Software Engineering and Project Management

Dr.Arun Shivashankarappa

Source *: Roger S. Pressman: Software Engineering-A Practitioners approach, 7th Edition, Tata McGraw Hill.*

Course Outcomes : At the end of the course, the student will be able to:

- Differentiate process models to judge which process model has to be adopted for the given scenarios.
- Derive both functional and nonfunctional requirements from the case study.
- Analyze the importance of various software testing methods and agile methodology.
- Illustrate the role of project planning and quality management

What is Software ?

Software is a program or set of programs containing instructions that provide desirable functions

What is Engineering ?

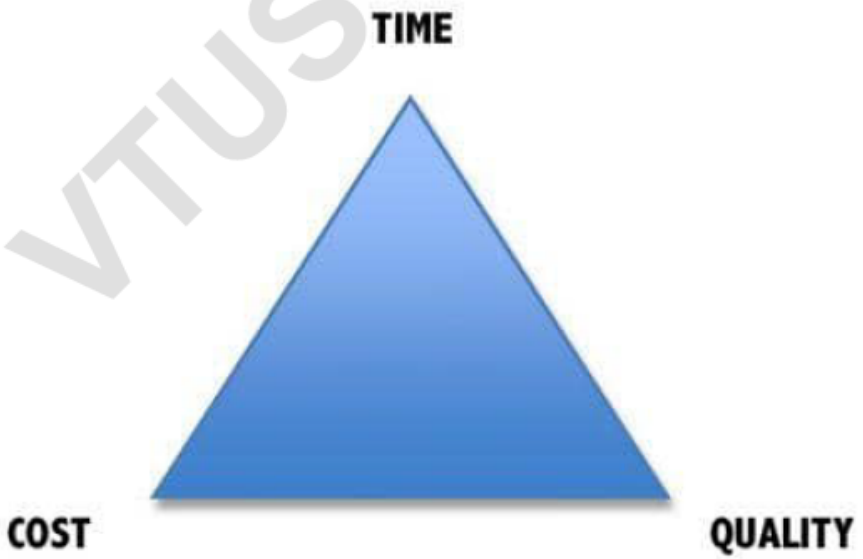
Application of science, tools and methods to find the cost effective solution to problems

What is Software Engineering?

Software Engineering is a systematic, disciplined and quantifiable approach used in the development, operation and maintenance of software

Why is Software Engineering important?

Complex systems need a disciplined approach for designing, developing and managing them.



Software Development Crises

Projects were:

- Late.
- Over budget.
- Unreliable.
- Difficult to maintain.
- Performed poorly.

Software Crisis

Example 1: 2009,Computer glitch delays flights

Saturday 3rd October 2009-London, England (CNN)

- Dozens of flights from the UK were delayed Saturday after a glitch in an air traffic control system in Scotland, but the problem was fixed a few hours later.
- The **Air Trafic Service** agency said it reverted to backup equipment as engineering worked on the system.
- The problem did not create a safety issue but could cause delays in flights.
- Read more at: http://edition.cnn.com/2009/WORLD/europe/10/03/uk.flights.delayed

Software Crisis

Example 2: Ariane 5 Explosion

- June 4th 1996, from French Guiana
- European Space Agency spent 10 years and $7 billion to produce Ariane 5.
- Crash after 36.7 seconds, due to software bug in the rockets inertial Reference System.
- Caused by an overflow error. Trying to store a 64-bit number into a 16-bit space.
- Watch the video: http://www.youtube.com/watch?v=z-r9cYp3tTE

Software Crisis

Example 3: 1992, London Ambulance Service

- Considered the largest ambulance service in the world.
- Overloaded problem.
- It was unable to keep track of the ambulances and their statuses. Sending multiple units to some locations and no units to other locations.
- Generates many exceptions messages.
- 46 deaths.

Therefore...

A well-disciplined approach to software development and management is necessary. This is called Software engineering.

TYPES OF SOFTWARE

1. System software:

- a collection of programs written to service other programs.
- system software (e.g., compilers, editors, and file management utilities) processes complex, but determinate information structures.
- systems applications (e.g., operating system components, drivers, networking software, telecommunications processors) process largely indeterminate data.

2. Application software

- Program or group of programs designed for end users. Eg: MS word, PP, Excel, Email app etc

- **Hotel management software**
- **Hospital Management software**
- **College management software**
- **Human resource management software**

3. Embedded software

- **resides within a product or system and is used to** implement and control features and functions for the end user and for the system itself
- Embedded software can perform limited and esoteric functions (e.g., key pad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems).

4. Product-line software

- **designed to provide a specific capability for use by** many different customers.
- Product-line software can focus on a limited and esoteric marketplace (e.g., inventory control products) or address mass consumer markets (e.g., word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, and personal and business financial applications).

5. Web applications—called "WebApps,"

- **this network-centric software category** spans a wide array of applications.
- In their simplest form, WebApps can be little more than a set of linked hypertext files that present information using text and limited graphics.
- However, as Web 2.0 emerges, WebApps are evolving into sophisticated computing environments that not only provide stand-alone features, computing functions, and content to the end user, but also are integrated with corporate databases and business applications.

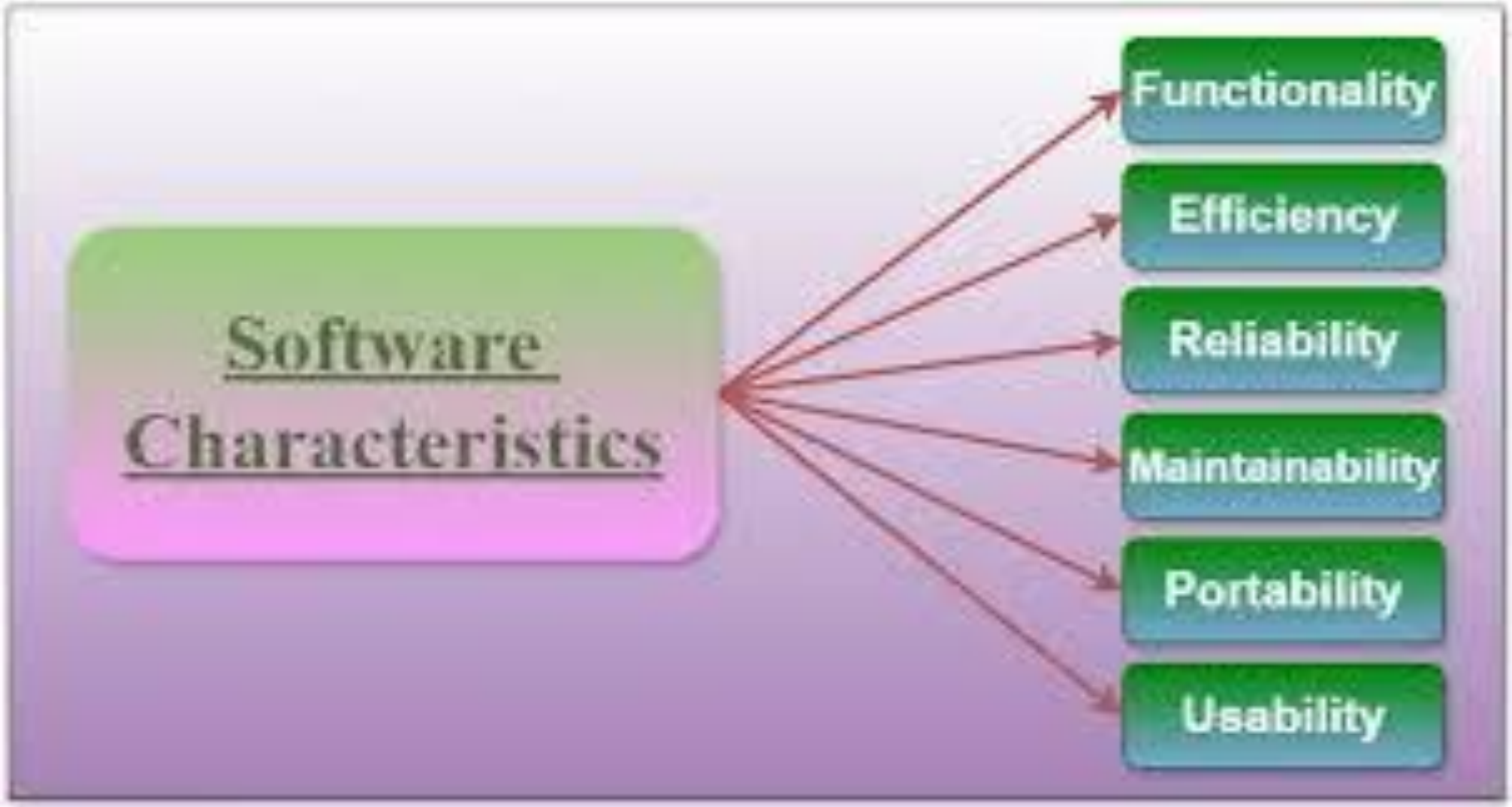
6. Artificial intelligence software

- **makes use of non numerical algorithms to** solve complex problems that are not amenable to computation or straightforward analysis.
- Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.

