Software Lifecycles Models

Software Engineering
Lecture 17

Bernd Bruegge

Applied Software Engineering

Technische Universitaet Muenchen

Outline of Today's Lecture

- Modeling the software life cycle
- Sequential models
  - Pure waterfall model
  - V-model
  - Sawtooth model
- Iterative models
  - Boehm's spiral model
  - Unified Process
- Entity-oriented models
  - Issue-based model

Typical Software Life Cycle Questions

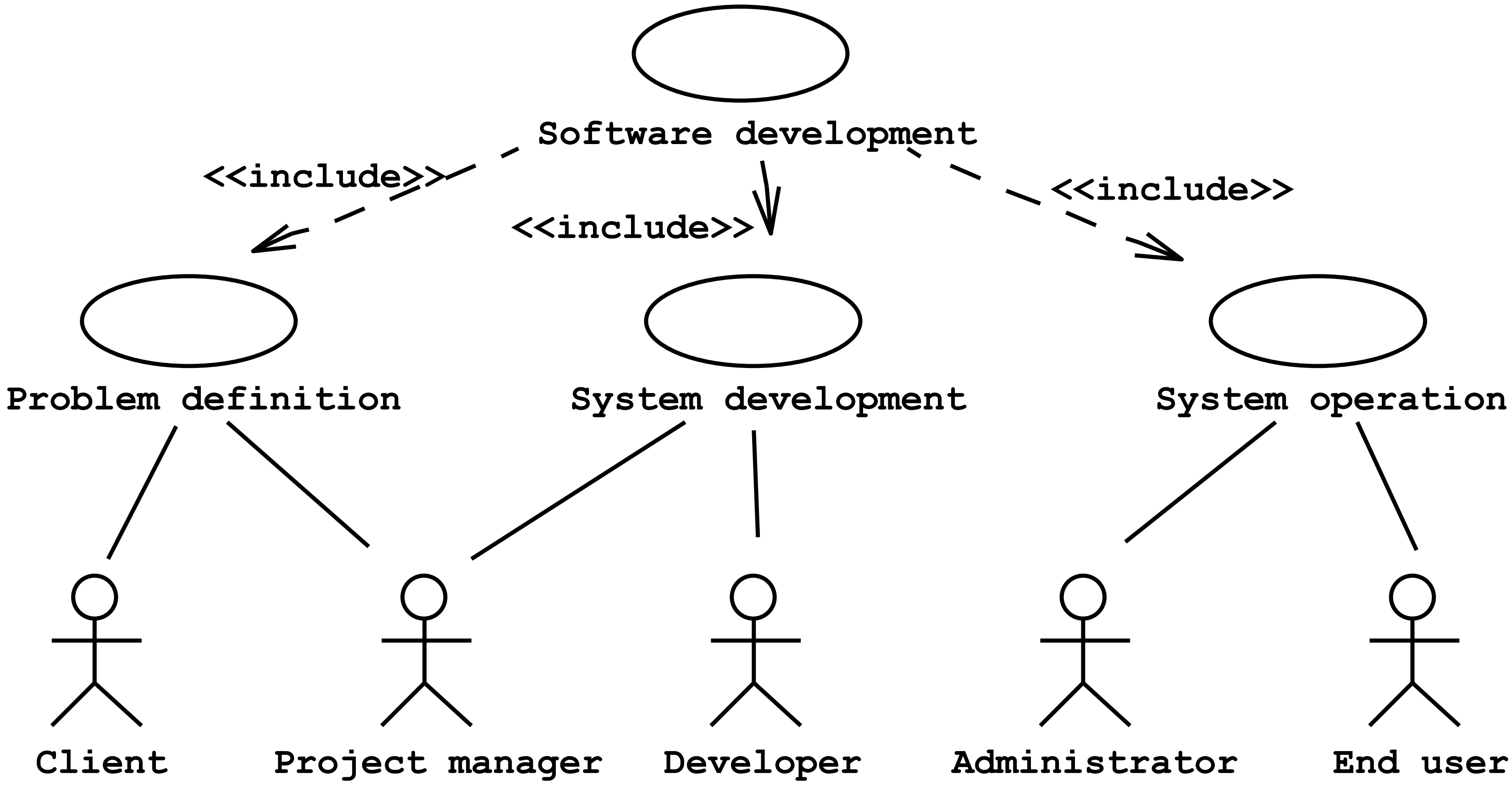
- *Which activities* should we select for the software project?
- What are the *dependencies between activities*?
- How should we *schedule the activities*?
- To find these activities and dependencies we can use the same modeling techniques we use for software development:
  - Functional Modeling of a Software Lifecycle
    - Scenarios
    - Use case model
  - Structural modeling of a Software Lifecycle
    - Object identification
    - Class diagrams
  - Dynamic Modeling of a Software Lifecycle
    - Sequence diagrams, statechart and activity diagrams

Definitions

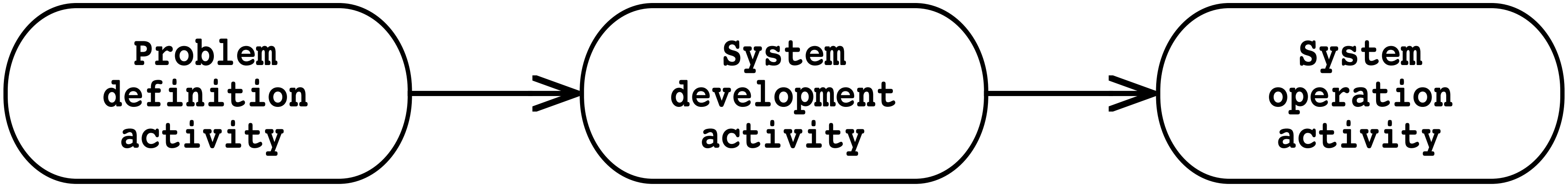
- Software life cycle:
  - Set of activities and their relationships to each other to support the development of a software system ▷

- Software development methodology:
  - A collection of techniques for building models applied across the software life cycle

Functional Model of a simple life cycle model

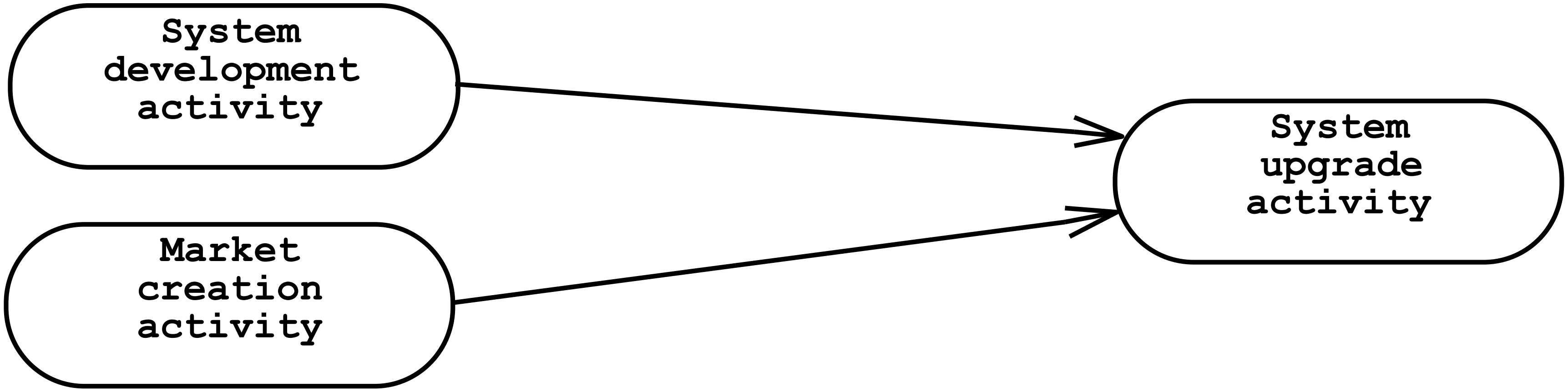


Activity Diagram for the same Life Cycle Model


Problem definition activity
System development activity
System operation activity

Software development goes through a linear progression of states called software development activities

Another simple Life Cycle Model

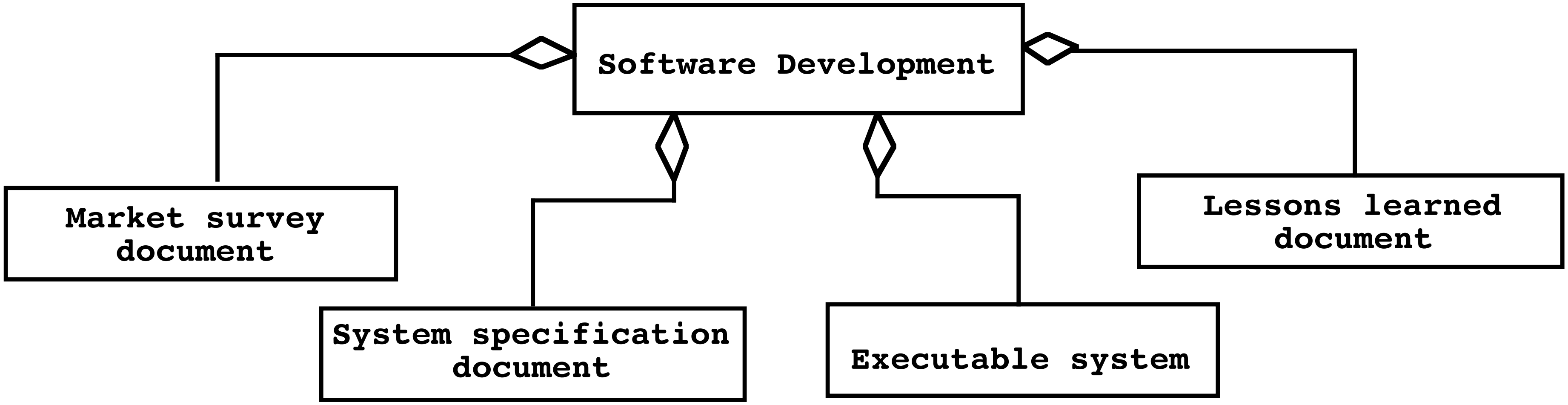


System Development and Market creation can be done in parallel. They must be done before the system upgrade activity

Two Major Views of the Software Life Cycle

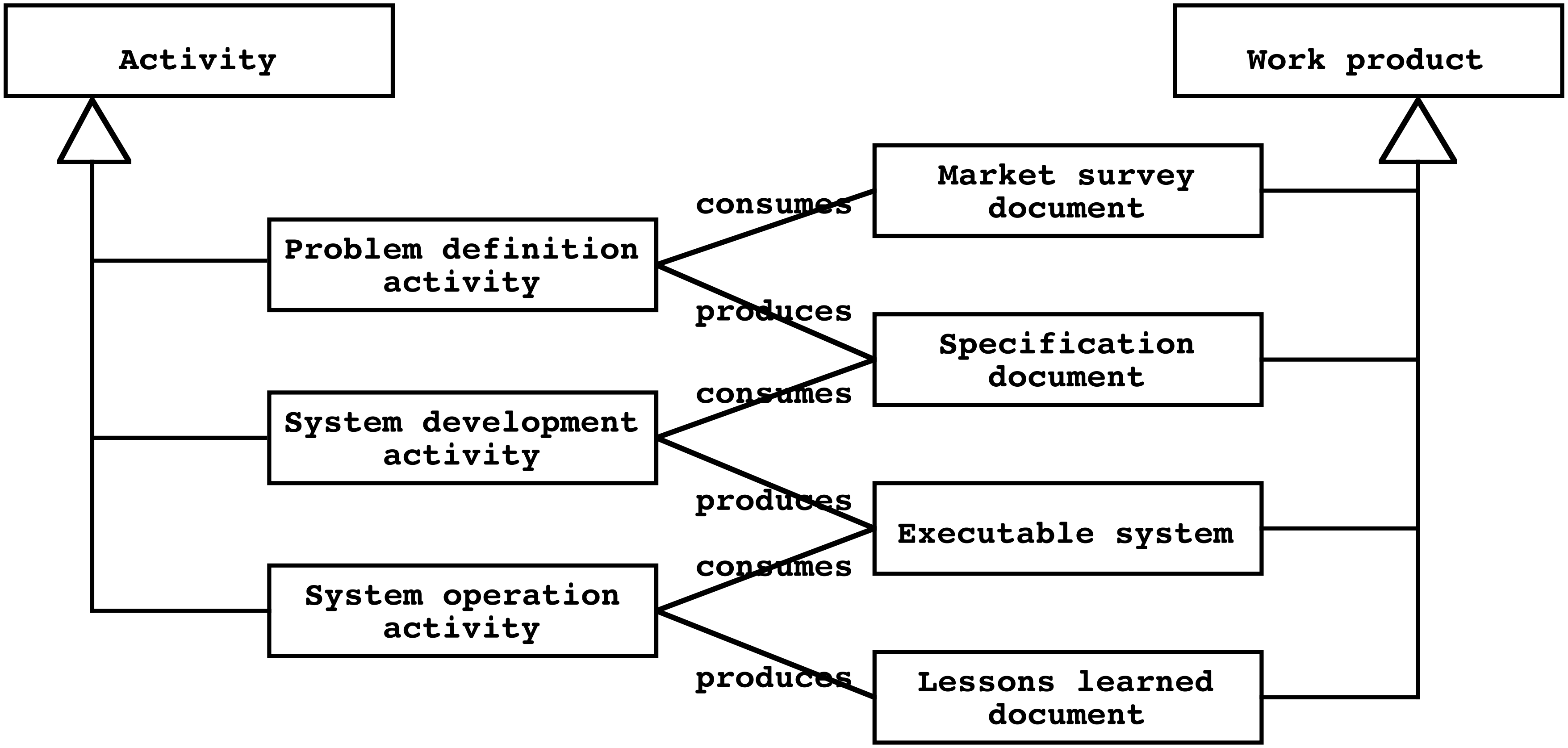
- Activity-oriented view of a software life cycle
    - Software development consists of a set of development activities
    - all the examples so far
- Entity-oriented view of a software life cycle
    - Software development consists of the creation of a set of deliverables.

Entity-centered view of Software Development

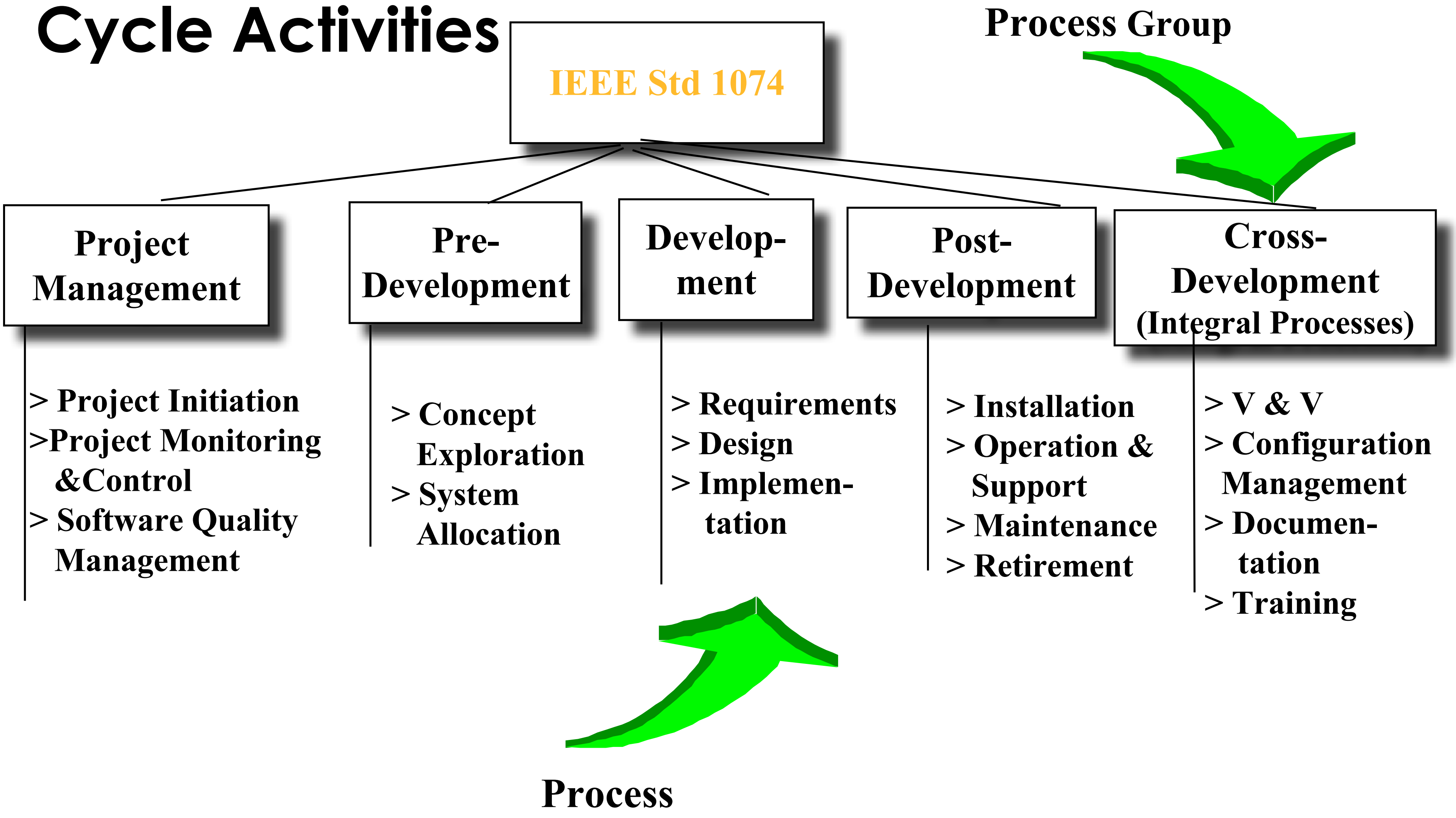


Software development consists of the creation of a set of deliverables

Combining Activities and Entities in One View



IEEE Std 1074: Standard for Software Life Cycle Activities



Process Group

IEEE Std 1074

Project Management
- > Project Initiation
- >Project Monitoring &Control
- > Software Quality Management

Pre-Development
- > Concept Exploration
- > System Allocation

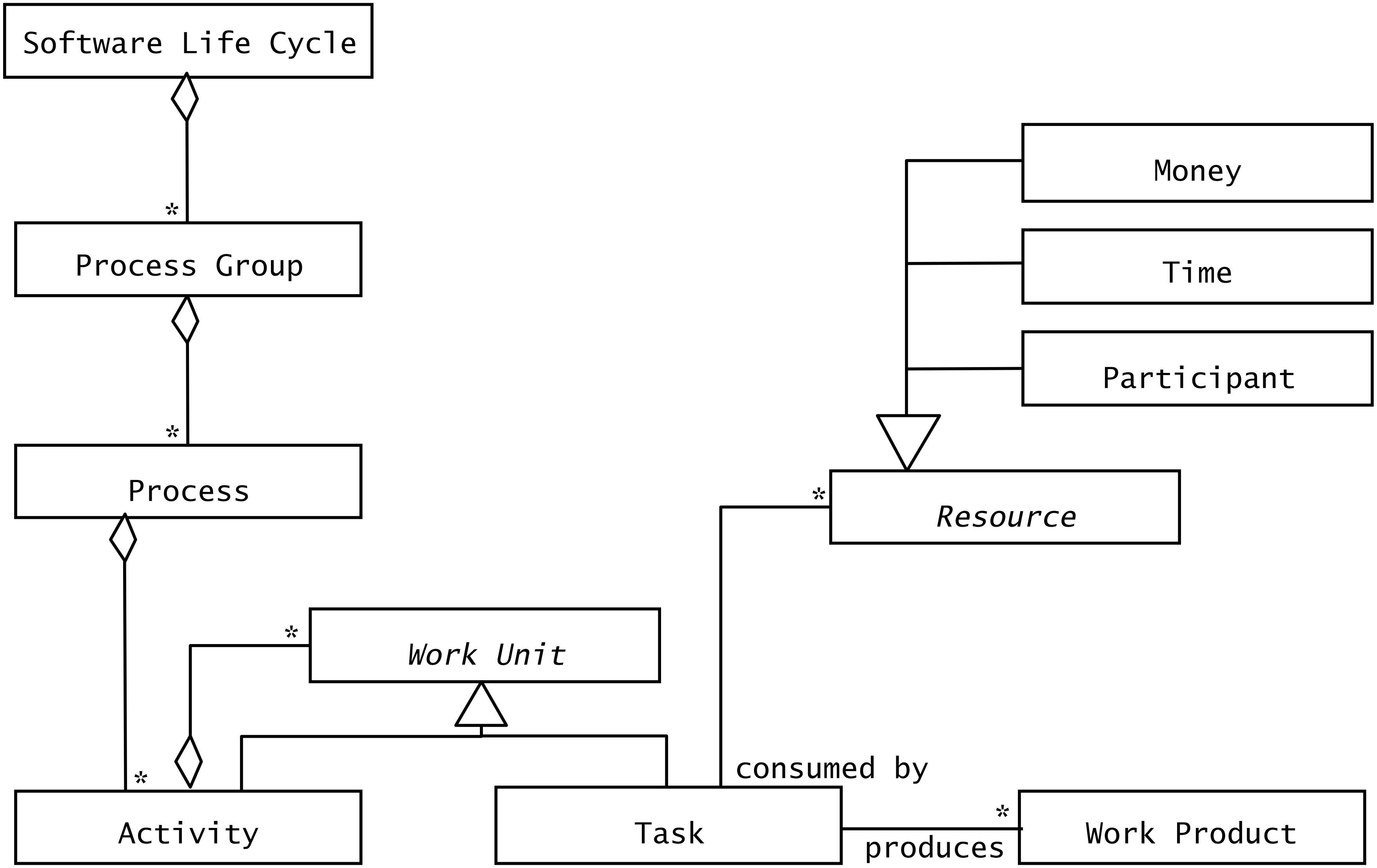
Develop-ment
- > Requirements
- > Design
- > Implemen-tation

