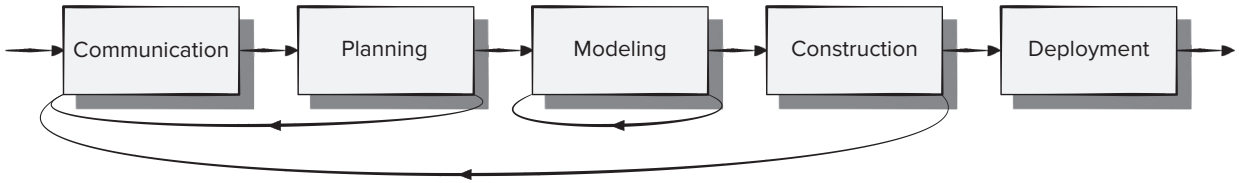
SOFTWARE ENGINEERING

Week 2_1
Prescriptive Software Process Models

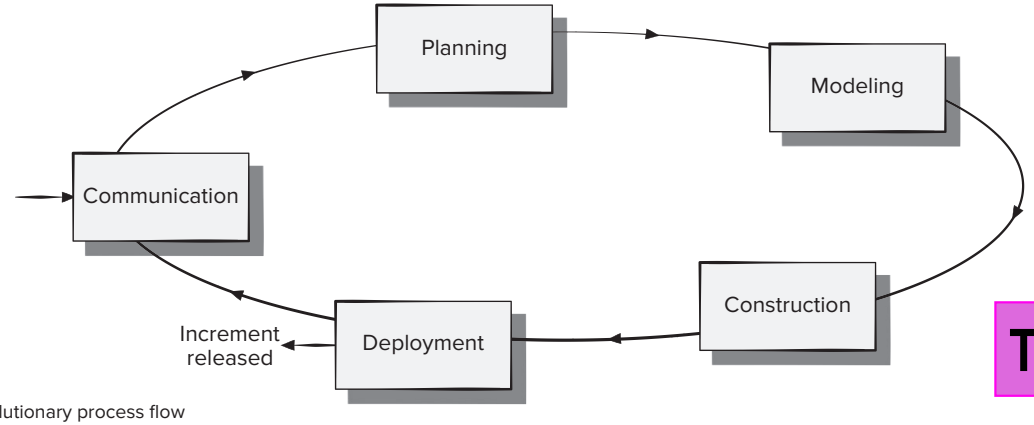
Course ID 06016410,
06016321

Nont Kanungsukkasem, B.Eng., M.Sc., Ph.D.
nont@it.kmitl.ac.th

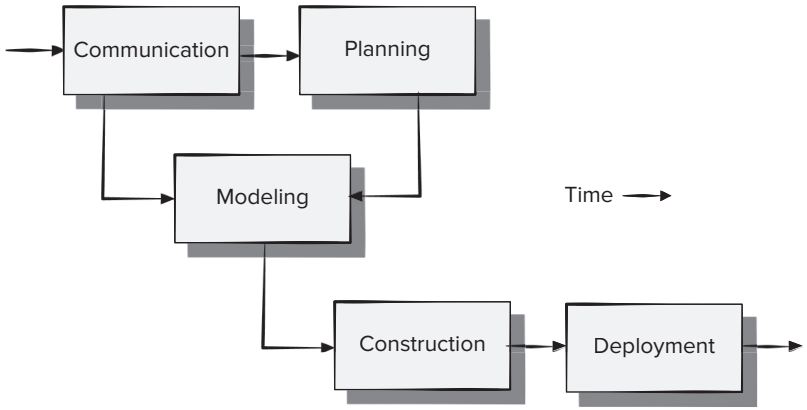
Process



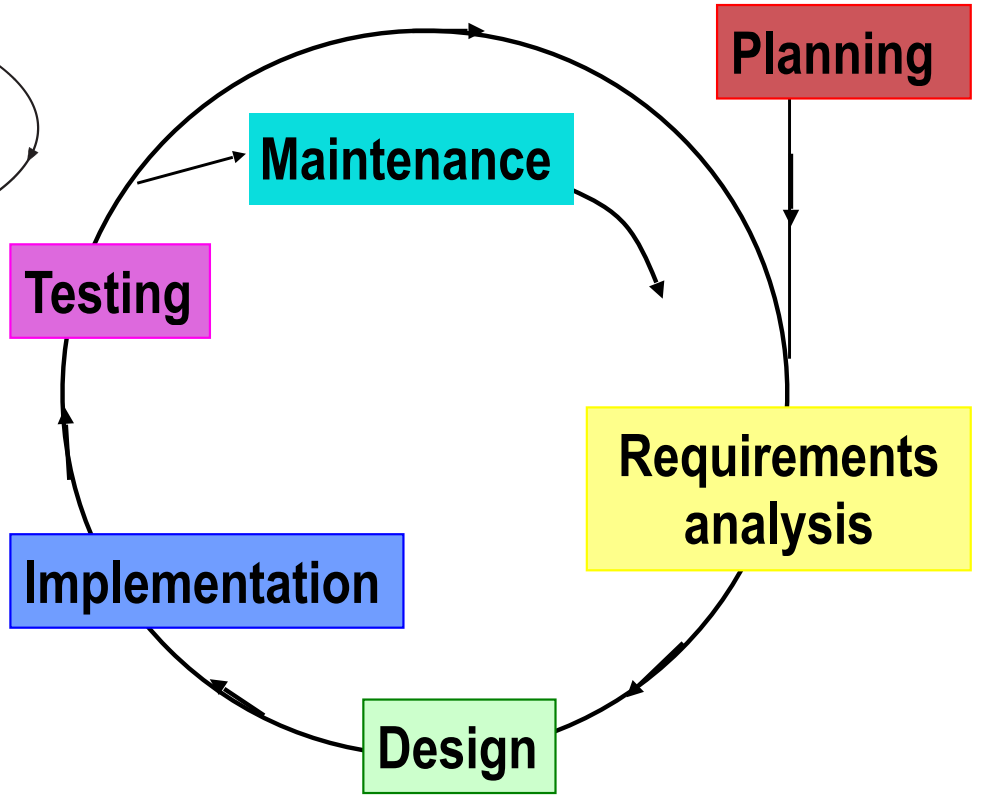
(b) Iterative process flow



(c) Evolutionary process flow

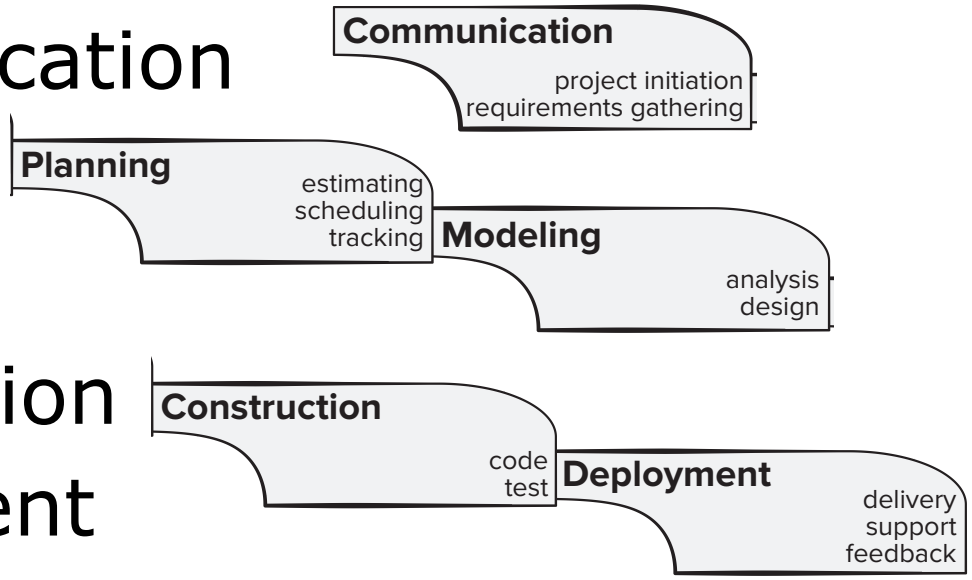


(d) Parallel process flow



Process

- Generic process framework for software engineering encompasses five activities:
  - Communication
  - Planning
  - Modeling
  - Construction
  - Deployment



Prescriptive Models

- Prescriptive process models define a predefined set of process elements and a predictable process workflow.
- Prescriptive process models strive for structure and order in software development.
- Activities and tasks occur sequentially with defined guidelines for progress.

