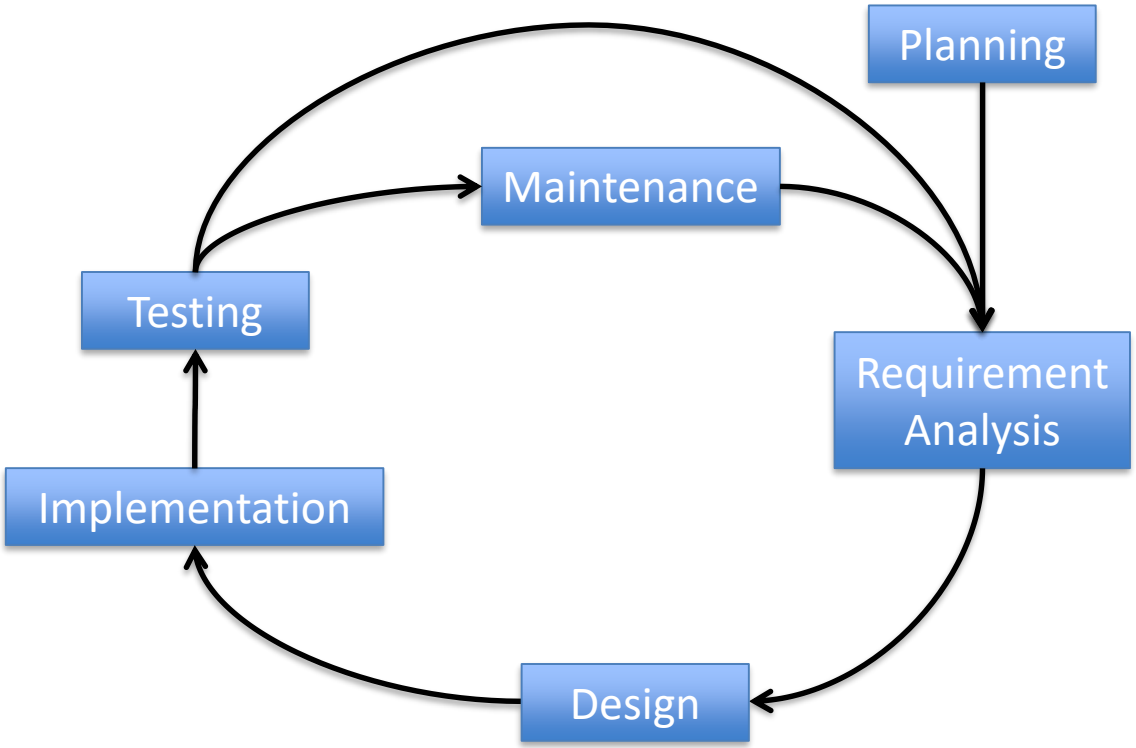
Software Development Process

Lecture 2

Sequential and Iterative and Incremental Models

Typical Software Project Phases

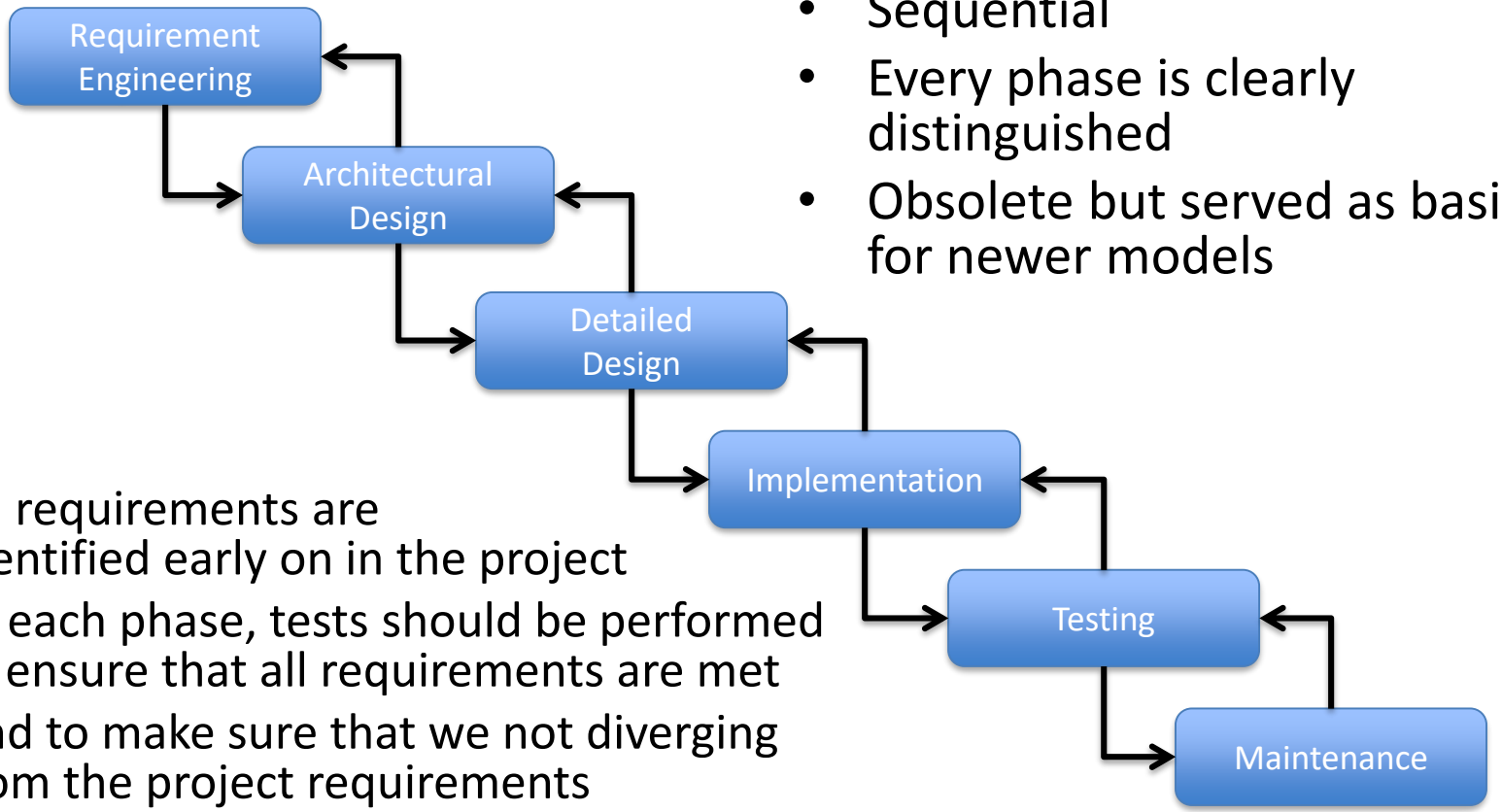


- How should we schedule these phases?

Software Development Process Goals

- The whole development process must be controlled
  - A software project is usually large and a lot of people are involved
  - Development time could also be very long
- Software process acts as a guideline to control the software development activities
- Example of different models:
  - Sequential Models
  - Iterative and Incremental Models
  - Agile Processes