Post-Development
- > Installation
- > Operation & Support
- > Maintenance
- > Retirement

Cross-Development (Integral Processes)
- > V & V
- > Configuration Management
- > Documen-tation
- > Training

Process

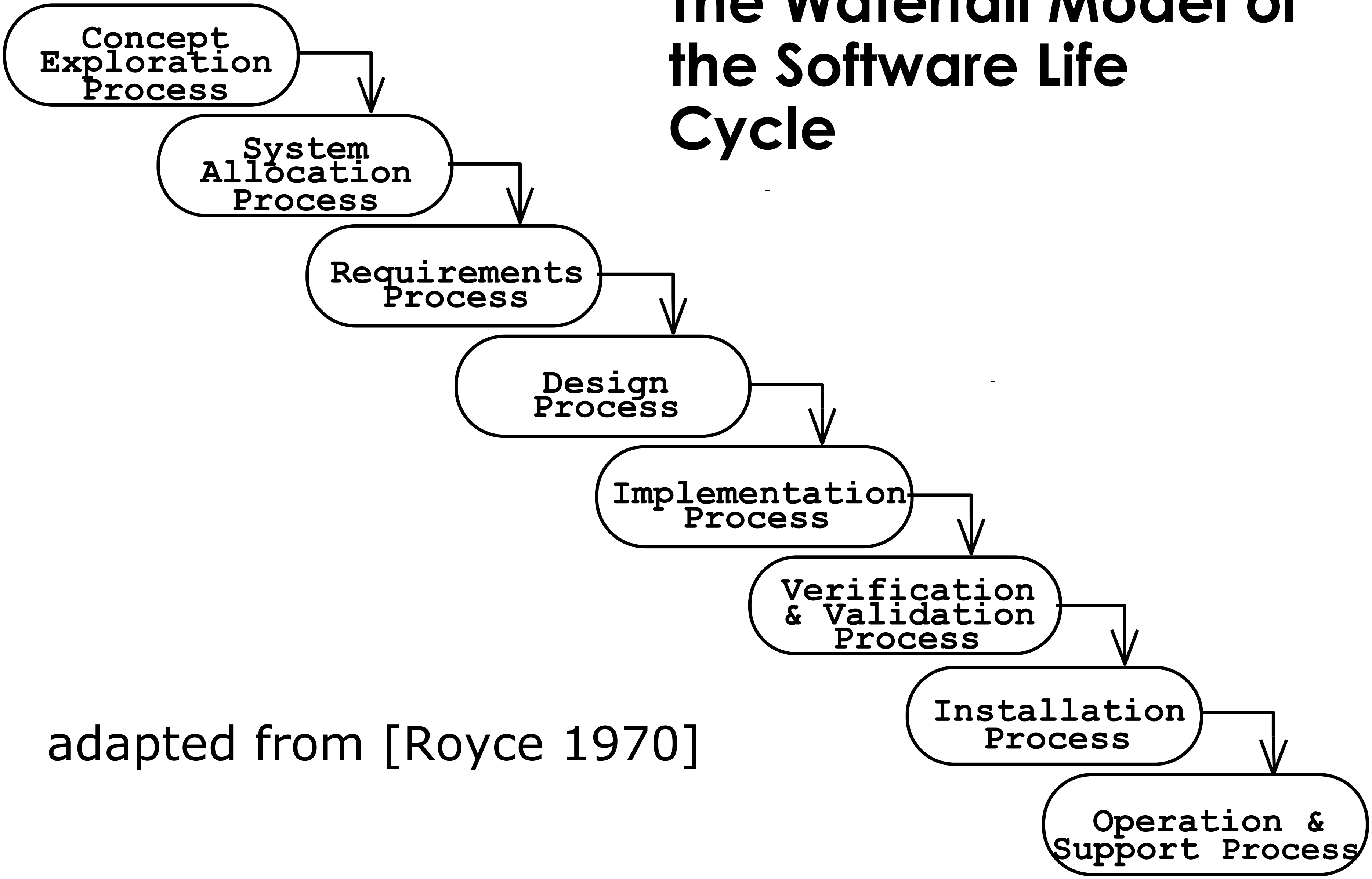
Object Model of the IEEE 1074 Standard



Life Cycle Modeling

- Many models have been proposed to deal with the problems of defining activities and associating them with each other
  - The first model proposed was the waterfall model [Royce]
  - Spiral model [Boehm]
  - Objectory process [Jacobsen]
  - Rational process [Kruchten]
  - Unified process [Jacobsen, Booch, Rumbaugh]

The Waterfall Model of the Software Life Cycle

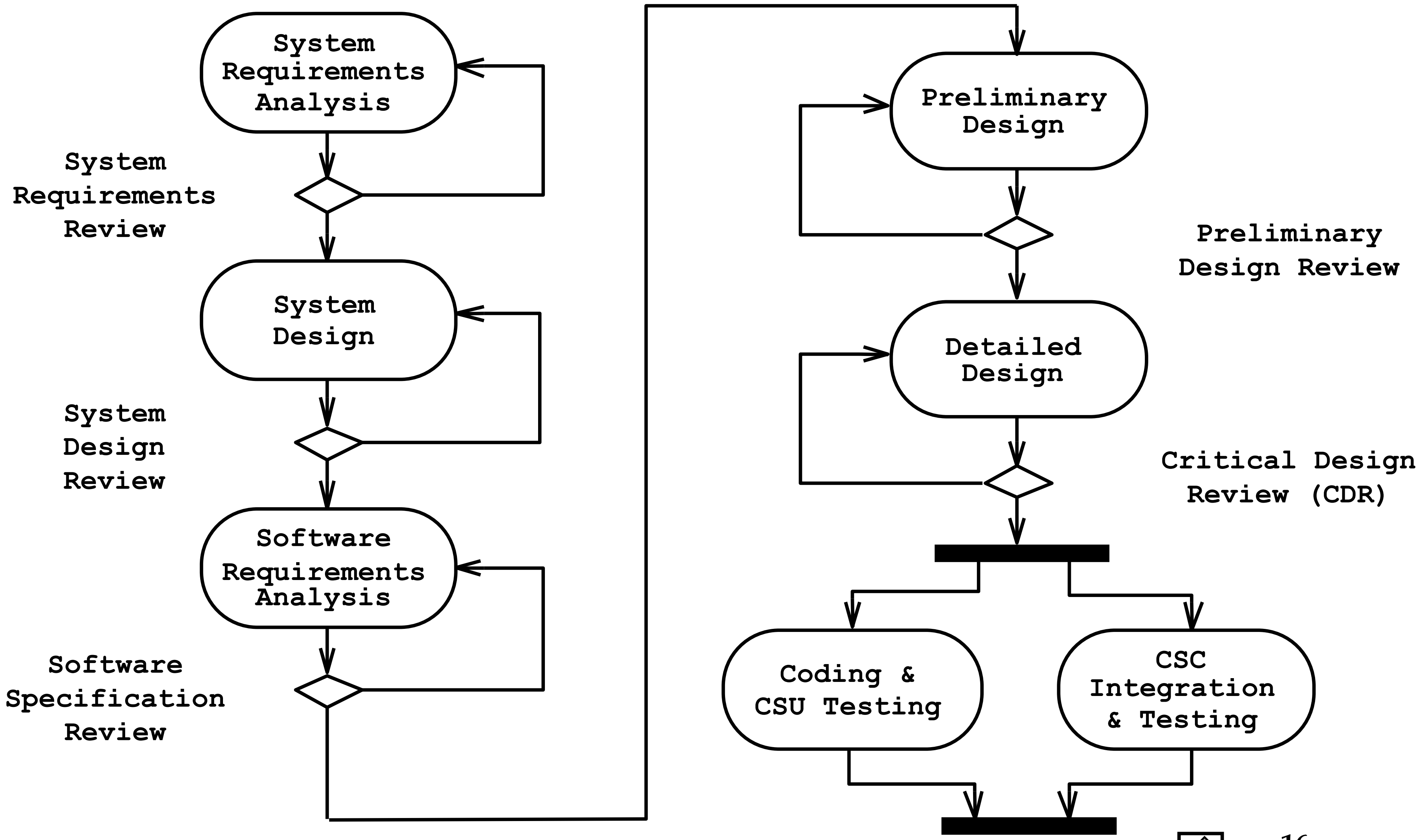

Concept Exploration Process
System Allocation Process
Requirements Process
Design Process
Implementation Process
Verification & Validation Process
Installation Process
Operation & Support Process

adapted from [Royce 1970]

DOD Standard 2167A

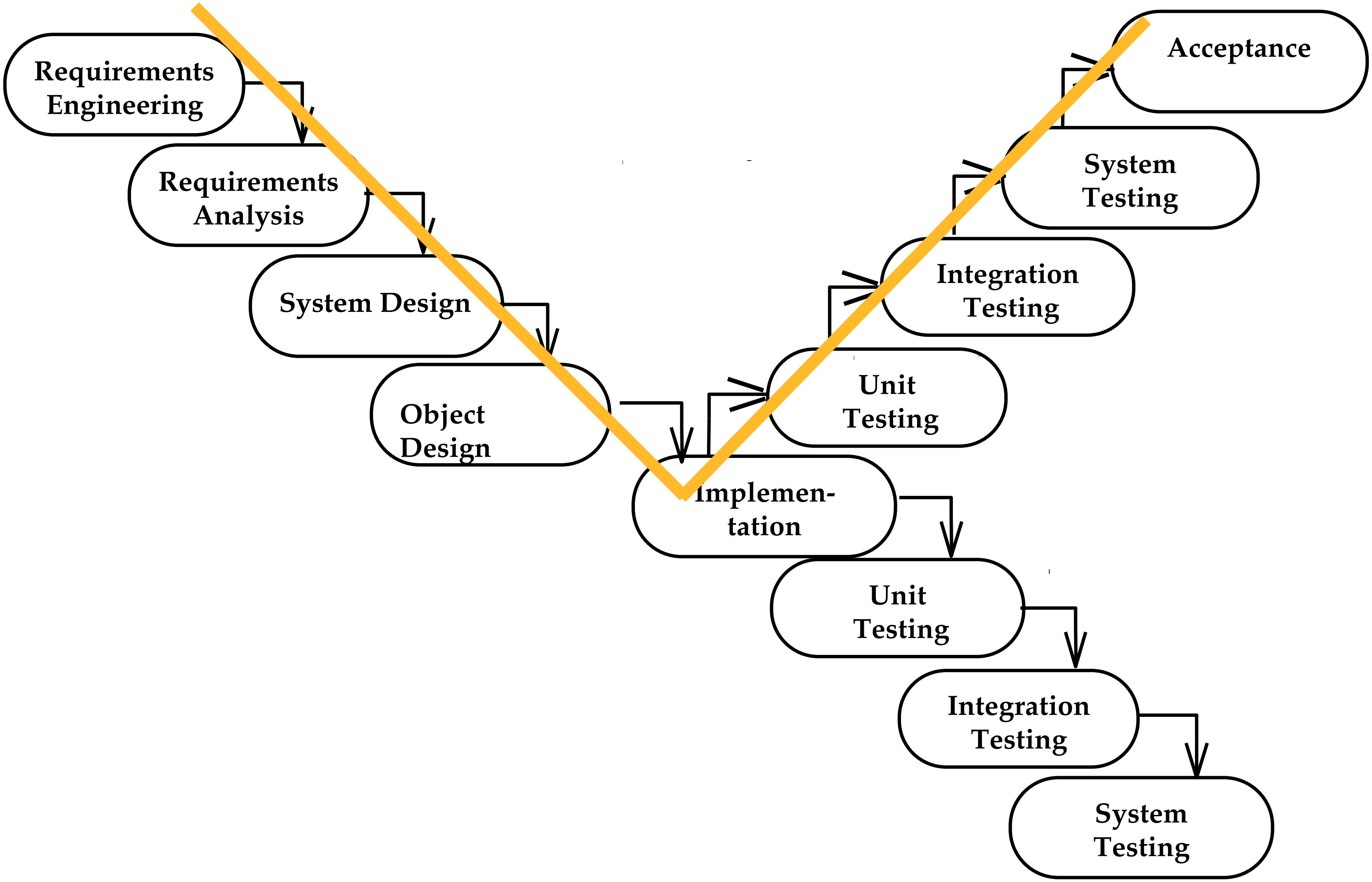
- Example of a waterfall model with the following software development activities
  - System Requirements Analysis/Design
  - Software Requirements Analysis
  - Preliminary Design and Detailed Design
  - Coding and CSU testing
  - CSC Integration and Testing
  - CSCI Testing
  - System integration and Testing
- Required by the U.S. Department of Defense for all software contractors in the 1980-90’s.

Activity Diagram of MIL DOD-STD-2167A

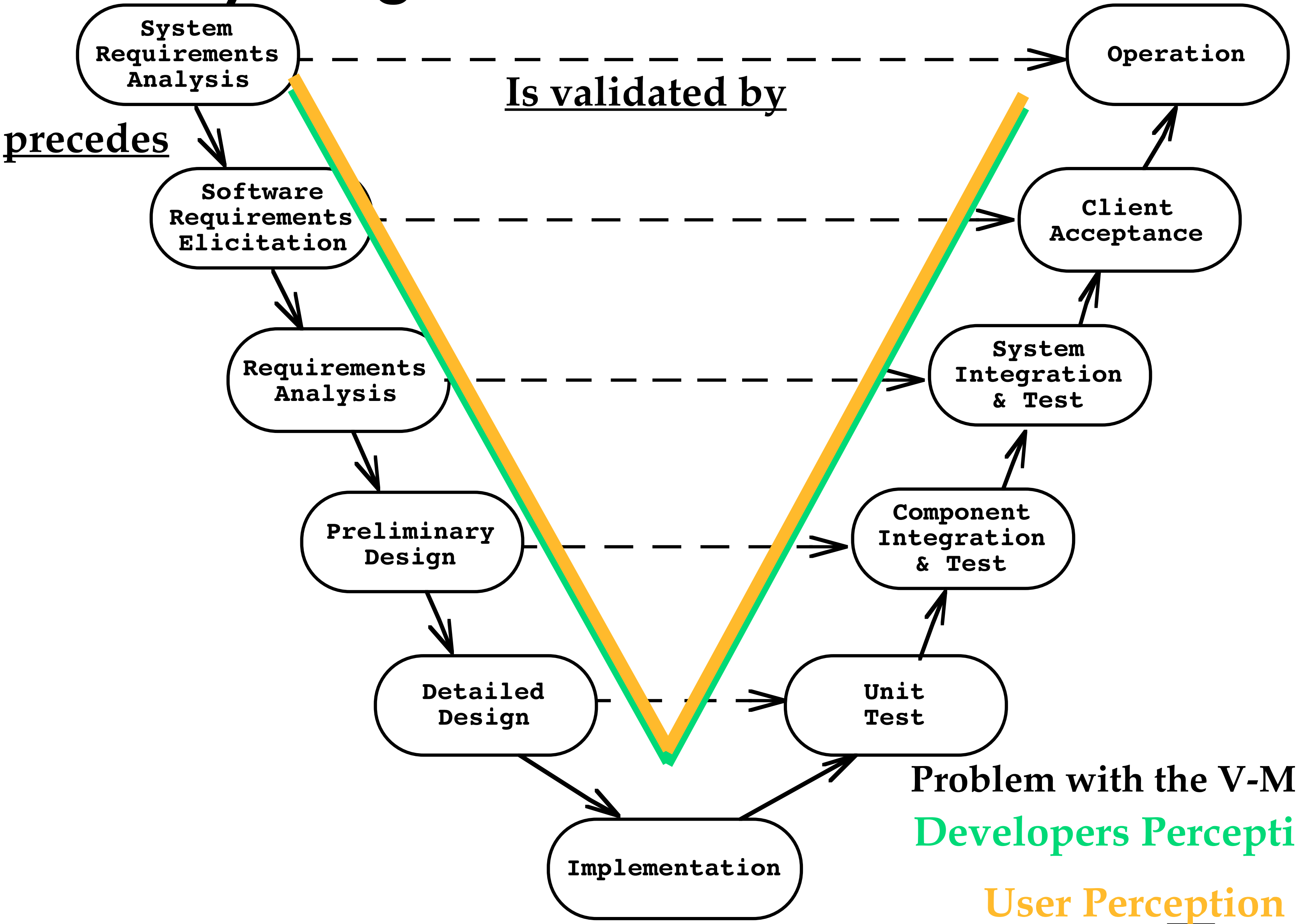

System Requirements Analysis
System Requirements Review
System Design
System Design Review
Software Requirements Analysis
Software Specification Review
Preliminary Design
Preliminary Design Review
Detailed Design
Critical Design Review (CDR)
Coding & CSU Testing
CSC Integration & Testing

...

From the Waterfall Model to the V Model



Activity Diagram of the V Model



Problem with the V-Model:

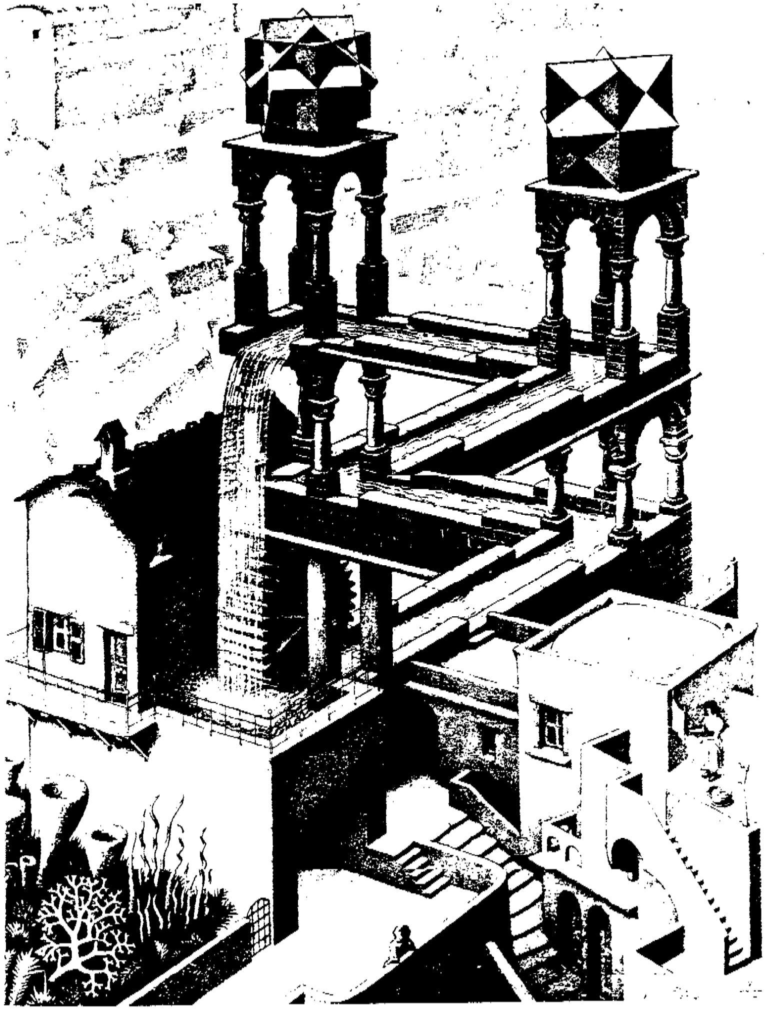
Developers Perception =

User Perception

Properties of Waterfall-based Models

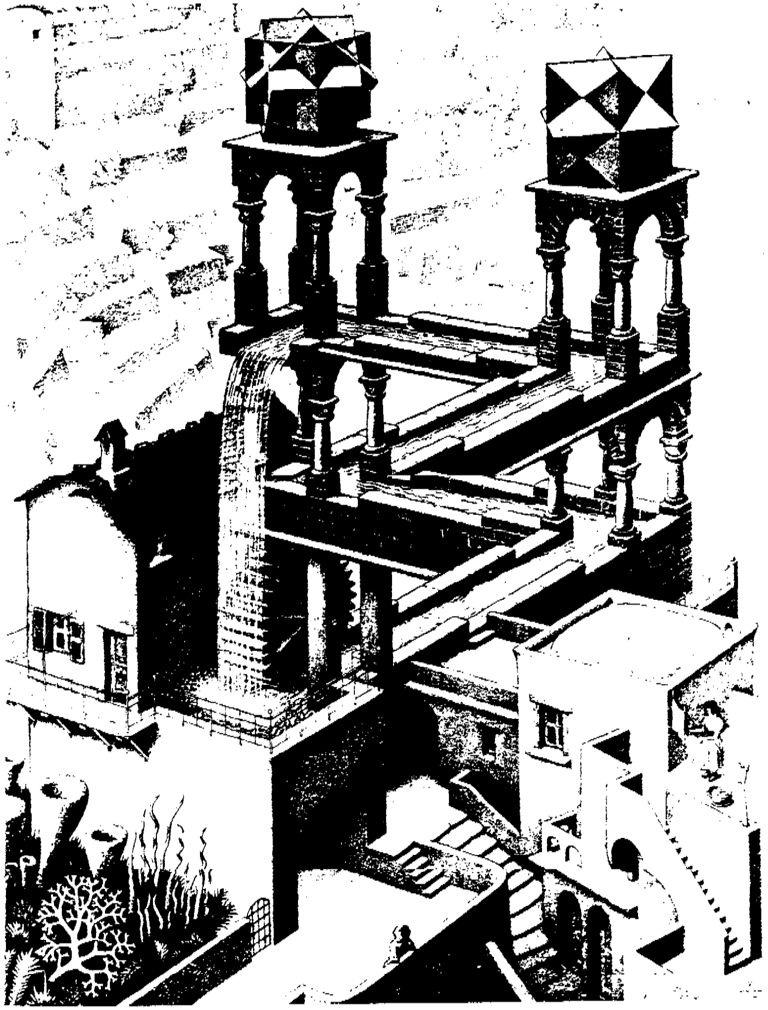
- Managers love waterfall models
  - Nice milestones
  - No need to look back (linear system)
  - Always one activity at a time
  - Easy to check progress during development: 90% coded, 20% tested
- However, software development is non-linear
  - While a design is being developed, problems with requirements are identified
  - While a program is being coded, design and requirement problems are found
  - While a program is tested, coding errors, design errors and requirement errors are found.

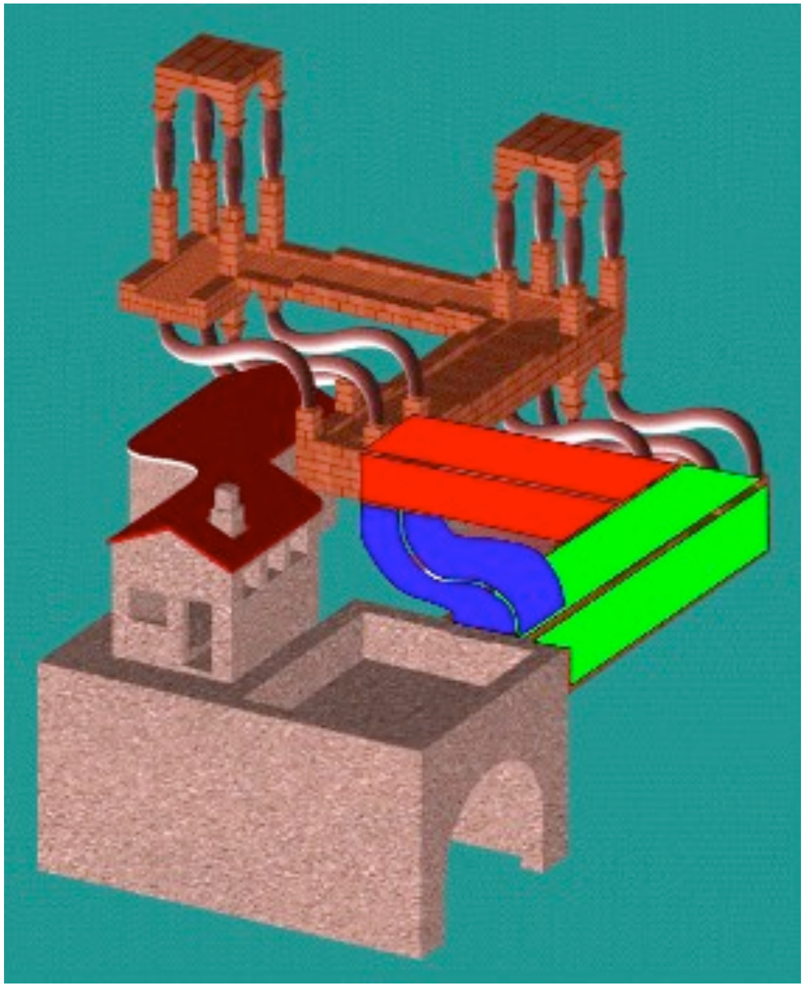
The Alternative: Allow Iteration



Escher was the first:-)

Construction of Escher's Waterfall Model





http://www.cs.technion.ac.il/~gershon/EscherForReal/

Spiral Model

- The spiral model focuses on *addressing risks incrementally*, in order of priority.
- It consists of the following set of activities
  - Determine objectives and constraints
  - Evaluate alternatives
  - Identify risks
  - Resolve risks by assigning priorities to risks
  - Develop a series of prototypes for the identified risks starting with the highest risk
  - Use a waterfall model for each prototype development
  - If a risk has successfully been resolved, evaluate the results of the round and plan the next round
  - If a certain risk cannot be resolved, terminate the project immediately
- This set of activities is applied to a couple of so-called rounds.

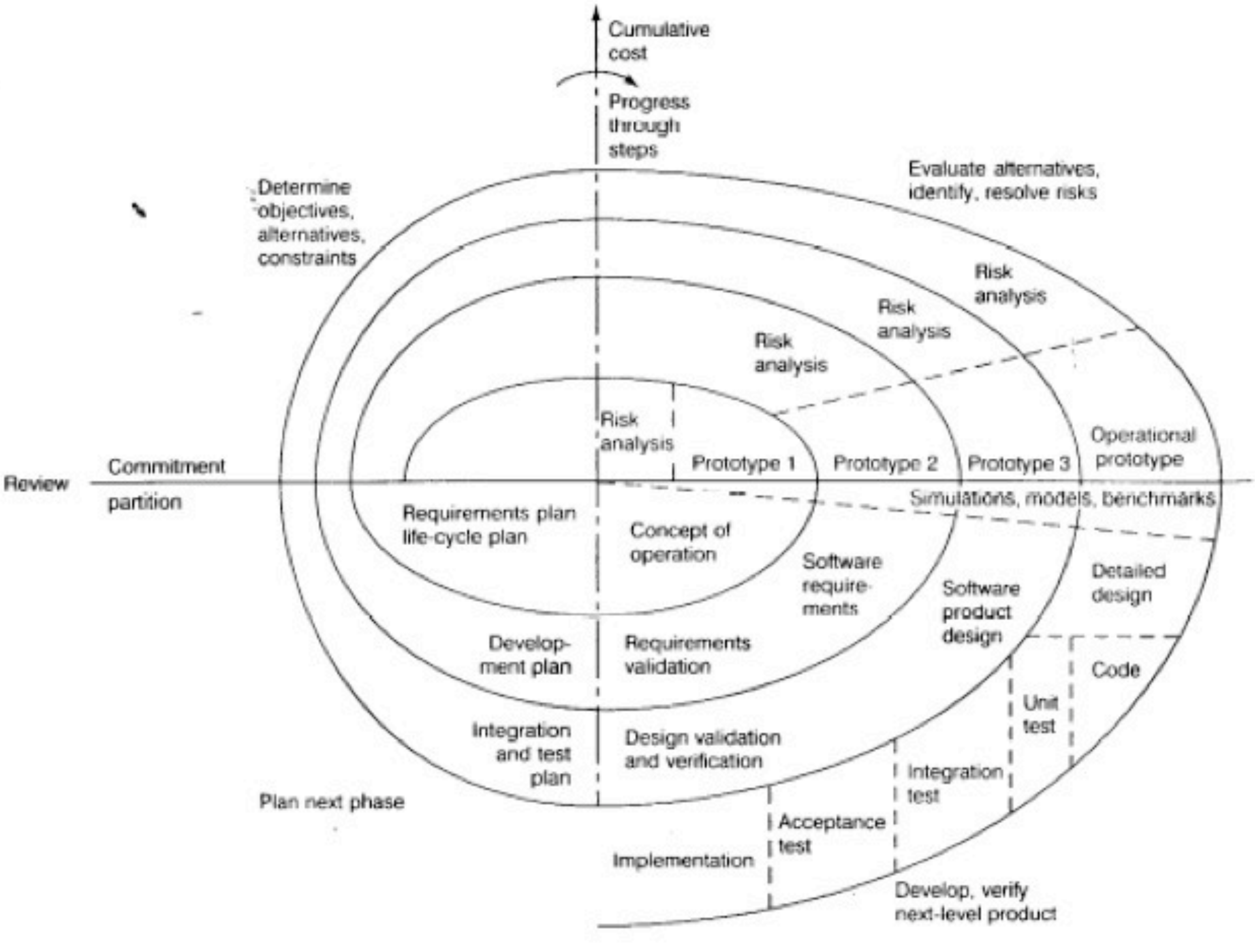
Rounds in Boehm’s Spiral Model

- Concept of Operations
- Software Requirements
- Software Product Design
- Detailed Design
- Code
- Unit Test
- Integration and Test
- Acceptance Test
- Implementation

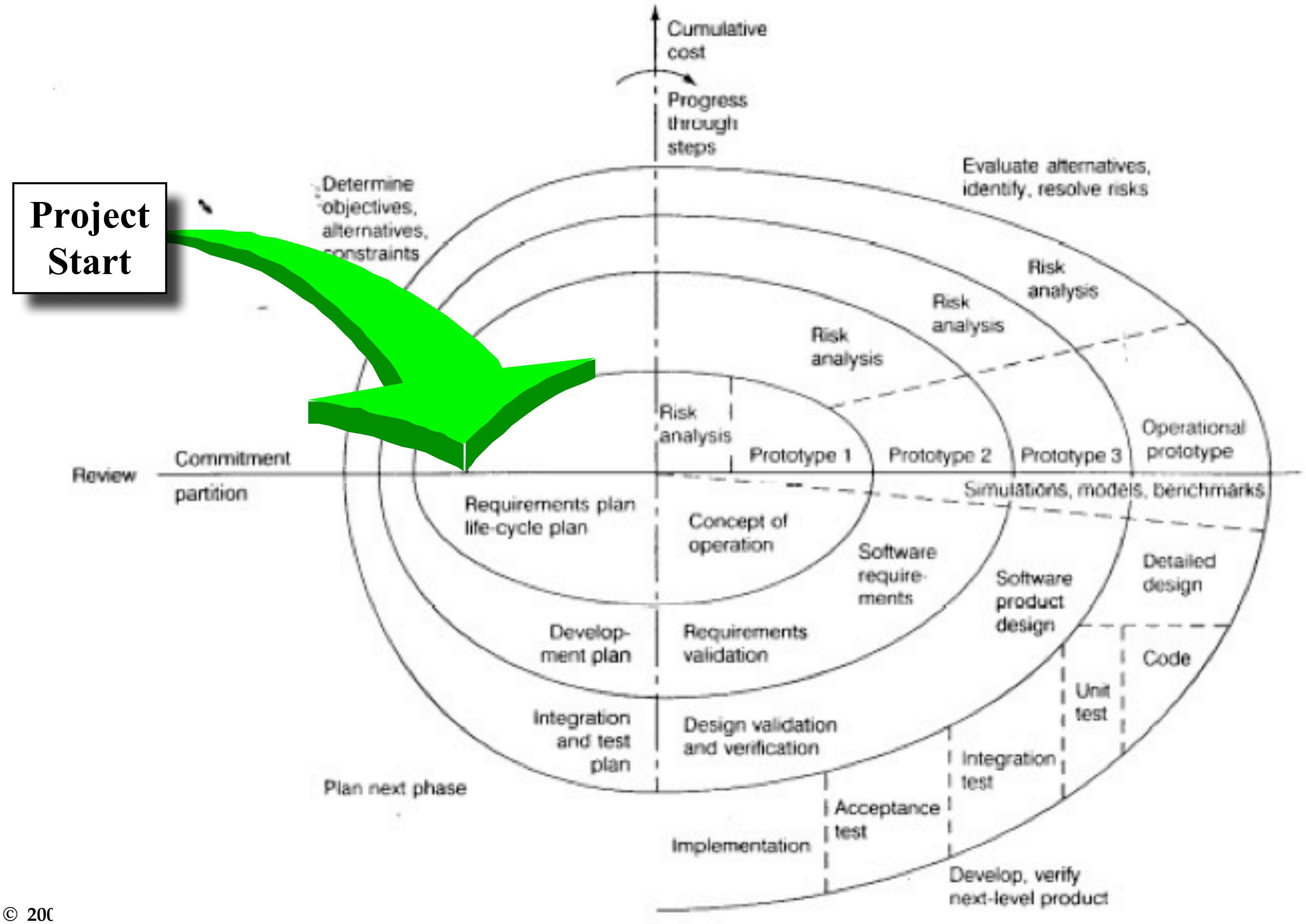
- For each round go through these activities:
  - Define objectives, alternatives, constraints
  - Evaluate alternatives, identify and resolve risks
  - Develop and verify a prototype
  - Plan the next round.

Disccourse on Prototyping

Diagram of Boehm's Spiral Model



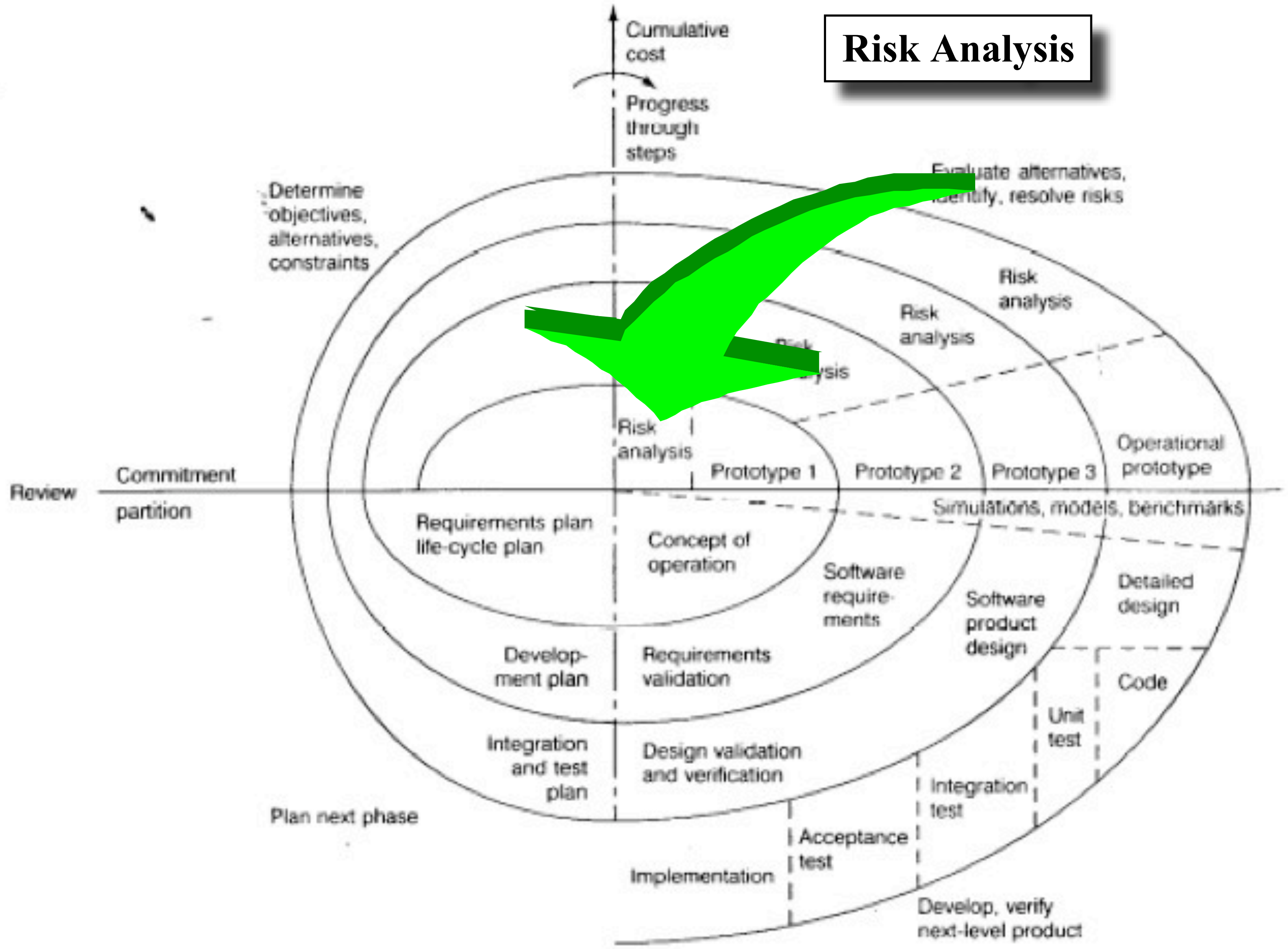
Round 1, Concept of Operations:
Determine Objectives,Alternatives & Constraints



Project Start

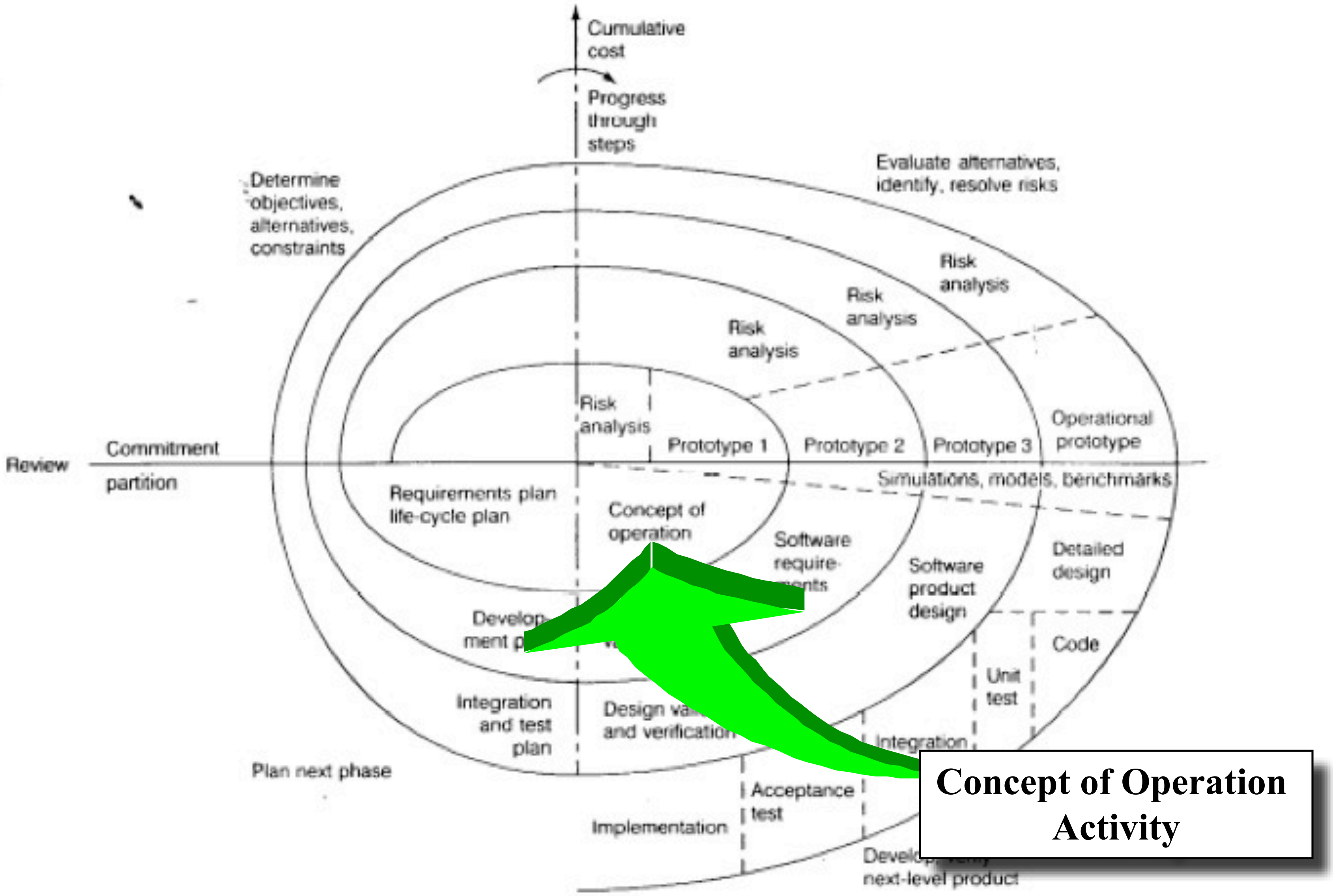
© 200

Round 1, Concept of Operations:
Evaluate Alternatives, identify & resolve Risks



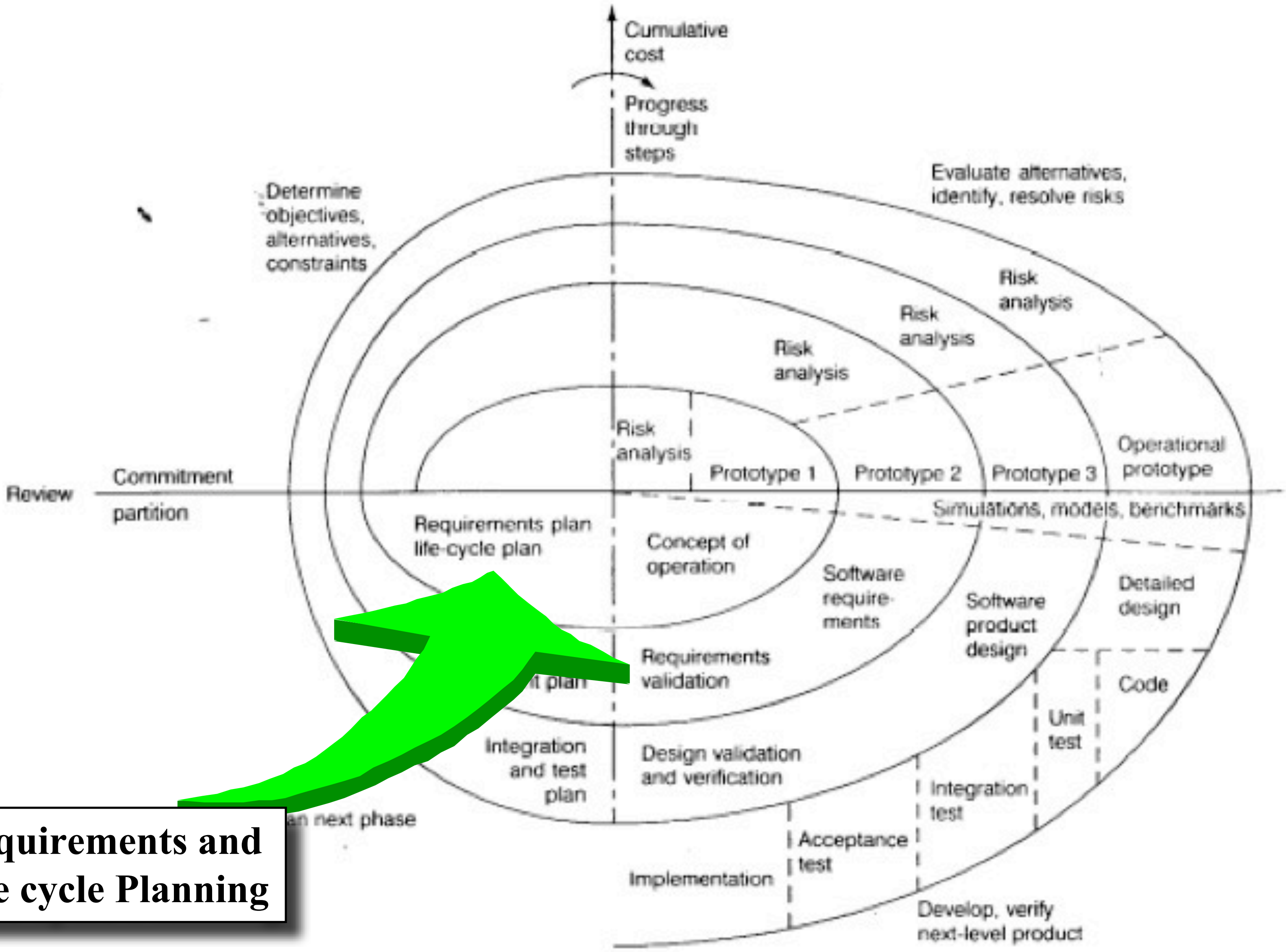
Risk Analysis

Round 1, Concept of Operations:
Develop and Verify



Concept of Operation Activity

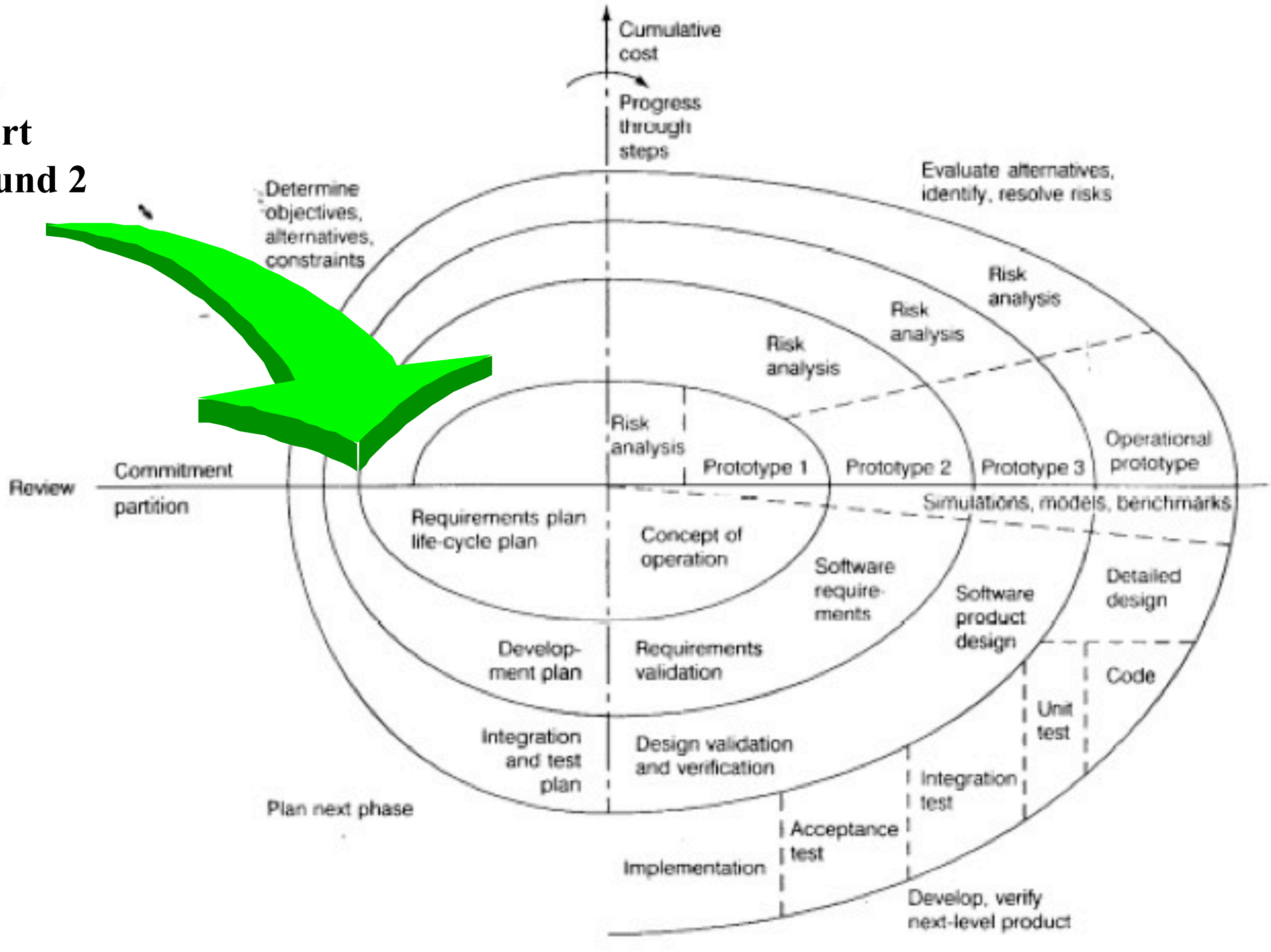
Round 1, Concept of Operations:
Prepare for Next Activity



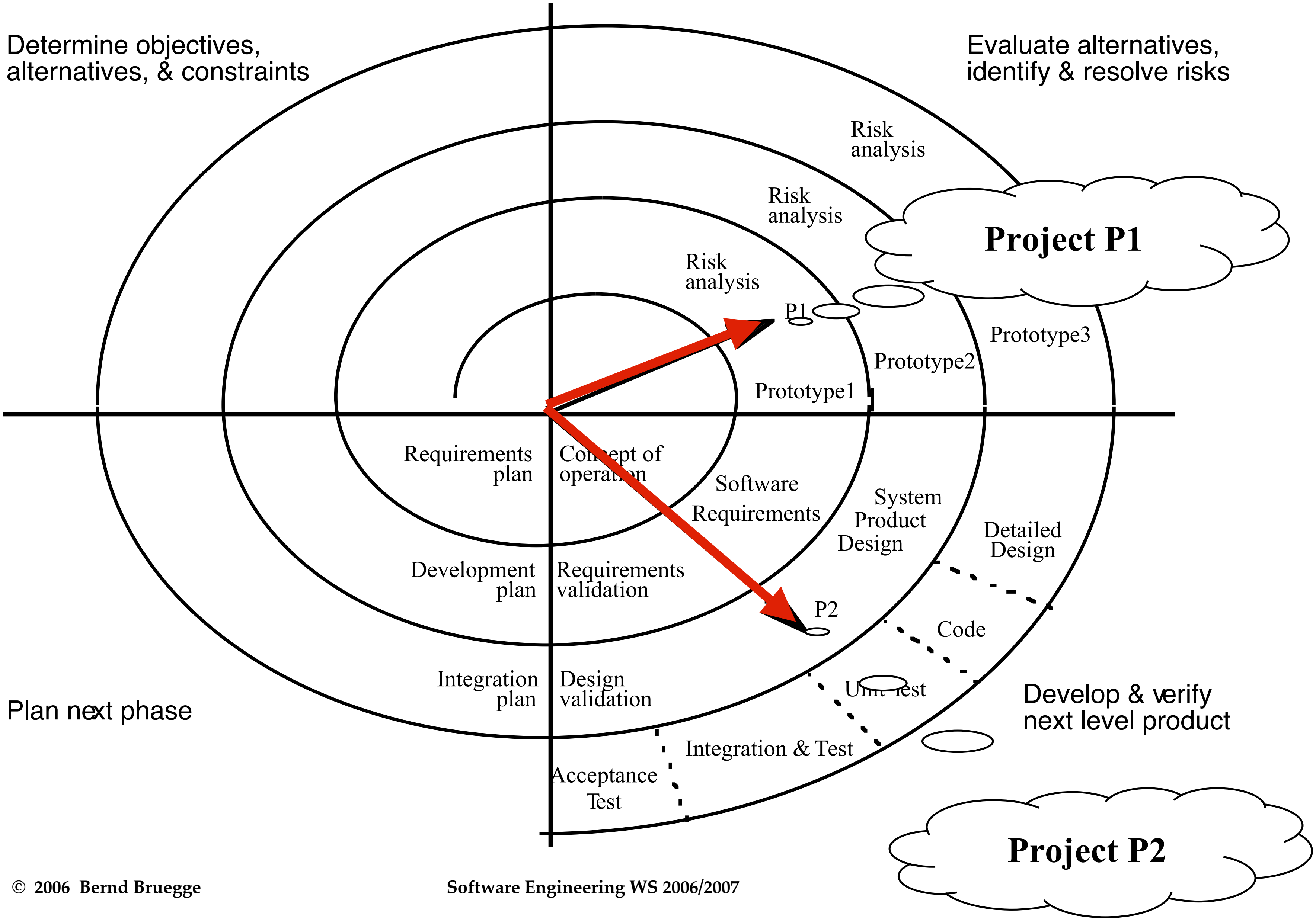
Requirements and Life cycle Planning

Round 2, Software Requirements:
Determine Objectives,Alternatives & Constraints

Start
of Round 2



Comparison of Projects



Outline of Today’s Lecture

- ✓ Modeling the software life cycle
- ✓ Sequential models
  - ✓ Pure waterfall model
  - ✓ V-model
  - ✓ Sawtooth model
- ✓ Iterative models
  - ✓ Boehm’s spiral model
  - ➡ Unified Process
- Entity-oriented models
  - Issue-based model

Unified Process

- The Unified Process is another iterative process model
- States of a software system developed with the Unified Process
    - Inception, Elaboration, Construction, Transition
- Artifacts Sets
    - Management Set, Engineering Set
- Workflows
    - Management, Environment, Requirements, Design, Implementation, Assessment, Deployment
- Iterations are managed as software projects
- Project participants are called stakeholders.

The Unified Process

- The Unified Process supports the following
  - Evolution of project plans, requirements and software architecture with well-defined synchronization points
  - Risk management
  - Evolution of system capabilities through demonstrations of increasing functionality
- Big emphasis on the difference between *engineering* and *production*
- This difference is modeled by introducing two major stages:
  - Engineering stage
  - Production stage.

Difference: Engineering vs. Production

- Engineering Stage:
    - Focuses on analysis and design activities, driven by unpredictable teams
- Production Stage:
    - Focuses on construction, test and deployment, driven by more predictable but larger teams

Focus Factor	Engineering Stage	Production Stage
Risk	Schedule, technical feasibility	Cost
Activities	Planning, Analysis, Design	Implementation, Integration
Artifacts	Requirement Analysis and System Design Documents	Baselines, Releases
Quality Assessment	Demonstration, Inspection	Testing

Phases in the Unified Process

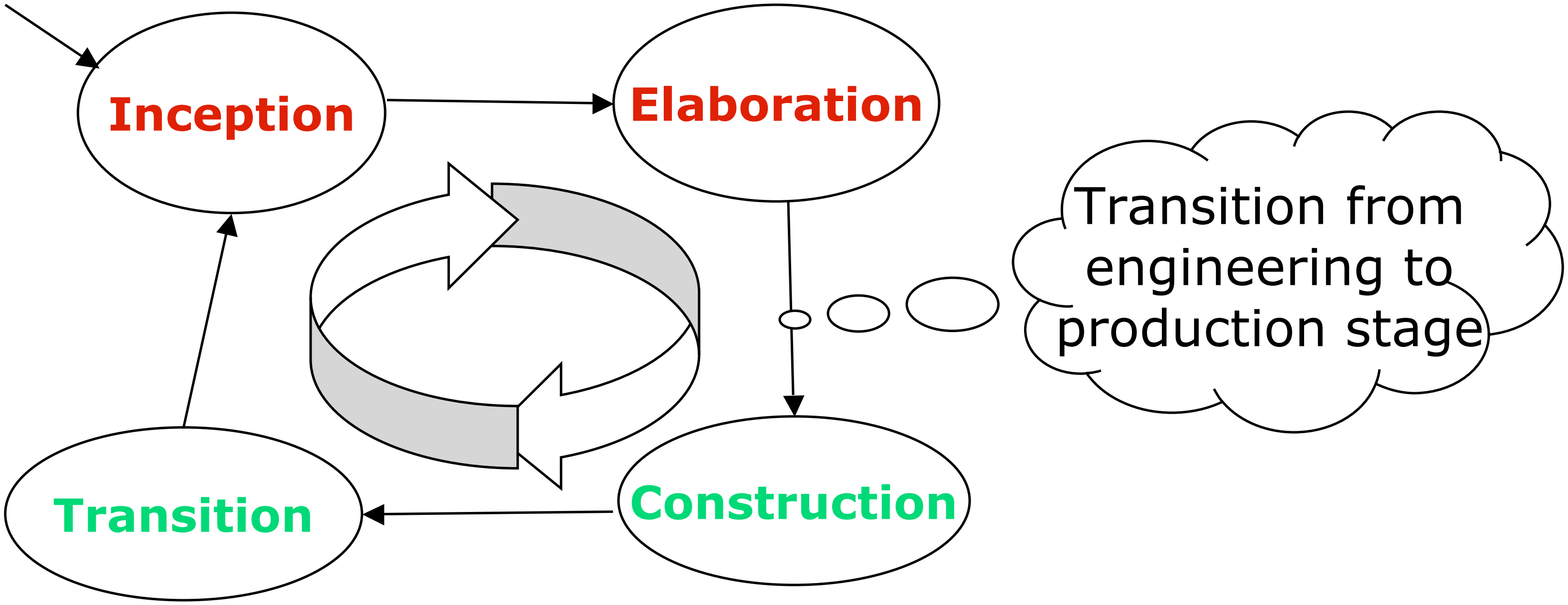
The 2 major stages decomposed into 4 phases

Engineering stage

1. Inception phase
2. Elaboration phase

Production phase

3. Construction phase
4. Transition phase



The phases describe states of the software system to be developed.

Inception Phase: Objectives

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

Elaboration Phase: Objectives

At the end of this phase, the “engineering” of the system is complete

A decision must be made:

- Commit to production phase?
- Move to an operation with higher cost risk and inertia (i.e. bureaucracy)

Main questions:

- Are the system models and project plans stable enough?
- Have the risks been dealt with?
- Can we predict cost and schedule for the completion of the development for an acceptable range?

Construction Phase: Objectives

- Minimize development costs by optimizing resources
  - Avoid unnecessary restarts (modeling, coding)
- Achieve adequate quality as fast as possible
- Achieve useful version
  - Alpha, beta, and other test releases

Transition Phase

- The transition phase is entered
  - when a baseline is mature enough that it can be deployed to the user community
- For some projects the transition phase is
  - the starting point for the next version
- For other projects the transition phase is
  - a complete delivery to a third party responsible for operation, maintenance and enhancement of the software system.

Transition Phase: Objectives

- Achieve independence of users
- Produce a deployment version is complete and consistent
- Build a release as rapidly and cost-effectively as possible.

Iteration in the Unified Process

- Each of the four phases introduced so far (inception, elaboration, construction, transition) consists of one or more iterations
- An iteration represents a set of activities for which
  - have a milestone (“a well-defined intermediate event”)
  - the scope and results are captured with work-products called artifacts.

Artifact Sets

- Artifact set
  - A set of work products that are persistent and in a uniform representation format (natural language, Java, UML,…)
  - Every element in the set is developed and reviewed as a single entity
- The Unified Process distinguishes five artifact sets:
  - Management set
  - Requirements set
  - Design set
  - Implementation set
  - Deployment set

Also called Engineering set.

Artifact Sets in the Unified Process

Requirements Set	Design Set	Implementation Set	Deployment Set
1. Vision document ("problem statement") 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

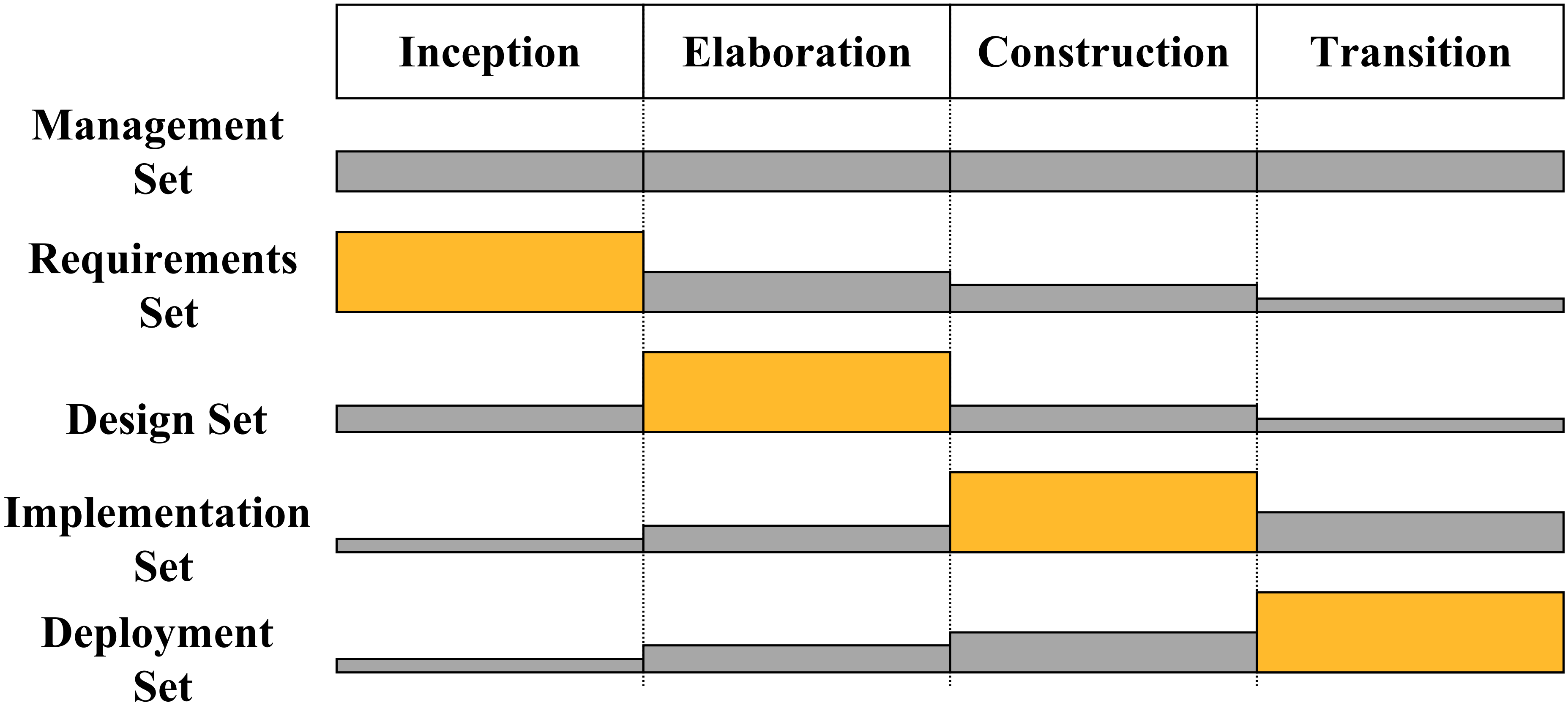
1. Work breakdown structure
2. Business Case
3. Release specifications
4. Software Project Management Plan

Operational Artifacts

1. Release descriptions
2. Status assessments
3. Software change order database
4. Deployment documents
5. Environment

Focus on Artifact Sets during Development

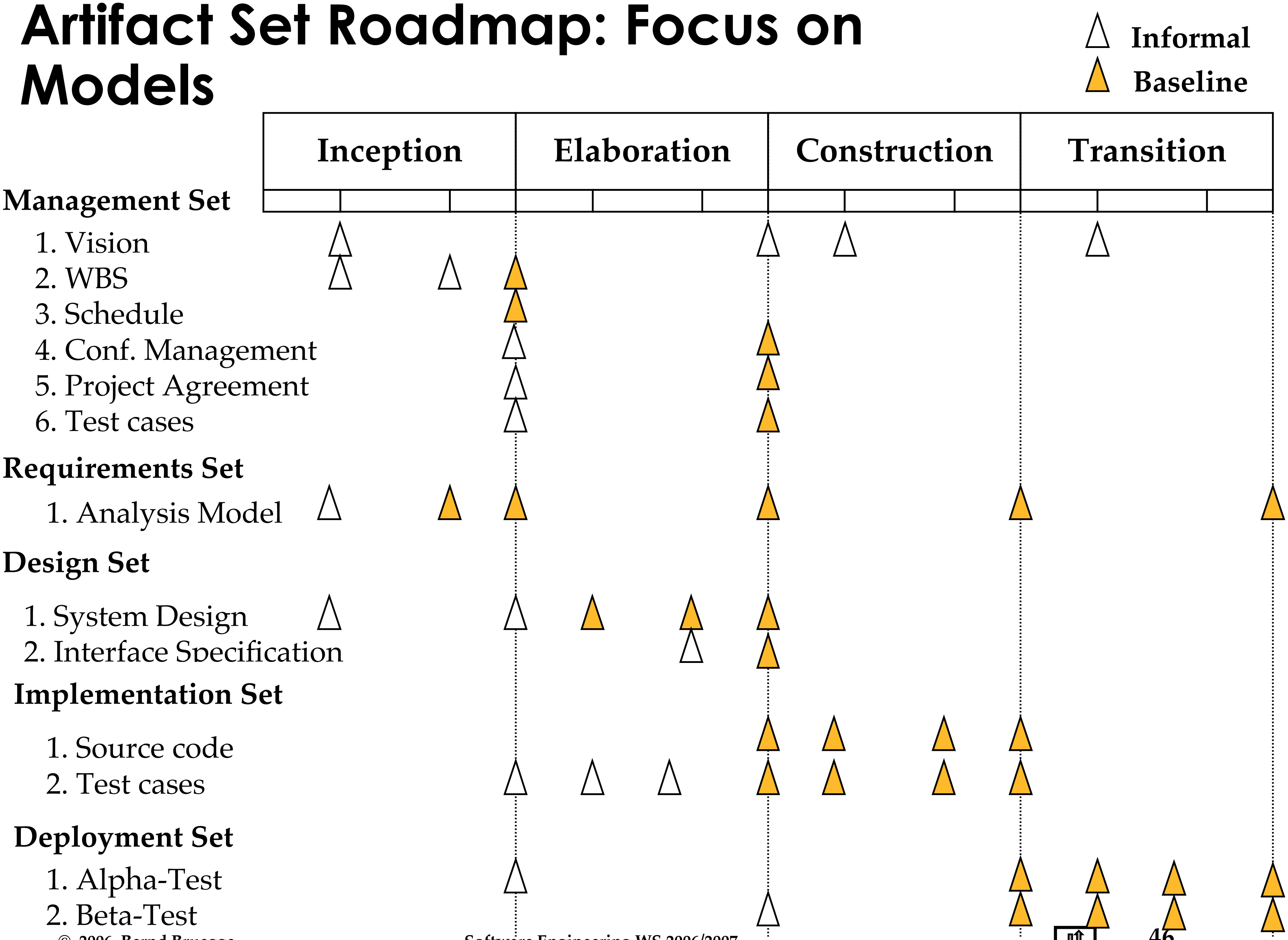
- Each artifact set is the predominant focus in one stage of the unified process



Management of Artifact Sets

- Some artifacts are changed only after a phase
- Other artifacts are updated after each minor milestone, i.e. after an iteration
- The project manager is responsible
  - to manage and visualize the sequence of artifacts across the software lifecycle activities
  - This visualization is often called artifact roadmap.

Artifact Set Roadmap: Focus on Models



△ Informal

▲ Baseline

Inception | Elaboration | Construction | Transition

Management Set

1. Vision
2. WBS
3. Schedule
4. Conf. Management
5. Project Agreement
6. Test cases

Requirements Set

1. Analysis Model

Design Set

1. System Design
2. Interface Specification

Implementation Set

1. Source code
2. Test cases

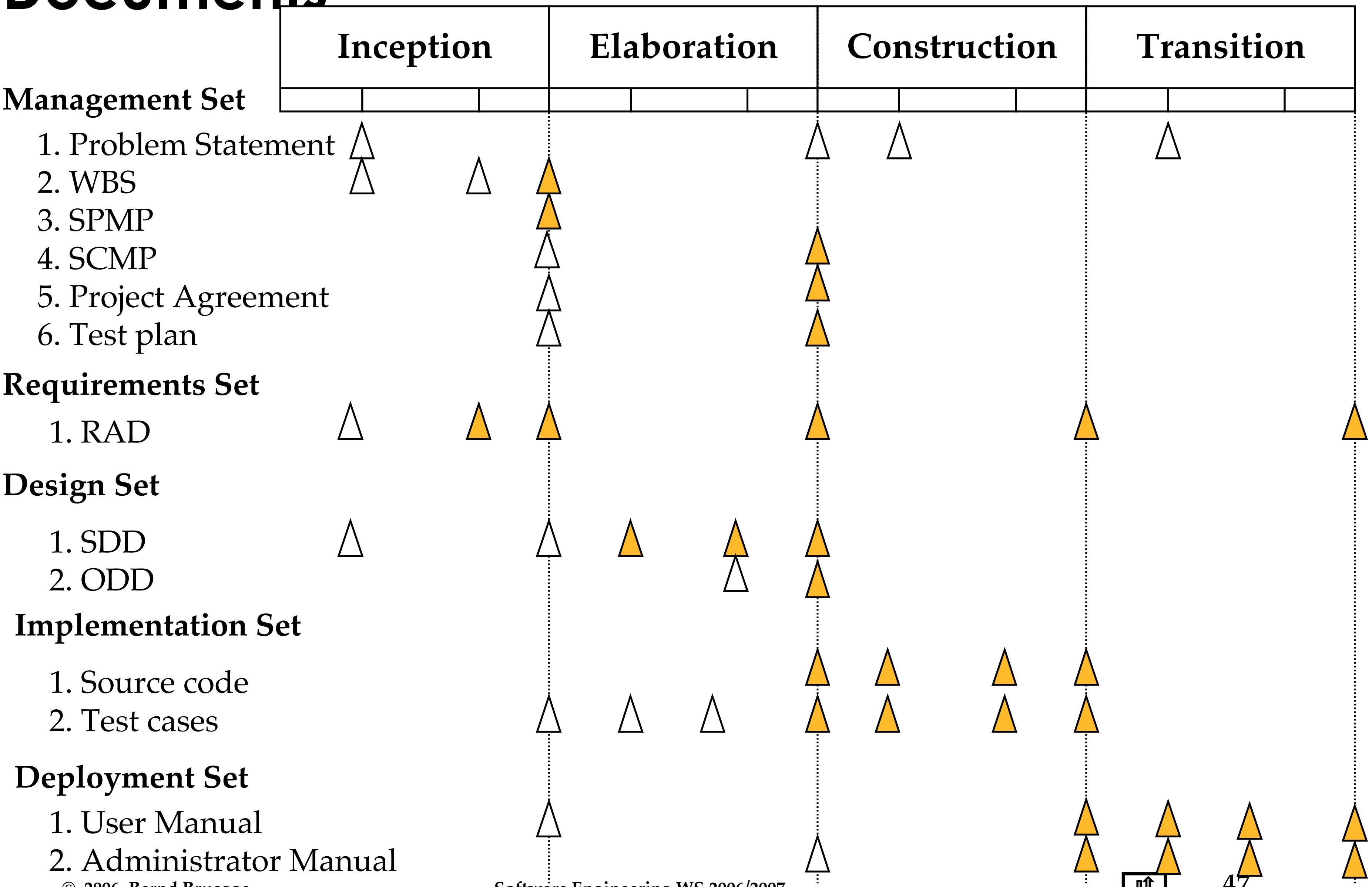
Deployment Set

1. Alpha-Test
2. Beta-Test

Artifact Set Roadmap: Focus on Documents

△ Informal

▲ Baseline



Models vs. Documents

- Documentation-driven approach
  - The production of the documents drives the milestones and deadlines
- Model-driven approach
  - The production of the models drive the milestones deadlines

- Main goal of a modern software development project
  - Creation of models and construction of the software system
  - The purpose of documentation is to support this goal.

Reasons for Documentation-Driven Approach

- No rigorous engineering methods and languages available for analysis and design models
- Language for implementation and deployment is too cryptic
- Software project progress needs to be assessed
  - Documents represent a mechanism for demonstrating progress
- People want to review information
  - but do not understand the language of the artifact
- People wanted to review information,
  - but do not have access to the tools to view the information.

Artifact-Driven Approach

- Provides templates for documents at the start of the project
- Instantiates documents automatically from these templates
  - Enriches them with modeling and artifact information generated during the project
- Tools automatically generate documents from the models. Examples:
  - Schedule generator
  - Automatic requirements document generator
  - Automatic interface specification generator
  - Automatic analysis and design documents generator
  - Automatic test case generator.

“Process” is an overloaded term

- The Unified Process distinguishes between macro and micro process:
  - The macro process models the software lifecycle
  - The micro process models activities that produce artifacts
- Another meaning for process:
  - Business process
    - The policies, procedures and practices in an organization pursuing a software-intensive line of business.
    - Focus: Organizational improvement, long-term strategies, and return on investment (ROI)
- The micro processes are called workflows in the Unified Process.

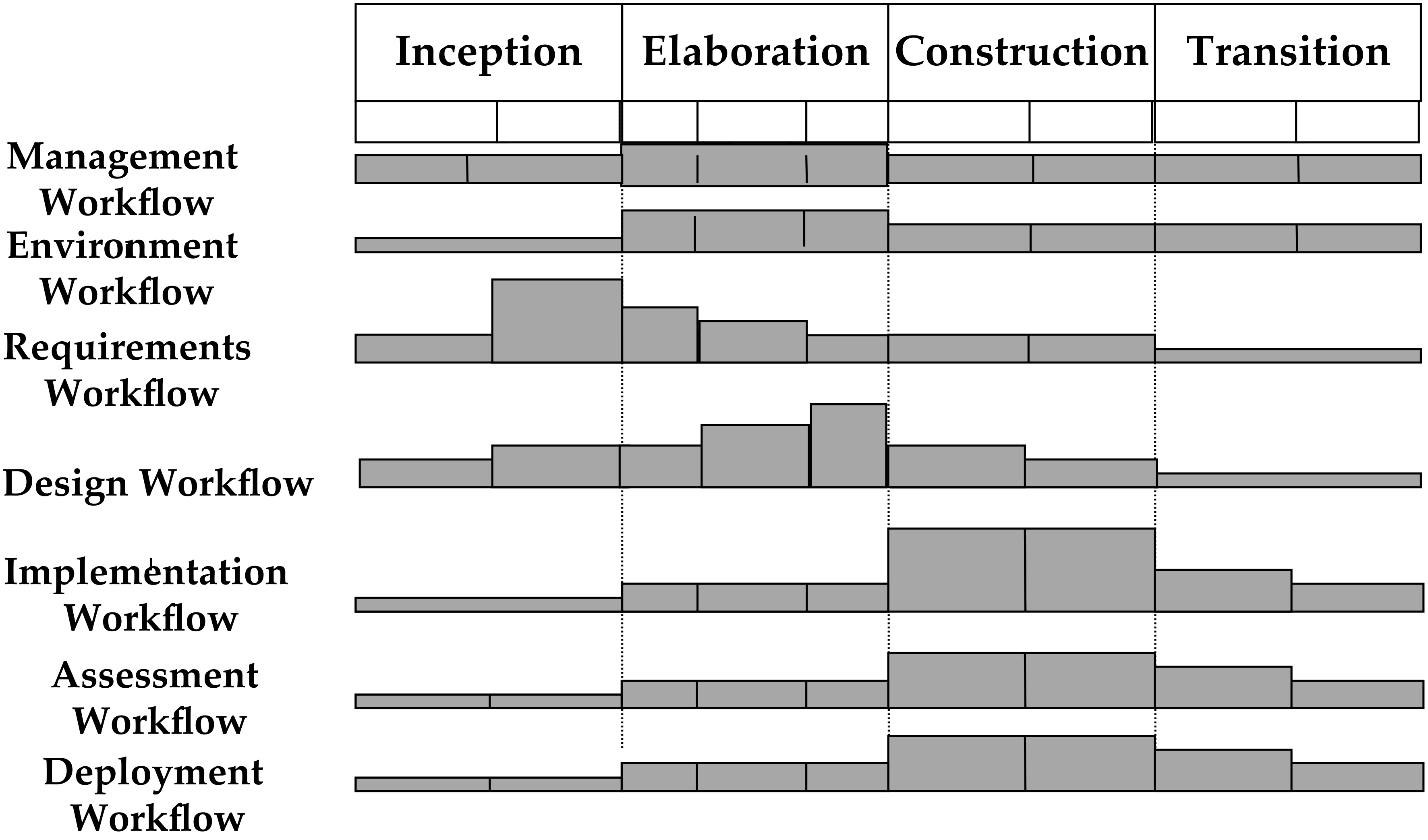
Workflows in the Unified Process (1)

- Management workflow
  - Planning the project (Problem statement, SPMP, SCMP, Test plan)
- Environment workflow
  - Automation of process and maintenance environment. Setup of infrastructure (Communication, Configuration management, ...).
- Requirements workflow
  - Analysis of application domain and creation of requirements artifacts (analysis model).
- Design workflow
  - Creation of solution and design artifacts (system design model, object design model).

Workflows in the Unified Process (2)

- Implementation workflow
  - Implementation of solution, source code testing, maintenance of implementation and deployment artifacts (source code).
- Assessment workflow
  - Assess process and products (reviews, walkthroughs, inspections, testing…)
- Deployment workflow
  - Transition the software system to the end user

Workflows work across Phases



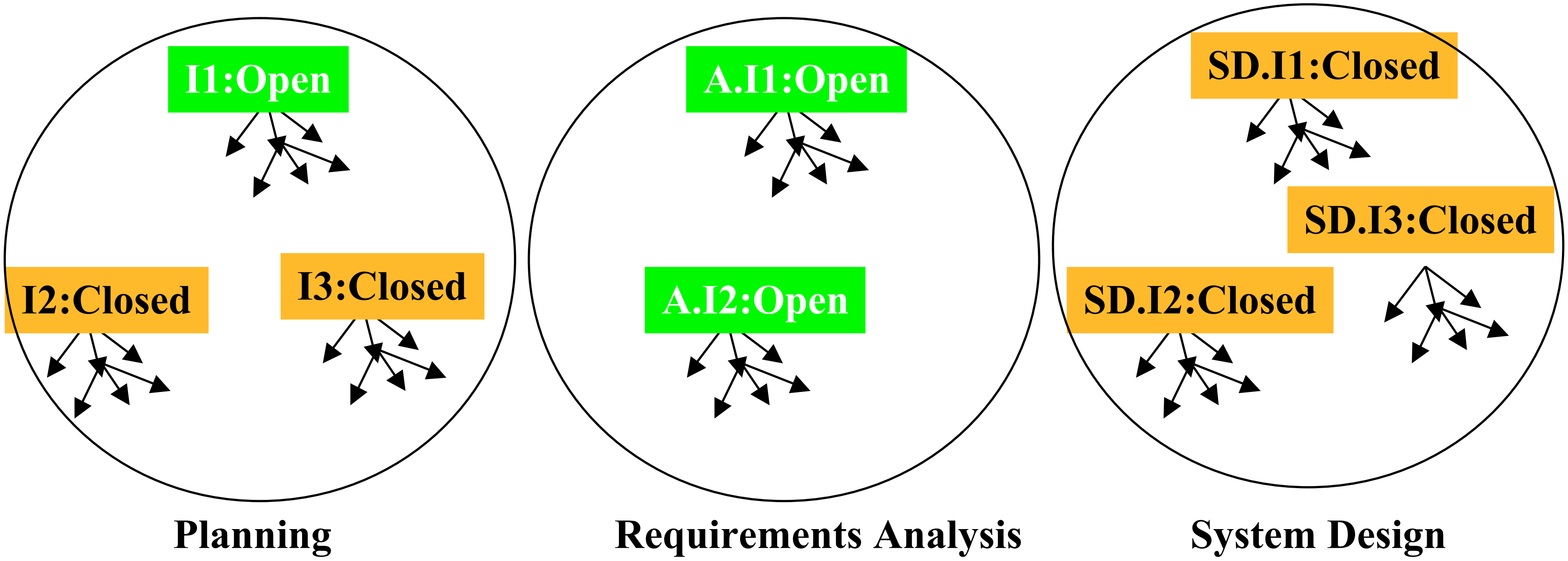
- Workflows create artifacts (documents, models)
- Workflows consist of one or more iterations per phase

Limitations of Waterfall and iterative Models

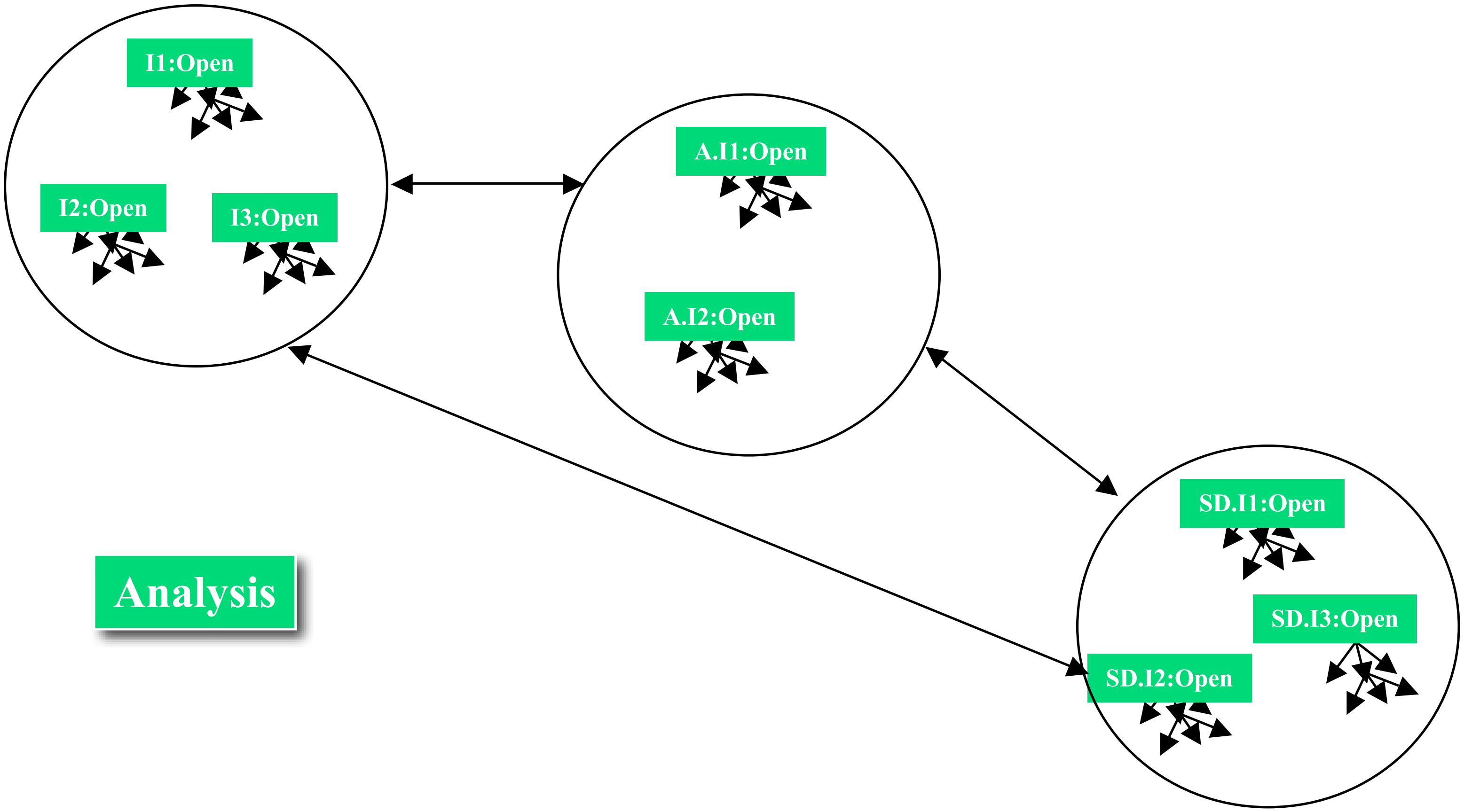
- Neither of these models deal well with frequent change
    - The Waterfall model assumes that once you are done with a phase, all issues covered in that phase are closed and cannot be reopened
    - The Spiral and Unified Process model can deal with change between phases, but do not allow change within a phase
- What do you do if change is happening more frequently?
    - “The only constant is the change”

An Alternative: Issue-Based Development

- A system is described as a collection of issues
  - Issues are either closed or open
  - Closed issues have a resolution
  - Closed issues can be reopened (Iteration!)
- The set of closed issues is the basis of the system model

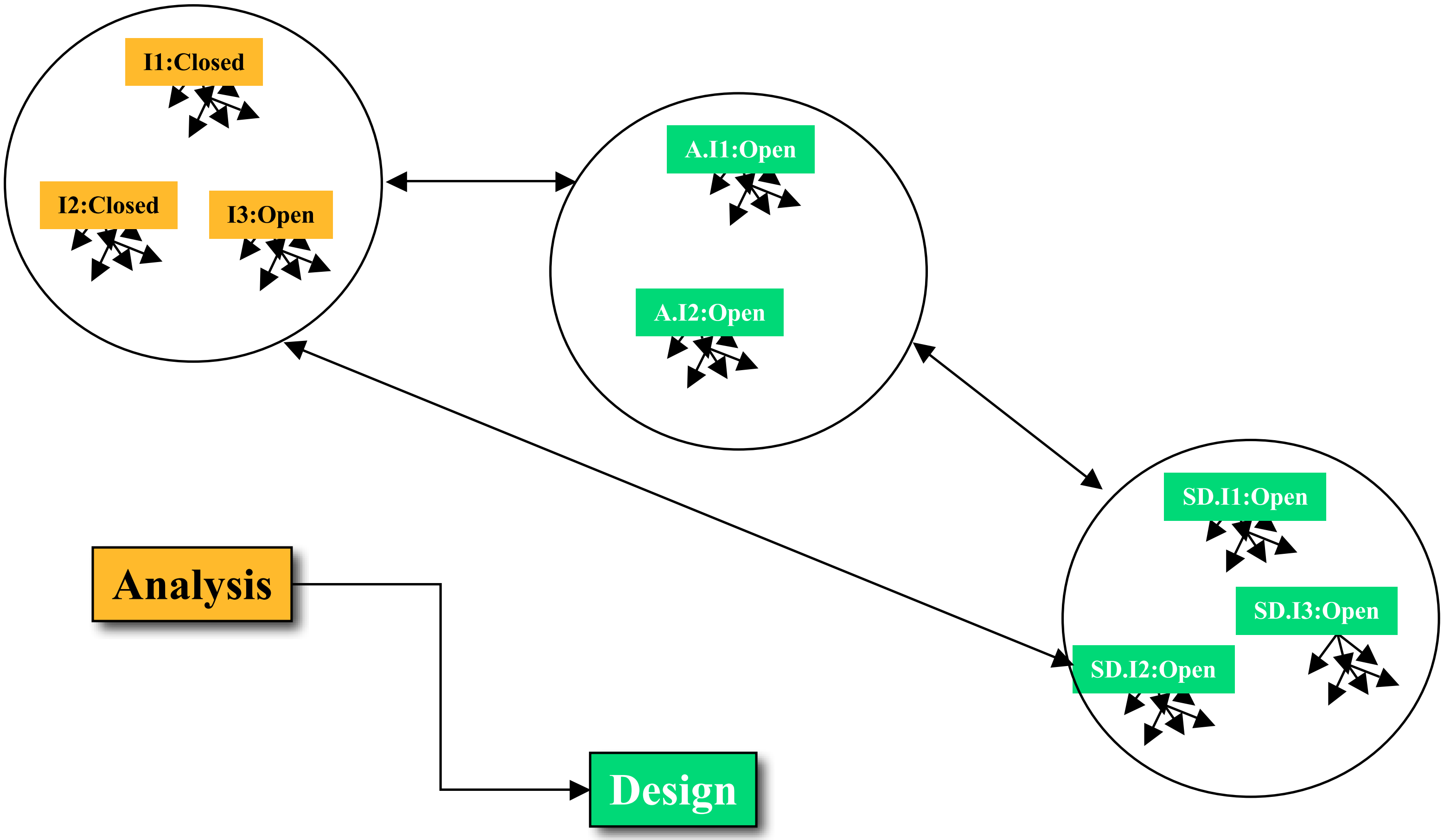


Waterfall Model: Analysis Phase

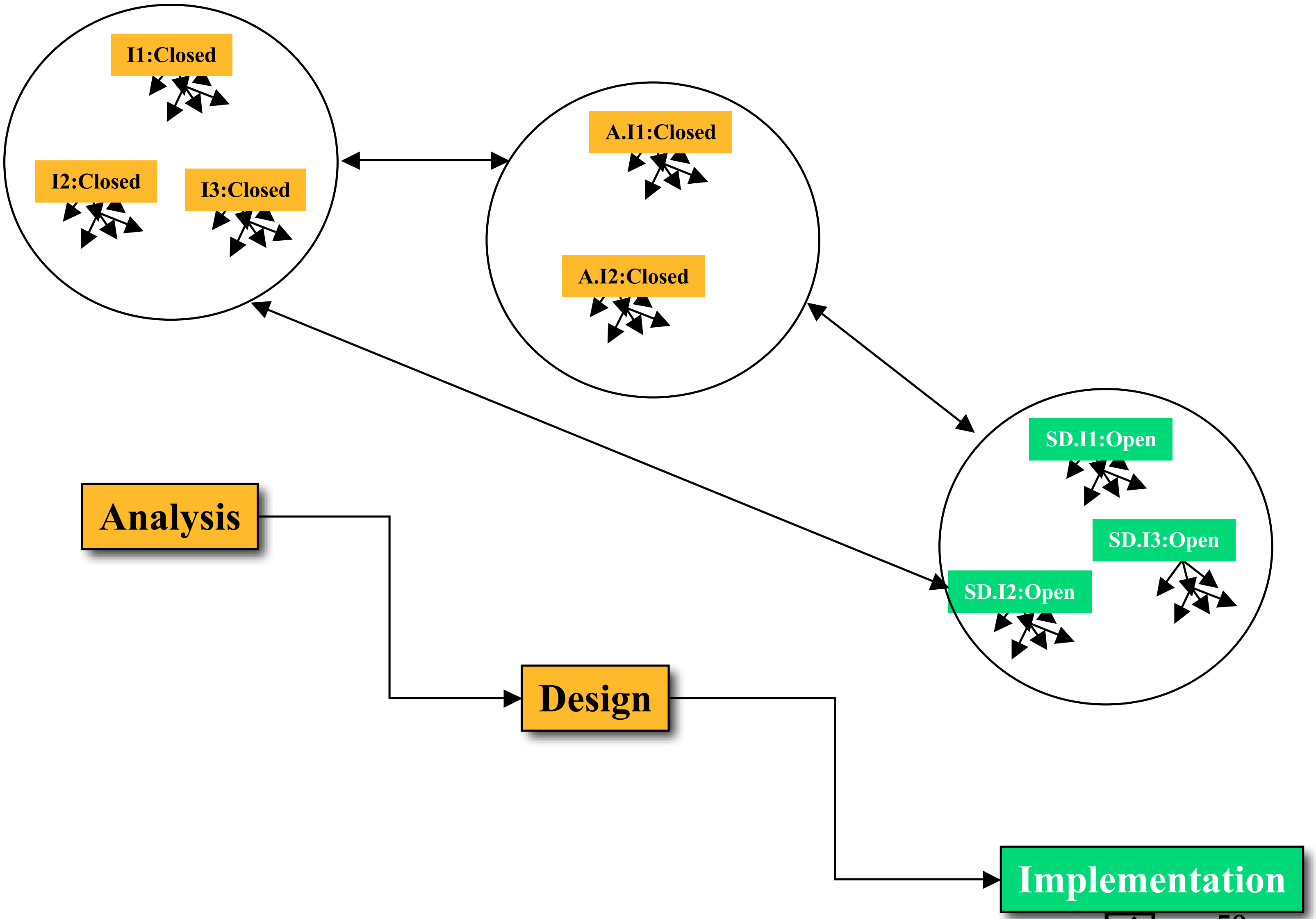


Analysis

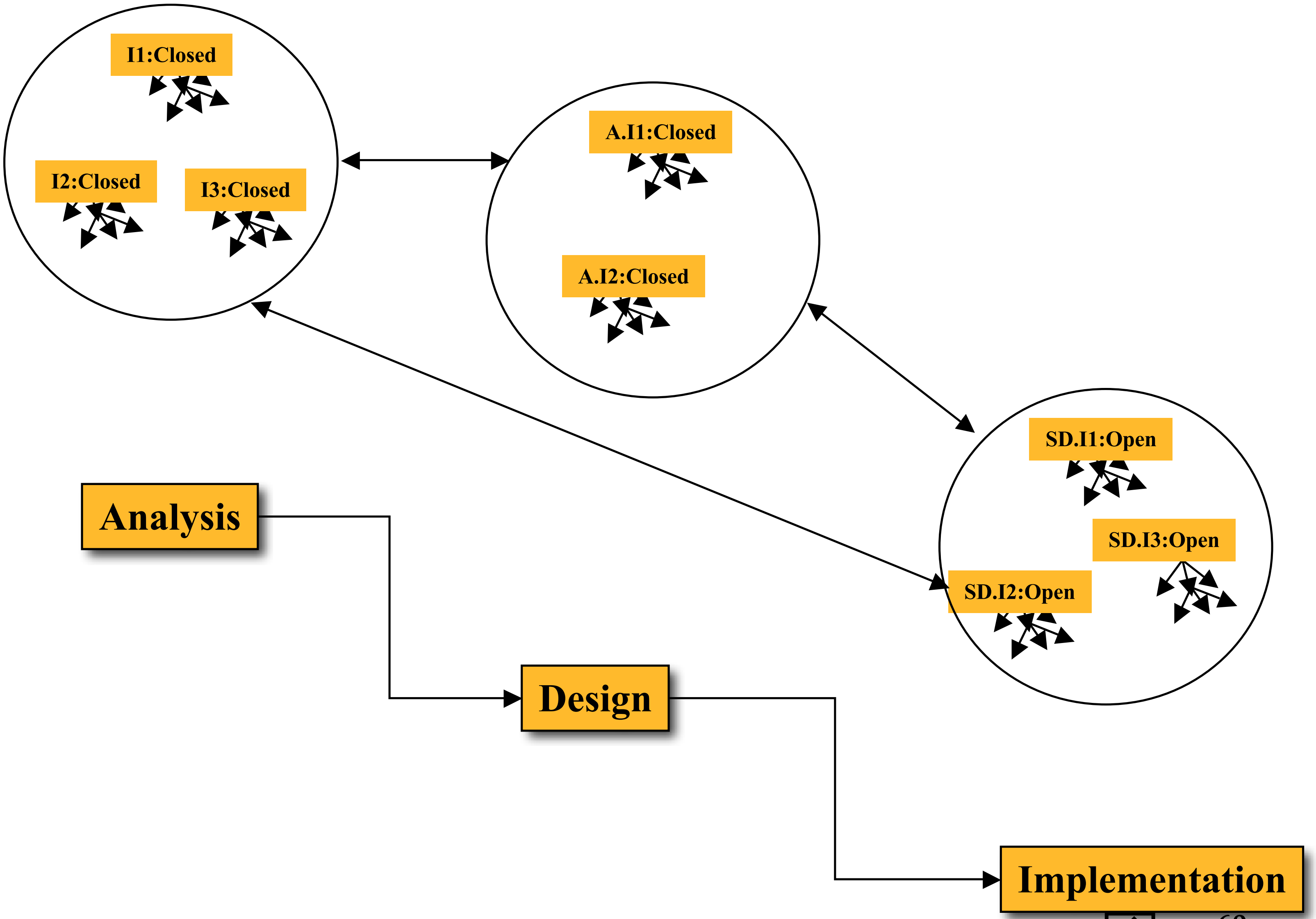
Waterfall Model: Design Phase



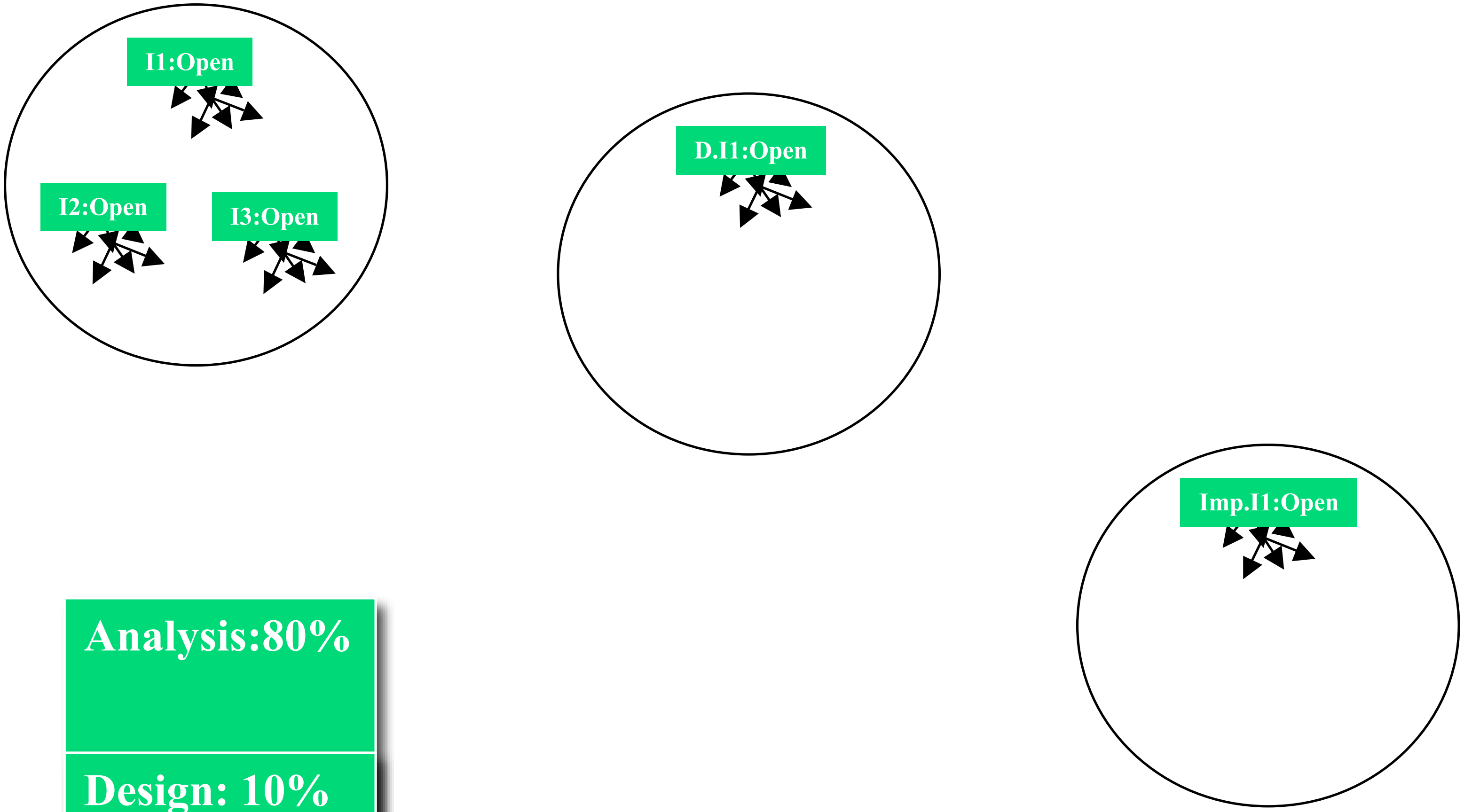
Waterfall Model: Implementation Phase



Waterfall Model: Project is Done



Issue-Based Model: Analysis Phase



I1:Open

I2:Open

I3:Open

D.I1:Open

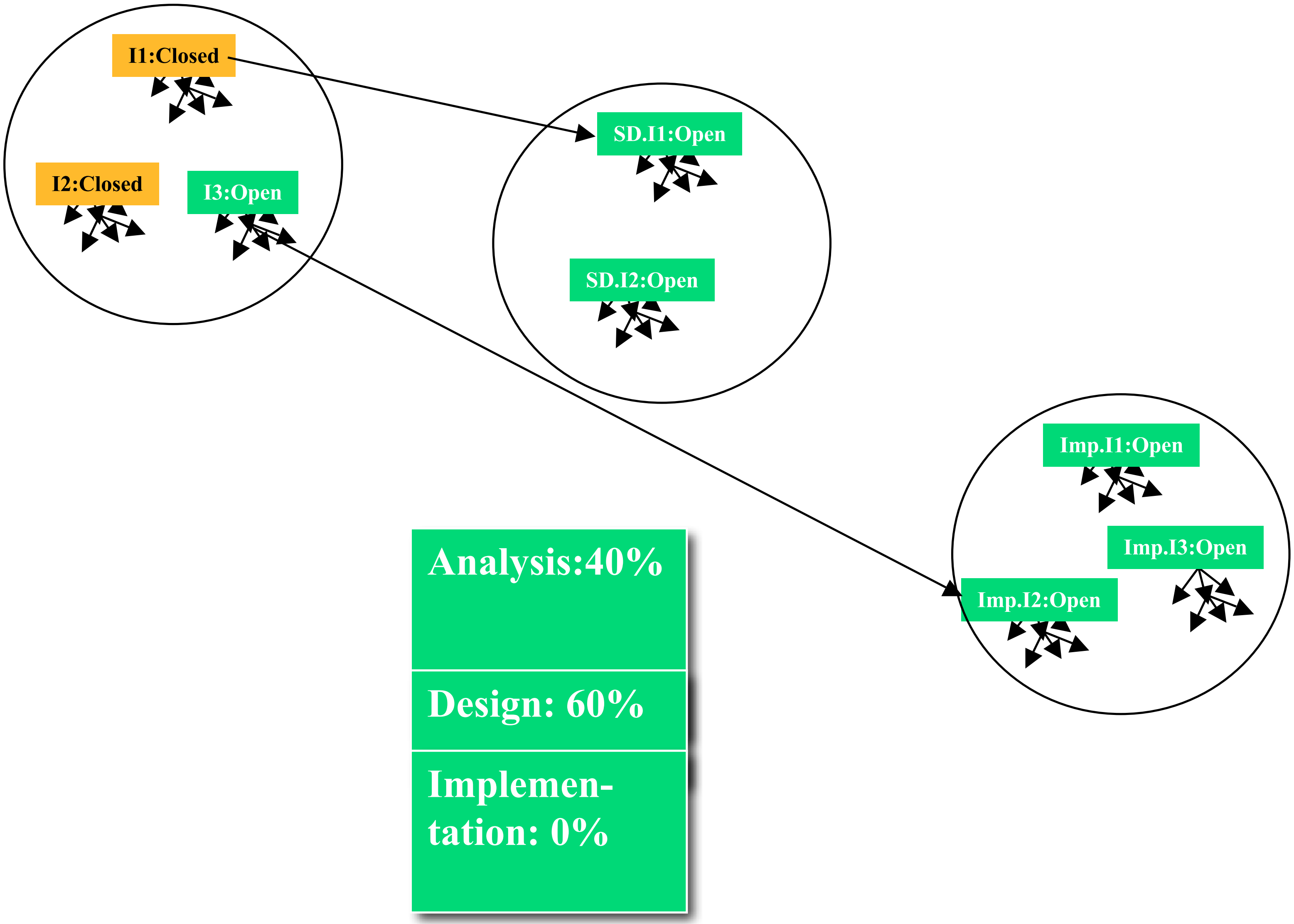
Imp.I1:Open

Analysis:80%

Design: 10%

Implementation: 10%

Issue-Based Model: Design Phase

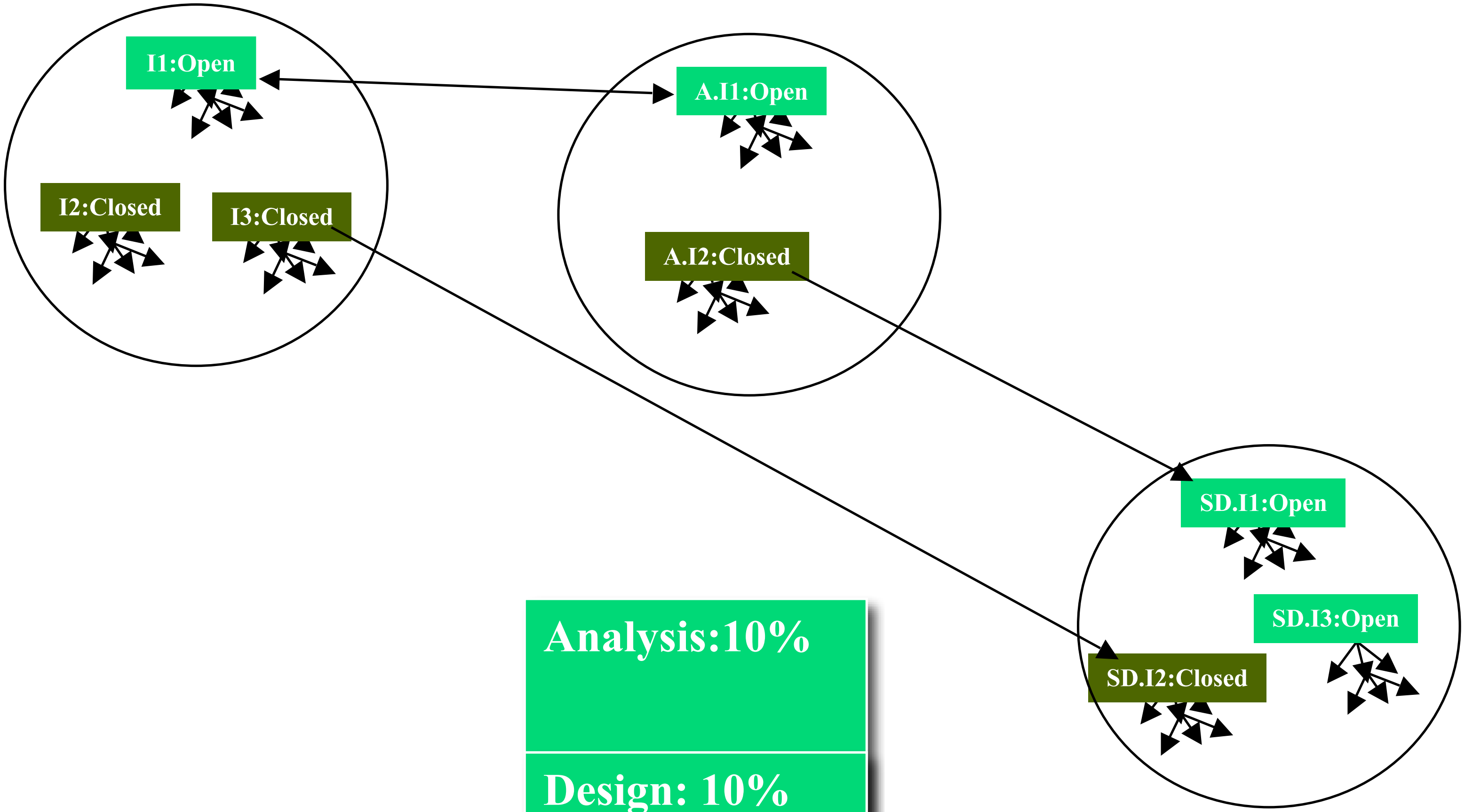


Analysis:40%

Design: 60%

Implementation: 0%

Issue-Based Model: Implementation Phase



I1:Open

I2:Closed

I3:Closed

A.I1:Open

A.I2:Closed

SD.I1:Open

SD.I3:Open

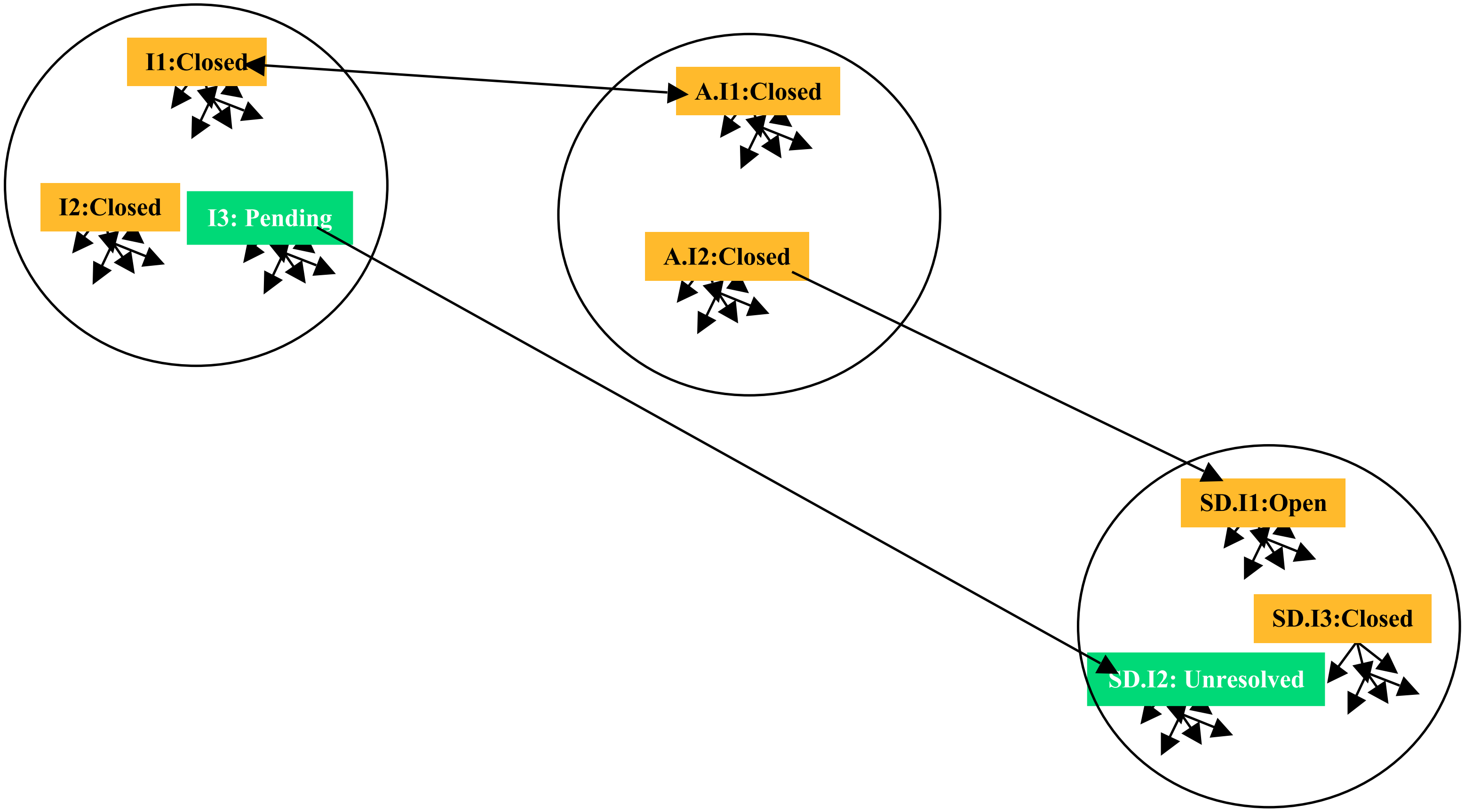
SD.I2:Closed

Analysis:10%

Design: 10%

Implementation: 60%

Issue-Based Model: Prototype is Done



Frequency of Change and Choice of Software Lifecycle Model

PT = Project Time, MTBC = Mean Time Between Change

- Change rarely occurs (MTBC » PT)
  - Waterfall Model
  - Open issues are closed before moving to next phase
- Change occurs sometimes (MTBC ≈ PT)
  - Boehm’s Spiral Model, Unified Process
  - Change occurring during phase may lead to iteration of a previous phase or cancellation of the project
- Change is frequent (MTBC « PT)
  - Issue-based Development (Concurrent Development)
  - Phases are never finished, they all run in parallel.

Summary Unified Process

- Unified Process: Iterative software lifecycle model
  - Emphasis on early construction of a software architecture
  - Emphasis on early demonstrations of the system
- Definitions
  - 4 phases: Inception, Elaboration, Construction, Transition
  - 7 workflows: Management, environment, requirements, design, implementation, assessment, deployment.
  - 5 artifact sets: Management set, requirements set, design set, implementation set, deployment set
- Iteration: Repetition within a workflow.
- A unified process iteration should be treated as a software project.

Summary

- Software life cycle models
  - Sequential models
    - Pure waterfall model and V-model
  - Iterative model
    - Boehm’s spiral model
    - Unified process
  - Entity-oriented models
    - Issue-based model
    - Sequential models can be modeled as special cases of the issue-based model
- Prototyping
  - A specific type of system model
    - Illustrative, functional and exploratory prototypes
  - Revolutionary and evolutionary prototyping
  - Time-boxed prototyping is a better term than rapid prototyping.

Additional References

- Walker Royce
  - Software Project Management, Addison-Wesley, 1998.
- Ivar Jacobsen, Grady Booch & James Rumbaugh
  - The Unified Software Development Process, Addison Wesley, 1999.
- Jim Arlow and Ila Neustadt
  - UML and the Unified Process: Practical Object-Oriented Analysis and Design, Addison Wesley, 2002.
- Philippe Kruchten
  - Rational Unified Process, Addison-Wesley, 2000.

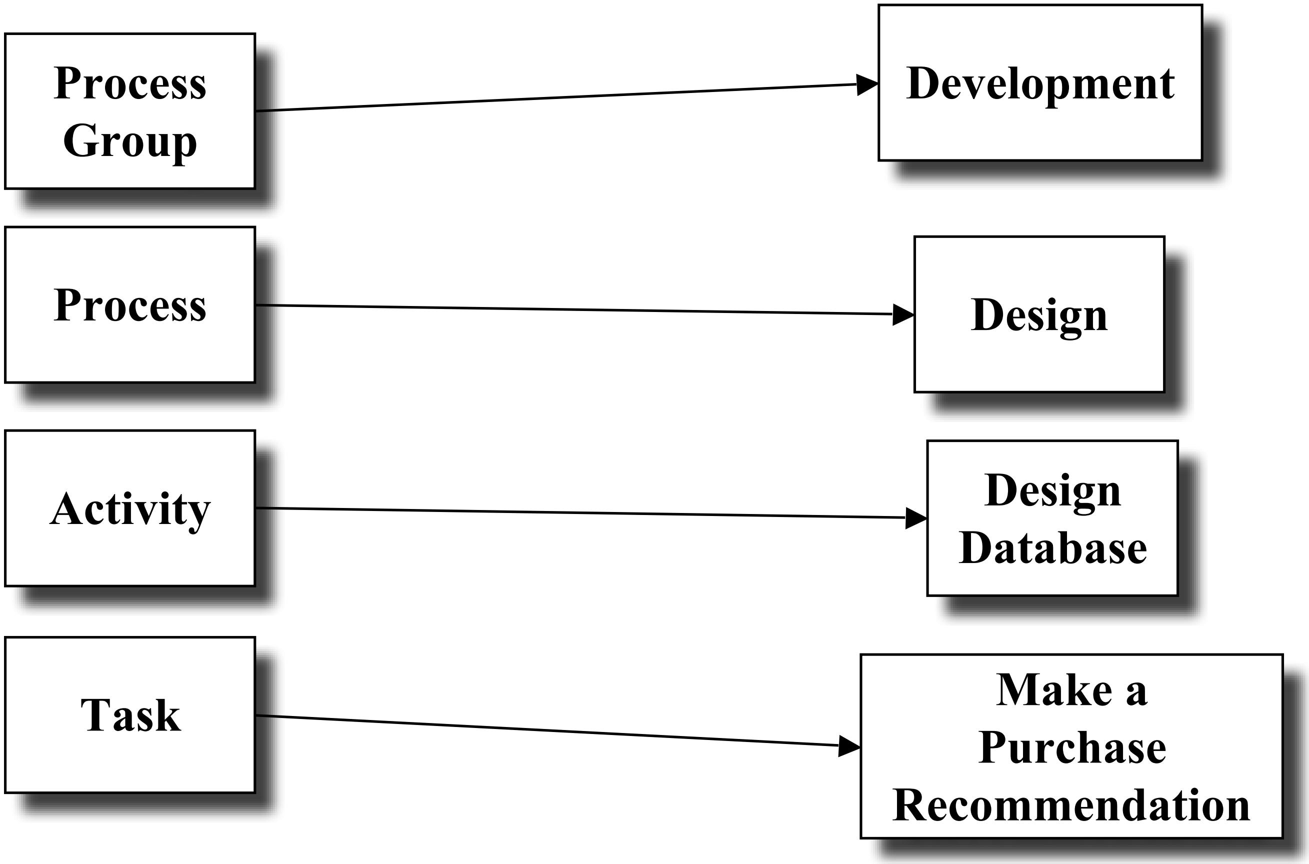
Additional and Backup Slides

Phase vs. Iteration

- *A phase* creates formal, stake-holder approved versions of artifacts ("major milestones")
  - A phase to phase transition is triggered by a business decisions
- An *iteration* creates informal, internally controlled versions of artifacts ("minor milestones")
  - Iteration to iteration transition is triggered by a specific software development activity.

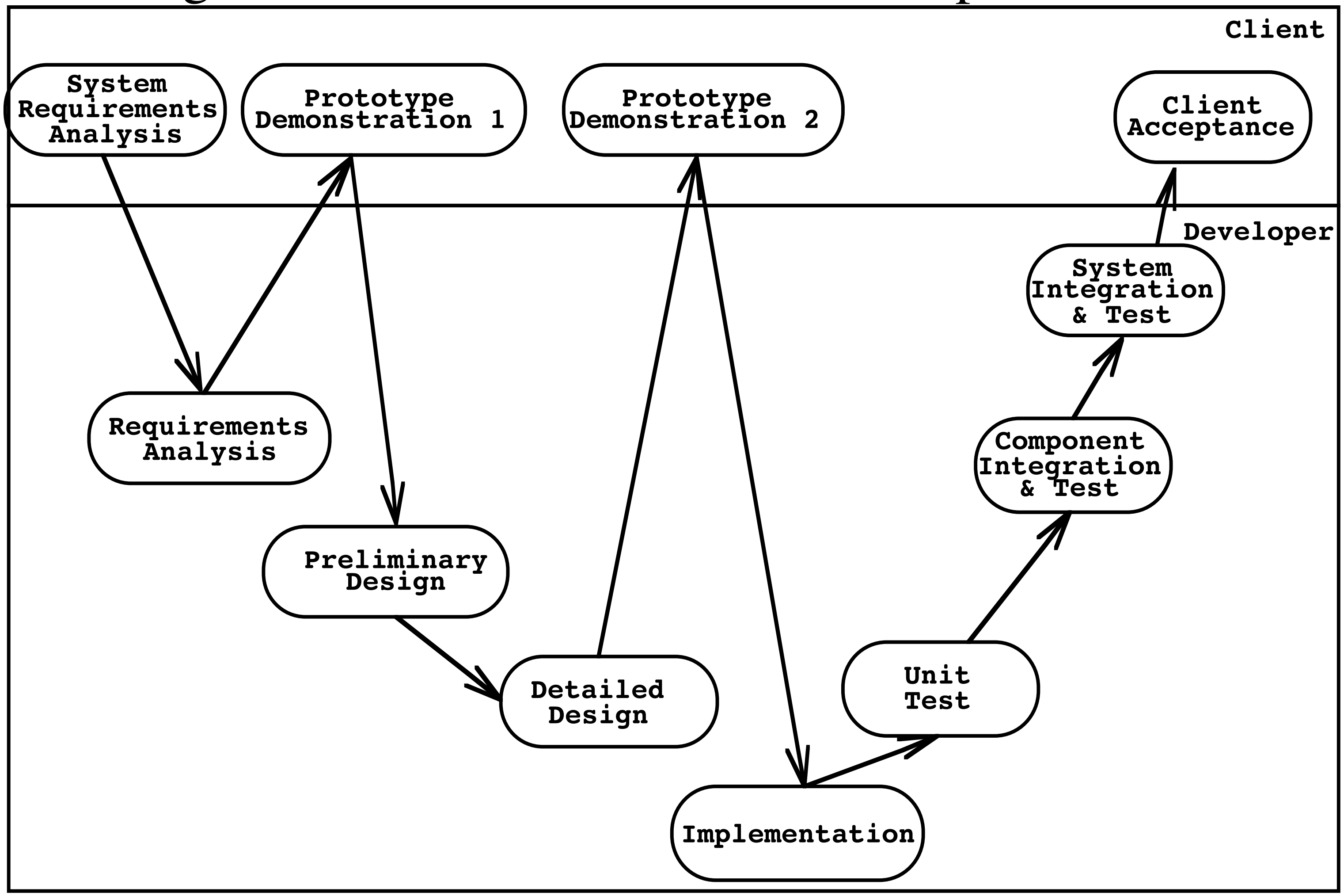
Processes, Activities and Tasks

- Process Group: Consists of a set of processes
- Process: Consists of activities
- Activity: Consists of sub activities and tasks



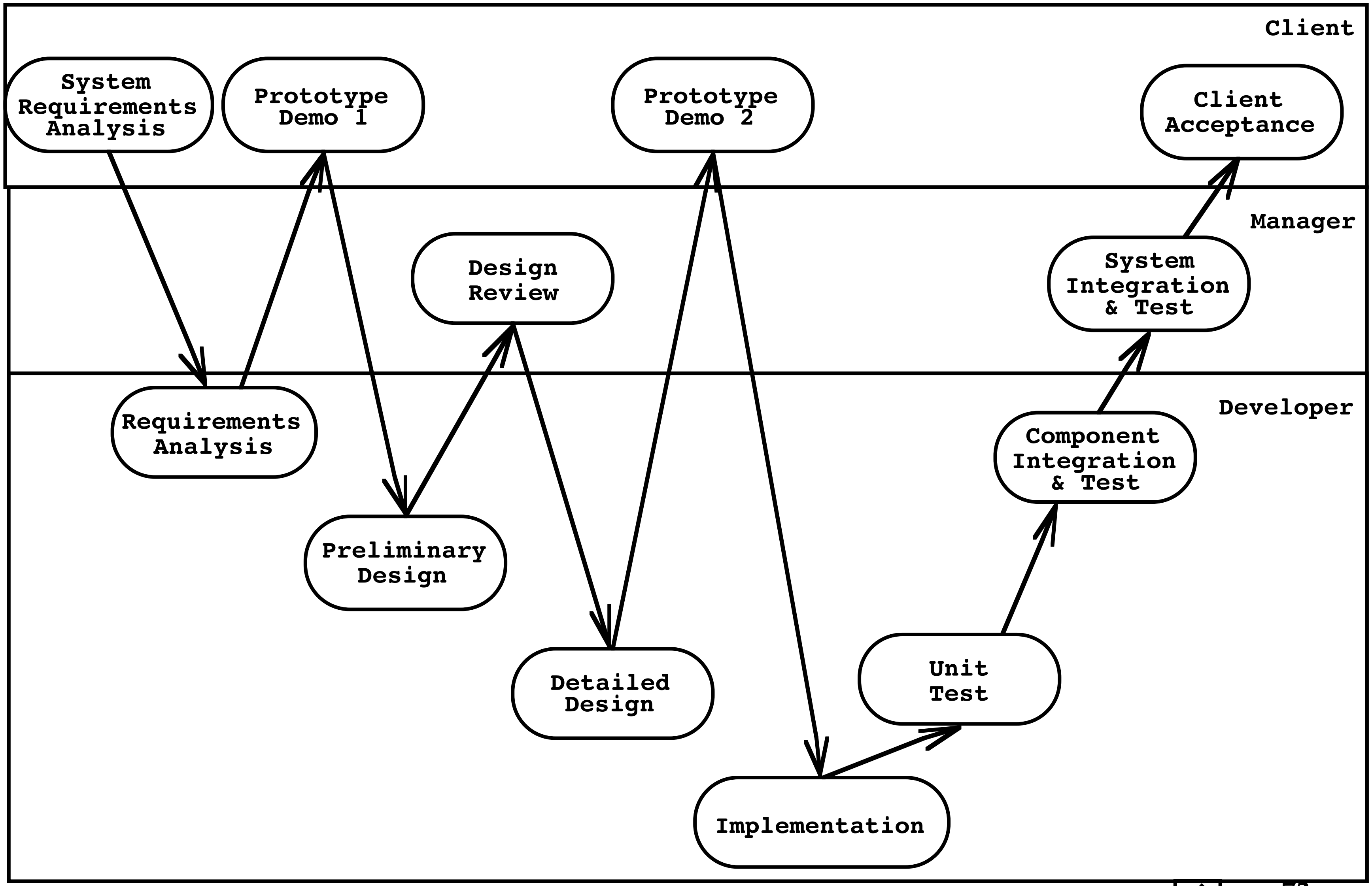
Sawtooth Model

Distinguishes between client and developers


Client
System Requirements Analysis
Prototype Demonstration 1
Prototype Demonstration 2
Client Acceptance
Developer
System Integration & Test
Requirements Analysis
Component Integration & Test
Preliminary Design
Detailed Design
Unit Test
Implementation

The Sharktooth Model

distinguishes between client, project manager and developers



“Process“ is overloaded in the Unified Process

- Meta Process (Also called “Business process”)
  - The policies, procedures and practices in an organization pursuing a software-intensive line of business.
  - Focus: Organizational improvement, long-term strategies, and return on investment (ROI)
- Macro Process (“Lifecycle Model”)
  - The set of processes in a software lifecycle and dependencies among them
  - Focus: Producing a software system within cost, schedule and quality constraints
- Micro Process
  - Techniques for achieving an artifact of the software process.
  - Focus: Intermediate baselines with adequate quality and functionality, as economically and rapidly as practical.

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 and staffing problems.

Elaboration Phase: Activities

- Elaborate the problem statement (“vision”)
  - Work 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 problems and decisions
- Identify lessons learned from the inception phase
  - Redesign the software architecture if necessary

Construction Phase: Activities

- Resource management, control and process optimization
- Complete development
- Test against evaluation criteria
- Assess releases against acceptance criteria.

Transition Phase: Activities

- All the activities of deployment-specific engineering
  - Commercial packaging and production
  - Sales rollout kit development
  - Field personnel training
- Assess deployment baselines against the acceptance criteria in the requirements set.

Inception Phase: Evaluation Criteria

- 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: Evaluation Criteria

- Apply the following questions to the results of the inception phase:
  - Is the problem statement stable?
  - Is the architecture stable?
  - Have major risk elements have been resolved?
  - Is the construction plan realizable?
  - Do all stakeholders agree that the problem solved if the current plan is executed?
  - Are the actual expenses versus planned expenses so far acceptable?

Construction Phase: Evaluation Criteria

- Apply the following questions to the results of the construction phase:
  - Is there a release *mature* enough to be deployed?
  - Is the release *stable* enough to be deployed?
  - Are the stakeholders ready to move to the transition phase?
  - Are actual expenses versus planned expenses so far acceptable?

Transition Phase: Evaluation Criteria

- Is the user satisfied?
- Are actual expenses versus planned expenses so far acceptable?

Rationale for Notations in Artifact Sets (cont'd)

- Implementation set:
  - Notation: Programming language
  - Goal: Capture the building blocks of the solution domain in human-readable format.
- Deployment set:
  - Form: Machine language
  - Goal: Capture the solution in machine-readable format.

Rationale for Notations in the Artifact Sets

- Management Set:
  - Notation: Ad hoc text, graphics, textual use cases
  - Goal: Capture plans, processes, objectives, acceptance criteria.
- Requirements set:
  - Notation: Structured text, models in UML
  - Goal: Capture problem in language of problem domain
- Design set:
  - Notation: Structured text, models in UML
  - Goal: Capture the engineering blueprints

Workflows in the Unified Process

- Management workflow
- Environment workflow
- Requirements workflow
- Design workflow
- Implementation workflow
- Assessment workflow
- Deployment workflow

Managing Projects in the Unified Process

- How should we manage the construction of software systems with the Unified Process?
  - Treat the development of a software system with the Unified Process as a set of several iterations
    - Some of these can be scheduled in parallel, others have to occur in sequence
  - Define a single project for each iteration
  - Establish work break down structures for each of the 7 workflows.

Industry Distribution across Maturity Levels (State of the Software Industry in 1995)

Maturity Level	Frequency
1 Initial	70%
2 Repeatable	15%
3 Defined	< 10%
4 Managed	< 5%
5 Optimizing	< 1%

Source: Royce, Project Management, P. 364

Insert: Types of Prototypes

- Illustrative Prototype
  - Develop the user interface with a set of storyboards
  - Implement them on a napkin or with a user interface builder (Visual Basic, Revolution...)
  - Good for first dialog with client
- Functional Prototype
  - Implement and deliver an operational system with minimum functionality
  - Then add more functionality
  - No user interface
- Exploratory Prototype ("Hack")
  - Implement part of the system to learn more about the requirements
  - Good for paradigm breaks.

Types of Prototyping

- Revolutionary Prototyping
  - Also called specification prototyping
  - Get user experience with a throw-away version to get the requirements right, then build the whole system
    - Advantage: Can be developed in a short amount of time
    - Disadvantage: Users may have to accept that features in the prototype are expensive to implement
- Evolutionary Prototyping
  - The prototype is used as the basis for the implementation of the final system
    - Advantage: Short time to market
    - Disadvantage: Can be used only if target system can be constructed in prototyping language.

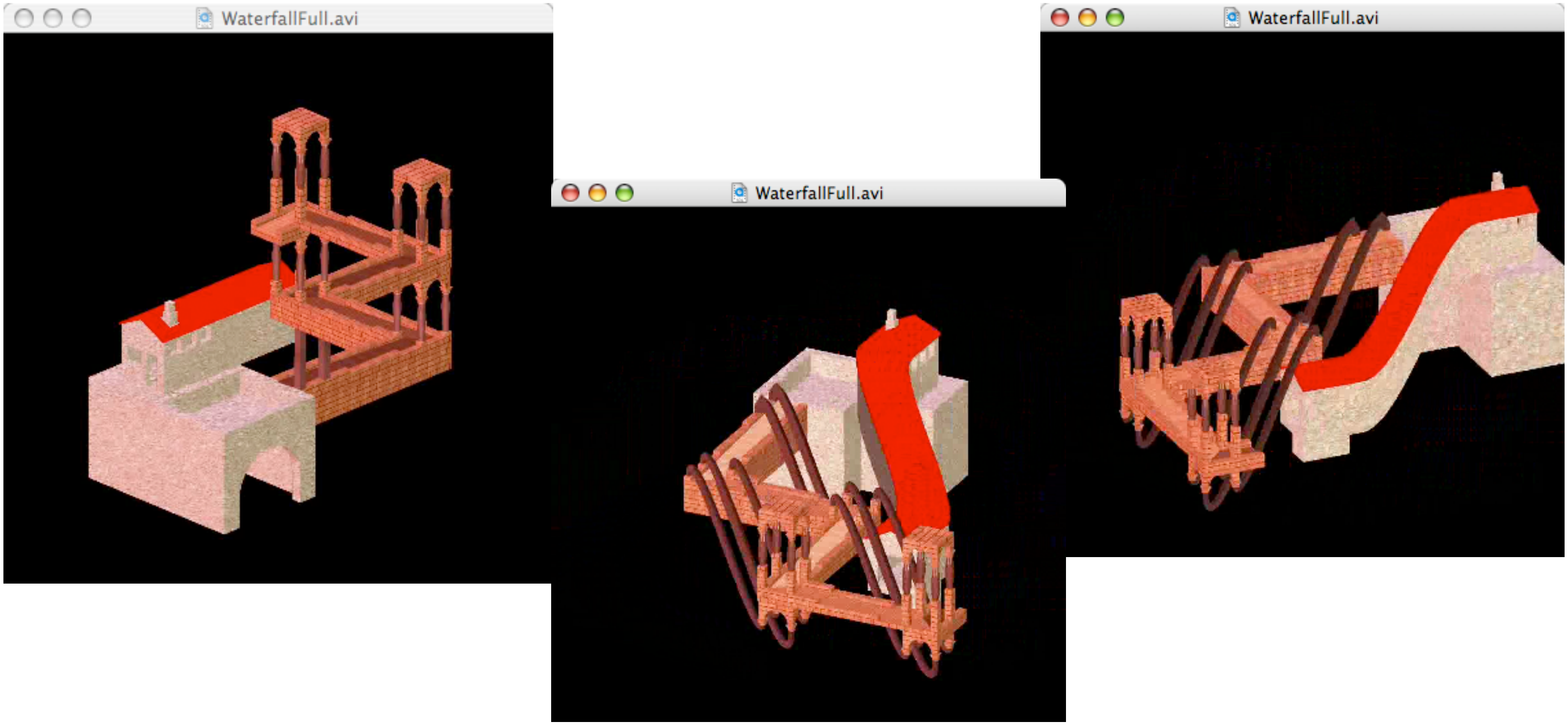
Prototyping vs Rapid Development

- Revolutionary prototyping is sometimes called *rapid prototyping*
- Rapid Prototyping is not a good term because it confuses prototyping with rapid development
  - Prototyping is a technical issue: It is a particular model of development used in a life cycle process
  - Rapid development is a management issue: It is a particular way to control a project
- Prototyping can go on forever, if it is not restricted:
    - "Time-boxed prototyping" limits the duration of the prototype development.

References

- Readings used for this lecture
    - [Bruegge-Dutoit] Chapter 12
    - [Humphrey 1989] Watts Humphrey, Managing the Software Process, SEI Series in Software Engineering, Addison Wesley, ISBN 0-201-18095-2
- Additional References
    - [Royce 1970] Winston Royce, Managing the Development of Large Software Systems, Proceedings of the IEEE WESCON, August 1970, pp. 1-9
    - SEI Maturity Questionaire, Appendix E.3 in [Royce 1998], Walker Royce, **Software Project Management, Addison-Wesley, ISBN0-201-30958-0**

Movie of Escher's Waterfall Model

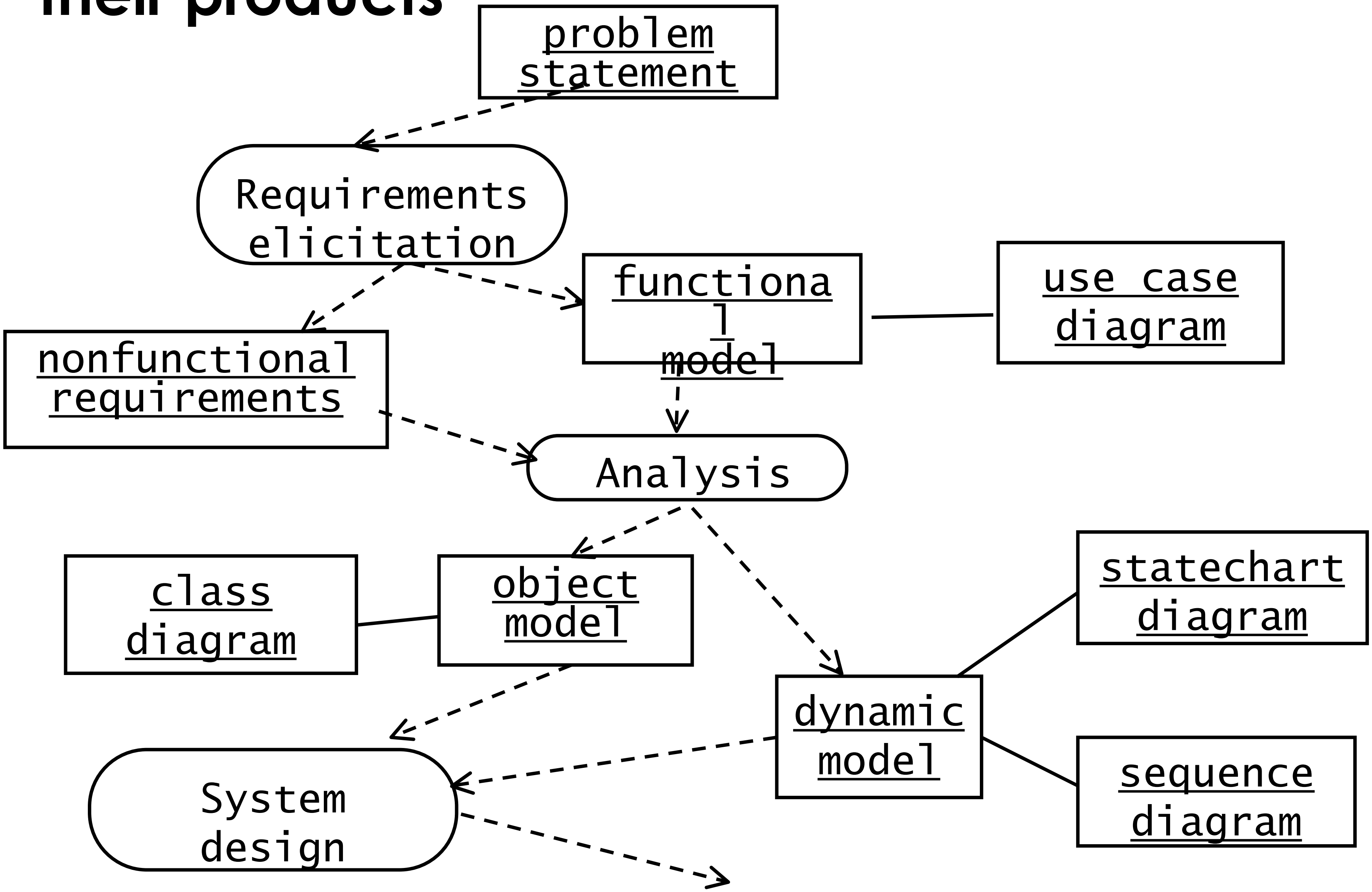


Escher for Real

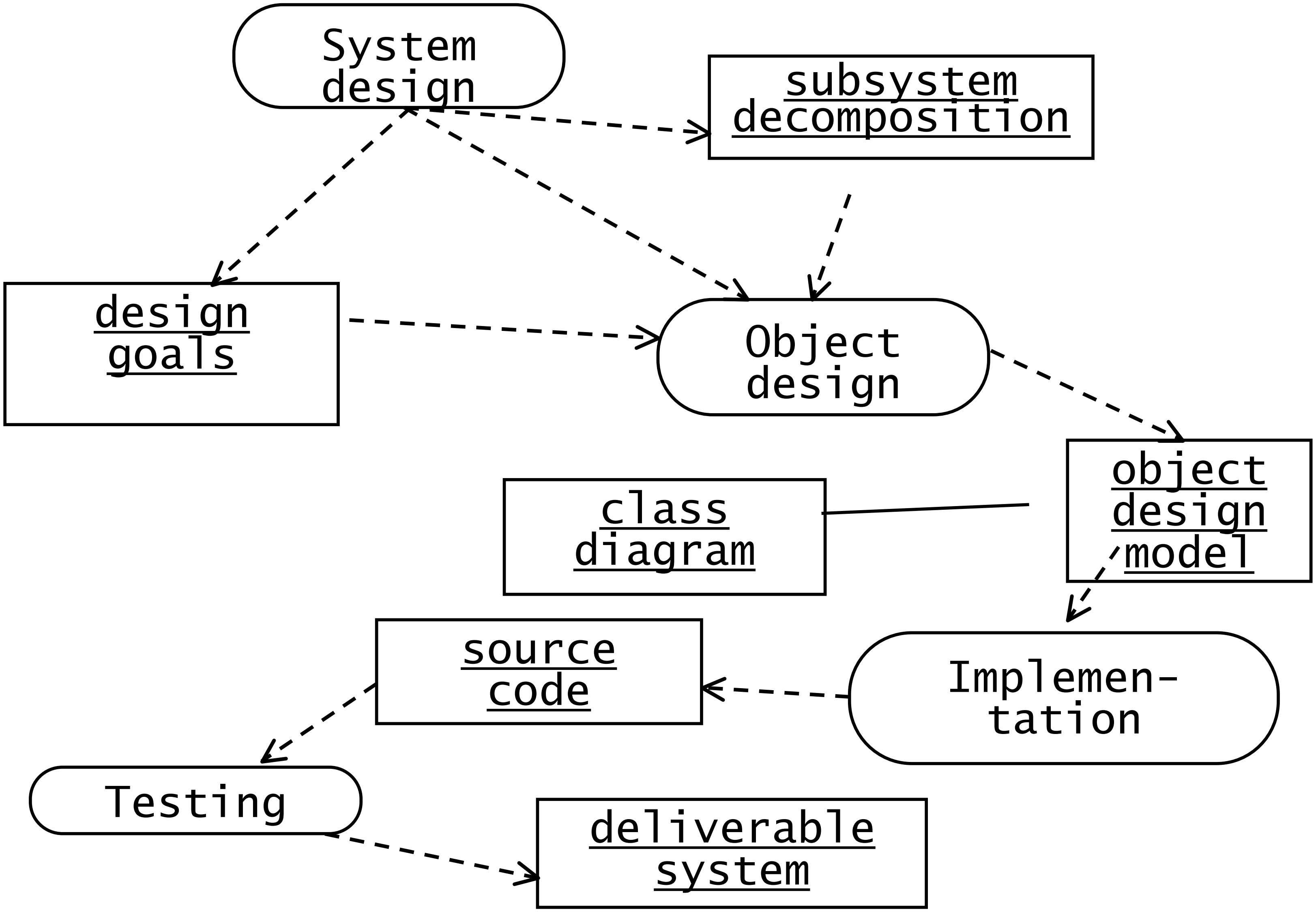
http://www.cs.technion.ac.il/~gershon/EscherForRealWaterfallFull.avi

(C) Copyright 2002-5 Gershon Elber,Computer Science Department, Technion

OOSE-Book: Development activities and their products



OOSE- Development activities (cont'd)


System design
subsystem decomposition
design goals
Object design
object design model
class diagram
source code
Implemen-tation
Testing
deliverable system