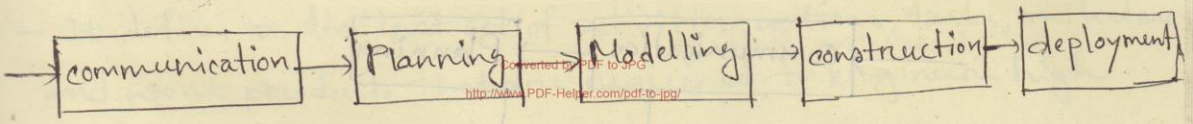
(P).Chapter(3)
Process Models

● How the frame work activity and their actions and tasks that occur within each activity are organized with respect to sequence and time?

① Linear Process flow
② iterative Process flow.
③ Evolutionary Process flow.
④ Parallel process flow.

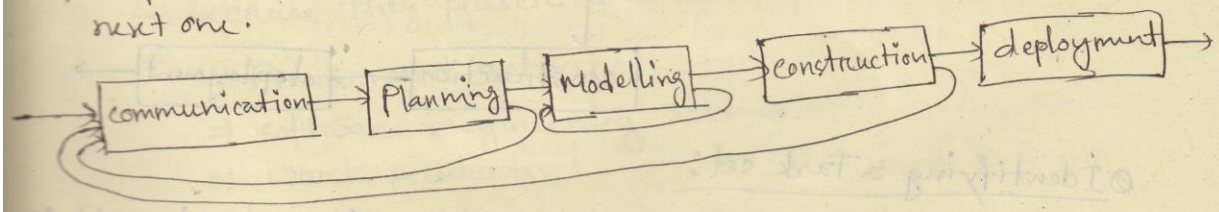
☐ Linear Process flow:

⇒ executes of frame work activities in a sequence.



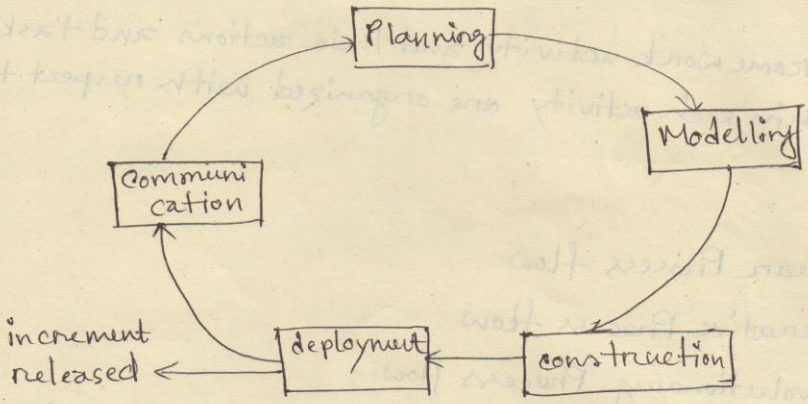
☐ Iterative Process flow:

⇒ Repeats one or more of the activities before proceeding to the next one.



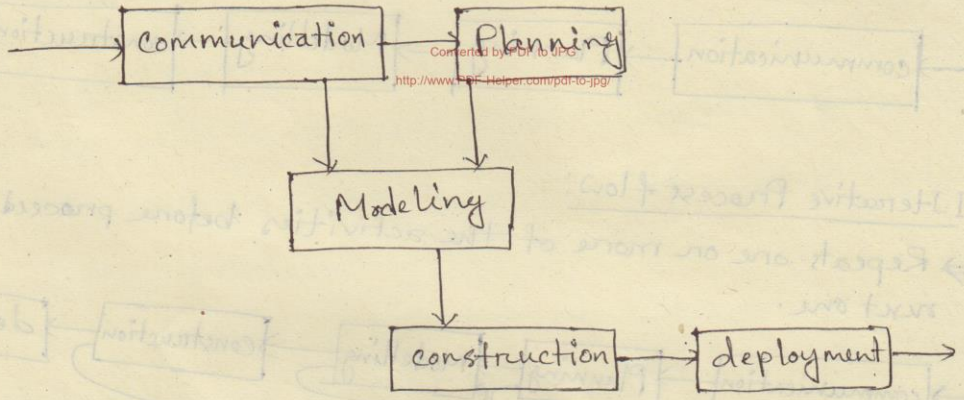
☑ Evolutionary Process Flow:

→ executes activities in circular manner.
→ Each circuit leads to a more complete version of software.