Software engineering challenges in 2025

- **Open-world computing:** The challenge is to develop systems and application software that will allow **mobile devices**, personal computers, and enterprise systems to communicate across vast networks
- **Netsourcing:** The challenge for software engineers is to architect simple (e.g., personal financial planning) and sophisticated applications that provide a benefit to targeted end-user markets
- **Open source:Open source—a growing trend that results in distribution of source code for** systems applications so that many people can contribute to its development.

Software Characteristics



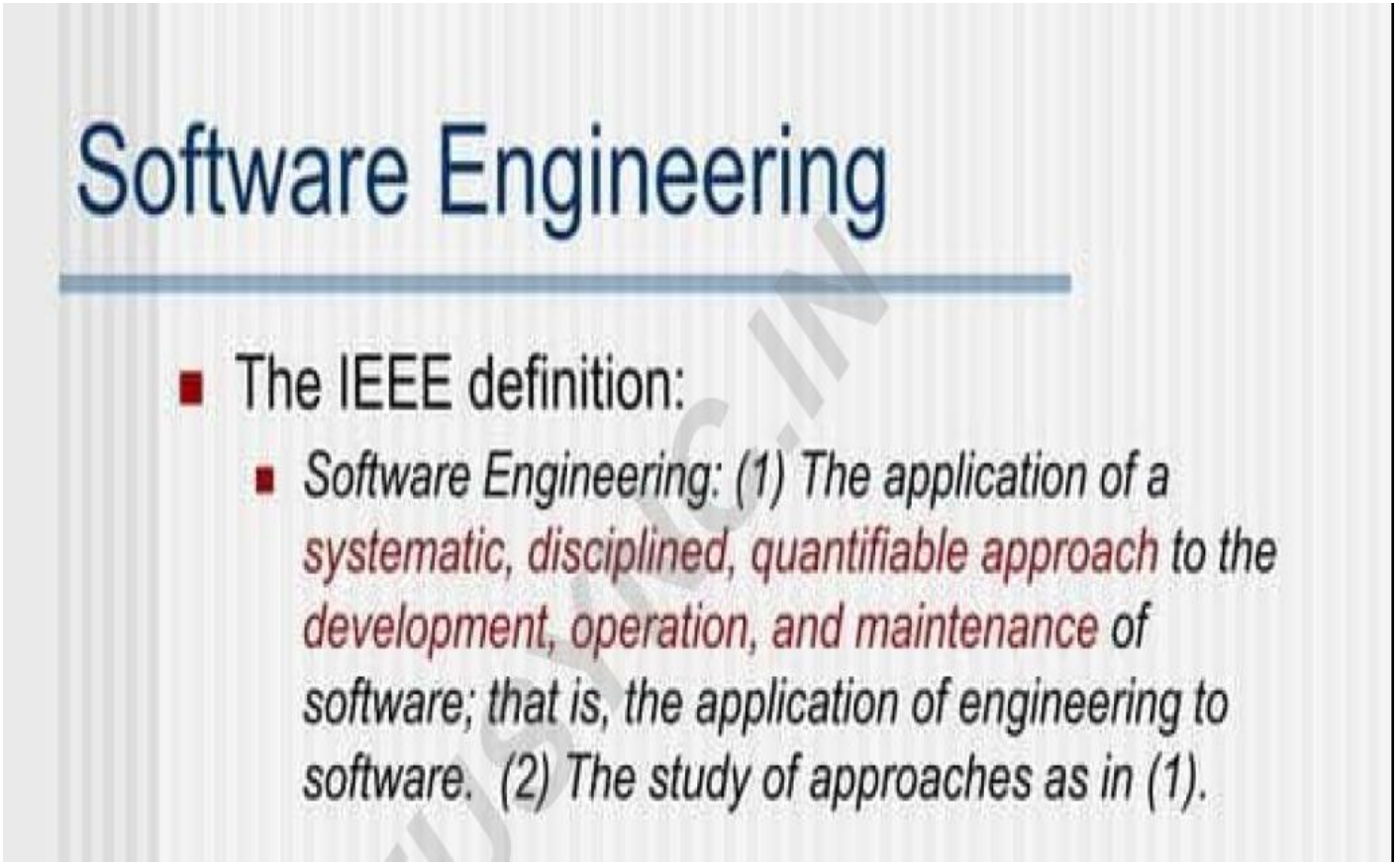
Characteristics of WebApps - I

- **Network intensiveness.** A WebApp resides on a network and must serve the needs of a diverse community of clients.
- **Concurrency.** A large number of users may access the WebApp at one time.
- **Unpredictable load.** The number of users of the WebApp may vary by orders of magnitude from day to day.
- **Performance.** If a WebApp user must wait too long (for access, for server-side processing, for client-side formatting and display), he or she may decide to go elsewhere.
- **Availability.** Although expectation of 100 percent availability is unreasonable, users of popular WebApps often demand access on a "24/7/365" basis.

Characteristics of WebApps - II

- **Data driven.** The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user. Eg: e-commerce or financial application
- **Content sensitive.** The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.
- **Continuous evolution.** Unlike conventional application software that evolves over a series of planned, chronologically-spaced releases, Web applications evolve continuously.
- **Immediacy.** Although *immediacy*—the compelling need to get software to market quickly—is a characteristic of many application domains, WebApps often exhibit a time to market that can be a matter of a few days or weeks.
- **Security.** Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application.
- **Aesthetics.** An undeniable part of the appeal of a WebApp is its look and feel.

Software Engineering



1.5.2 General Principles

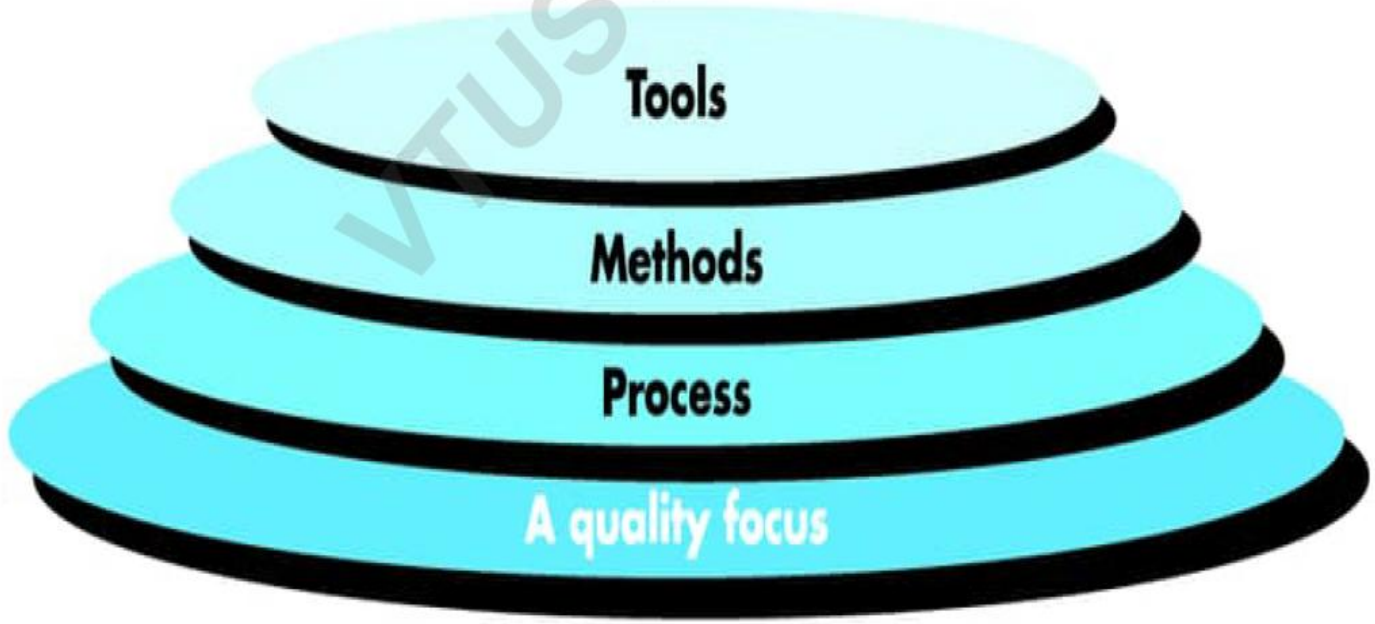
- The dictionary defines the word *principle as* "an important underlying law or assumption required in a system of thought."
- David Hooker [Hoo96] has proposed seven principles that focus on software engineering practice as a whole. They are

