Lifecycle Modeling

Lecture 15

Problems with Software Development

- Requirements are constantly changing
    - The client might not know all the requirements in advance
- Frequent changes are difficult to manage
    - Identifying checkpoints for planning and cost estimation is difficult
- There is more than one software system
    - New system must often be backward compatible with existing system (“legacy system”)

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

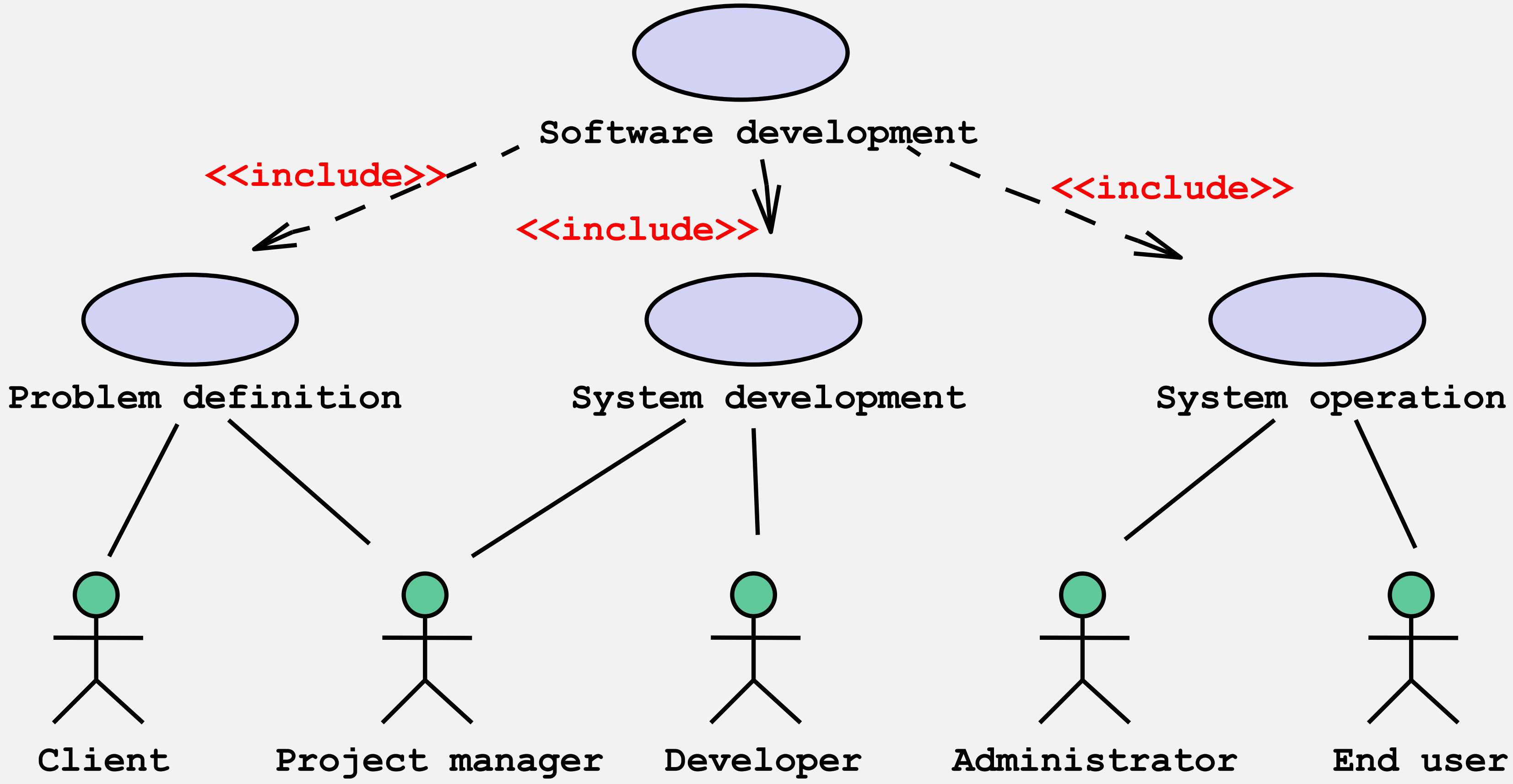
Software life cycle:

Set of activities and their relationships to each other to support the development of a software system

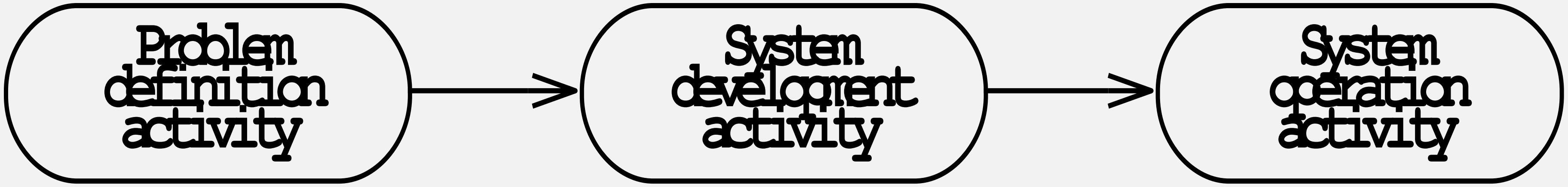
Software development methodology:

A collection of techniques for building models applied across the software life cycle

Functional Model of a simple life cycle model

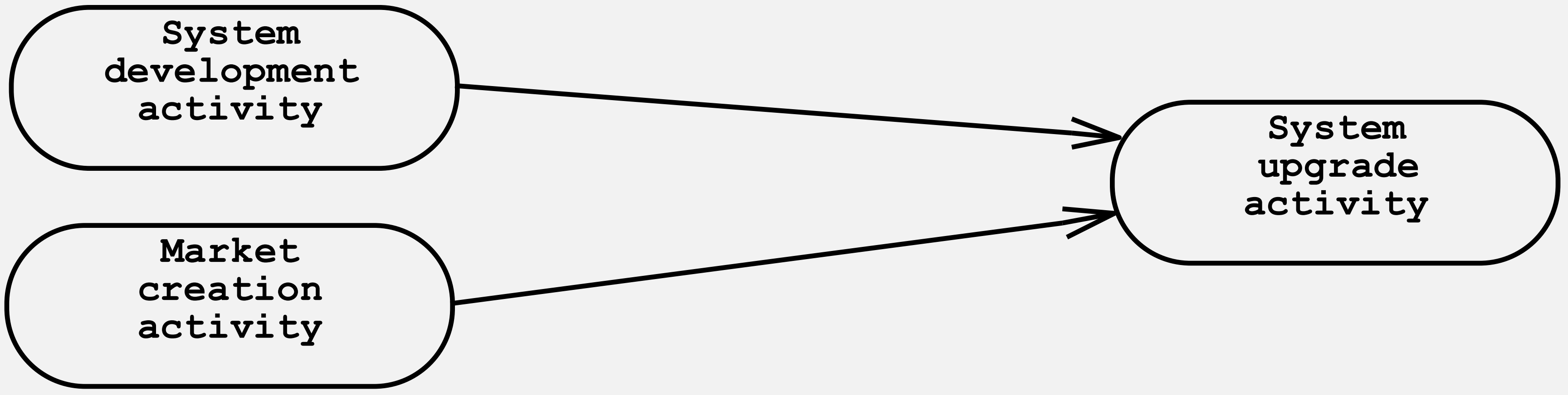


Activity Diagram for t
he same Life Cycle Model


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

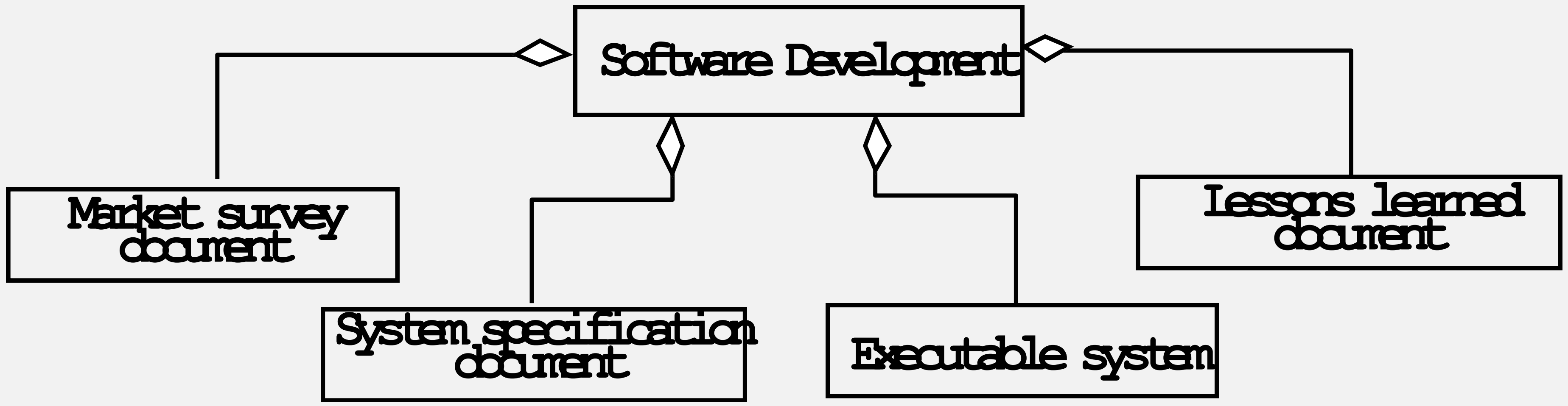


System Development and Market creation can be done in parallel. They must be done before the system upgrade activity

Two Major Views of the Software Life Cycle

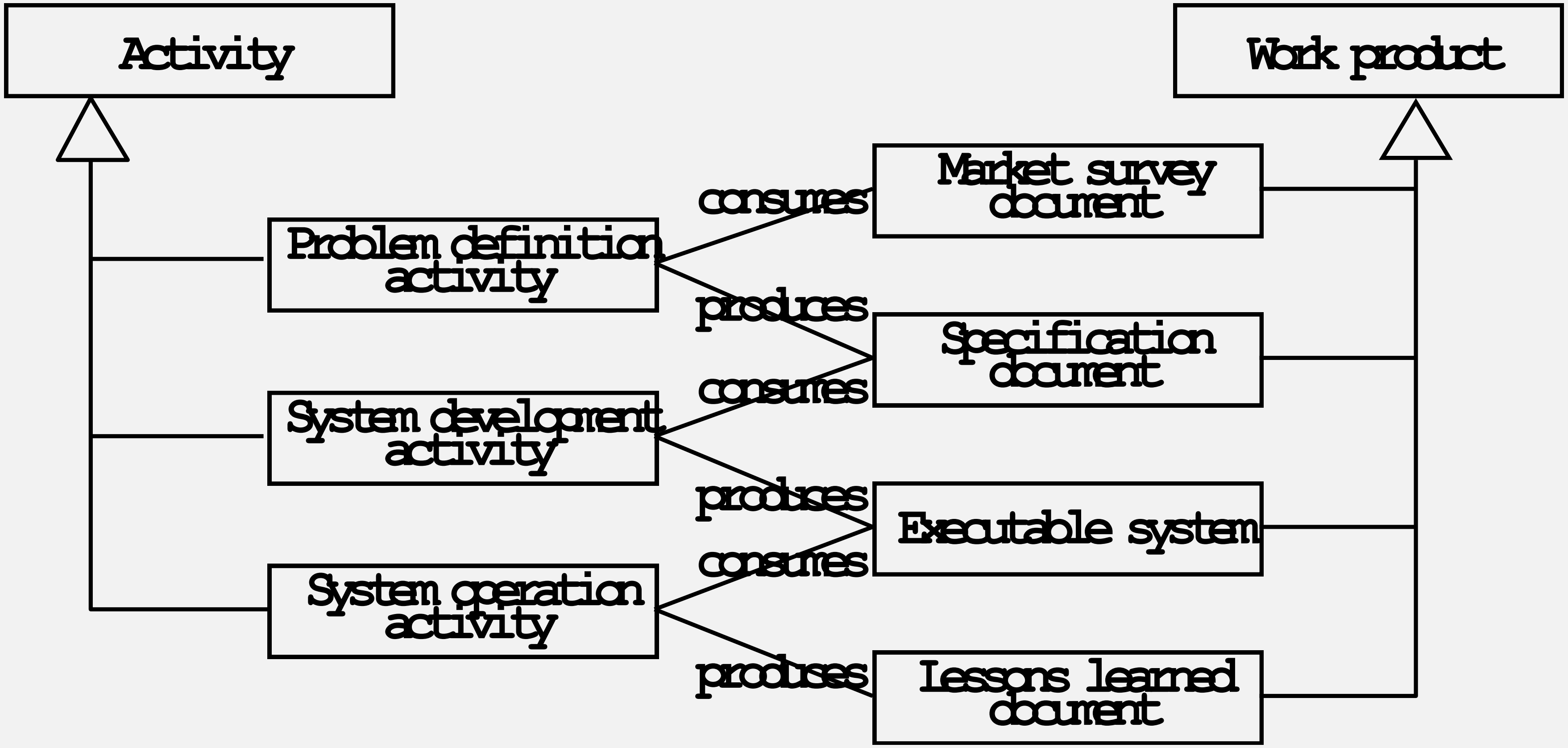
- Activity-oriented view of a software life cycle
  - Software development consists of a set of development activities
  - all the examples so far
- Entity-oriented view of a software life cycle
  - Software development consists of the creation of a set of deliverables.

Entity-centered view of Software Development

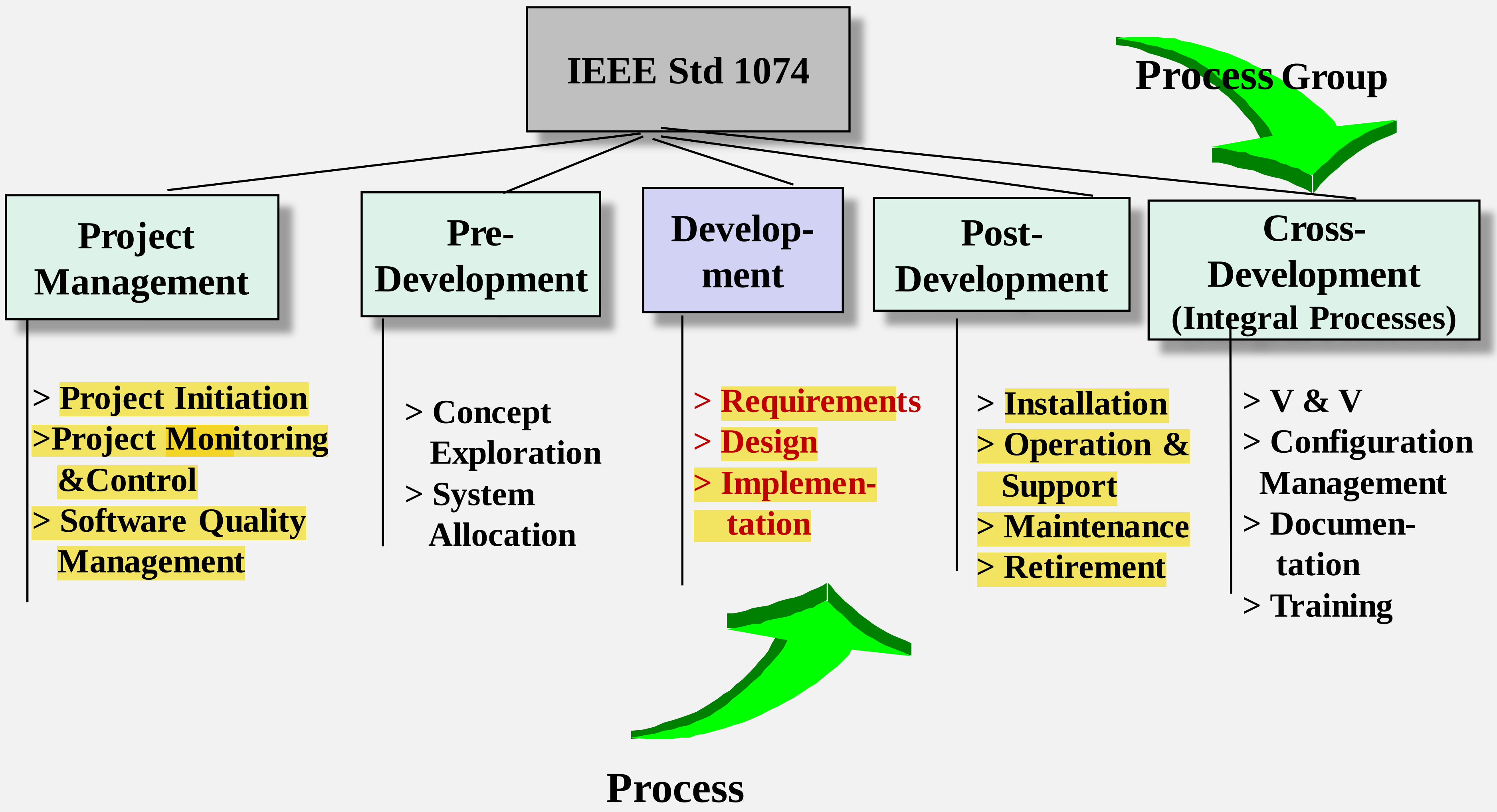


Software development consists of the creation of a set of deliverables

Combining Activities and Entities in One View



IEEE Std 1074: Standard for Software Life Cycle Activities



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

Pre-Development:
- > Concept Exploration
- > System Allocation

Development:
- > Requirements
- > Design
- > Implemen-tation

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

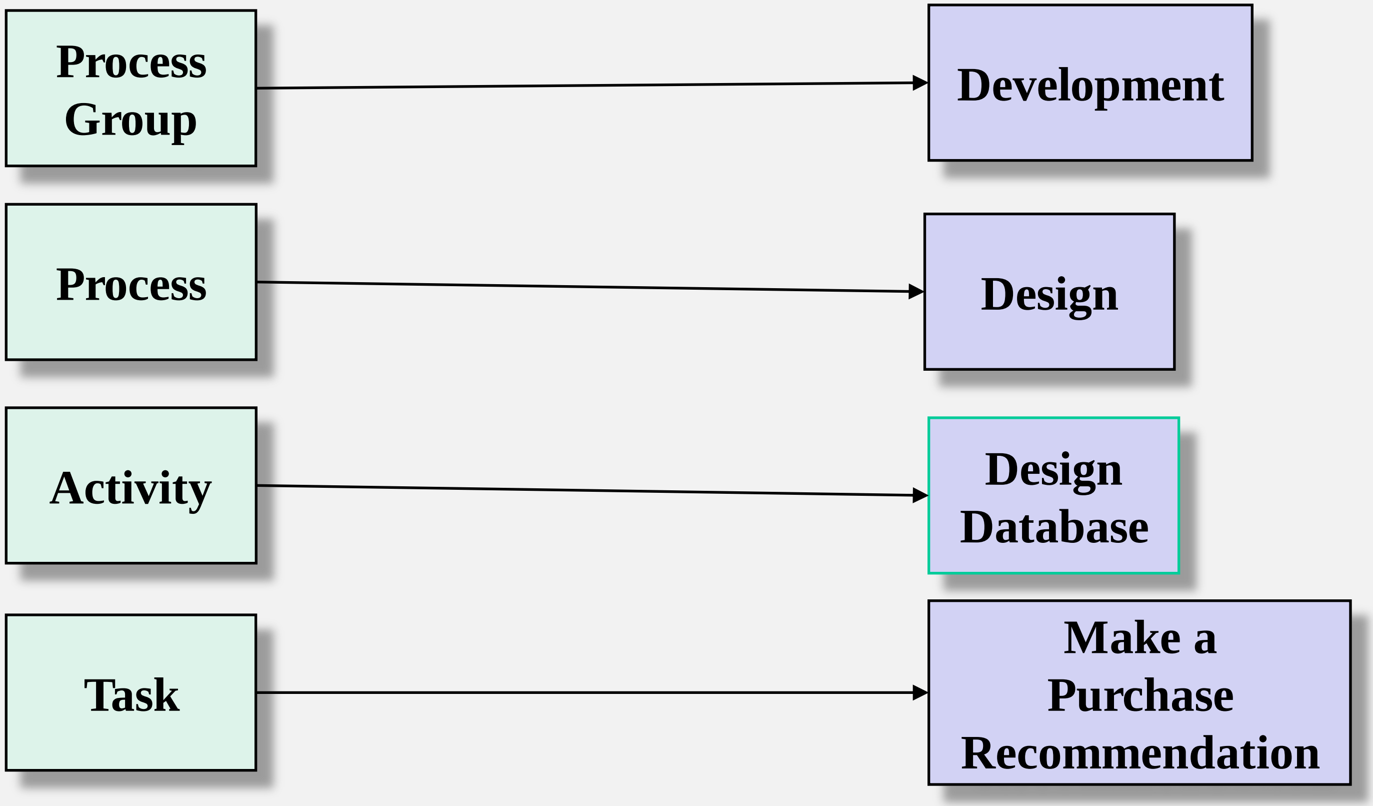
Cross-Development (Integral Processes):
- > V & V
- > Configuration Management
- > Documen-tation
- > Training

Processes, Activities and Tasks

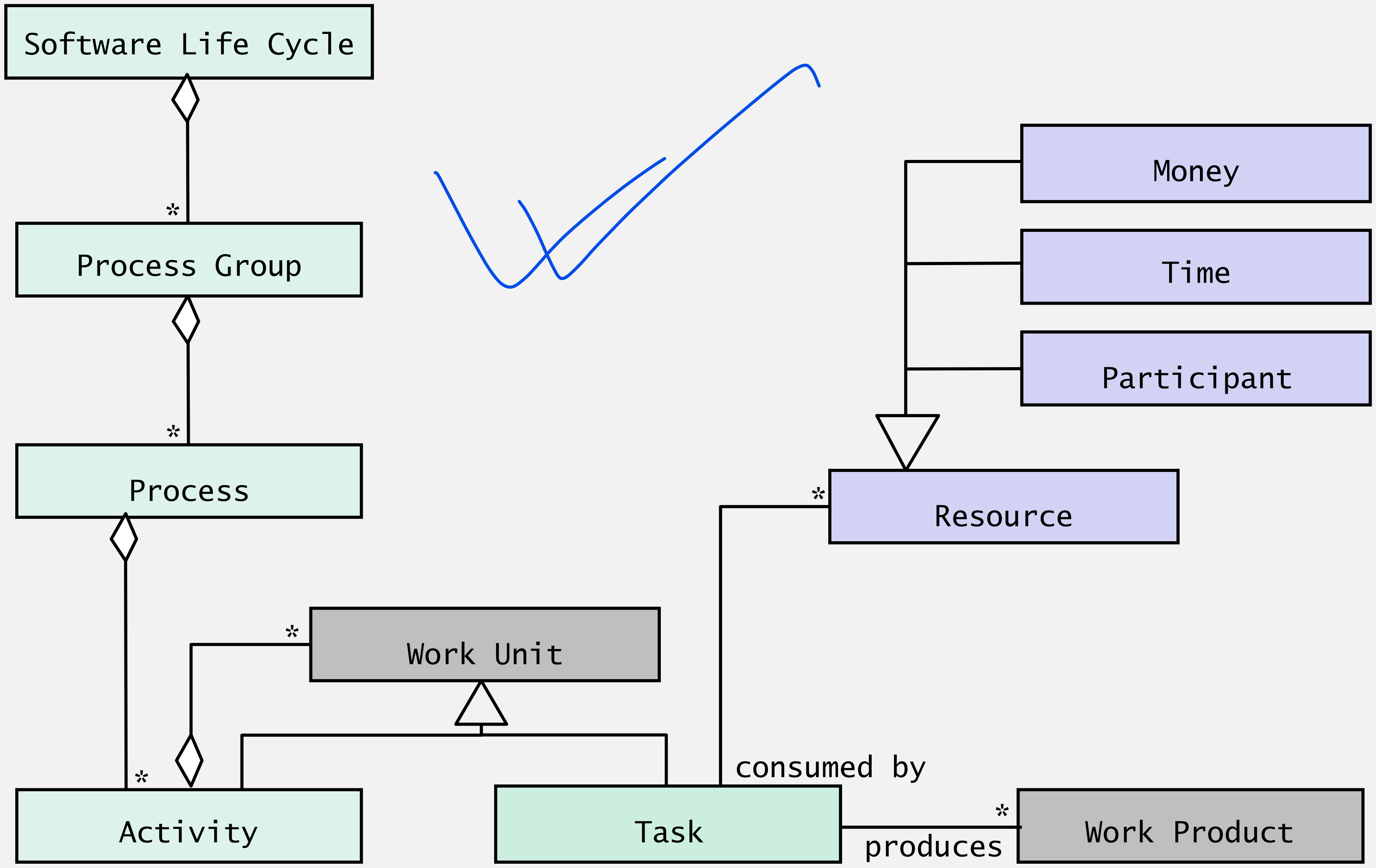
Process Group: Consists of a set of processes

Process: Consists of activities

Activity: Consists of sub activities and tasks



Object Model of the IEEE 1074 Standard



Process Maturity

A software development process is mature

- if the development activities are well defined and
- if management has some control over the quality, budget and schedule of the project

Process maturity is described with

- a set of maturity levels and
- the associated measurements (metrics) to manage the process

Assumption:

- With increasing maturity the risk of project failure decreases

CMM: Capability Maturity Model

CMM levels

1. Initial Level

   also called ad hoc or chaotic

2. Repeatable Level

   Process depends on individuals ("champions")

3. Defined Level

   Process is institutionalized (sanctioned by management)

4. Managed Level

   Activities are measured and provide feedback for resource allocation (process itself does not change)

5. Optimizing Level

   Process allows feedback of information to change process itself

What does Process Maturity Measure?

The real indicator of process maturity is the level of predictability of project performance (quality, cost, schedule).

Level 1: Random, unpredictable performance

Level 2: Repeatable performance from project to project

Level 3: Better performance on each successive project

Level 4: Substantial improvement (order of magnitude) in one dimension of project performance

Level 5: Substantial improvements across all dimensions of project performance.

Key Process Areas

To achieve a specific level of maturity, the organization must demonstrate that it addresses all the key process areas defined for that level.

There are no key process areas for Level 1

KPA Level 2: Basic software project management practice

KPA Level 3: Infrastructure for single software life cycle model

KPA Level 4: Quantitative understanding of process and deliverables

KPA Level 5: Keep track of technology and process changes

Pros and Cons of Process Maturity

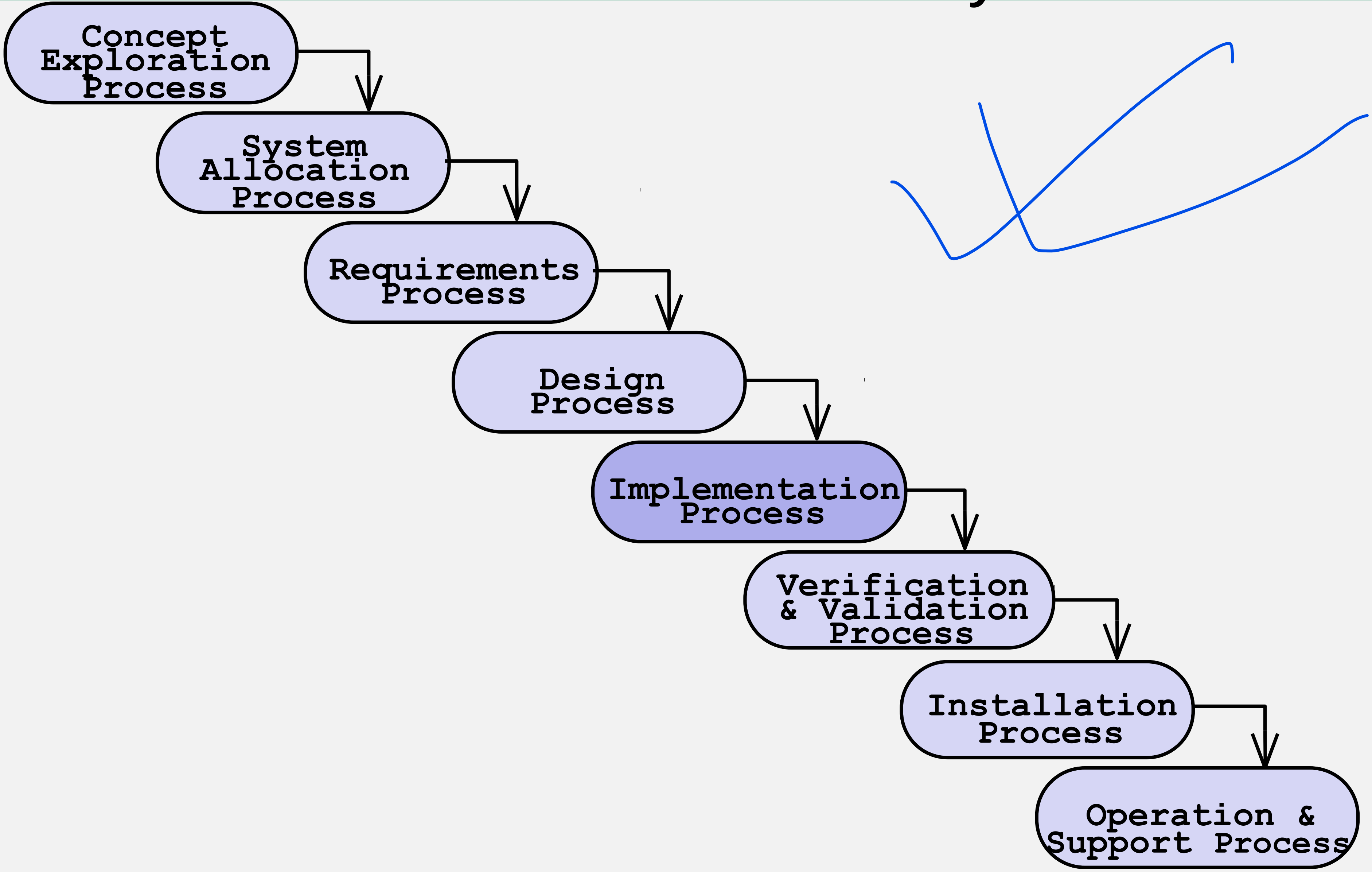
Benefits:

- Increased control of projects
- Predictability of project cost and schedule
- Objective evaluations of changes in techniques, tools and methodologies
- Predictability of the effect of a change on project cost or schedule

Problems:

- Need to watch a lot (“Big brother“, „big sister“)
- Overhead to capture, store and analyse the required information

The Waterfall Model of the Software Life Cycle



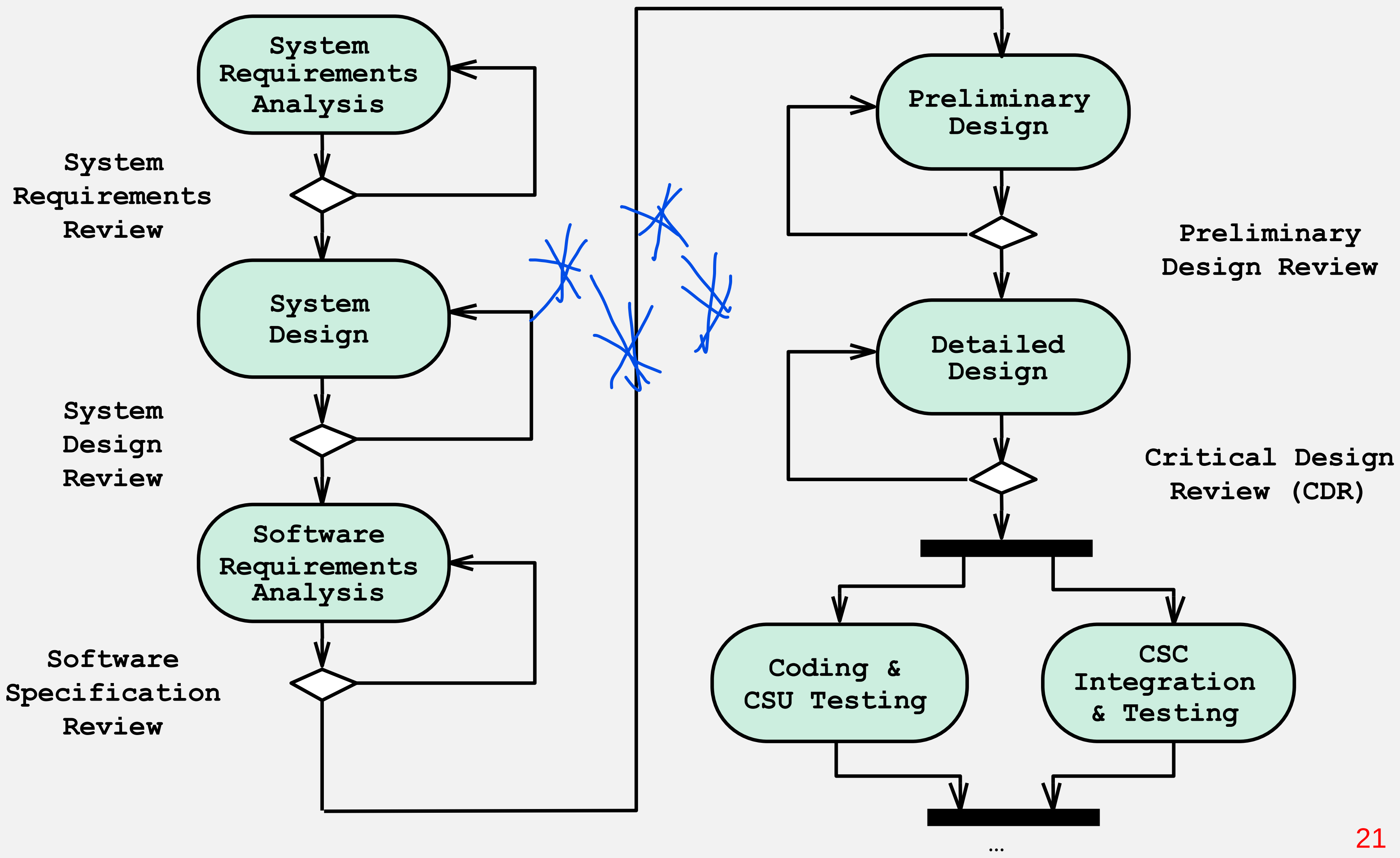
Concept Exploration Process
System Allocation Process
Requirements Process
Design Process
Implementation Process
Verification & Validation Process
Installation Process
Operation & Support Process

Example of a waterfall model : DOD Standard 2167A

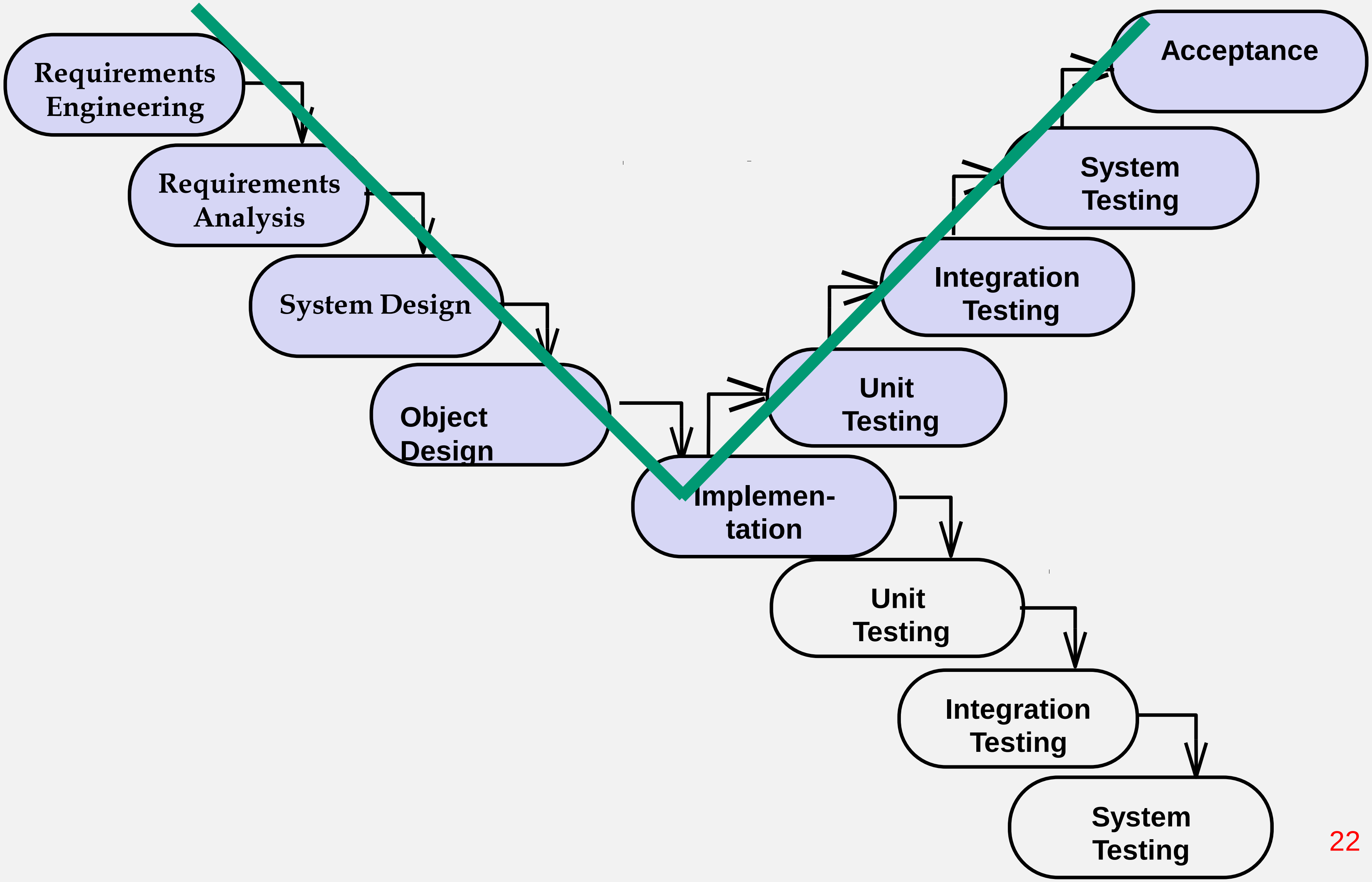
Software development activities:

- System Requirements Analysis/Design
- Software Requirements Analysis
- Preliminary Design and Detailed Design
- Coding and Unit testing
- Integration Testing
- System integration and Testing

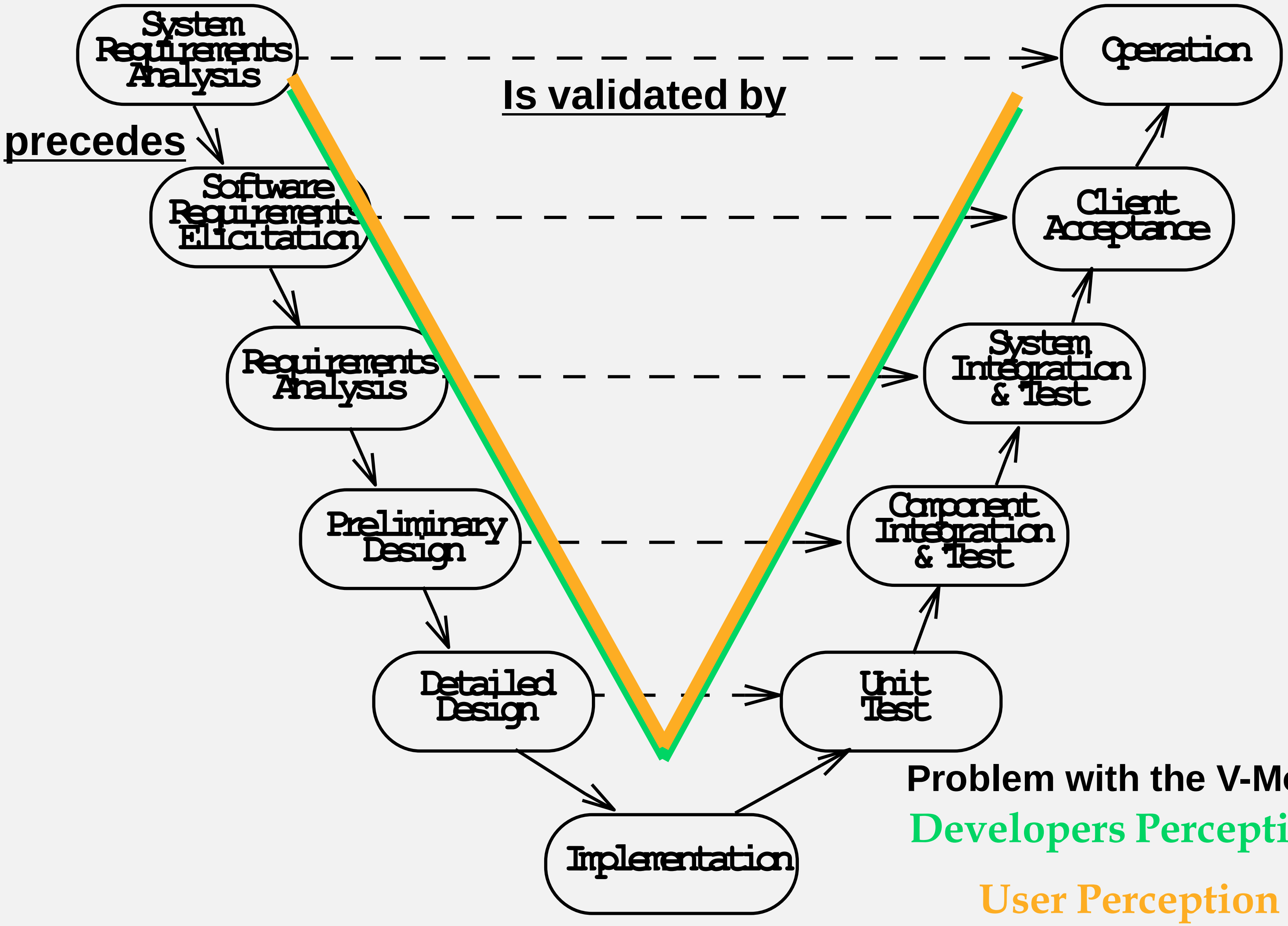
Activity Diagram of MIL DOD-STD-2167A


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.

Spiral Model

The spiral model proposed by Boehm has 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

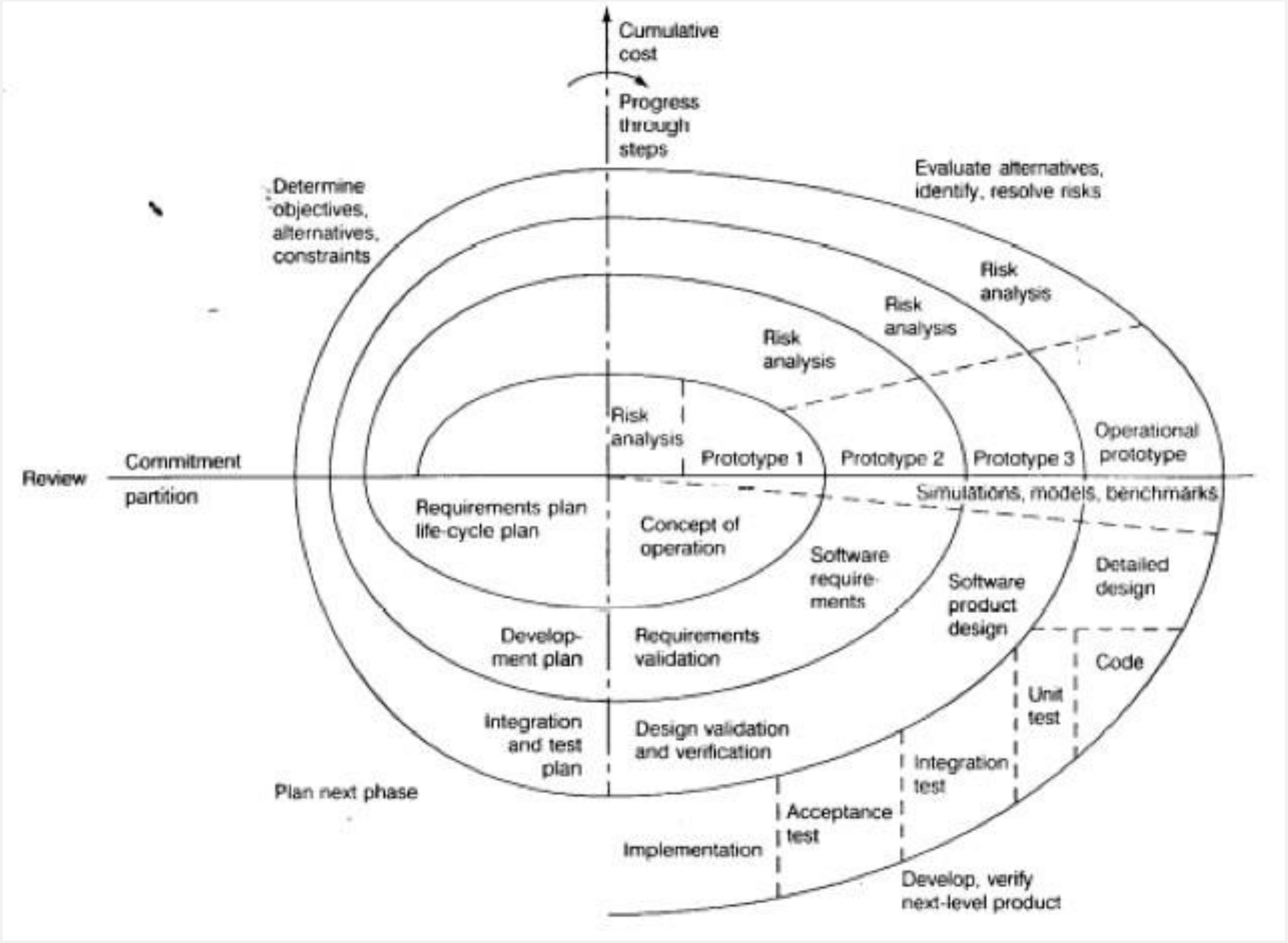
Acceptance Test

Implementation

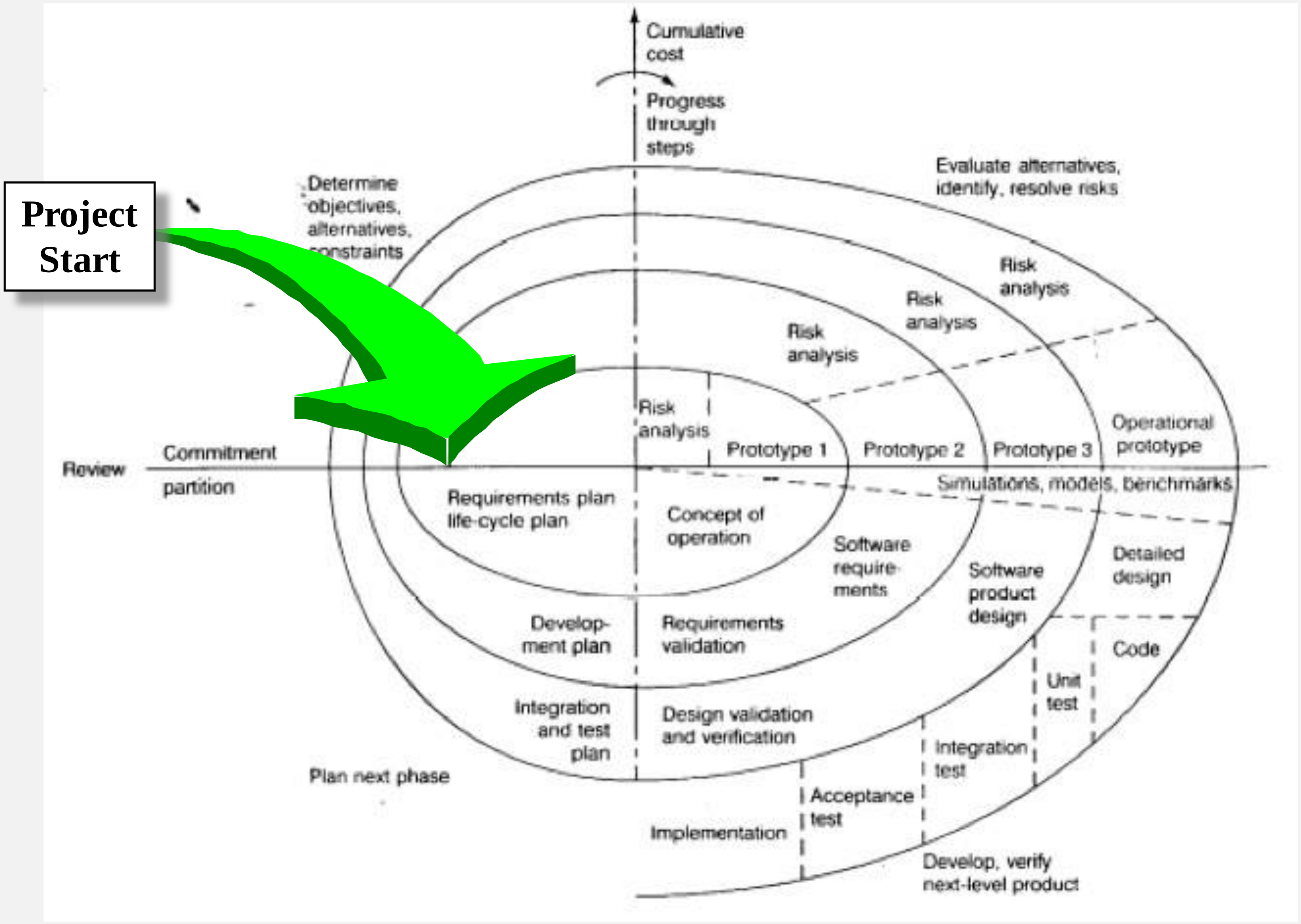
For each round go through these activities:

- Define objectives, alternatives, constraints
- Evaluate alternatives, identify and resolve risks
- Develop and verify a prototype
- Plan the next round.

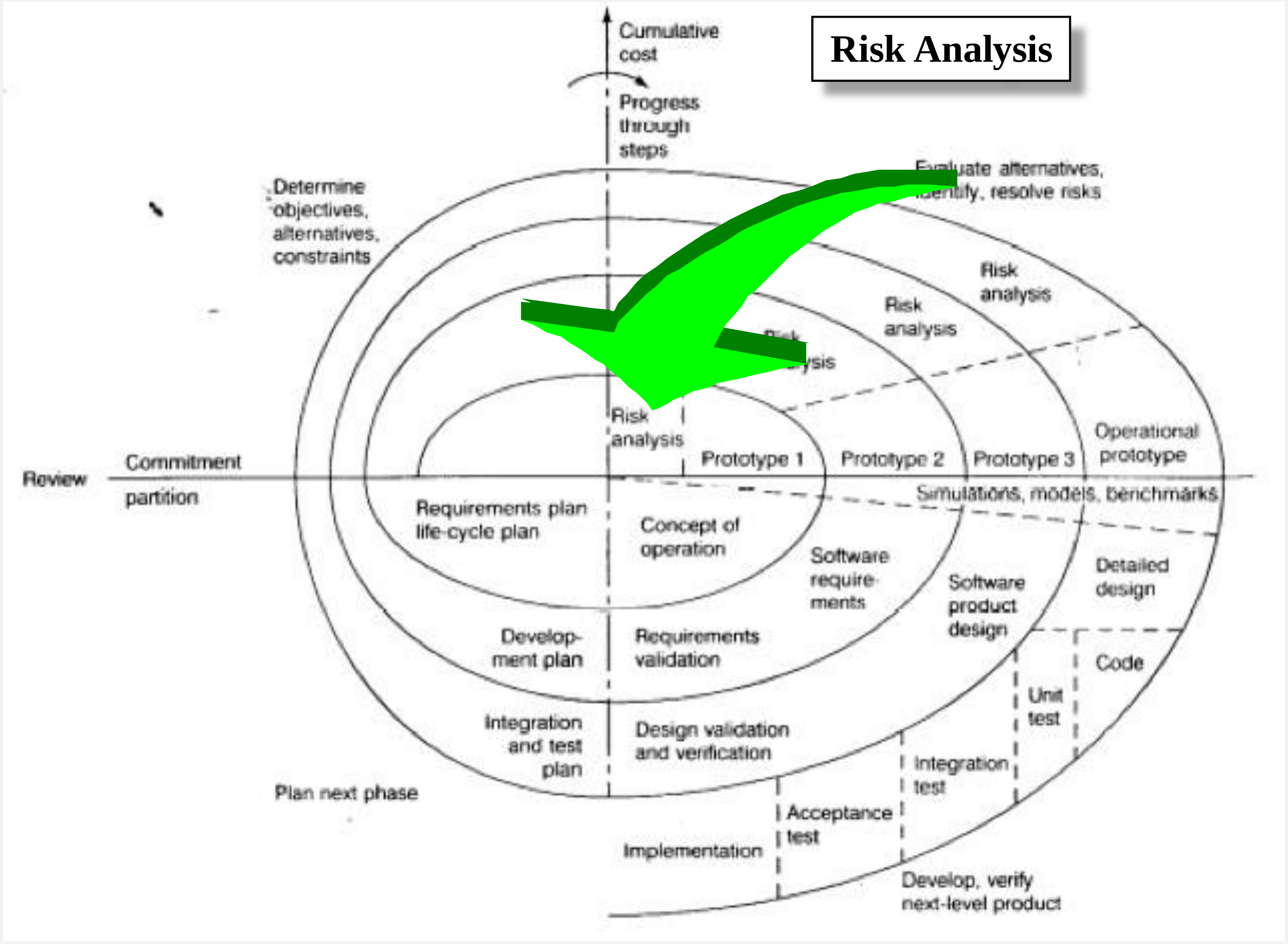
Diagram of Boehm's Spiral Model



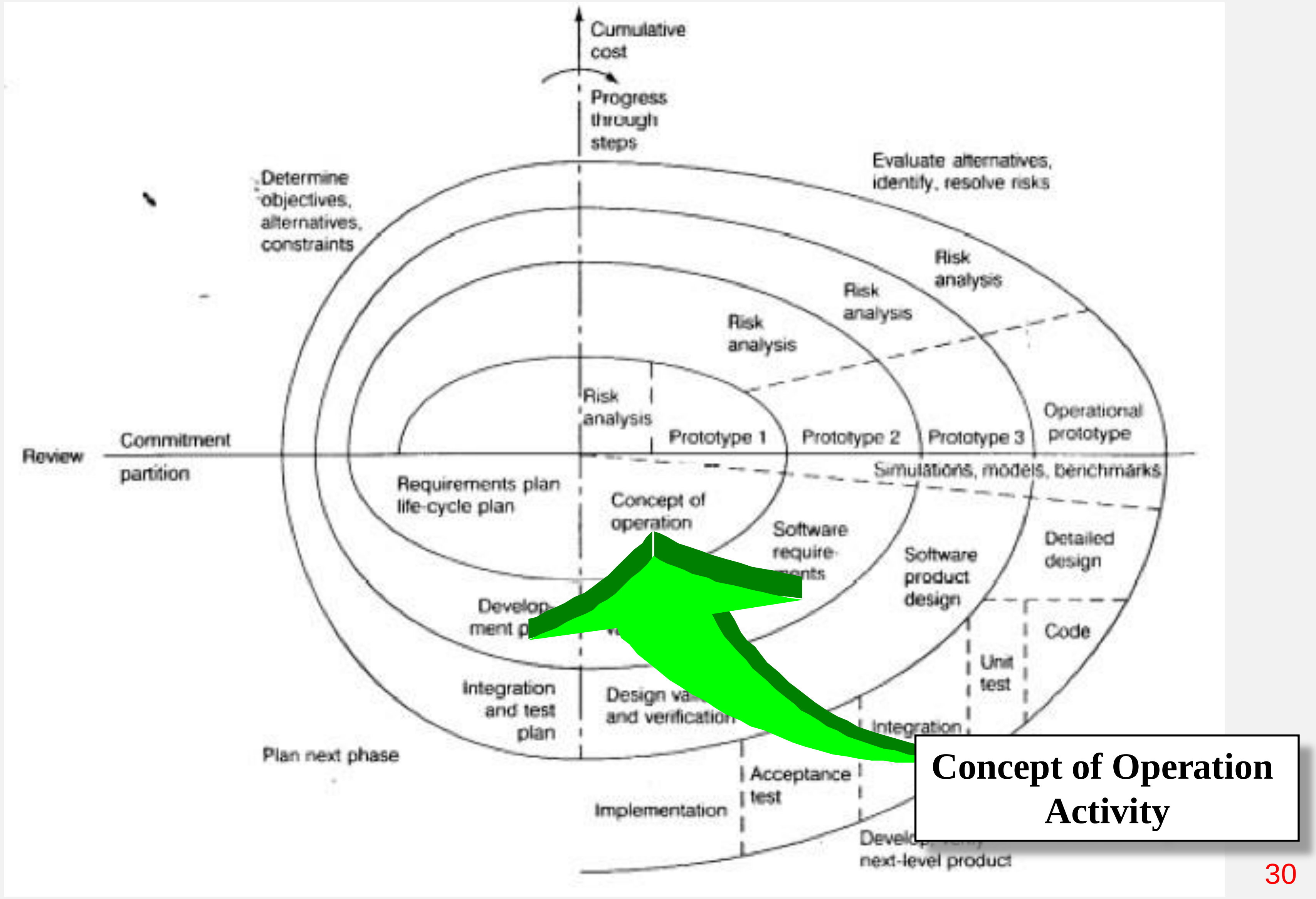
Round 1, Concept of Operations, Quadrant IV: Determine Objectives,Alternatives & Constraints



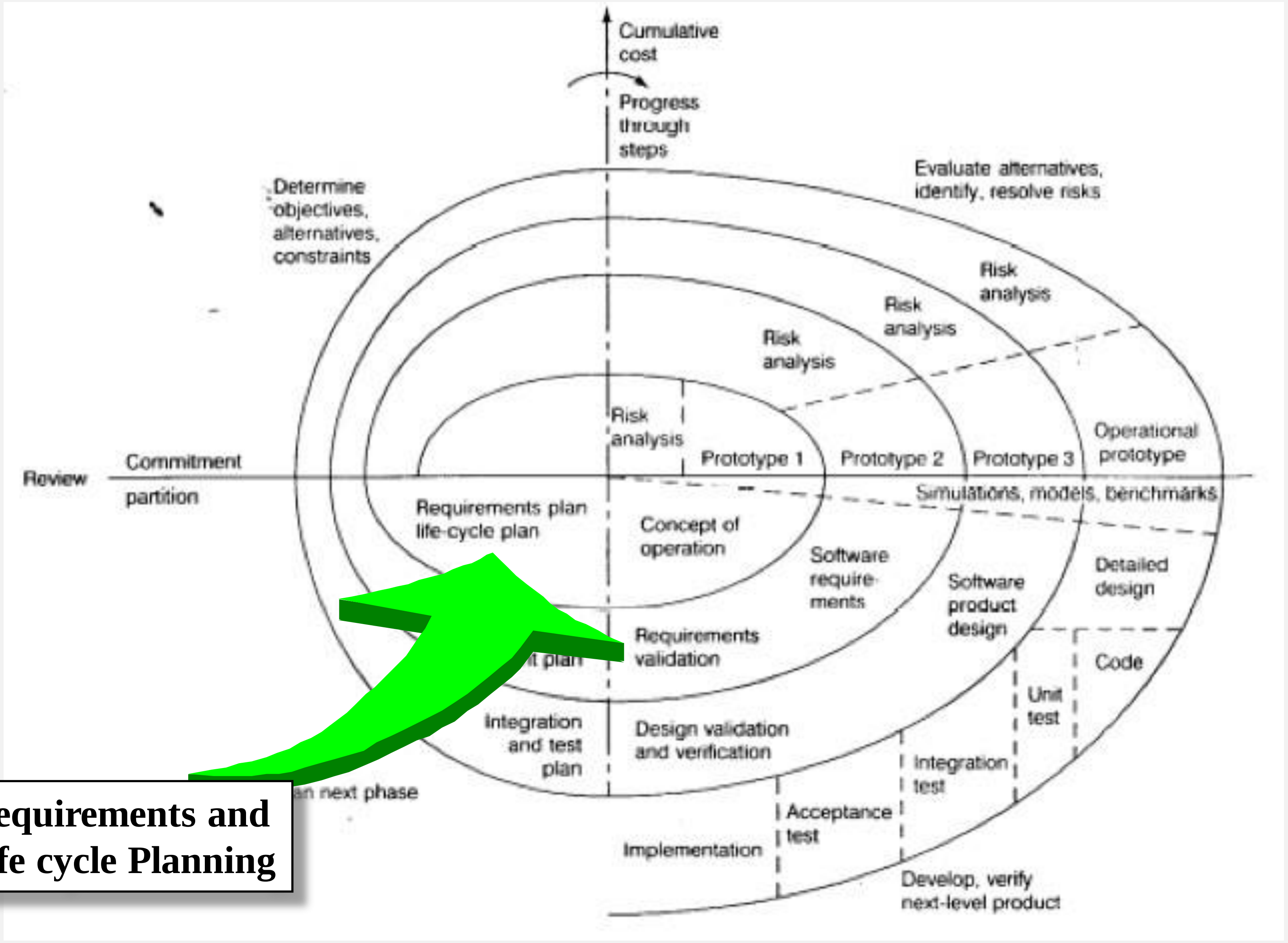
Round 1, Concept of Operations, Quadrant I: Evaluate Alternatives, identify & resolve Risks



Round 1, Concept of Operations, Quadrant II: Develop and Verify



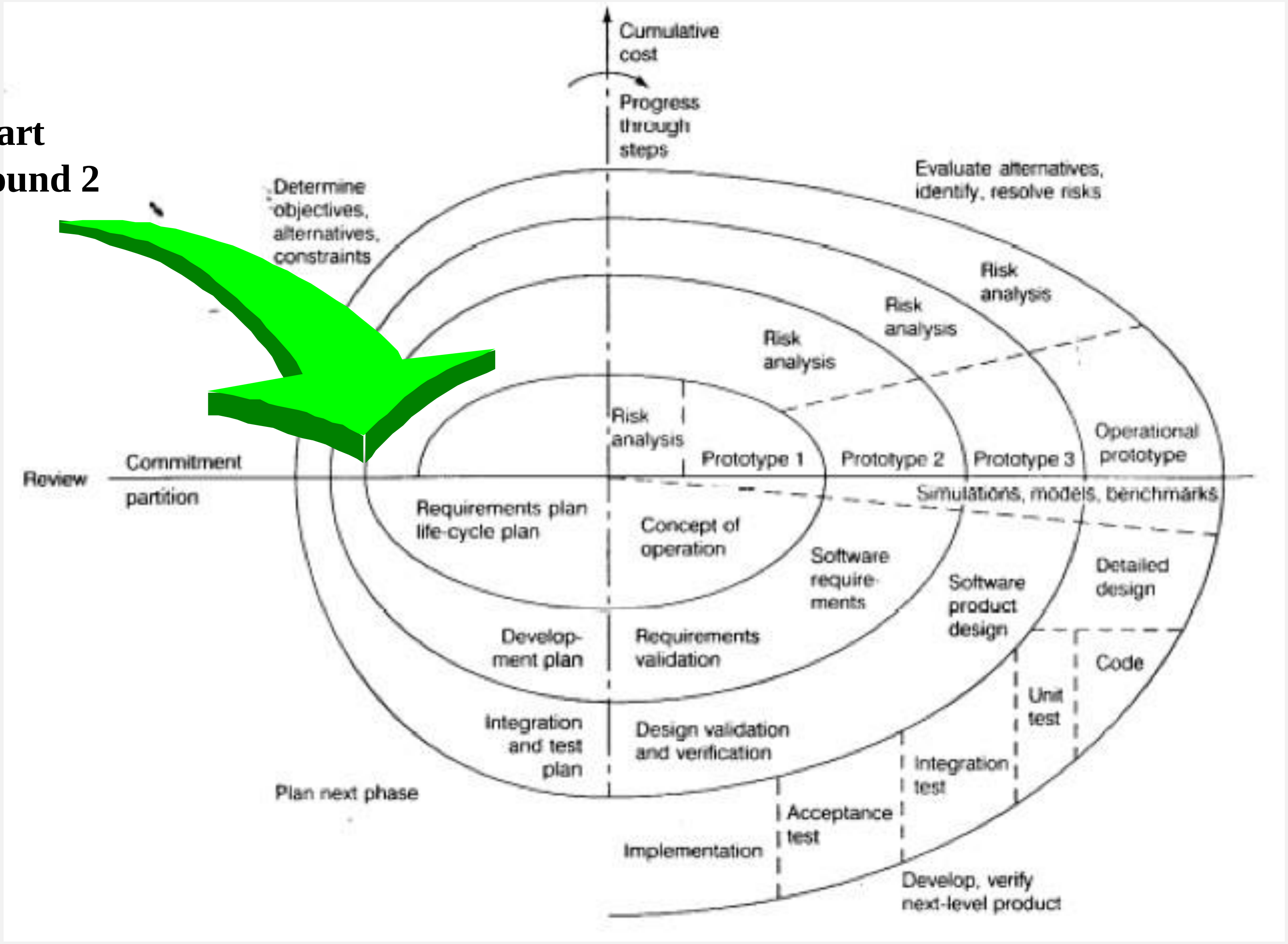
Round 1, Concept of Operations, Quadrant III: Prepare for Next Activity



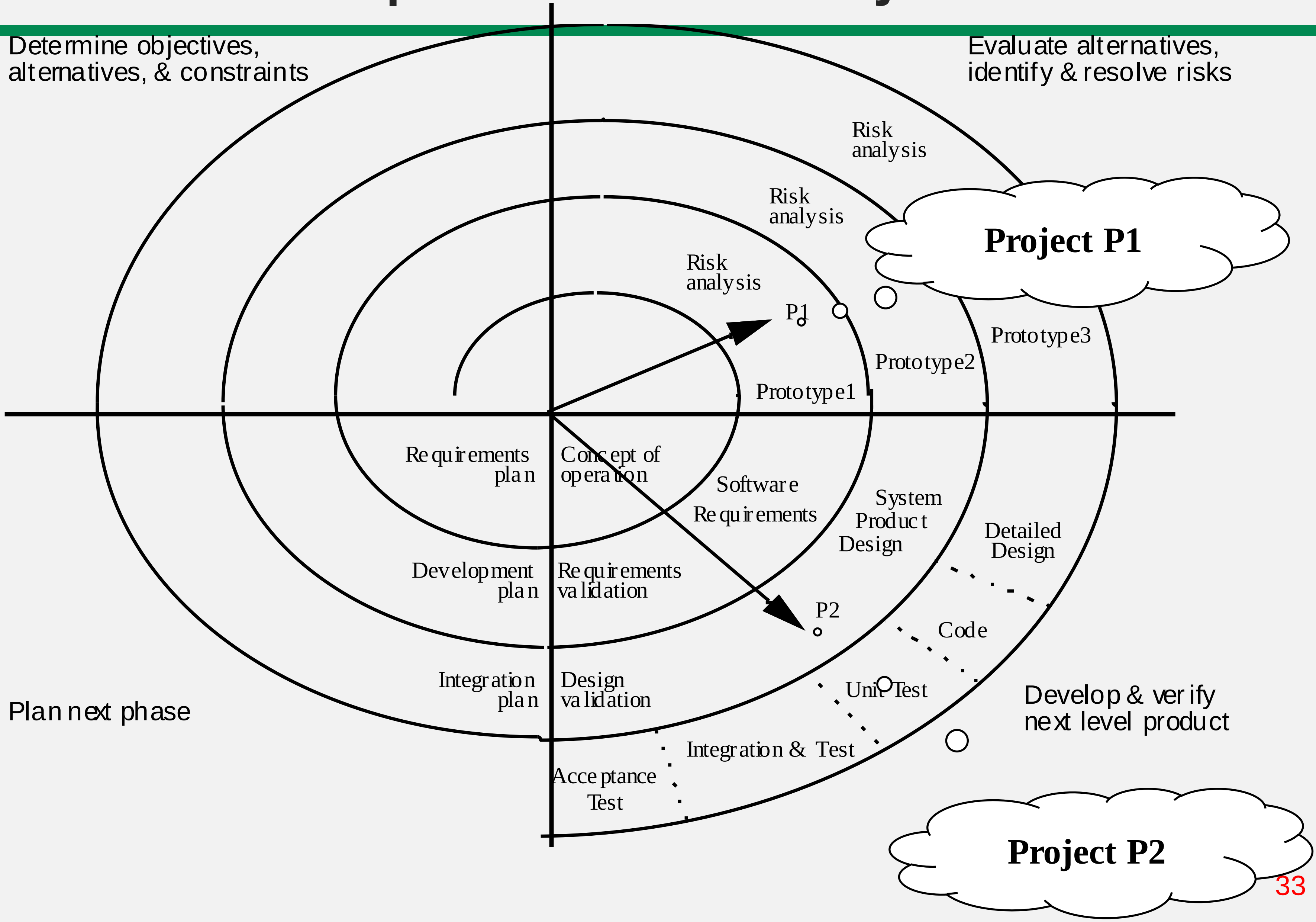
Requirements and Life cycle Planning

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

Start of Round 2



Comparison of Projects



Limitations of Waterfall and Spiral 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 phases, but does not allow change within a phase

What do you do if change is happening more frequently?

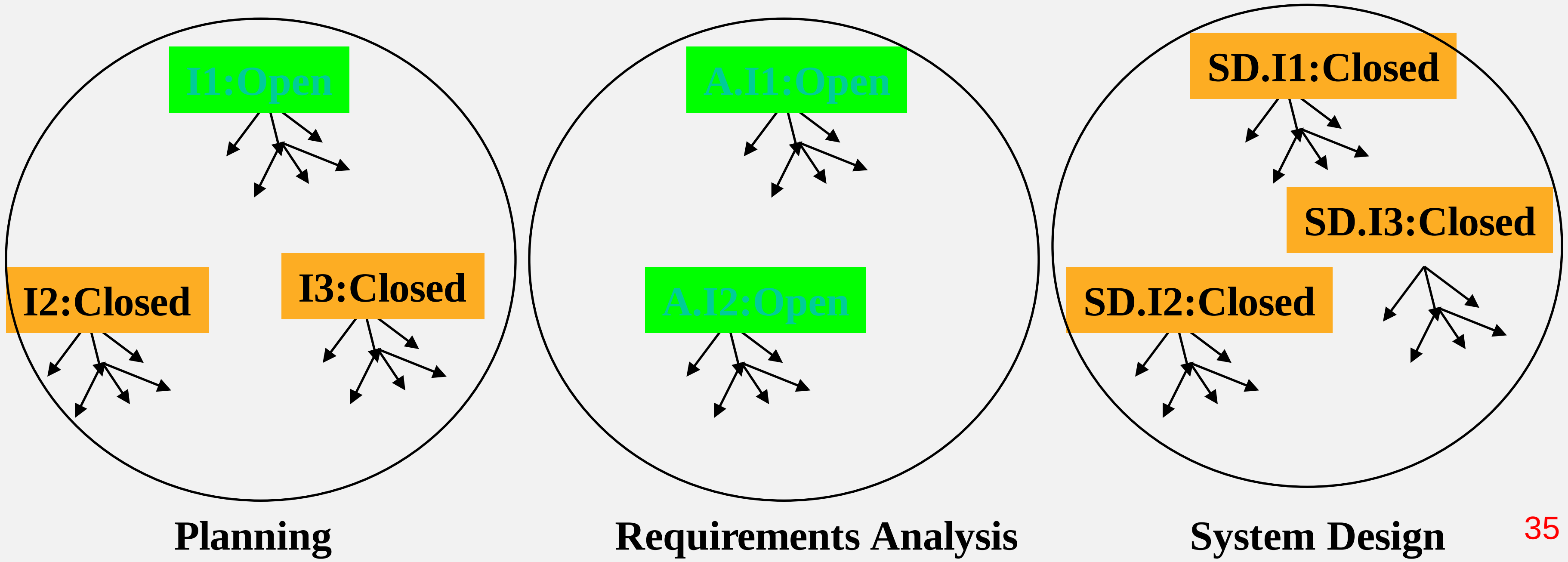
“The only constant is the change”

An Alternative: Issue-Based Development

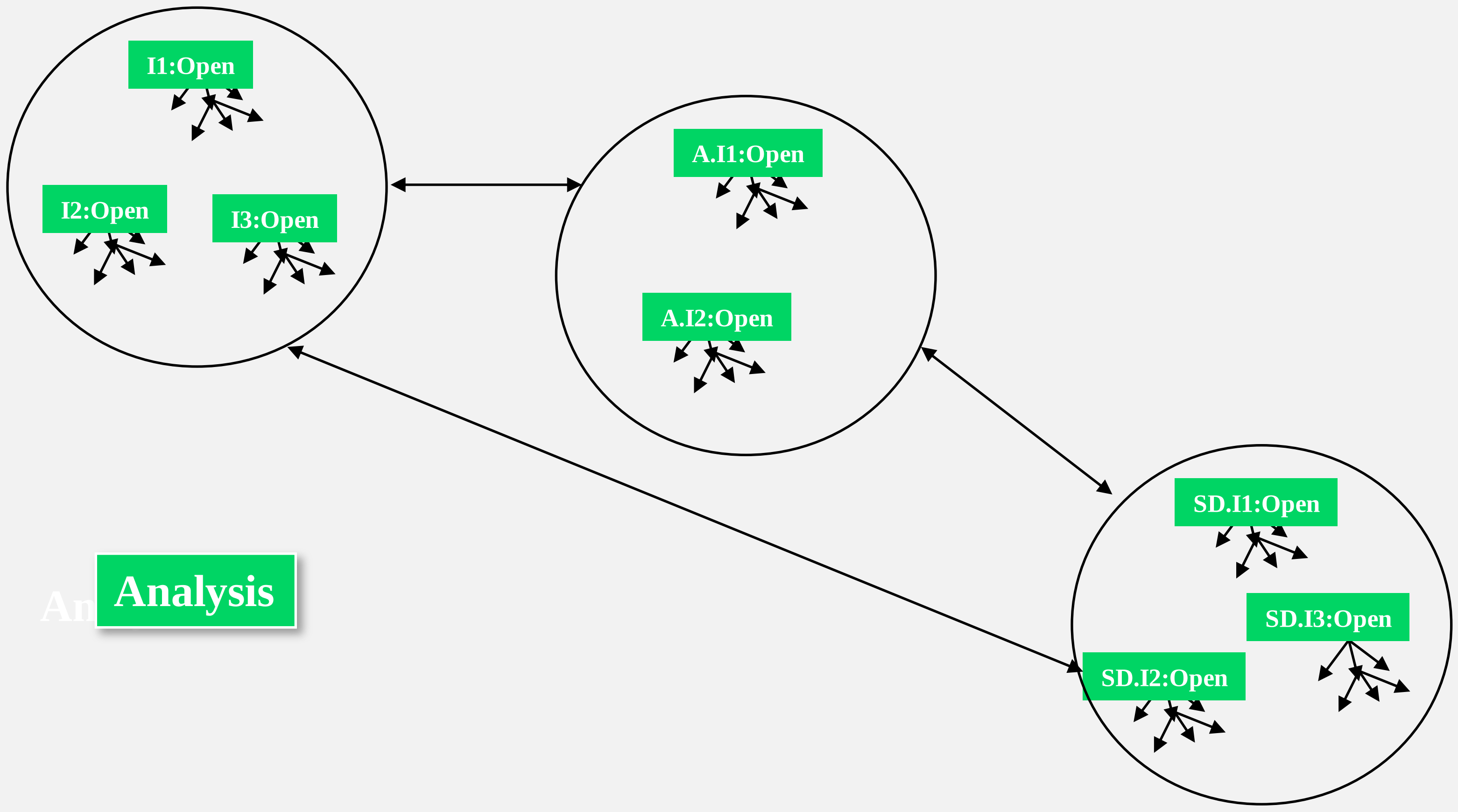
A system is described as a collection of issues

- Issues are either closed or open
- Closed issues have a resolution
- Closed issues can be reopened (Iteration!)

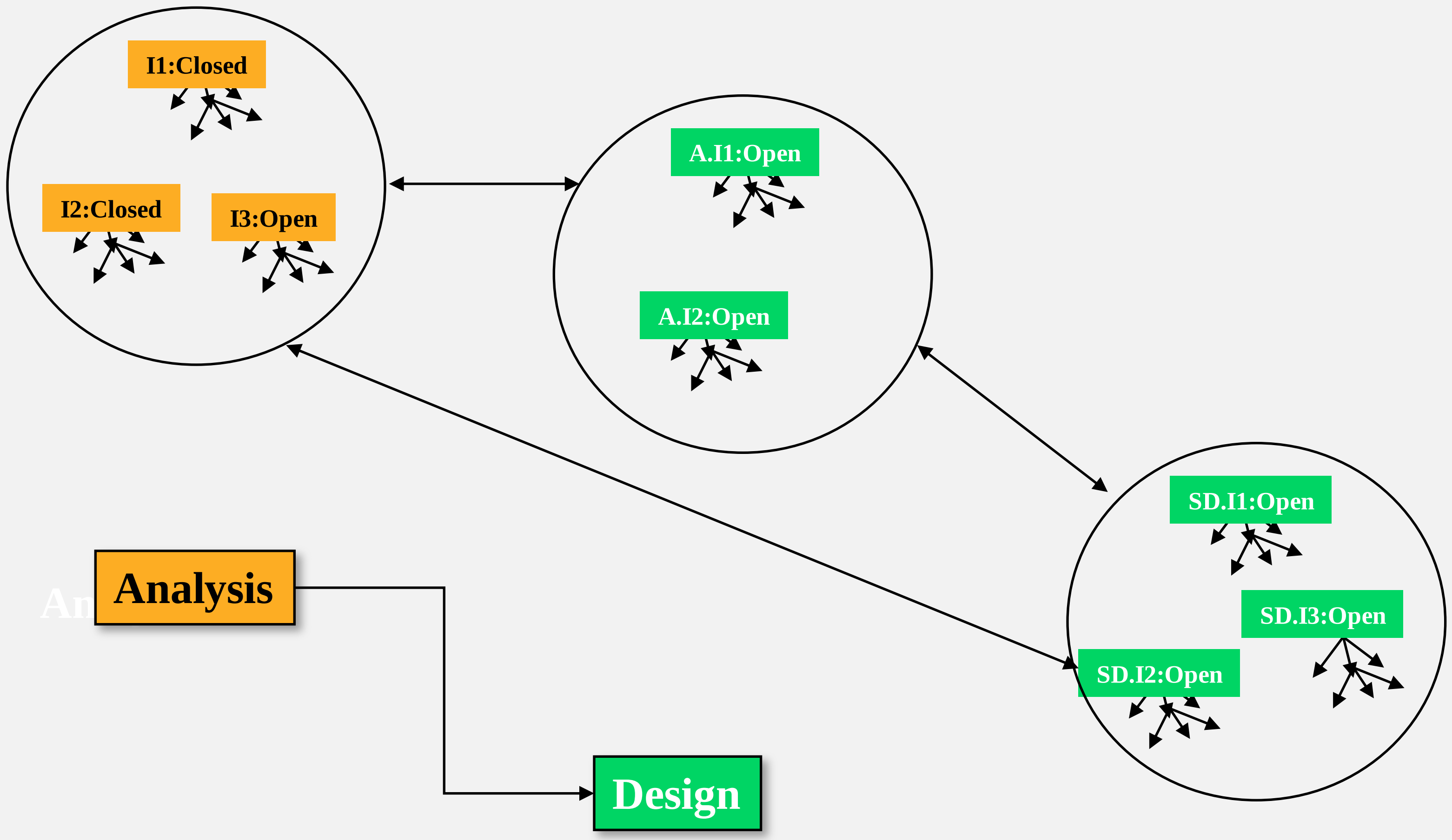
The set of closed issues is the basis of the system model



