Lecture – 04

Software Engineering

SECTION - A

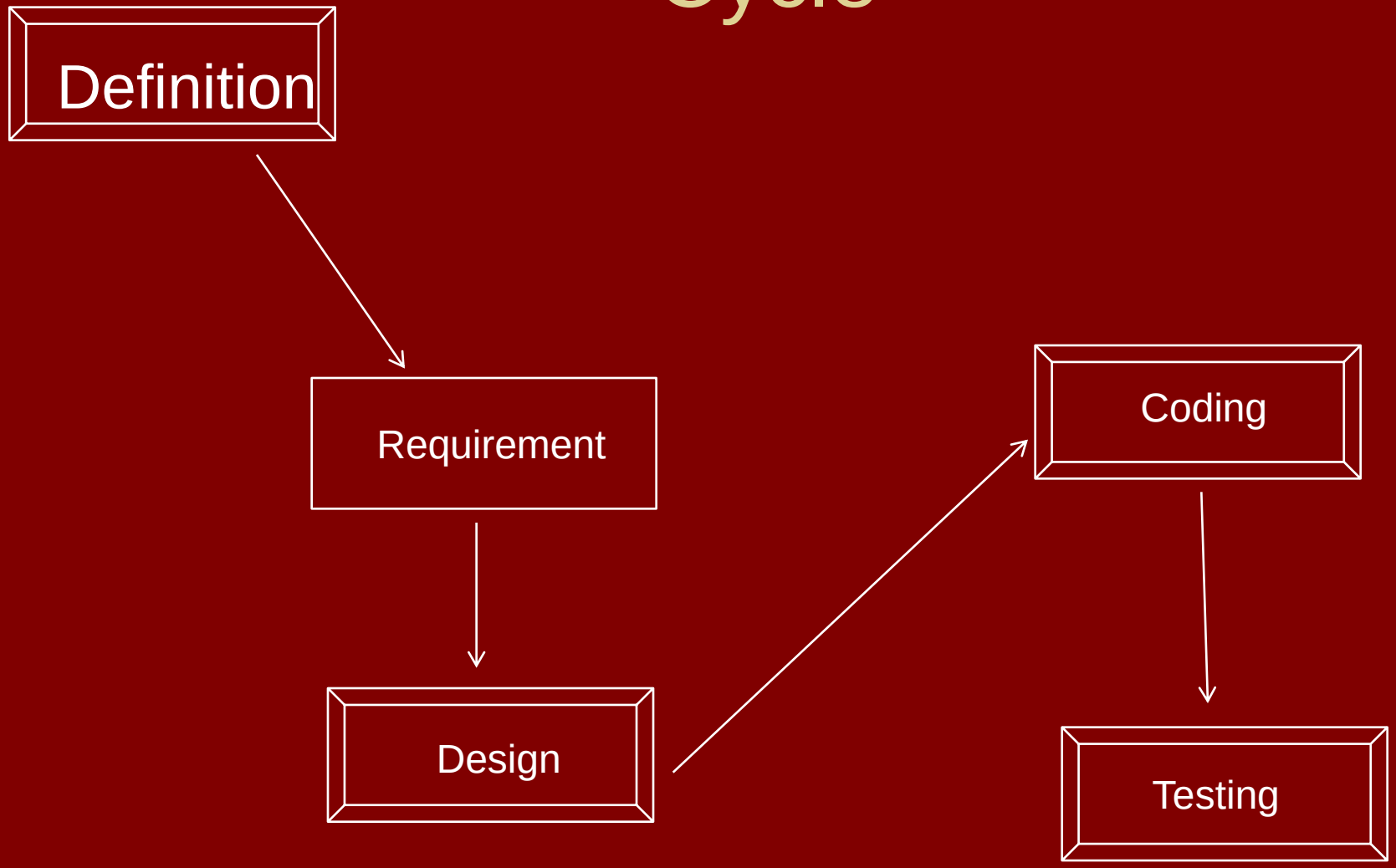
Outline

- Quick Recap of Lecture – 03.
- Waterfall model for development
- Prototyping Model
- Iterative enhancement Model
- Spiral Model

Software Life Cycle Models

- “The period of time that starts when a software product is conceived and ends when the product is no longer available for use.
- The software life cycle typically includes a requirement phase, design phase, implementation phase, test phase, operation and maintenance phase”.

The “Standard” Development Cycle



Definition (cont.)

Project Planning

- Allocate resources
- Estimate costs
- Define work tasks
- Define schedule

System analysis

Allocate system resources to

- Hardware
- Software
- Users

Development

- Software design
- User interface design
- High-level design
  - Define modular components
  - Define major data structures
- Detailed design
  - Define algorithms and procedural detail

Development (cont.)

Coding

- Develop code for each module
- Unit testing

Integration

- Combine modules
- System testing

Maintenance

- Correction - Fix software defects
- Adaptation - Accommodate changes
  - New hardware
  - New company policies
- Enhancement - Add functionality
- Prevention - make more maintainable

Waterfall Model for Development

- Here, steps (phases) are arranged in linear order.
  - A step take inputs from previous step, gives output to next step (if any)
  - Exit criteria of a step must match with entry criteria of the succeeding step.
- It follows ‘specify, design, build’ sequence that is intuitively obvious and appears natural.

