MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

CS233:Software Engineering and Project Management

Computer Science and Engineering

S.Y. Trimester III

CS233:Software Engineering and Project Management

Examination Scheme :

Continuous Assessment: 50 Marks End Semester Examination :50 Marks Credit:2

- **Course Objectives:**

1. To identify the systematic way of Software Development and Software Lifecycle Process model.
2. To identify the principles of Agile Software Development and Practices.
3. To explain the Project Management principles and Project Metrics

- **Course Outcomes:**

After completion of this course students will be able to:

1. Analyze process models and its appropriate selection for Development of Software Projects.
2. Define scope of Project, design Software Requirement Specifications (SRS) and plan a project.
3. Make use of the principles and processes of Agile Methodology.
4. Choose modern tools for Software Development and Project Management

Pre-requisites

1. Principles of Programming Languages and Functional Programming
2. Object Oriented Programming

Syllabus

SOFTWARE PROCESS

Introduction to Software Engineering, Importance of Software Engineering, Software Engineering Practice, Software Myths, Software Process, Perspective and Specialized Process Models. Case Studies.

REQUIREMENT ENGINEERING

Software Requirements, User requirements, System requirements, Software Requirements Specification, Requirement Engineering Process, Software Design: Abstraction, Modularity, Cohesion & Coupling, Scenario based modeling, SSAD (ER diagram, Data Flow Diagram DFD), OOAD, Unified Modeling Language (UML) Case Studies.

AGILE MANAGEMENT AND PRACTICES

Agile manifesto, The Fundamentals of Agile Software Development, Aspects of Agile Approaches, Practices and Processes, Techniques in Agile Projects, Extreme Programming, Other process models of Agile Development and Tools.

Tools in Agile Projects-. Case Studies

SOFTWARE PROJECT MANAGEMENT

Project Management Principles , Process and Project Metrics., Function Point analysis, LOC , Make/Buy Decision, COCOMO II -Project Planning, SWOT analysis, Functions of manager, Team building & development, Risk Management, CPM/PERT, Case Studies and Example

Unit 1. SOFTWARE PROCESS

- Introduction to Software Engineering
- Importance of Software Engineering
- Software Engineering Practice
- Software Myths
- Software Process
- Software Perspective and Specialized Process Models.
- Case Studies

What is Software?

Software is: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately manipulate information and (3) documentation that describes the operation and use of the programs.

The IEEE definition of Software Engineering

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

Without software, most computers would be useless. For example, without your

internet browser software, you could not surf the Internet or read this page.

Without an operating system, the browser could not run on your computer

Software

How do you get software?

- Software can be purchased at a retail computer store or online and come in a box containing all the disks (floppy diskette, CD, DVD, or Blu-ray), manuals, warranty, and other documentation.
- Software can also be downloaded to a computer over the Internet. Once downloaded, setup files are run to start the installation process on your computer.

Free software : There are also a lot of free software programs available that are separated into different categories.

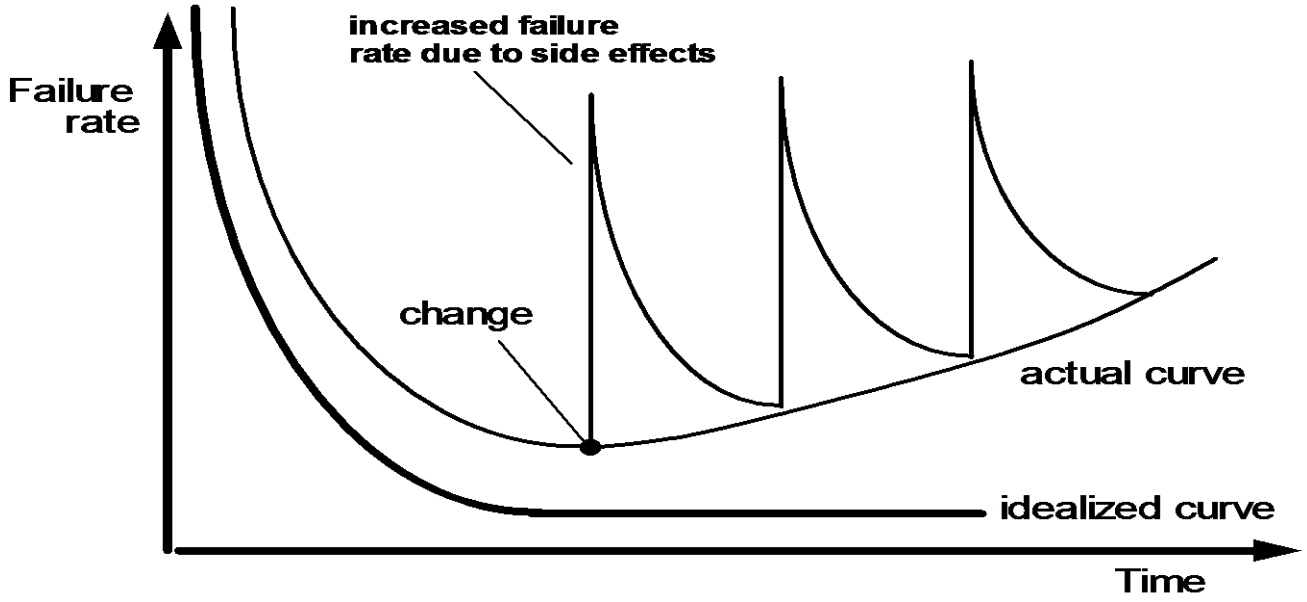
- **Shareware or trial software** is software that gives you a few days to try the software before you have to buy the program. After the trial time expires, you'll be asked to enter a code or register the product before you can continue to use it.
- **Freeware** is completely free software that never requires payment, as long as it is not modified.
- **Open source software** is similar to freeware. Not only is the program given away for free, but the source code used to make the program is as well, allowing anyone to modify the program or view how it was created.

Examples

Software	Examples
Antivirus	AVG, Housecall, McAfee, and Norton.
Audio / Music program	iTunes and WinAmp.
Database	Access, MySQL, and SQL.
Device drivers	Computer drivers.
E-mail	Outlook and Thunderbird.
Game	Madden NFL football, Quake, and World of Warcraft.
Internet browser	Firefox, Google Chrome, and Internet Explorer.
Movie player	VLC and Windows Media Player.
Operating system	Android, iOS, Linux, macOS, and Windows.

Characteristics of Software

- Software is developed or engineered, it is not manufactured in the classical sense.
- Software doesn't "wear out."
- Although the industry is moving toward component-based construction, most software continues to be custom-built.



Software Applications

1. **System software**: System software serves as the interface between the hardware and the end users. Some examples of system software are Operating System, Compilers, Interpreter, Assemblers, etc.

2. **Application software:** e.g. Payroll Software, Student Record Software, Inventory Management Software, Income Tax Software, Railways Reservation Software, Microsoft Office Suite Software, Microsoft Word

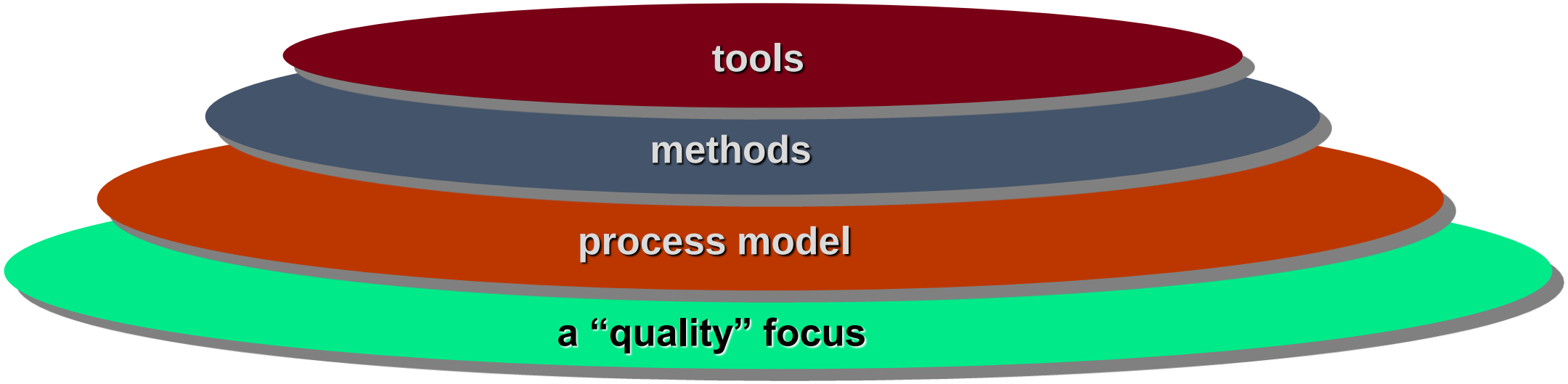
3. **Engineering/scientific software:** e.g. CAD/CAM, automatic stress analysis, nuclear biology, Space shuttle

4.**Embedded software:** Embedded software is a piece of software that is **embedded in hardware or non-PC devices**. It is written specifically for the particular hardware that it runs on and usually has **processing and memory constraints** because of the device's limited computing capabilities e.g. s/w used in mobiles, micro oven, car locking etc

5.**WebApps (Web applications):** client–server computer program which the runs in a web browser. Common web applications include webmail, online retail sales, and online auction.

