Software Engineering I: Software Technology

WS 2008

Software Lifecycle Models

Bernd Bruegge

Chair for Applied Software Engineering

Technische Universitaet Muenchen

Outline of Lecture: Today and Friday

- Announcements
- Modeling the software life cycle
- Sequential models
  - Pure waterfall model, V-model
- Iterative models
  - Boehm's spiral model, Unified Process
- Entity-oriented models
  - Issue-based models and agile models.

Today

Friday

Announcements

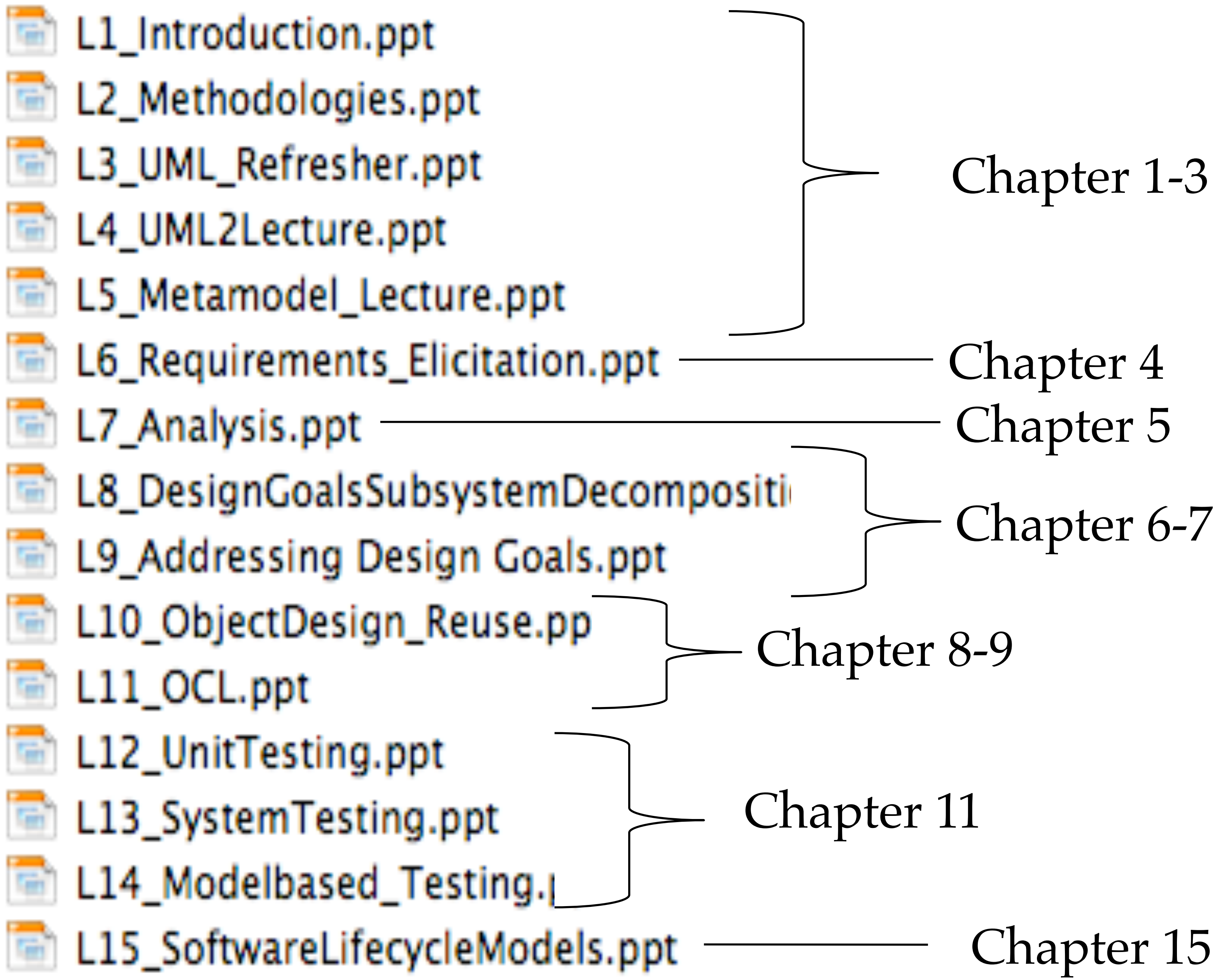
- Lecture Evaluation
- Lecture Schedule for the remaining semester.

Lecture Evaluation

- **Positive:**
  - Real-World Speakers, Praxisanbindung, hoher Praxisbezug, Flughafen-Präsentationen
  - Art der Vorlesungspräsentation angenehm
- **Negative:**
  - Mid-term klausur: first open book,then closed book, then not relevant,
  - Schwachsinnige Klausur-Regelung
  - Musterlösungen nicht rechtzeitig
  - I don't think interactive exercises don't work well with this kind of material and number of students
  - Kein Konzept hinter den Übungen
  - Schwer den Gesamtüberblick zu behalten

Lecture Evaluation (2)

- Your Suggestions:
  - Less Slides
  - One continuous project in the exercises
  - Title of the slides should be related to the table of contents with the book
- Our Suggestions:
  - Become a tutor for my lecture in the summer 2009
  - Great chance to improve software exercises
  - Great way to learn project management hands-on



L1_Introduction.ppt

L2_Methodologies.ppt

L3_UML_Refresher.ppt

L4_UML2Lecture.ppt

L5_Metamodel_Lecture.ppt

Chapter 1-3

L6_Requirements_Elicitation.ppt — Chapter 4

L7_Analysis.ppt — Chapter 5

L8_DesignGoalsSubsystemDecompositi

L9_Addressing Design Goals.ppt

Chapter 6-7

L10_ObjectDesign_Reuse.pp

L11_OCL.ppt

Chapter 8-9

L12_UnitTesting.ppt

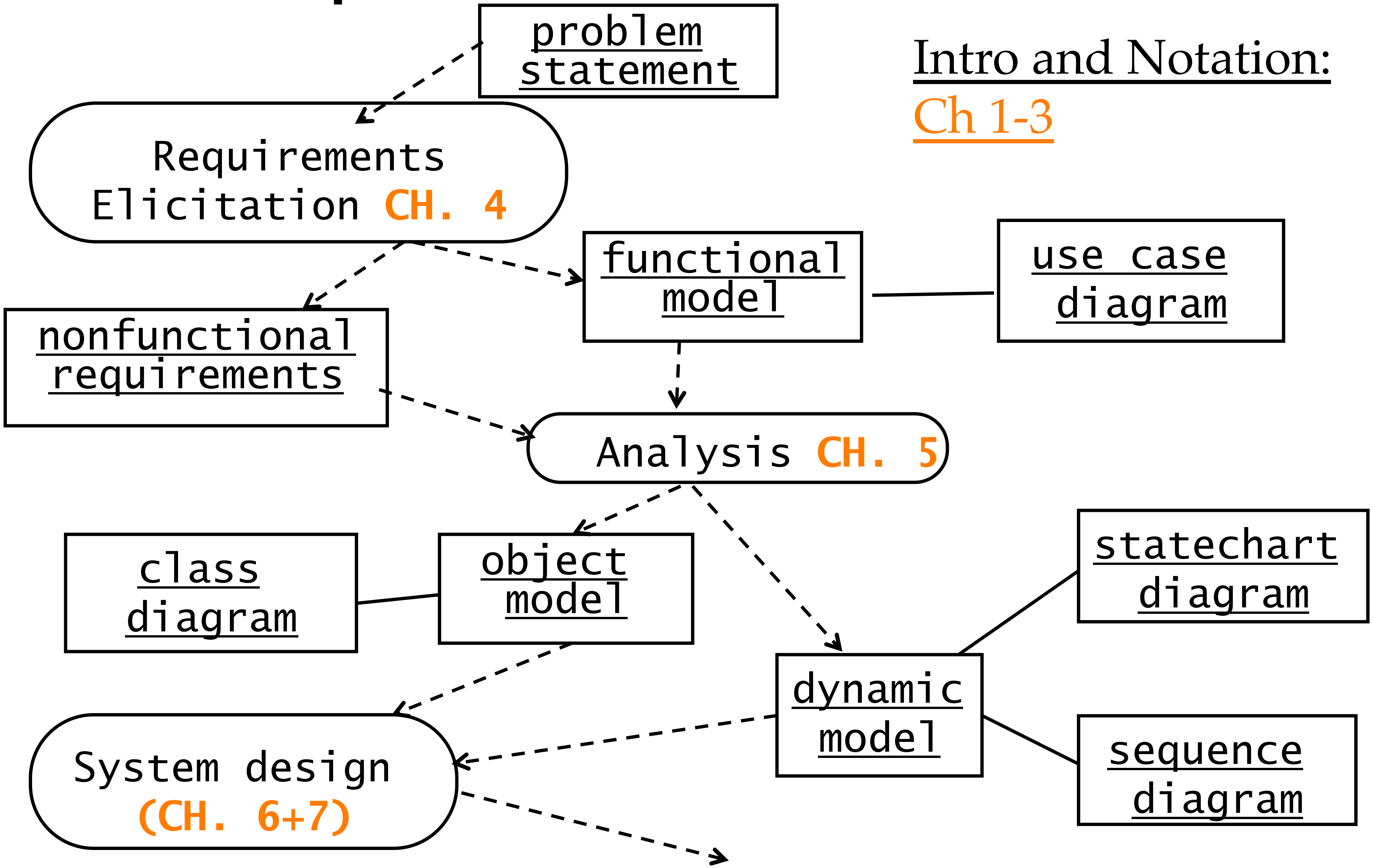
L13_SystemTesting.ppt

L14_Modelbased_Testing.

Chapter 11

L15_SoftwareLifecycleModels.ppt — Chapter 15

OOSE Development Activities: Relationship Book Chapters and Lecture Slides



problem statement

Requirements Elicitation CH. 4

functional model

use case diagram

nonfunctional requirements

Analysis CH. 5

class diagram

object model

statechart diagram

dynamic model

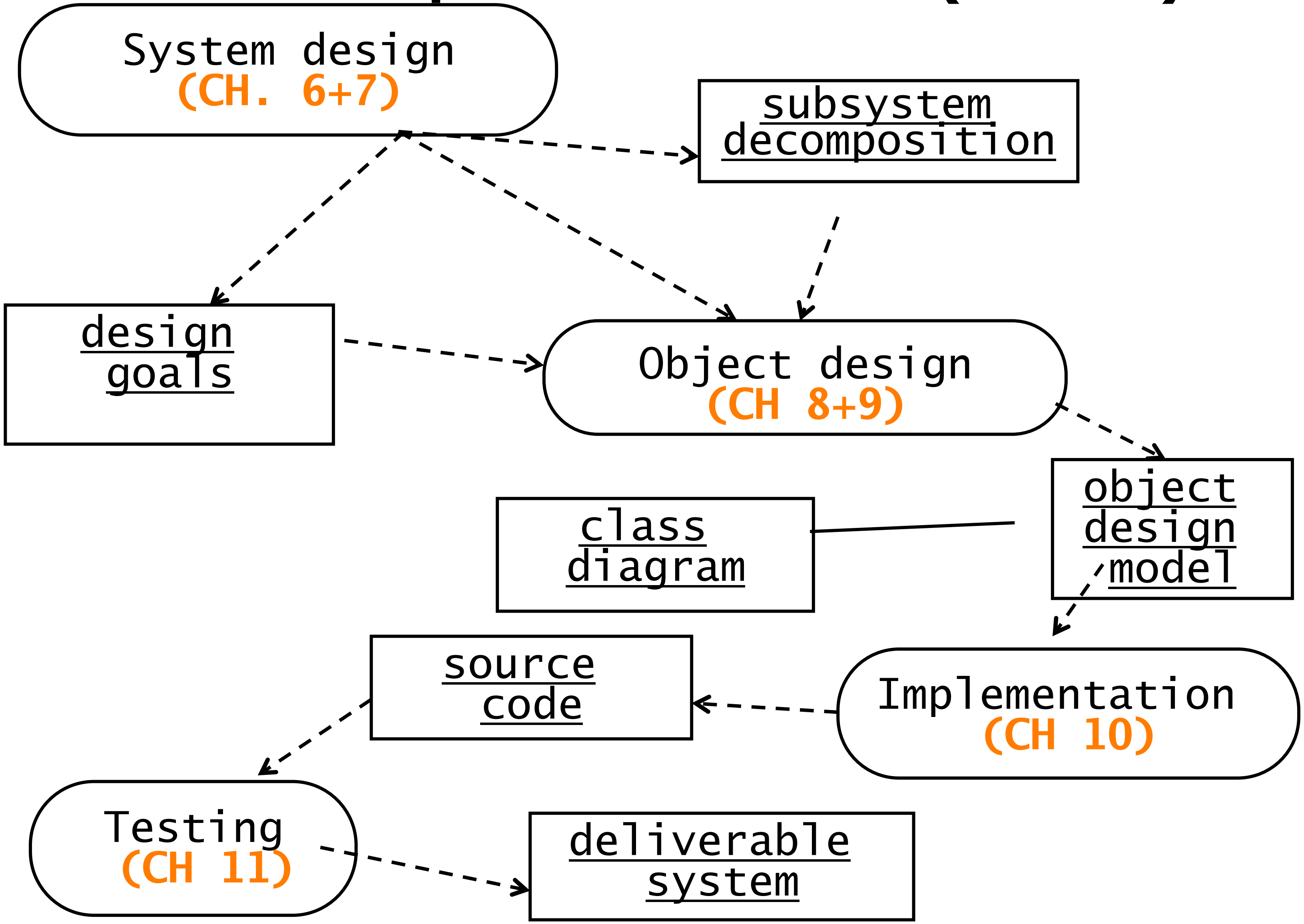
sequence diagram

System design (CH. 6+7)

Intro and Notation:

Ch 1-3

OOSE- Development activities (cont'd)



Remaining Class Schedule

- **Jan 13 and Jan 16: Software Lifecycle (**Ch 15)
- **Jan 20, 16:15-17:45**
  - Build and Release Management, Configuration Management (Ch 12)
- **Jan 23:** No class
- **Jan 27, 16:15-17:45** Invited Lecture, Rolf Schumann, Better Place Inc., "Software Requirements for Green Technologies"
- **Jan 30, 9:15-10:00:** Methodologies II (Ch 16)
- **Feb 3, 16:15-17:45**: Invited Lecture, Klaus Eberhardt, iteratec GmbH, „Why Projects Fail"
- **Feb 5, 18:00-19:30:** Final Exam,
  Location: Maschinenwesen 0001

What about Chapters 12 and Chapter 15?

- Rationale Management and Project Management will be covered in another lecture in the summer
- Software Engineering II: Project Organization and Management (“POM”, Module IN2083)
  - Elective (“Wahlpflichtfach”) for Diplom students, 3rd level module for master students
  - 2V + 2 Ü
  - Accompanied with a continous project throughout the lectures
  - http://drehscheibe.in.tum.de/myintum/kurs_verwaltung/cm.html?id=IN2083
- See also
  - http://www.in.tum.de/fuer-studierende-der-tum/module-und-veranstaltungen/vorschau-veranstaltungen.html

Outline of Lecture: Today and Friday

- ✓ Announcements
- Modeling the software life cycle
- Sequential models
  - Pure waterfall model, V-model
- Iterative models
  - Boehm’s spiral model, Unified Process
- Entity-oriented models
  - Issue-based models and agile models.

<u>Today</u>

<u>Friday</u>

Definitions

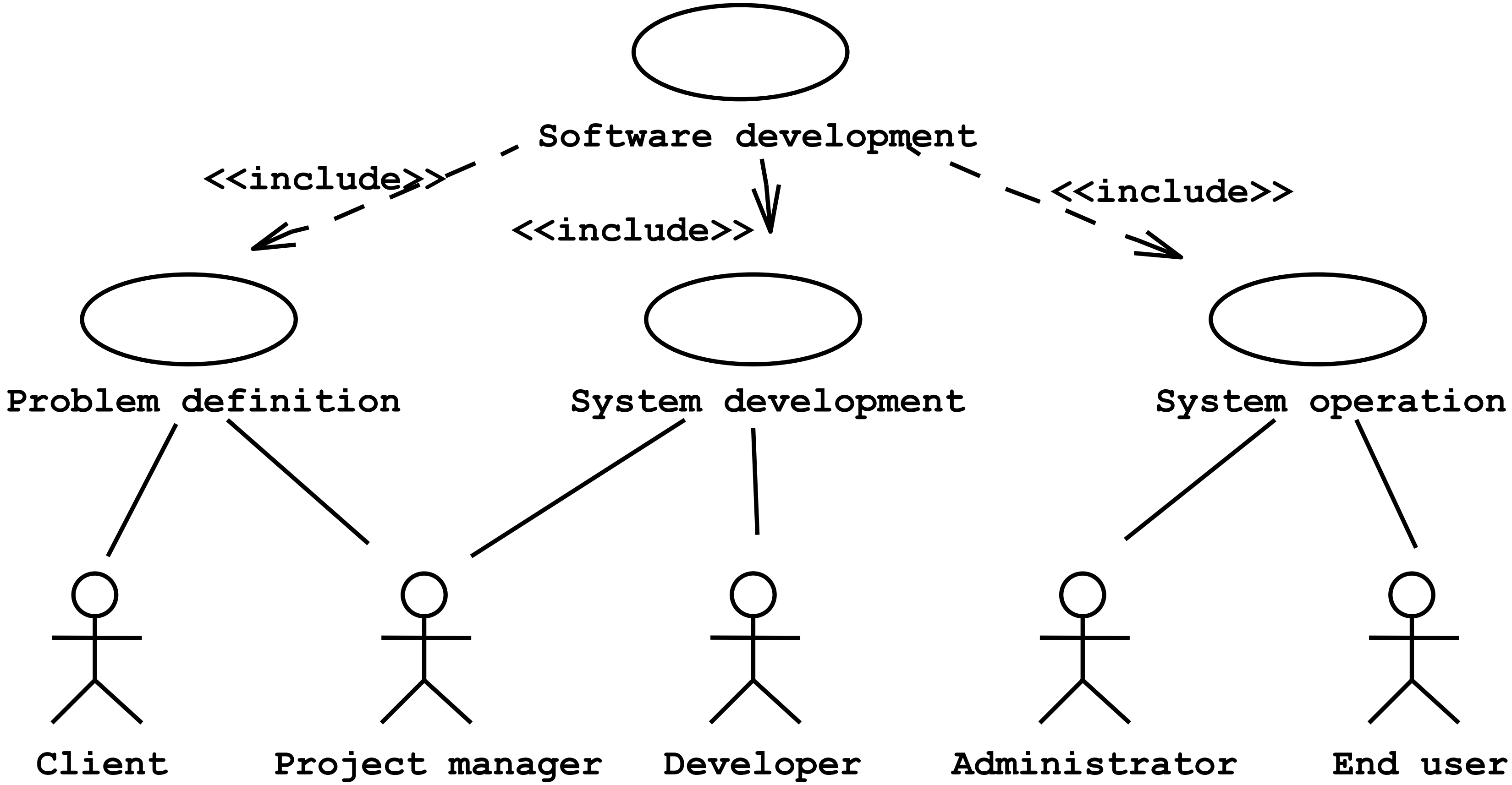
- Software life cycle
  - Set of activities and their relationships to each other to support the development of a software system
- Software development methodology
  - A collection of techniques for building models applied across a software life cycle
  - It also specifies what to do, when something is *missing* or things go *wrong*.

Typical Software Life Cycle Questions

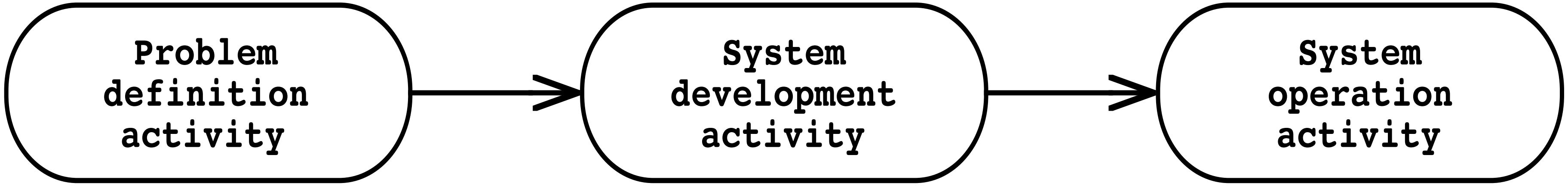
- *Which activities* should we select for the software project?
- What are the *dependencies between activities*?
- How should we *schedule the activities?*
- To find these activities and dependencies we can use the same modeling techniques we use for software development:
  - ⇨ Functional model of a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

These questions are also crucial for the design of a lecture
Slide 7 + 8 present a dynamic model of SE I☺

Functional Model of a simple Life Cycle Model

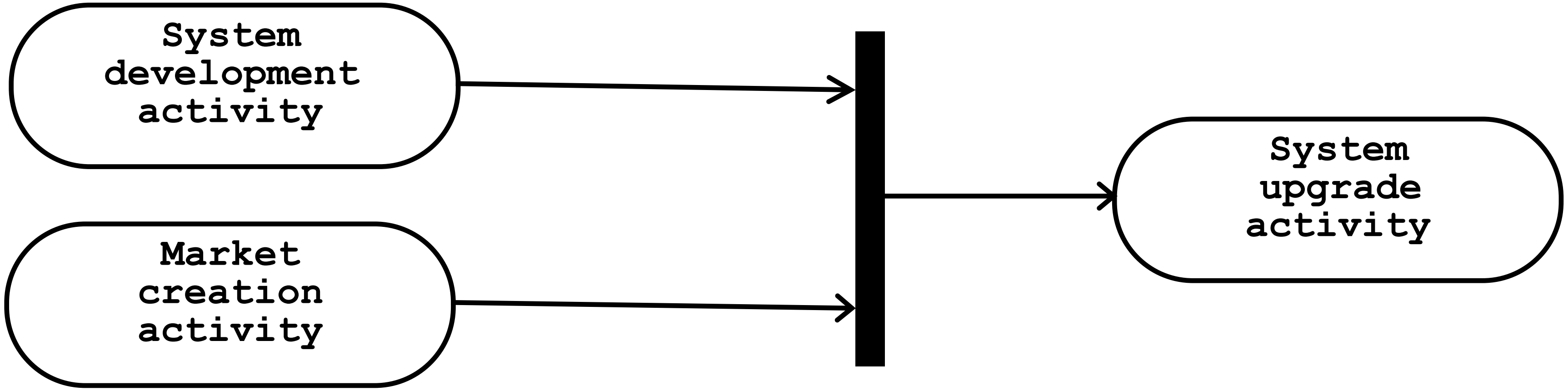


Activity Diagram for the same Life Cycle Model


Problem definition activity
System development activity
System operation activity

Interpretation:
Software development goes through a linear progression of states called `Problem definition activity`, `System development activity` and System operation activity.

Another Life Cycle Model



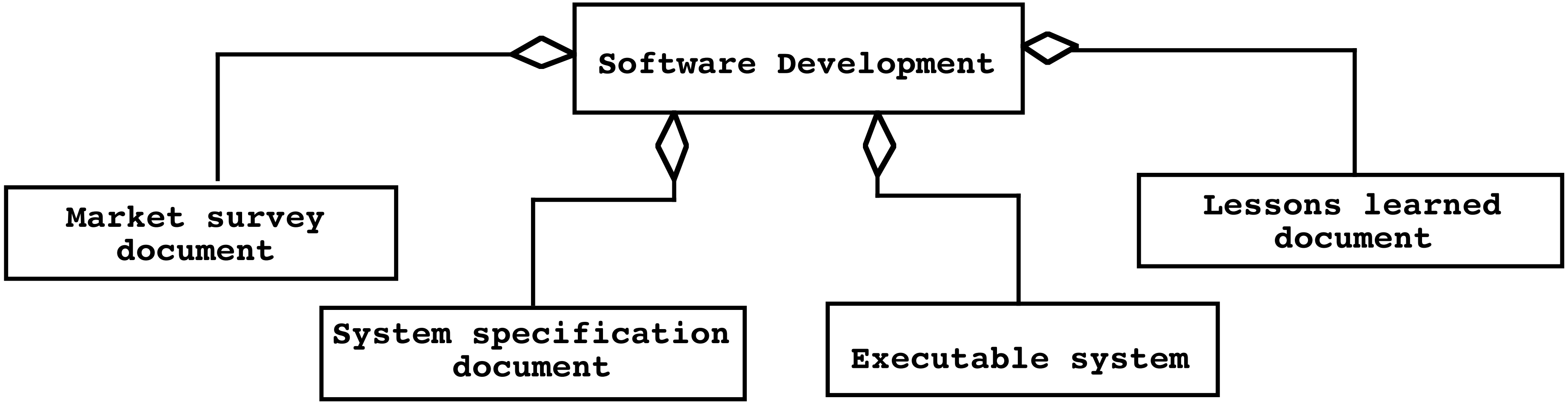
Interpretation:

`System development` and `Market creation` can be done in parallel. They must be finished before the `System upgrade` activity can start.

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.

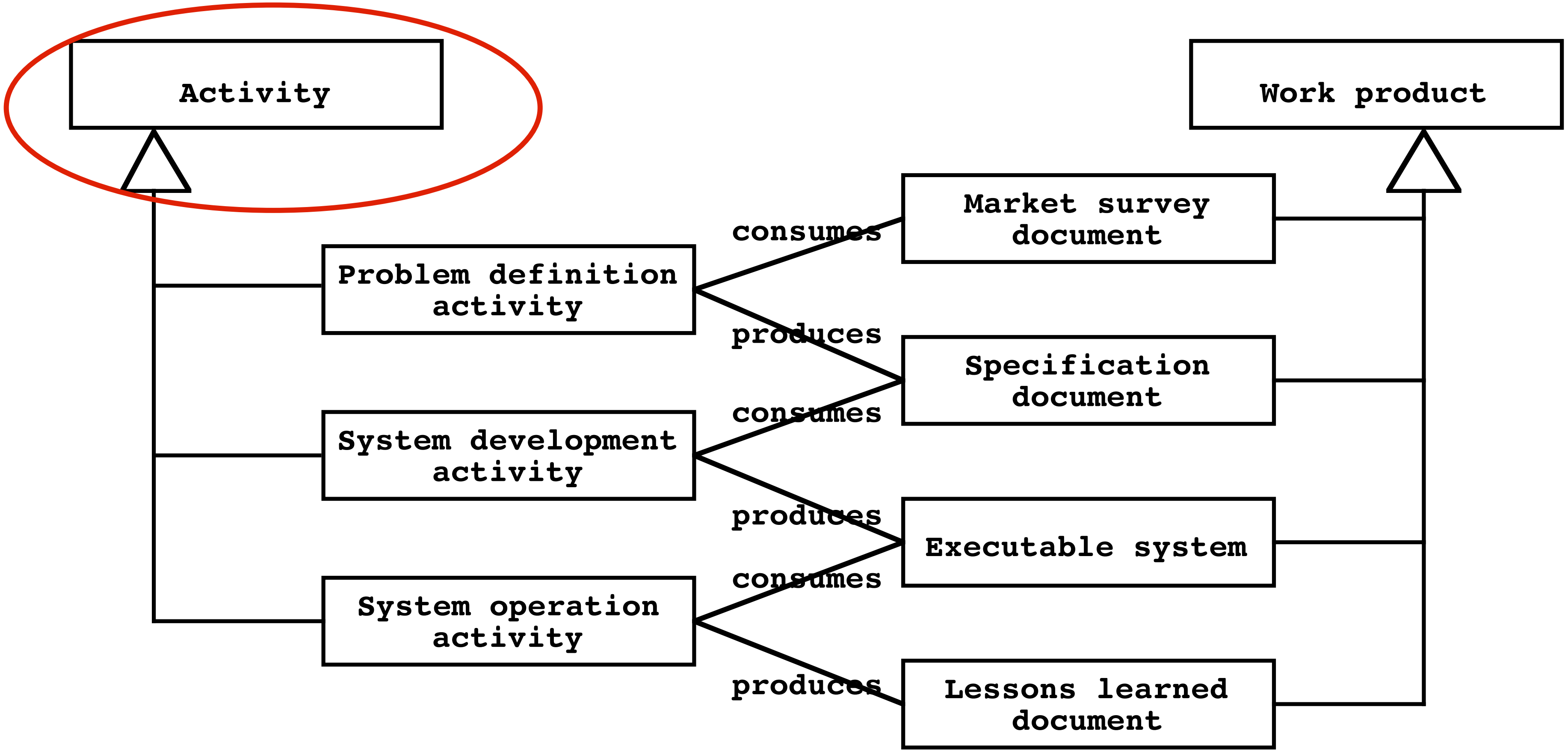
Entity-centered view of Software Development



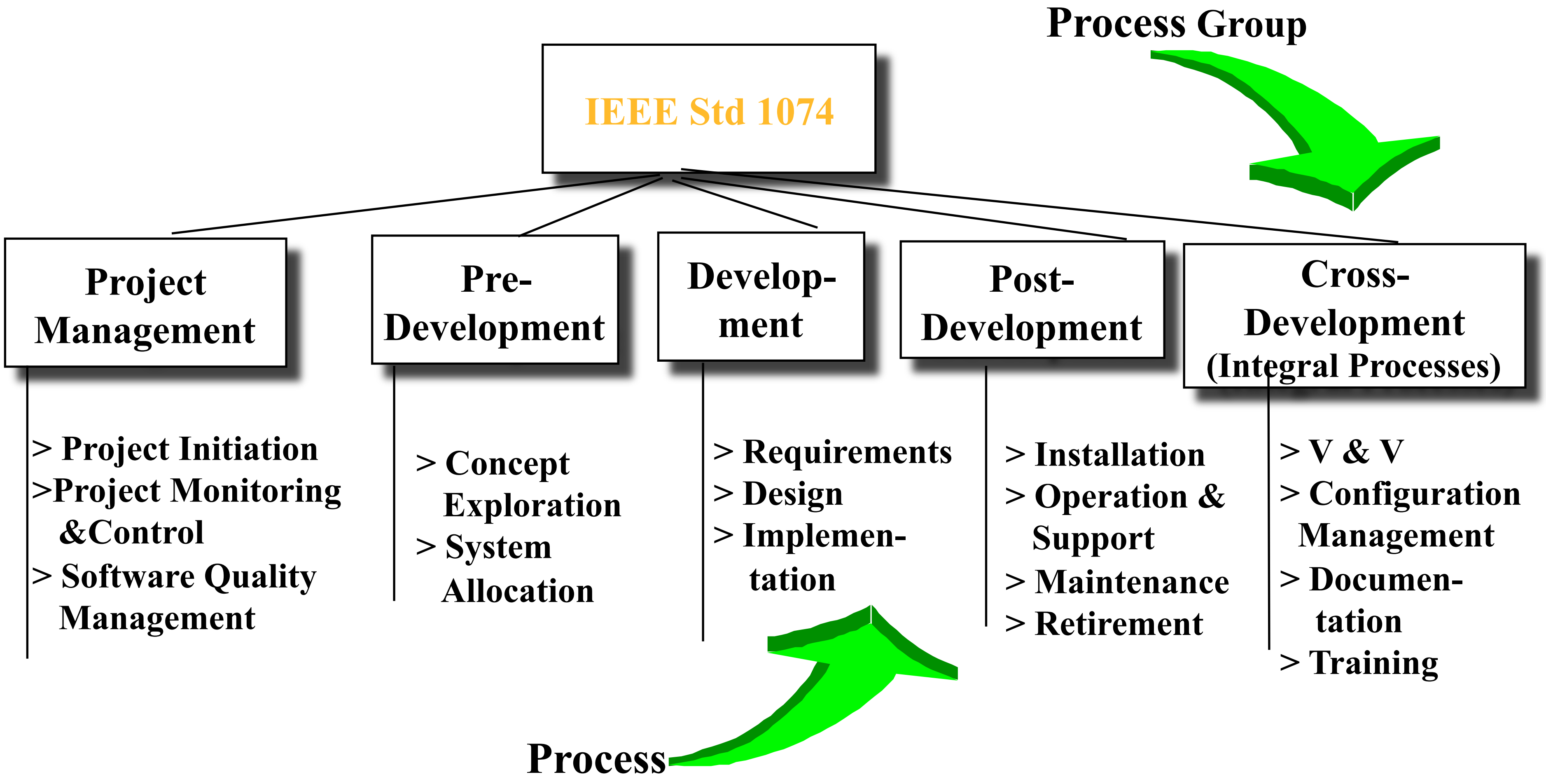
Interpretation:

Software development *consists of* the creation of a set of deliverables: `Market survey document, System specification document, Executable system, Lessons learned document.`

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

IEEE

IEEE: Institute for Electrical and Electronics Engineers ("I-triple-e")

- Founded in 1963, initial focus on telephone,radio, electronics, http://www.ieee.org/portal/site
- Largest subgroup with 100,000 members: IEEE Computer Society, founded in 1971
  - "Computer Magazine", Transactions, eg. "Transactions on Software Engineering"
- Largest standards-making organization in the world
- Well-known examples: IEEE 802.3 and IEEE 802.11
  - IEEE 802.3 Ethernet
  - IEEE 802.11 Wireless LAN, also called WiFi
    - 802.11b, 802.11g, 802.11n
    - 2.4-5 GHz, 11 Mbit/s, 54 Mbit/s, 248 Mbit/s.

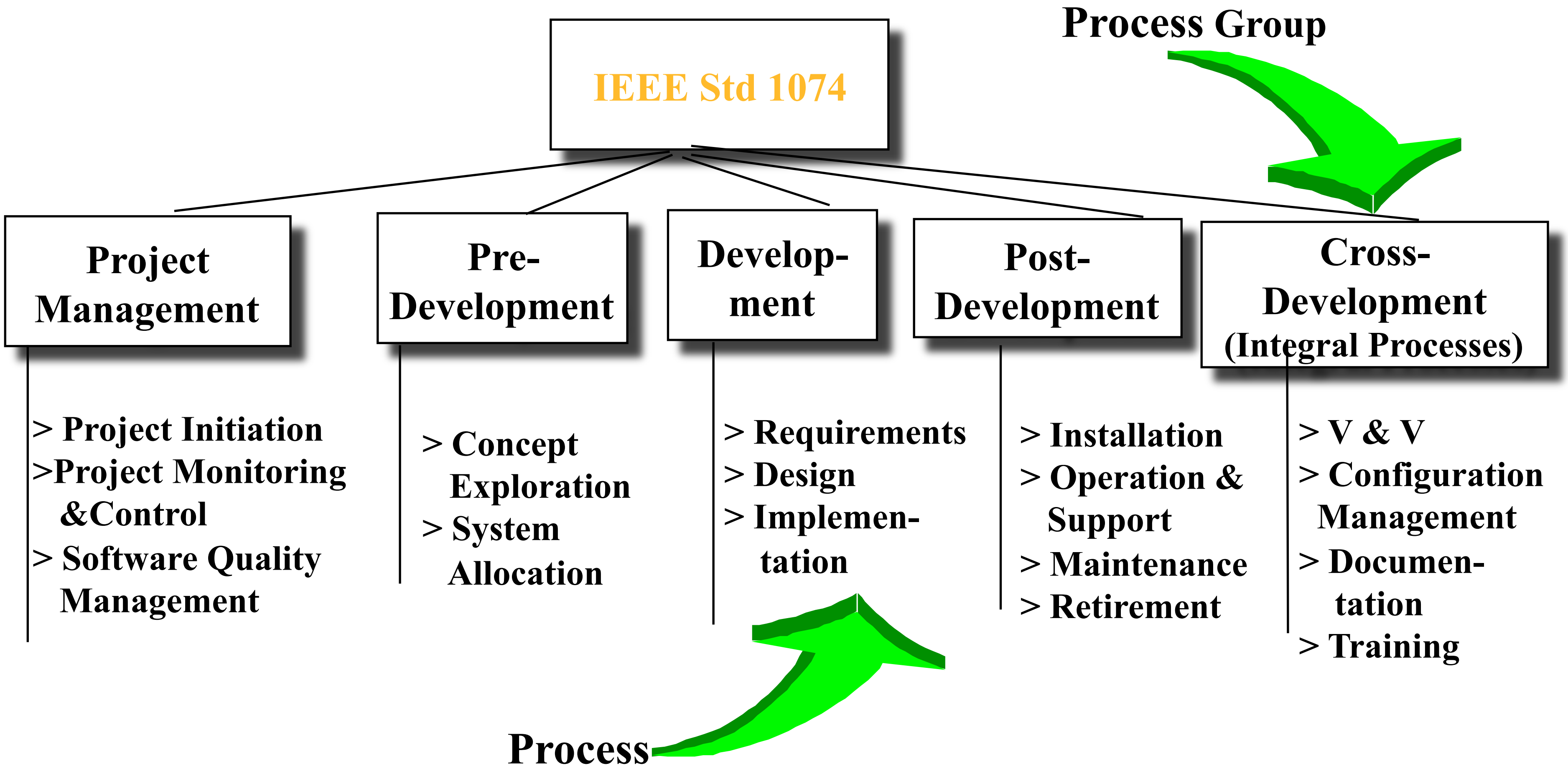
ACM

- Association for Computing Machinery
- Founded in 1947
- 80,000 members
- Web Portal: http://www.acm.org/
- Organized in local chapters and special interest groups
- There are even student chapters
  - You can start one here at TUM!
    - http://www.acm.org/chapters/stu/
- Main publication:
  - Communications of the ACM, short CACM
- Digital Library
  - http://portal.acm.org/dl.cfm

GI

- Gesellschaft für Informatik
  - Supports computer science in research, education and applications
- Founded in 1969, 24,500 members (2,500 students)
- Website: http://www.gi-ev.de/
- Digital Library:
  - http://www.gi-ev.de/service/digitale-bibliotheken/io-port/
  - Also access to IEEE digital library
  - http://www.gi-ev.de/service/digitale-bibliotheken/ieee/
- Interesting conference: Software Engineering 2009
  - In Kaiserslautern http://www.se2009.de/
  - The last one was in Munich: http://se2008.in.tum.de
  - Videos of Key Lectures: http://se2008.in.tum.de/videos-se-2008.html