1. **The First Principle: *The Reason It All Exists***
   - A software system exists for one reason: *to provide value to its users.*
2. **The Second Principle: *KISS (Keep It Simple, Stupid!)***
   - *All design should be as simple as possible, but no simpler*
3. **The Third Principle: *Maintain the Vision***
4. **The Fourth Principle: *What You Produce, Others Will Consume***
5. **The Fifth Principle: *Be Open to the Future***

A system with a long lifetime has more value.

6. **The Sixth Principle: *Plan Ahead for Reuse***
   - *Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated.*
7. **The Seventh principle: *Think!***
   - *Placing clear, complete thought before action almost always produces better results.*

- And yet, a “systematic, disciplined, and quantifiable” approach applied by one software team may be burdensome to another.
- We need discipline, but we also need adaptability and agility.
- Software engineering is a layered technology



Software Engineering(Conti..)

- Any engineering approach (including software engineering) must rest on an **organizational commitment to quality.** The bedrock that supports software engineering is a quality focus.
- **The foundation for software engineering is the *process layer*.** Process defines a framework that must be established for effective delivery of software engineering technology.
- **The software process forms the basis for management control of software projects** and establishes 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 *methods* *provide the technical how-to's for building software.***

  Methods encompass a broad array of tasks that include **communication, requirements analysis, design modeling, program construction, testing, and support**.
- Software engineering ***tools*** ***provide automated or semi automated support for the* process and the methods**. Tools are integrated ,
- A system for the support of software development, called ***computer-aided software engineering, is established.***

1.4 The Software Process

A generic process framework for software engineering encompasses five activities:

- **Communication:** The intent is to understand stakeholders' objectives for the project and to gather requirements that help define software features and functions.
- **Planning:** the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule
- **Modeling:** models helps to better understand software requirements and the design that will achieve those requirements.
- **Construction:** This activity combines code generation (either manual or automated) and the testing that is required to uncover errors in the code.
- **Deployment:** The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.

- **Software project tracking and control**—allows the software team to assess progress against the project plan and take any necessary action to maintain the schedule.
- **Risk management**—assesses risks that may affect the outcome of the project or the quality of the product.
- **Software quality assurance**—defines and conducts the activities required to ensure software quality.
- **Technical reviews**—assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.

- **Measurement—**defines and collects process, project, and product measures that assist the team in delivering software that meets stakeholders' needs; can be used in conjunction with all other framework and umbrella activities.
- **Software configuration management—**manages the effects of change throughout the software process.
- **Reusability management—**defines criteria for work product reuse (including software components) and establishes mechanisms to achieve reusable components.
- **Work product preparation and production—**encompasses the activities required to create work products such as models, documents, logs, forms, and lists.

2.1 A Generic Process Model

Software Process includes:

Tasks: They focus on a small, specific objective.

Action: It is a set of tasks that produce a major work product.

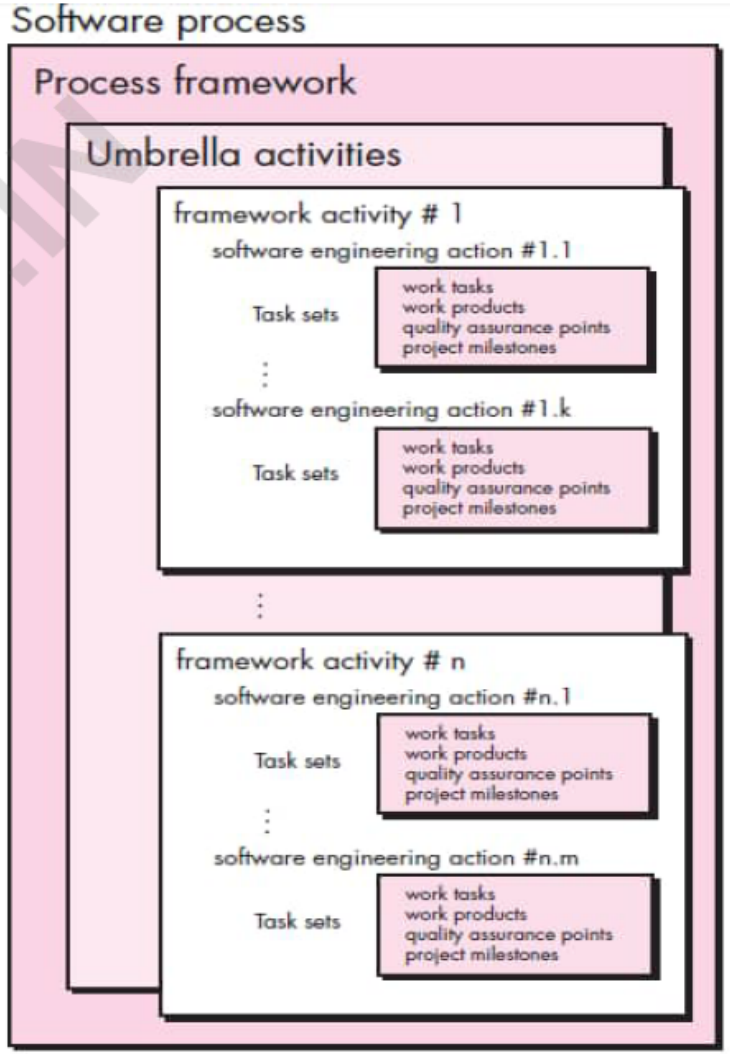
Activities: Activities are groups of related tasks and actions for a major objective.