Waterfall Model …

- Produces many intermediate deliverables, usually documents
  - Standard formats defined.
  - Acts as ‘baseline’ used as reference (for contractual obligations, for maintenance)
  - Important for quality assurance, project management, etc.
- It is widely used (with minor variations) when requirements are well understood.

Waterfall Model

System engineering

- software part of some larger system.
- establish requirements for all elements of the system; assign some to software.

Analysis Project planning

- understand information domain, functions, performance and interfacing. Project plans made

design

-translate requirements into s/w architecture, data structures and procedural details. A ‘detailed design’ step can be added.

code

-programming

Testing & integration

- test logic and function interface

Installation & maintenance

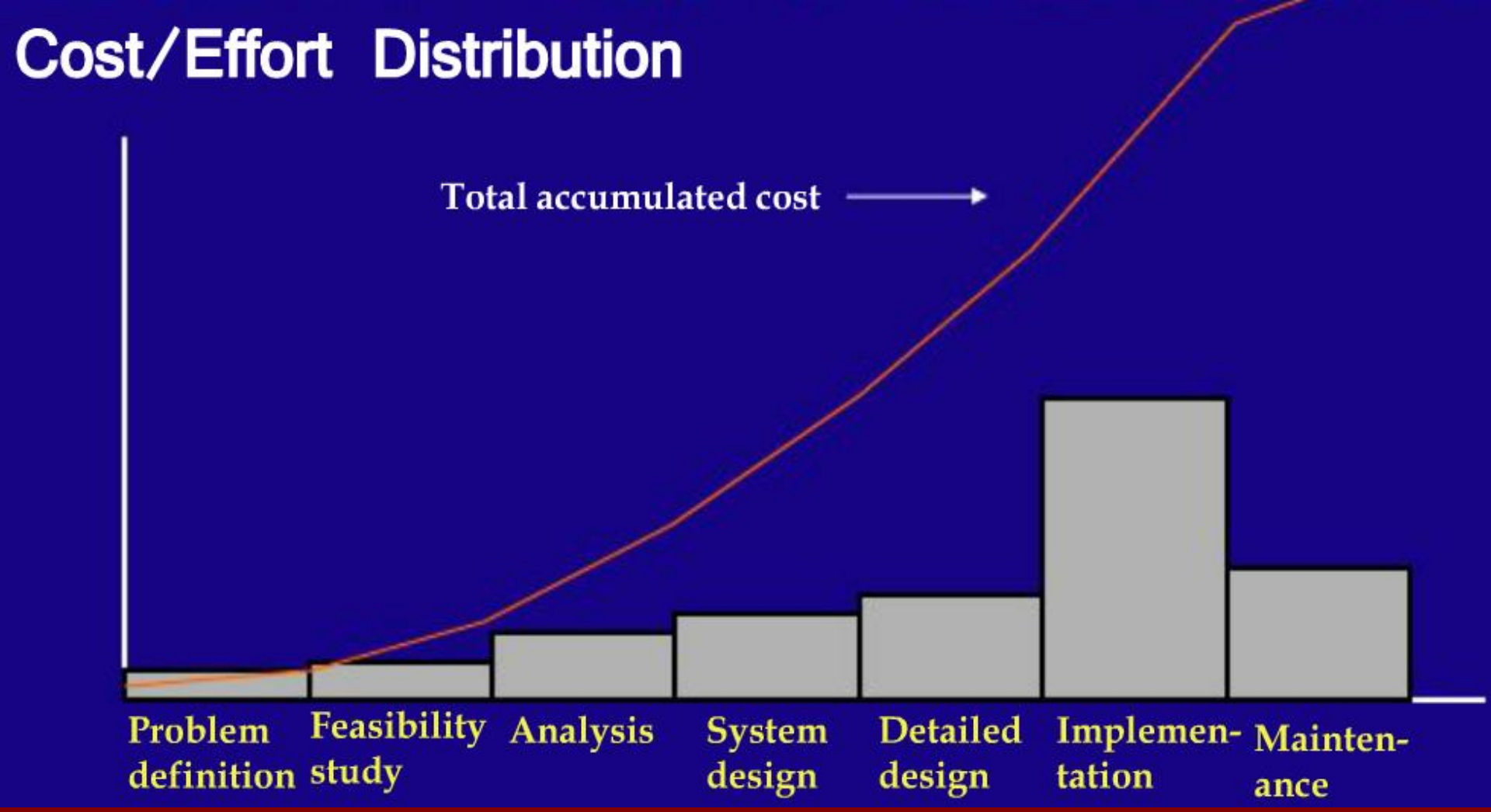
deployment; make changes for

- Errors, performance
- changes in requirement

Deliverables in Waterfall model

- Project plan and feasibility report
- Requirements document (SRS : Software Requirement Specifications)
- System design document
- Detailed design document
- Test plans and test reports
- Source code
- Software manuals (user manual, installation manual)
- Review reports

Cost/Effort Distribution



- Accumulated cost increases dramatically as programmers, operators, technical writes and computer time is committed.
- Cost of discovering and fixing errors also increases with steps.

Shortcomings of Waterfall Model

- Requirements may not be clearly known, especially for applications not having existing (manual) counterpart.
  - Railway reservation: manual system exists, so SRS can be defined.
  - On-line container management for railways – new.
  - Knowledge management for a cetral bank – new.

Shortcomings …

- Requirements change with time during project life cycle itself.
  - Users may find solution of little use.
  - Better to develop in parts in smaller increments; this is common for packages, system software
- Considered documentation-heavy: so much documentation may not be required for all types of projects.

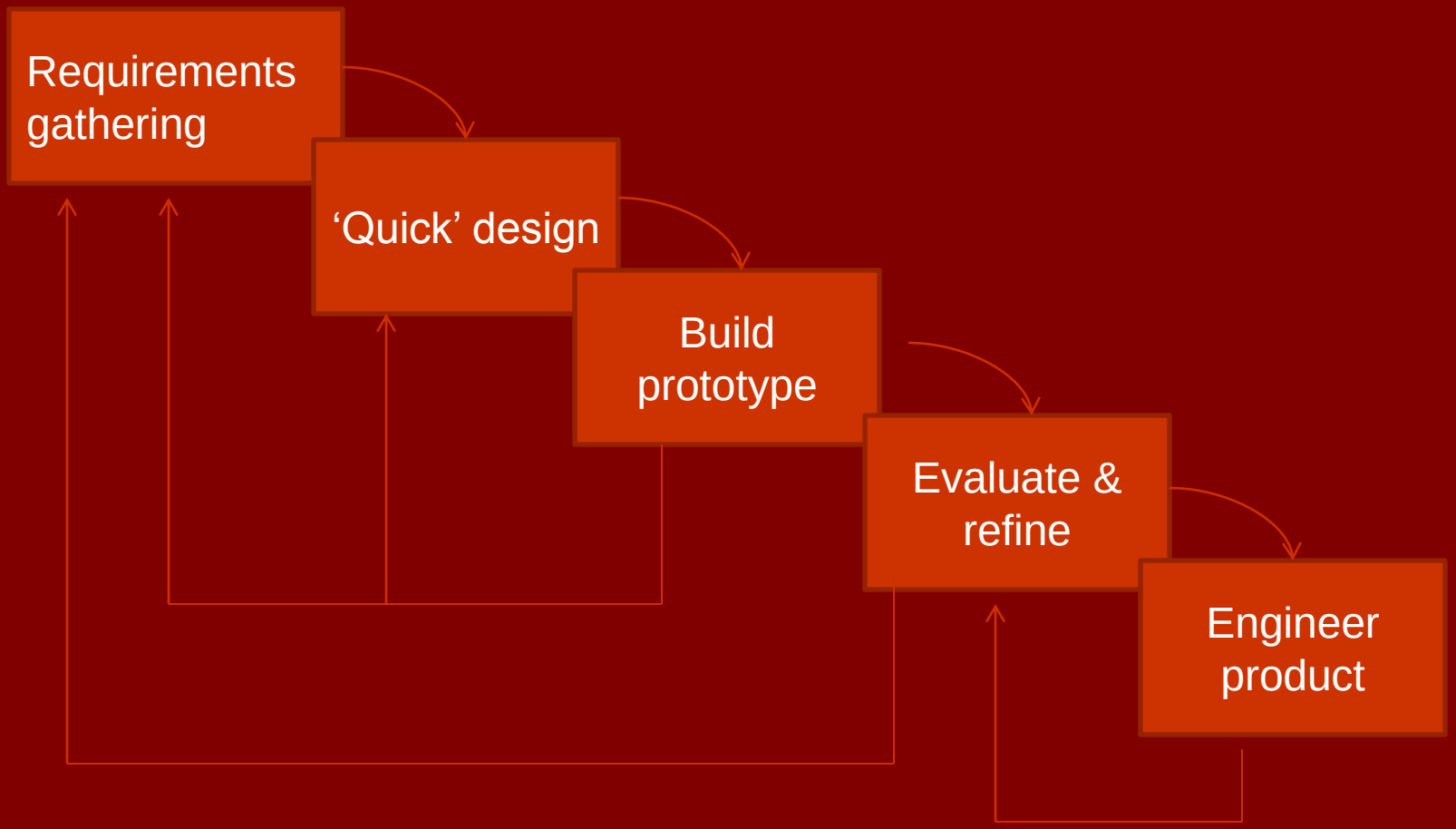
Prototyping Model

- When customer or developer is not sure
  - Of requirements (inputs, outputs)
  - Of algorithms, efficiency, human-machine interaction.
- A throw-away prototype built from currently known user needs.
- Working or even ‘paper’ prototype.

Prototyping …

- Quick design focuses on aspects visible to user; features clearly understood need not be implemented.
- Prototype is tuned to satisfy customer needs
  - Many iterations may be required to incorporate changes and new requirements.
- Final product follows usual define-design-build-test life cycle.

Prototyping



Limitations of Prototyping

- **Customer may want prototype itself !**
- **Developer may continue with implementation choices made during prototyping**
  - **may not give required quality, performance, ..**
- **Good tools need to be acquired for quick development.**
- **May increase project cost**

Iterative Development

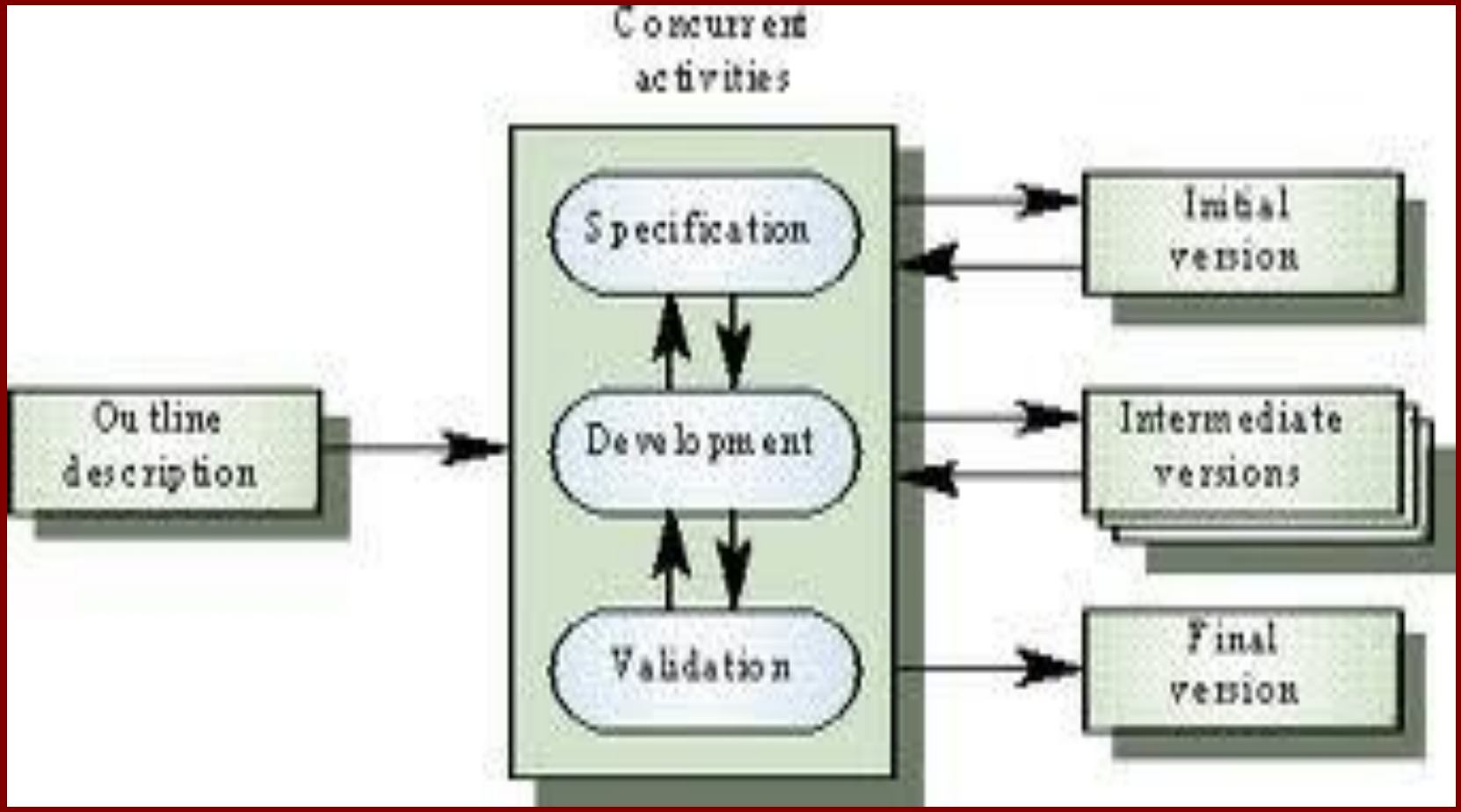
- Useful for product development where developers define scope, features to serve many customers
- Early version with limited features important to establish market and get customer feedback
- Initial version may follow any method.
- A list of features for future versions maintained.
- Each version is analyzed to decide feature list for next iteration.

Lecture – 05

Introduction

- Evolutionary Model
- Spiral Model
- Incremental Model

Evolutionary Development model



Evolutionary Development …

- **Main characteristics:**
  - **The phases of the software construction are interleaved**
  - **Feedback from the user is used throughout the entire process.**
  - **The software product is refined through many versions.**
- **Types of evolutionary development**
  - **Exploratory development**
  - **Throw-away prototyping**

Evolutionary Development …

- **Advantages**:
  - Deals constantly with changes
  - Provides quickly an initial version of the system.
  - Involves all development teams.
- **Disadvantages**:
  - Quick fixes may be involved.
  - “invisible” process, not well-supported by documentation
  - The system’s structure can be corrupted by continuous change.

Evolutionary Development …

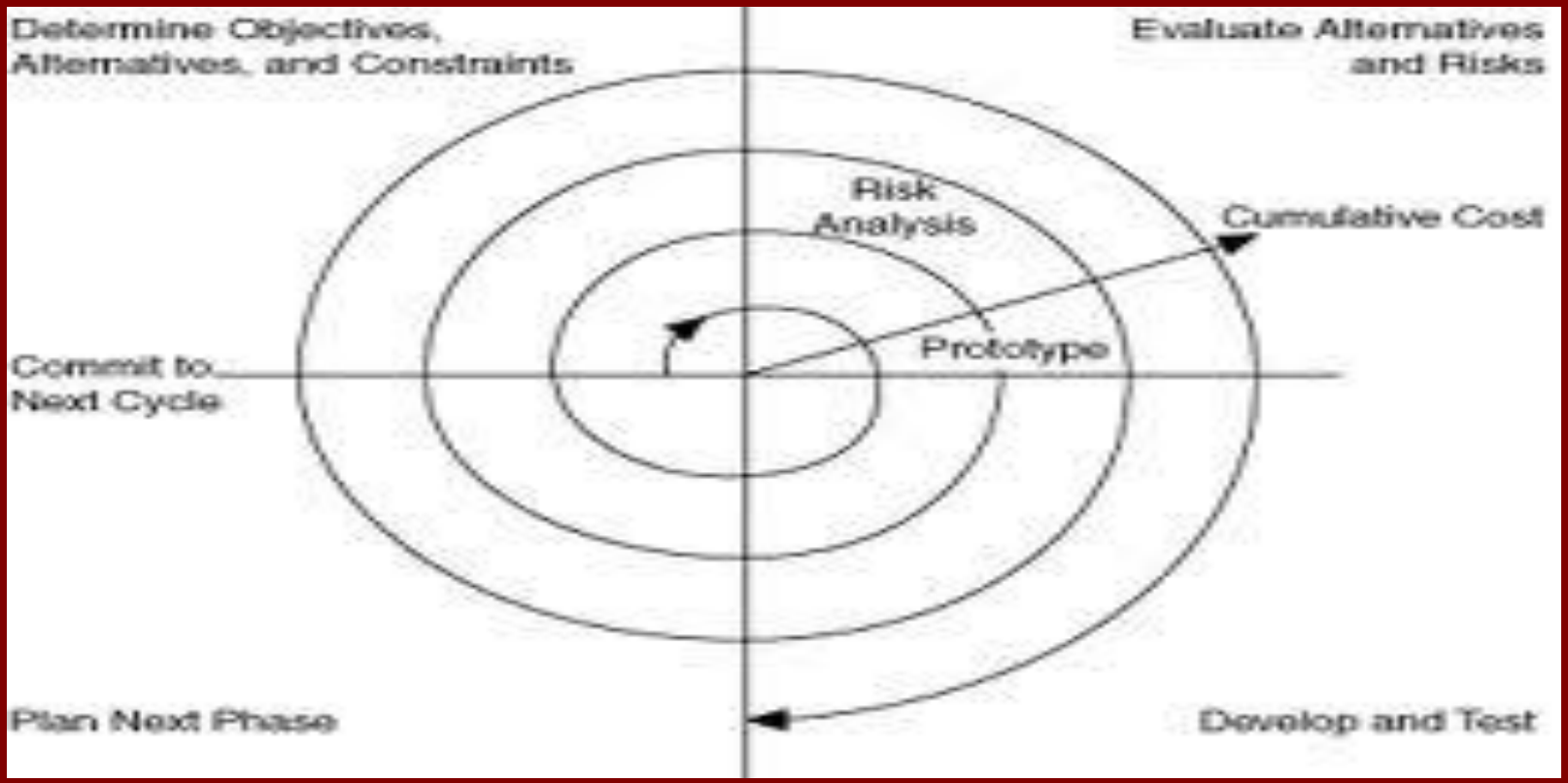
- Disadvantages [cont'd]:
  - Special tools and techniques may be necessary
  - The client may have the impression the first version is very close to the final product and thus be less patient.

- Applicability:
  - When requirements are not well understood.
  - When the client and the developer agree on a "rapid prototype" that will be thrown away
  - Good for small and medium-sized software systems.

Spiral Model

Activities are organized in a spiral having many cycles.

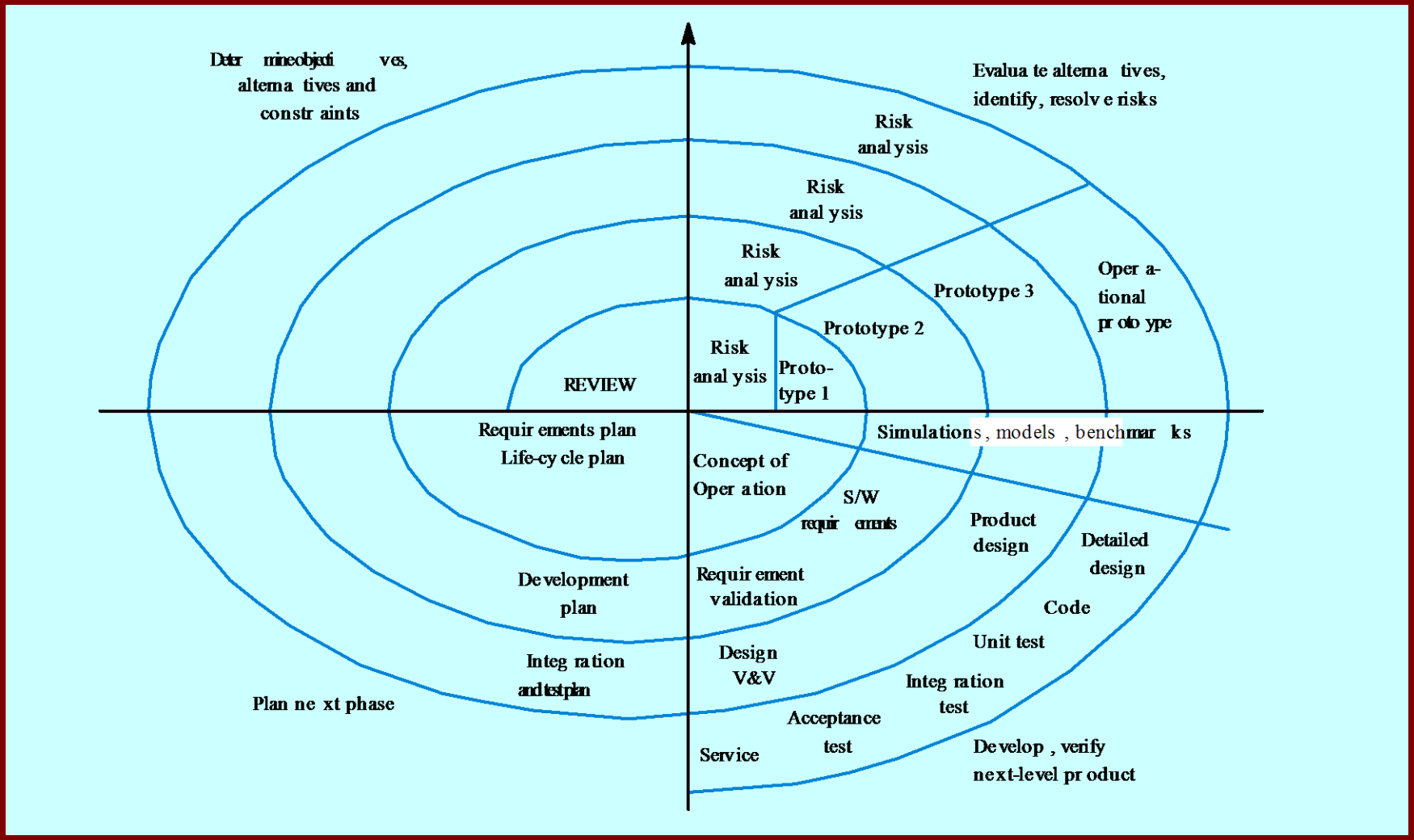
Four quadrants in each cycle.



Spiral Model …

- Prototyping, simulations, benchmarking may be done to resolve uncertainties/ risks.
- Development step depends on remaining risks; e.g.,
  - Do prototype for user interface risks.
  - Use basic waterfall model when user interface and performance issues are understood but only development risk remains.
- Risk driven : allows us mix of specification- oriented, prototype-oriented, simulation based or any other approach.

Spiral model of the software process



Spiral Model

- Main characteristics:
  - Also a hybrid model that support process iteration.
  - The process is represented as a spiral, each loop in the spiral representing a process phase
  - Four sectors per loop: objective setting, risk assessment and reduction, development and validation, planning
  - Risk is explicitly taken into consideration.

Spiral Model

- **Advantages**:
  - Risk reduction mechanisms are in place,
  - Supports iteration and reflects real-world practices.
  - Systematic approach

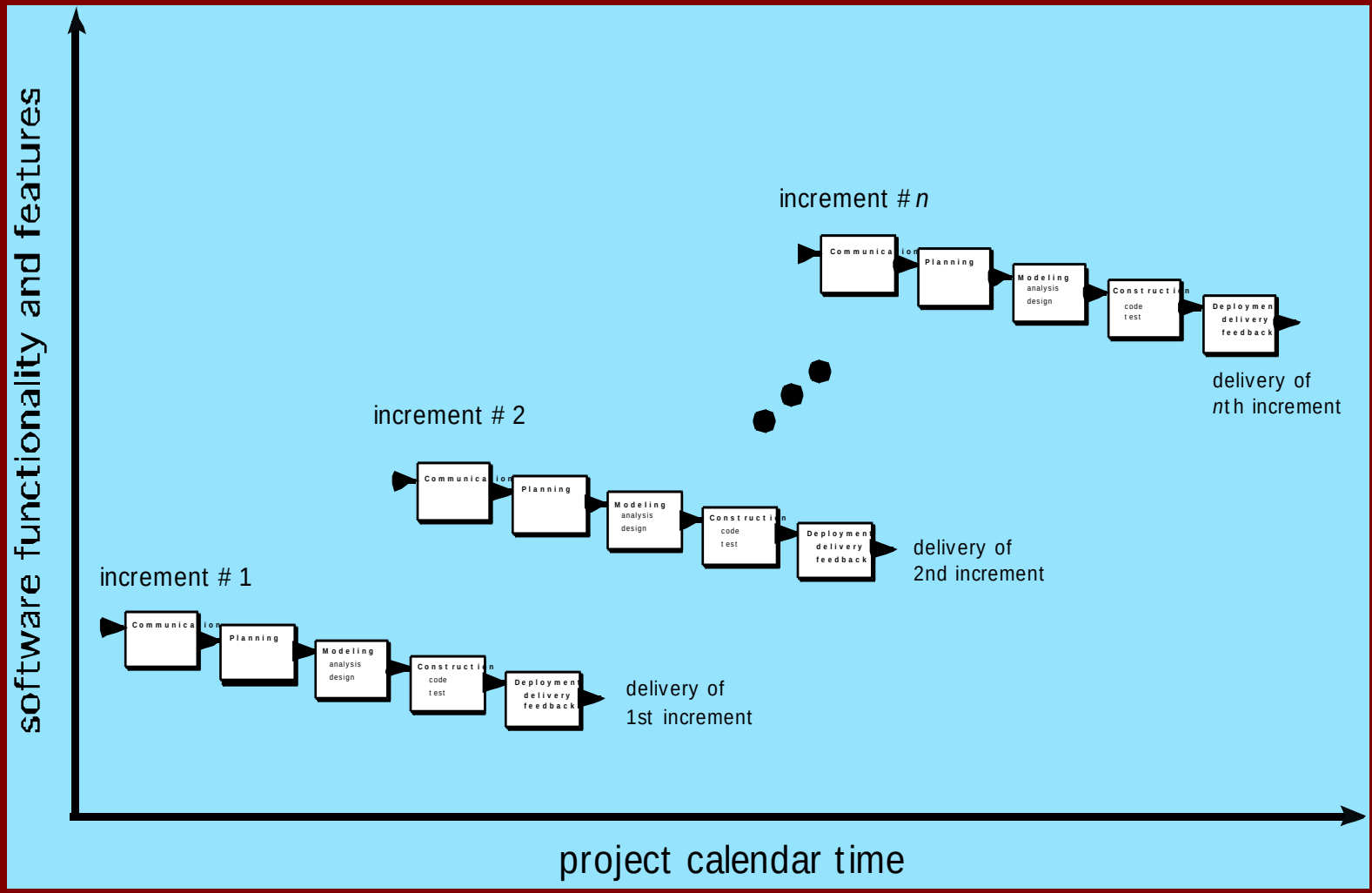
- **Disadvantages**:
  - Requires expertise in risk evaluation and reduction
  - Complex, relatively difficult to follow strictly
  - Applicable only to large systems.

- **Applicability**:
  - Internal development of large systems.

Incremental process models

- The incremental model
- The RAD model

The Incremental Model



The Incremental Model

- The incremental model applies linear sequences in a staggered(spread over a period of time) fashion as calendar time progresses.
- Each linear sequence produces deliverable “increments” of the software. For eg. Word processing software developed using the incremental paradigm might deliver basic file management editing, and document production functions in the first increment;
- More sophisticated editing, and document production capabilities in the second increment; spelling grammar checking in the third increment; and advanced page layout capability in the fourth increment.

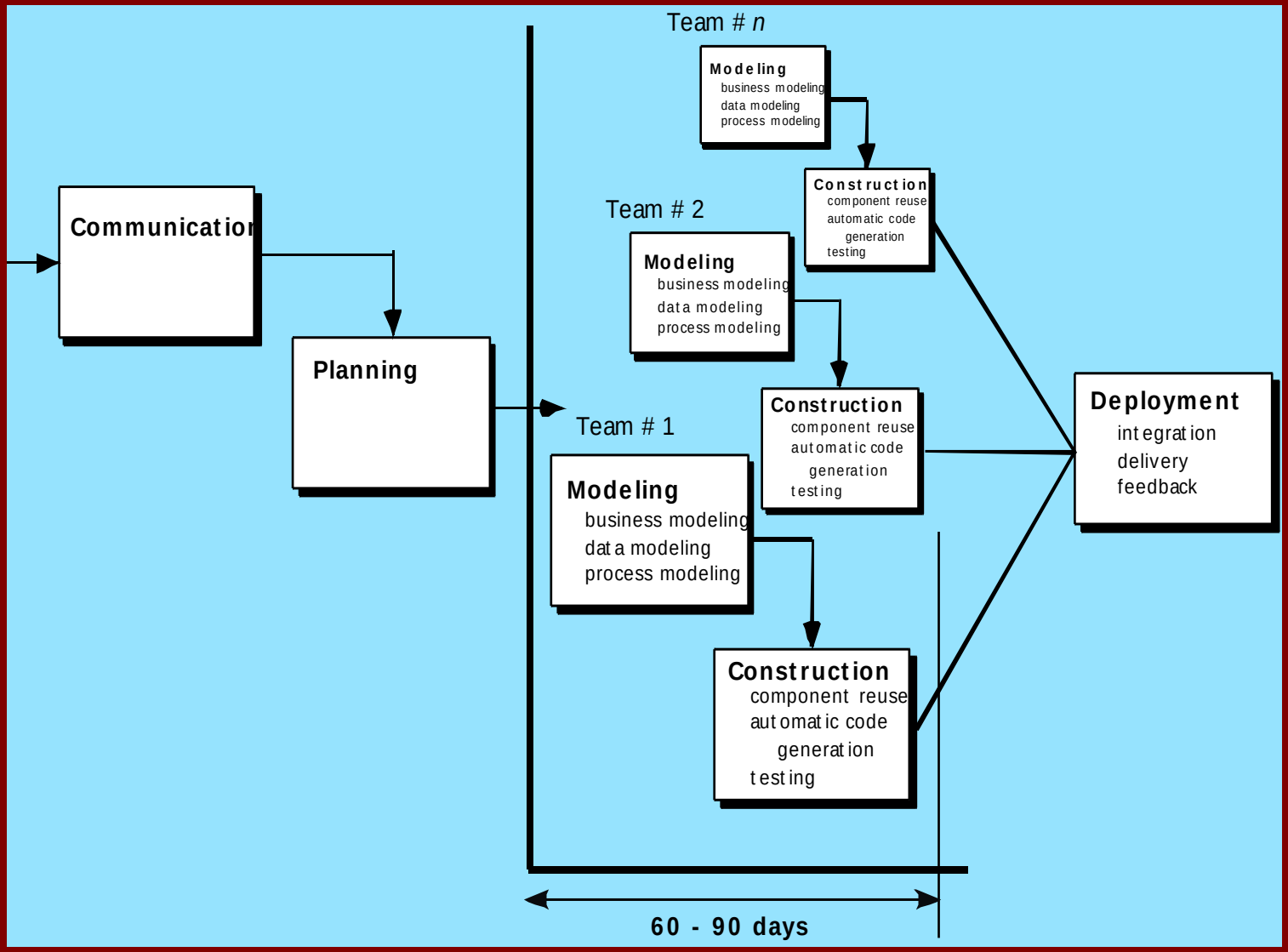
- When the incremental model is used, the first increment is often a core product; i.e. basic requirements are addressed, but many supplementary features known, others unknown) remain undelivered.
- The core product is used by the customer. As a result of use and/or evaluation, a plan is developed for the next increment.
- The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

Uses

- This model is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.
- Early increments can be implemented with fewer people. If the core product is well received, additional staff(if reqd) can be added to implement the next increment.

- In addition, increments can be planned to manage technical risks. For eg. A major system might require the availability of new hardware that is under development and whose delivery date is uncertain.
- It might be possible to plan early increments in a way that avoids the use of this hardware, there by enabling partial functionality to be delivered to end users without unreasonable delay.

The RAD Model



The RAD Model

- ***Rapid Application Model(RAD)*** is an incremental software process model that emphasizes a short development cycle.
- The RAD model is a" high-speed" adaptation of the waterfall model, in which rapid development is achieved by using a component based construction approach.
- If requirements are well understood and project scope is constrained (controlled) the RAD process enables a development team to create a "fully-functional system" with in a very short time period (eg. 60 to 90 days).

- ***Communication*** works to understand the business problem and the information characteristics that the software must accommodate.
- ***Planning*** is essential because multiple software teams work in parallel on different system functions.
- ***Modeling*** encompasses three major phases – business modeling, data modeling, process modeling – and establishes design representations that serve as the basis for RAD's construction activity.
- ***Construction*** emphasizes the use of pre-existing software components and the application of automated code generation.
- Finally, ***deployment*** establishes a basis for subsequent iterations, if required.
- The RAD process model is illustrated in fig. Obviously the time constraints imposed on a RAD project demand a "scalable scope"

- If a business application can be modularized in a way that enables each major function to be completed in less than three months (using the above described approach), it is a candidate for RAD. Each major function can be addressed by a separate RAD team and then integrated to form a whole.
- Like all process models, the RAD approach has drawbacks:

(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 rapid fire activities necessary to complete the system in a much abbreviated (reduced) time frame, RAD projects will fail.

(3) RAD may not be appropriate when technical risks are high (eg. When a new application makes heavy use of a new technology)

Assignment

Explain in detail:

- Evolutionary Model
- Spiral Model
- Incremental Model