☐ Parallel Process flow:

→ Executes one or more activities in parallel with other activities.



⊘ Identifying a Task set:

☐ What actions are appropriate for a framework activity given the nature of the problem, the characteristics of the people and the stakeholders?

Task set:

→ defines the actual work to be done to accomplish the objectives of a software engineering action.

* Actions:
  - → List of task to be accomplished.
  - → List of work products to be produced.
  - → List of quality assurance filters to be applied.

Prescriptive Model:

→ Prescriptive process models advocates an orderly approach to software engineering. It

→ It defines a distinct set of activities, actions, tasks, milestone, and work products that are required to engineer high quality software.

☐ Why "prescriptive"?

→ because they prescribe a set of process elements. Like

- ⇒ Framework activity
- ⇒ software engineering actions,
- ⇒ work products.
- ⇒ quality assurance.
- ⇒ change control mechanism for each projects.

→ provides stability, control and organization to an activity that can if left uncontrolled, become quite chaotic.

☐ Workflow = manner in which process elements are interrelated to one another.

Waterfall Model / Classic Life Cycle:

→ suggests a systematic, sequential approach to software dev
that begins with customer specification of requirements and progresses through planning, modeling, construction and deployment
cumulating in on going support of complete'd software.

→ oldest paradigm of software engineering.

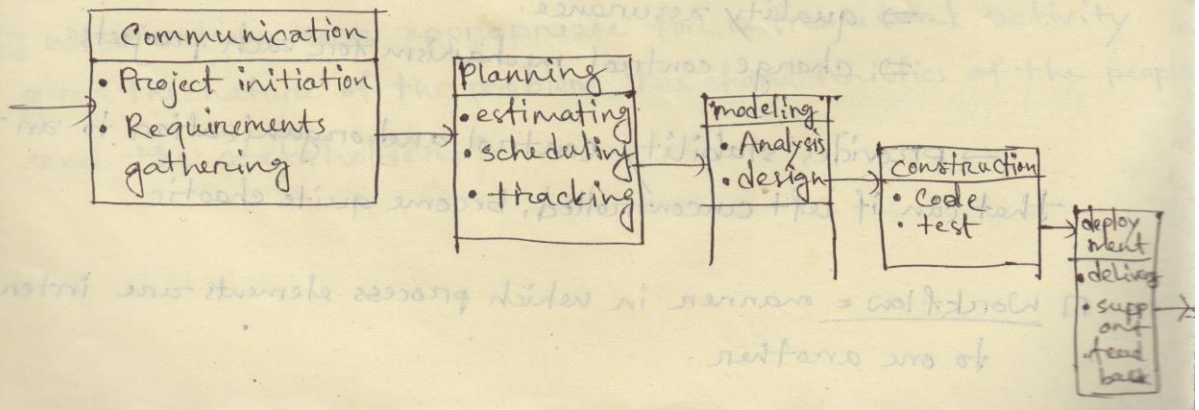
Limitations:

⇒ Real projects are rarely linear, requires iteration.
⇒ difficult for customers to state all requirements explicitly
⇒ Code will not be released until very late.
⇒ Blocking state in the model can occur.

Blocking State:

Some project members must wait for other members of the team to complete dependent tasks. Here, time spent waiting can exceed the time spent on productive work.

→ Blocking state tends more prevalent at the beginning and end of a linear sequential process.



☐ Advantages:

When requirements are well defined and resonably stable, it leads to linear fashion. It is useful in this way.