Process Framework Activities

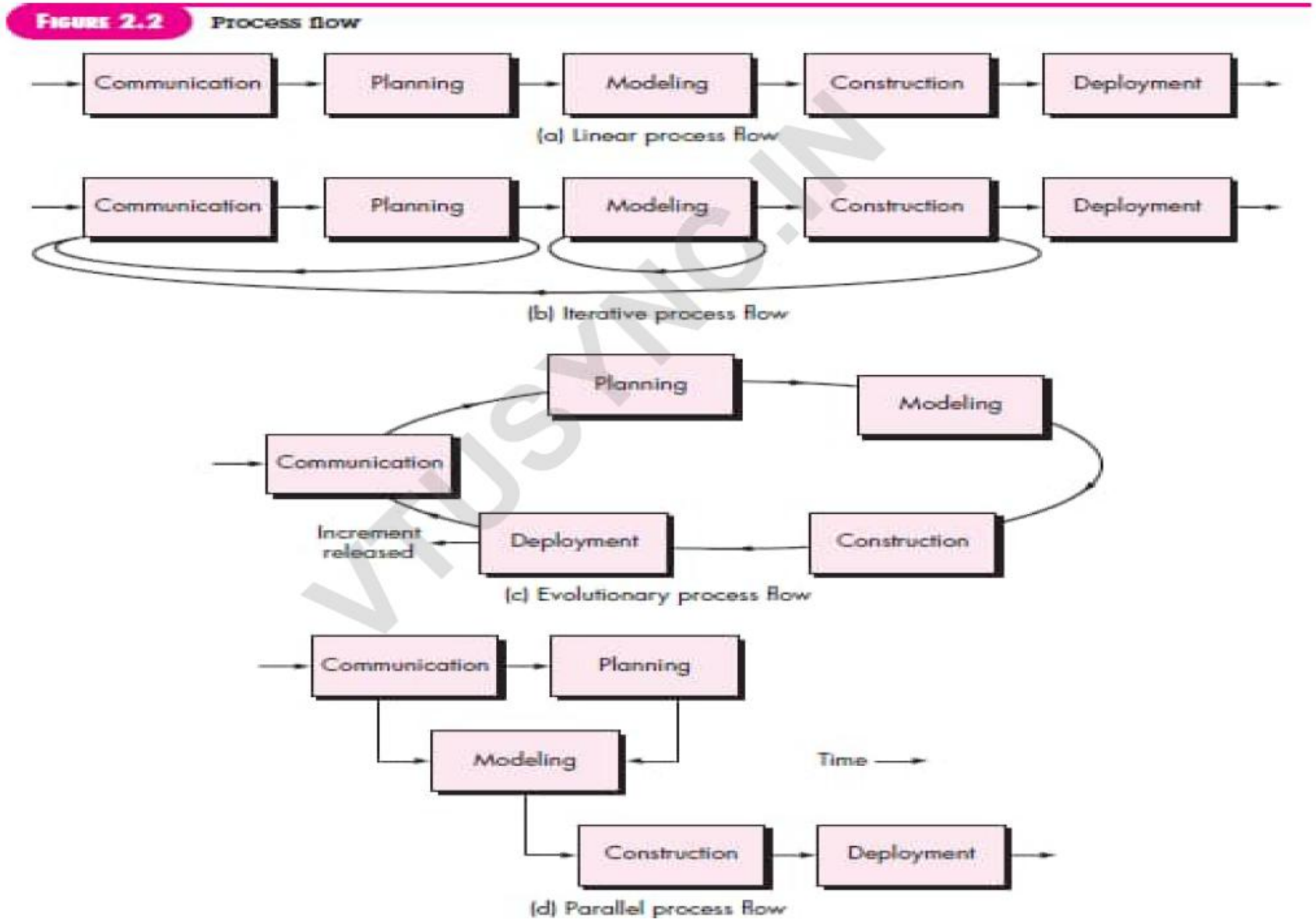
1. Communication
2. Planning
3. Modeling- Analysis of requirement, design
4. Construction- Coding, testing
5. Deployment

Umbrella Activities

1. Project tracking and control
2. Risk management
3. Quality assurance
4. Configuration management
5. Technical Review etc



Process Flow



Software Myths and Reality

- **Myth 1:**We have all the standards and procedures available for software development.
- **Fact**: all existing processes are incomplete as new software development is based on new and different problem.

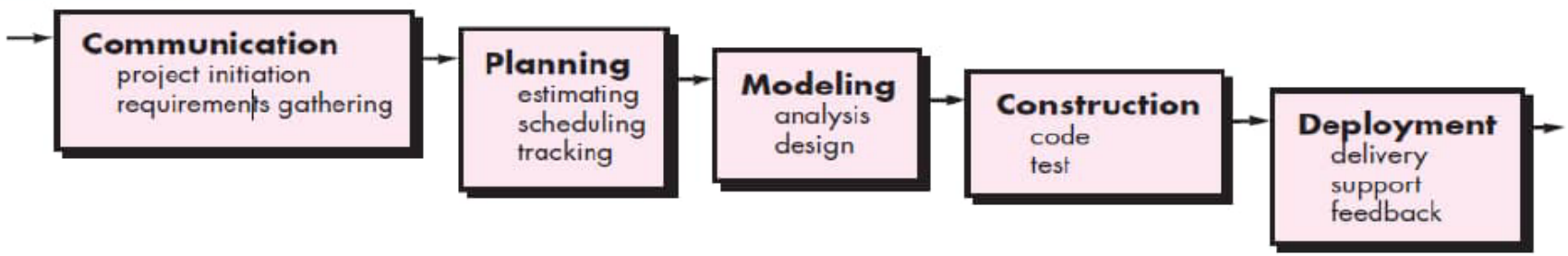
- **Myth 2:**With the addition of more people and program planners to Software development can help meet project deadlines (If lagging behind).
- **Fact**: Software is late, adding more people will merely make the problem worse. This is because the people already working on the project now need to spend time educating the newcomers and are thus taken away from their work.

- **Myth 3:**
- A general statement of intent is enough to start writing plans (software development) and details of objectives can be done over time.
- **Fact :** Unambiguous requirements (usually derived iteratively) are developed only through effective and continuous communication between customer and developer.
- **Myth 4:**
- Software requirements continually change, but change can be easily accommodated because software is flexible
- **Fact** : When requirements changes are requested early (before design or code has been started), the cost impact is relatively small. However, as time passes, the cost impact grows rapidly

2.3.1 Waterfall model

- It is used when the requirements for a problem are well understood—when work flows from **communication through deployment in a reasonably linear fashion.**
- It is only used when **requirements are well defined and reasonably stable.**
- The *waterfall model, sometimes called the classic life cycle, suggests a **systematic*** **sequential approach to software development** that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment.

FIGURE 2.3 The waterfall model



Waterfall model(Continued)

the problems that are sometimes encountered when the waterfall model is applied are:

1. **Real projects rarely follow the sequential flow that the model proposes.** Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds
2. **It is often difficult for the customer to state all requirements explicitly.** The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.
3. **The customer must have patience.** A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous.

Waterfall model(Continued)