6. **AI software:** e.g. Robotics, Expert system, pattern recognition, image voice, ANN

Software :A Layered Technology



Software Engineering

A Process Framework

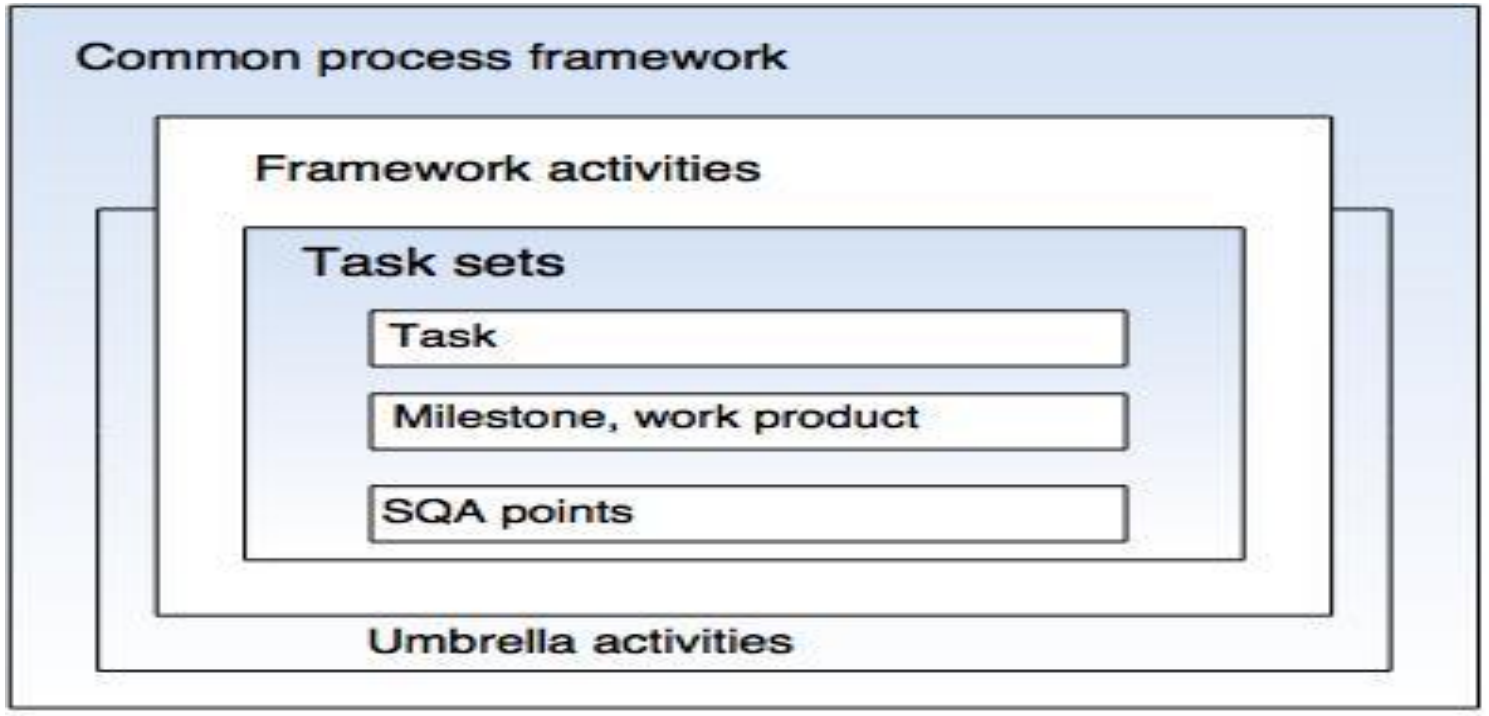


Fig.- A software process framework

Framework Activities

- Communication
- Planning
- Modeling
    - Analysis of requirements
    - Design
- Construction
    - Code generation
    - Testing
- Deployment

Umbrella Activities

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Work product preparation and production
- Reusability management
- Measurement
- Risk management

Identifying a Task Set

- Before you can proceed with the process model, a key question: what **actions** are appropriate for a framework activity

given the nature of the problem, the characteristics of the people and the stakeholders?

- **A task set defines the actual work to be done** to accomplish the objectives of a software engineering action.
  - A list of the task to be accomplished
  - A list of the work products to be produced
  - A list of the quality assurance filters to be applied

Identifying a Task Set

- For example, a small software project requested by one person with simple requirements, the communication activity might encompass little more than a phone all with the stakeholder. The work tasks of this action are:

  1. Make contact with stakeholder via telephone.
  2. Discuss requirements and take notes.
  3. Organize notes into a brief written statement of requirements.
  4. E-mail to stakeholder for review and approval.

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

Management Myths

- **Myth:** The members of an organization can acquire all-the information, they require from a manual, which contains standards, procedures, and principles.
- **Reality**:
  - 1. Standards are often incomplete, inadaptable, and outdated.
  - 2. Developers are often **unaware** of all the established standards.
  - 3.Developers rarely follow all the known standards because not all the standards tend to decrease the delivery time of software while maintaining its quality.
- **Myth:** If the project is behind schedule, increasing the number of programmers can reduce the time gap.
- **Reality:**
  - 1. Adding more manpower to the project, which is already behind schedule, further delays the project.
  - 2.New workers take longer to learn about the project as compared to those already working on the project.

Management Myths

- **Myth:** If the project is outsourced to a third party, the management can relax and let the other firm develop software for them.
- **Reality:** Outsourcing software to a third party does not help the organization, which is incompetent in managing and controlling the software project internally. The organization invariably suffers when it out sources the software project.

User Myths

- **Myth:** Brief requirement stated in the initial process is enough to start development; detailed requirements can be added at the later stages.
- **Reality:**

  1.Starting development with incomplete and ambiguous requirements often lead to software failure. Instead, a complete and formal description of requirements is essential before starting development.

  2.Adding requirements at a later stage often requires repeating the entire development process.

- **Myth:** Software is flexible; hence software requirement changes can be added during any phase of the development process.
- **Reality**: Incorporating change requests earlier in the development process costs lesser than those that occurs at later stages. This is because incorporating changes later may require **redesigning** and **extra resources**.

Developer Myths

- **Myth:** Software development is considered complete when the code is delivered.
- **Reality:** 50% to 70% of all the efforts **are expended** after the software is delivered to the user.
- **Myth:** The success of a software project depends on the quality of the product produced.
- **Reality:** The quality of programs is not the only factor that makes the project successful instead the documentation and software configuration also playa crucial role.
- **Myth:** Software engineering requires unnecessary documentation, which slows down the project.
- **Reality**: Software engineering is about creating quality at every level of the software project. Proper documentation enhances quality which results in reducing the amount of rework.

Developer Myths

- **Myth:** The only product that is delivered after the completion of a project is the working program(s).
- **Reality:** The deliverables of a successful project includes not only the working program but also the documentation to guide the users for using the software.
- **Myth:** Software quality can be assessed only after the program is executed.
- **Reality:** The quality of software can be measured during any phase of development process by applying some quality assurance mechanism. One such mechanism is formal technical review that can be effectively used during each phase of development to uncover certain errors.