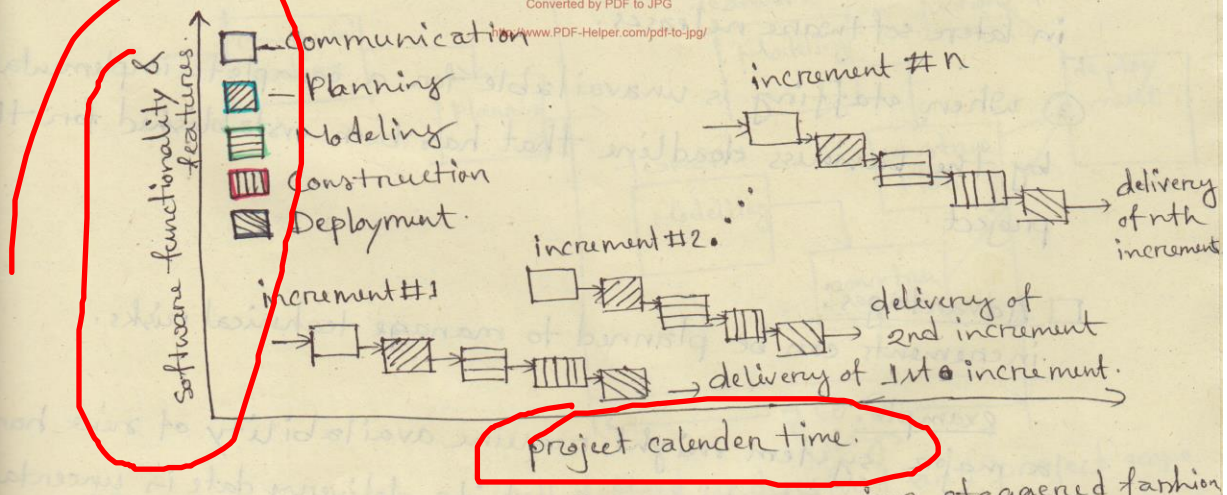
example: when well defined adaptation or enhancement to an existing system must be made.

● Why does waterfall model sometimes fail?

Limitations.

3.3. Incremental Process Models:

Incremental model combines elements of waterfall model applied in an iterative fashion.



→ Incremental model applies linear sequences in a staggered fashion as calender time progresses.

→ Each linear sequence produces deliverable "increments" of the software.

⇒ Core product = 1st increment of an incremental model.

☐ Comparison with prototyping:

→ Incremental and prototyping are both iterative in nature but unlike prototyping, incremental model focuses on delivery of an operational product with each increment.

☐ Suggested Scinerio:

① software requirements are reasonably well-defined. but scope of development effort precludes a purely linear process.

② Compelling need to provide a limited set of software functionali-ty to users quickly and then refine and expand on the functionality in later software releases.

③ When staffing is unavailable for a complete implementation by the business deadline that has been established for the project.

☐ Advantages:

increments can be planned to manage technical risks.

example:

a major system might require availability of new hardware that is under development and its delivery date is uncertain.

It might be possible to plan early increments in a way that avoids the use of this hardware, thereby enabling partial functionality to be delivered to end-users without inordinate delay.