That leads to a few questions …

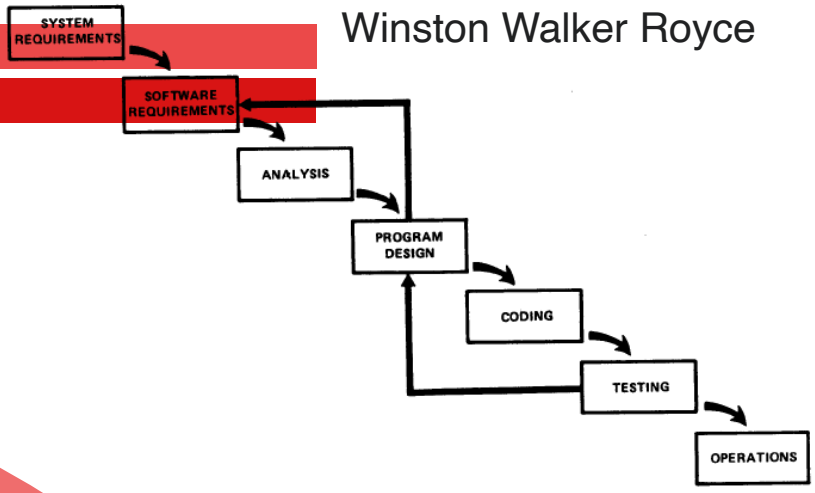
- If prescriptive process models strive for structure and order, are they appropriate for a software world that thrives on change?
- If we reject traditional process models (and the order they imply) and replace them with something *less structured*, do we make it impossible to achieve coordination and coherence in software work?

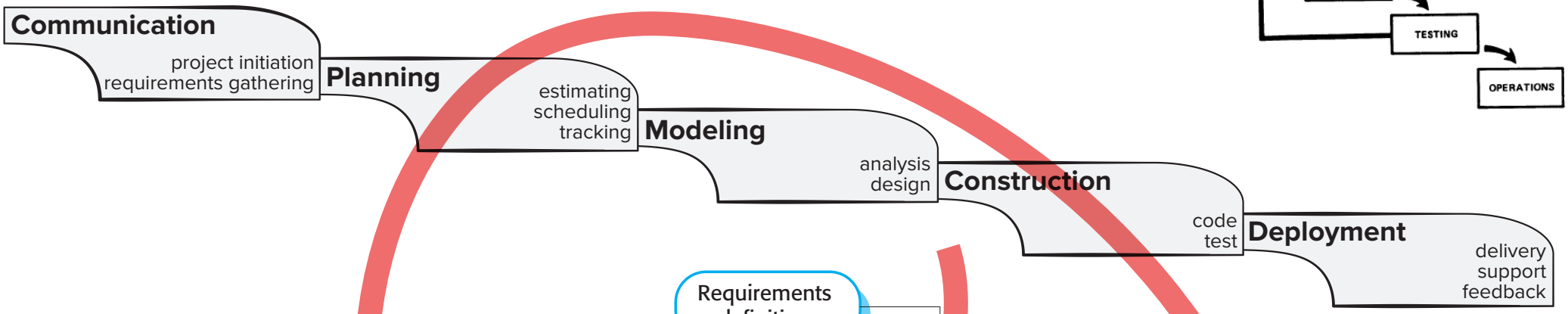
Prescriptive Models

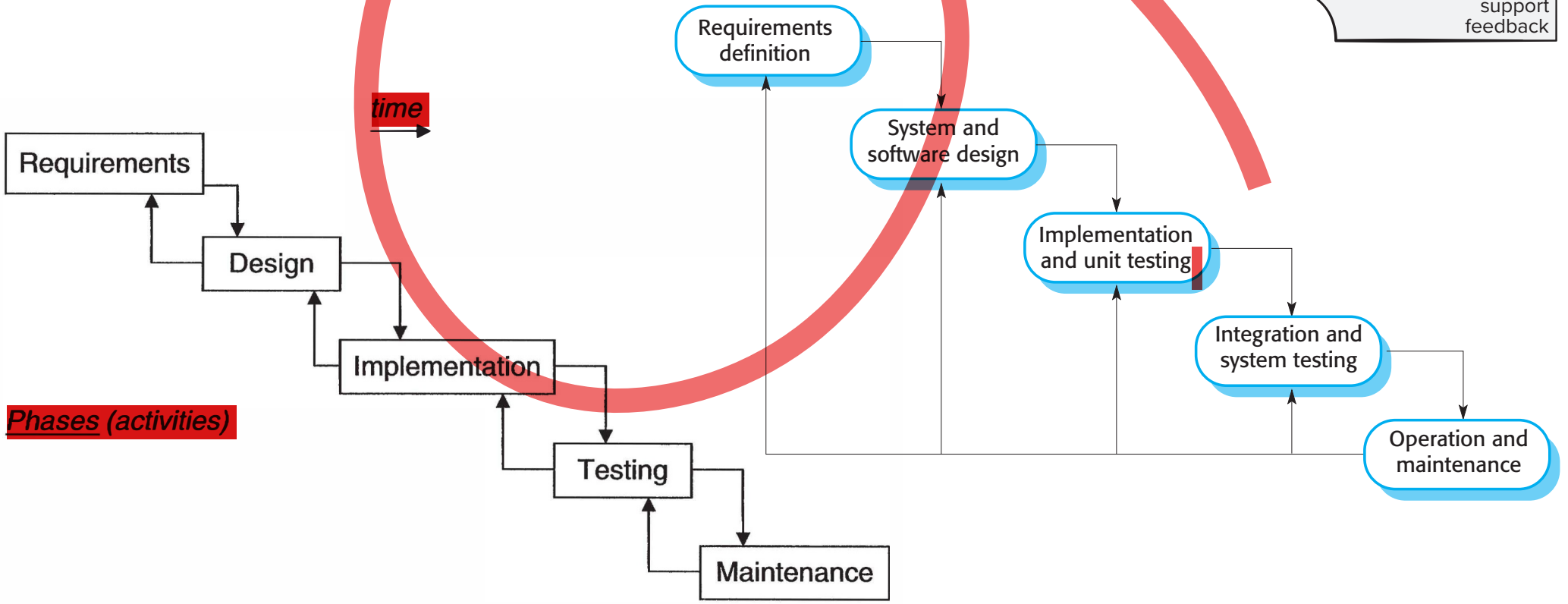
- Waterfall model and its variations
- Incremental model
- Prototyping models
- Spiral model
- Unified Process
- Other process models

