Topic 03:
Prescriptive Process Models

References

- [Pressman, 2010] Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, 7th Edition, Mc. Graw Hill International, USA, 2010.
- [Sommerville, 2011] Sommerville, Ian, *Software Engineering*, 9th Edition, Pearson-Addison Wesley, England, 2011.
- [Dennis] Dennis, Alan, et. al., System Analysis and Design with UML 3rd Edition, John Wiley & Sons, 2010

Outlines

- Prescriptive Process Models
- Waterfall Process Model
- Incremental Process Model
- Evolutionary Process Models
  - ❖Prototyping
  - ❖Spiral
- Unified Process

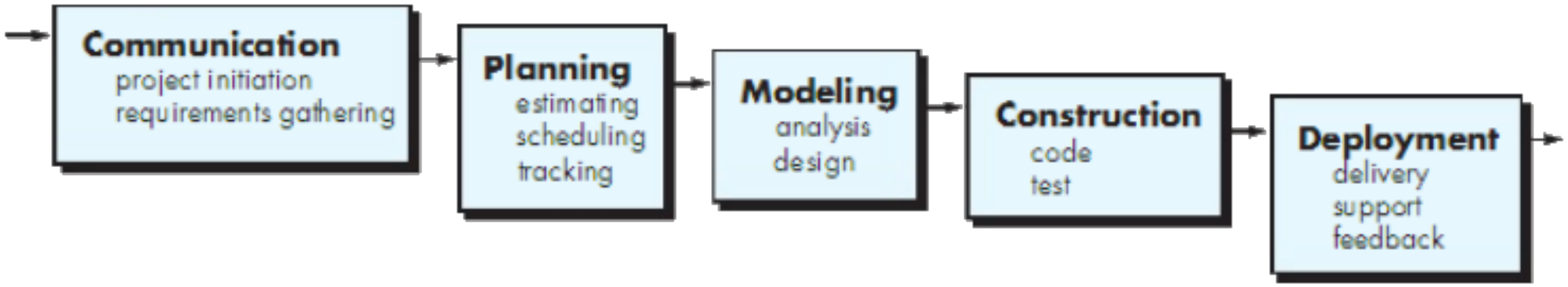
Prescriptive Process Models

- A prescribed set of process elements and a predictable process workflow
  - ❖Focuses on **structure** and **order**
- Also known as “conventional” / “traditional” process model

Waterfall Process Model

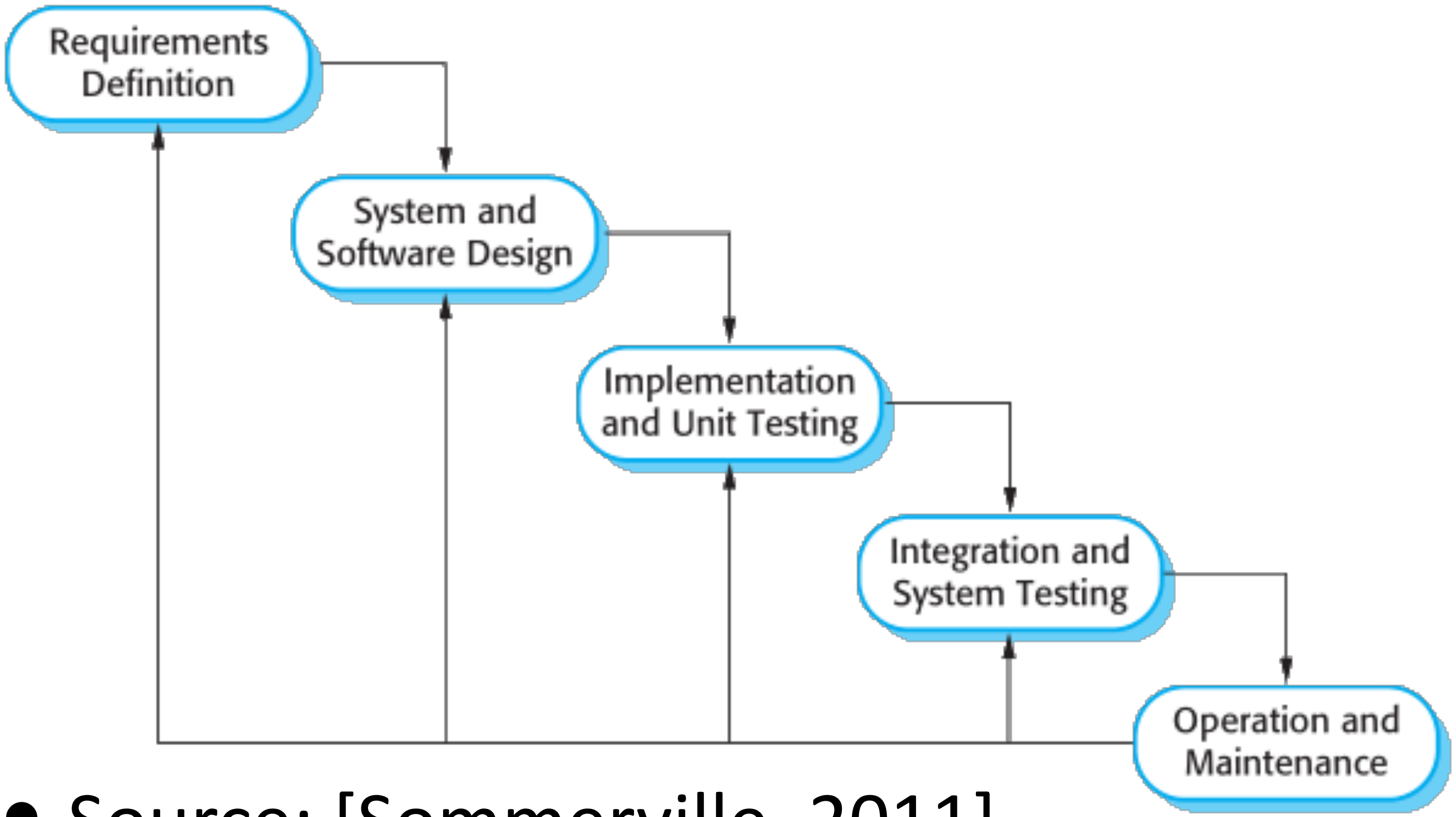
- Systematic, sequential approach to software development
- Each activities are done sequentially
- Proposed in 1970
    - Allows iterative flows (“feedback loops”)
    - Mainstream adopted the process in linear fashion, i.e. without the “feedback loops”

Waterfall Process Model



- Source: [Pressman, 2010]

Waterfall Process Model


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

- Source: [Sommerville, 2011]

Waterfall Process Model

- Use Waterfall when:
  - ❖Requirements are fixed and well defined since beginning of the project
  - ❖Development work flows linearly

Waterfall Process Model : Pros & Cons

- Similar to process models used in other engineering fields
- High project visibility, i.e. easy to track

- Hard to accommodate changes in the middle of project
- Real project rarely follow sequential flow
- Working product will not be available until late in the project timeline