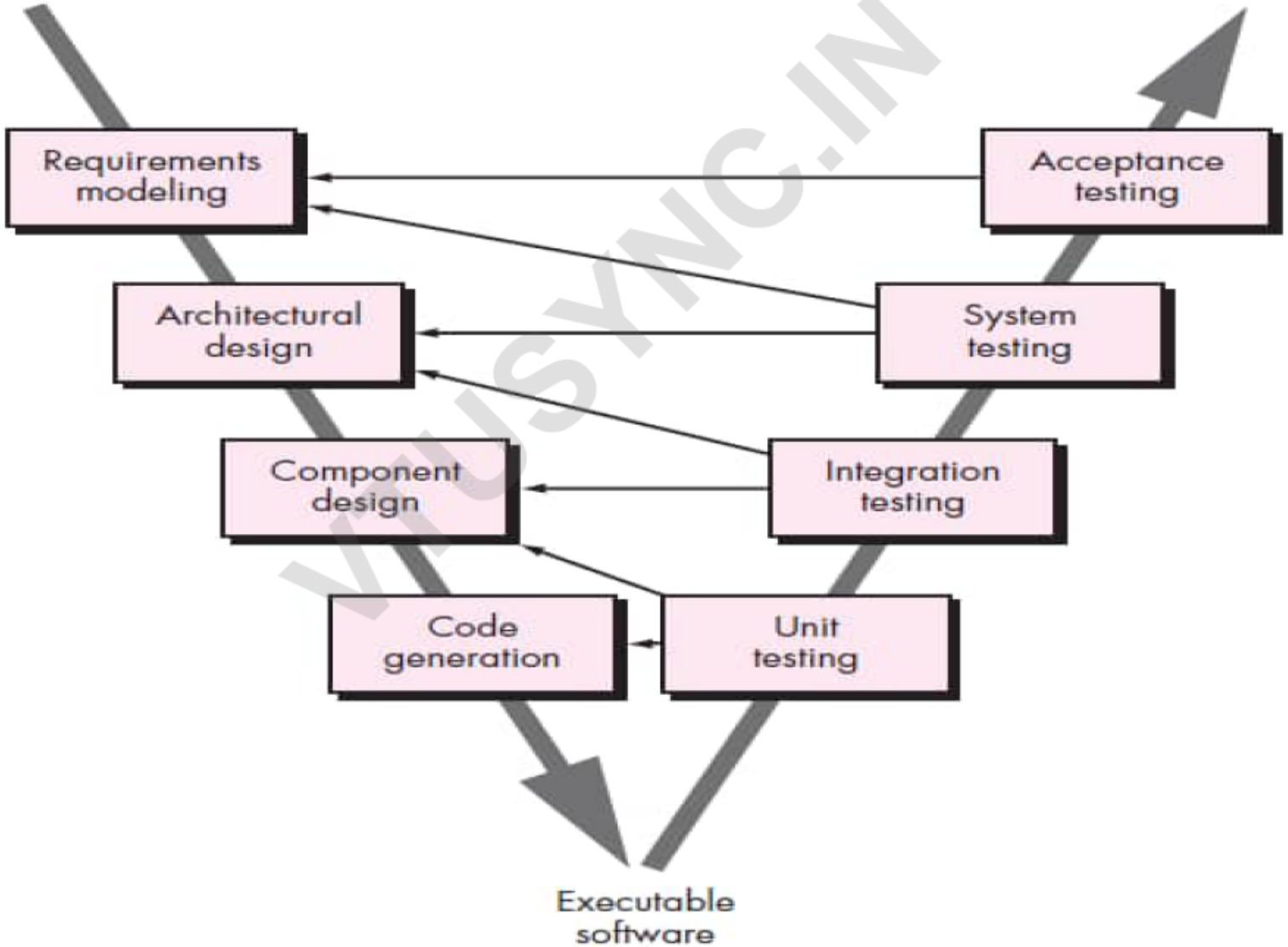
Advantages of waterfall model

- The waterfall model is simple and easy to understand, implement, and use.
- All the requirements are known at the beginning of the project, hence it is easy to manage.
- It avoids overlapping of phases because each phase is completed at once.
- This model works for small projects because the requirements are understood very well.
- This model is preferred for those projects where the quality is more important as compared to the cost of the project.

Disadvantages of the waterfall model

- This model is not good for complex and object oriented projects.
- It is a poor model for long projects.
- The problems with this model are uncovered, until the software testing.
- The amount of risk is high.

The V- model(Verification and validation model)



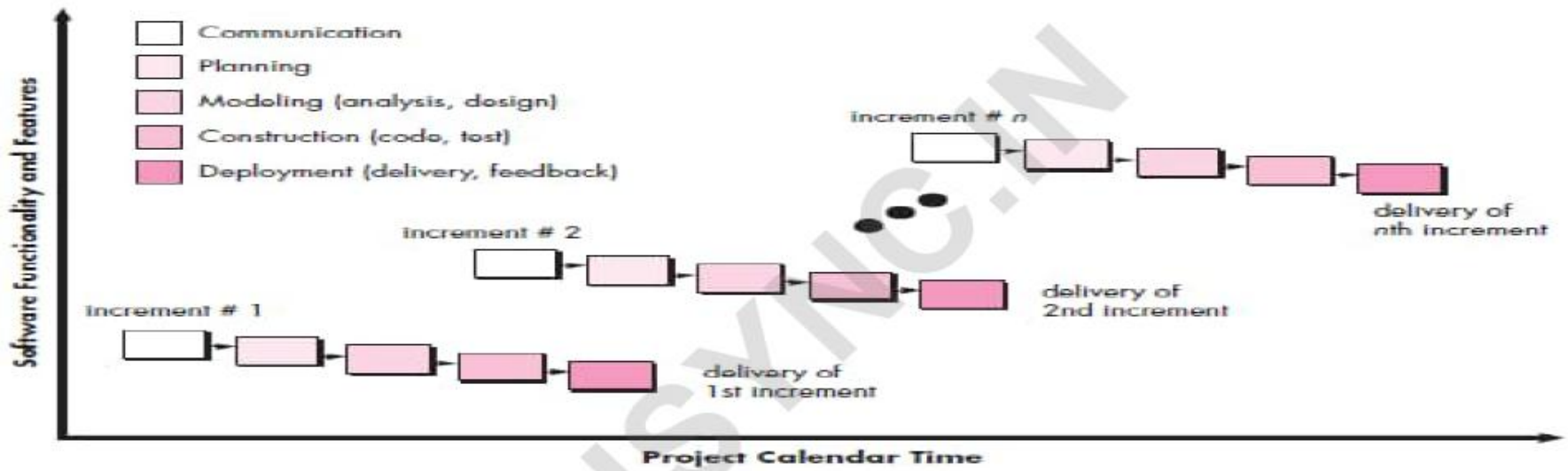
The V- model(Verification and validation model)

- A variation in the representation of the waterfall model is called the *V-model.*
- the V-model depicts the relationship of quality assurance actions to the actions associated with communication, modeling, and early construction activities.
- As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution.
- Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moved down the left side

2.3.2 Incremental Process Models

- There are many situations in which initial software requirements are reasonably well defined, but the overall scope of the development effort precludes a purely linear process.
- There may be 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, you can choose a process model that is designed to produce the software in increments.
- The *incremental model combines elements of linear and parallel process flows*
- For example, word-processing software(I incriment……. Iv incriment)
- When an incremental model is used, the first increment is often a *core product.*
- That is, basic requirements are addressed but many supplementary features (some known, others unknown) remain undelivered.
- The core product is used by the customer(or undergoes detailed evaluation).

Incremental Process Models(Conti)



- **Incremental development is particularly useful when staffing is unavailable** for a complete implementation by the business deadline that has been established for the project. Solution....
- **In addition, increments can be planned to manage technical risks**. For example, a major system might require the availability of new hardware that is under development and whose delivery date is uncertain. Solution: .....

Incremental Process Models(Conti..)

Advantages of incremental model

- This model is flexible because the cost of development is low and initial product delivery is faster.
- It is easier to test and debug during the smaller iteration.
- The working software generates quickly and early during the software life cycle.
- The customers can respond to its functionalities after every increment.

Disadvantages of the incremental model

- The cost of the final product may cross the cost estimated initially.
- This model requires a very clear and complete planning.
- The planning of design is required before the whole system is broken into small increments.
- The demands of customer for the additional functionalities after every increment causes problem during the system architecture.

2.3.3 Evolutionary Process Models

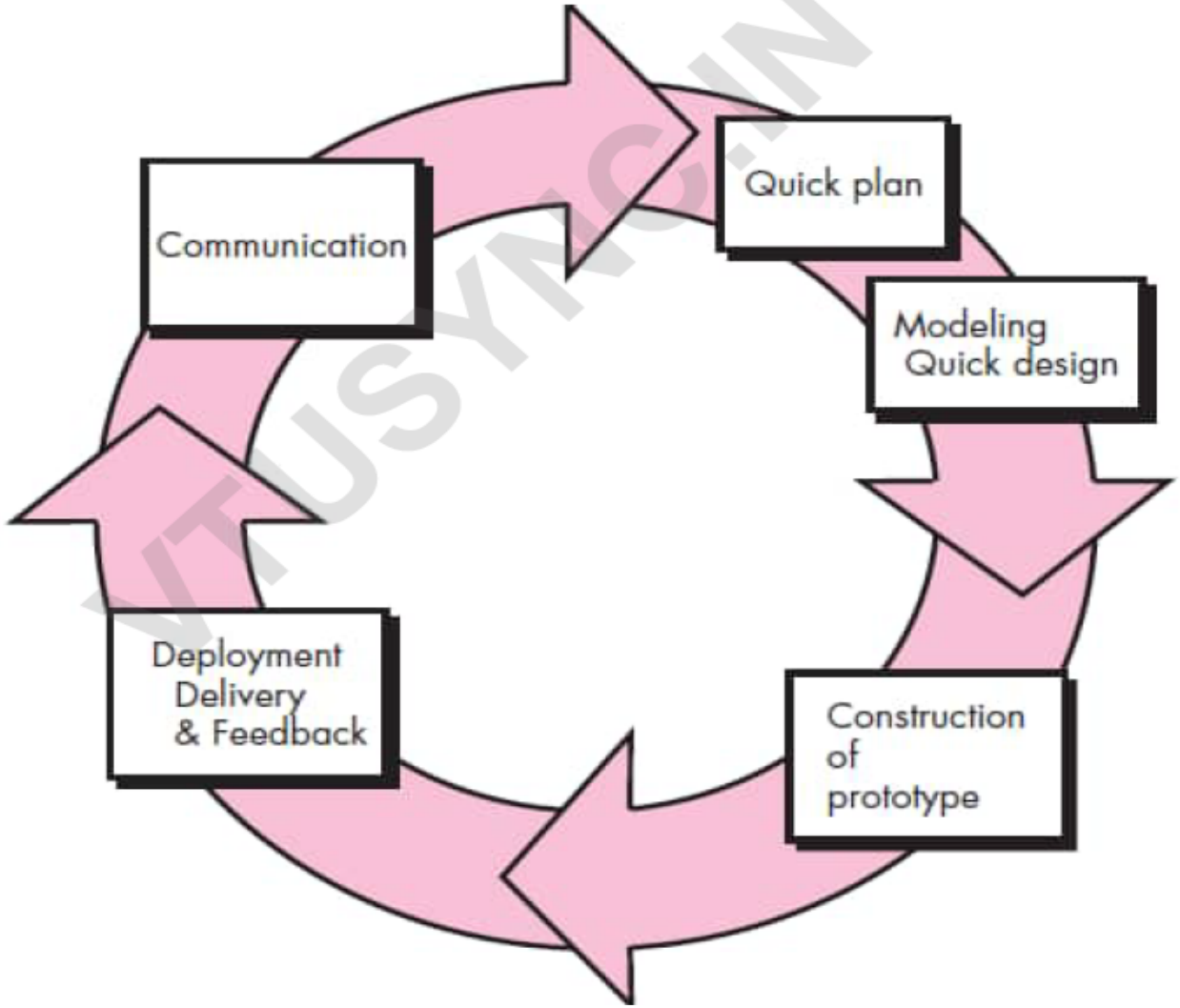
- Software, like all complex systems, evolves over a period of time.