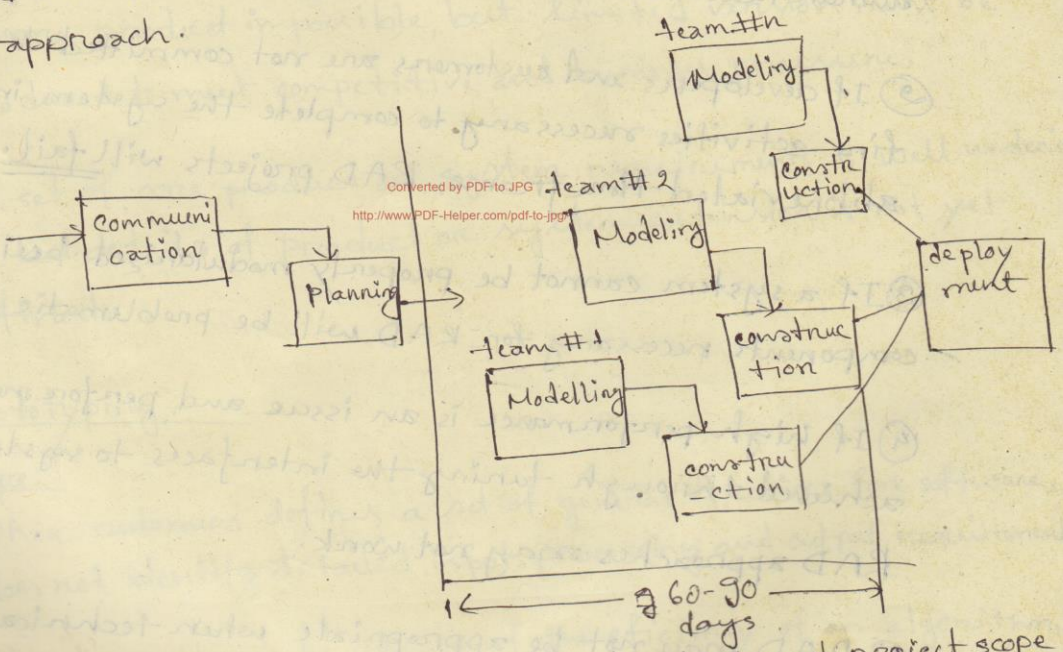
☐ RAD Model:

Ellaboration: Rapid Action Development.

Defination: RAD is an incremental software process model that emphasizes a short development cycle.

With appropriate diagram, describe that the RAD model is a "high speed" adaptation of waterfall model.

RAD model is a "high speed" adaptation of waterfall model. Rapid development is achieved by using a component based construction approach.



If requirements are well ~~defined~~ understood and project scope is constrained, RAD process enables a development team to create a "fully functional system" within a very short time period (60 to 90 days).

☐ Scope:

If a business application can be modularized in a way that enables each major functions to be completed in less than three months.

~~☐☐~~

● What are the drawbacks of this model?

☐ Limitations:

① For large but scalable projects RAD requires sufficient human resources to create the right number of RAD teams.

② If developers and customers are not committed to the rapid-fire activities necessary to complete the system in a much abbreviated time frame, RAD projects will fail.

③ If a system cannot be properly modularized, building the components necessary for RAD will be problematic.

④ If high performance is an issue and performance is to be achieved through tuning the interfaces to system components, RAD approaches may not work.

⑤ RAD may not be appropriate when technical risks are high.

3.4 Evolutionary Process Models:

- iterative
- characterized to enable software engineers to develop increasingly more complete versions of the software.

Scope:

(1) Business and product requirements often change as development proceeds. So, making a straight line path to an end product unrealistic.

(2) tight market deadlines make completion of a comprehensive software product impossible, but a limited version must be introduced to meet competitive and business pressure.

(3) a set of core product or system requirements is well understood but details of product or system extension is not yet defined.

Prototyping:

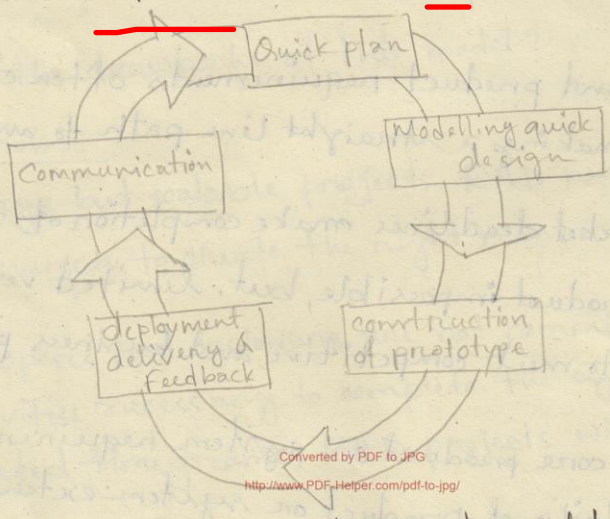
Scope:

(1) When customers defines a set of general objectives for software, but does not identify detailed input, processing and output requirements.