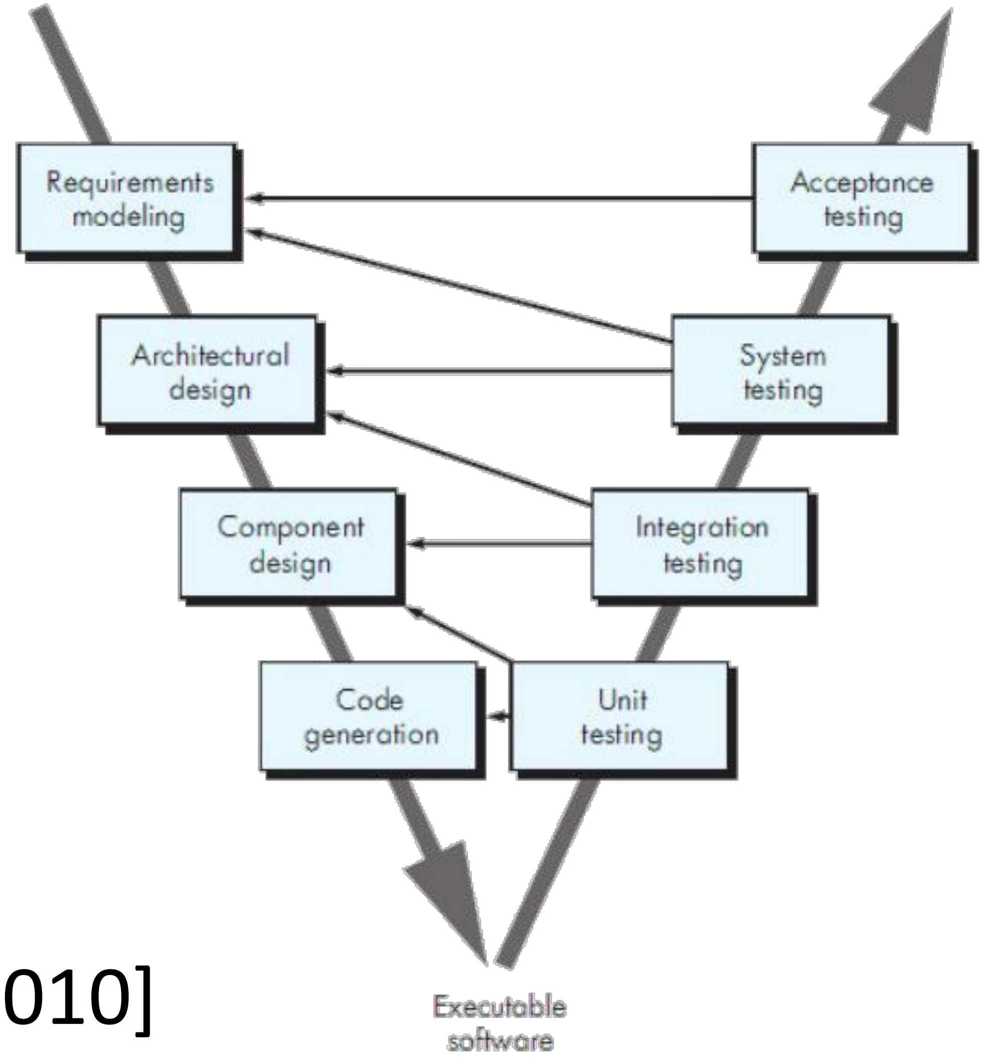
Waterfall [Dennis]

- Advantages:
  - ❖The system requirements are identified long before programming begins.
  - ❖Changes to the requirements are minimized as the project proceeds.
- Disadvantages:
  - ❖The design must be completely specified before programming begins → A long time elapses between the completion of the system proposal in the analysis phase and the delivery of the system.

Waterfall Process Model

- Variants:
  - **V-shaped model** : quality assurance actions (verification and validation) related to each activities
  - **Prototyping variant** : requirements and design prototypes

Waterfall Process Model : V-shaped Model

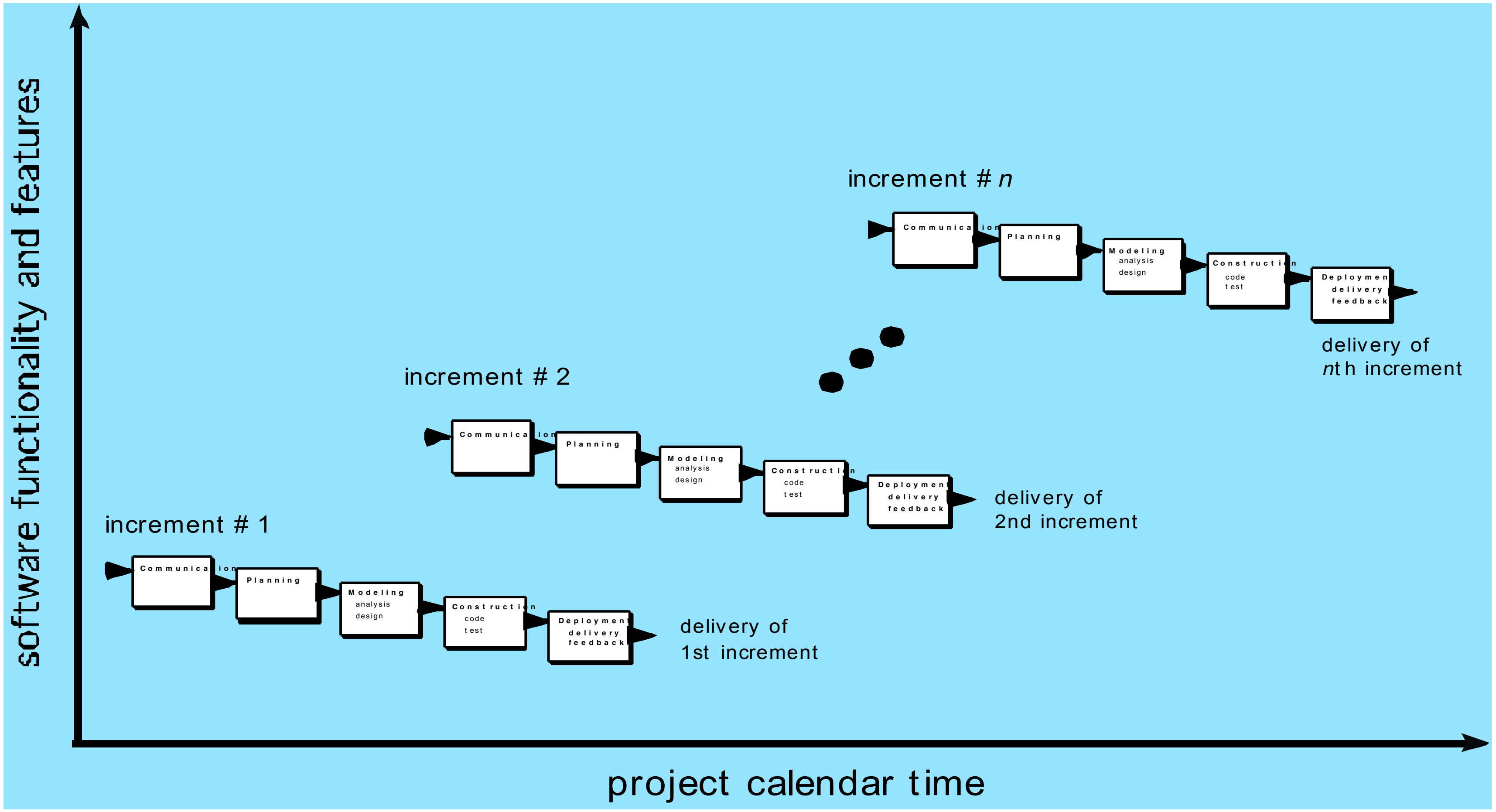


- Source: [Pressman, 2010]

Incremental Process Model

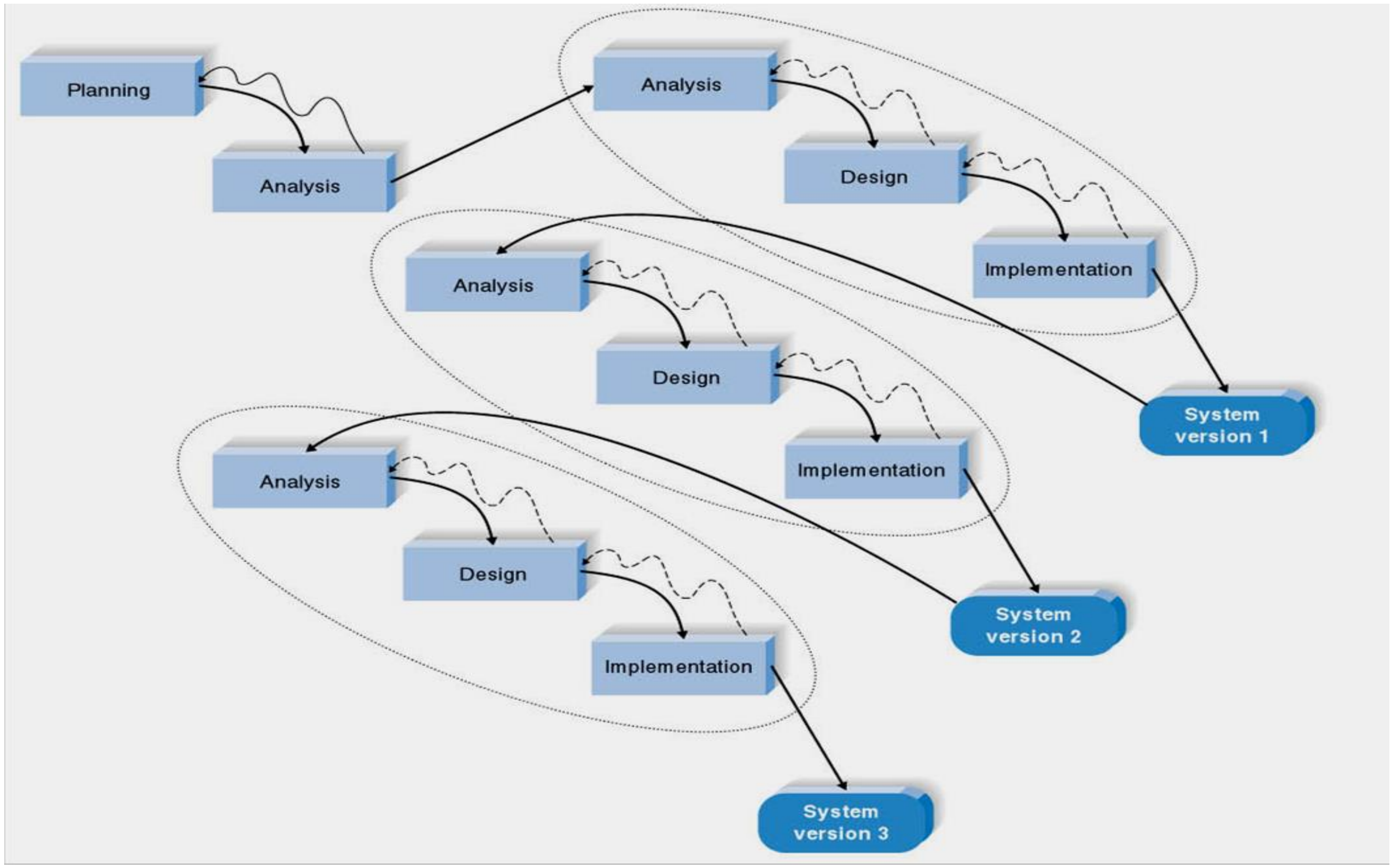
- Producing software by "increments"
  - ❖E.g. developing a word processor : 1st increment provides basic text editing and file management, 2nd increment provides text styling, and so forth
- Combines linear and parallel flows
- Incremental development reflects the way we solve problems
  - ❖Move toward a solution through series of steps, backtrack when made a mistake