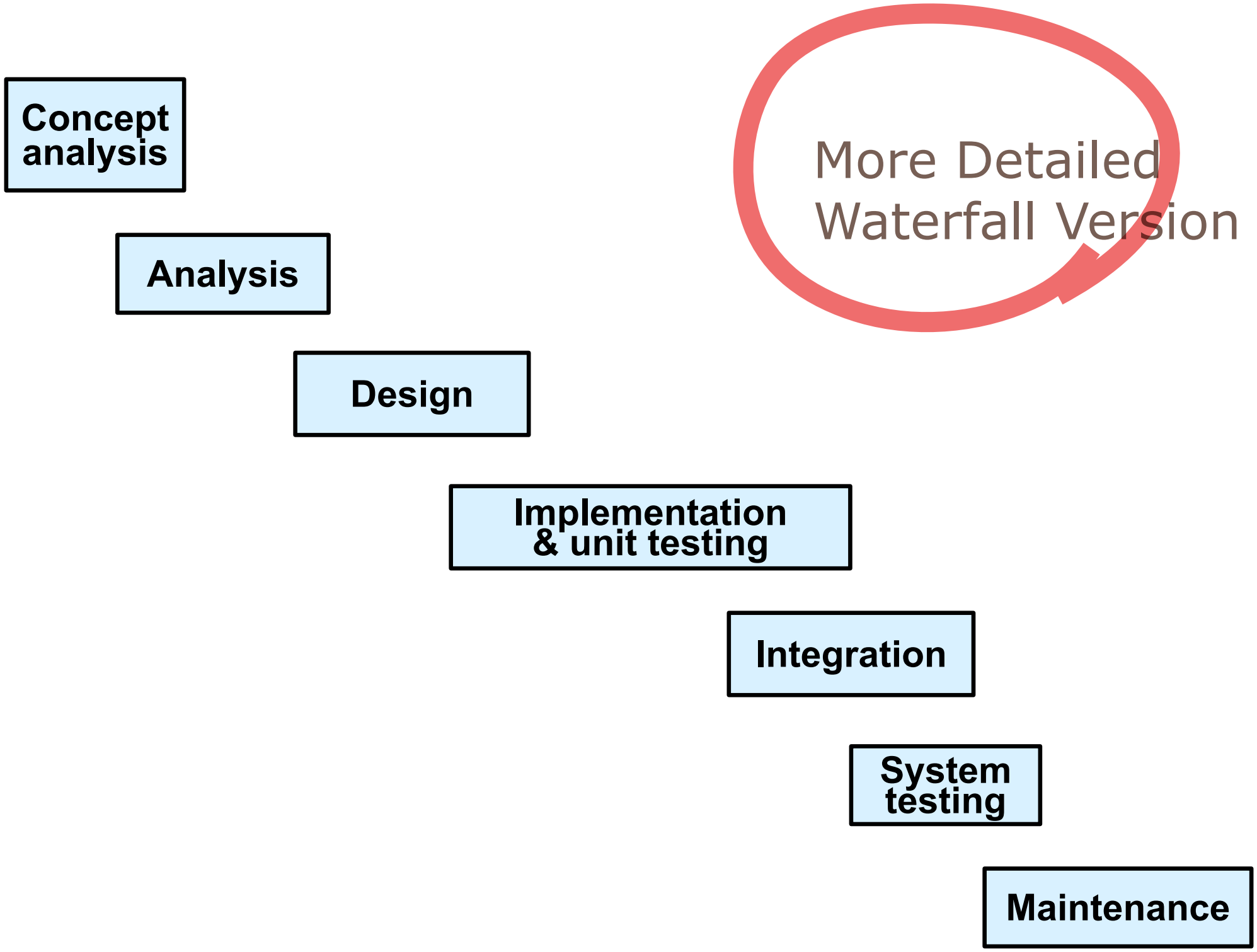
The Waterfall Model

Winston Walker Royce








Concept analysis
Analysis
Design
Implementation & unit testing
Integration
System testing
Maintenance
More Detailed Waterfall Version

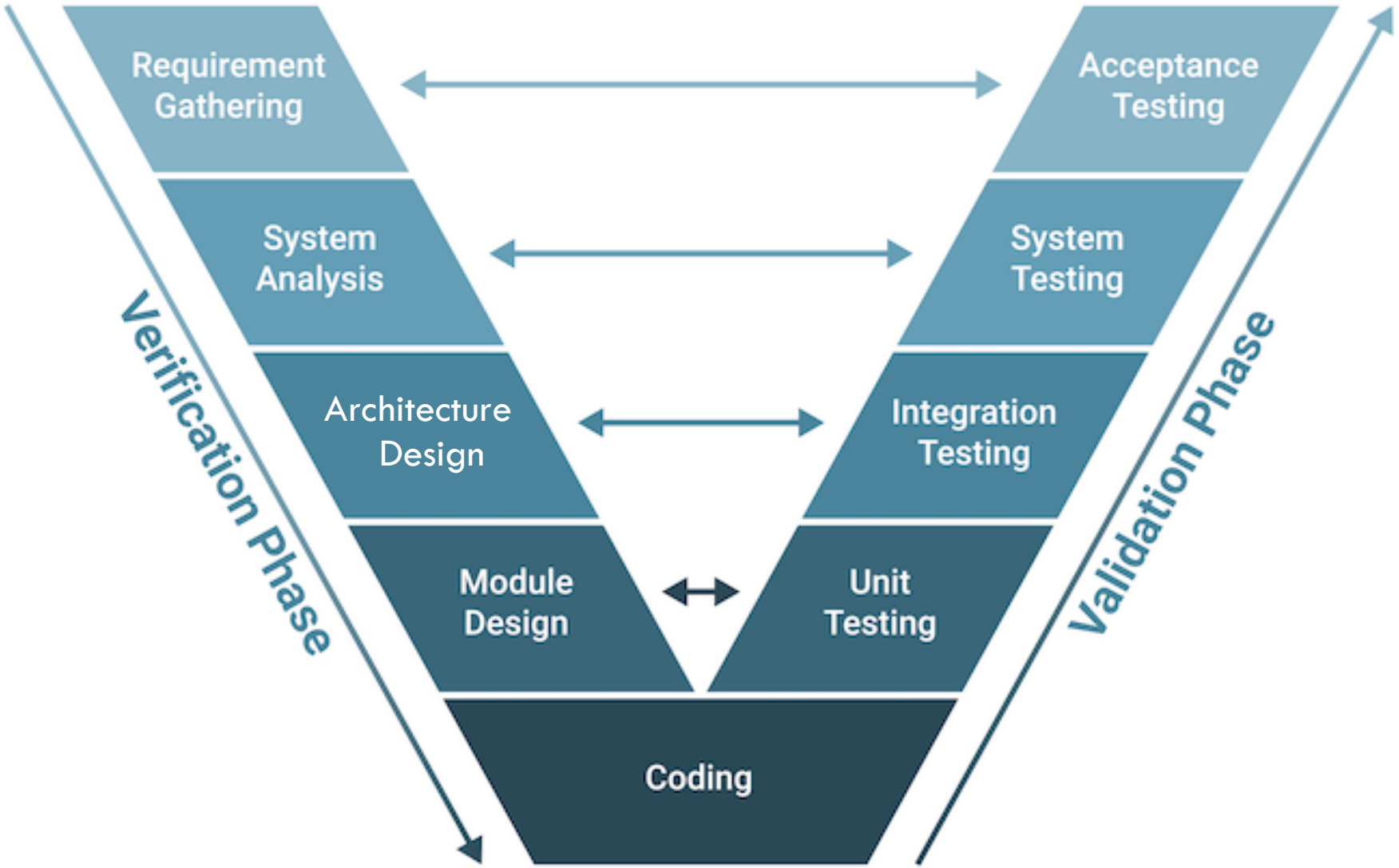
The Waterfall Model

- Pros
  - It is easy to understand and plan.
  - It is easy to manage due to the rigidity of the model.
  - It works for well-understood small projects.
  - Analysis and testing are straightforward.
- Cons
  - It does not accommodate change well.
  - There is a lack of parallelism
  - The use of resources is inefficient.
  - Testing occurs late in the process.
  - Customer approval is at the end.

V Model

A variation of the waterfall model that demonstrates how the testing activities are related to analysis and design

V Model


Requirement Gathering
System Analysis
Architecture Design
Module Design
Coding
Unit Testing
Integration Testing
System Testing
Acceptance Testing
Verification Phase
Validation Phase

Verification and Validation (V & V)

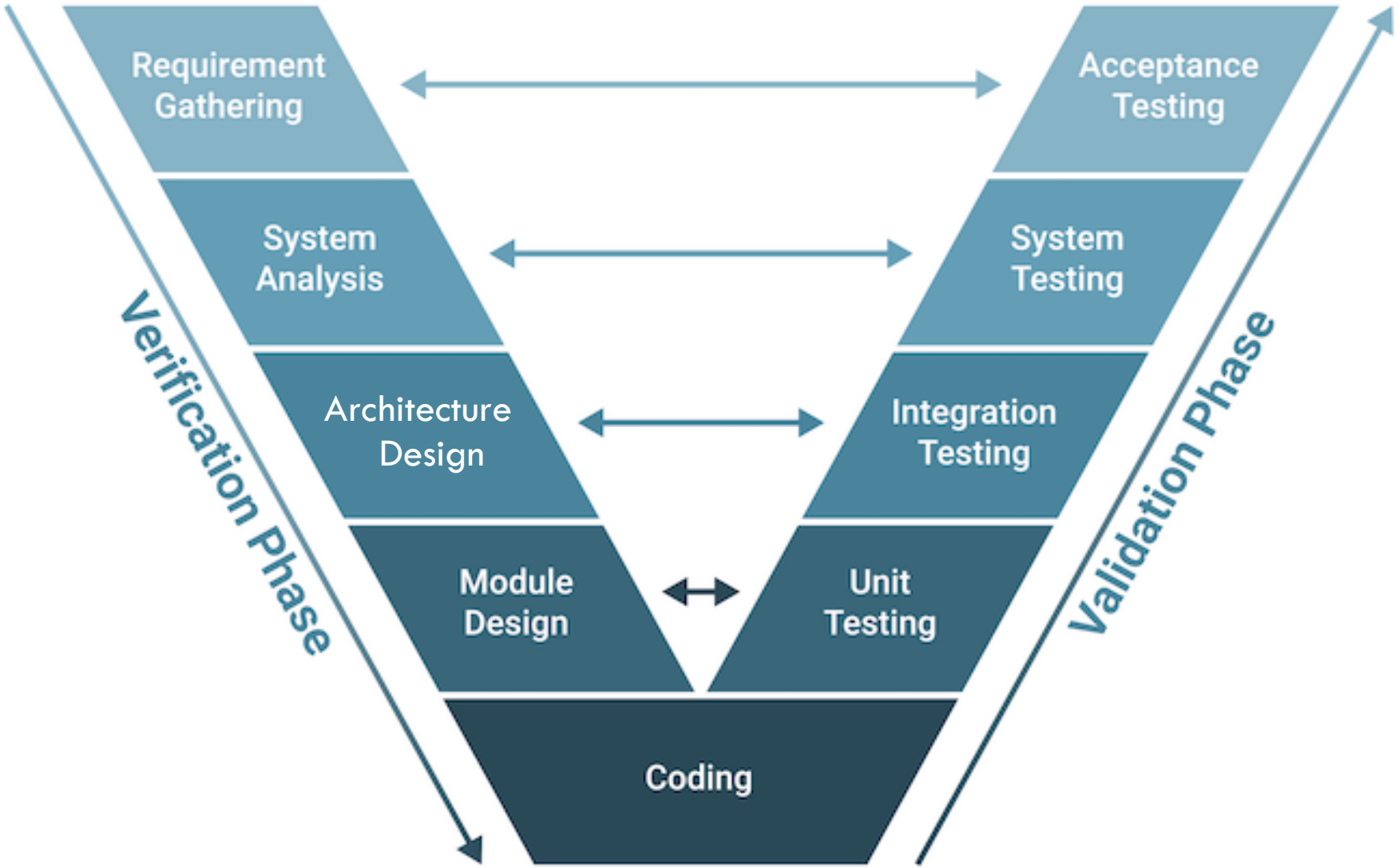
Verification - การทวนสอบซอฟต์แวร์ เน้นที่กระบวนการ

"Are we building
the product/software right?"