Factors

1. **Business and product requirements often change as development proceeds, making a straight line path to an end product unrealistic.**
2. **tight market deadlines; a limited version must be introduced to meet competitive or business pressure.**
3. **a set of core product or system requirements is well understood.**

- you need a process model that has been explicitly designed to accommodate a product that evolves over time.
- Evolutionary models are iterative.
- They are characterized in a manner that enables you to develop increasingly more complete versions of the software.

Evolutionary Process Models (Prototyping)



Evolutionary Process Models (Prototyping)

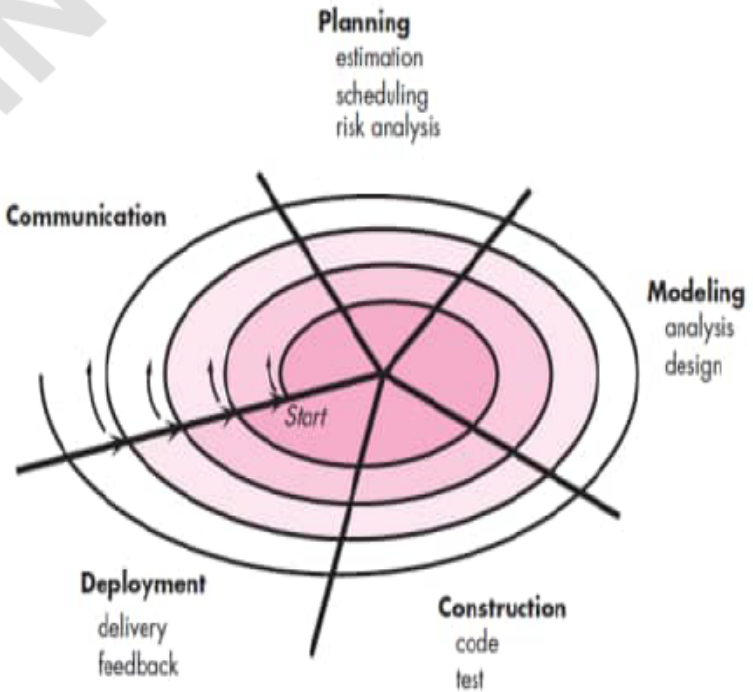
- prototyping can be problematic for the following reasons:

1. Stakeholders see what appears to be a working version of the software.
2. As a software engineer, you often make implementation compromises in order to get a prototype working quickly.

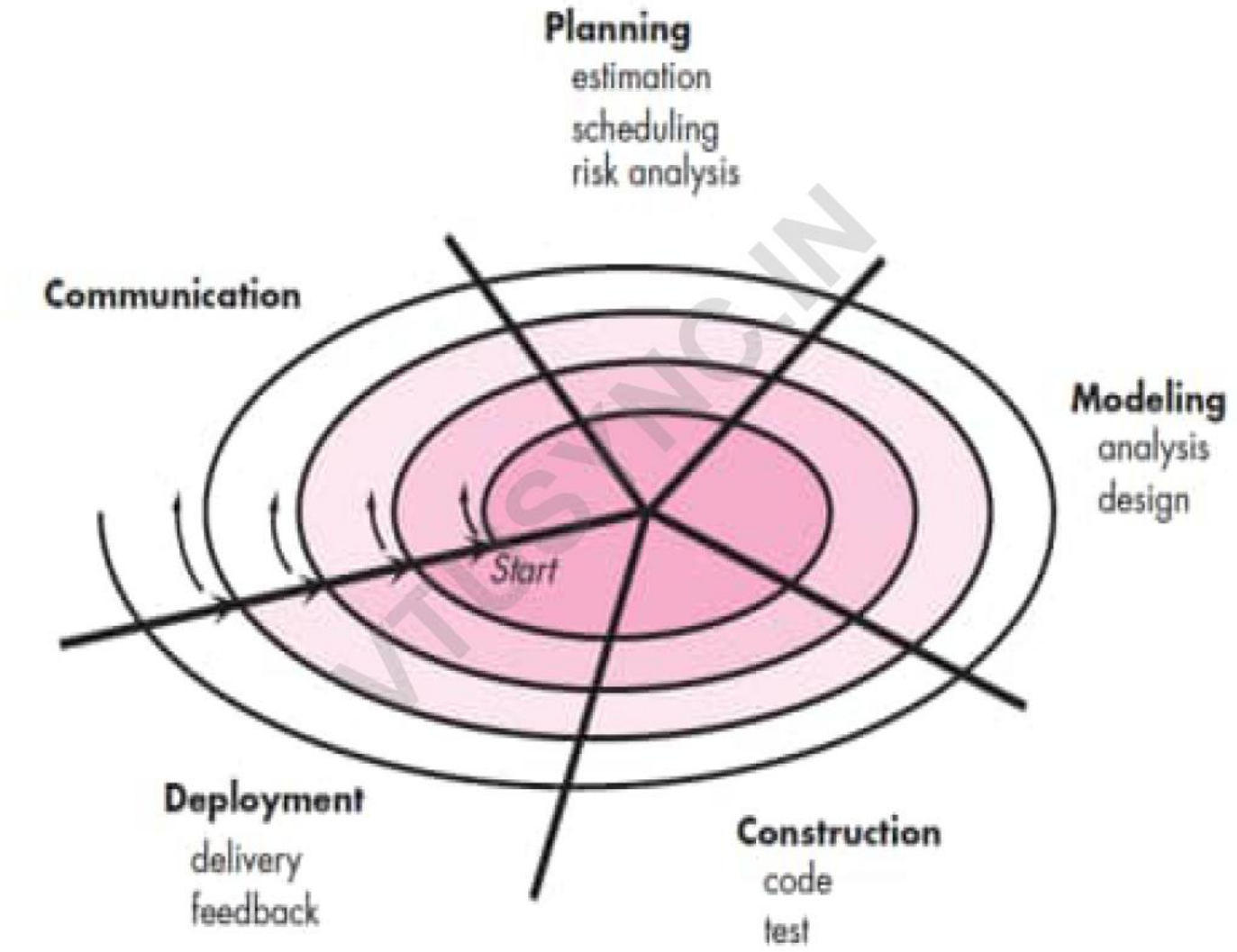
- The key is to define the rules of the game at the beginning;
- that is, all stakeholders should agree that the prototype is built to serve as a mechanism for defining requirements.
- It is then discarded (at least in part), and the actual software is engineered with an eye toward quality

Evolutionary Process Models(The Spiral model)

- couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.
- It provides the potential for rapid development of increasingly more complete versions of the software.
- The fig
- Each pass through the planning region results in adjustments to the project plan.
- Cost and schedule are adjusted based on feedback derived from the customer after delivery.
- In addition, the project manager adjusts the planned number of iterations required to complete the software.



