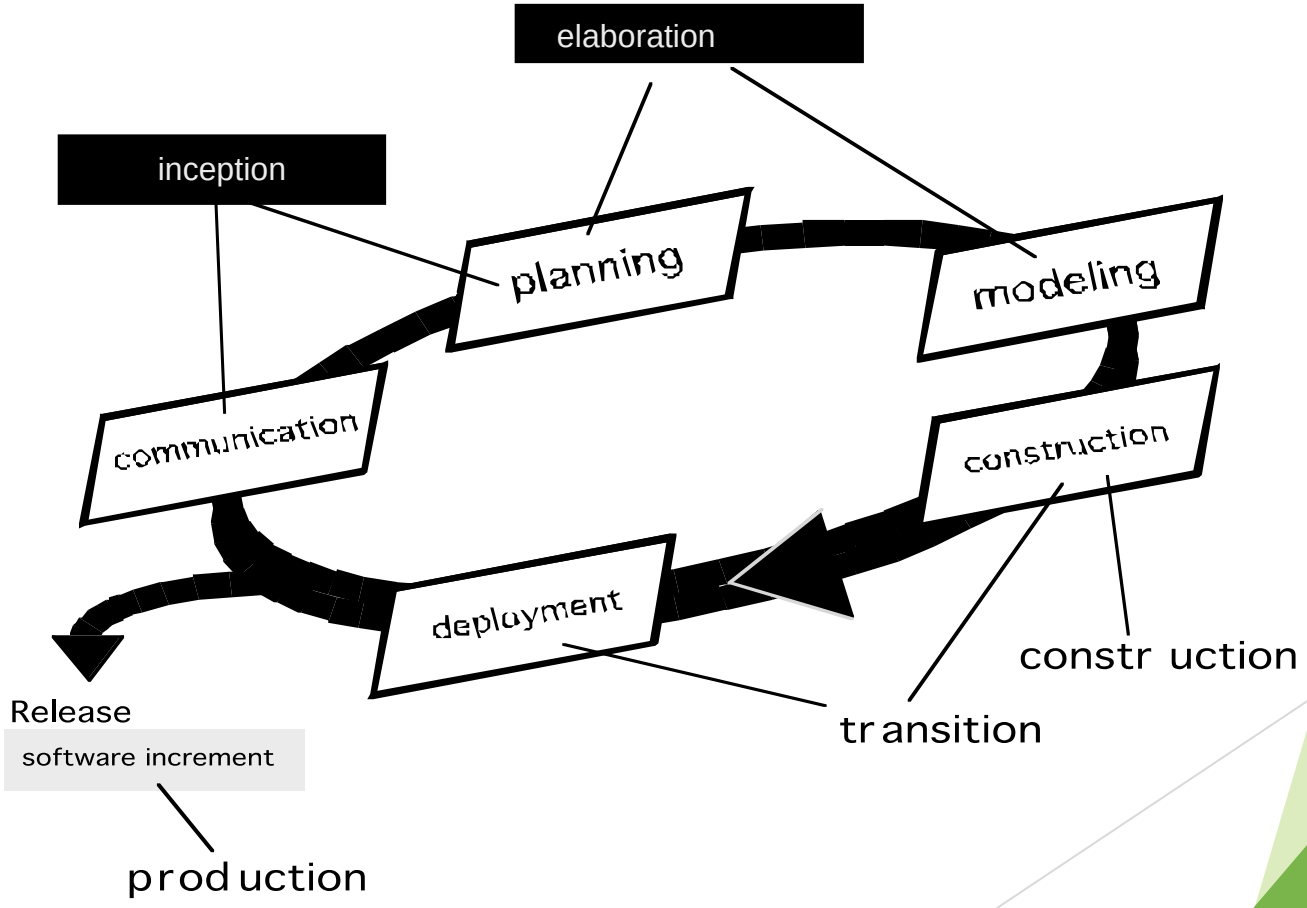
- The *concurrent development model*, sometimes called *concurrent engineering*, can be represented schematically as a series of framework activities, Software engineering actions of tasks, and their associated states.

- The concurrent model is often more appropriate for system engineering projects where different engineering teams are involved.