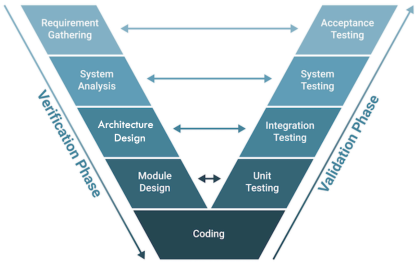
Validation - ทดสอบเพื่อยืนยันผล (การทำงาน) ของ ซอฟต์แวร์ หรือเน้นที่การทดสอบผลิตภัณฑ์ (ซอฟต์แวร์)

"Are we building
the right product/software?

V Model


Requirement Gathering
System Analysis
Architecture Design
Module Design
Coding
Unit Testing
Integration Testing
System Testing
Acceptance Testing
Verification Phase
Validation Phase

V model — Verification Phases

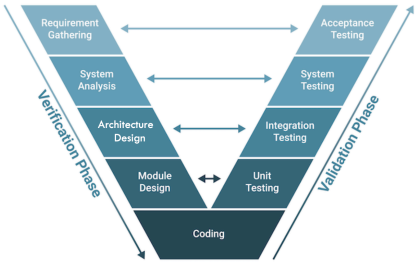


inception

1. (Business) **Requirement Analysis** – The first and important phase in SDLC, where the product requirements are understood from the customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

2. **System Design** – Comprising of understanding and detailing the complete *hardware and communication setup* for the product under development. System test design planning is done based on the system design. Doing this at an earlier stage leaves more time for actual test execution later.

V model — Verification Phases

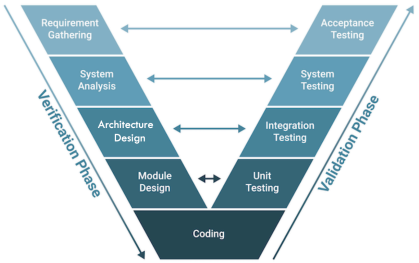


3. **Architectural Design –** *Architectural specifications* are understood and designed, based upon the technical and financial feasibility. Usually more than one technical approach is proposed to find out all possible alternatives, which will then be used for final decision making. System design is broken down further into modules taking up different functionality. This is also referred to as *High Level Design (HLD)*.

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined. With this information, integration tests can be designed and documented during this stage.

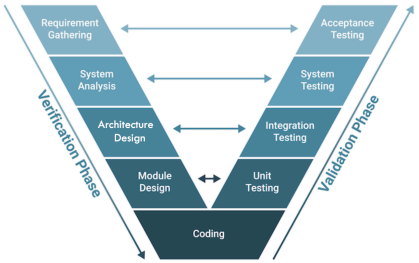
4. **Module Design –** The detailed *internal module design* for all the system modules is specified, referred to as *Low Level Design (LLD)*. It is important that the design is compatible with the other modules in the system architecture and the other external systems. Unit tests can be designed at this stage based on the internal module designs.

V model — Verification Phases



5. **Coding** or Implementation – The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements. The coding is performed based on the coding guidelines and standards, aka *coding conventions*. The code goes through numerous code reviews (this can be done by using *Pair Programming Technique*) and is optimized for best performance before the final build is checked into the repository.

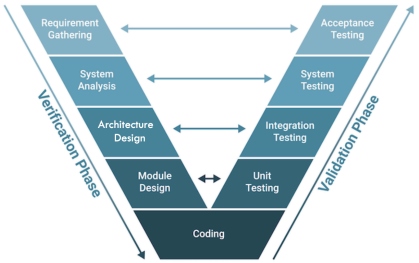
V model — Validation Phases



6. **Unit Testing** – The essential part of any development process, which helps eliminate the maximum faults and errors at a very early stage. Unit tests designed in the module design phase (4-th phase) are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

7. **Integration Testing** – Integration testing is associated with the architectural design phase (3-th phase). Integration tests are performed to test the coexistence and communication of the internal modules within the system.

V model — Validation Phases



8. **System Testing** – System testing is directly associated with the System design phase (2-th phase). System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during system test execution.

9. (User) **Acceptance Testing** – Acceptance testing is associated with the business requirement analysis phase (1-th phase) and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non functional issues such as load and performance defects in the actual user environment.