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

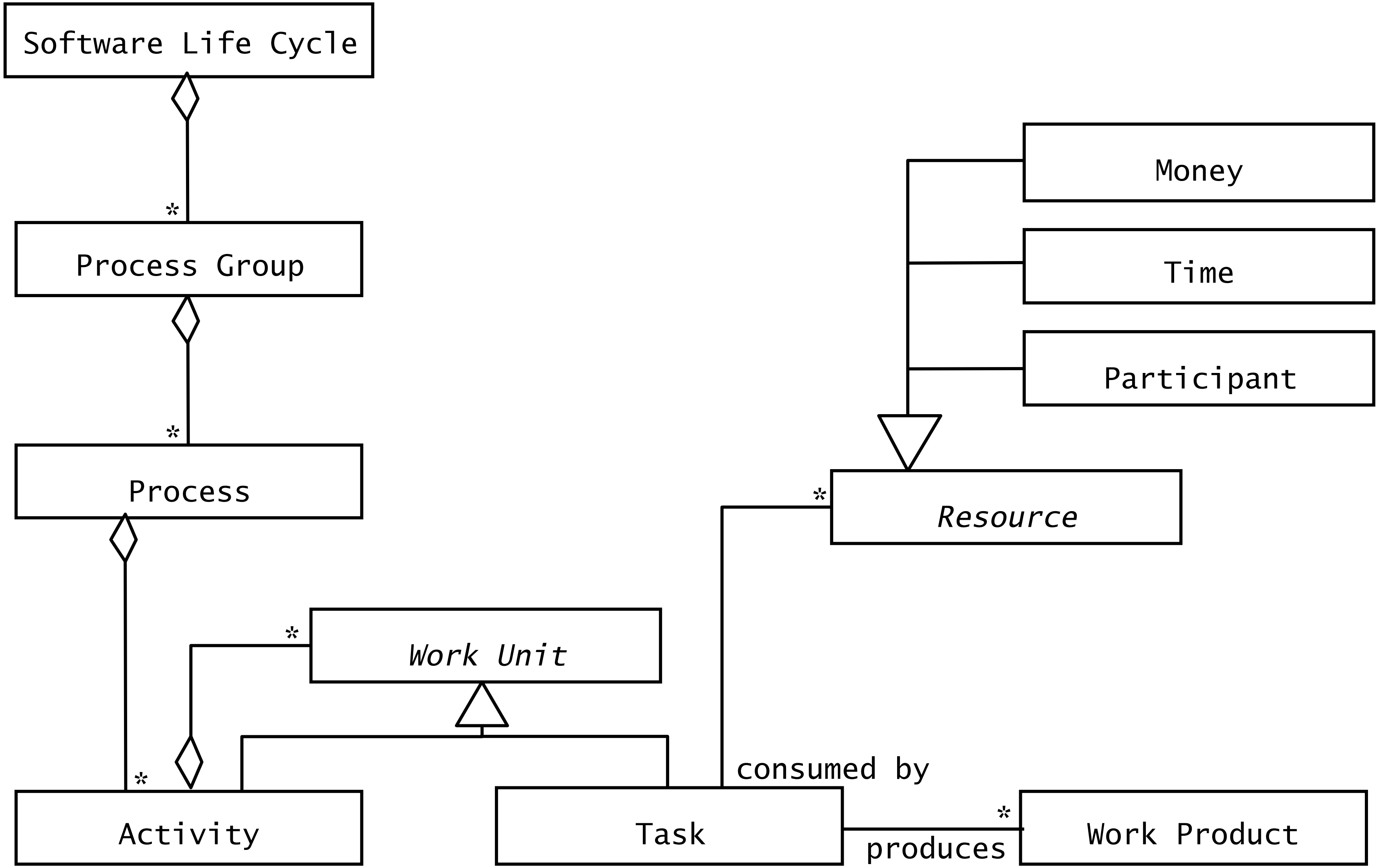
- > Installation
- > Operation & Support
- > Maintenance
- > Retirement

Cross-Development (Integral Processes)

- > V & V
- > Configuration Management
- > Documen-tation
- > Training

Process

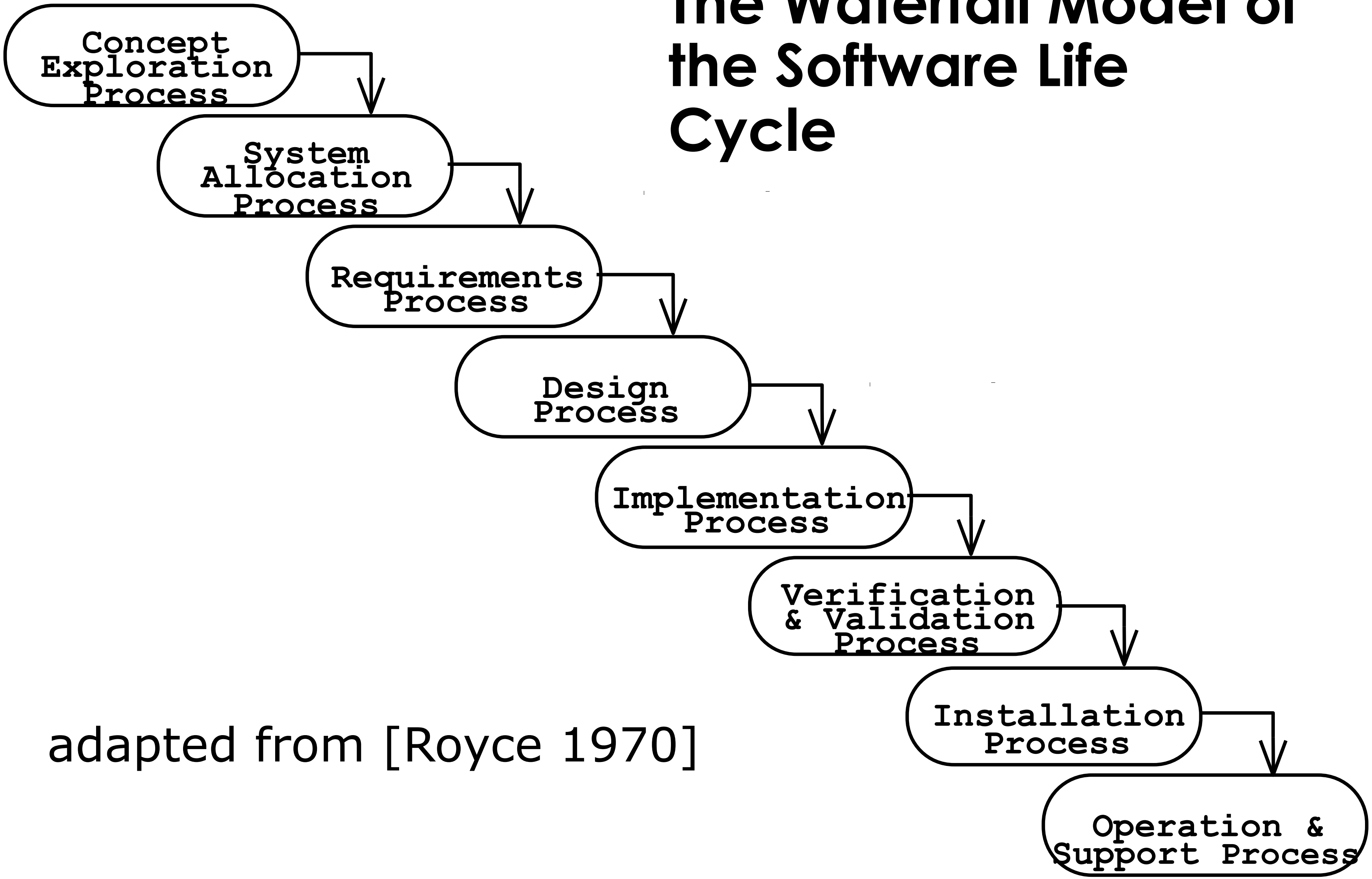
Object Model of the IEEE 1074 Standard



Life Cycle Modeling

- Many models have been proposed to deal with the problems of defining activities and associating them with each other
  - The waterfall model, 1970
  - V-Model, 1992, 1997
  - Spiral model, 1988
  - Rational process, 1996
  - Unified process, 1999
  - Agile models, 1999
  - V-Model XT, 2003
  - Open Unified Process (Part of the Eclipse Process Framework, open source project)
  - SPEM Software Process Engineering Meta-Model 2.0, 2008

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

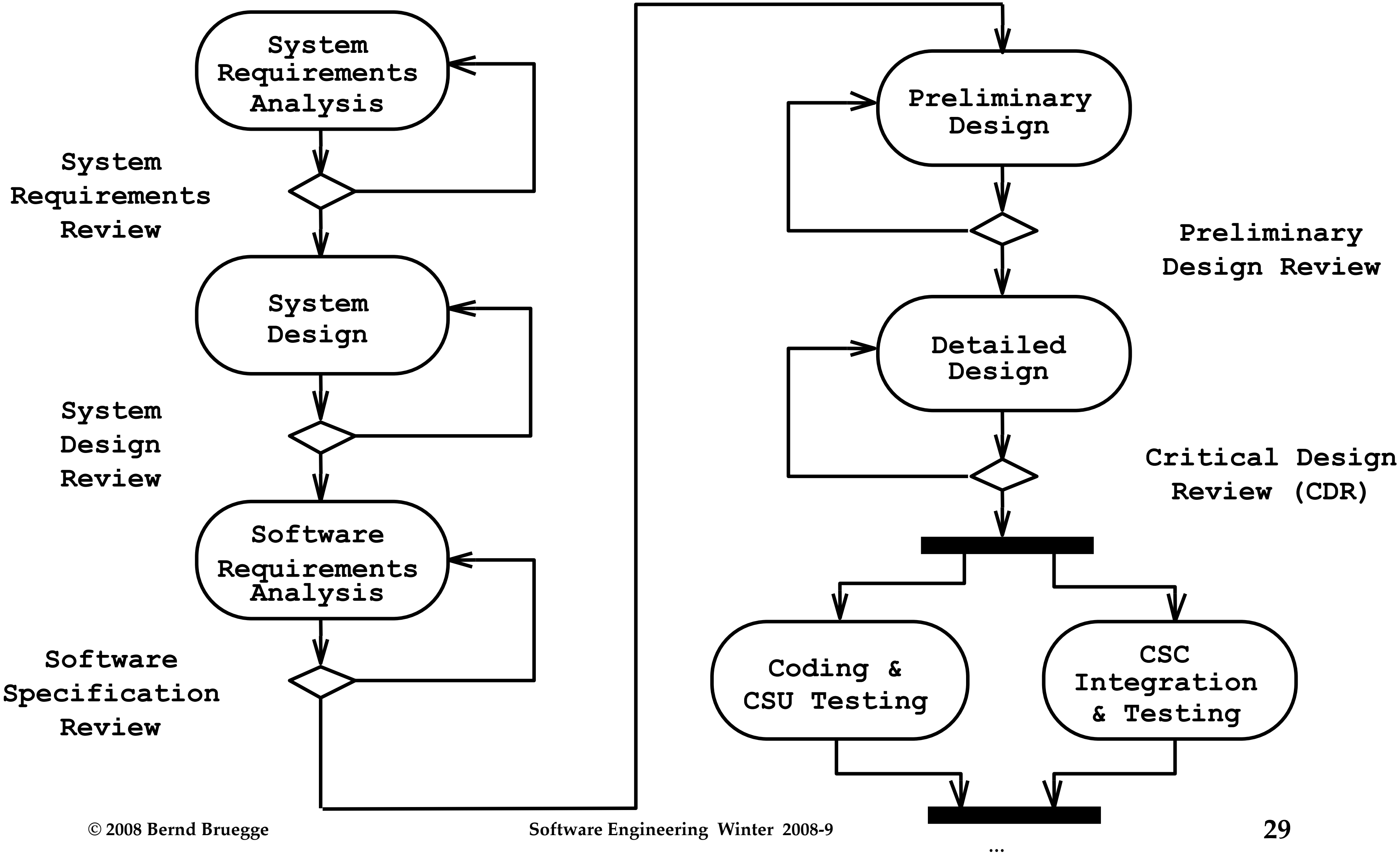
Operation & Support Process

adapted from [Royce 1970]

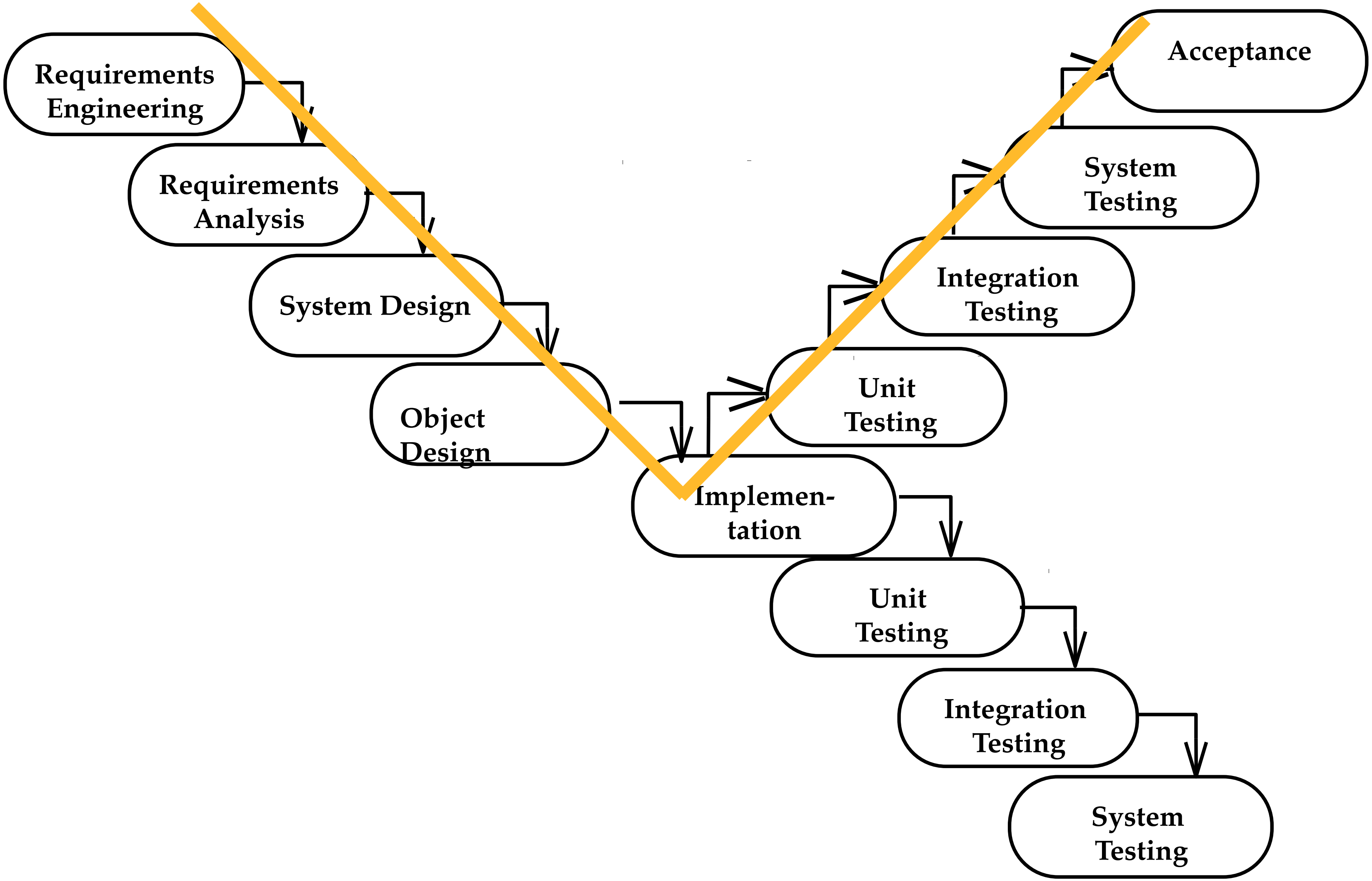
Example of a Waterfall Modell DOD Standard 2167A

- 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 (DOD) for all software contractors in the 1980-90’s.

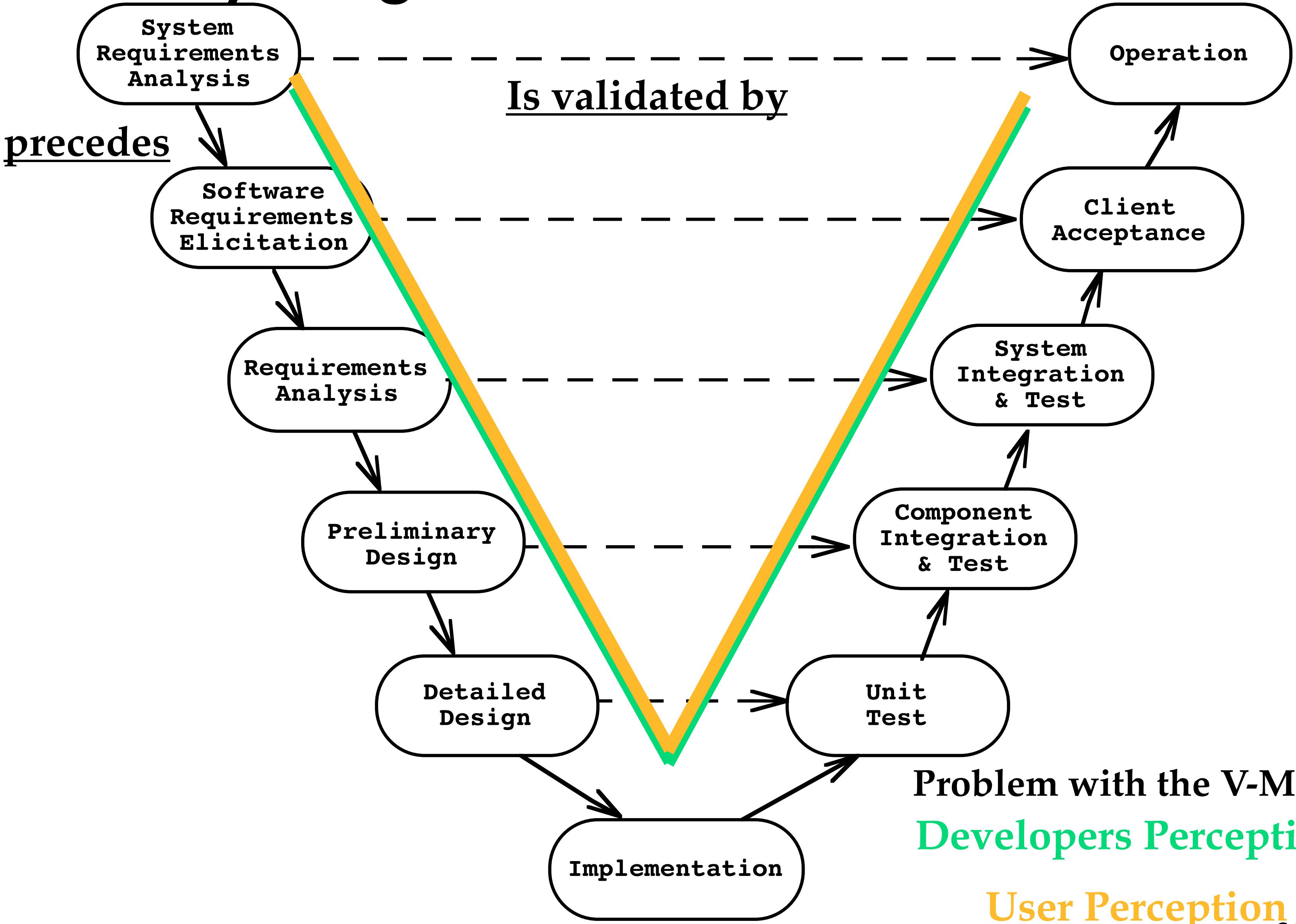
Activity Diagram of MIL DOD-STD-2167A


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

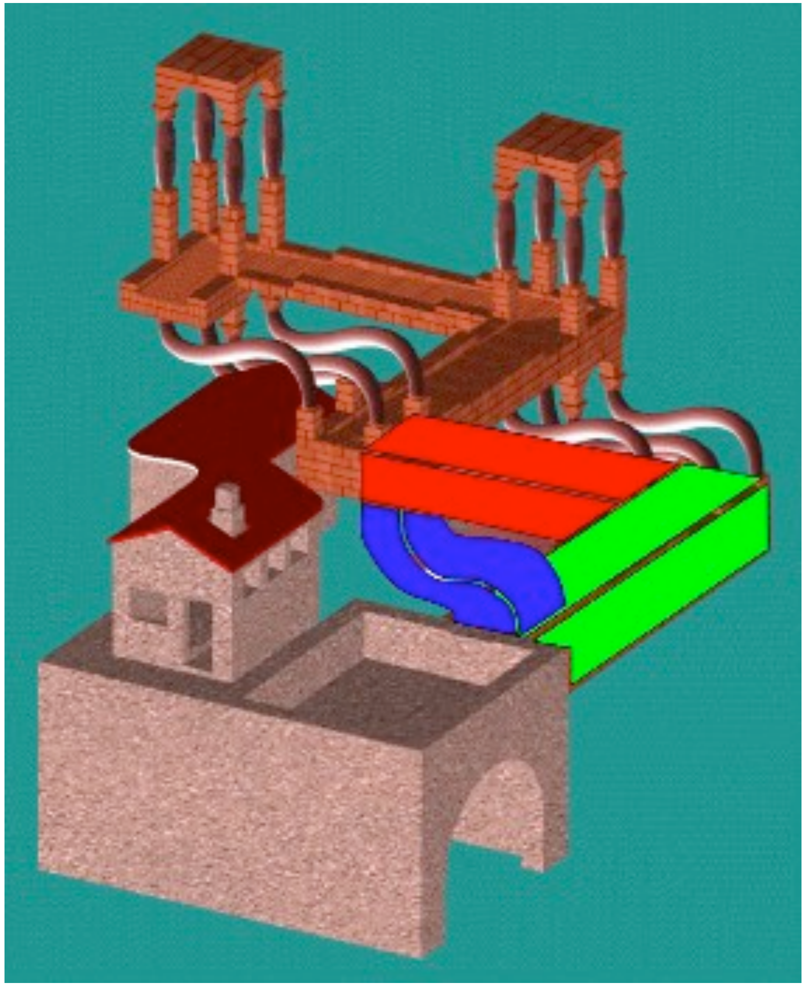
© 2008 Bernd Bruegge

The Alternative: Allow Iteration

Escher was the first:-)

http://www.mcescher.com/

Construction of Escher's Waterfall Model



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

Spiral Model

- The spiral model focuses on *addressing risks*
- This is done *incrementally*, in order of priority
- Main question: What is the highest risk?
  - Let's attack it first
- The spiral model contains of a set of activities
  - This set of activities is applied to a couple of so-called rounds.

Set of Activities in the Spiral Model

1. Determine objectives and constraints
2. Evaluate alternatives
3. Identify the risks
4. Assign priorities to the risks
5. Develop a prototype for each risk, starting with the highest priority
6. Follow a waterfall model for each prototype development
7. If a risk has been resolved, evaluate the results and plan the next round
8. If a risk cannot be resolved, terminate the project.

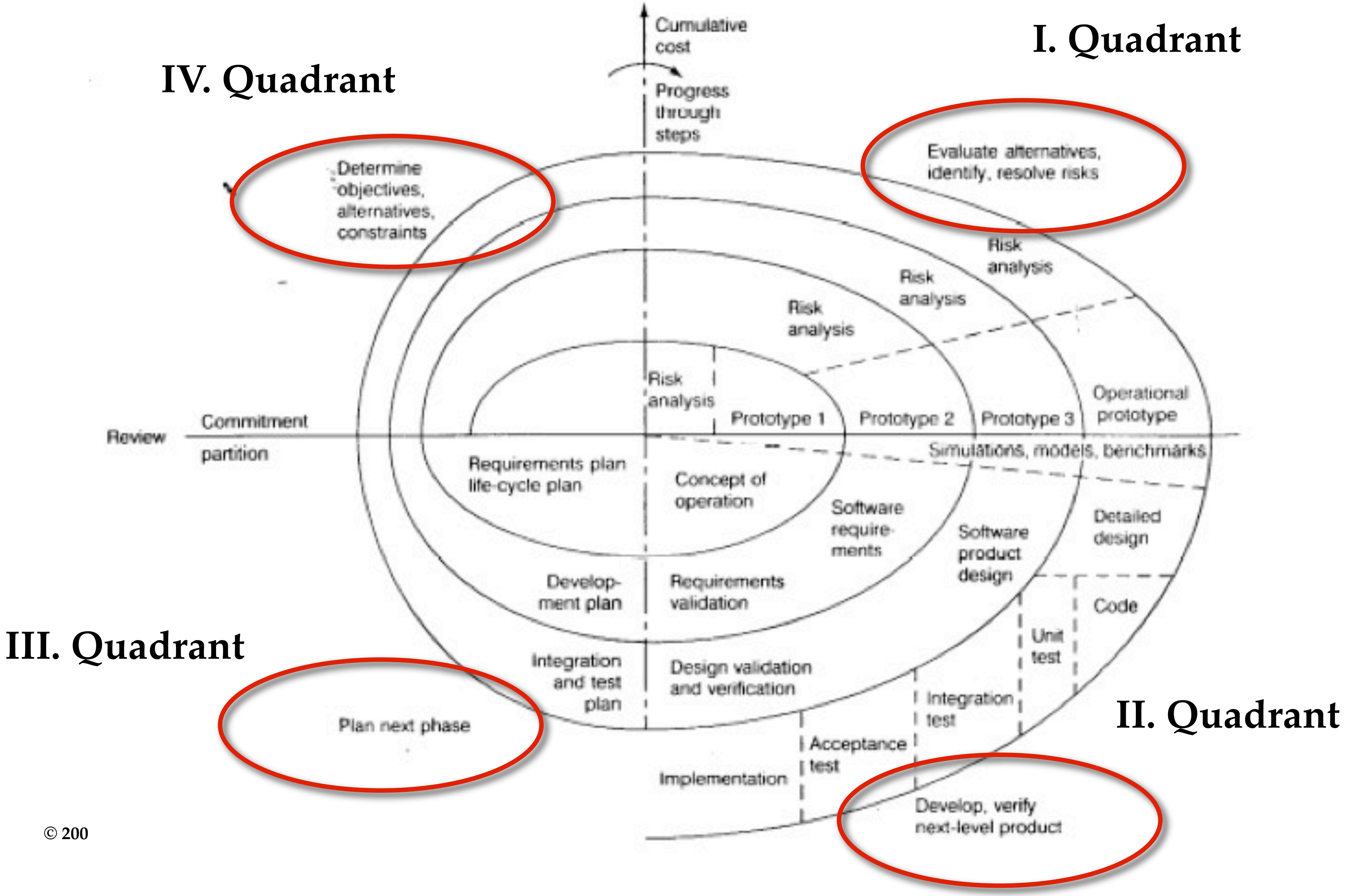
Rounds in Boehm's Spiral Model

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

- For each round, do the following:
  - Define objectives, alternatives, constraints
  - Evaluate alternatives, identify, prioritize and resolve risks
  - Develop a prototype
  - Plan the next round
  - Called the 4 Quadrants.

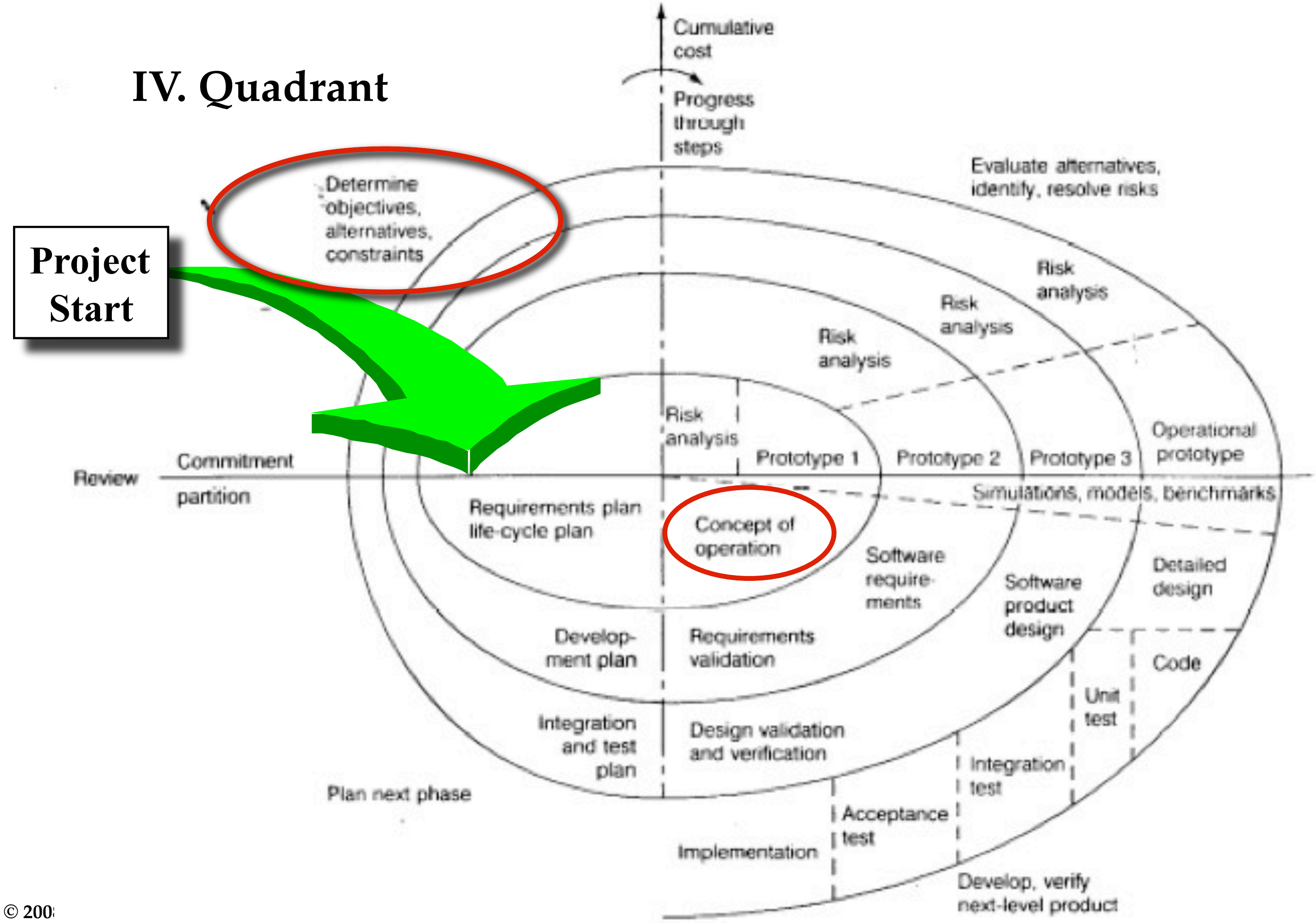
Disccourse on Prototyping

The 4 Quadrants in Boehm's Spiral Model

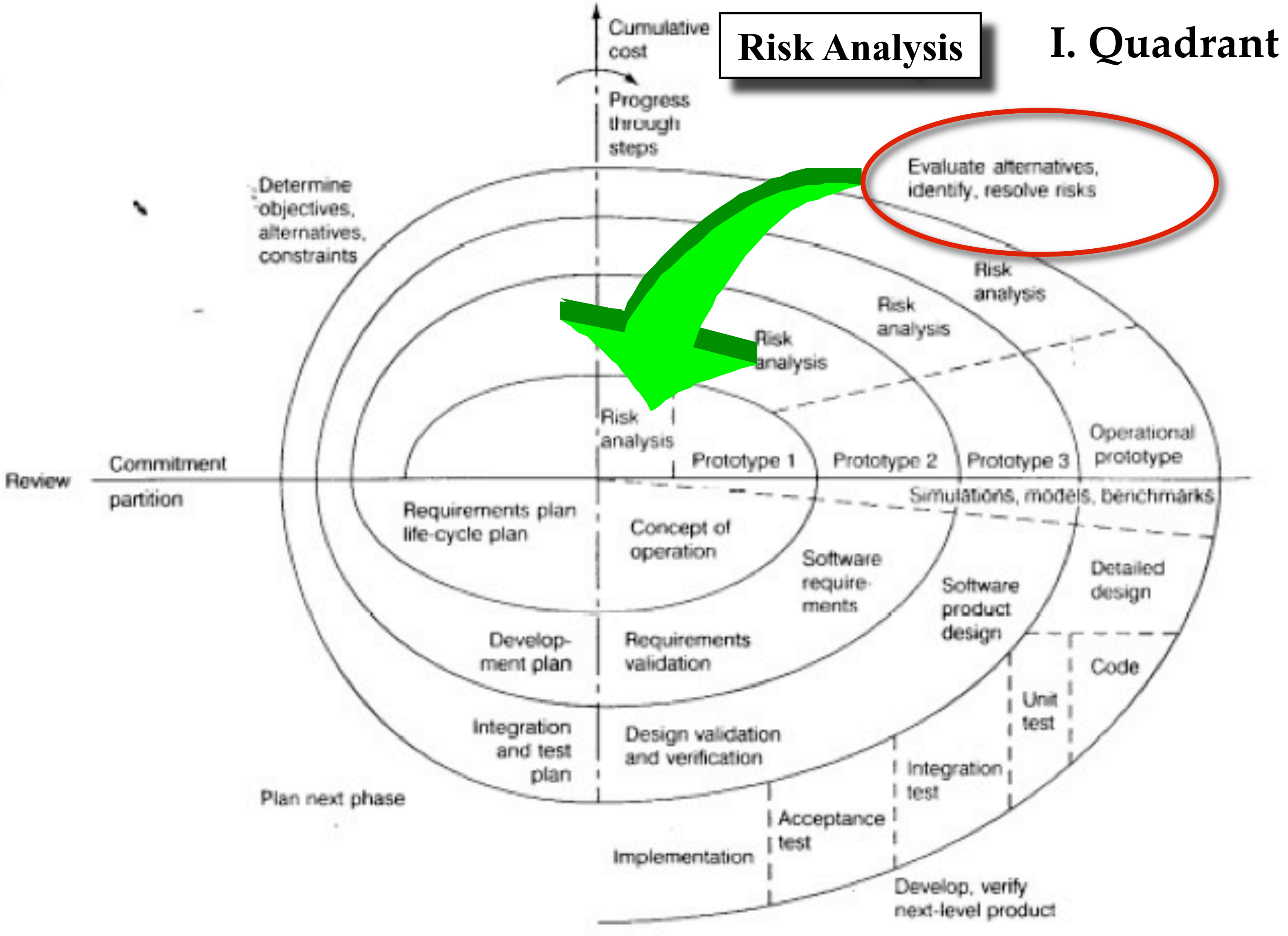


© 200

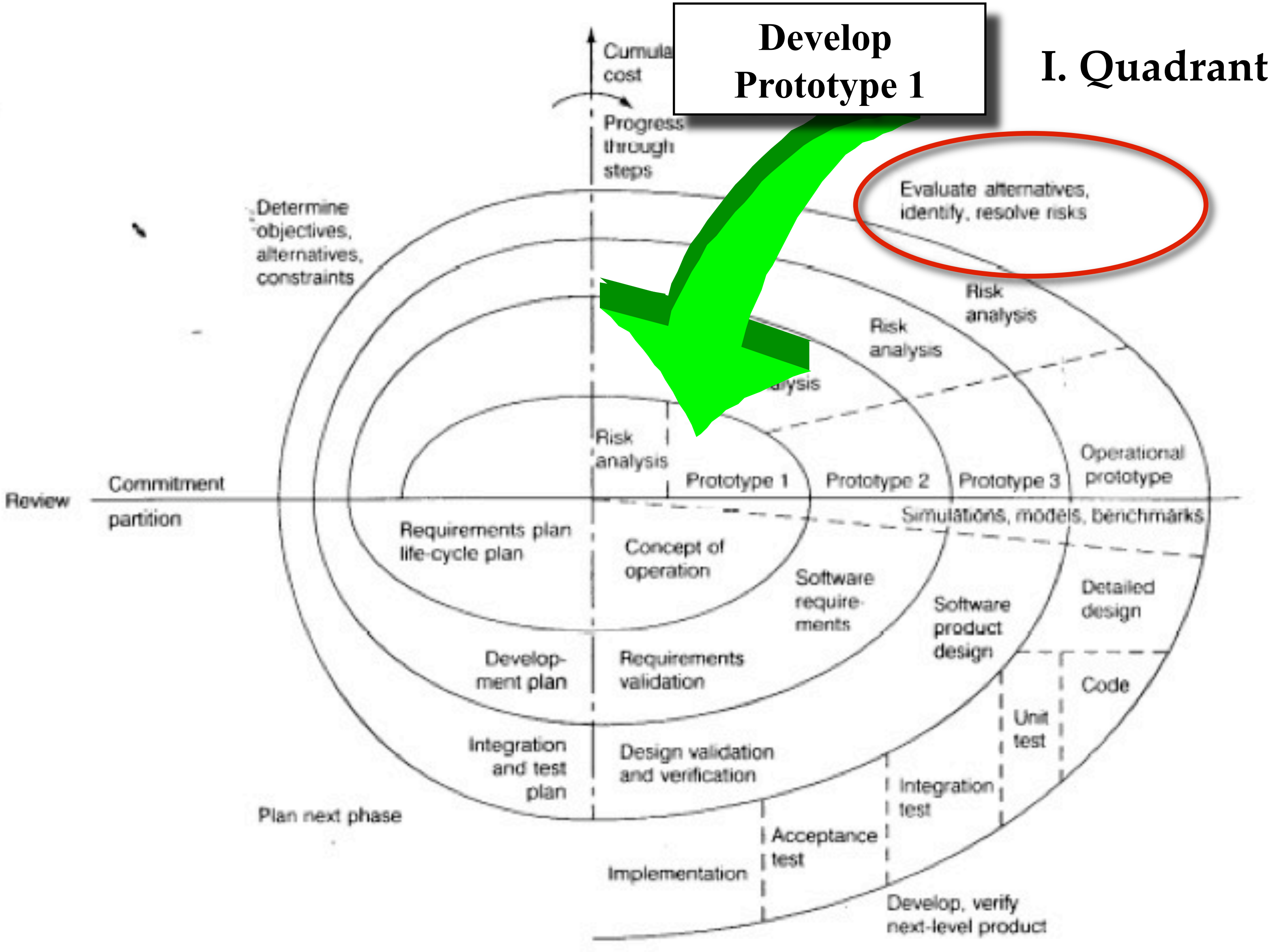
Round 1, Concept of Operations:
Determine objectives,alternatives & constraints



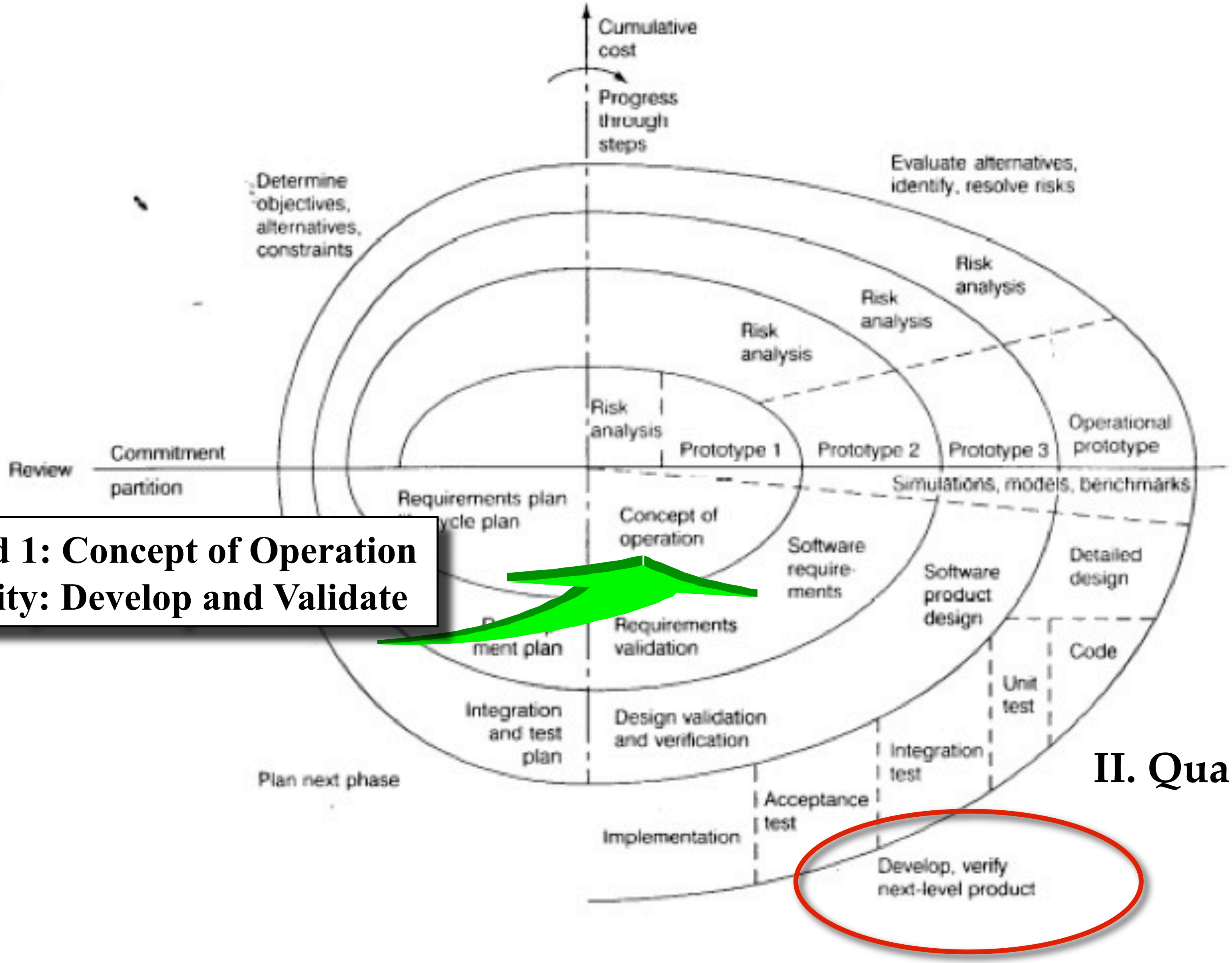
Round 1, Concept of Operations:
Evaluate alternatives, identify & resolve risks



Round 1, Concept of Operations:
Develop a prototype for the highest risk



Round 1, Concept of Operations:
Develop and validate

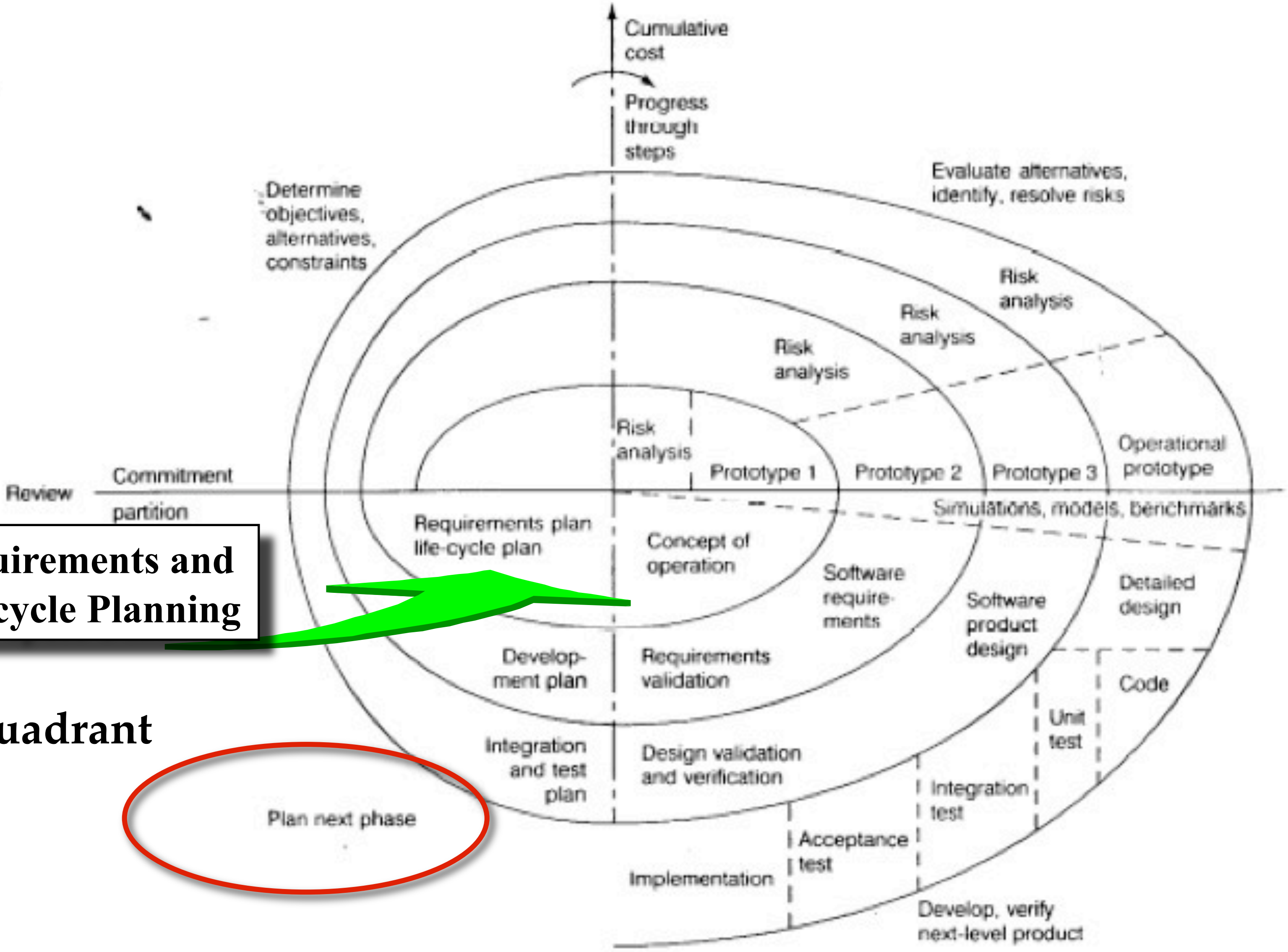


Round 1: Concept of Operation
Activity: Develop and Validate

II. Quadrant

© 200

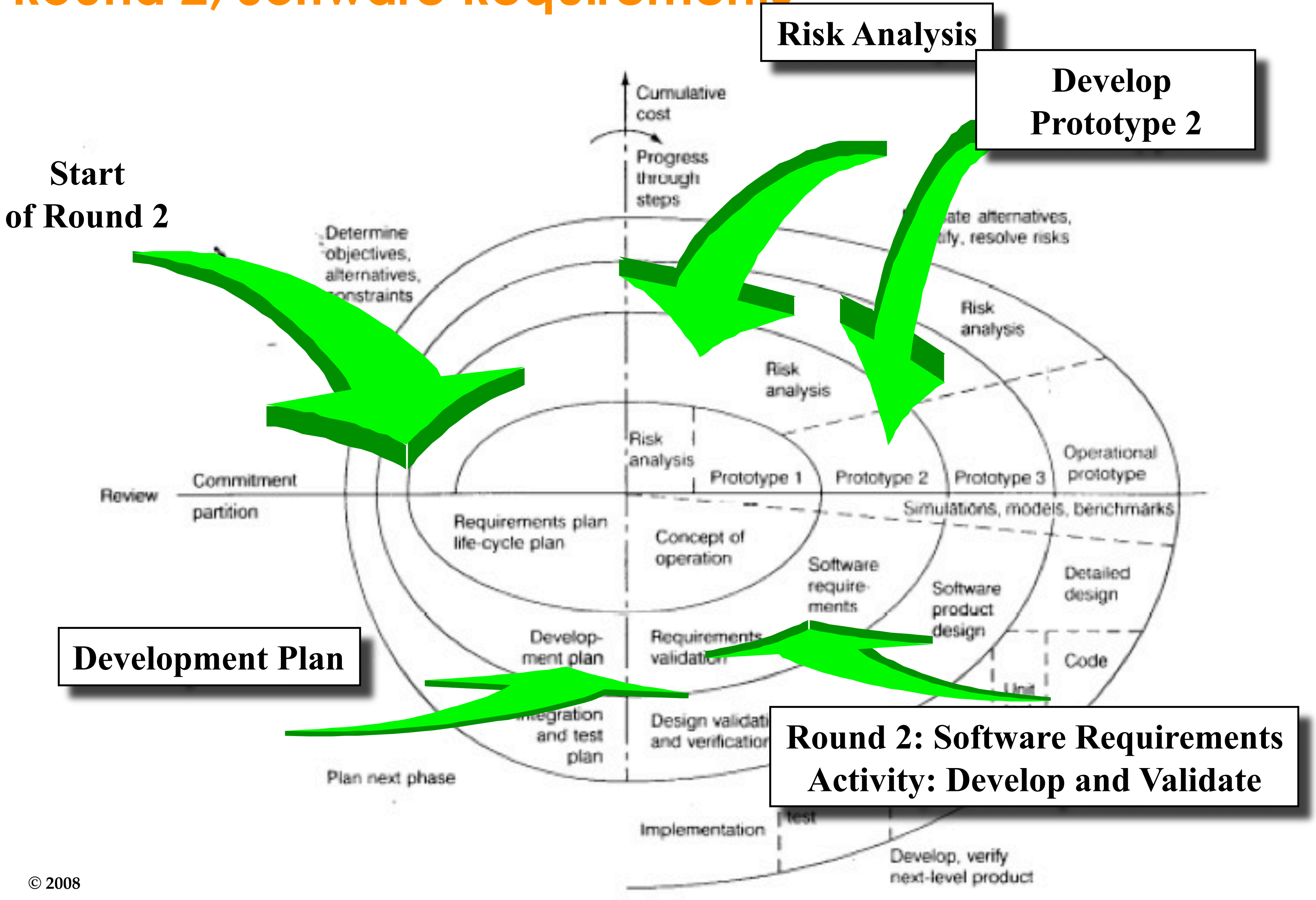
Round 1, Concept of Operations:
Prepare for Next Round



Requirements and Life cycle Planning

III. Quadrant

Round 2, Software Requirements



Risk Analysis

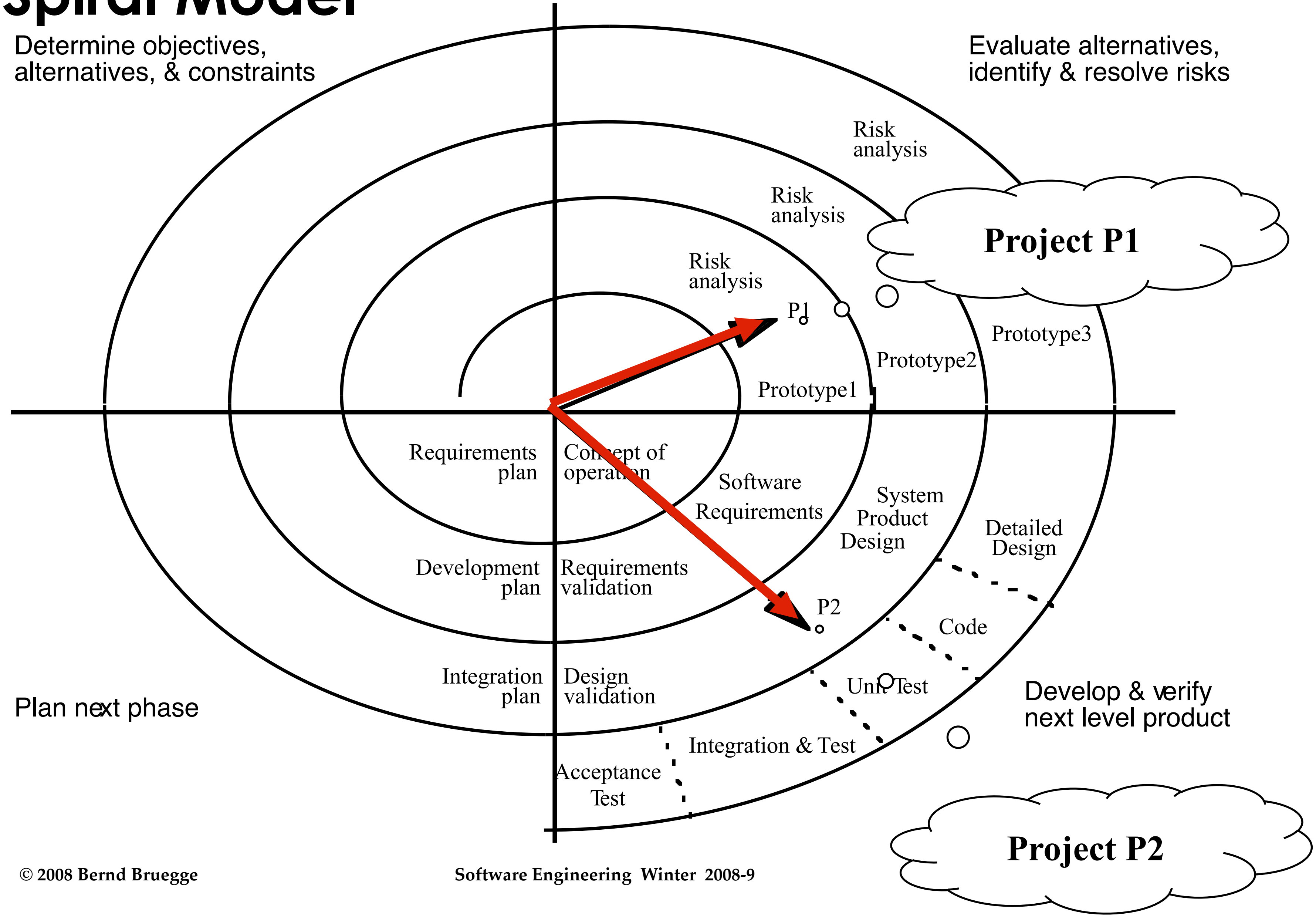
Develop Prototype 2

Start of Round 2

Development Plan

Round 2: Software Requirements Activity: Develop and Validate

Comparison of Projects on the basis of the Spiral Model



Are these models good enough for today's software development challenges?

Properties of Linear Lifecycle Models

- Managers love linear models
  - Nice milestones
  - No need to look back (linear system)
  - Always one activity at a time
  - Problem with progress checks: "The system is 90% coded, 90%, 90%...", "We are done 20% of our tests"
- The Spiral model has the property of many concatenated waterfalls
- Two "surviving models" in the evolution of activity-oriented software lifecycle models:
  - V-Model XT: successor of the V model
  - Unified Process: successor ofthe spiral model.

Outline of the Lecture

- ✓ Modeling the software life cycle
- ✓ Sequential models
  - ✓ Pure waterfall model
  - ✓ V-model
- ✓ Iterative models
  - ✓ Boehm’s spiral model

Jan 13

  - ⇨ Unified Process
- • Entity-oriented models
  - • Issue-based model

Jan 16

Exercise Session next Thursday

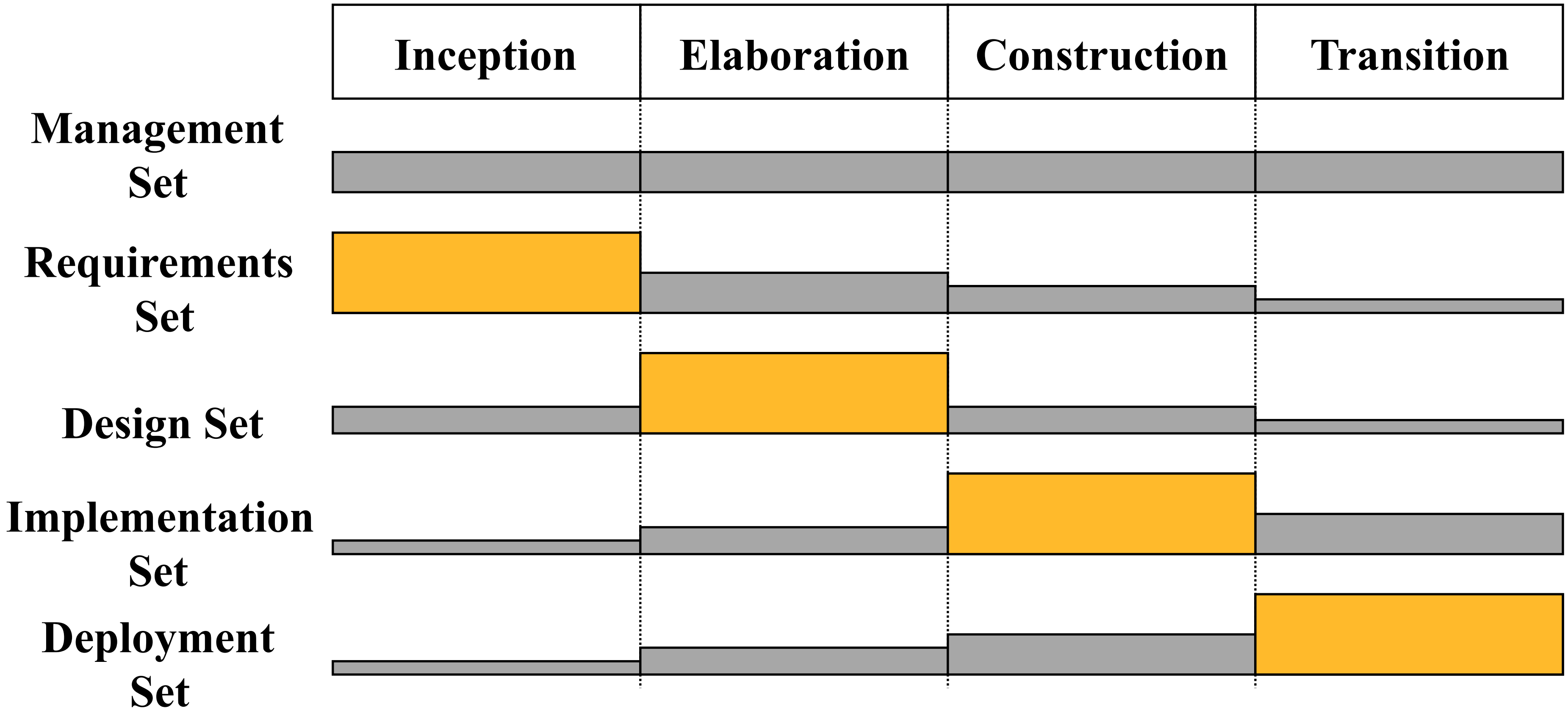
- Install Cruise Control on your Laptop before coming.
- You can work in teams of 3.
- One project with several new requirements, each team selects a different requirement and implements it.
- Duration: 90 minutes
- First price for best delivery: 1 bottle of champaign.
- Product: Race car crash game.

Unified Process

- The Unified Process is another iterative process model
- 4 states of a software system
    - Inception, Elaboration, Construction, Transition
- 2 Artifacts Sets
    - Management Set, Engineering Set
- 7 Workflows
    - Management, Environment, Requirements, Design, Implementation, Assessment, Deployment
- Project participants are called stakeholders.

Key Idea behind the Unified Process

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



Focus Areas in the Unified Process

- The Unified Process supports the following
  - Evolution of project plans, requirements and software architecture with *well-defined synchronization points*
  - Risk management: Contingency plans for risks
  - 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 risks, unpredictable issues, smaller teams
- Production Stage:
  - Focuses on construction, test and deployment, driven by more predictable issues, artifacts and quality assessment, 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, Reviews	Testing

Phases in the Unified Process

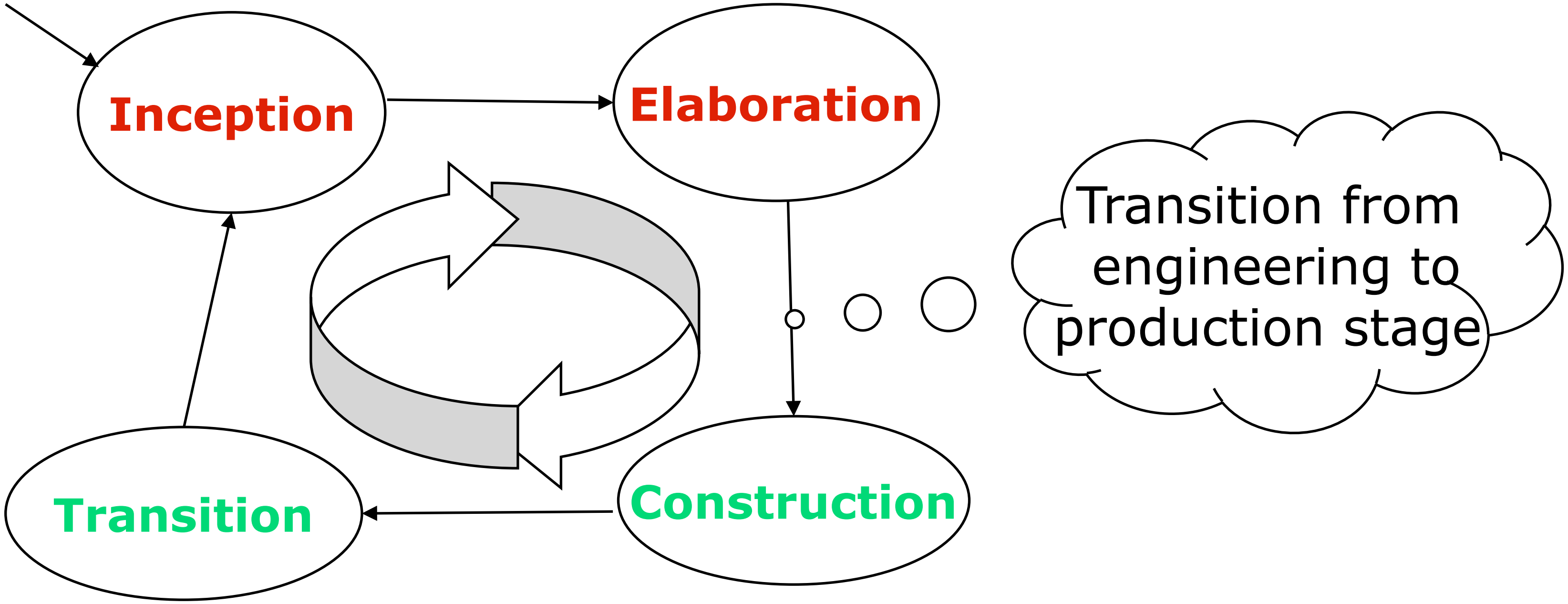
The 2 major stages are decomposed into 4 phases

Engineering stage

1. Inception phase
2. Elaboration phase

Production stage

3. Construction phase
4. Transition phase



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

- **Stages** and **phases** are nothing else but arbitrary names of the states - actually **superstates** and **states** - of a project.
-

Inception Phase: Objectives

- Establish the project’s scope
- Define acceptance criteria (for the client acceptance test)
- 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 (more “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 developers
- 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
  - milestones are defined (“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

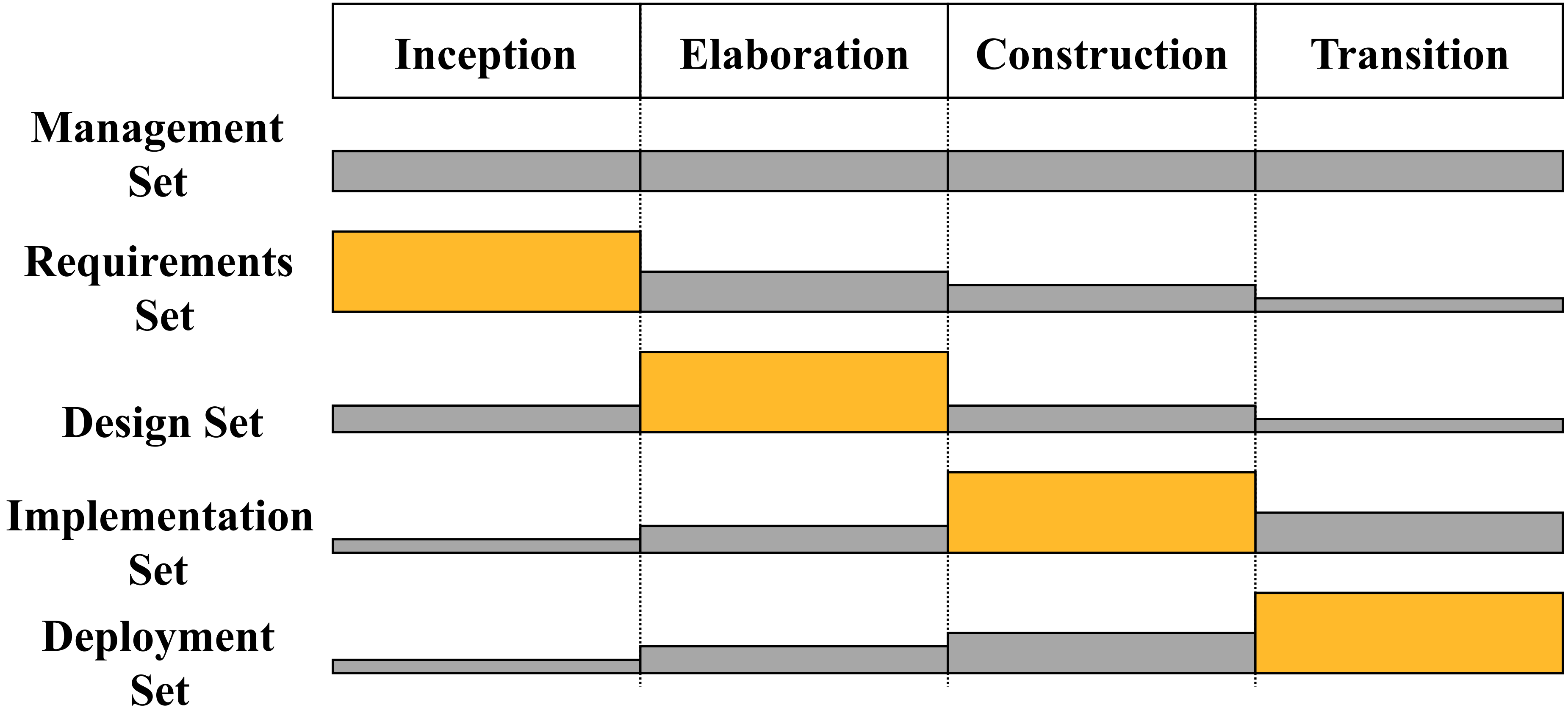
1. Work breakdown structure
2. Business Case
3. Release specifications
4. Software Project Management Plan

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



Inception

Elaboration

Construction

Transition

Management Set

Requirements Set

Design Set

Implementation Set

Deployment Set

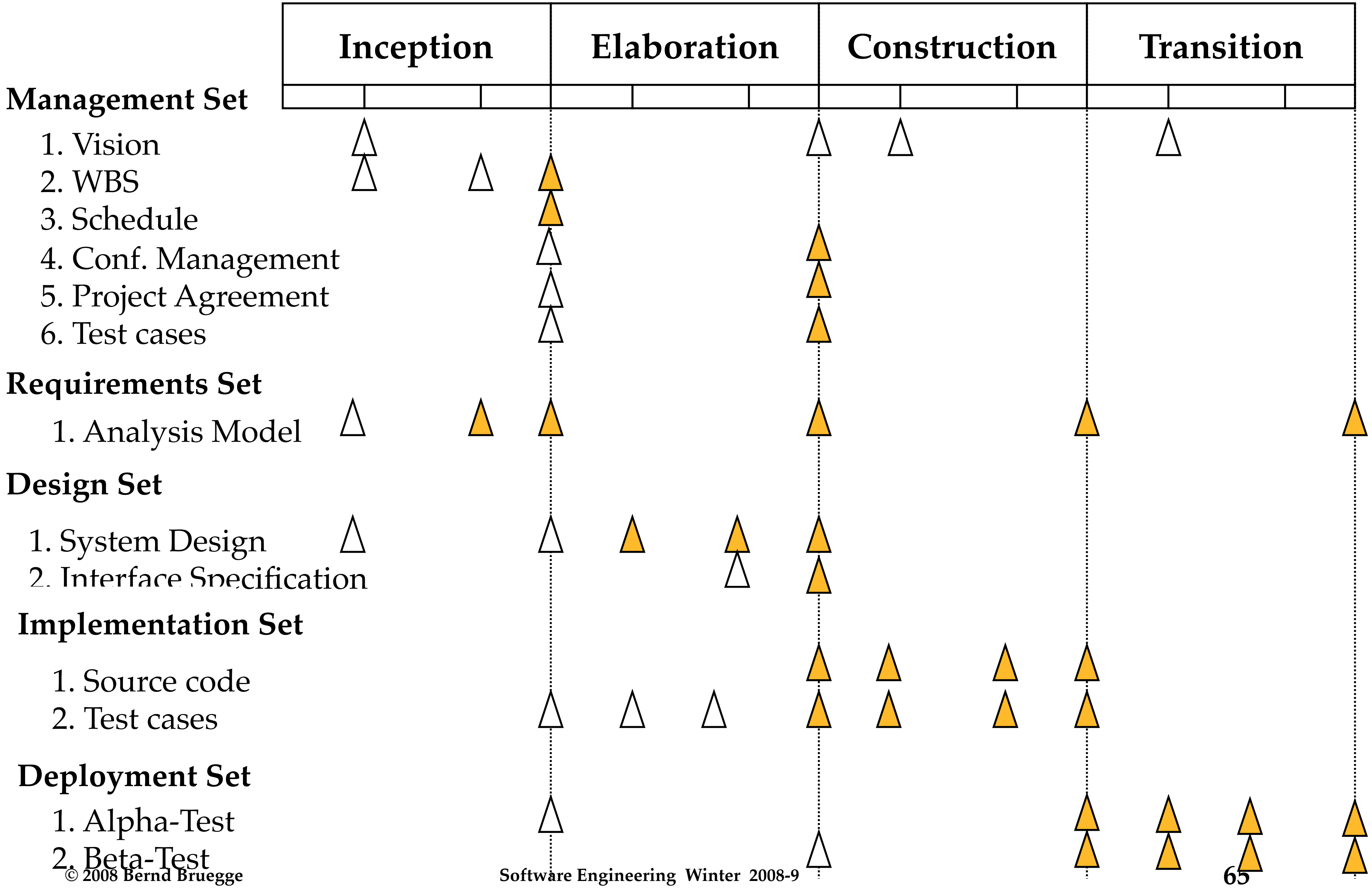
Management of Artifact Sets

- 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 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

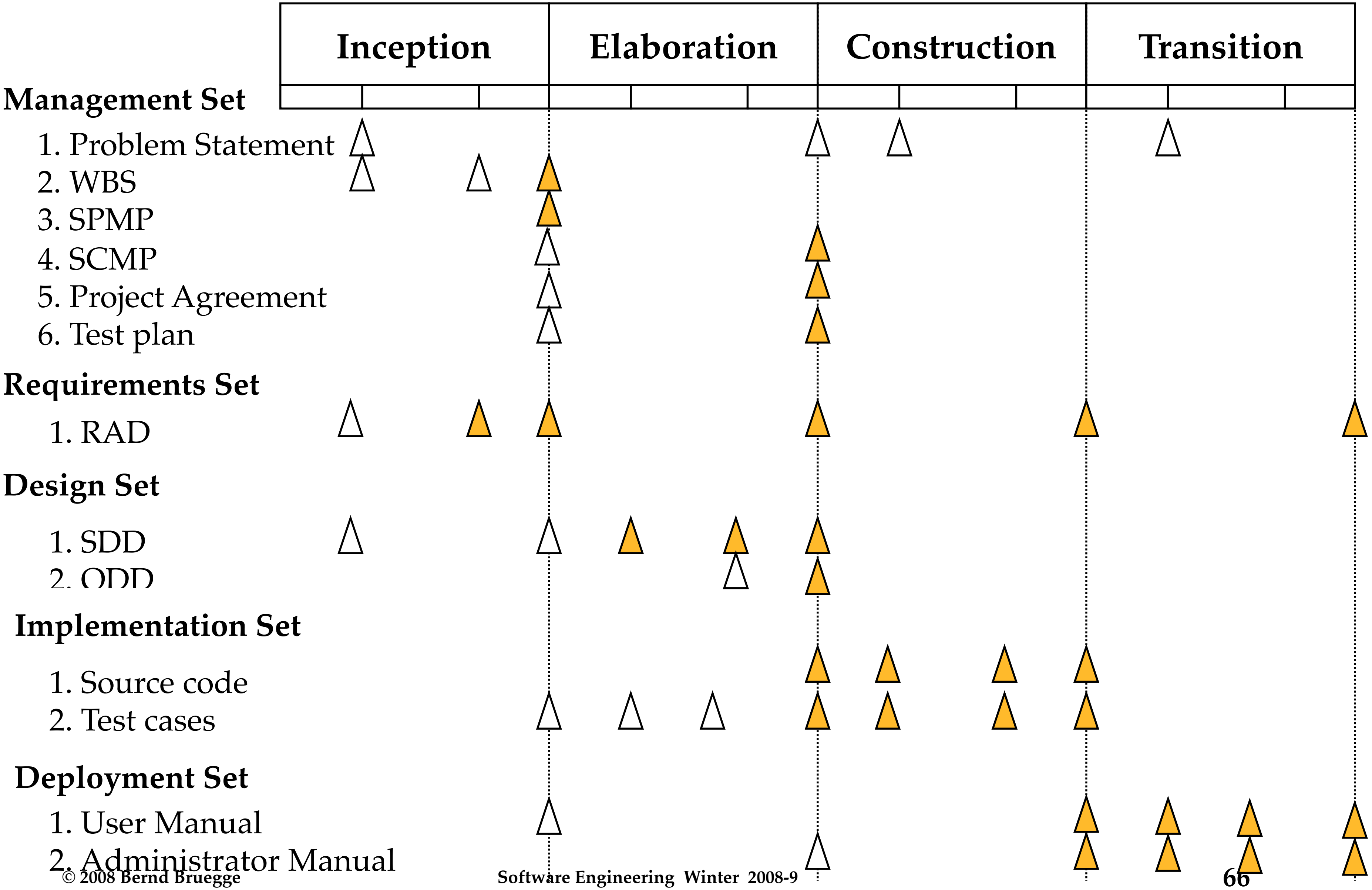
Deployment Set

1. Alpha-Test
2. Beta-Test

Artifact Roadmap: Focus on Documents

△ Informal

▲ Baseline



Management Set

1. Problem Statement
2. WBS
3. SPMP
4. SCMP
5. Project Agreement
6. Test plan

Requirements Set

1. RAD

Design Set

1. SDD
2. ODD

Implementation Set

1. Source code
2. Test cases

Deployment Set

1. User Manual
2. Administrator Manual

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

- Focus of a modern software development project is model-driven
  - 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.

Model-Driven Approach

- Provide document templates at project start
  - Project specific customization
- Instantiate documents automatically from these templates
  - Enriches them with modeling information generated during the project
- Automatically generates 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
  - Regression tester.

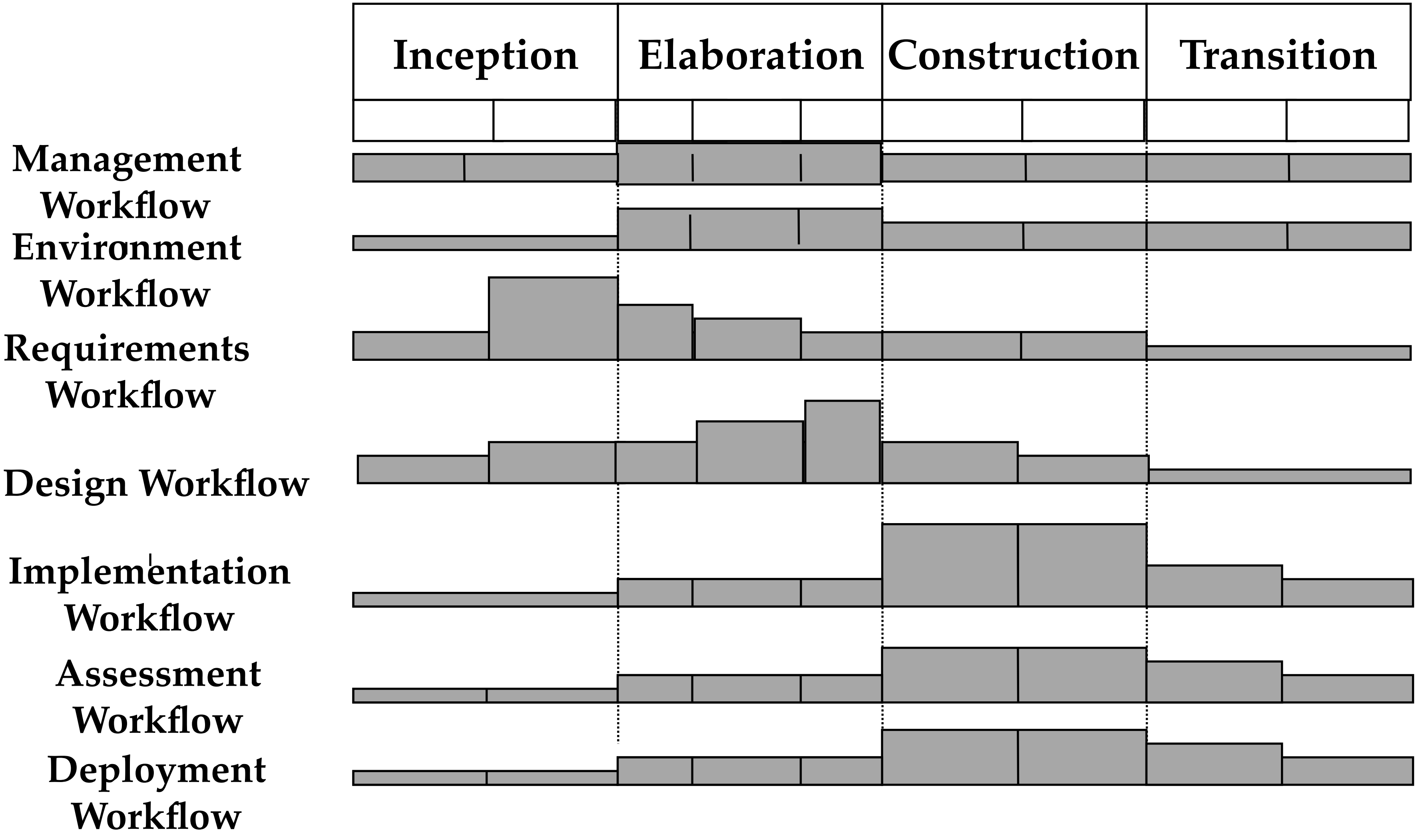
Workflows in the Unified Process (1)

- Management workflow
  - Planning of the project (Creation of problem statement, SPMP, SCMP, test plan)
- Environment workflow
  - Automation of process and maintenance environment. Setup of infrastructure (communication infrastructure, configuration management, build environment).
- 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, unit testing, integration testing, system testing, regression testing)
- Deployment workflow
  - Transition the software system to the end user.

Workflows vs Phases



Workflows vs Phases

- A Phase describes the status of a software system
  - Inception, elaboration, construction, transition
- Workflows can consist of one or more iterations per phase
  - "We are in the 3rd iteration in the design workflow", "We are in the 3rd iteration during design"
- Workflows create artifacts (models, documents) for the artifact sets
  - Management set, engineering set.

Managing Projects in the Unified Process

- How to 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.

The term “Process“ has many meanings 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 (Grady Booch)
  - Techniques for achieving an artifact of the software process.
  - Focus: Intermediate baselines with adequate quality and functionality, as economically and rapidly as practical.

Phase vs. Iteration

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

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 model can deal with change between rounds, but do not allow change within a round
  - The Unified Process model can deal with change in an iteration, but it has problems to deal with change within a iteration
- What do we do if change is happening more frequently?
  - “The only constant is the change” (Hammer & Champy, Reengineering).

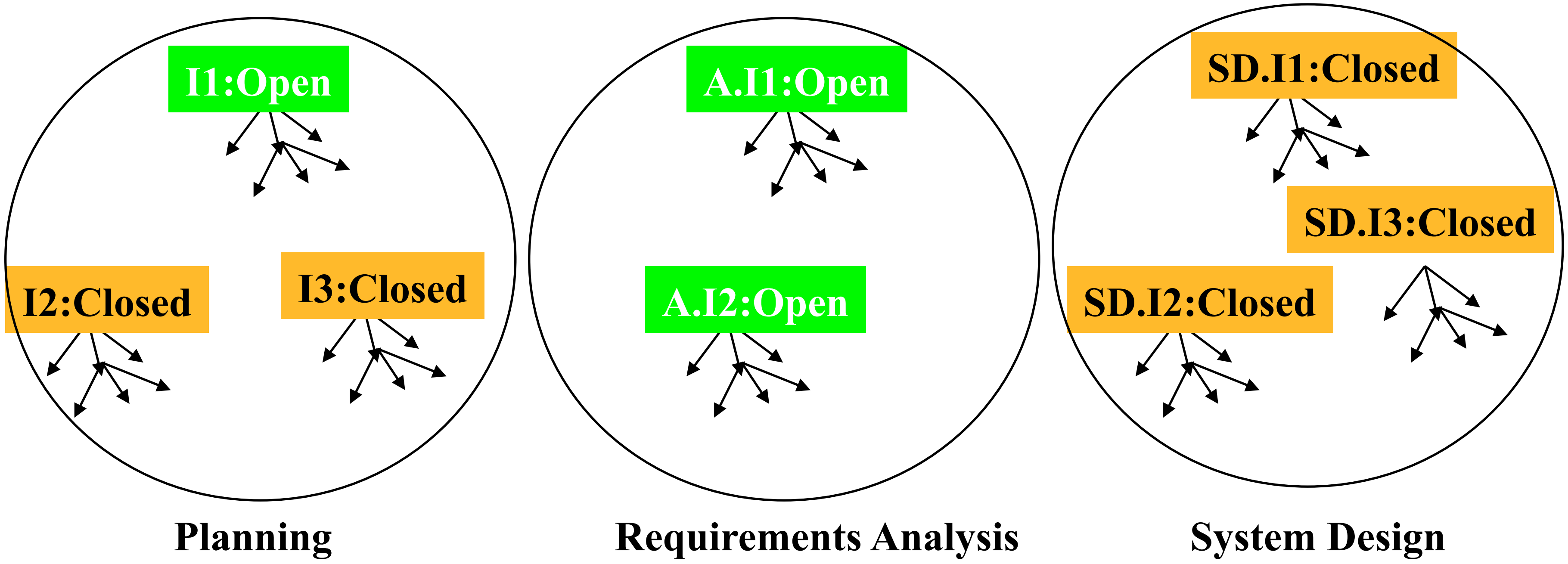
Frequency of Change and Choice of Software Lifecycle Model

PT = Project Time, MTBC = Mean Time Between Change

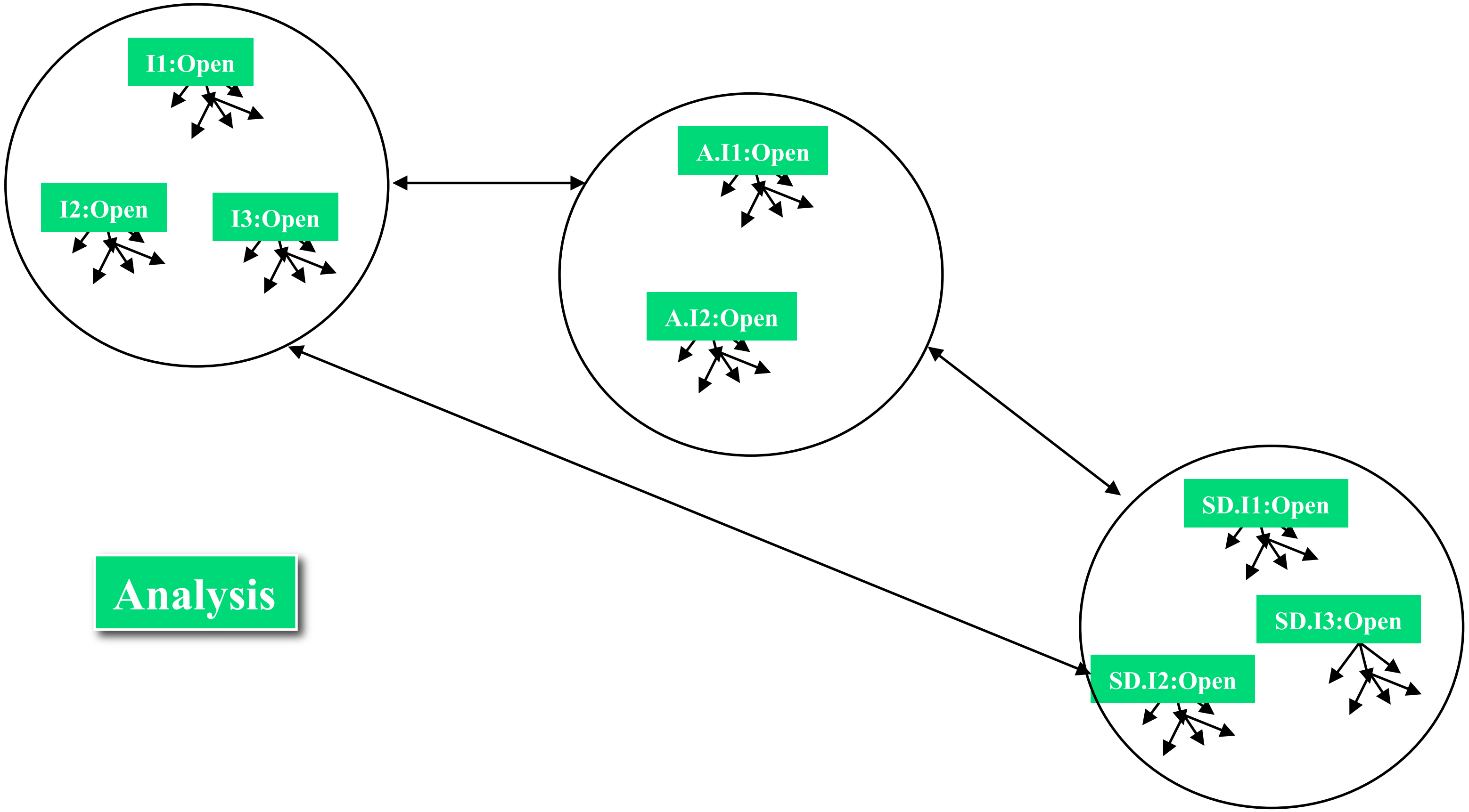
- Change rarely occurs (MTBC » PT)
  - Waterfall Model
  - Open issues are closed before moving to next phase
- Change occurs sometimes (MTBC ≈ PT)
  - Boehm's Spiral Model, Unified Process
  - Change occurring during phase may lead to iteration of a previous phase or cancellation of the project
- Change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

An Alternative: Issue-Based Development

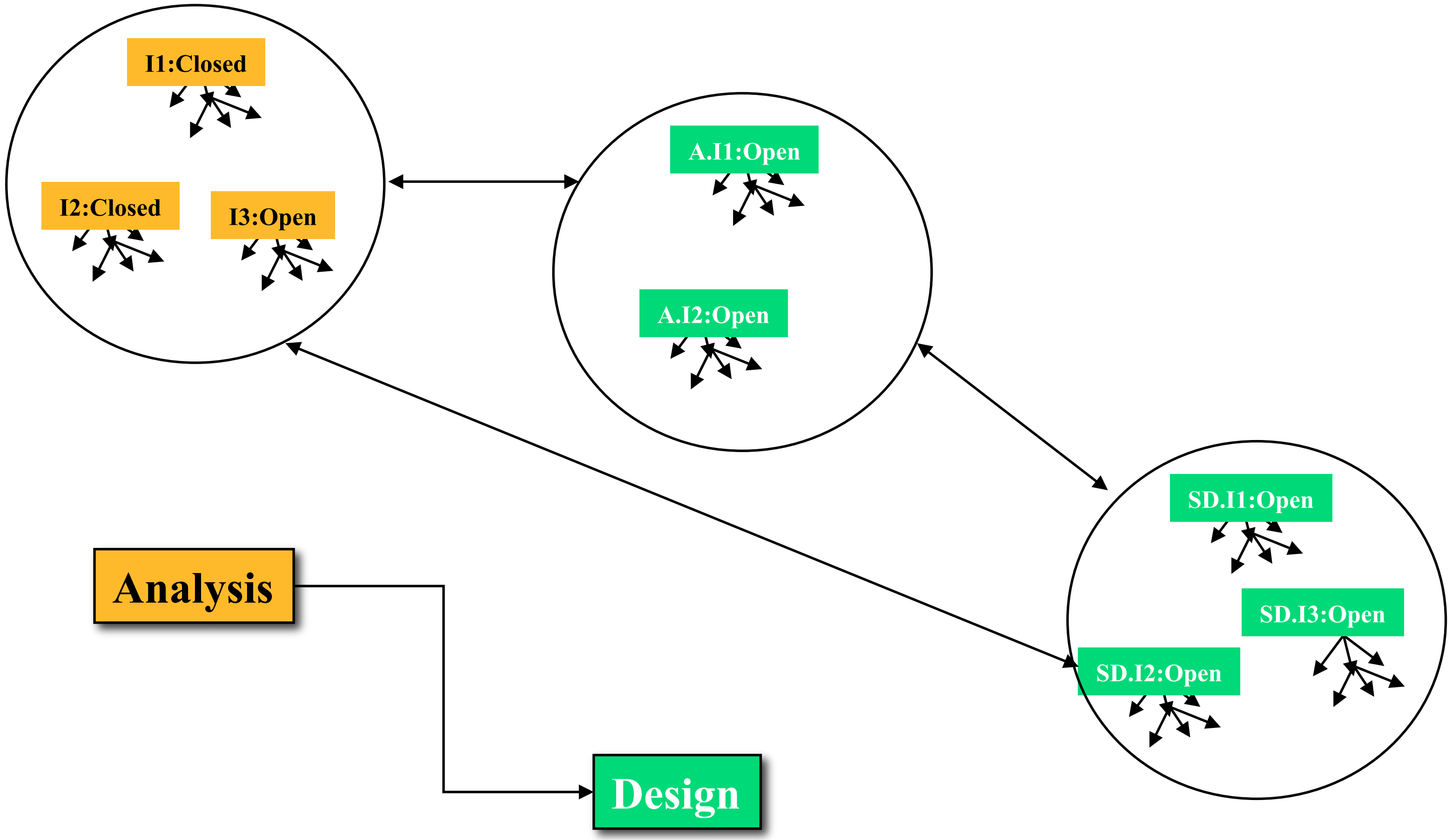
- A system is described as a collection of issues
  - Issues are either closed or open
  - Closed issues have a resolution
  - Closed issues can be reopened (Iteration!)
- The set of closed issues is the basis of the system model



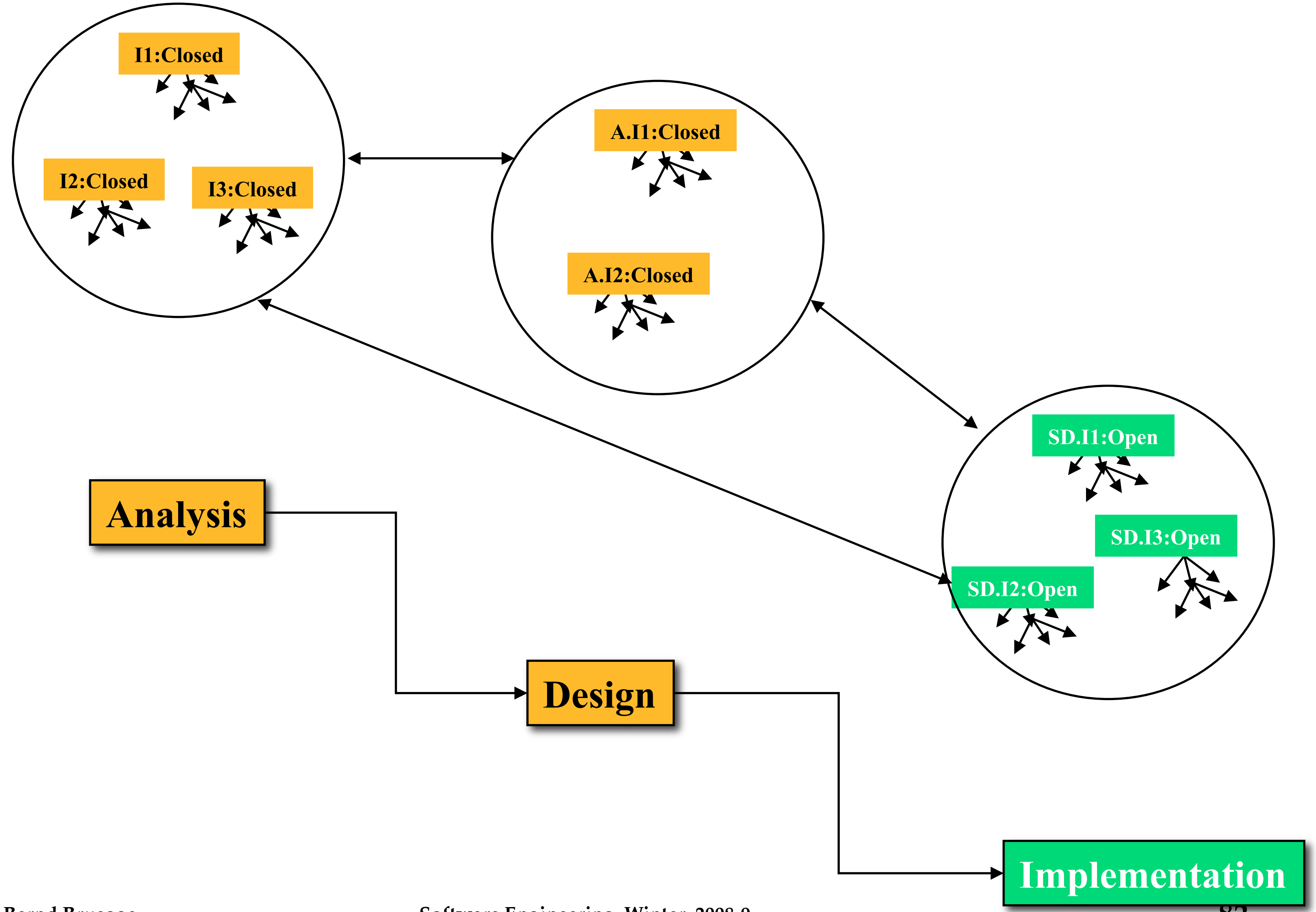
Waterfall Model: Analysis Phase



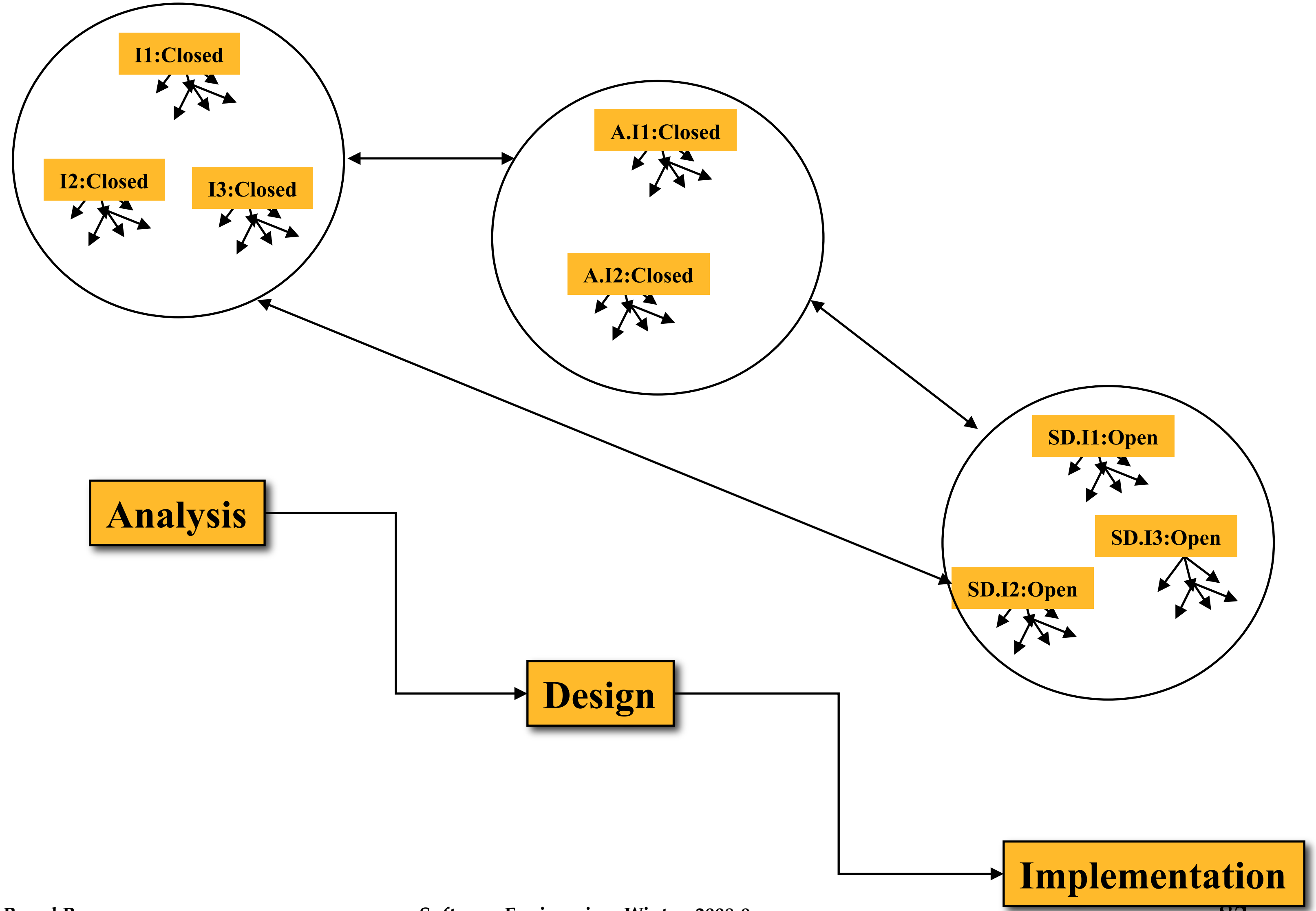
Waterfall Model: Design Phase



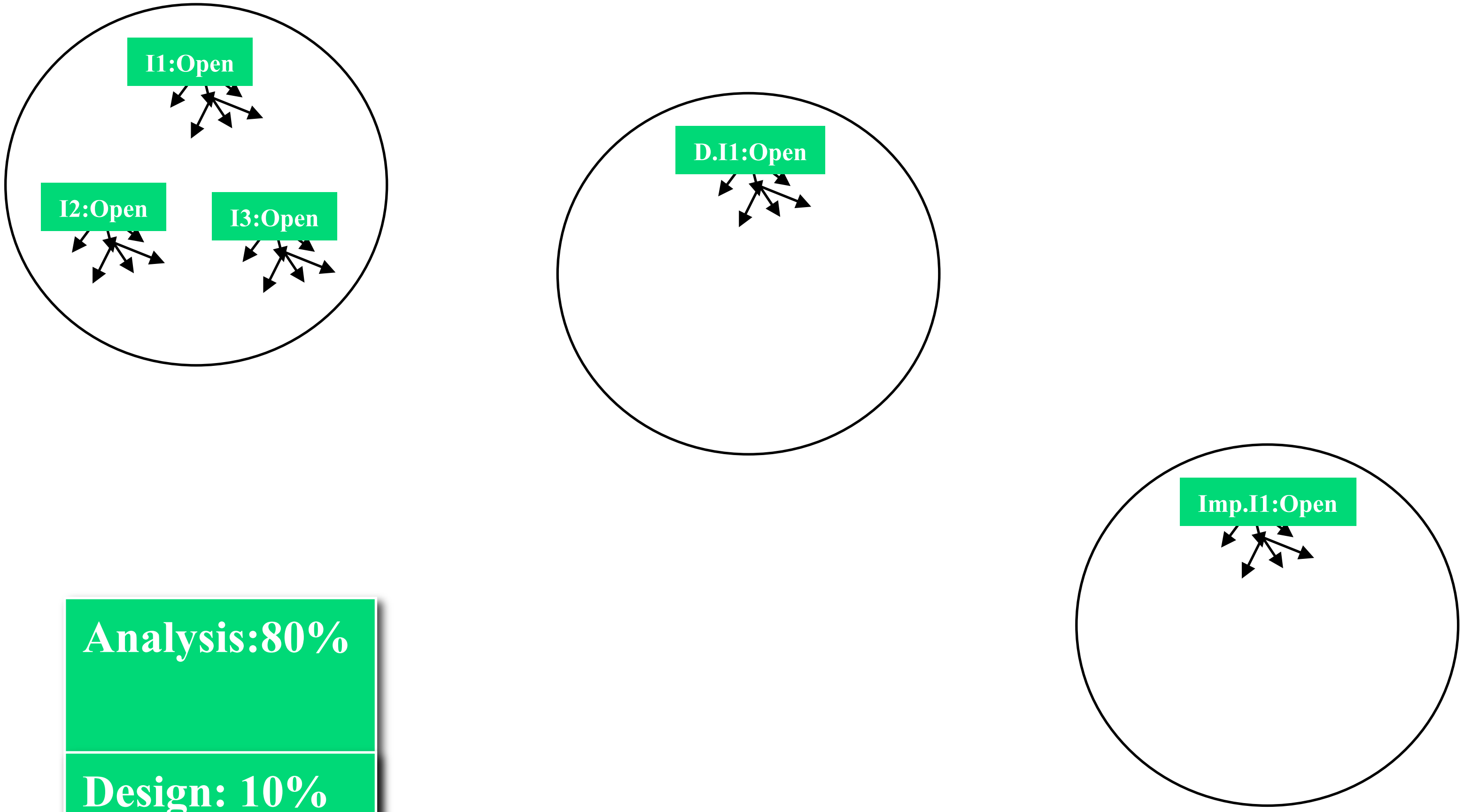
Waterfall Model: Implementation Phase



Waterfall Model: Project is Done



Issue-Based Model: Analysis Phase

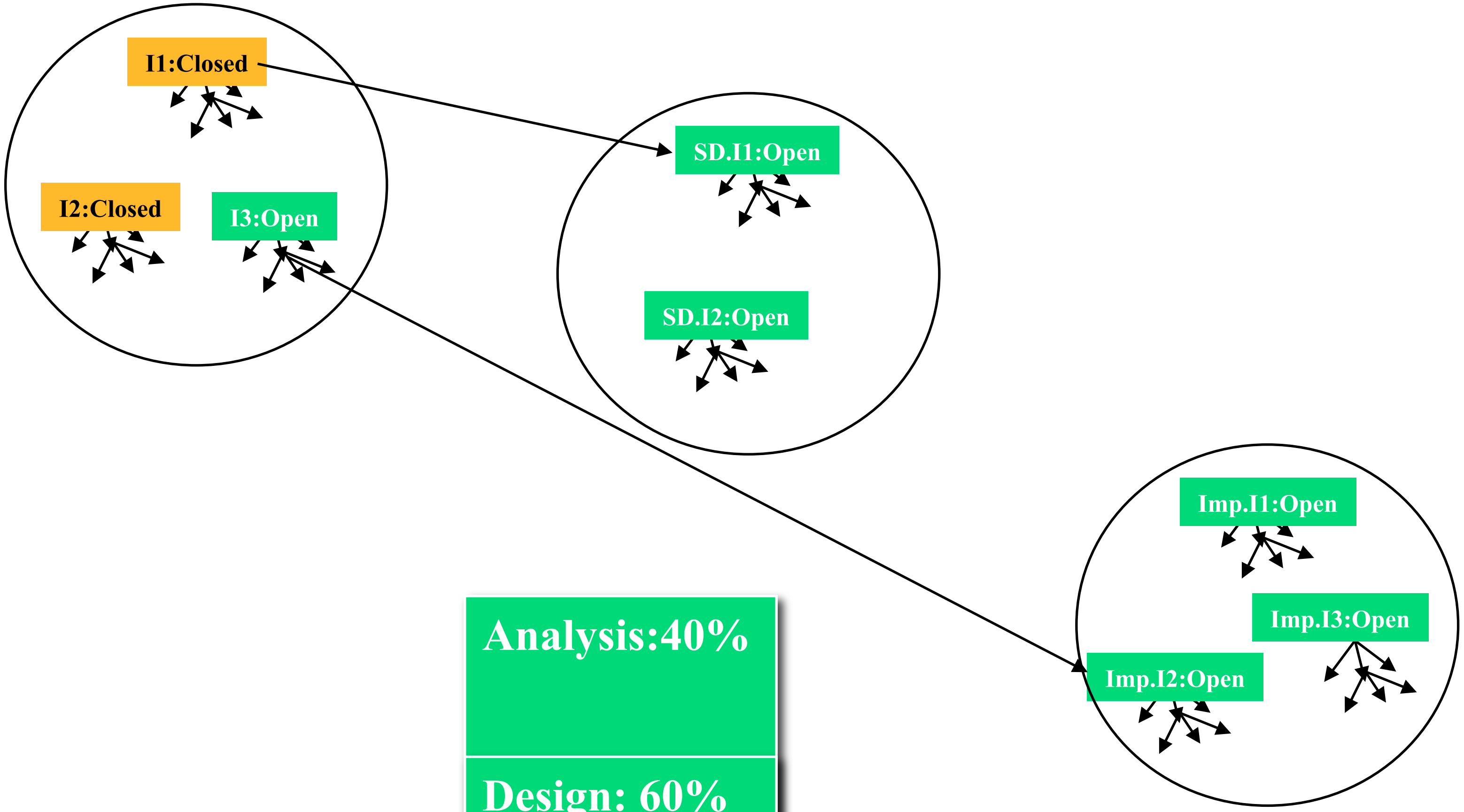


Analysis:80%

Design: 10%

Implementation: 10%

Issue-Based Model: Design Phase

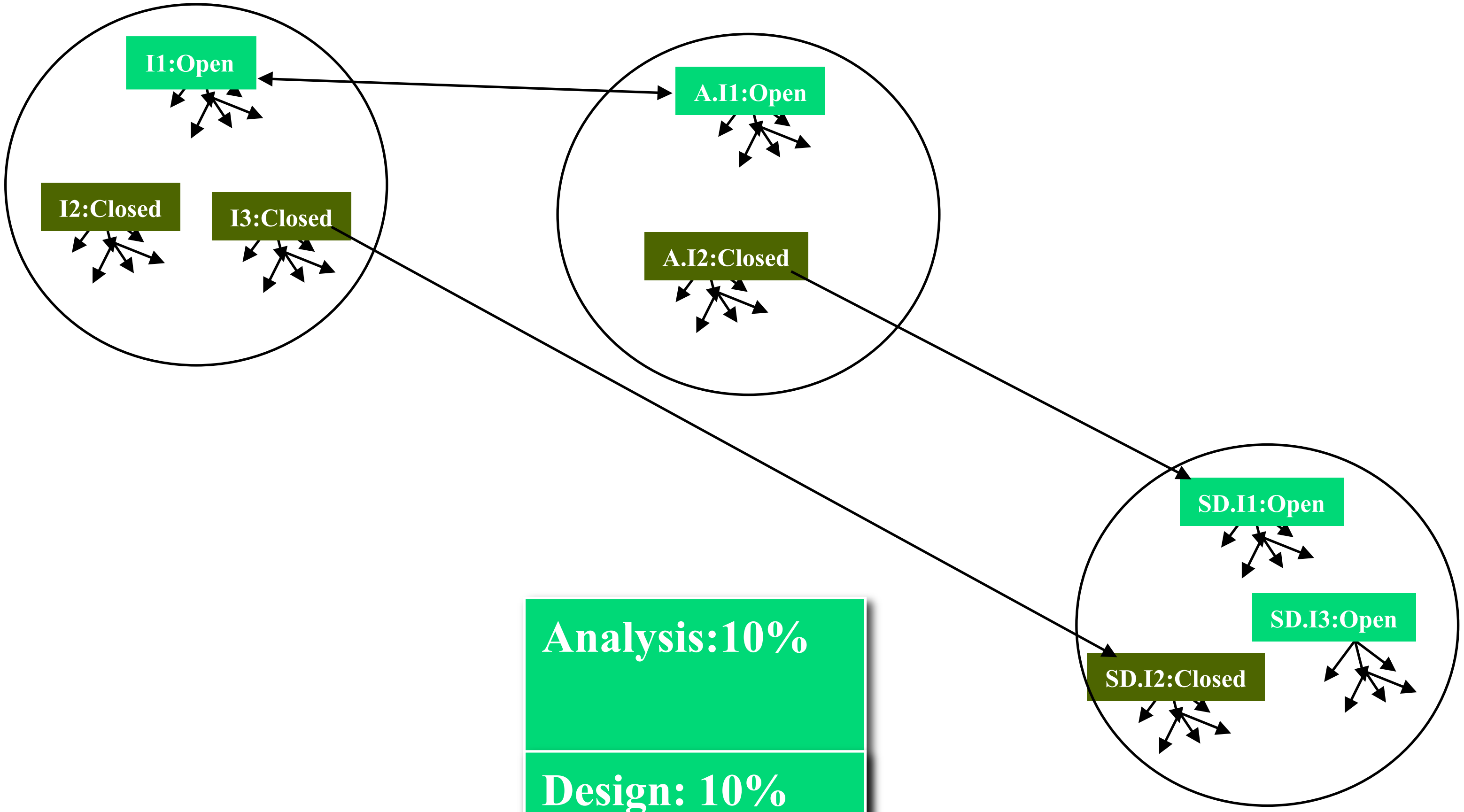


Analysis:40%

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

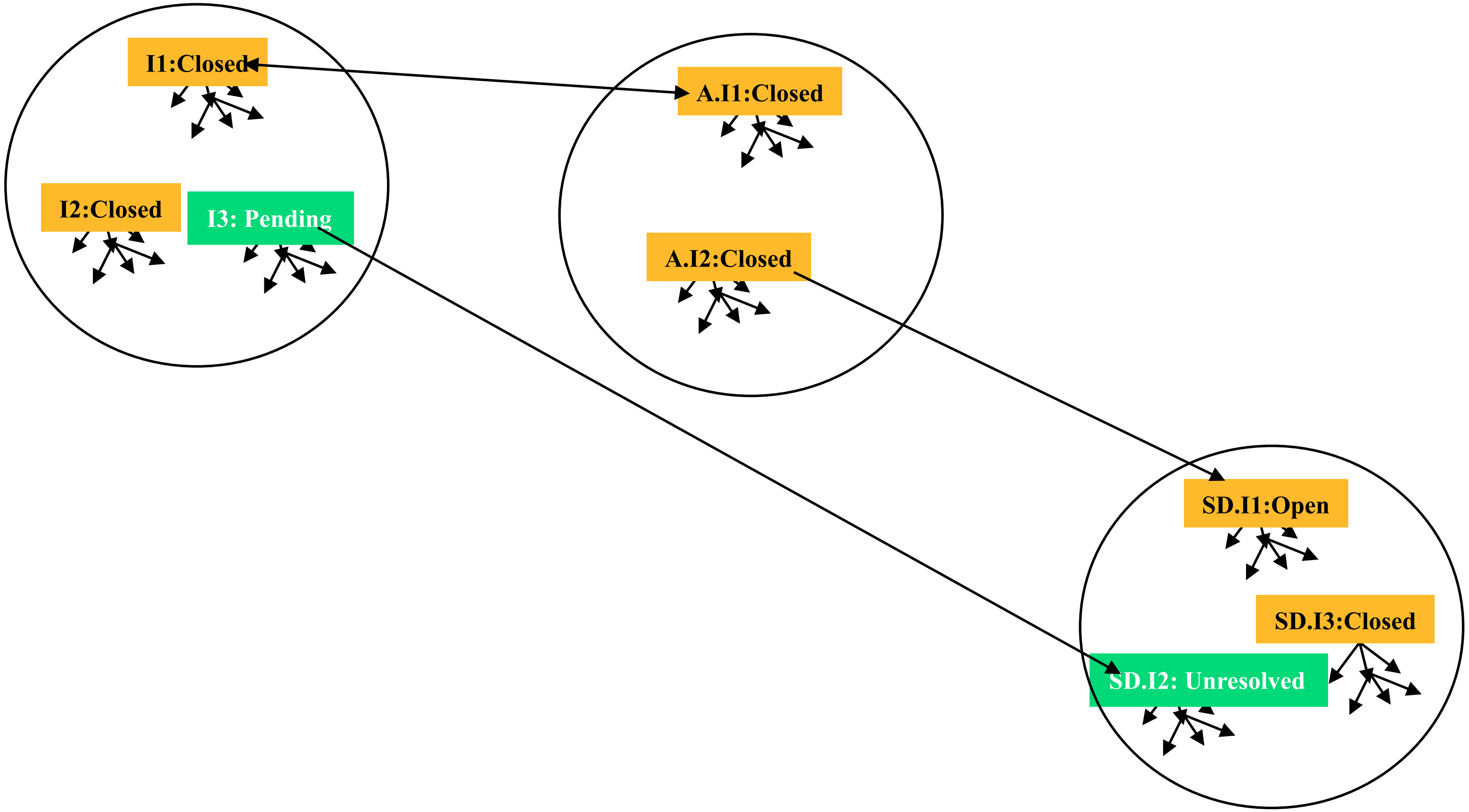
SD.I2:Closed

Analysis:10%

Design: 10%

Implementation: 60%

Issue-Based Model: Prototype is Done



Summary Unified Process

- Unified Process: Iterative software lifecycle model
  - 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.
  - An iteration in the unified process is 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
- Prototype
  - A specific type of system demonstrating one aspect of the system model without being fully operational
    - Illustrative, functional and exploratory prototypes
- Prototyping
    - 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.
- Andreas Rausch et. al
  - Das V-Modell XT: Grundlagen, Methodik und Anwendungen Springer, 2006.
- V-Model XT Online Dokumentation
  - ftp://ftp.tu-clausthal.de/pub/institute/informatik/v-modell-xt/Releases/1.2.1/Documentation/V-Modell-XT-Complete.pdf.

Additional Readings (2)

- Winston Royce
  - Managing the Development of Large Software Systems, Proceedings of the IEEE WESCON, August 1970, pp. 1-9
- Walker Royce
  - Software Project Management, Addison-Wesley, ISBN0-201-30958-0
- Watts Humphrey
  - Managing the Software Process, SEI Series in Software Engineering, Addison Wesley, ISBN 0-201-18095-2

Additional Readings (3)

- George Pólya
  - How to Solve It, 2nd ed., Princeton University Press, 1957, ISBN 0-691-08097-6
- Software Process Engineering Metamodel SPEM 2.0
  - http://www.omg.org/spec/SPEM/2.0/PDF