Incremental Process Model



- Source: [Pressman, 2010]

Phased Development [Dennis]



Incremental Process Model

- Use incremental model when:
  - ❖Requirements are well understood
    - Can be partitioned into subsystems
  - ❖There is a need to deliver a set of functionality to users quickly
- Incremental development may be hard to adopt in large organisations that have established bureaucratic procedures

Incremental Process Model : Pros & Cons

+

Cost of accommodating changes is reduced

Easier to get customer feedback

Rapid delivery and deployment of useful software

−

The process is not visible

System structure tends to degrade as new increments are added

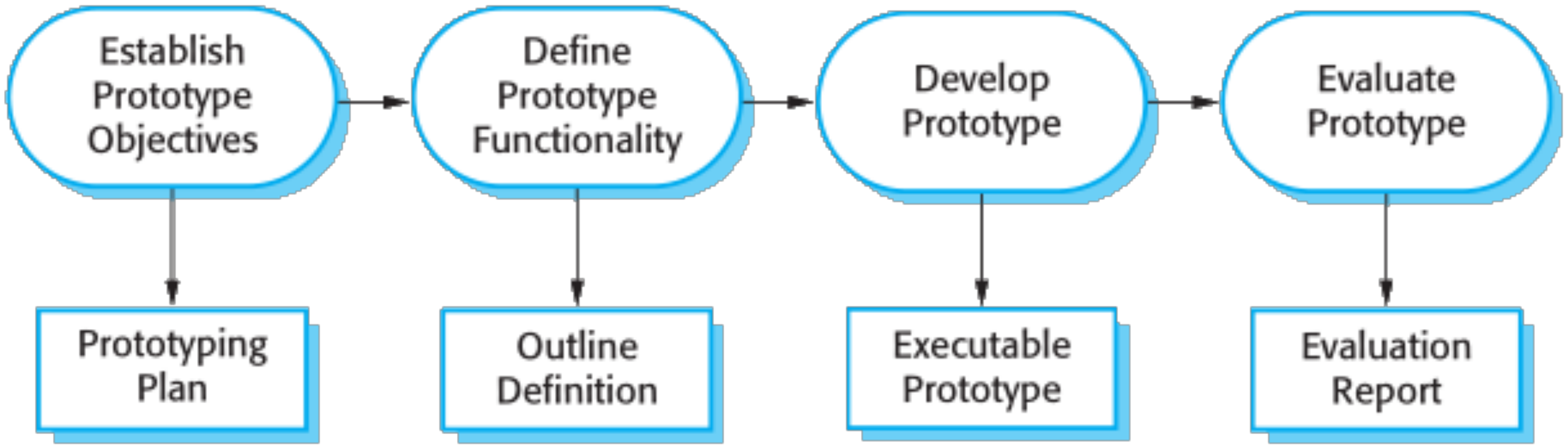
Evolutionary Process Models

- Software is expected to evolve over a period of time
- Initial requirements are well understood, but details of product or extensions have yet to be defined
- Producing software iteratively, increasingly more complete over time
- Not quite similar to incremental model!

Prototyping

- Prototype : “an initial version of a software system that is used to demonstrate concepts, try out design options, and find out more about problems and its possible solutions” [Sommerville, 2011]
- Assists you and other stakeholders to better understand what is to be built when requirements are fuzzy
- Users get a feel for the “actual system” and developers get to build something immediately
- Can be used as a stand-alone process or incorporated into activities in a process model

Prototyping

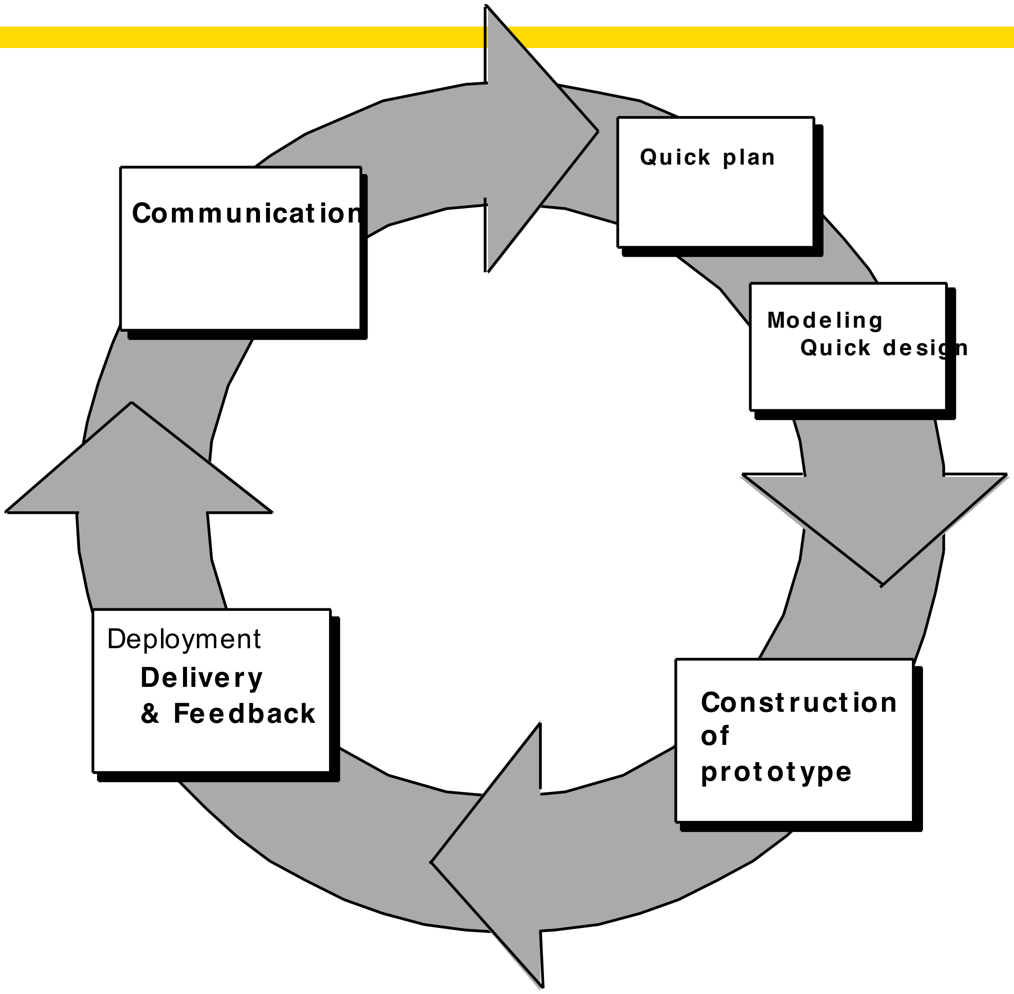


- Source: [Sommerville, 2011]

Prototyping

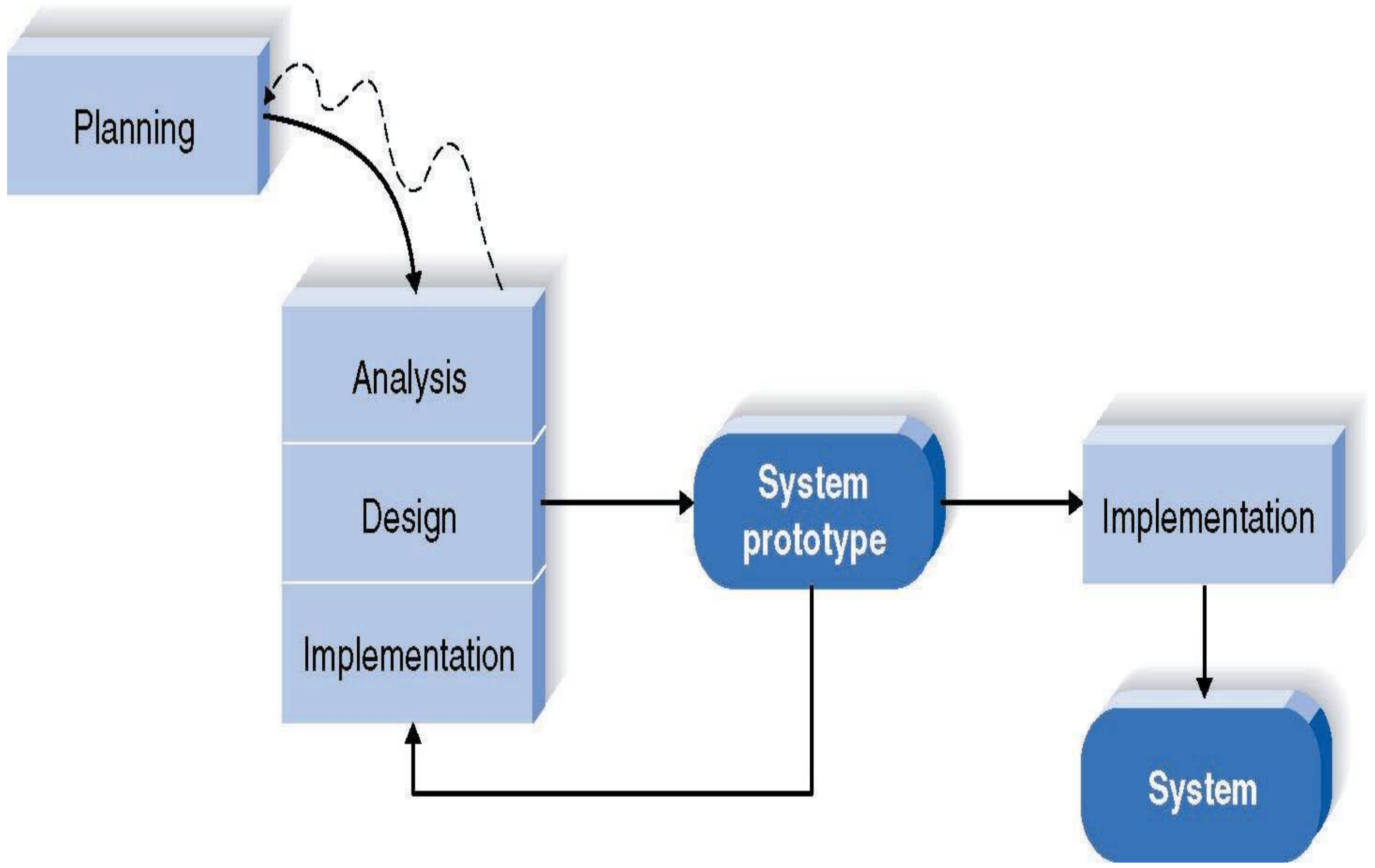
- Establish Prototype Objectives
  - ❖Make sure both team and customer have the same visions regarding the prototype
- Define Prototype Functionality
  - ❖Determine features and functions that will be included in the prototype
- Develop Prototype
  - ❖Design considerations may be relaxed to reduce cost and time
- Evaluate Prototype

Prototyping



- Source: [Pressman, 2010]

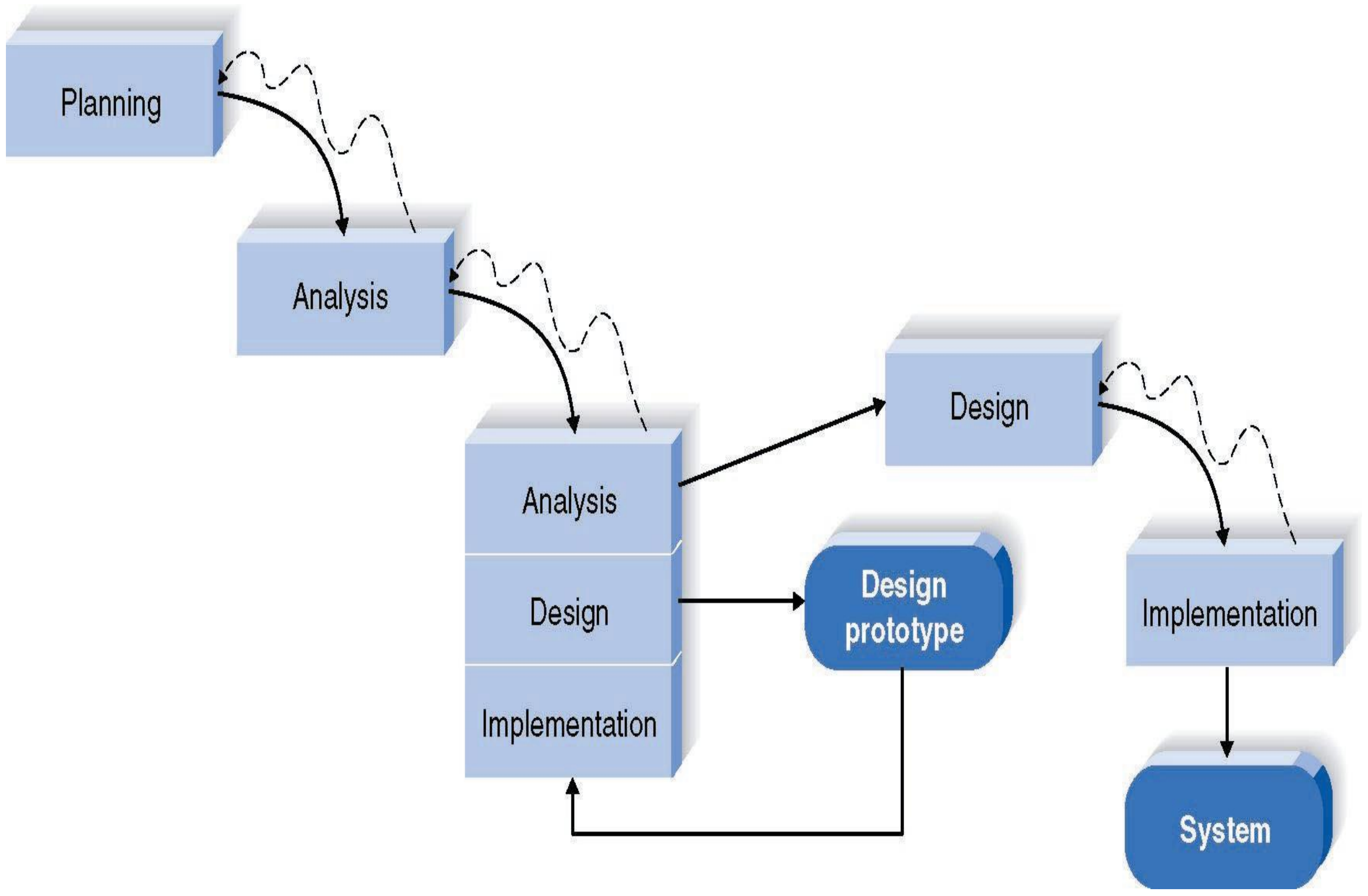
How Prototyping Works [Dennis]



Prototyping

- Communication
  - ❖Determine the objectives; identify known requirements
- Quick Plan
  - ❖Plan prototype iteration
- Modelling & Quick Design
  - ❖Focuses on elements that visible to end users
- Construction
- Deployment, Delivery, and Feedback

Throwaway Prototyping [Dennis]



Prototyping

- Throwaway prototype vs. prototype as actual system
  - ❖Compromises often made to get prototype done quickly
  - ❖Stakeholders often see the prototype as the “working product”
  - ❖In one idealistic view, first prototype should be “thrown away”
  - ❖Evolutionary view suggests prototype may evolve into actual system

Prototyping : Pros & Cons

Can clarify user requirements better

Better visibility of project’s progress

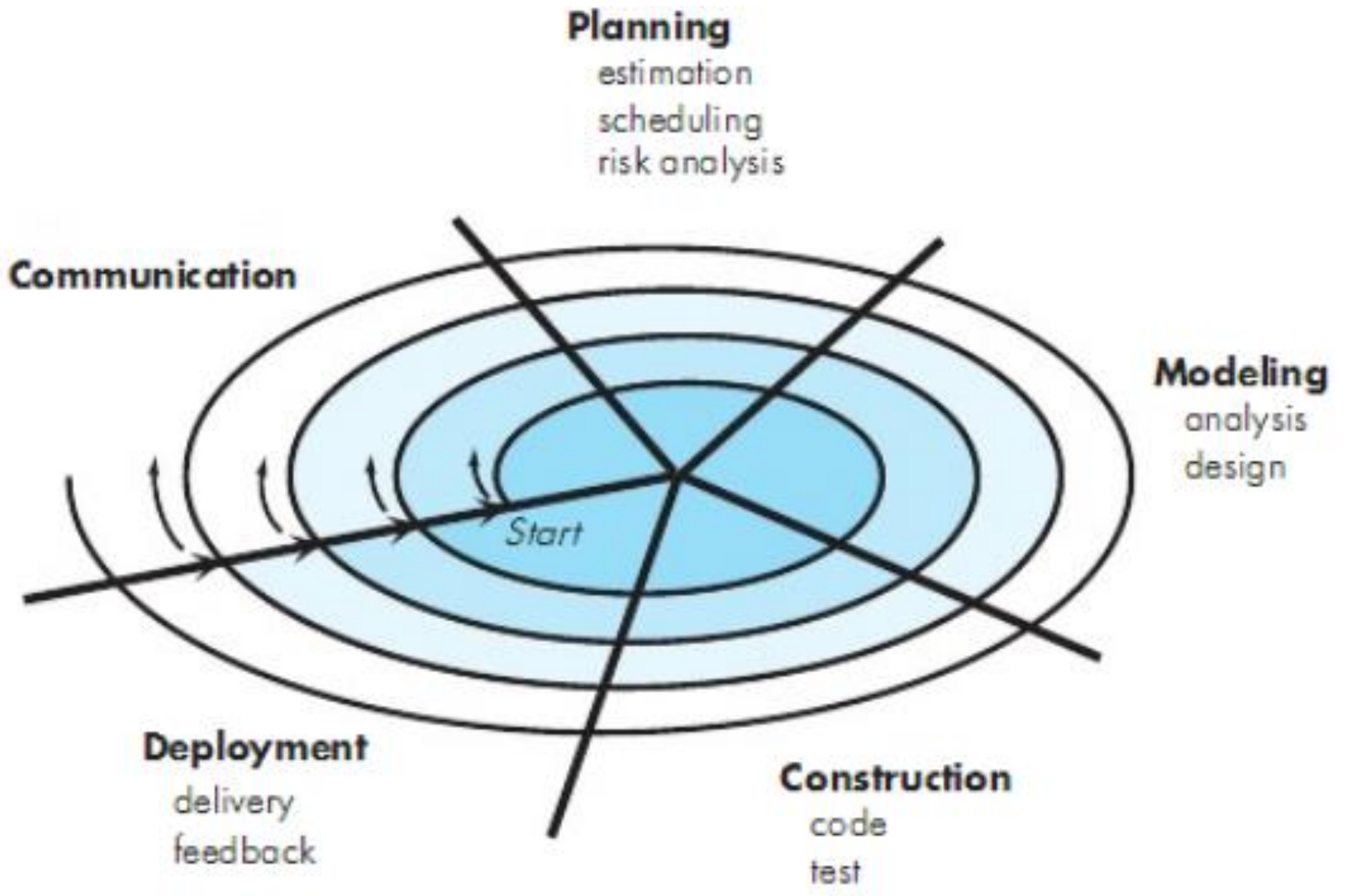
Less comprehensive analysis

Rapid changes may hard to document

Spiral Model

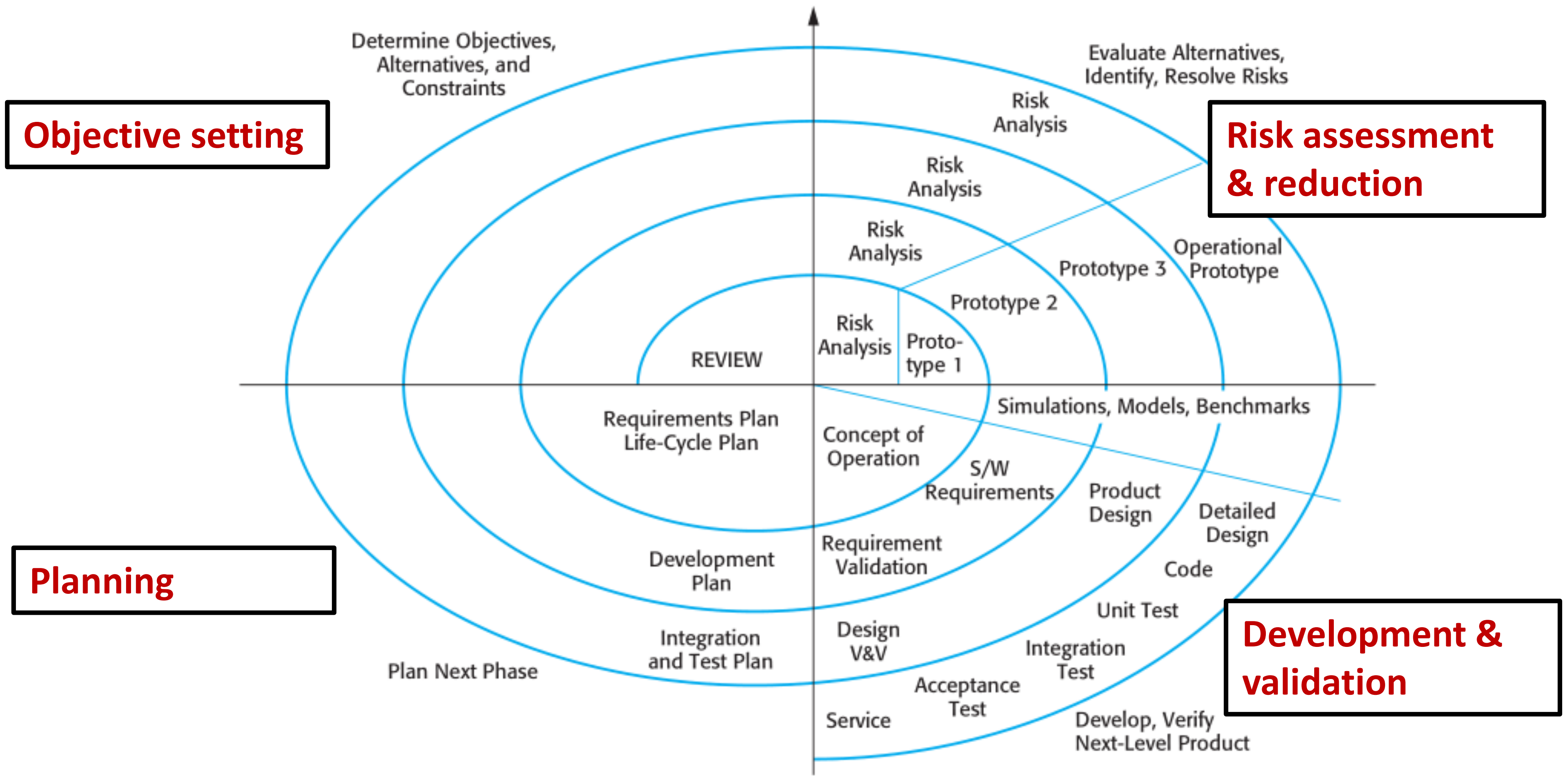
- Represented as a spiral instead of sequence of activities
- Each loop in the spiral represents a phase of the software process
- Includes explicit risk management activities

Spiral Model



- Source: [Pressman, 2010]

Spiral Model



- Source: [Sommerville, 2011]

Spiral Model

- Objective setting
  - ❖ Define objectives, constraints, deliverable that will be produced in current loop
  - ❖ Identify risks
- Risk assessment and reduction
  - ❖ Conduct detailed risk analysis
  - ❖ Try to reduce / resolve the risk

Spiral Model

- Development and validation
  - Choose a development model
- Planning
  - Review project
  - Decide whether to move to next phase or stay in current phase

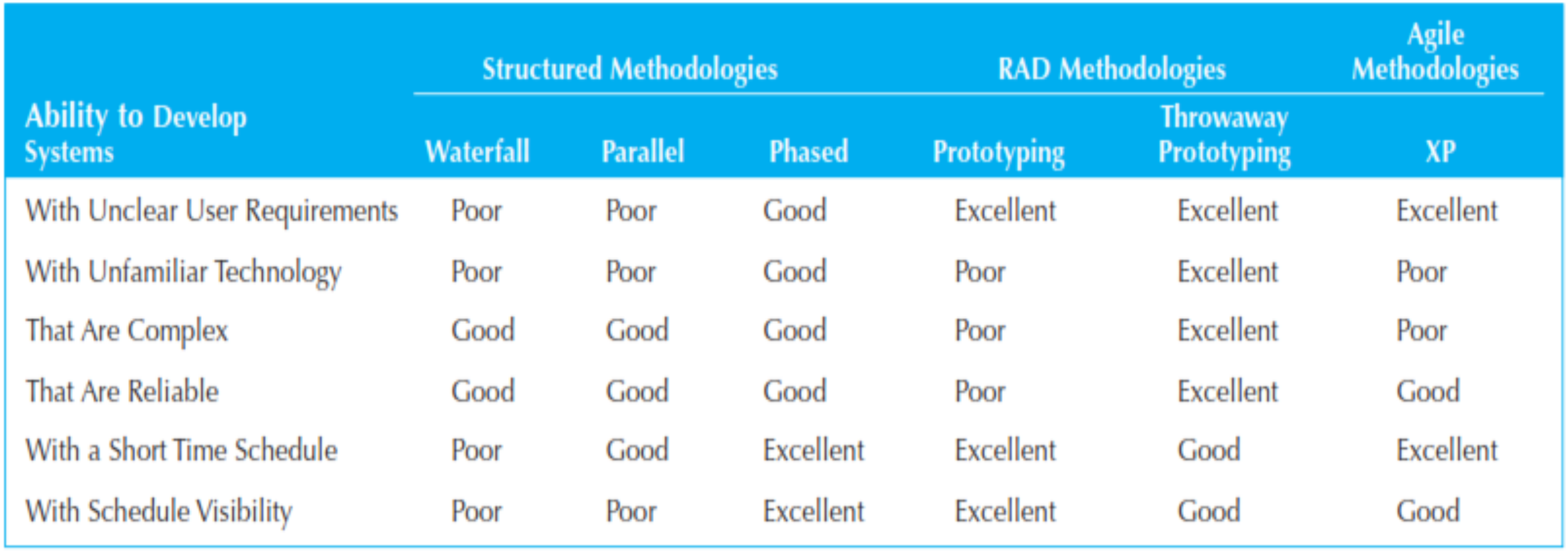
Spiral Model

- Example:
  - ❖First loop might result in the development of a product specification
  - ❖Subsequent loops might be used to develop a prototype
  - ❖Then, progressively more sophisticated versions of the software
  - ❖Each pass through the planning sector result in adjustments to project plan

Spiral Model : Pros & Cons

+	−
Explicit recognition of risk	Requires expertise in risk assessment
Can be adapted to apply throughout the life of the computer software	May be difficult to convince customers that evolutionary process is controllable
Allows iterative and concurrent elements of any process models described before	

Selecting Process Model/Methodology



	Structured Methodologies			RAD Methodologies		Agile Methodologies
Ability to Develop Systems	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP
With Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent
With Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Poor
That Are Complex	Good	Good	Good	Poor	Excellent	Poor
That Are Reliable	Good	Good	Good	Poor	Excellent	Good
With a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent
With Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Good

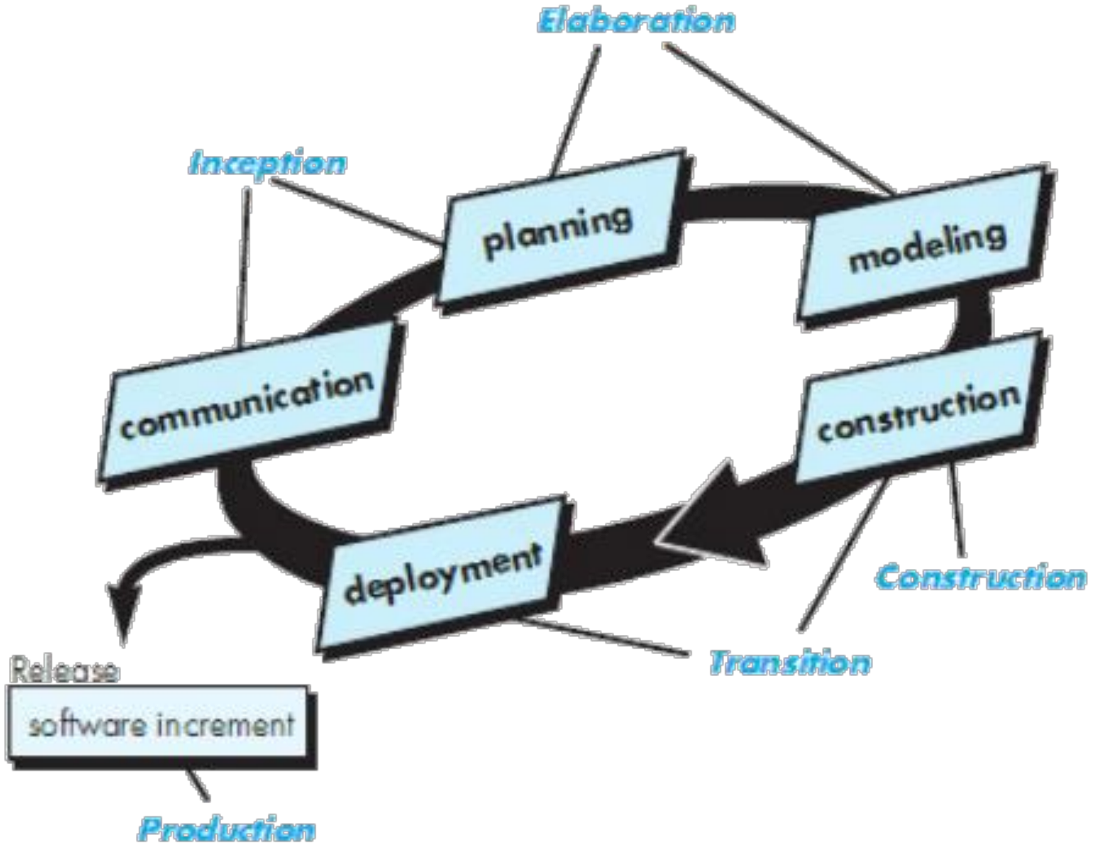
Unified Process

- Ivar Jacobson, Grady Booch, James Rambaugh (1990)
- An extensible process framework
  - Tailor as you need
  - Not all software projects are equal
- Specific implementations exist
  - **Rational Unified Process (Rational -> IBM)**
  - OpenUP (Eclipse)
  - Agile Unified Process

Unified Process : Characteristics

- Use-case driven
  - ❖ “Recognises the importance of customer communication & streamlined methods for describing the customer’s view of the system (use-case)”
  - ❖ Use-case driven means that use cases are the primary modeling tools defining the behavior of the system.
- Incremental & iterative
  - ❖ “Suggests a process flow that is iterative and incremental, providing the evolutionary feel that is essential in modern software development”
- Architecture-centric
  - ❖ “Emphasises the important role of software architecture”
  - ❖ Should support architectural views of a system: functional, static, and dynamic.
    - The functional, or external, view describes the behavior of the system from the perspective of the user.
    - The structural, or static, view describes the system in terms of attributes, methods, classes, and relationships.
    - The behavioral, or dynamic, view describes the behavior of the system in terms of messages passed among objects and state changes within an object.

Unified Process



- Source : [Pressman]

Unified Process

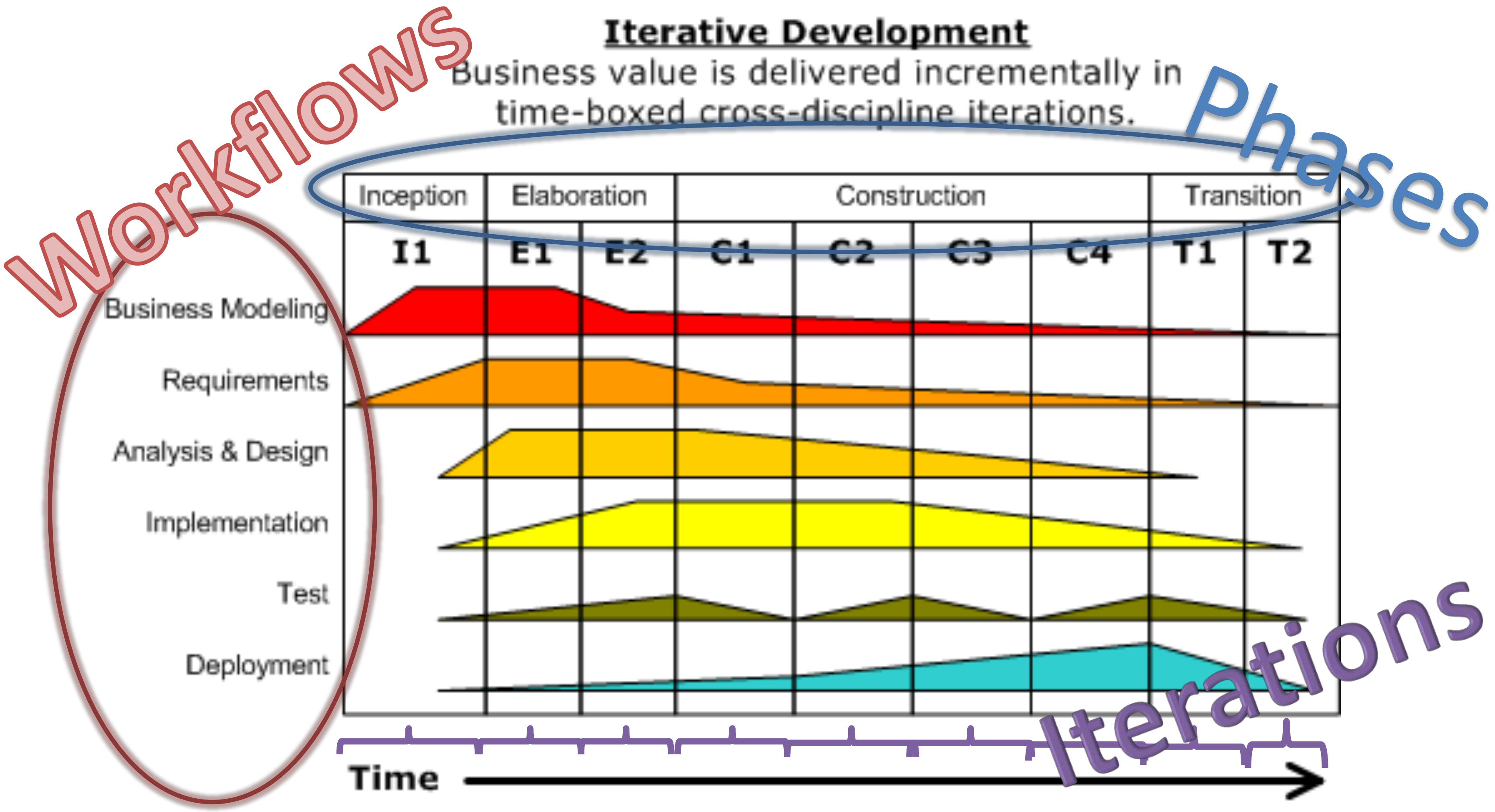
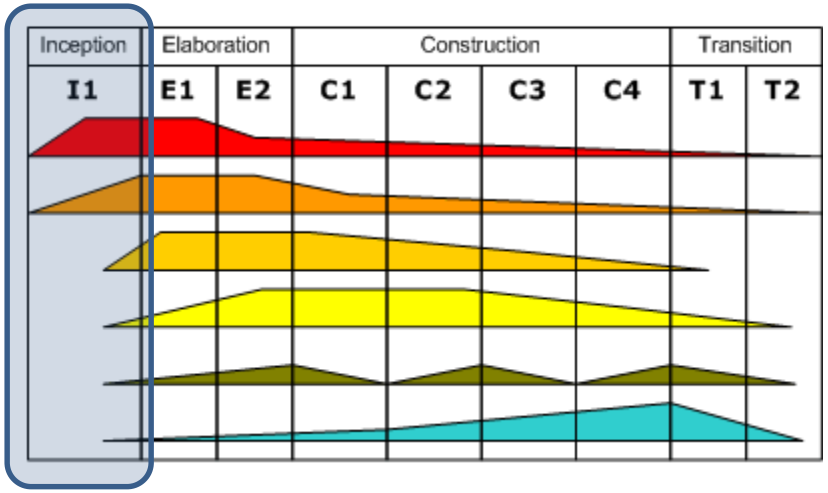


Image source : https://upload.wikimedia.org/wikipedia/commons/1/19/Development-iterative.png (public domain image)

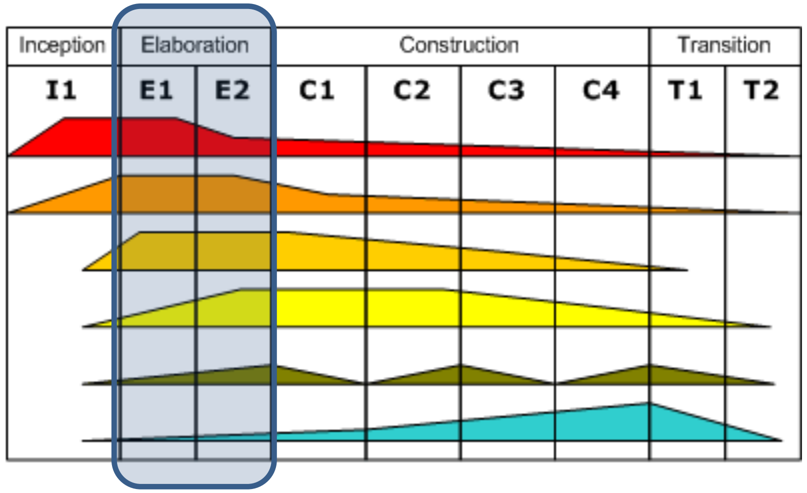
Unified Process : Inception Phase

- Goal : establish a business case for the system
- Communication activity
  - ❖Identify requirements as preliminary use-cases
  - ❖Outline basic architecture of the system
- Planning activity
  - ❖Plan for iterative, incremental project

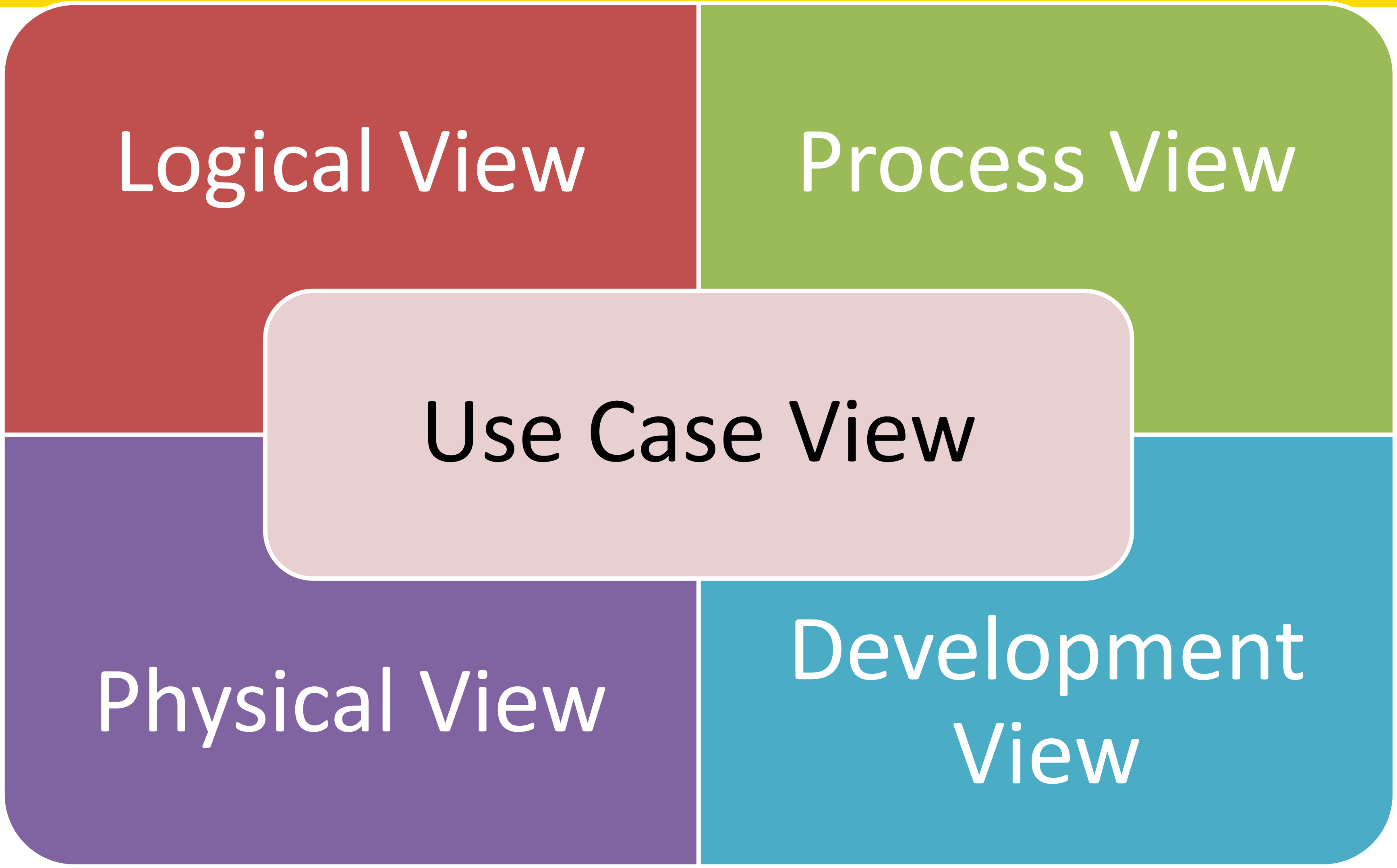


Unified Process : Elaboration Phase

- Goal : understand the problem domain, establish an architectural framework, develop the project plan, and identify key project risks
- Planning activity
  - ❖ Plans are carefully reviewed to ensure scope, risks, and delivery dates remain reasonable
- Modelling activity
  - ❖ Refine and expand preliminary use-cases and architectural representation
    - 5 different views of the system
    - Executable architectural baseline ("first cut" executable system)

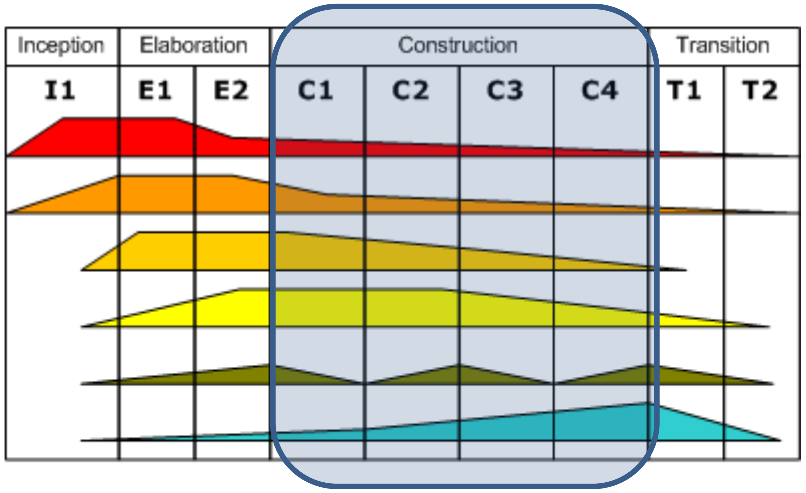


Five Different Views of the System



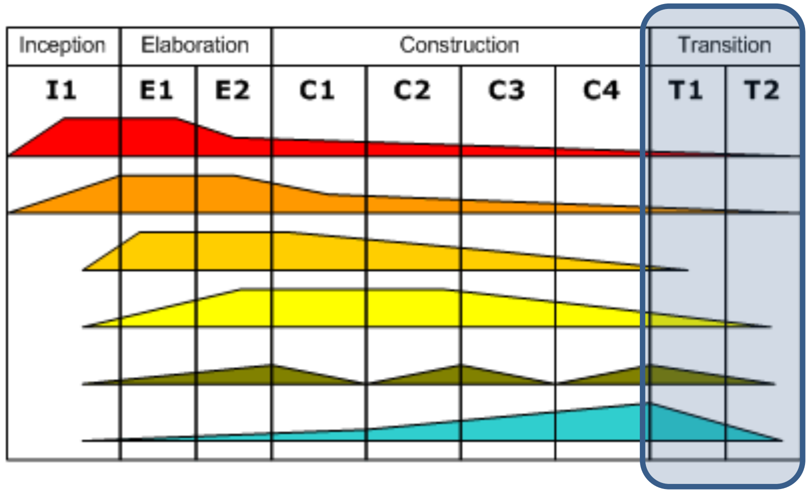
Unified Process : Construction Phase

- Goal : develop a working software system and associated documentations
- Construction activity
  - ❖ Code generation
    - Complete the requirement and design models
    - Implement all necessary and required features and functions for the software increment
  - ❖ Testing
    - Design and execute unit tests
    - Conduct integration activities
    - Derive and execute a suite of acceptance tests from use cases



Unified Process : Transition Phase

- Goal : have a documented software system that is running correctly in its operational environment
- Construction activity
  - ❖ Beta testing
- Deployment activity
  - ❖ Create necessary support information (e.g. user manuals, troubleshooting guides, installation procedures)

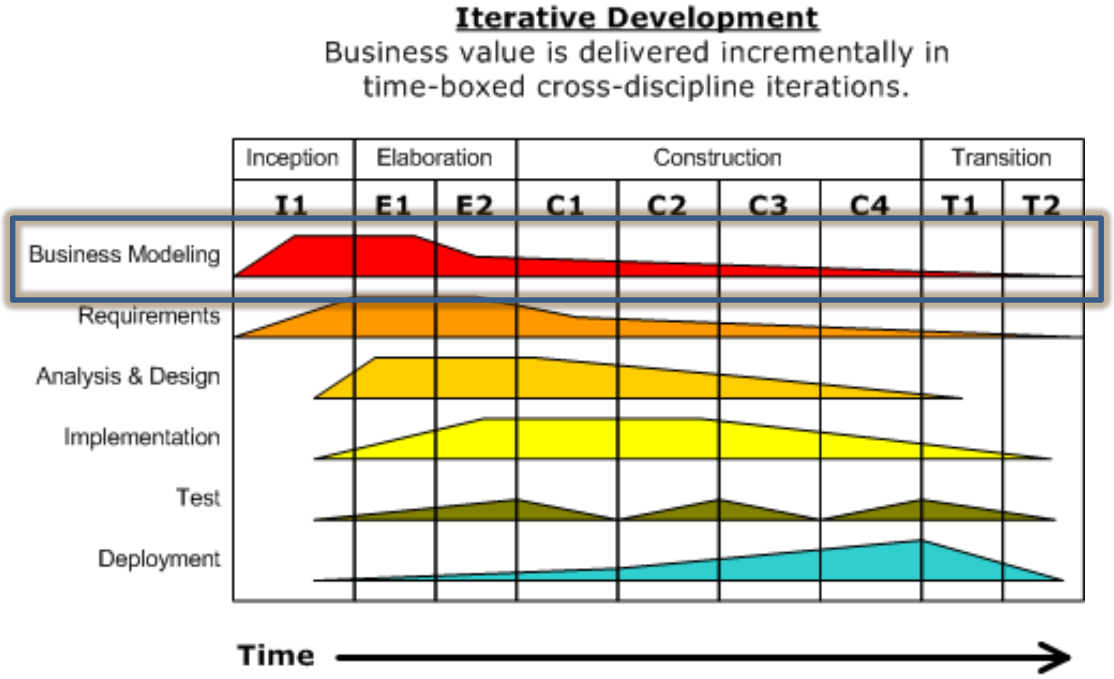


Unified Process : Workflows

- 6 core process workflows
  - ❖Business modelling
  - ❖Requirements
  - ❖Analysis & design
  - ❖Implementation
  - ❖Testing
  - ❖Deployment
- 3 core supporting workflows
  - ❖Configuration & change management
  - ❖Project management
  - ❖Environment

Unified Process : Workflows

- Note that activities (workflows) in Unified Process do not occur in a sequence
- Work efforts on workflows may change across phases



Unified Process : Work Product

Inception phase

Vision document
Initial use-case model
Initial project glossary
Initial business case
Initial risk assessment.
Project plan,
  phases and iterations.
Business model,
  if necessary.
One or more prototypes

Elaboration phase

Use-case model
Supplementary requirements
  including non-functional
Analysis model
Software architecture
  Description.
Executable architectural
  prototype.
Preliminary design model
Revised risk list
Project plan including
  iteration plan
  adapted workflows
  milestones
  technical work products
Preliminary user manual

Construction phase

Design model
Software components
Integrated software
  increment
Test plan and procedure
Test cases
Support documentation
  user manuals
  installation manuals
  description of current
    increment

Transition phase

Delivered software increment
Beta test reports
General user feedback

Unified Process : Best Practices

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes in software

Q & A