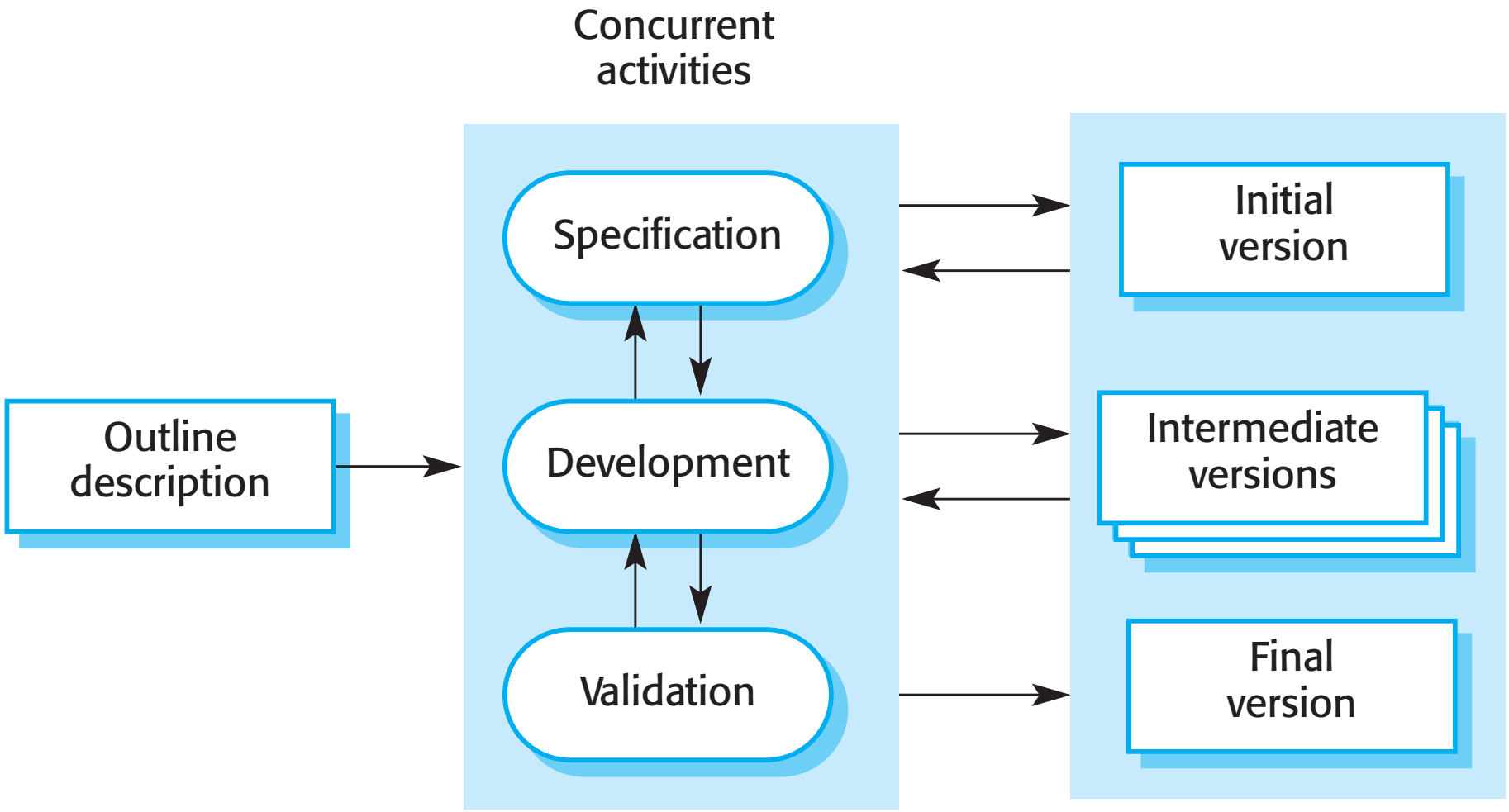
V model

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

V model

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

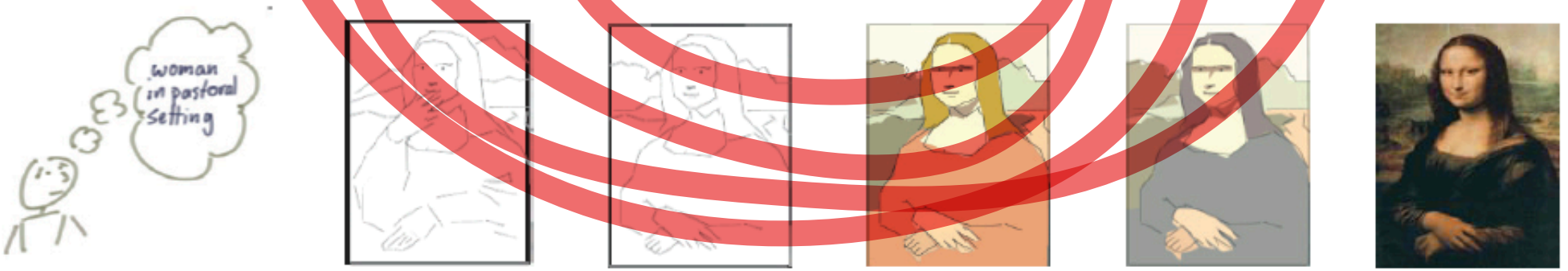
Incremental Development



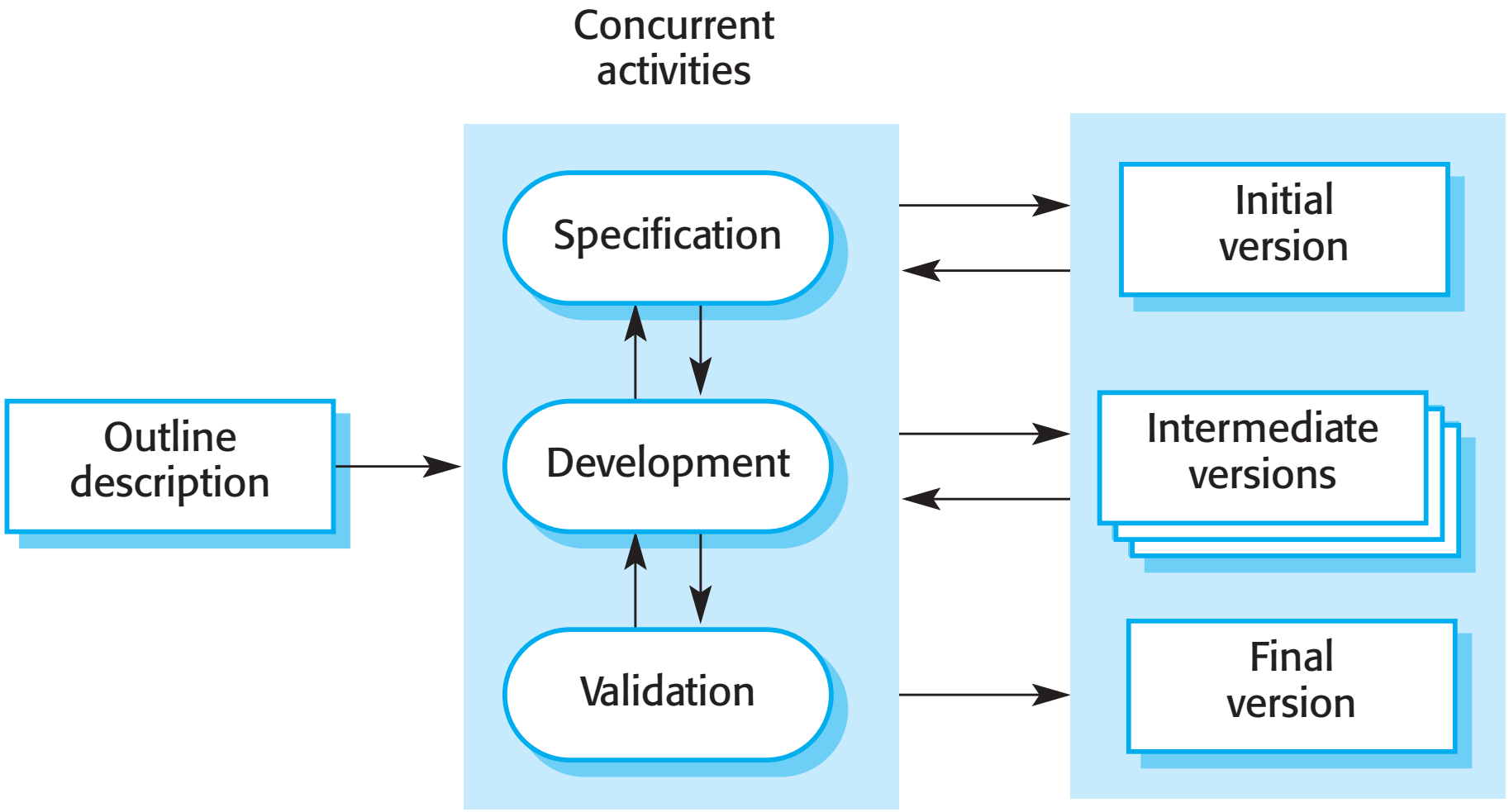
Incremental vs. Iterative

Incremental Approach: Developing the product in pieces considering the original idea

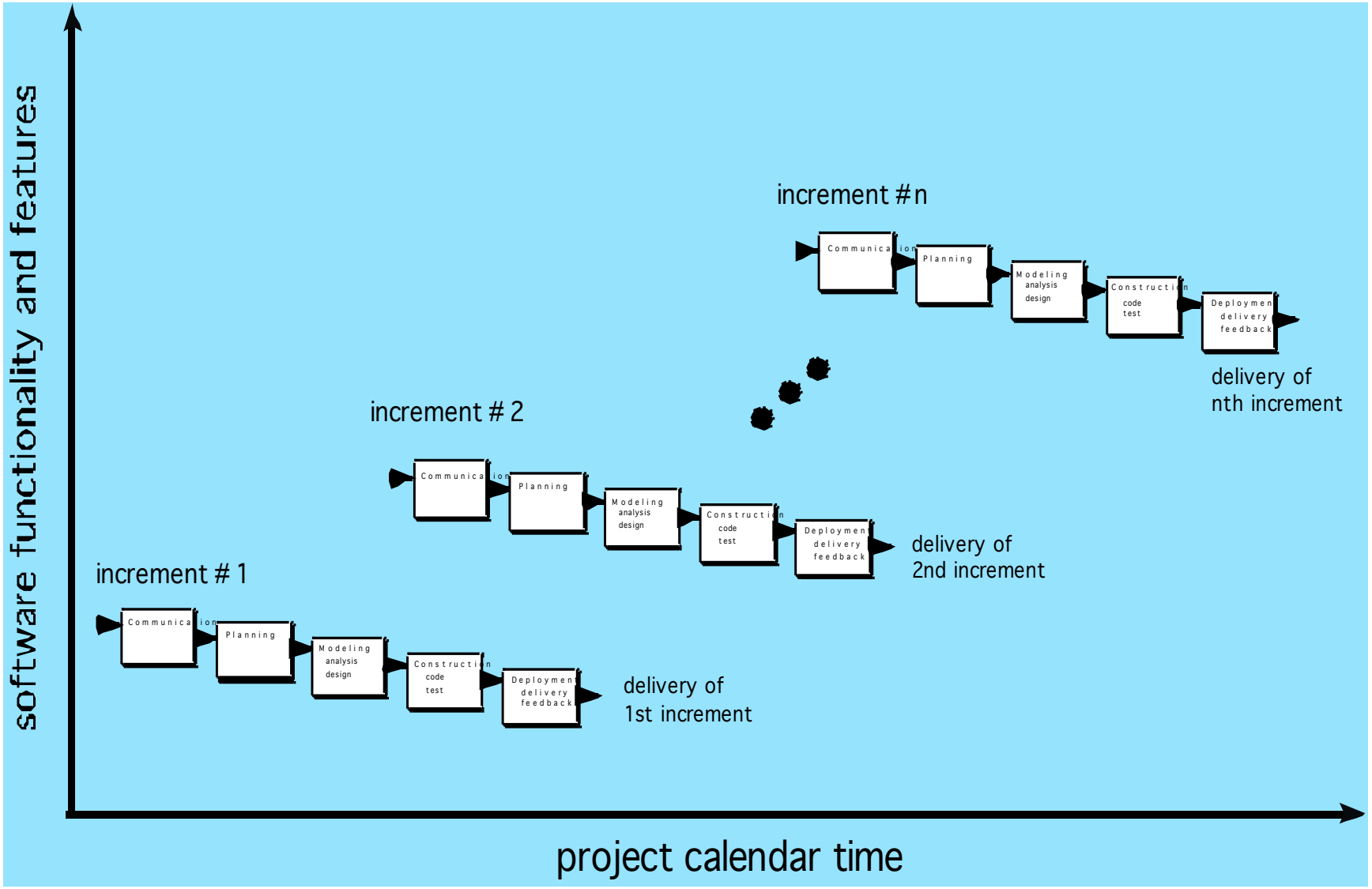
Iterative Approach: Developing the raw version of the product, validating it, and then building it further to increase its quality and functionality