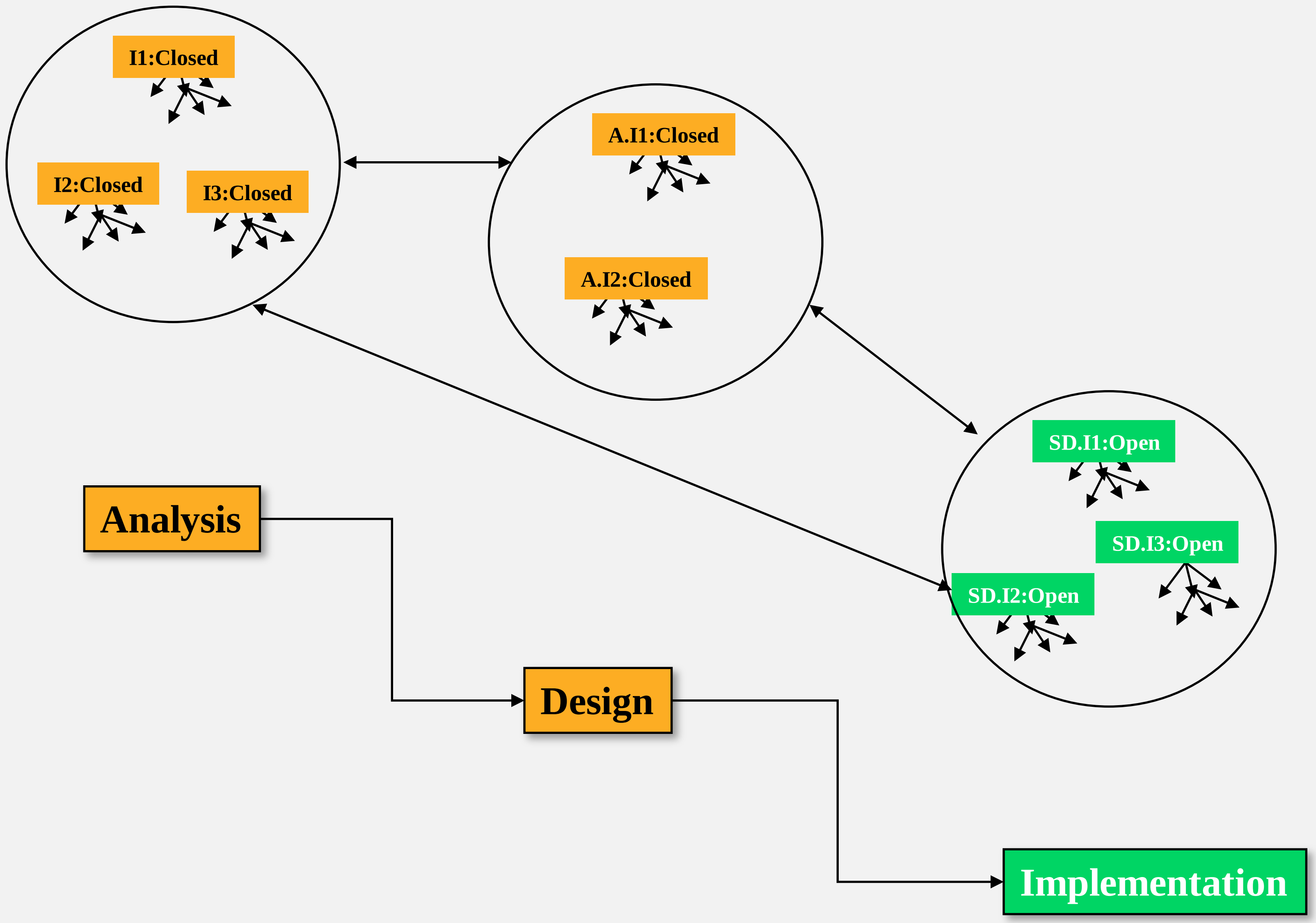
Waterfall Model: Analysis Phase



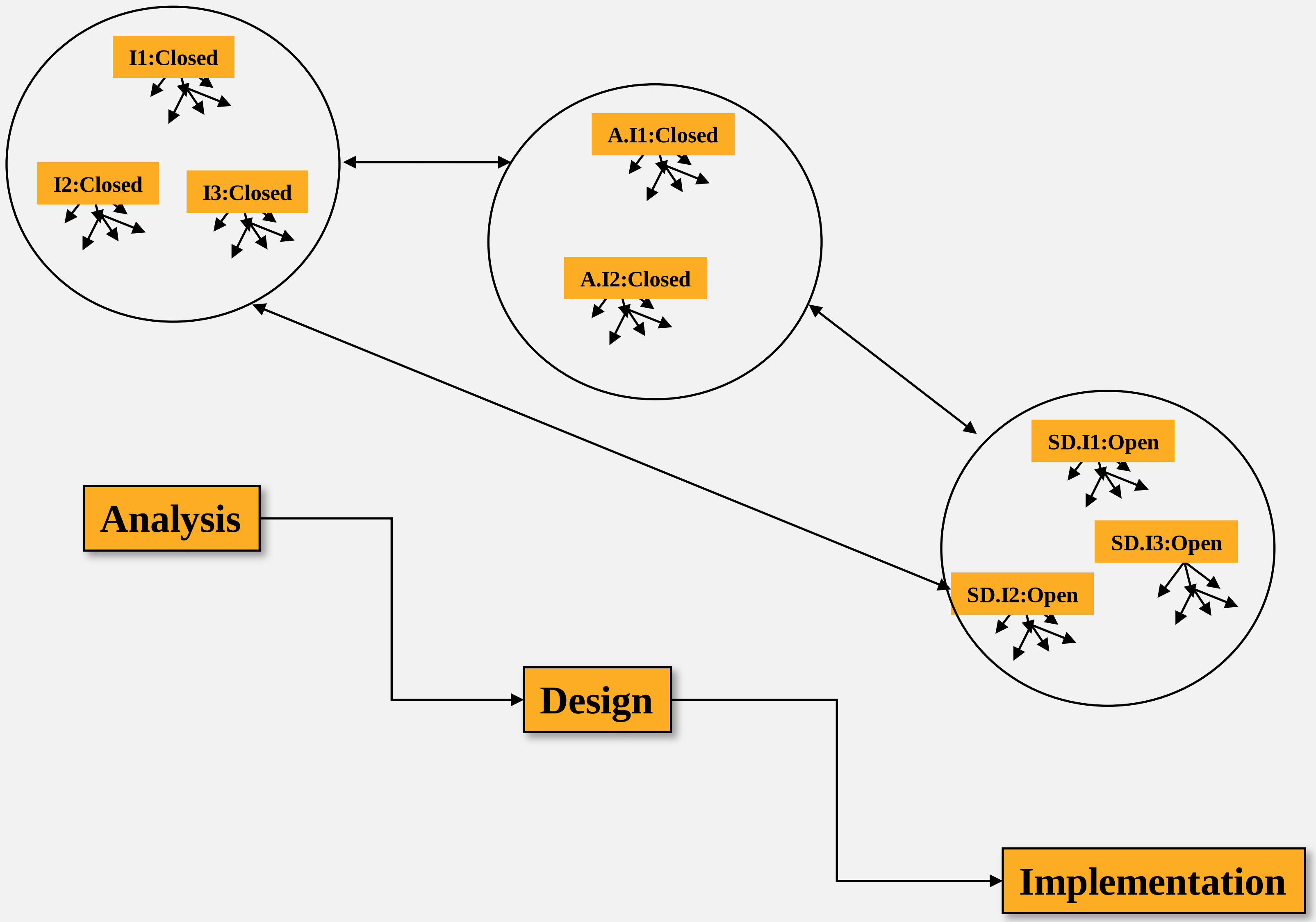
Waterfall Model: Design Phase



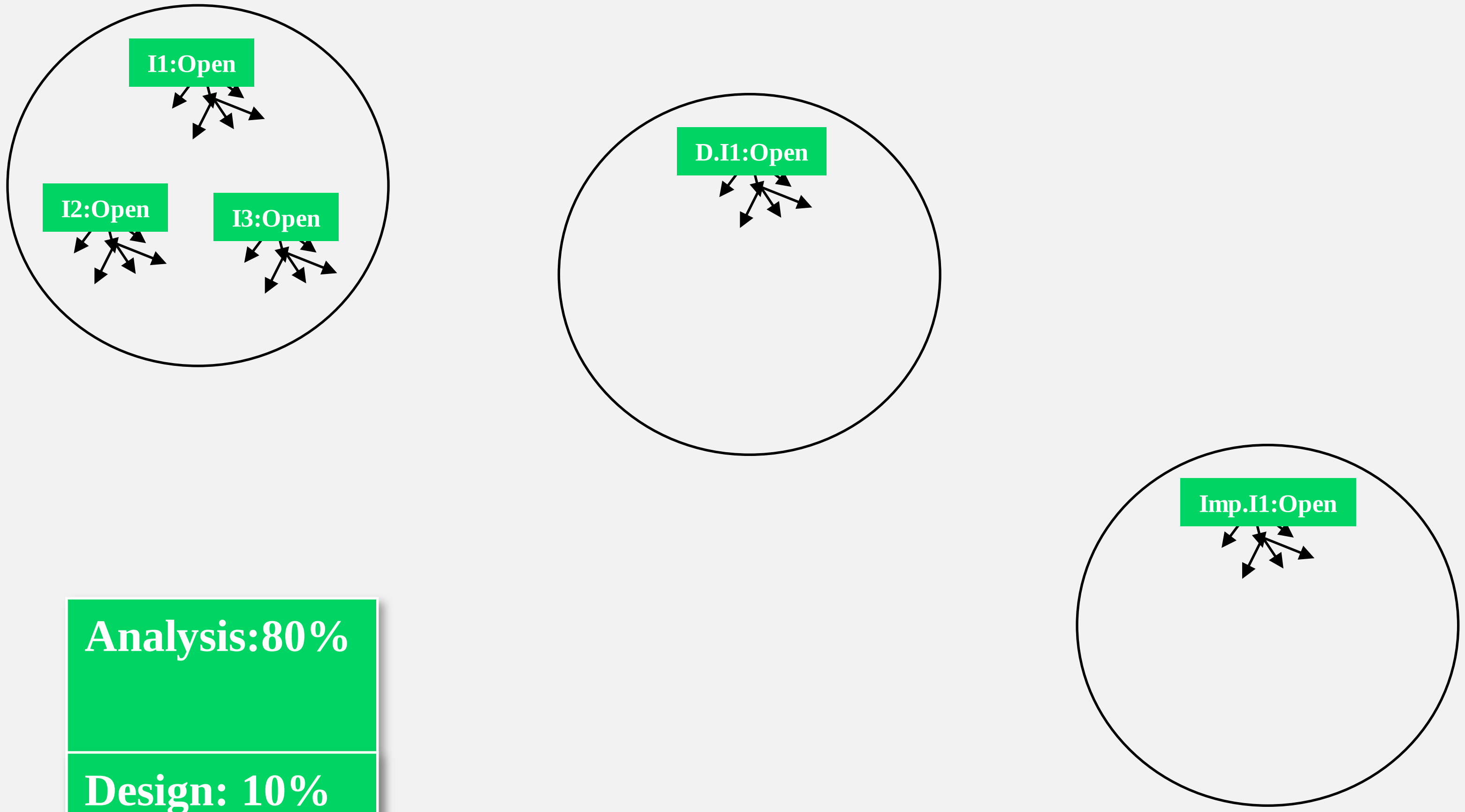
Waterfall Model: Implementation Phase



Waterfall Model: Project is Done



Issue-Based Model: Analysis Phase

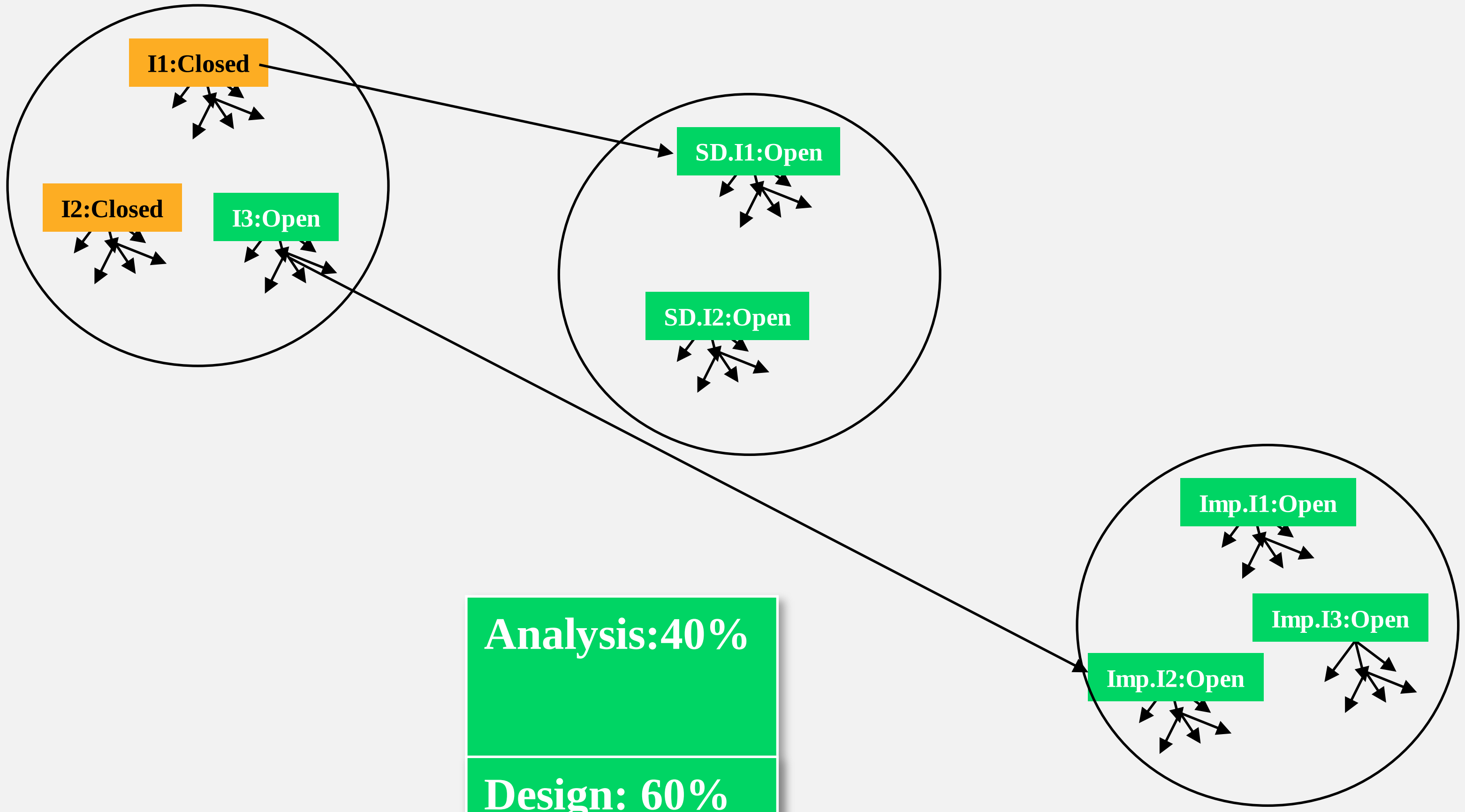


Analysis:80%

Design: 10%

Implemen-
tation: 10%

Issue-Based Model: Design Phase

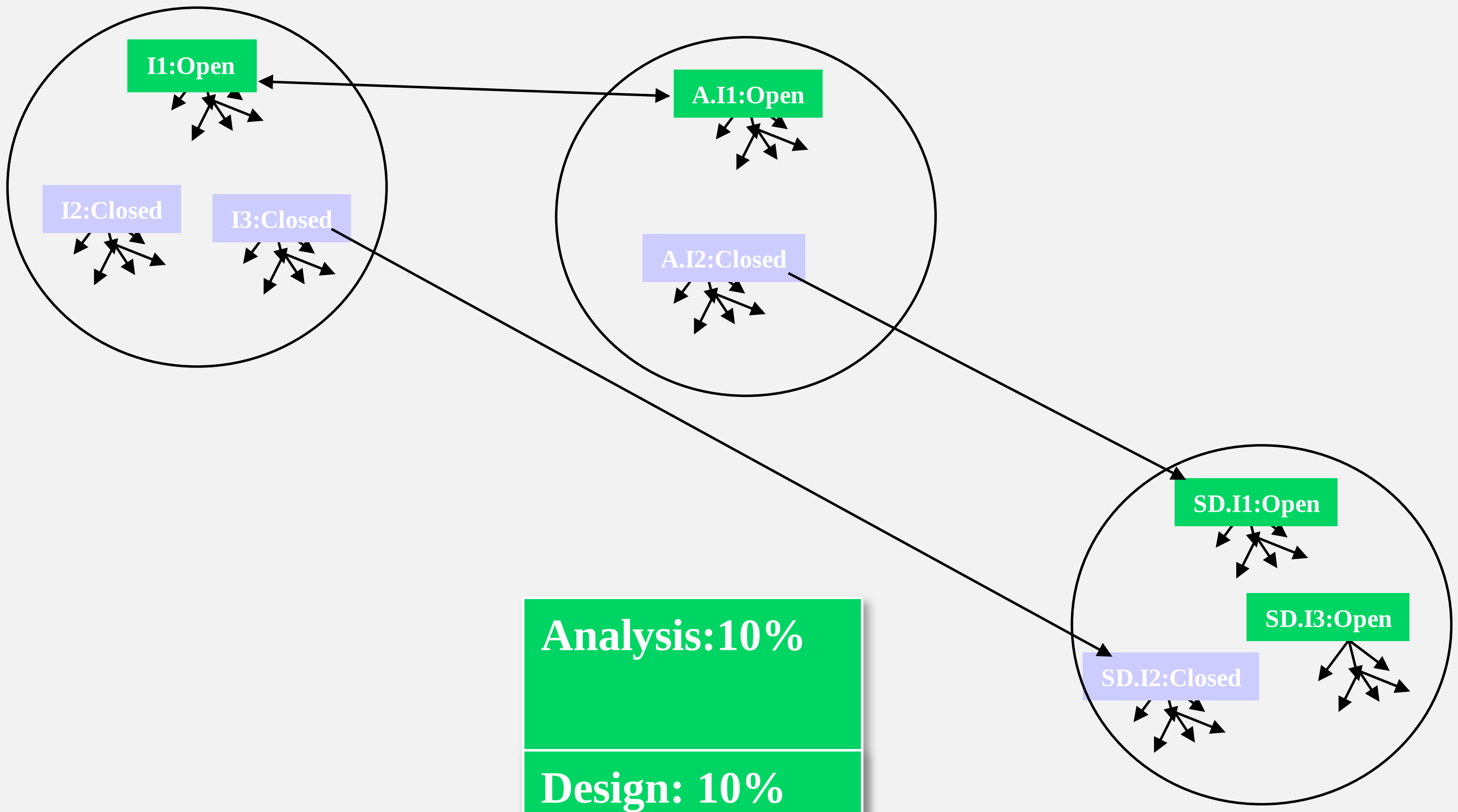


Analysis:40%

Design: 60%

Implementation: 0%

Issue-Based Model: Implementation Phase

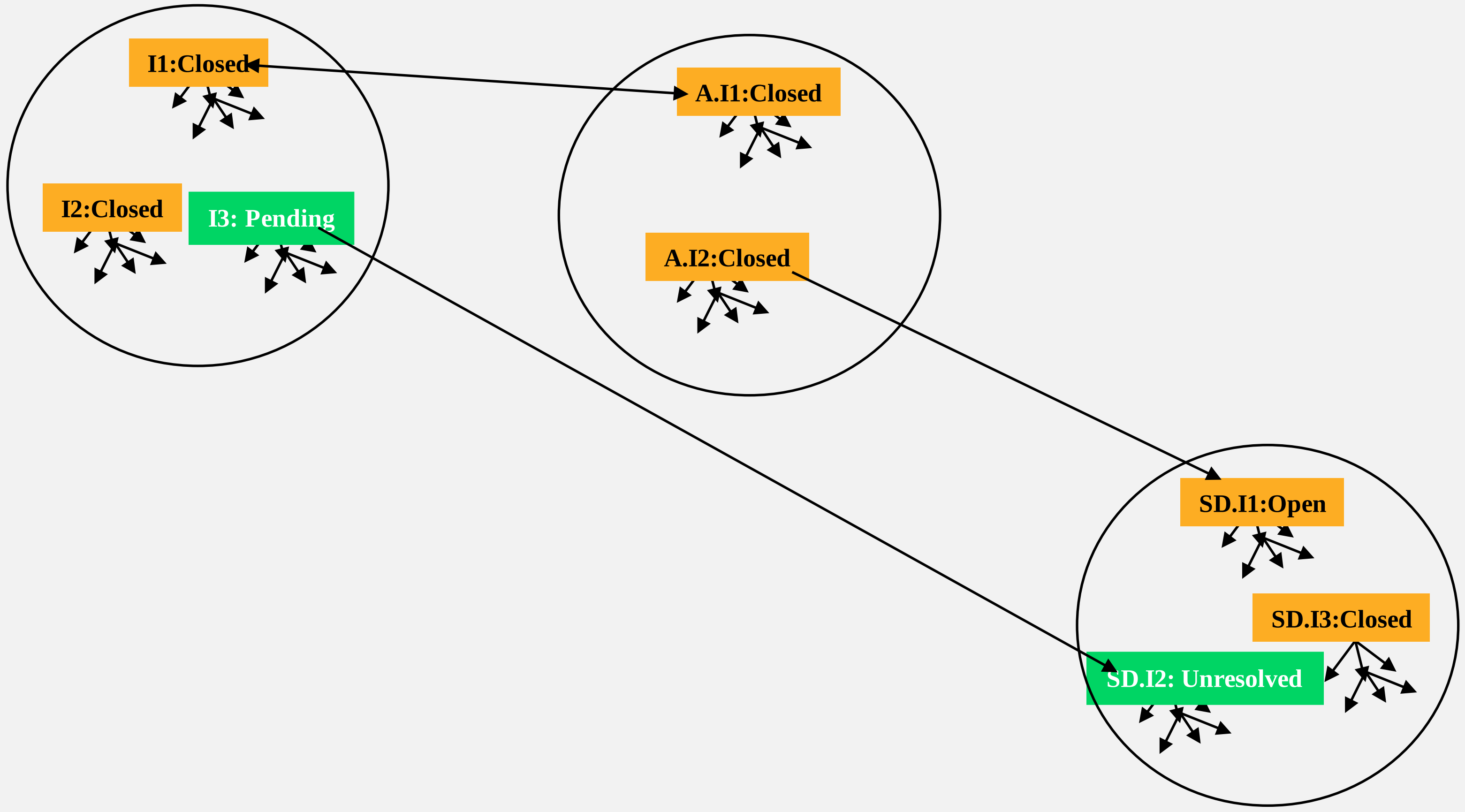


Analysis:10%

Design: 10%

Implemen-
tation: 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)

- Linear Model (Waterfall, V-Model)
- Open issues are closed before moving to next phase

Change occurs sometimes (MTBC ≈ PT)

- Iterative model (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 Model (Concurrent Development, Scrum)
- Phases are never finished, they all run in parallel.

Thank You