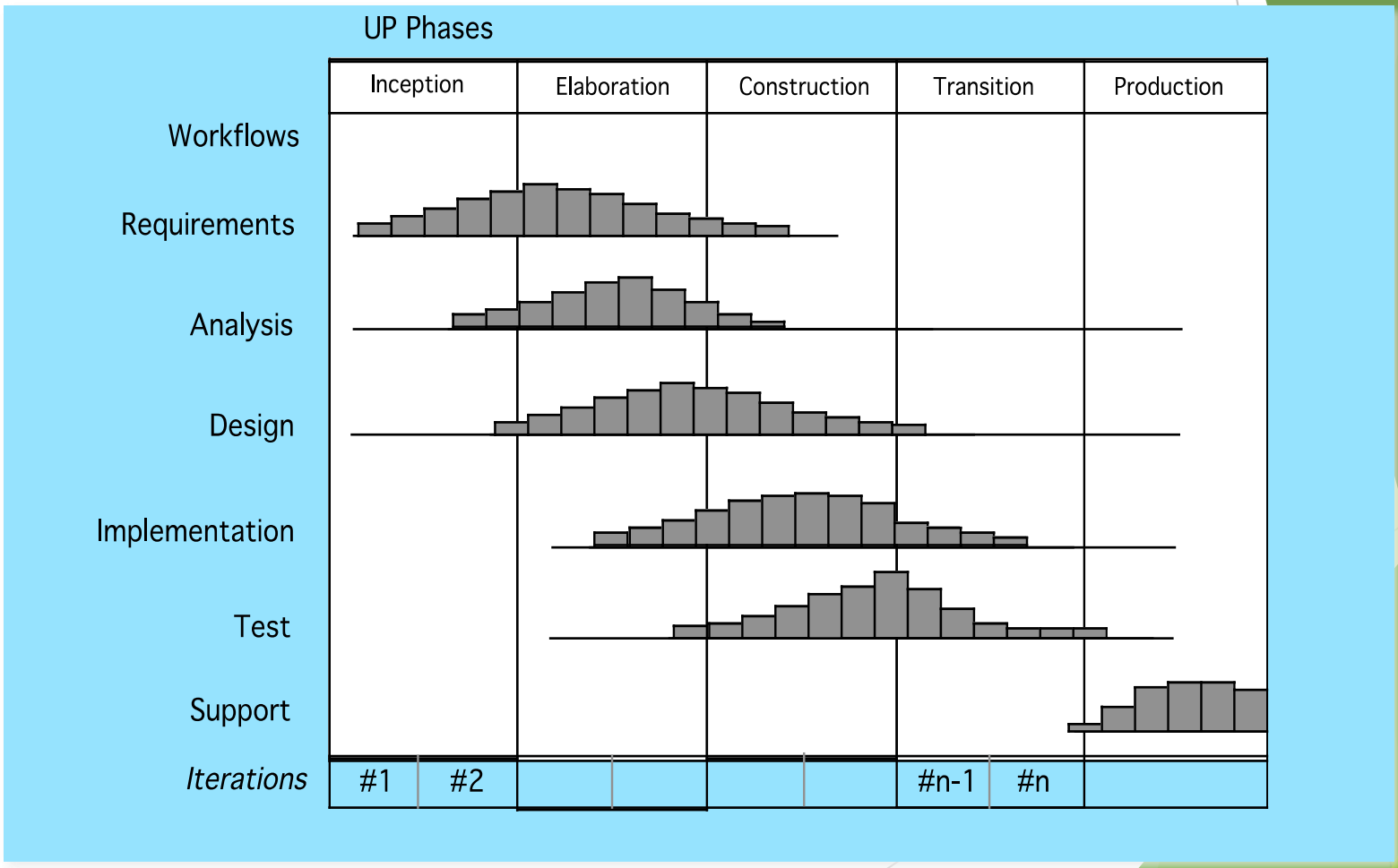
Other Process Models

- Component based development—the process to apply when reuse is a development objective (like spiral model)
- Formal methods—emphasizes the mathematical specification of requirements (easy to discover and eliminate ambiguity, incompleteness and inconsistency)
- Aspect Oriented software development (AOSD)—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects*
- Unified Process—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML) to model and develop object-oriented system iteratively and incrementally.

The Unified Process (UP)



UP Phases


UP Phases
Inception
Elaboration
Construction
Transition
Production
Workflows
Requirements
Analysis
Design
Implementation
Test
Support
Iterations
#1
#2
#n-1
#n