Incremental Development



Incremental Development



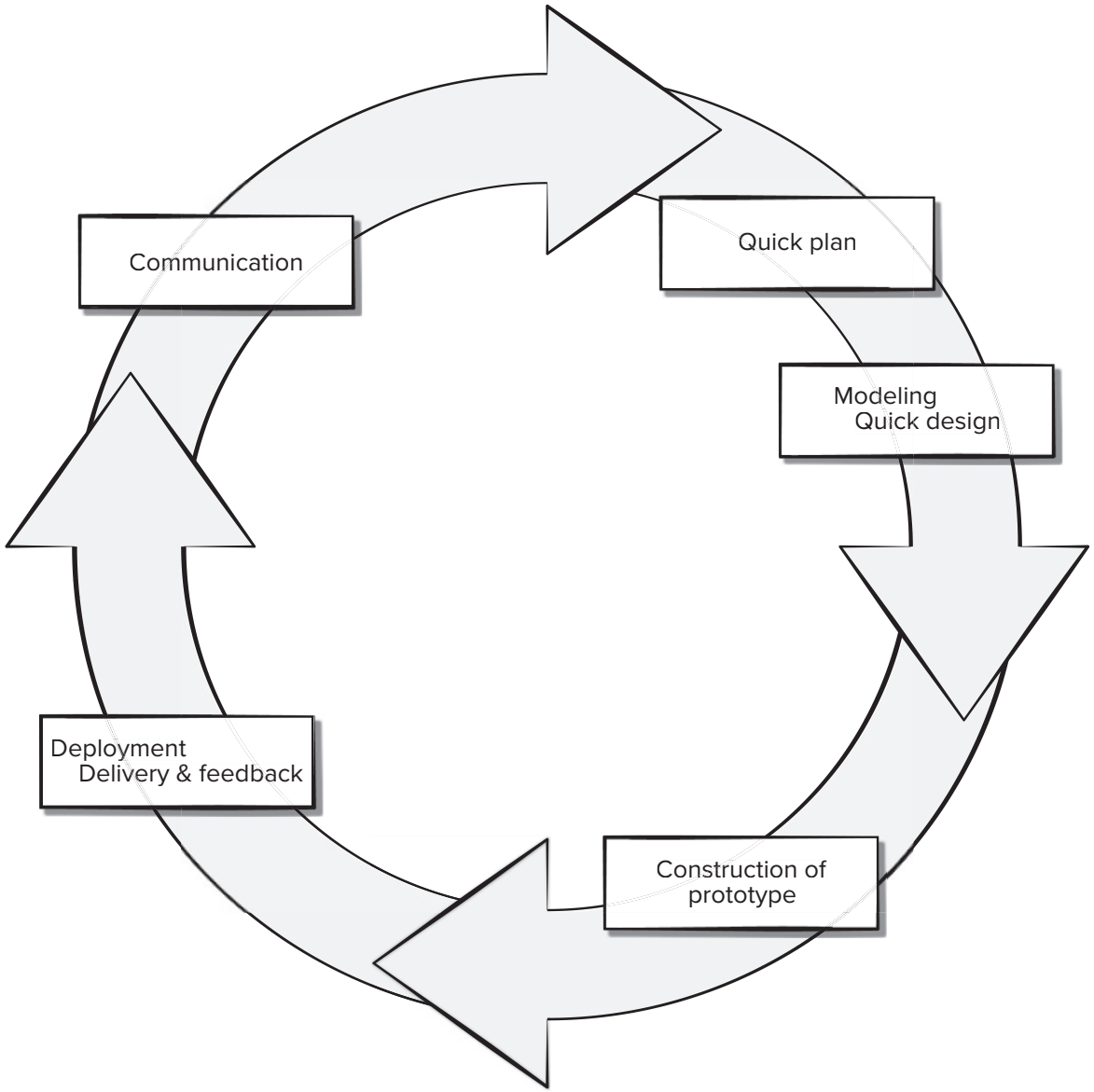
Incremental Development

- Pros (over the waterfall model)
  - The cost of implementing requirements changes is reduced.
  - It is easier to get customer feedback on the development work that has been done.
  - Early delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental Development

- Cons
  - The process is not visible. Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost effective to produce documents that reflect every version of the system.
  - System structure tends to degrade as new increments are added.