A typical spiral model


Planning
estimation
scheduling
risk analysis
Communication
Modeling
analysis
design
Start
Deployment
delivery
feedback
Construction
code
test

Evolutionary Process Models(The Spiral model)

ADVANTAGES

- The spiral model is a realistic approach to the development of large-scale systems and software,
- The spiral model uses prototyping as a risk reduction mechanism
- Enables you to apply the prototyping approach at any stage in the evolution of the product.
- The spiral model demands a direct consideration of technical risks at all stages of the project and, if properly applied, should reduce risks before they become problematic.

DISADVANTAGES

- the evolutionary approach is controllable.
- It demands considerable risk assessment expertise
- If a major risk is not uncovered and managed, problems will undoubtedly occur.

2.3.4 Concurrent Models

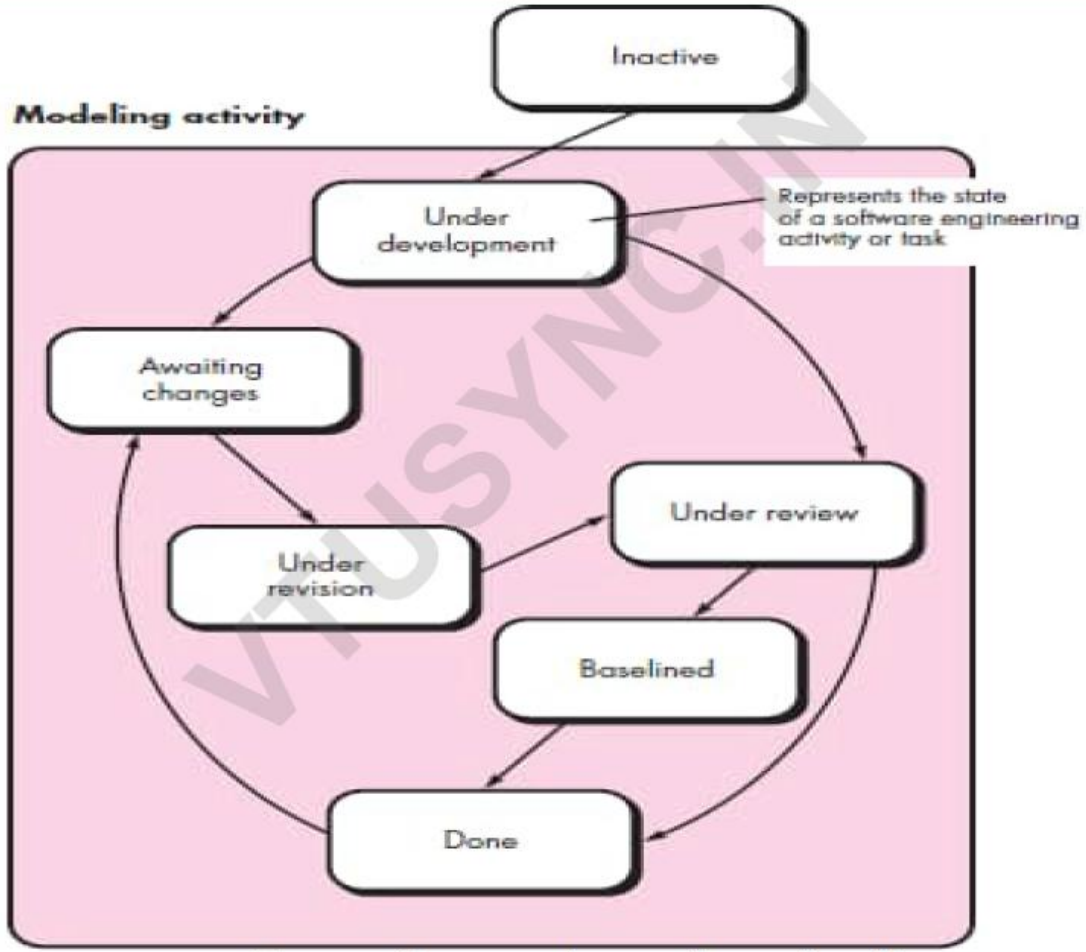


Fig: One element of the concurrent process model

2.4 SPECIALIZED PROCESS MODELS

these models tend to be applied when a specialized or narrowly defined software engineering approach is chosen.

I. Component-Based Development

- Commercial off-the-shelf (COTS) software components, developed by vendors who offer them as products, provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software that is to be built.
- The *component-based development model incorporates many of the characteristics of the spiral model.*
- *It is evolutionary in nature, demanding an iterative approach to the creation of software. However, the component-based development model constructs applications from prepackaged software components.*
- Modeling and construction activities begin with the identification of candidate components.

SPECIALIZED PROCESS MODELS(Conti..)

model incorporates the following steps:

1. Available component-based products are researched and evaluated for the application domain in question.
2. Component integration issues are considered.
3. A software architecture is designed to accommodate the components.
4. Components are integrated into the architecture.
5. Comprehensive testing is conducted to ensure proper functionality.

- leads to software reuse, and reusability provides software engineers with a number of measurable benefits.

SPECIALIZED PROCESS MODELS(Conti..)

II. The Formal Methods Model

- encompasses a set of activities that leads to formal mathematical specification of computer software.
- Formal methods enable you to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.
- A variation on this approach, called *clean room software engineering* is currently applied by some software development organizations.
- When formal methods are used during development, they provide a mechanism for eliminating many of the problems that are difficult to overcome using other software engineering paradigms.
- When formal methods are used during design, they serve as a basis for program verification and therefore enable you to discover and correct errors that might otherwise go undetected.
- concern about its applicability in a business environment has been voiced:

1. The development of formal models is currently quite time consuming and expensive.
2. Because few software developers have the necessary background to apply formal methods, extensive training is required.
3. It is difficult to use the models as a communication mechanism for technically unsophisticated customers.

Formal methods model applications

- **Aerospace (Avionics):** Ensuring the safe and correct operation of aircraft systems.
- **Medical Devices:** Validating the reliability of software in life-critical medical applications.
- **Security-Critical Systems:** Eliminating vulnerabilities in systems where security breaches could have severe impacts.

Aspect-Oriented Programming (AOP)

- AOP is a programming technique that focuses on modularizing cross-cutting concerns, which are features that cut across different parts of an application. Cross-cutting concerns include things such as logging, security, performance monitoring, error handling, and more.
- **Security:**
- **Transaction Management:**
- **Telecommunication Systems**:
- **E-commerce Platforms**:

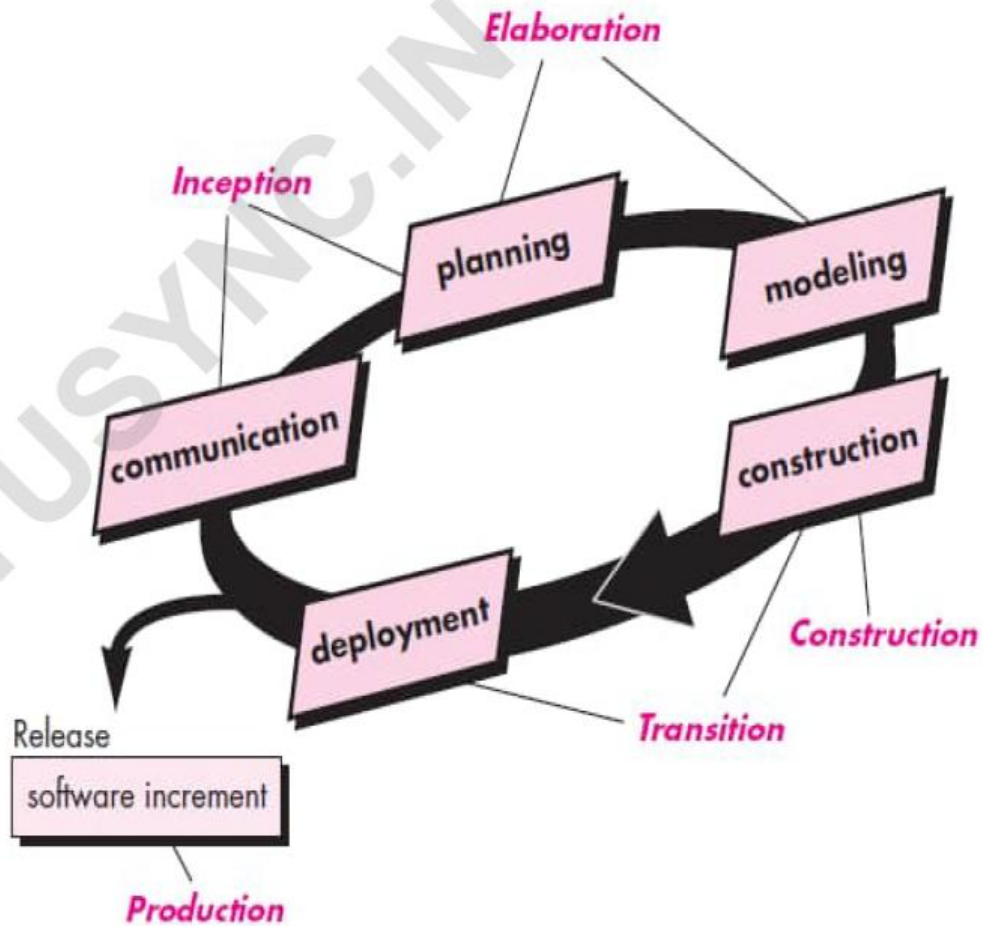
2.5 The Unified Process

- In their seminal book on the *Unified Process, Ivar Jacobson, Grady Booch, and James* Rumbaugh [Jac99] discuss the need for a "use case driven, architecture-centric, iterative and incremental"
- Unified Process is an attempt to draw on the best features and characteristics of traditional software process models, but implements many of the best principles of agile software development
- The Unified Process recognizes the importance of customer communication and streamlined methods for describing the customer's view of a system (the use case).
- It emphasizes the important role of software architecture and helps the architect focus on the right goals, such as understandability, reliance to future changes, and reuse
- It suggests a process flow that is iterative and incremental, providing the evolutionary feel that is essential in modern software development.

2.5.2 Phases of the Unified Process

FIGURE 2.9

The Unified Process



2.5.2 Phases of the Unified Process (Conti..)

1. **The *inception phase*** *of the UP encompasses both customer communication and* planning activities.

- By collaborating with stakeholders, business requirements for the software are identified; a rough architecture for the system is proposed; and a plan for the iterative, incremental nature of the ensuing project is developed.
- Fundamental business requirements are described through a set of preliminary use cases that describe which features and functions each major class of users desires.
- Architecture at this point is nothing more than a tentative outline of major subsystems and the function and features that populate them.
- Later, the architecture will be refined and expanded into a set of models that will represent different views of the system.
- Planning identifies resources, assesses major risks, defines a schedule, and establishes a basis for the phases that are to be applied as the software increment is developed.

2.5.2 Phases of the Unified Process (Conti..)

2. The ***elaboration phase*** *encompasses the communication and modeling activities of* the generic process model

- Elaboration refines and expands the preliminary use cases that were developed as part of the inception phase and expands the architectural representation to include five different views of the software—the use case model, the requirements model, the design model, the implementation model, and the deployment model.
- In some cases, elaboration creates an "executable architectural baseline" [Arl02] that represents a "first cut" executable system. The
- The plan is carefully reviewed at the culmination of the elaboration phase to ensure that scope, risks, and delivery dates remain reasonable.

2.5.2 Phases of the Unified Process (Conti..)

3. The *construction phase* *of the UP is identical to the construction activity defined* for the generic software process.

- Using the architectural model as input, the construction phase develops or acquires the software components that will make each use case operational for end users.
- To accomplish this, requirements and design models that were started during the elaboration phase are completed to reflect the final version of the software increment.
- All necessary and required features and functions for the software increment (i.e., the release) are then implemented in source code.
- As components are being implemented, unit tests are designed and executed for each.
- Integration activities (component assembly and integration testing) are conducted.
- Use cases are used to derive a suite of acceptance tests that are executed prior to the initiation of the next UP phase.

2.5.2 Phases of the Unified Process (Conti..)

4. The *transition phase:* *of the UP encompasses the latter stages of the generic construction* activity and the first part of the generic deployment (delivery and feedback) activity.

- Software is given to end users for beta testing and user feedback reports both defects and necessary changes.
- The software team creates the necessary support information (e.g., user manuals, troubleshooting guides, installation procedures) that is required for the release.

5. The *production phase :* *of the UP coincides with the deployment activity of the* generic process.

- The ongoing use of the software is monitored, support for the operating environment (infrastructure) is provided, and defect reports and requests for changes are submitted and evaluated.

Personal and Team Process Models

- The best software process is one that is close to the people who will be doing the work
- In an ideal setting, you would create a process that best fits your needs, and at the same time, meets the broader needs of the team and the organization
- Watts Humphrey ([Hum97] and [Hum00]) argues that it is possible to create a "personal software process" and/or a "team software process." Both require hard work, training, and coordination, but both are achievable

1. **Personal Software Process (PSP):** emphasizes personal measurement of both the work product that is produced and the resultant quality of the work product. The PSP model defines five framework activities:

a. **Planning** b. **High level design** c. **High level design review** d. **Development** e. **Postmortem**

Personal and Team Process Models(Conti..)

PSP (Conti..)

a. **Planning:** This activity isolates requirements and develops both size and resource estimates, a defect estimate is made. Finally, development tasks are identified and a project schedule is created.

b. **High-level design:** External specifications for each component to be constructed are developed and a component design is created. Prototypes are built when uncertainty exists. All issues are recorded and tracked.

c. **High-level design review**: Formal verification methods are applied to uncover errors in the design. Metrics are maintained for all important tasks and work results.

d. **Development:** The component-level design is refined and reviewed. Code is generated, reviewed, compiled, and tested. Metrics are maintained …

e. **Postmortem:** Using the measures and metrics collected (this is a substantial amount of data that should be analyzed statistically), the effectiveness of the process is determined. Measures and metrics should provide guidance for modifying the process to improve its effectiveness.

Personal and Team Process Models(Conti..)

(PSP Continued..)

- PSP stresses the need to identify errors early and, just as important, to understand the types of errors that you are likely to make.
- PSP represents a disciplined, metrics-based approach to software engineering that may lead to culture shock for many practitioners.
- when PSP is properly introduced to software engineers [Hum96], the resulting improvement in software engineering productivity and software quality are significant.
- PSP is intellectually challenging andm demands a level of commitment (by practitioners and their managers) that is not always possible to obtain.
- Training is relatively lengthy, and training costs are high.

Personal and Team Process Models

2. **Team Software Process (TSP):** The goal of TSP is to build a "self directed" project team that organizes itself to produce high-quality software. Humphrey [Hum98] defines the following objectives for TSP:

a) Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPTs) of 3 to about 20 engineers.

b) Show managers how to coach and motivate their teams and how to help them sustain peak performance.

c) Accelerate software process improvement by making CMM Level 5 behavior normal and expected.

d) Provide improvement guidance to high-maturity organizations.

e) Facilitate university teaching of industrial-grade team skills

- TSP defines the following framework activities: **project launch, high-level design, implementation, integration and test, and postmortem.**

Team Software Process (TSP) (Conti)

- these activities enable the team to plan, design, and construct software in a disciplined manner while at the same time quantitatively measuring the process and the product. The postmortem sets the stage for process improvements.
- TSP makes use of a wide variety of scripts, forms, and standards that serve to guide team members in their work.
- TSP recognizes that the best software teams are self-directed.
- Like PSP, TSP is a rigorous approach to software engineering that provides distinct and quantifiable benefits in productivity and quality.
- The team must make a full commitment to the process and must undergo thorough training to ensure that the approach is properly applied.

THANK YOU