  - Open Source Process

Waterfall Model

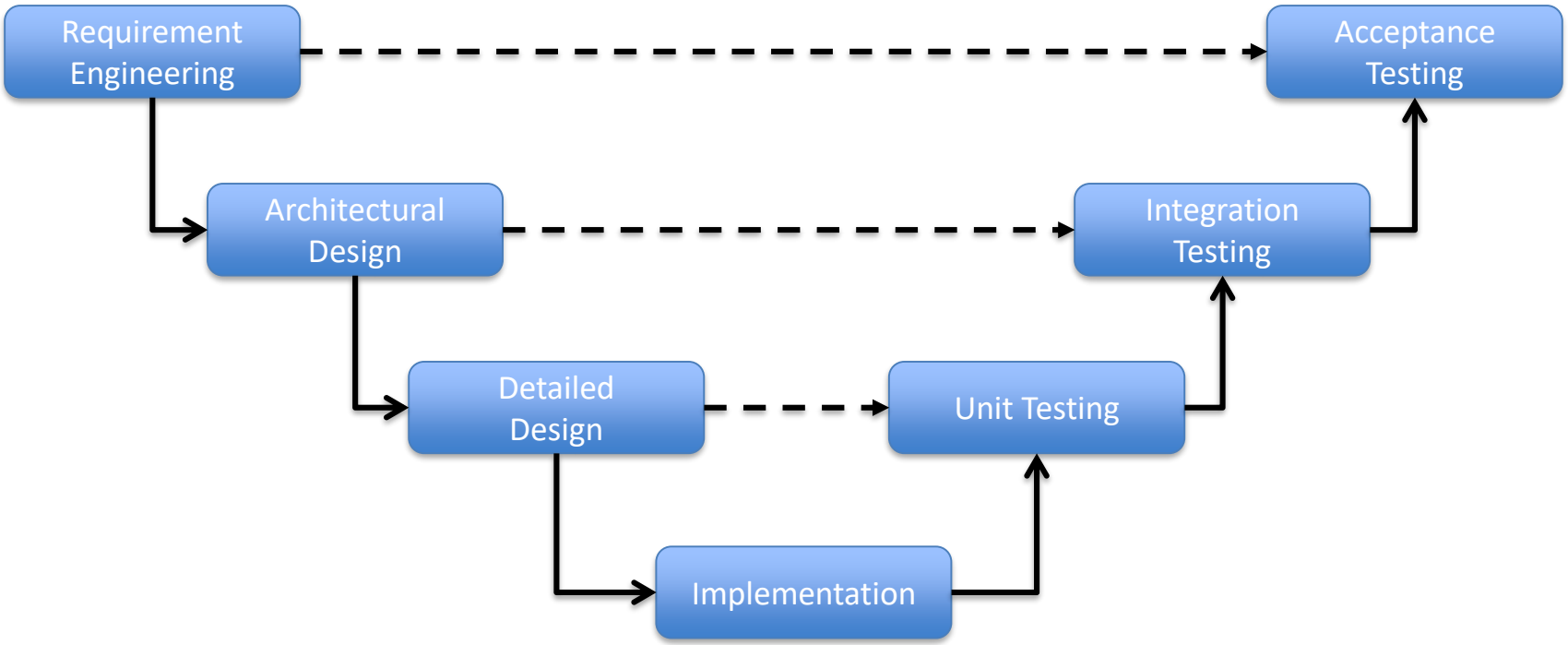


- Sequential
- Every phase is clearly distinguished
- Obsolete but served as basis for newer models

- All requirements are identified early on in the project
- At each phase, tests should be performed to ensure that all requirements are met
- And to make sure that we not diverging from the project requirements
- We might have to move back to previous phase

V-Model

- Sequential
- V-Model shows how a software product is validated
- It relates different kinds of testing to corresponding design phases
- Test plans are developed after each phase on the left is done



Advantages of Sequential Models

- Simple and easy to use: Phases are well-defined and executed sequentially
- Easy to manage: The model is rigid. Each phase has specific deliverables and review process
- Facilitates allocation of resources: Different phases require different personnel with different skills
- Works well for project with requirements that are well-understood
  - Short and clear projects are ideal

Disadvantages of Sequential Model

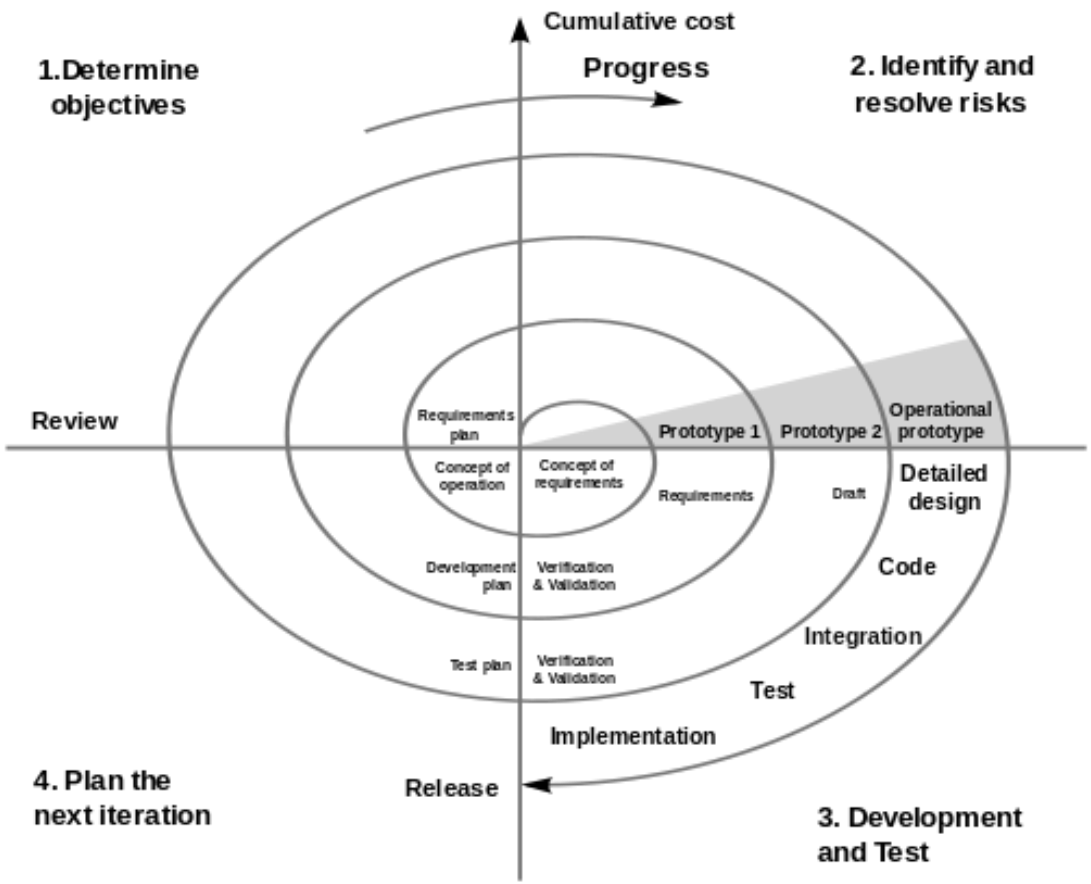
- Requirements must be known beforehand: Does not work with projects with hazy knowledge
- No feedback from stakeholders until testing phases
- Problems with projects might not be discovered until testing phases
- Lack of parallelism: Second phase cannot be executed along with the first phase
- Inefficient use of resource: Due to lack of parallelism, team members (e.g., developers) must wait until other teams (e.g., designers) finish their work

Iterative and Incremental Development

- Escher's waterfall
- **Iterative**: Repeating the same (waterfall) process over and over again
- **Incremental**: Adding pieces of the system at different times and rates

Image: http://en.wikipedia.org/wiki/Waterfall_(M._C._Escher)

Spiral Model



- Proposed by Boehm in 1988
- It is a risk-driven approach combined with waterfall model
- Risk management is included in each iteration
- Number of iterations (cycles) depends on the size of the project

Image: http://en.wikipedia.org/wiki/Spiral_model

Spiral Model Activities

1. Determine objectives
   - Requirement elicitations
   - Feasibility studies
2. Identify and resolve risks
   - Risk analysis (cost overruns, wrong calculations, etc.)
   - Evaluate alternatives
   - Planning risk mitigation strategy
   - Develop series of prototypes to identify risks
   - Use a waterfall model for each prototype development
   - Customer can abort the project if the risks are too great

Spiral Model Activities (2)

3. Development and test
   - Implementation
   - Conduct testing
4. Evaluation
   - Customer evaluates the product

Prototyping

- Risk-management technique
- A partial implementation of the target product
- Can be used for:
  - Identifying *risky* parts of the project
  - Determine the idea about customer's requirements
  - Gather look-and-feel in GUI
- However, prototypes could also be expensive and complex
- We shall build a prototype if the development cost is low and the yielded value is high

Prototypes Types

- Illustrative Prototype
  - Develop the user interface with a set of storyboards
  - Implement them on a napkin or with a user interface builder
  - Good for (early) client discussion
- Functional Prototype
  - Implement and deliver an operational system with minimum functionality
  - Then add more functionality
- Exploratory Prototype ("*Hack*")
  - Implement part of the system to learn more about the requirements.
  - Good for paradigm breaks.

(Dis)Advantages of Spiral Model

- Advantages:
  - Fast development
  - Risks are managed throughout the process
  - Software evolves as the project progress
    - Good for large-scale project
  - Planning is integrated into the process
    - Planning is included in each cycle, keeping the process on track
- Disadvantages:
  - Risk analysis requires an expert
  - Risk management is expensive and may not be necessary for small projects
  - Cannot handle ***changes*** very well
  - Tedious documentation

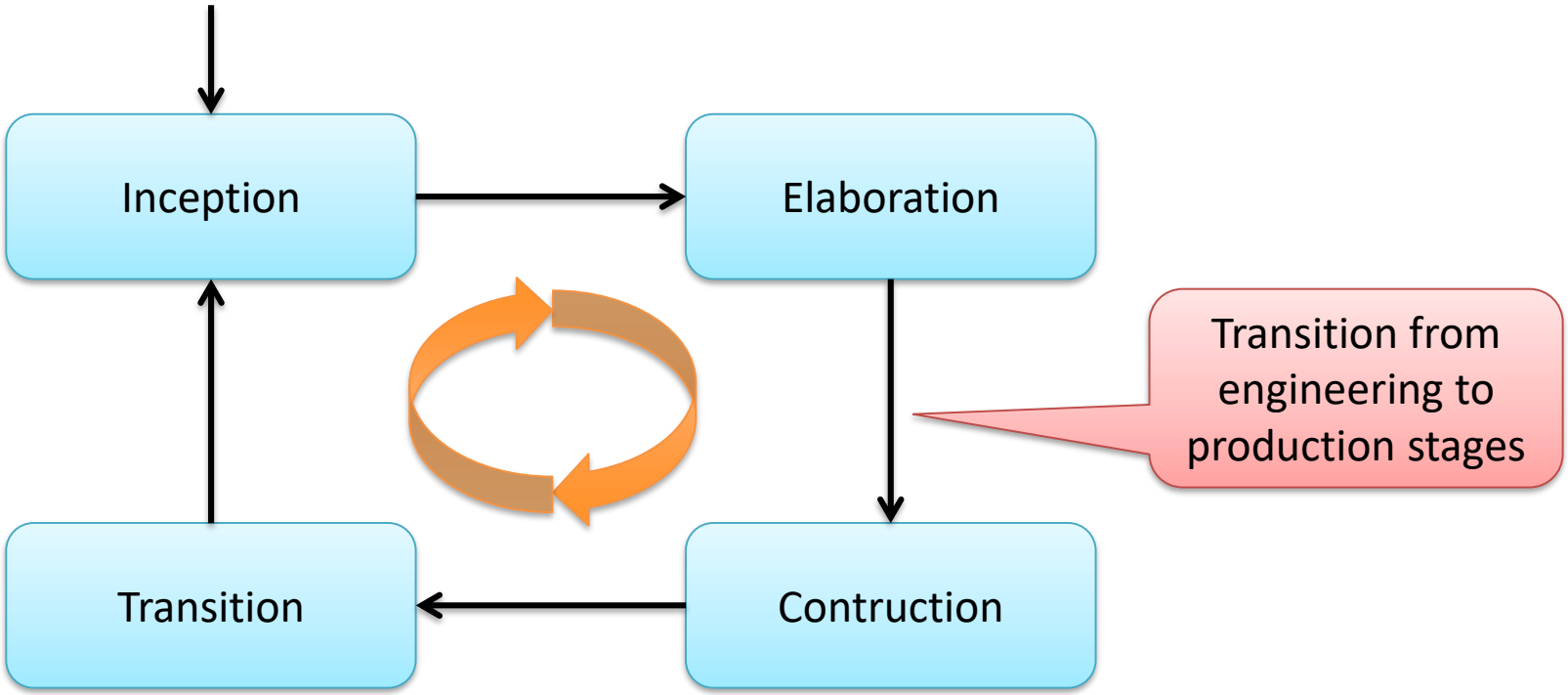
Unified Process (UP)

- Full name: **Unified Software Development Process** (**USDP**)
- Developed by Booch, Jacobson, and Rumbaugh in 1999
- Aimed to be extensible framework that can be customized to fit different projects
- IBM Rational Software division refined the process and commercialize it as **Rational Unified Process (RUP)**
  - There are some other refinements too
- UP is use-case driven and iterative and incremental
  - It uses use cases as basis for all development processes
  - Each **iteration** implements some use cases and scenarios

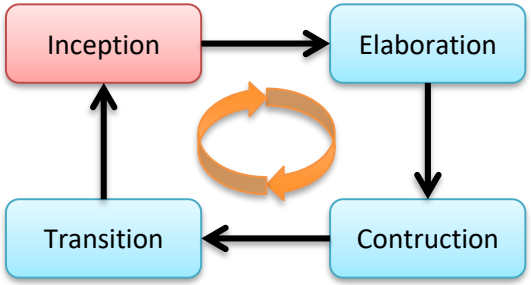
UP's Engineering and Production

- UP divides the development process into
  - **Engineering Stage**: Driven by less predictable but smaller teams, focusing on design and synthesis activities
    - **Inception phase**
    - **Elaboration phase**
  - **Production Stage**: Driven by more predictable but larger teams, focusing on construction, test and deployment activities
    - **Construction phase**
    - **Transition phase**

UP's Software Development Phases

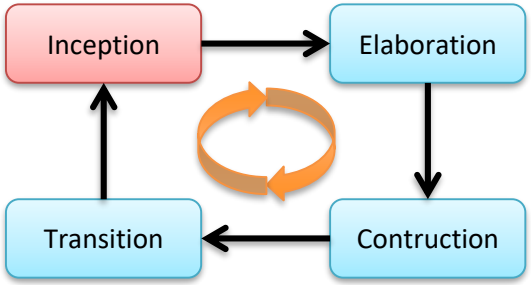


Inception Phase: Objectives



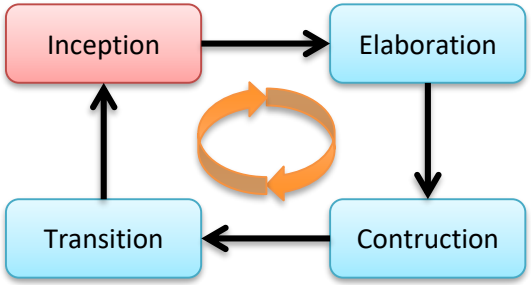
- **Establish** the project ***scope***
- **Identify** the critical ***use cases*** and ***scenarios***
- **Define** acceptance ***criteria***
- **Demonstrate** at least one candidate software ***architecture***
- **Estimate** the ***cost*** and ***schedule*** for the project
- **Define** and ***estimate*** potential ***risks***

Inception Phase: Activities



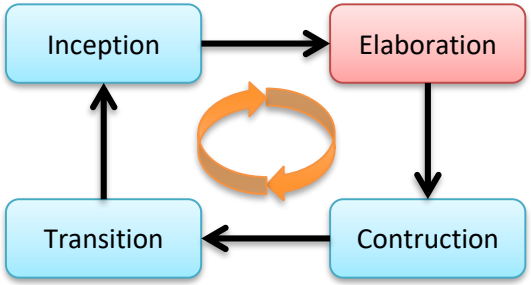
- **Formulate** the ***scope*** of the project
  - Capture requirements
  - Result: problem space and acceptance criteria are defined
- **Design** the software ***architecture***
  - Evaluate design trade-offs, investigate solution space
  - Result: Feasibility of at least one candidate architecture is explored, initial set of build vs. buy decisions
- **Plan** and prepare a ***business case***
  - Evaluate alternatives for risks, staffing problems, plans.

Inception Phase: Evaluation



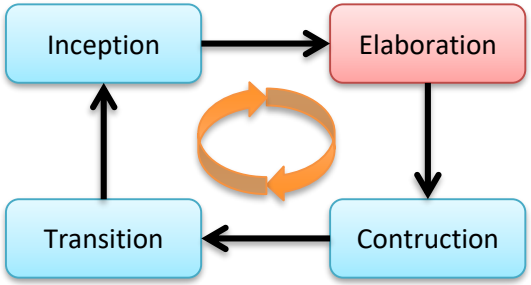
- Do all **stakeholders concur** on the scope definition and cost and schedule estimates?
- Are the **requirements understood**, are the critical use cases adequately modeled?
- Is the software **architecture understood**?
- Are cost, schedule **estimates, priorities, risks** and development processes **credible**?
- Is there a **prototype** that helps in evaluating the criteria?

Elaboration Phase: Objectives



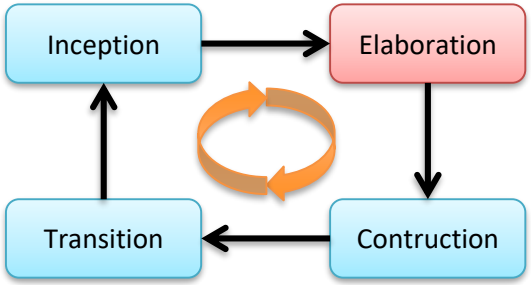
- **Baseline** the **software architecture**
  - Establish a configuration management plan in which all changes are tracked and maintained
- **Baseline** the **problem statement**
- **Baseline** the **software project management plan** for the construction phase
- **Demonstrate** that the **architecture** supports the requirements at a reasonable cost in a reasonable time
- **Baseline**: An agreed-to description of the attributes of a product, at a point in time, which serves as a basis for defining change

Elaboration Phase: Activities



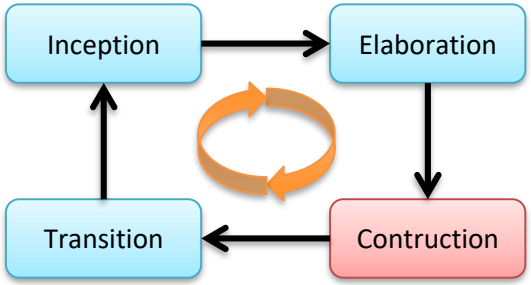
- **Elaborate** the ***problem statement*** (***vision***) by working out the critical use cases that drive technical and managerial decisions
- **Elaborate** the ***infrastructure***
- **Tailor** the ***software process*** for the construction stage, identify tools
- **Establish** intermediate ***milestones*** and evaluation ***criteria*** for these milestones
- **Identify** buy/build ("make/buy") problems and make decisions
- **Identify** ***lessons learned*** from the inception phase to redesign the software architecture if necessary
  - It is always necessary

Elaboration Phase: Evaluation



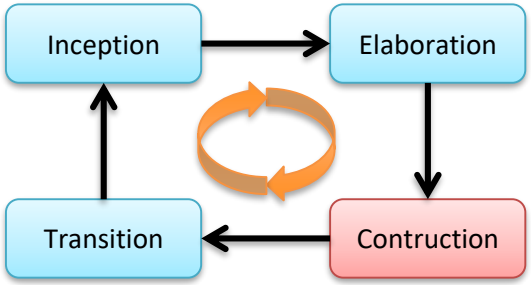
- Is the **problem statement stable**?
- Is the **architecture stable**?
- Does the executable demonstration show that the major **risk** elements have been **addressed** and credibly resolved?
- Is the construction **plan credible**? By what claims is it backed up?
- Do all **stakeholders** (project participants) **agree** that the vision expressed in the problem can be met if the current plan is executed?
- Are actual **resource expenditures** versus planned expenditures so far **acceptable**?

Construction Phase: Objectives



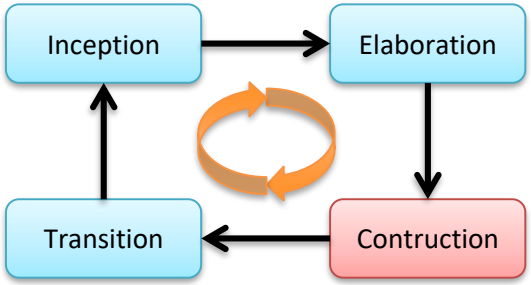
- Minimize development costs by optimizing resources
- Achieve adequate quality as rapidly as practical
- Achieve useful version (alpha, beta, and other test releases) as soon as possible

Construction Phase: Activities



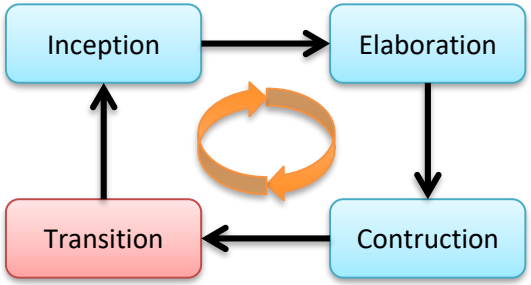
- Resource management, control and process optimization
- Complete component development and testing against evaluation criteria
- Assessment of product releases against acceptance criteria

Construction Phase: Evaluation



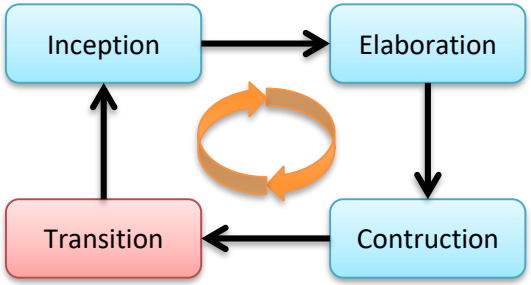
- Is the **product** baseline **mature** enough to be deployed in the user community?
  - Existing faults are not obstacles to do the release
- Is the **product** baseline **stable** enough to be deployed in the user community?
  - Pending changes are not obstacles to do the release
- Are the **stakeholders ready** for the transition of the software system to the user community?
- Are actual resource **expenditures** versus planned expenditures so far **acceptable**?

Transition Phase



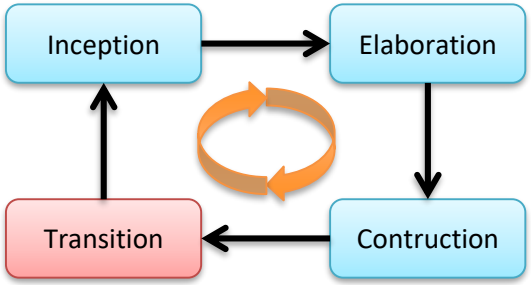
- The transition phase is entered when
  - the system has been built with acceptable quality levels and documentation
  - the system can be deployed to the user community
- For some projects the transition phase means the starting point for another version of the software system
  - Back to Inception
- For other projects the transition phase means the complete delivery of the software system

Transition Phase: Objectives



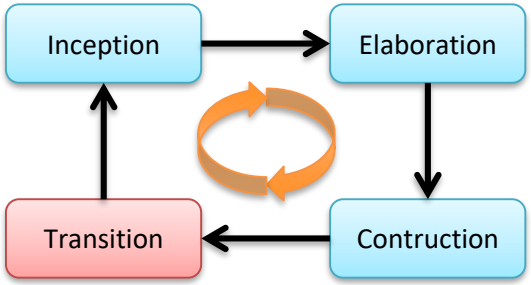
- Achieve **independence of user** so that the users can support themselves
- **Deployment baseline** is complete and consistent with the criteria in the project agreement
- The **final baseline** can be built as rapidly and cost-effectively as possible.

Transition Phase: Activities



- Synchronization and **integration** of concurrent development increments into one consistent deployment baseline
- Commercial **packaging** and **production**
- Sales **rollout kit** development
- Field personnel **training**
- **Test** of deployment baseline against the acceptance criteria.

Transition Phase: Evaluation



- Is the user **satisfied**?
- Are actual resource **expenditures** versus planned expenditures so far **acceptable**?

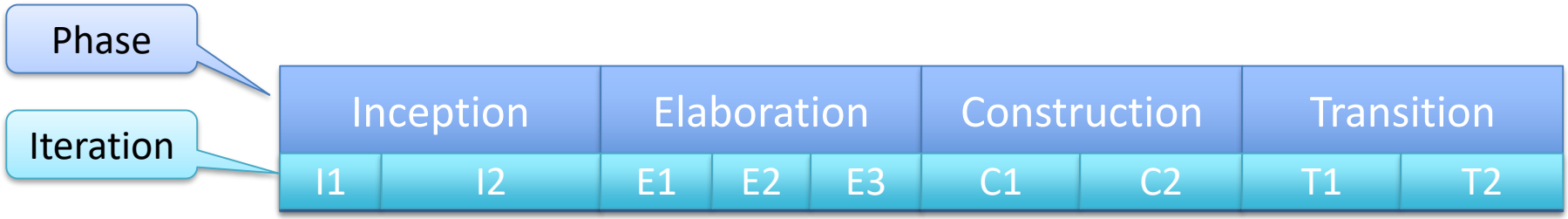
Unified Process Iterations

- Each of the four phases (inception, elaboration, construction, transition) consists of one or more **iterations**
- An iteration represents:
  - A set of **milestone** activities
  - A well-defined intermediate event
- The scope and results of each iteration are captured via work products (or **artifacts**)

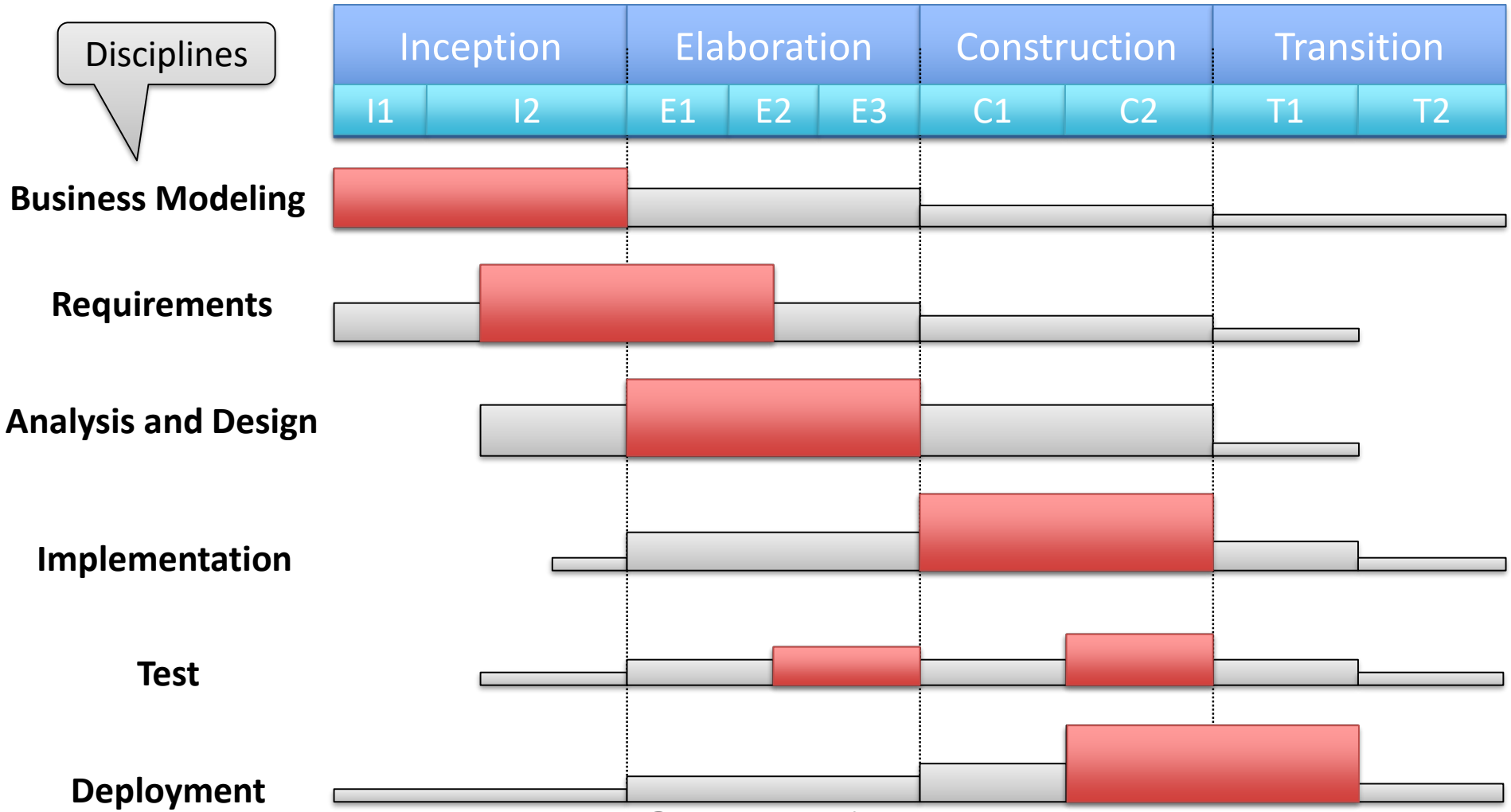
UP's Phase vs. Iteration

- A phase creates a formal, stake-holder approved version of artifacts
  - It leads to a **major milestone**
  - Phase to phase transition: Triggered by a significant business decision (not by the completion of a software development activity)
- An iteration creates an informal, internally controlled version of artifacts
  - It leads to a **minor milestone**
  - Iteration to iteration transition: Triggered by a specific software development activity

Each Phase has One or More Iterations



Each Iteration Cycles Through Disciplines (Workflows)



Artifact and Artifact Set

- **Artifact**: A work product in a uniform representation format (natural language, UML, Java, binary code,…)
- **Artifact set**: A set of artifacts developed and reviewed as a single entity
- The Unified Process distinguishes five artifact sets
  - Management set
  - Requirements set
  - Design set
  - Implementation set
  - Deployment set

Requirements set, Design set, Implementation set, Deployment set: Also called Engineering set

Artifact Sets in the Unified Process

Engineering Set

Requirements Set	Design Set	Implementation Set	Deployment Set
1. Vision document 2. Requirements model(s)	1. Design model(s) 2. Test model 3. Software architecture	1. Source code baselines 2. Compile-time files 3. Component executables	1. Integrated product executable 2. Run-time files 3. User documentation

Management Set

Planning Artifacts

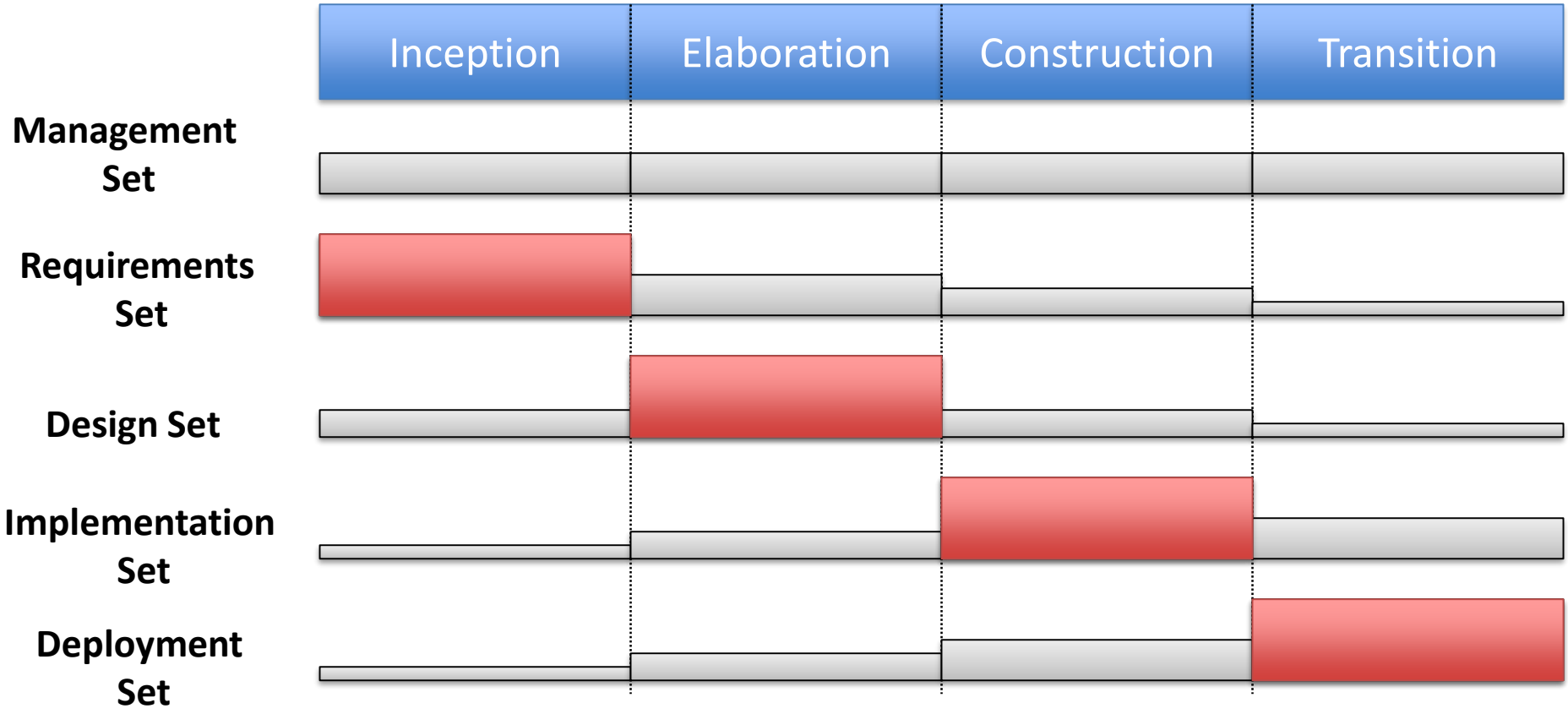
1 Software Project Management Plan (SPMP)
2. Software Configuration Management Plan (SCMP)
3. Work breakdown structure
4. Business Case
5. Release specifications

Operational Artifacts

1. Release descriptions
2. Status assessments
3. Change Management database
4. Deployment documents
5. Environment.

Software Life-Cycle and Artifact Sets

- Each artifact set is the predominant focus in one stage of the unified process



(Dis)Advantages of Unified Process

- Advantages:
  - UP is inclusive: Most of software development works are included in the framework
    - Business models and project management
    - Development and deployment
  - It is mature and widely used
- Disadvantage:
  - Not suitable for small projects: There are too many works to do
  - Customizing UP to fit a project requires UP expert
  - Going through all workflows in each iteration requires both time and resources
- **Note**: UP is actually flexible and these disadvantages can be avoided by adapting UP to your working environment