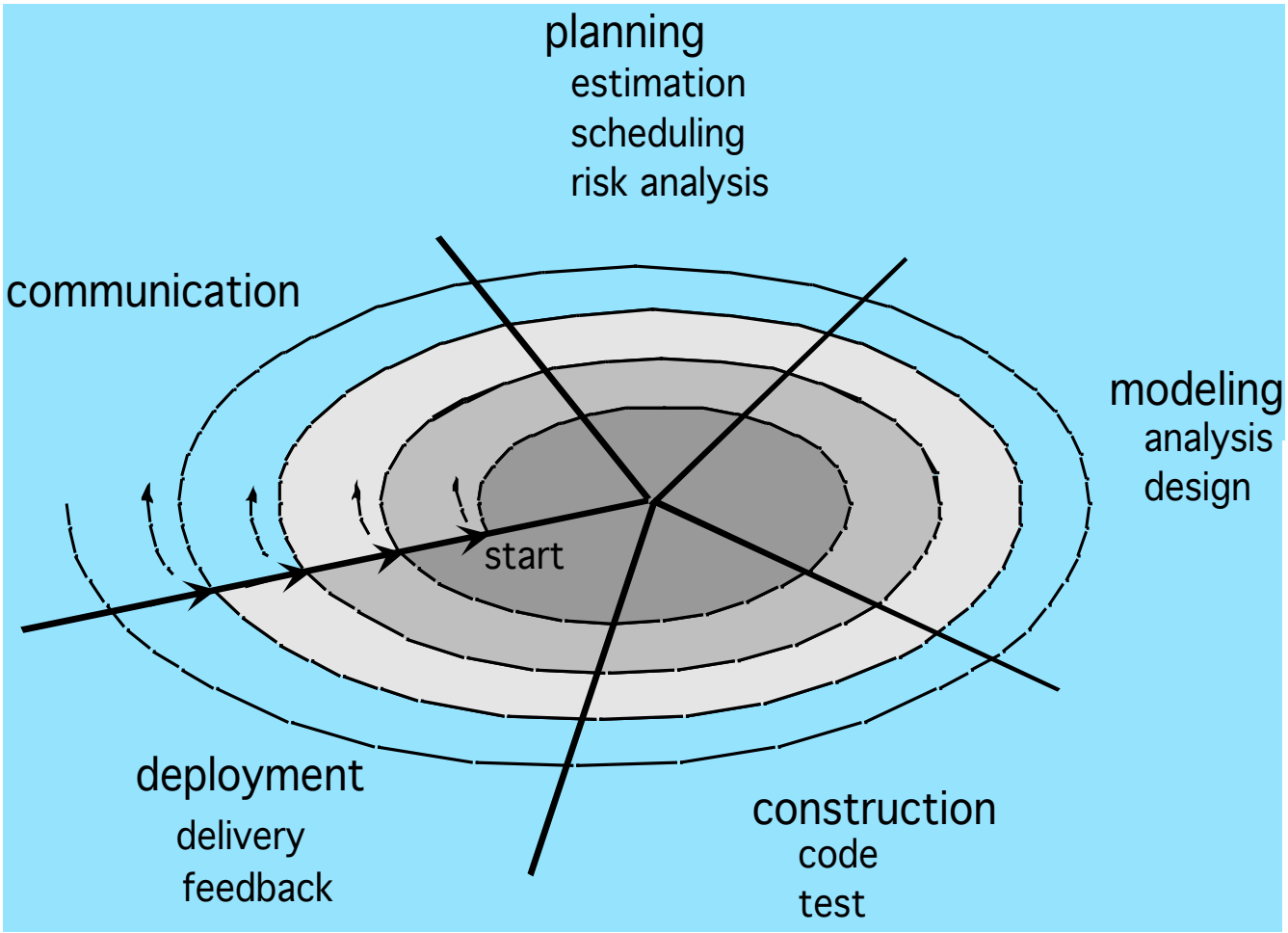
Prototyping

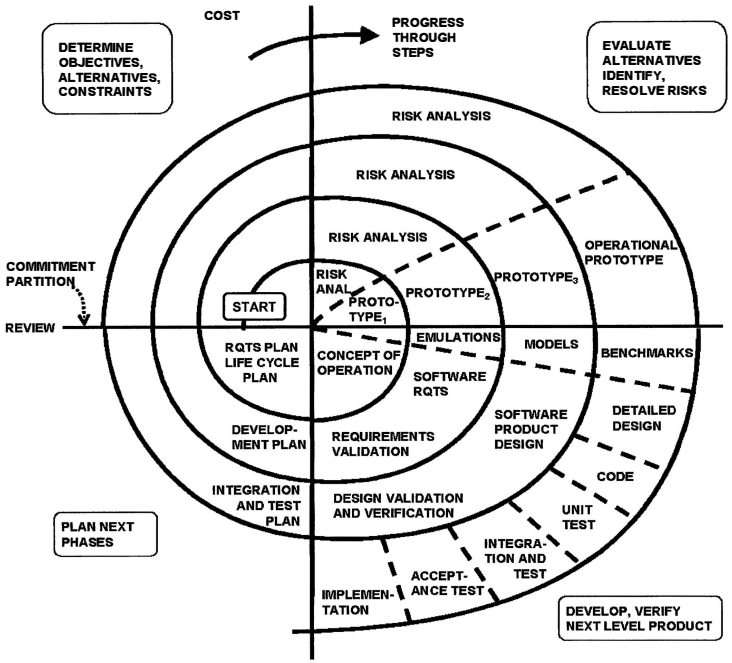


Prototyping

- Pros
  - There is a reduced impact of requirement changes.
  - The customer is involved early and often.
  - It works well for small projects.
  - There is reduced likelihood of product rejection.
- Cons
  - Customer involvement may cause delays.
  - There may be a temptation to “ship” a prototype.
  - Work is lost in a throwaway prototype.
  - It is hard to plan and manage.

Spiral



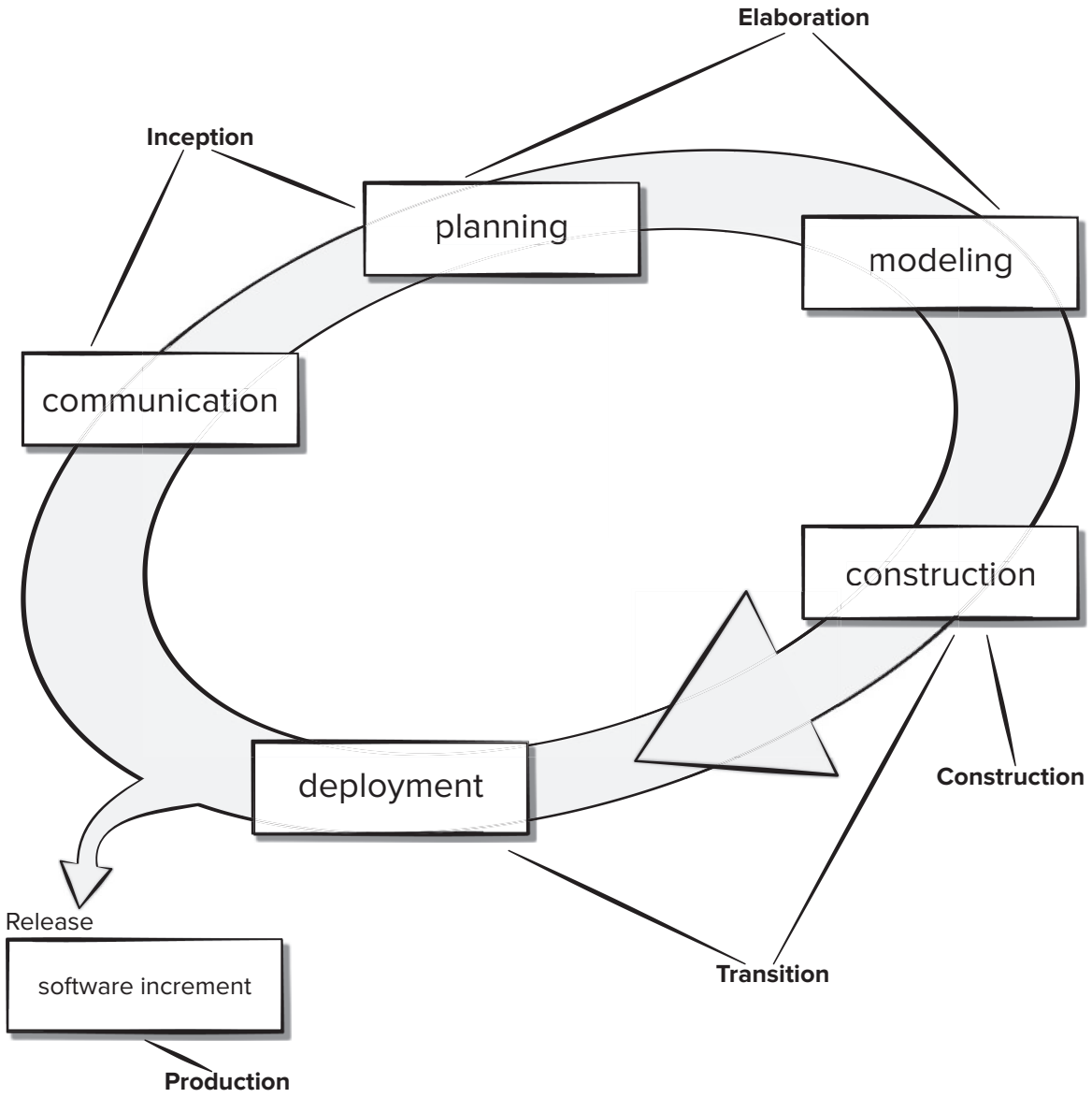


Barry Boehm "A Spiral Model of Software Development and Enhancement" IEEE Computer, vol.21, #5, May 1988, pp 61-72.

Spiral

- Pros
  - There is continuous customer involvement.
  - Development risks are managed.
  - It is suitable for large, complex projects.
  - It works well for extensible products.
- Cons
  - Risk analysis failures can doom the project.
  - The project may be hard to manage.
  - It requires an expert development team.

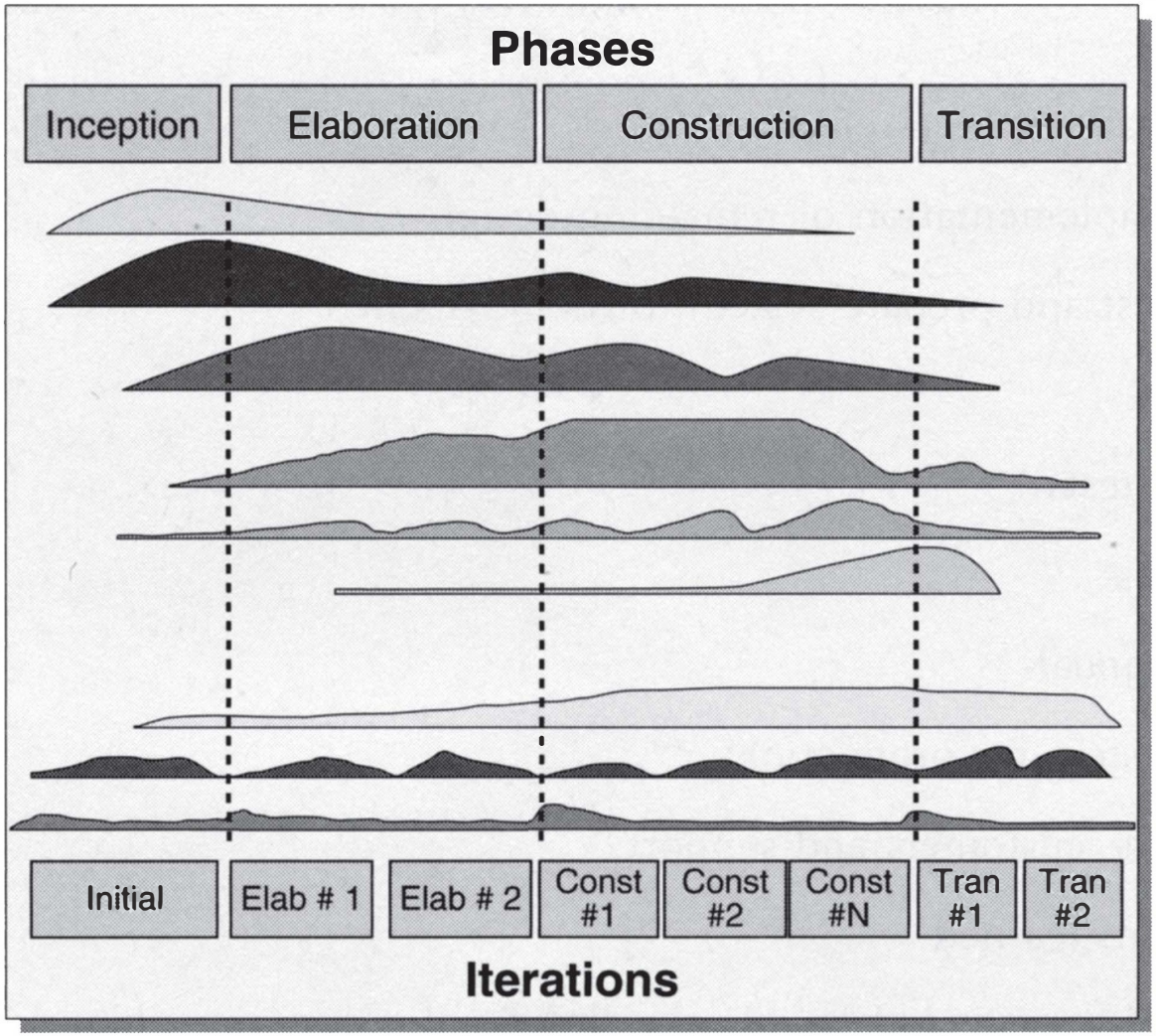
The Unified Process (UP)