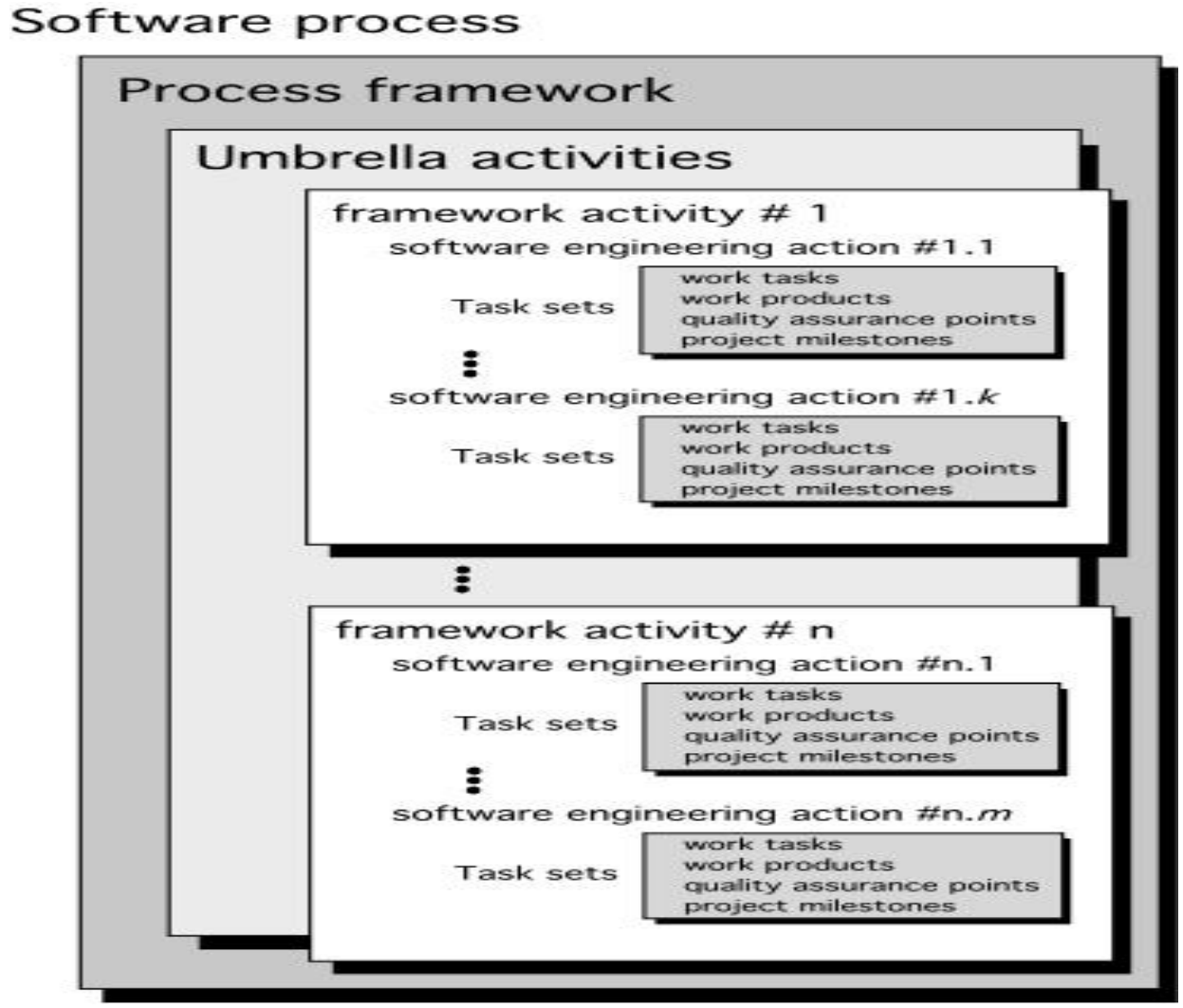
The software process

- A **structured set of activities** required to develop a software system.
- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Definition of Software Process

- A **framework** for the activities, actions, and tasks that are required to build high-quality software.
- SP defines the approach that is taken as software is engineered.
- Is not equal to software engineering, which also encompasses **technologies** that populate the process– technical methods and automated tools.

A Generic Process Model



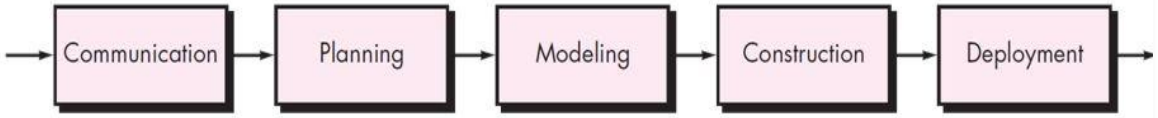
A Generic Process Model

- A generic process framework for software engineering defines **five framework activities-communication, planning, modeling, construction, and deployment.**
- In addition, a set of umbrella activities- project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others are applied throughout the process.
- Next question is: how the framework activities and the actions and tasks that occur within each activity are organized **with respect to sequence and time**? See the **process flow** for answer.

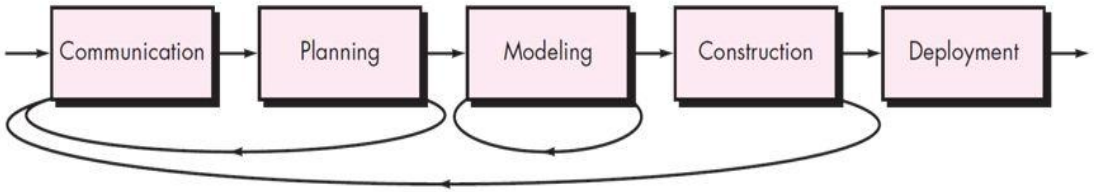
Process Flow

Process Flow

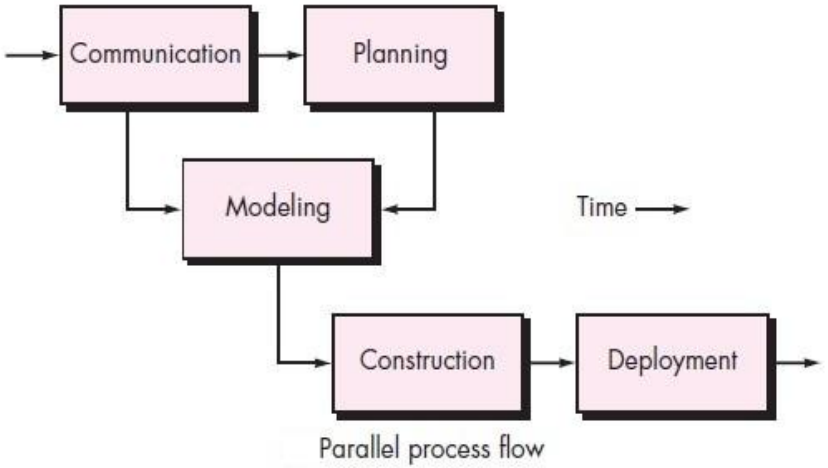
- Linear process flow



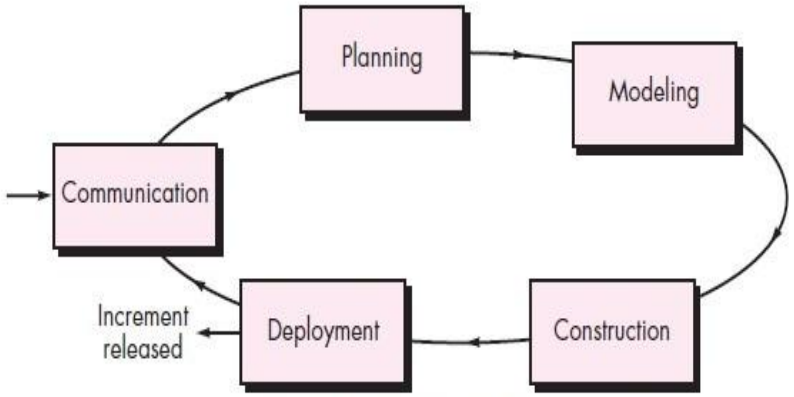
- Iterative process flow



3



Parallel process flow



Evolutionary process flow

Process Flow

- Linear process flow executes each of the five activities in sequence.
- An iterative process flow repeats one or more of the activities before proceeding to the next.
- An evolutionary process flow executes the activities in a circular manner. Each circuit leads to a more complete version of the software.
- A parallel process flow executes one or more activities in parallel with other activities (modeling for one aspect of the software in parallel with construction of another aspect of the software.

Adapting a Process Model

- the overall flow of **activities, actions, and tasks** and the interdependencies among them
- the degree to which actions and tasks are defined within each framework activity
- the degree to which work products are identified and required
- the manner which quality assurance activities are applied
- the manner in which project tracking and control activities are applied
- the overall degree of detail and rigor with which the process is described
- the degree to which the customer and other stakeholders are involved with the project
- the level of autonomy given to the software team
- the degree to which team organization and roles are prescribed

Prescriptive Models & Specialized Process Models

- .**WATERFALL MODEL**

  INCREMENTAL PROCESS MODEL

  - The Incremental Model
  - The RAD Model

  - **EVOLUTIONARY PROCESS MODELS**
    - Prototyping.
    - The Spiral model
    - The Concurrent Development Model

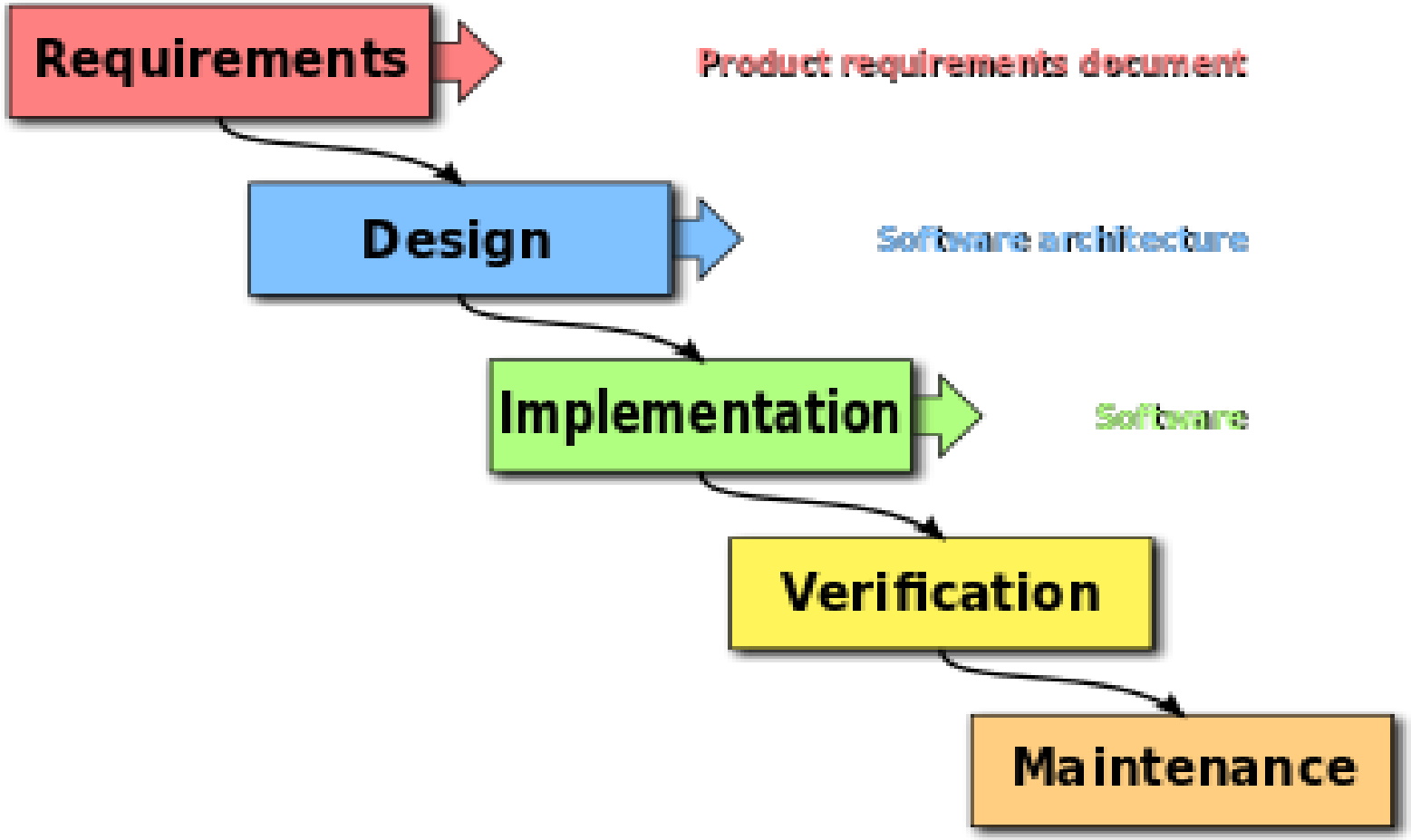
  SPECIALIZED PROCESS MODELS.

    - Component-based Development
    - The Formal Methods Model
    - Aspect-Oriented Software Development

  THE UNIFIED PROCESS.

  AGILE PROCESS

The Waterfall Model



The Waterfall Model

1) **Requirement Gathering and analysis**: This is the first phase of water-fall model. In this phase requirements of the system or project that is to be developed are captured from client and all these requirements are documented in a requirement specification document or prepare a SRS. This SRS is verify by the client and after acceptance next phase is begin.

2) **System Design**: In this phase prepare a design of a system. Plan all the system hardware and software requirements like what type of programming languages are used like Java, PHP, .net or database like Oracle, MySQL, etc. It helps in defining the overall system architecture.

3)**Implementation**: Once the design of a system is prepare, the system is first developed in small programs and these programs are called units, which are integrated in the next phase. In this phase we coding the software and performing unit testing.

The Waterfall Model

4)**Testing**: After performing the unit testing all the units are integrated into a system and performing system testing. Testing is perform to find any bugs or failure in the system or to verify that system is built as per the requirements of client.

5)**Deployment**: Once the testing phase is done, the product or software is deployed or install in the customer environment.

6)**Maintenance:** Maintenance is require to fix the issues and to released some better versions to enhance the product.

The Waterfall Model

Advantages:

- Due to its simplicity this model is easily understandable and use by any non-technical person.
- Waterfall model is useful for smaller projects and it gives an appropriate result.
- System Requirements are well documented and understand by all the projects team members
- Each phase work independently and do not overlap the other phases.
- Customer interaction is only at the beginning of the project and at the last of the project.

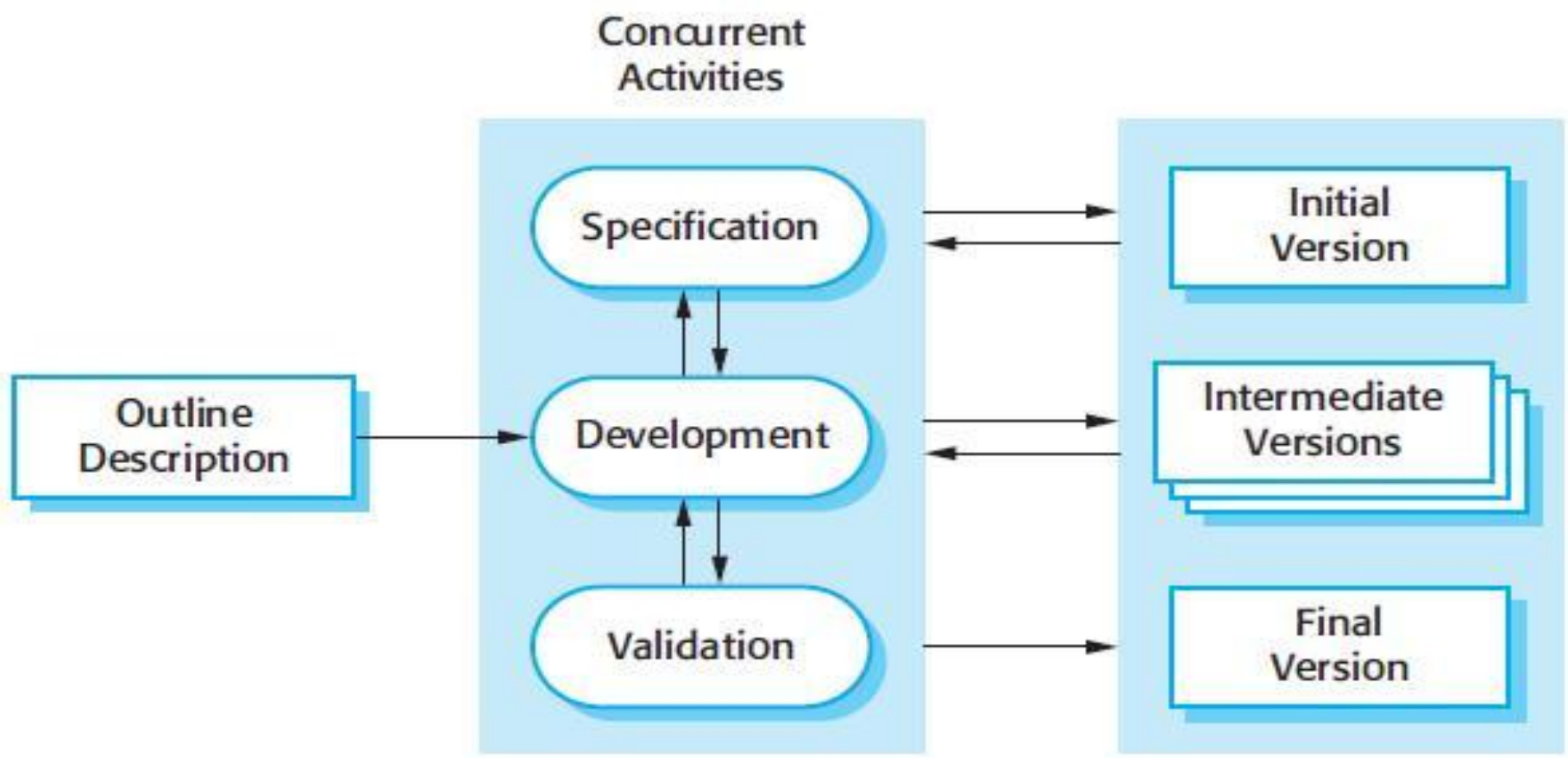
The Waterfall Model

Problems/Disadvantages:

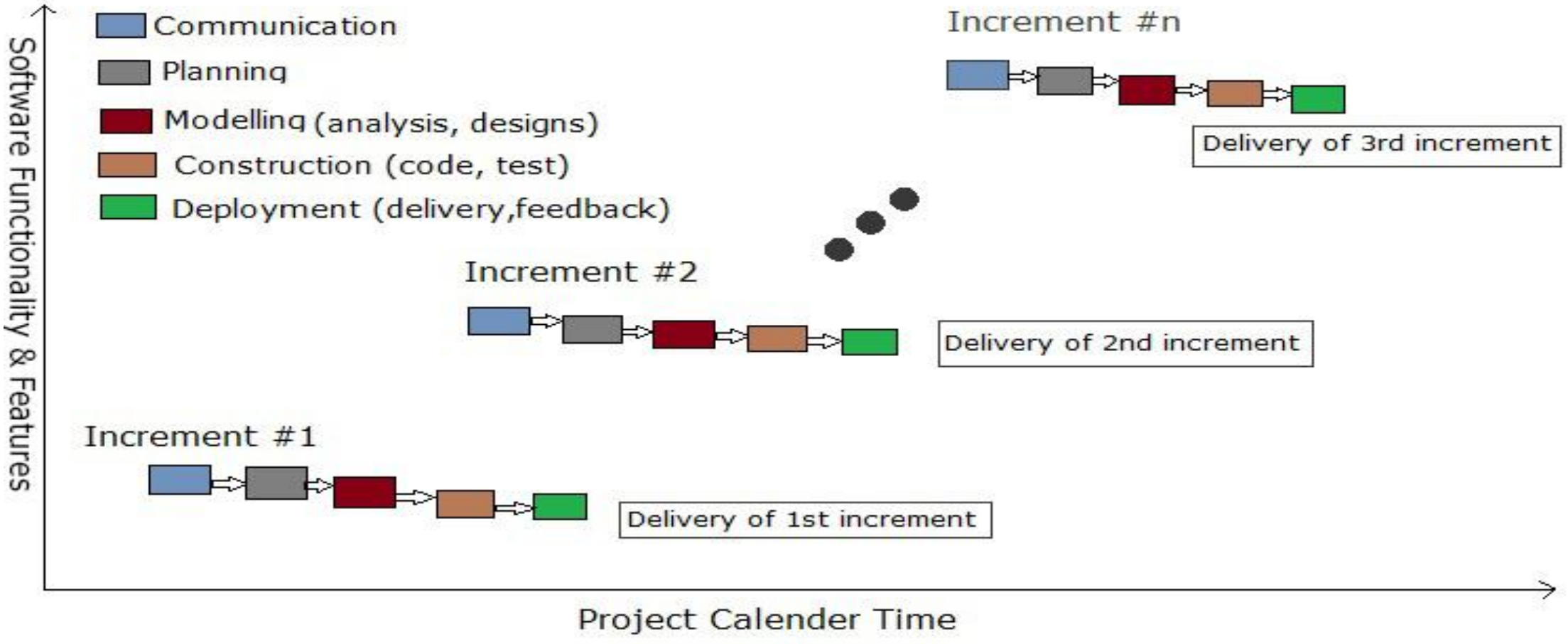
- This model is not desirable for complex and bigger project where requirement changes frequently and risk factor is higher.
- This model also not good for ongoing projects.
- Customer interaction is less and it cannot adopt the changes in system requirements.
- This model require more documentation which is not suitable for big projects.
- Customer feedback only at the end of the product.

Small change makes lot of problems in the development

Incremental development



The Incremental Model



Incremental model

- The incremental model is a method of software development where the product is designed, implemented and tested **incrementally** (a little more is added each time) until the product is finished.
- It involves both development and maintenance.
- The product is defined as finished when it satisfies all of its requirements.
- The incremental model applies the waterfall model incrementally.
- The series of releases is referred to as "**increments**", with each increment providing more functionality to the customers.
- After **the first increment**, a **core product** is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly.
- This process continues, with increments being delivered until the complete product is delivered. The incremental philosophy is also used in the **agile process model**.

Incremental model: Characteristics

Characteristics of Incremental Model

1. System is broken down into many **mini development projects**.
2. Partial systems are built to produce the final system.
3. First tackled **highest priority requirements.**
4. The requirement of a portion is frozen **once the incremented portion is developed**.

Tasks involved:

Communication: helps to understand the objective.

Planning: required as many people (software teams) work on the same project but different function at same time.

Modeling: involves business modeling, data modeling, and process modeling.

Construction: this involves the reuse software components and automatic code.

Deployment: integration of all the increments

Advantages & Disadvantages

Advantages of Incremental model:

i) Generates working software quickly and early during the software life cycle.

ii) This model is more flexible – less costly to change scope and requirements.

iii) It is easier to test and debug during a smaller iteration.

iv) In this model customer can respond to each built.

v) Lowers initial delivery cost.

Disadvantages of Incremental model:

i) Needs good planning and design.

ii) Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.

iii) Total cost is higher than other traditional model

RAD Model

- RAD is a **Rapid Application Development model**.
- Using the RAD model, software product is developed in a **short period** of time.
- The initial activity starts with the **communication** between customer and developer.
- **Planning** depends upon the initial requirements and then the requirements are divided into groups.
- Planning is more important to work together on different modules
- **The RAD model consist of following phases:**
  1) Business Modeling
  2) Data modeling
  3) Process modeling
  4) Application generation
  5) Testing and turnover

RAD Model

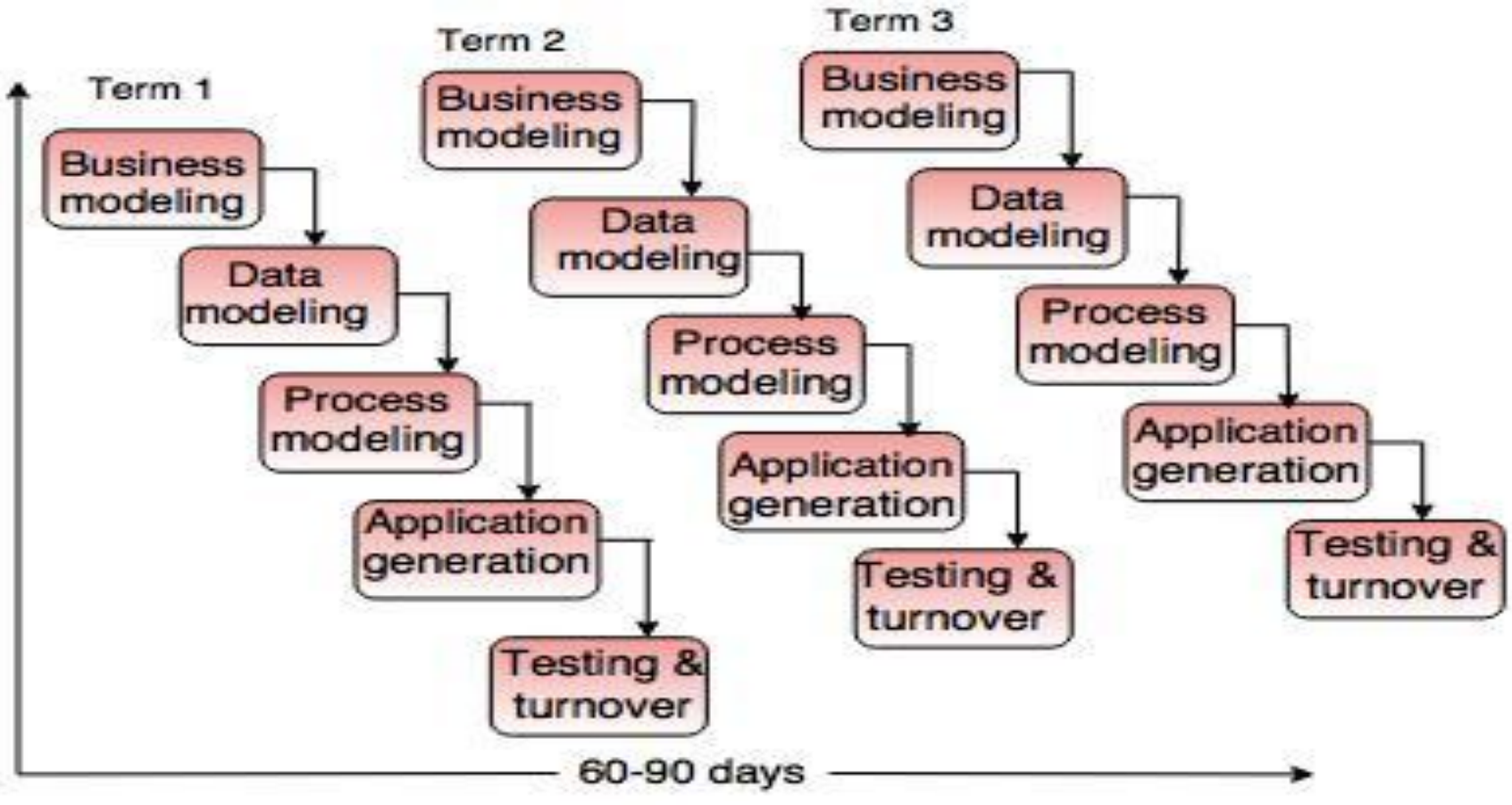


Fig. - RAD Model

RAD Model

1) Business Modeling

- consists of the flow of information between various functions in the project.
- E.g. what type of information is produced by every function and which are the functions to handle that information.
- It is necessary to perform complete business analysis to get the essential business information.

2) Data modeling

- The information in the business modeling phase is refined into the set of objects and it is essential for the business.
- The attributes of each object are identified and defined the relationship between objects.

3) Process modeling

- The data objects defined in the data modeling phase are changed to fulfil the information flow to implement the business model.
- The process description is created for adding, modifying, deleting or retrieving a data object.

A general model of the design process

4) Application generation

- In the application generation phase, the actual system is built.
- To construct the software the automated tools are used.

5) Testing and turnover

- The prototypes are independently tested after each iteration so that the overall testing time is reduced.
- The data flow and the interfaces between all the components are fully tested. Hence, most of the programming components are already tested.

Advantages of RAD Model

- The process of application development and delivery are fast.
- This model is flexible, if any changes are required.
- Reviews are taken from the clients at the staring of the development hence there are lesser chances to miss the requirements.

Disadvantages of RAD Model

- The feedback from the user is required at every development phase.
- This model is not a good choice for long term and large projects.

Evolutionary Process Models

Evolutionary Process Models

Evolutionary models are iterative type models.

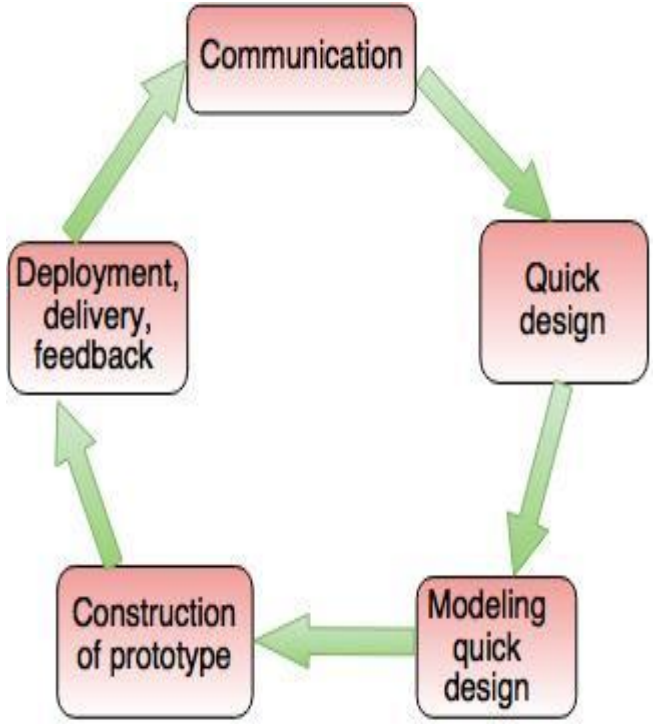
They allow to develop more complete versions of the software.

Following are the evolutionary process models.

1. The prototyping model
2. The spiral model
3. Concurrent development model

Evolutionary Models: Prototyping

- 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.



Construction of prototype

Fig. - The Prototyping Model

Prototyping Model

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.

Prototyping Model

Advantages of Prototyping Model

- 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.

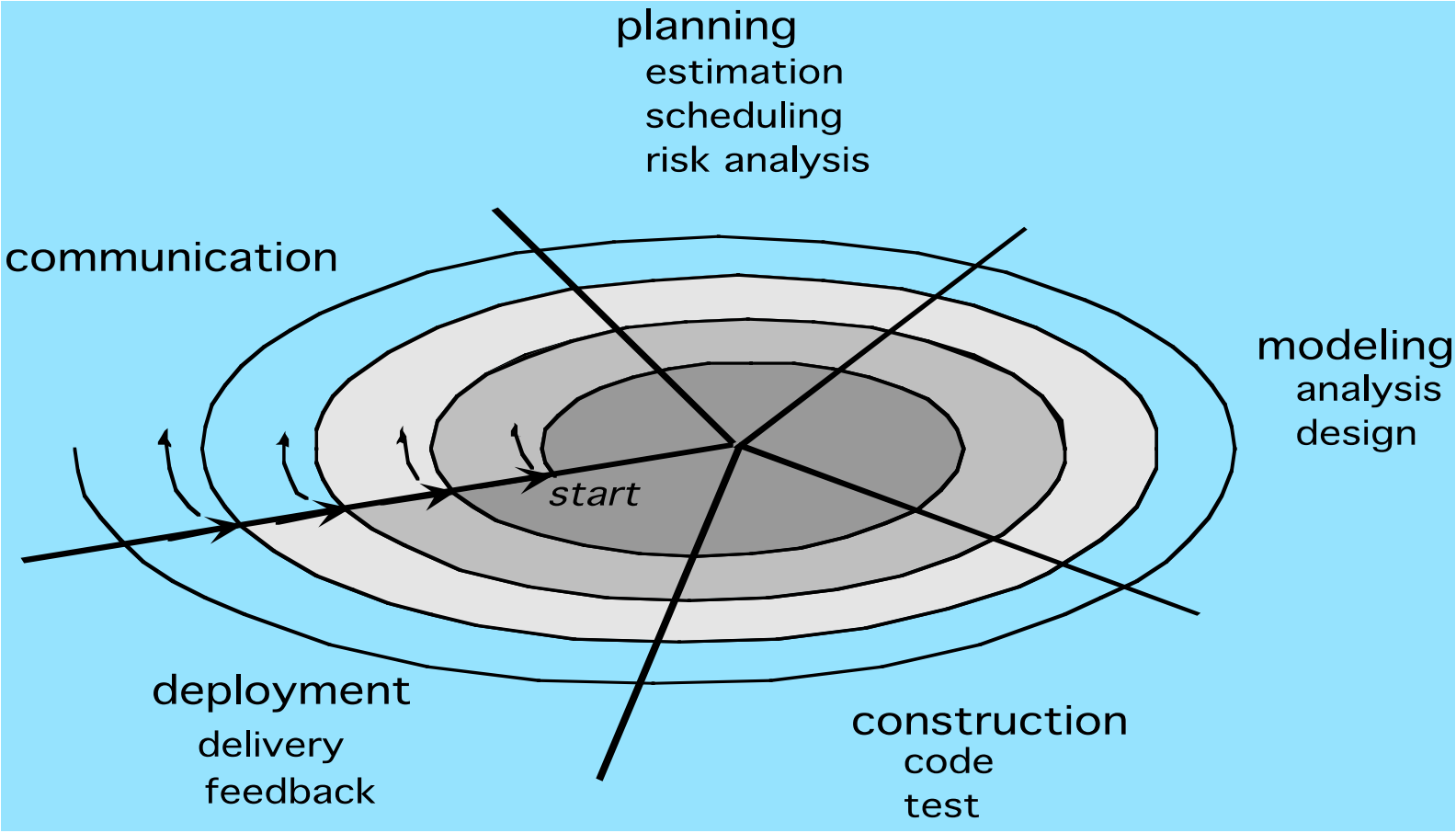
Disadvantages of Prototyping Model:

- 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.

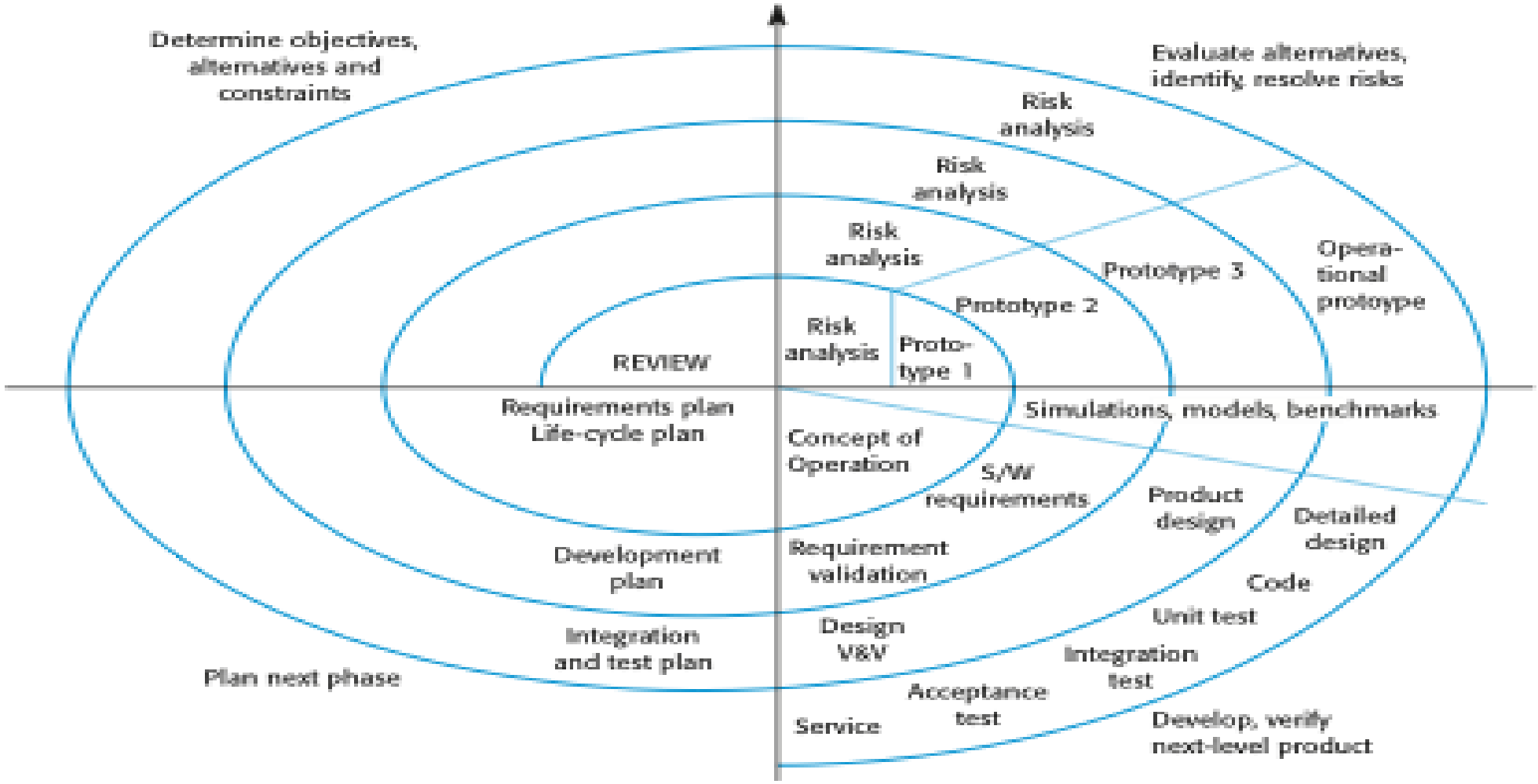
spiral model

- Spiral model is a risk driven process model.
- It is used for generating the software projects.
- In spiral model, an alternate solution is provided if the risk is found in the risk analysis, then **alternate solutions** are suggested and implemented.
- It is a combination of prototype and sequential model or waterfall model.
- In one iteration all activities are done, for large project's the output is small.

Evolutionary Models: The Spiral



Boehm's Spiral Model



Spiral model

- Phases of the spiral model is same as that of the process model.

Advantages of Spiral Model

- It reduces high amount of risk.
- It is good for large and critical projects.
- It gives strong approval and documentation control.
- In spiral model, the software is produced early in the life cycle process.
- **Disadvantages of Spiral Model**
- It can be costly to develop a software model.
- It is not used for small projects.

concurrent development 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

concurrent development model

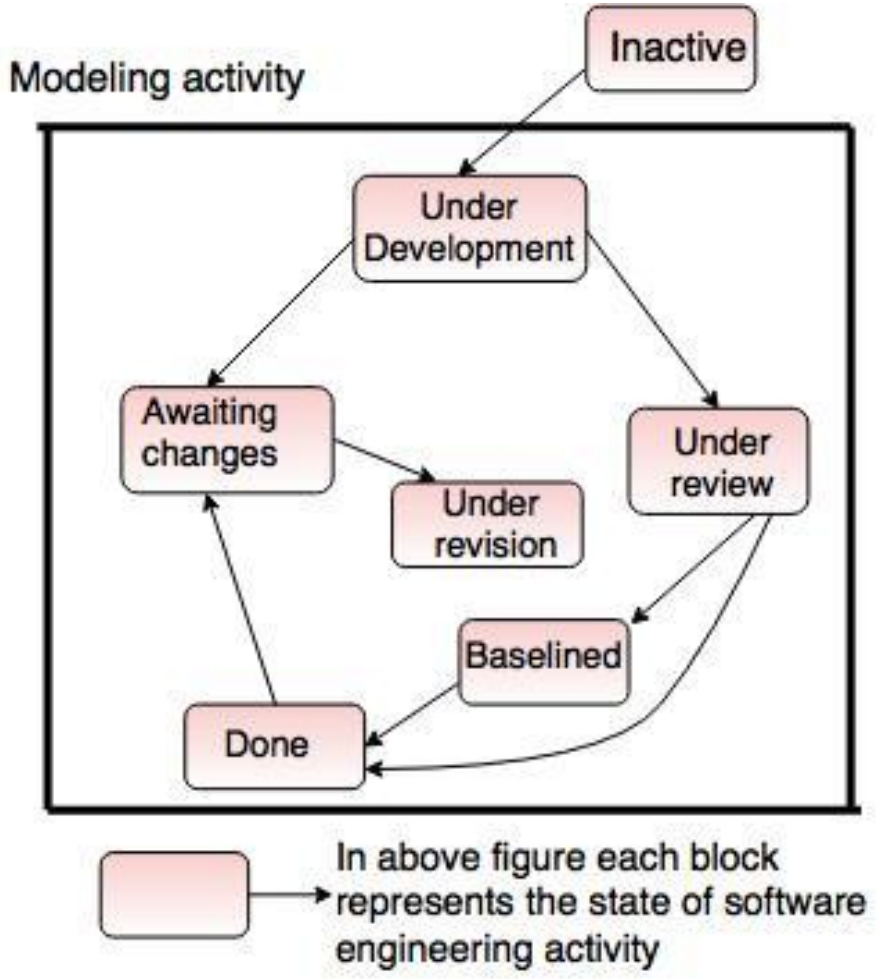


Fig. - One element of the concurrent process model

concurrent development model

- The concurrent development model is called as concurrent model.
- The concurrent process model activities moving from one state to another state.
- E. g. The modeling activity which existed in none state while initial communication was completed, now makes a transition into under development state.
- If customer indicates that changes in requirements must be made, modeling activity moves from under development state into awaiting changes state
- All activities exist concurrently but reside in different states
- Events generated at one point in the process network trigger transitions among the states

concurrent development model

Advantages of the concurrent development model

- This model is applicable to all types of software development processes.
- It is easy for understanding and use.
- It gives immediate feedback from testing.
- **It provides an accurate picture of the current state of a project.**

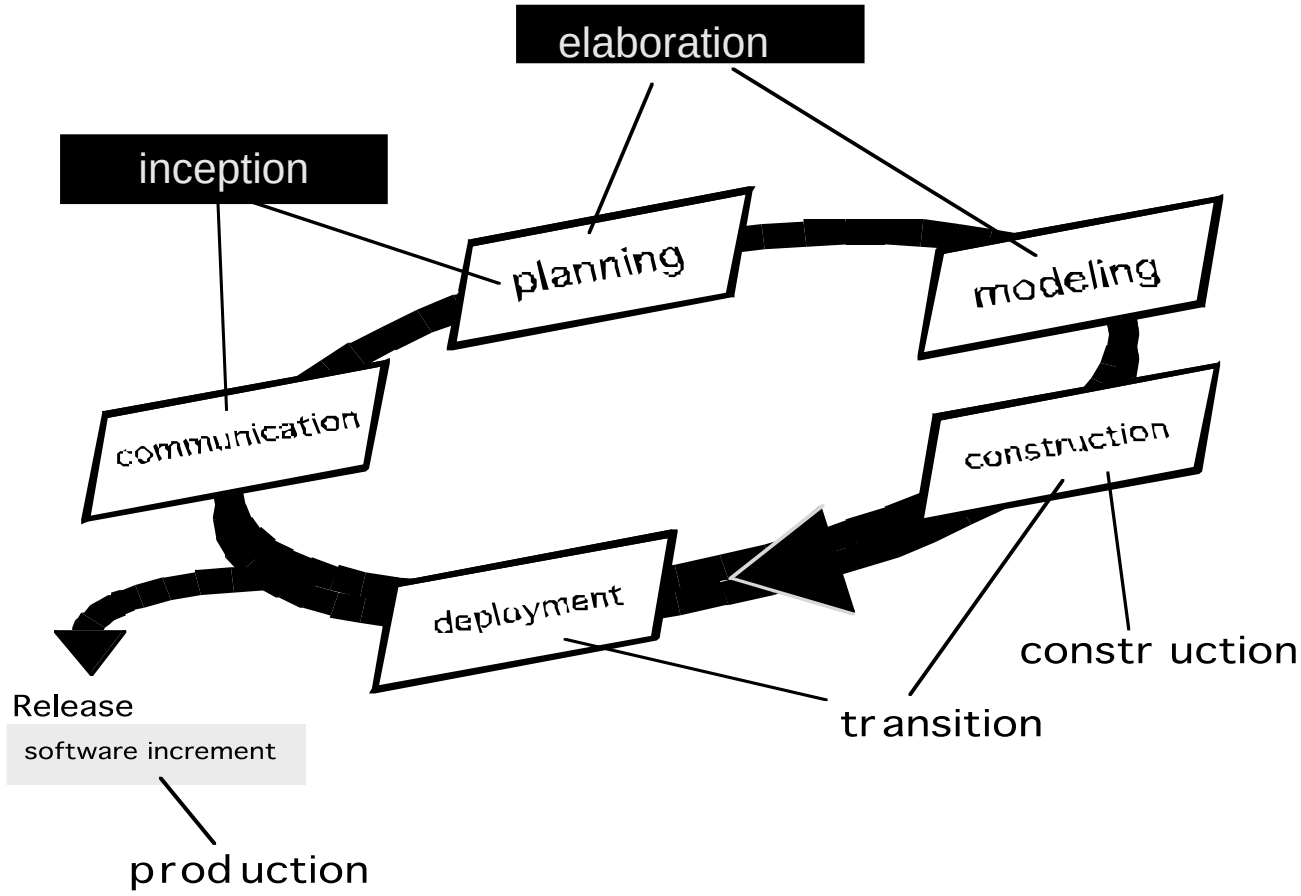
Disadvantages of the concurrent development model

- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.

The Unified Process

- Developed by I.Jacobson, G.Booch and J.Rumbaugh
- A modern generic process derived from the work on the UML and associated process.
- It is a use-case driven, architecture-centric, iterative and incremental software process
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A proactive perspective that suggests good practice.

Phases in the UP (Cont...)



Phases of UP - Inception

- Defines scope of the project
- Encompasses both customer communication and planning activities
- Fundamental business requirements are described through a set of preliminary use-cases
- A use-case describes a sequence of actions that are performed by an actor (e.g., a person, a machine, another system) as the actor interacts with the software
- A rough architecture for the system is also proposed

Phases of UP - Elaboration

- Encompasses customer communication and modeling activities
- Refines and expands the preliminary use-cases
- Expands the architectural representation to include five different views of the software
    - The use-case model
    - The analysis model
    - The design model
    - The implementation model
    - The deployment model
- In some cases, elaboration creates an “executable architectural baseline” that represents a “first cut” executable system

Phases of UP - Construction

- Makes each use-case operational for end-users
- 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

Phases of UP - Transition

- Involves many activities like delivering, training, supporting, and maintaining the product.
- Software is given to end-users for beta testing
- The software team creates the necessary support information

User manuals

Trouble-shooting guides

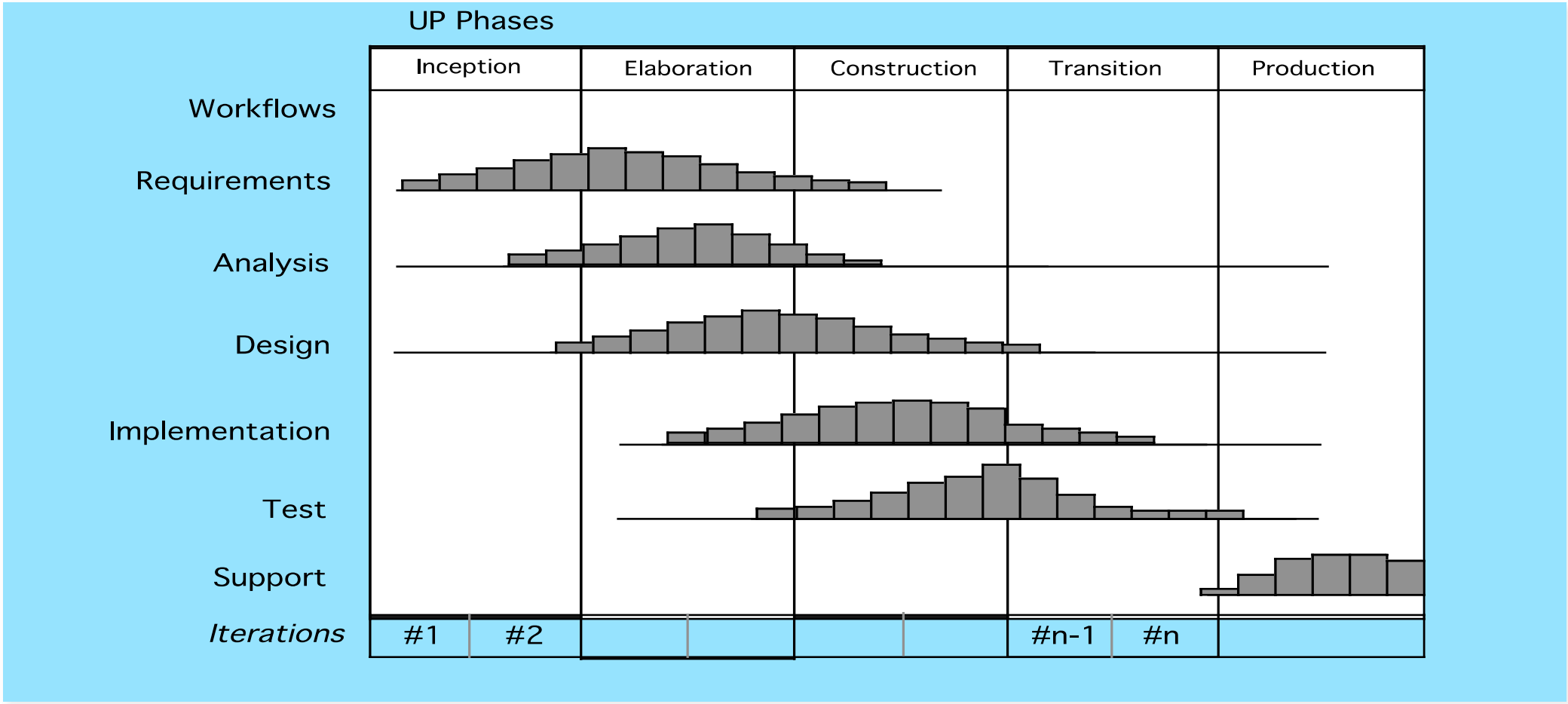
Installation procedures

- At the conclusion of the transition phase, the software increment becomes a usable software release

Phases of UP - Production

- Coincides with the deployment activity of the generic process
- The on-going use of the software is monitored
- Support for the operating environment (infrastructure) is provided
- Defect reports and requests for changes are submitted and evaluated

Phases in the UP (Cont...)



Unified process work products

Inception phase	Elaboration phase	Construction phase	Transition phase
vision document	use-case model	design model	SW increment
initial use-case model	requirements	SW components	beta test reports
initial business case	analysis model	test plan	user feedback
initial risk list	preliminary model	test procedure	...
project plan	revised risk list	test cases	
prototype(s)	preliminary manual	user manual	
...	...	installation manual	
		...	

Static workflows in the UP

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

Static workflows in the UP

Workflow	Description
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system
Project management	This supporting workflow manages the system development
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

UP good practice

- **Develop software iteratively**
  - Plan increments based on customer priorities and deliver highest priority increments first.
- **Manage requirements**
  - Explicitly document customer requirements and keep track of changes to these requirements.
- **Use component-based architectures**
  - Organize the system architecture as a set of reusable components.

UP good practice (cont…)

- **Visually model software**
  - Use graphical UML models to present static and dynamic views of the software.
- **Verify software quality**
  - Ensure that the software meet’s organizational quality standards.
- **Control changes to software**
  - Manage software changes using a change management system and configuration management tools.

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

Selection of a Life Cycle Model

- **Selection of a Life Cycle Model:**

  Selection of a model is based on:

a) Requirements

b) Development team

c) Users

d) Project type and associated risk

Based On Characteristics Of Requirements

Requirements	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Are requirements easily understandable and defined?	Yes	No	No	No	No	Yes
Do we change requirements quite often?	No	Yes	No	No	Yes	No
Can we define requirements early in the cycle?	Yes	No	Yes	Yes	No	Yes
Requirements are indicating a complex system to be built	No	Yes	Yes	Yes	Yes	No

41

Based On Status Of Development Team

Development team	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Less experience on similar projects?	No	Yes	No	No	Yes	No
Less domain knowledge (new to the technology)	Yes	No	Yes	Yes	Yes	No
Less experience on tools to be used	Yes	No	No	No	Yes	No
Availability of training if required	No	No	Yes	Yes	No	Yes

42

Based On User's Participation

Involvement of Users	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
User involvement in all phases	No	Yes	No	No	No	Yes
Limited user participation	Yes	No	Yes	Yes	Yes	No
User have no previous experience of participation in similar projects	No	Yes	Yes	Yes	Yes	No
Users are experts of problem domain	No	Yes	Yes	Yes	No	Yes

43

Based On Type Of Project With Associated Risk

Project type and risk	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Project is the enhancement of the existing system	No	No	Yes	Yes	No	Yes
Funding is stable for the project	Yes	Yes	No	No	No	Yes
High reliability requirements	No	No	Yes	Yes	Yes	No
Tight project schedule	No	Yes	Yes	Yes	Yes	Yes
Use of reusable components	No	Yes	No	No	Yes	Yes
Are resources (time, money, people etc.) scare?	No	Yes	No	No	Yes	No

44

Summary

- Process is a means to achieve project objectives of high Quality
- Software process models are abstract representations of these processes
- Process models define generic process, which can form basis of project process
- Process typically has stages, each stage focusing on an identifiable task
- Many models for development process have been proposed
- Waterfall model, Evolutionary development and component-based software Engineering, Iterative process models are some process model.
- The Unified Process is a generic process model that separates activities from phases
- A prototype can be used to give end-users a concrete impression of the system's capabilities

References

- Roger S Pressman, Software Engineering: A Practitioner's Approach, Mcgraw-Hill, ISBN: 0073375977, Seventh Edition, 2014

- Pankaj Jalote, Software Engineering: A Precise Approach, Wiley India.2010.