(2) developers may be unsure of the efficiency of an algorithm, adaptability of operating system or the form of human machine interaction should take.

Briefly explain how the prototyping model serve as a mechanism for identifying software requirements.

Prototype model begins with communication. Software engineer and customer meet and define the overall objectives for software, identifies known requirements and outlines.



This iteration is planned and designed quickly. Quick design focuses on representation of those aspects of software that is visible to customer/end-users.

From quick design prototype is generated and deployed. This is then evaluated and feedback is used to refine requirements of the software.

Iteration occurs as the prototype is tuned to satisfy the needs for of the customer, while at the same ~~ty~~ time enabling the developer better understand what needs to be done.

This is how, the prototype model serves as a mechanism for identifying software requirements.

Why prototyping is important in software engineering?

→ It can be implemented in the context of any one of the process models.

→ serves as the mechanism for identifying software requirements.

→ If a working prototype is built, developers attempts to make use of existing program fragments or applied tools that enables working program to be generated quickly.

→ both customers and developers like the prototyping paradigm as users get a feel for actual system and developers get to build something immediately.

Drawbacks:

① Customer sees what appears to be a working version of software, unaware that in the rush to get it working we haven't considered overall software quality or long-maintainability. So, too often development management relents.

② An inappropriate operating system or programming language may be used simply as it is available and known. or an inefficient algorithm may be implemented simply to demonstrate capability. But as ~~designer~~ developer gets comfortable to these choices, less-than-ideal choice has now become an integral part of system.

Spiral Model:

☐ Anchor-point milestones:

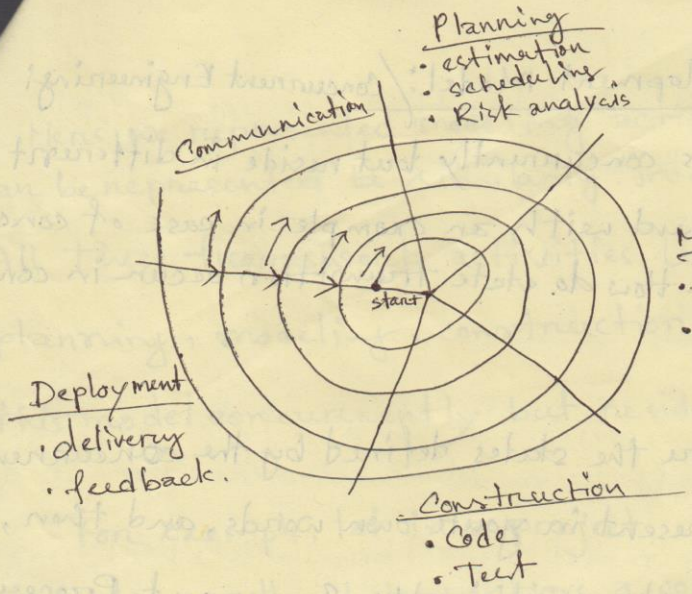
Combination of work products and conditions that are attained along the path of the spiral.

- Is it possible to combine process models? If so, provide an example.
- Explain how the spiral model couples the iterative nature of prototyping with controlled systematic aspects of waterfall model.

⇒ Spiral model uses prototyping as a risk reduction mechanism but more importantly, enables the developers to apply prototyping approach at any stage in the evolution of the product.

It maintains the systematic stepwise approach suggested by waterfall model but incorporates into an iterative framework that more realistically reflects the real world.

In this way, spiral model couples the iterative nature of prototyping with controlled systematic aspects of waterfall model.



☐ Advantages:

→ Provides potential for rapid development of increasingly more complete version of software.

→ Realistic approach to development of large scale systems and softwares.

→ demands a direct consideration of technical risks at all stages and should reduce risks before they become problematic.

☐ Limitation:

→ difficult to convince customers that the evolutionary approach is controllable.

→ demands considerable risk assessment expertise and relies on this expertise for success.

→ If fixed-budget development is demanded, spiral can be a problem as, at each circuit completion, project cost is revisited and revised.

Concurrent ~~Development~~ Development Model: / Concurrent Engineering:

- "All activities exists concurrently but reside in different stat
  justify the statement with an example in case of concurrent development model. How do state transition occur in concurrent development model.

- Describe what are the states defined by the concurrent Process Model represent in your own words, and then, indicate how they come into play within the Concurrent Process Model.

⇒ Concurrent development model can be represented schematically as a series of framework activities, software Engineering actions and tasks and their associated states.

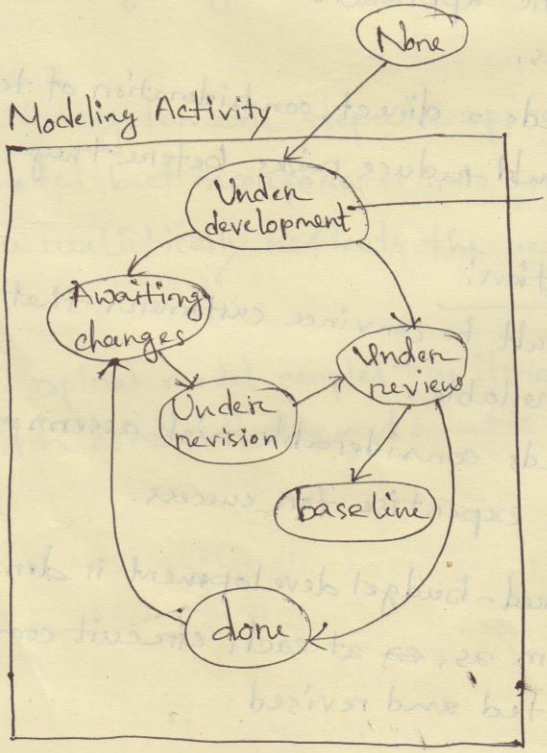
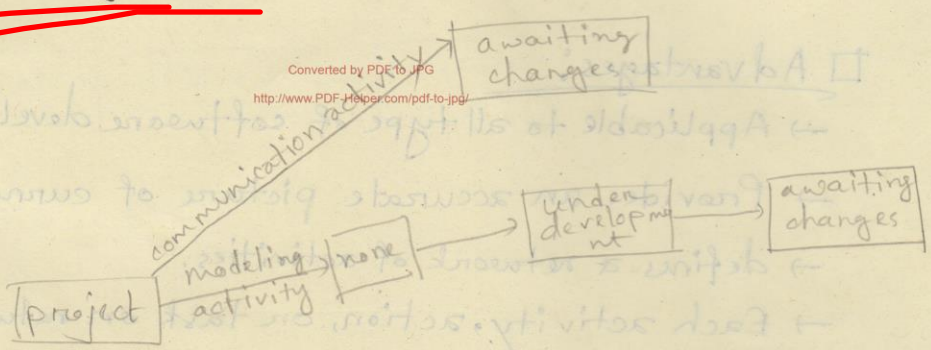


Fig: 3.6
One element of concurrent process model.

Here we represented modeling activity, but other activities can be represented similarly in an analogous manner.

⇒ All these framework activities like communication, planning, modeling, construction, deployment exists in this model concurrently but reside in different states.

For example, let in a project, at the early stage, we have a communication activity that has completed its first iteration and exists in the awaiting change state.



While communication activity is in awaiting change state, modeling activity which was in none state, enters into under development state. Modeling activity moves from under development state to awaiting changes state when customer indicates that changes in requirement must be made.

[Justified]

⇒ Concurrent process model defines a series of events that will triggen transition from state to state for each of the software engineering activities, actions or tasks.

Fig: 3.6

for example, during early stages of design, an inconsistency in in the analysis model is uncovered which generates the event - analysis model correction, which will triggen the analysis action from done state into awaiting change state

☐ Advantages:

→ Applicable to all type of software development
→ Provides an accurate picture of current state of project
→ defines a network of activities.
→ Each activity, action, or task on network exists simultaneously with other activities, actions or tasks
→ Event generated at one point in process network triggers transition among states.