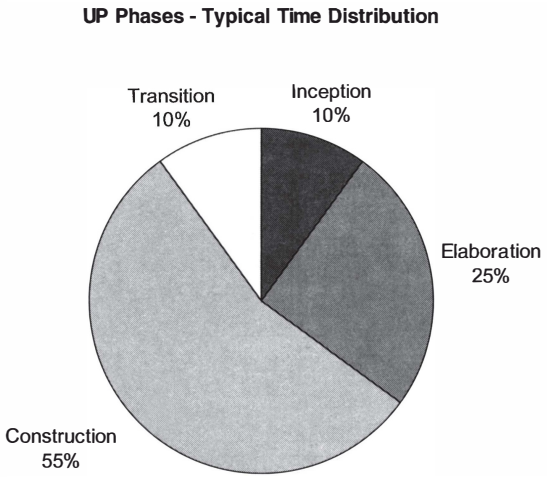


The UP is a "use-case driven, architecture-centric, iterative and incremental" software process

UP Phases

http://www.ambysoft.com/downloads/managersIntroToRUP.pdf





UP is simply a generic framework that should be customized for specific organizations or projects.
The best-known and extensively documented refinement of the UP is the Rational Unified Process (RUP)

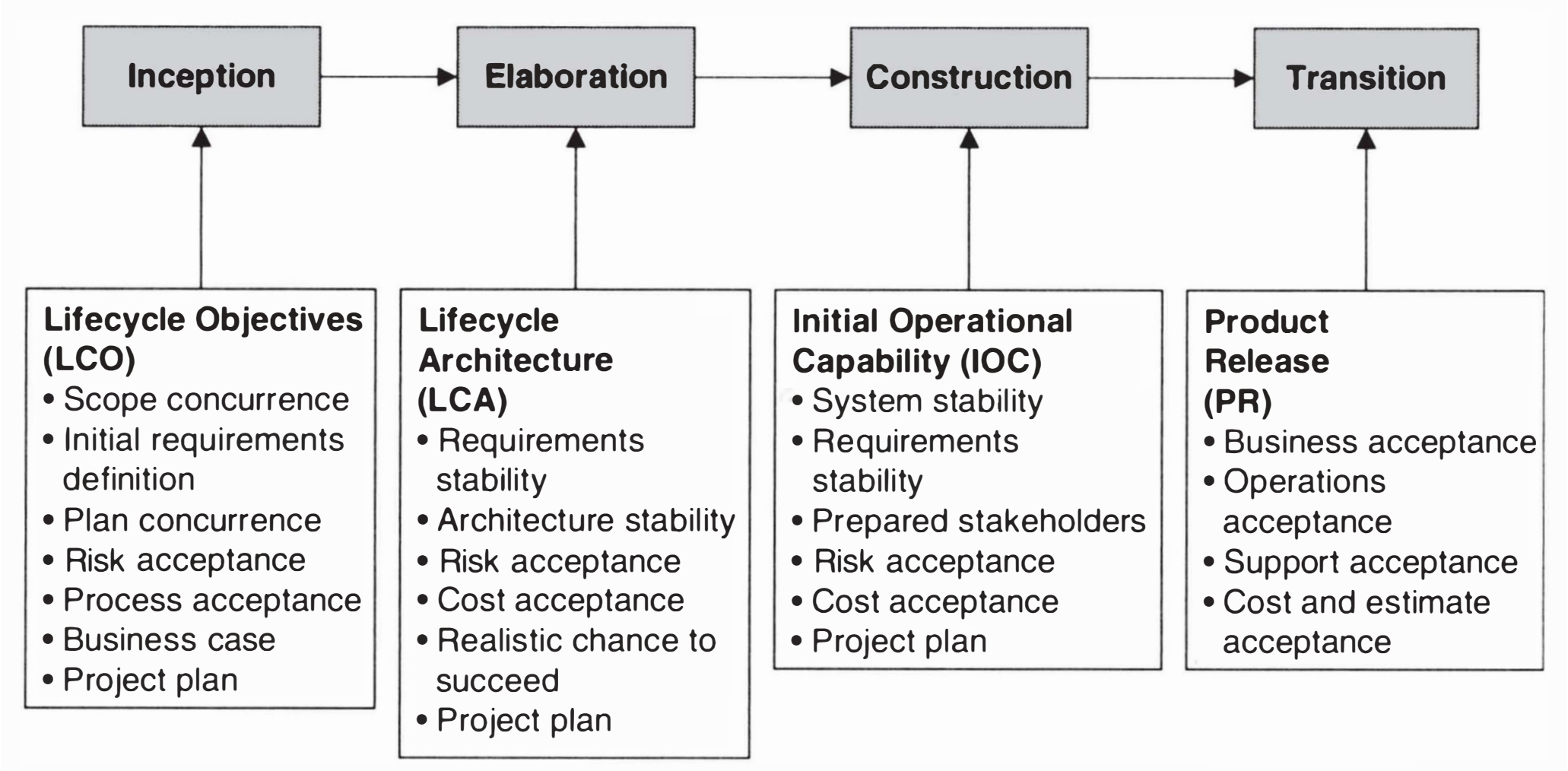
UP Phases

- **Inception**
  - Establish feasibility
  - Make business case
  - Establish product vision and scope
  - Estimate cost and schedule, including major milestones
  - Assess critical risks
  - Build one or more prototypes
- **Elaboration**
  - Specify requirements in greater detail
  - Create architectural baseline
  - Perform iterative implementation of core architecture
  - Refine risk assessment and resolve highest risk items
  - Define metrics
  - Refine project plan, including detailed plan for beginning Construction iterations

UP Phases

- **Construction**
  - Complete remaining requirements
  - Do iterative implementation of remaining design
  - Thoroughly test and prepare system for deployment
- **Transition**
  - Conduct beta tests
  - Correct defects
  - Create user manuals
  - Deliver the system for production
  - Train end users, customers and support
  - Conduct lessons learned

Objectives for UP

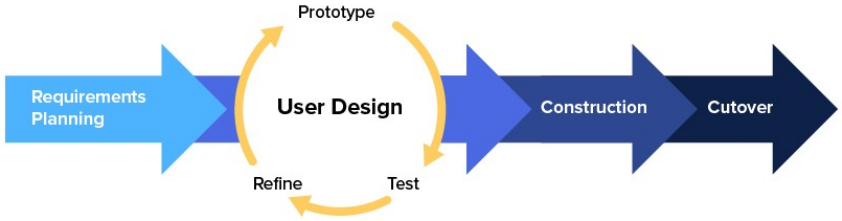


UP

- Pros
  - Quality documentation is emphasized.
  - There is continuous customer involvement.
  - It accommodates requirements changes.
  - It works well for maintenance projects
- Cons
  - Use cases are not always precise.
  - It has tricky software increment integration.
  - Overlapping phases can cause problems.
  - It requires an expert development team.

Still Other Process Models

- RAD (Rapid Application Development)—an adaptive model based on prototyping and quick feedback with less emphasis on specific planning.



- Component based development—the process to apply when reuse is a development objective
- Formal methods—emphasizes the mathematical specification of requirements
- AOSD—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects*