The McGraw-Hill Companies

Object-Oriented Software Engineering: An Agile Unified Methodology
by David Kung

Lec.02 Process and Methodology

Jackeycheung@HKMU

Key Takeaway Points

- A software process defines the phases of activities or what need to be performed to construct a software system.
- A software methodology details the steps or how to perform the activities of a software process. A methodology is an implementation of a process.
- Software development needs a software process and a methodology.

Challenges of System Development

Project Reality 1. Many systems require many years to develop.

Project Challenge 1. How do we schedule, and manage the work without knowing exactly what the customer wants, and what may happen in the future?

Project Reality 2. Many software projects require collaboration of multiple departments and teams.

Project Challenge 2. How do we divide the work among the departments and teams, and integrate the components produced by them?

Project Reality 3. Different departments or teams may use different processes, methods, and tools. They may reside at different locations.

Project Challenge 3. How do we ensure proper communication and coordination among the departments and teams?

Challenges of Systems Development

System Reality 1. Many systems need to satisfy numerous requirements and constraints.

System Challenge 1. How do we develop systems to ensure that the requirements and constraints are met?

System Reality 2. Requirements and constraints may change from time to time.

System Challenge 2. How do we design the processes and the products to cope with change?

System Reality 3. A system may consist of hardware, software and third party components using different programming languages and run on multiple platforms and machines located at different places.

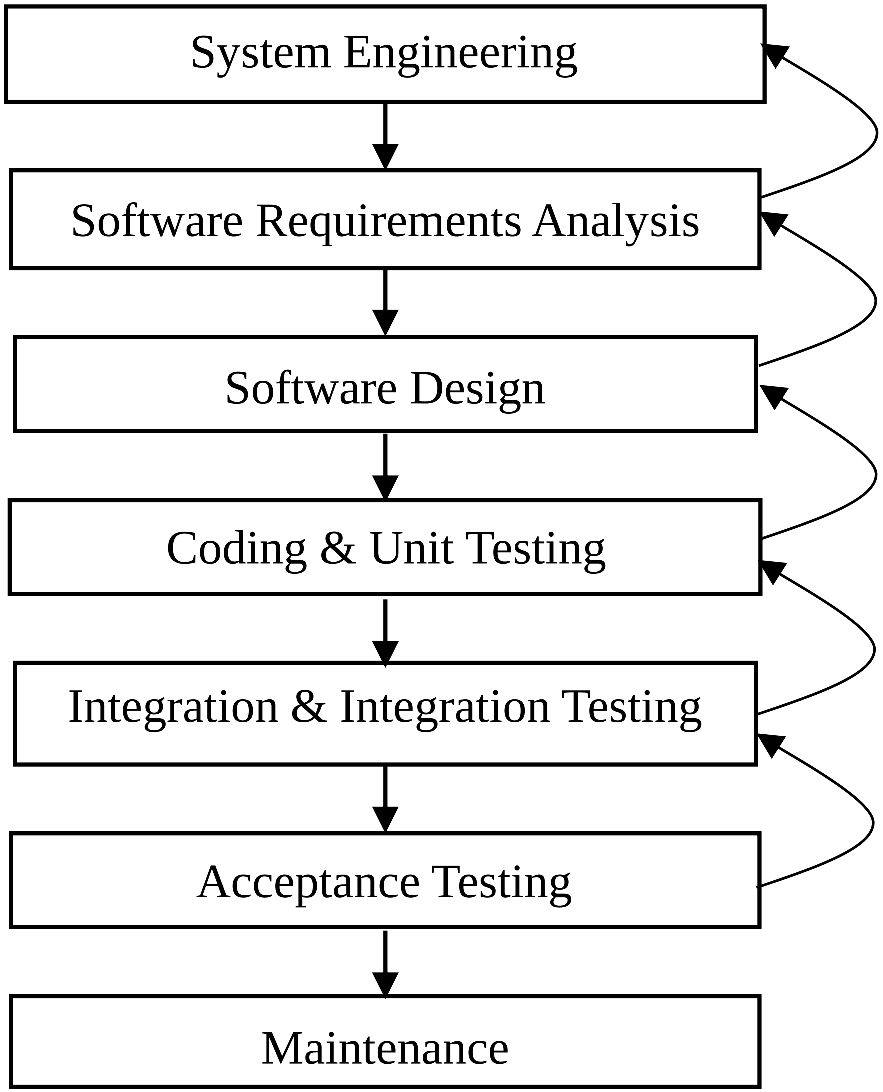
System Challenge 3. How do we design the system to hide these differences?

Software Process

System development challenges call for an engineering approach for software development. A software process is required.

Definition 2.1 A *software process* defines a series of activities performed to construct a software system. Each activity produces some artifacts, which are the input to other phases. Each phase has a set of entrance criteria and a set of exit criteria.

The Waterfall Process



Merits of the Waterfall Model

- The simple, straight sequence of phases of the waterfall simplifies project management.
- It supports function-oriented project organization:
  - Each project is carried out by a pipeline of functional teams.
  - Each functional team is specialized in one function such as requirements analysis, design, implementation, integration and testing, and so forth.

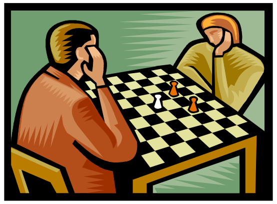
Problems of the Waterfall Model

- It is inflexible to requirements change.
- The long development duration means the system is outdated when it is delivered.
- Users cannot experiment with the system to provide early feedback.
- The customer has to wait until the entire system is implemented and deployed to reap the benefits.
- The customer may lose the entire investment if the project fails.

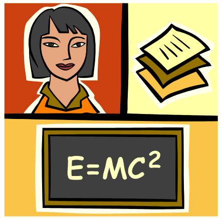
Properties of a Tame Problem

1) A tame problem can be completely specified.
2) For a tame problem, the specification and the solution can be separated.
3) For tame problems, there are stopping rules.
4) A solution to a tame problem can be evaluated in terms of correct or wrong.
5) Each step of the problem-solving process has a finite number of possible moves.
6) These is a definite chain of cause-effect reasoning.
7) The solution can be tested immediately; once tested, it remains correct forever.
8) The solution can be adapted for solving similar problems.
9) The solution process is a scientific process.
10) If the problem is not solved, simply try again.

Examples of Tame Problem



Chess playing

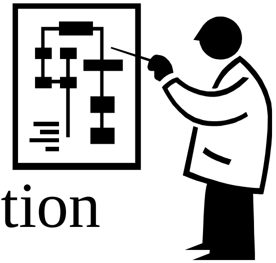


Math problems

Operations research

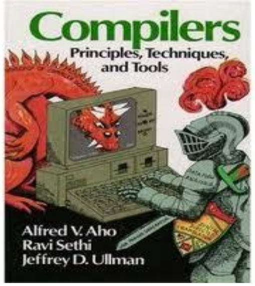
Many computer science problems

Query optimization



Operating systems

AI problems



Compiler construction

Why are these tame problems?

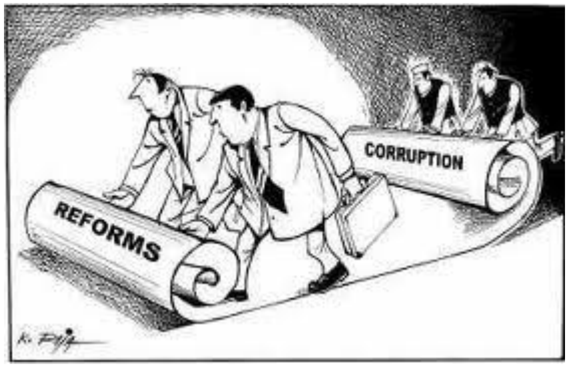
Software Development Is a Wicked Problem

1) A wicked problem does not have a definite formulation.
2) The specification and solution cannot be separated.
3) There is no stopping rule – you can always do it better.
4) The solutions can only be evaluated in terms of good or bad, and the judgment is usually subjective.
5) Each step of the problem-solving process has an infinite number of choices – everything goes as a matter of principle.
6) Cause-effect reasoning is premise-based, leading to varying actions, but hard to tell which one is the best.
7) The solution is subject to life-long testing.
8) Every wicked problem is unique.
9) The solution process is a political process.
10) The problem-solver has no right to be wrong because the consequence is disastrous.

Examples of Wicked Problem

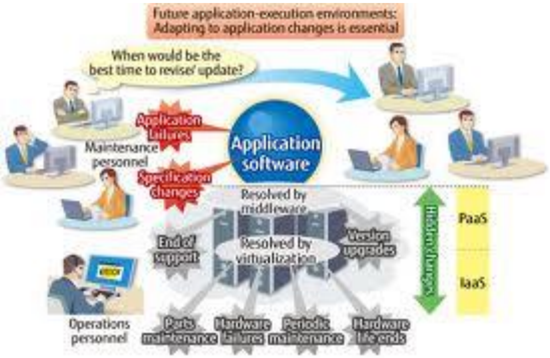
Urban planning

National policy making



Economic reforms

Why are these wicked problems?



Application software development

Software Process Models

- Prototyping Process Model
- Evolutionary Process Model
- Spiral Process Model
- Unified Process Model
- Personal Software Process Model
- Team Software Process Model
- Agile Process Models

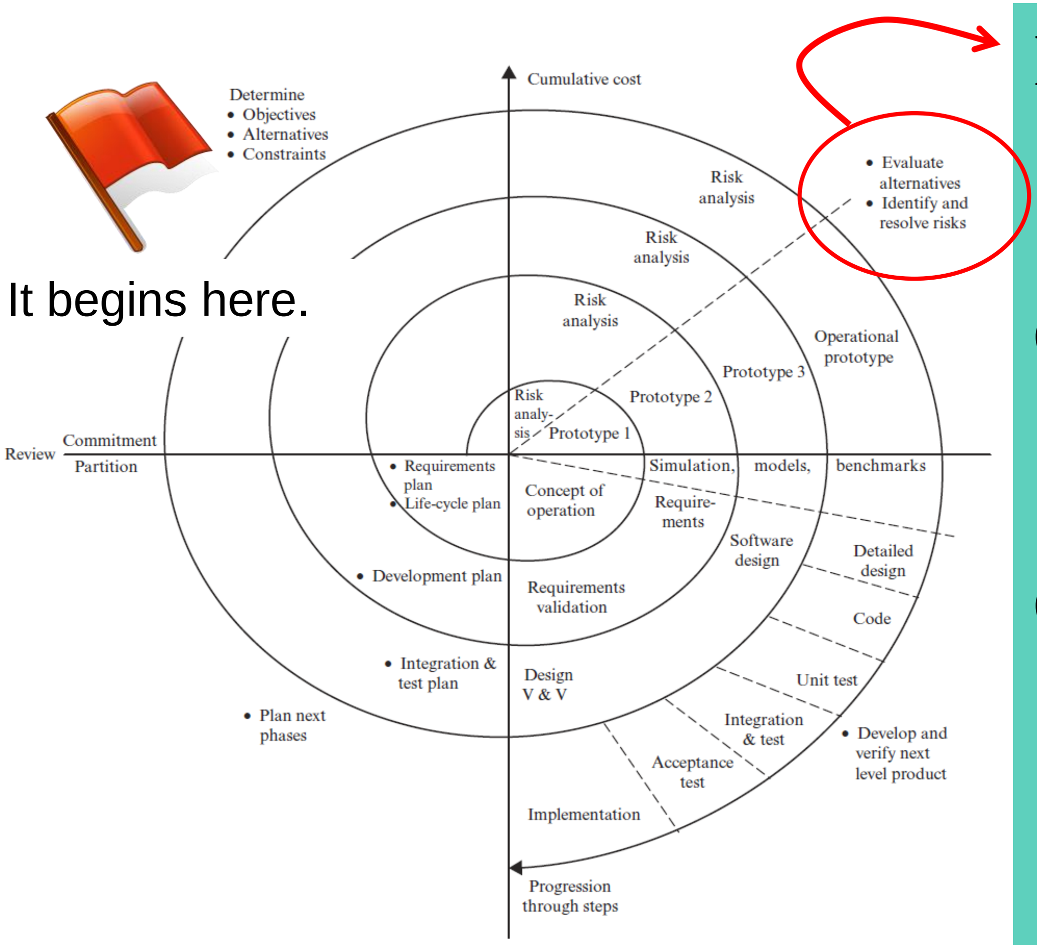
Prototyping Process Model

- Prototypes of the software system are constructed to:
  - acquire and validate requirements
  - assess the feasibility of the project and/or the feasibility of the requirements and constraints
- Simple prototypes as well as sophisticated prototypes are used, depending on the needs of the project.
- Prototypes are classified into throwaway prototypes and evolutionary prototypes.

Evolutionary Prototyping Model

- Throwaway prototypes waste time and effort.
- Evolutionary prototyping model lets the prototype evolve into the production system.
- It is most suited for the development of exploratory types of systems such as intelligent systems, research software, and systems that actively interact with and control the environment.
- It is not suitable for projects that require a predictable schedule of progress.

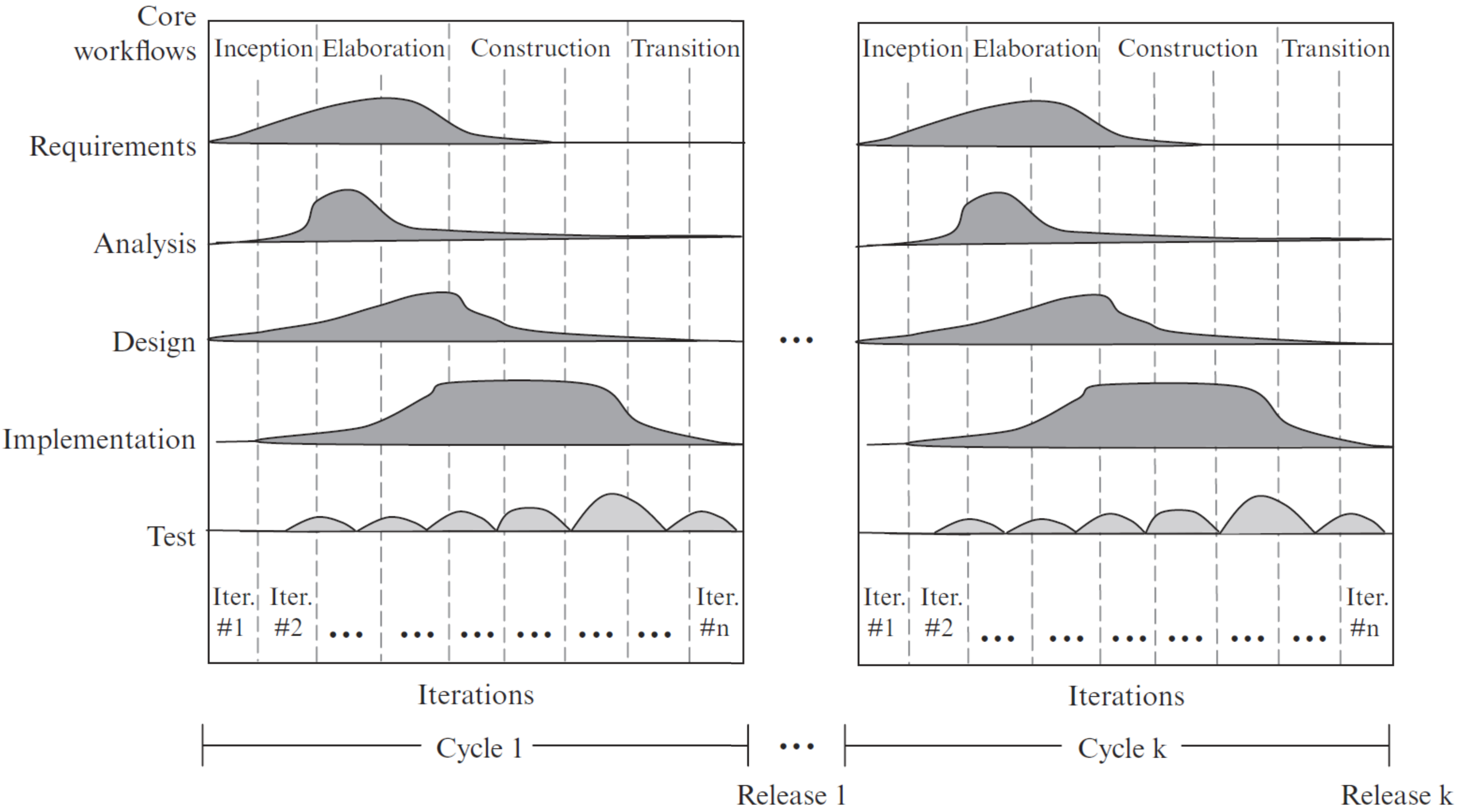
Spiral Process Model



It begins here.

```
If risks remains {
   plan next phase (SW)
   conduct prototyping }
else if  risks resolved {
    proceed as waterfall
    (SE) }
else if prototype works
    & robust {
    proceed as
    evolutionary model
    (NE corner) }
```

Rational Unified Process (RUP)



Rational Unified Process (RUP)

- **Inception** consists of the first 1-2 iterations. It produces a simplified use case model, a tentative architecture, and a project plan.
- **Elaboration** consists of the next N iterations. It produces the architectural design and implements the most critical use cases.
- **Construction**, during which remaining use cases are iteratively implemented and integrated into the system.
- **Transition,** during which the system is deployed, users are trained, and defects are corrected.

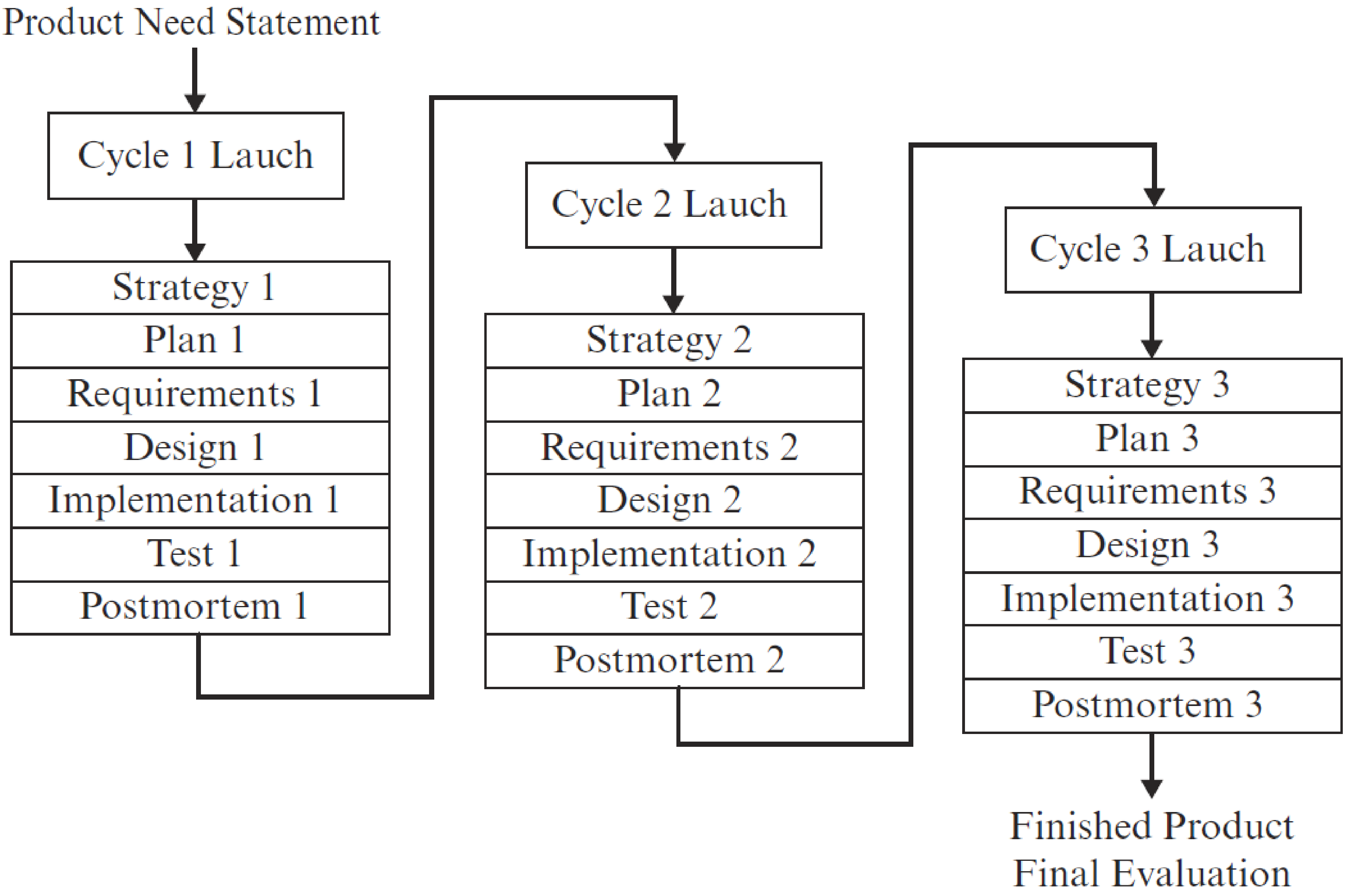
Personal Software Process Model

- PSP is a comprehensive framework for training software engineers.
- It consists of scripts, forms, standards and guidelines used in the training.
- It helps the software engineer identify areas for improvement.
- It prepares the software engineer to work in a team project.

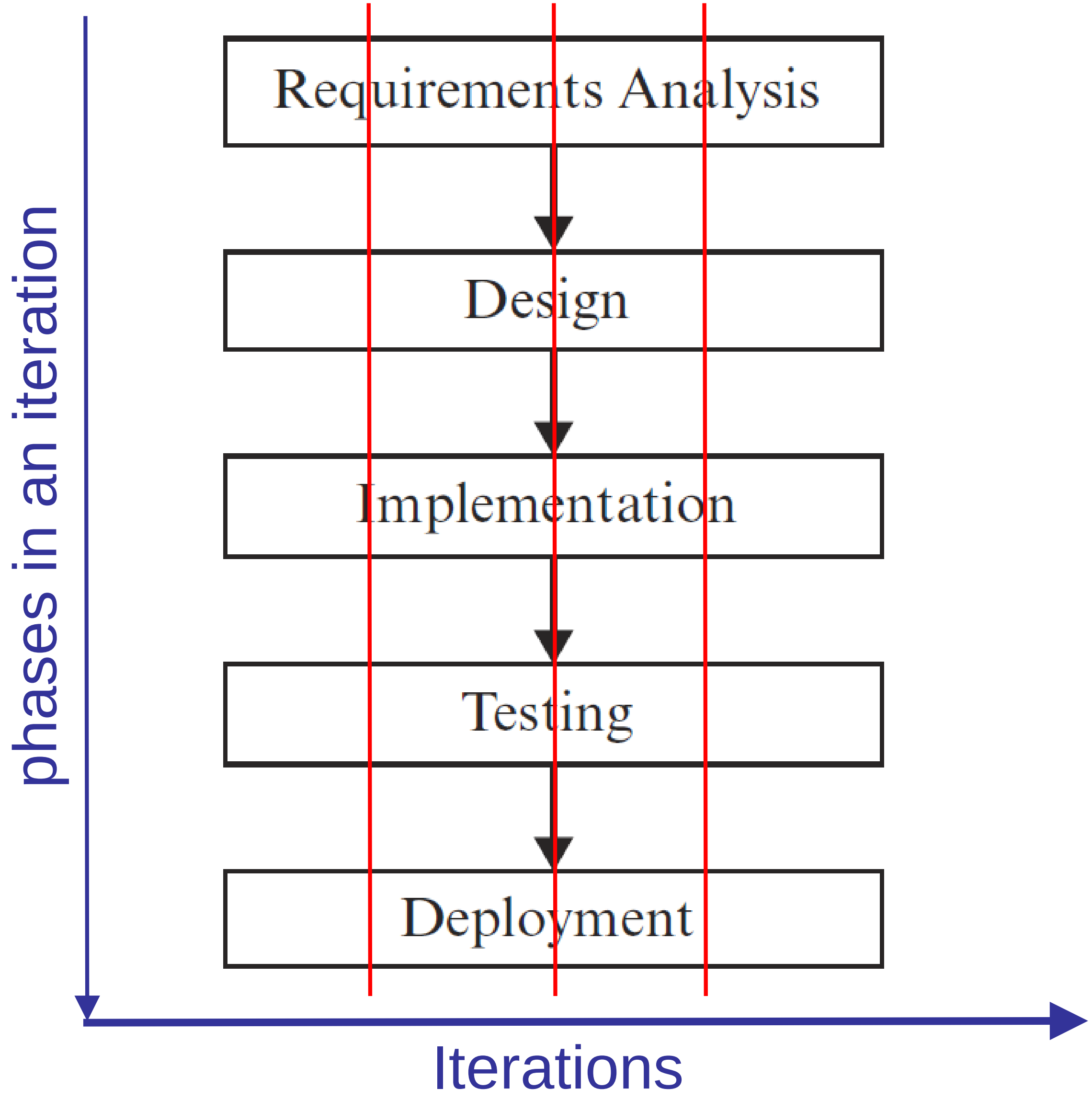
Personal Software Process Model

- PSP framework consists of a series of predefined processes:
  - PSP0 & PSP0.1. These introduce process discipline and measurement including baseline, time recording, defect recording, defect type, coding standards, and process improvement.
  - PSP1 & PSP1.1. These introduce size estimation, planning and scheduling.
  - PSP2 & PSP2.1. These introduce quality management and design including code review and design review.
  - PSP3.0. It guides component level development.

Team Software Process



Agile Process Models



Process and Methodology

- Methodology is often confused with process.
- A process is a set of inter-related activities to be carried out to construct something (usually a system).
- The activities are carried out in phases.
- Each phase produces some products which are the input of the next phase.
- The completion of each phase establishes a milestone.
- A methodology implements a process or a phase of a process.

Methodology

- A methodology is a cook-book for performing a task. It describes
  - steps to accomplish a series of subtasks
  - input and output of each step
  - representations of input and output
  - entrance and exit conditions for each step
  - procedures for carrying out each step
  - methods and techniques used by each step
  - relationships, or control flow and data flow between the steps

Process and Methodology

Process

- Defines a framework of phased activities
- Specifies phases of WHAT
- Does not dictate representations of artifacts
- It is paradigm-independent
- A phase can be realized by different methodologies.

Examples

Waterfall, spiral, prototyping, unified, and agile processes

Methodology

- Defines steps to carry out phases of a process
- Describes steps of HOW
- Defines representations of artifacts (e.g., UML)
- It is paradigm-dependent
- Steps describe procedures, techniques & guidelines

Examples

Structured analysis/structured

design (SA/SD), Object Modeling Technique (OMT), Scrum, DSDM, FDD, XP, and Crystal Orange

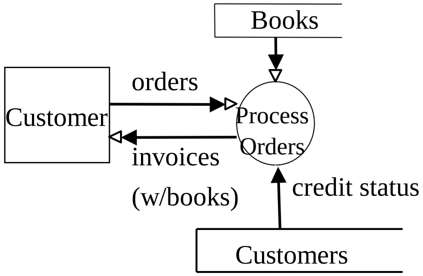
Software Paradigm

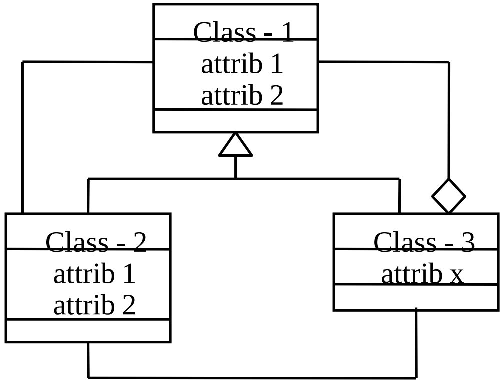
- A software paradigm is a style of software development that constitutes a way of viewing the reality.
- Examples:
  - procedural paradigm
  - OO paradigm, and
  - data-oriented paradigm

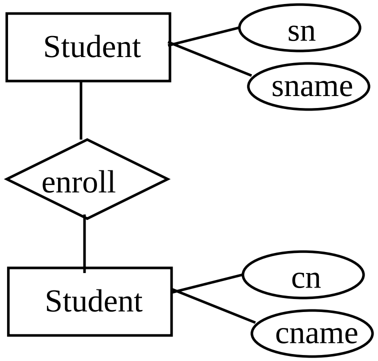
Three Paradigms in History

- Procedural paradigm views the world and system as:
  - a network of processes
  - a process is refined by lower level processes
  - basic building blocks and starting point are processes
- OO paradigm views the world and system as:
  - interrelated and interacting objects
  - basic building blocks are objects
- Data-oriented paradigm views the world and system as:
  - interrelated data entities, processed by transactions
  - basic building blocks are data entities and relationships

Paradigm and Methodology





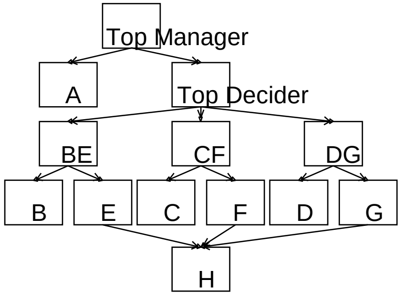


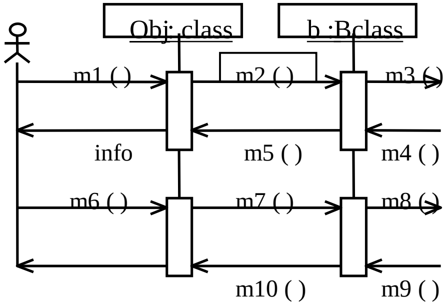
Domain model

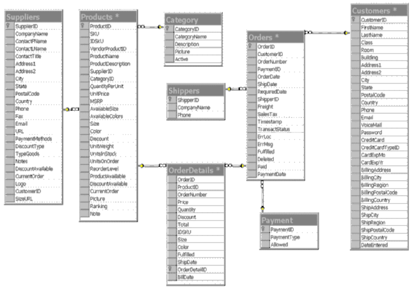
Structured Analysis

Object-Oriented Analysis

Data-Oriented Analysis







Sequence diagram

Structured Design

Object-Oriented Design

Data-Oriented Design

Structured Programming

Object-Oriented Programming

Programming in 4GL (e.g., SQL)

Procedural Paradigm

OO Paradigm

Data-Oriented Paradigm

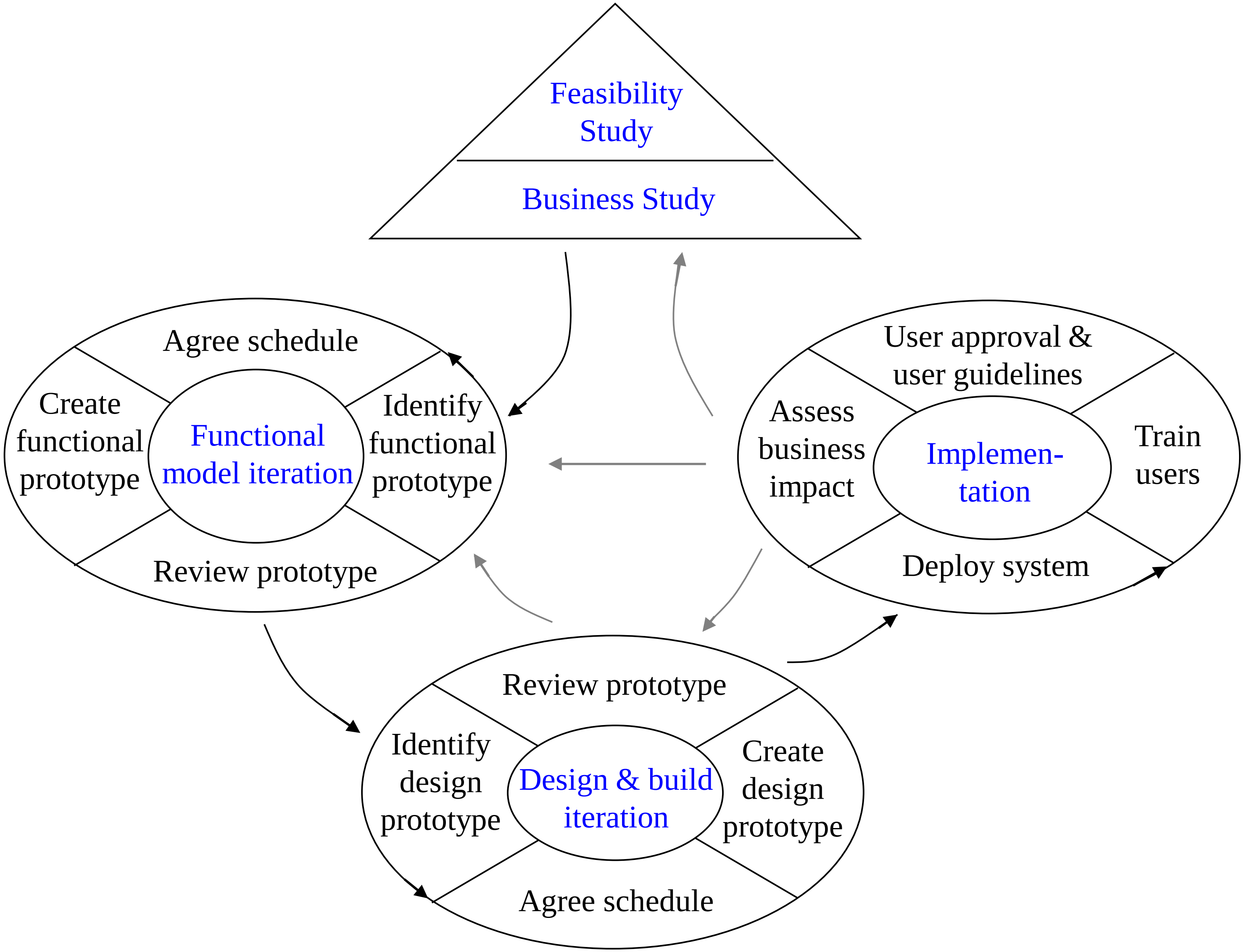
Some Well-Known Agile Methods

- Dynamic Systems Development Method (DSDM)
- Feature Driven Development (FDD)
- Scrum
- Extreme Programming (XP)
- Crystal Clear
- Lean Development

DSDM Unique Key Features

- It is a framework that works with Rational Unified Process and XP.
- It is based on the 80-20 principle.
- It is suitable for agile as well as plan-driven projects.

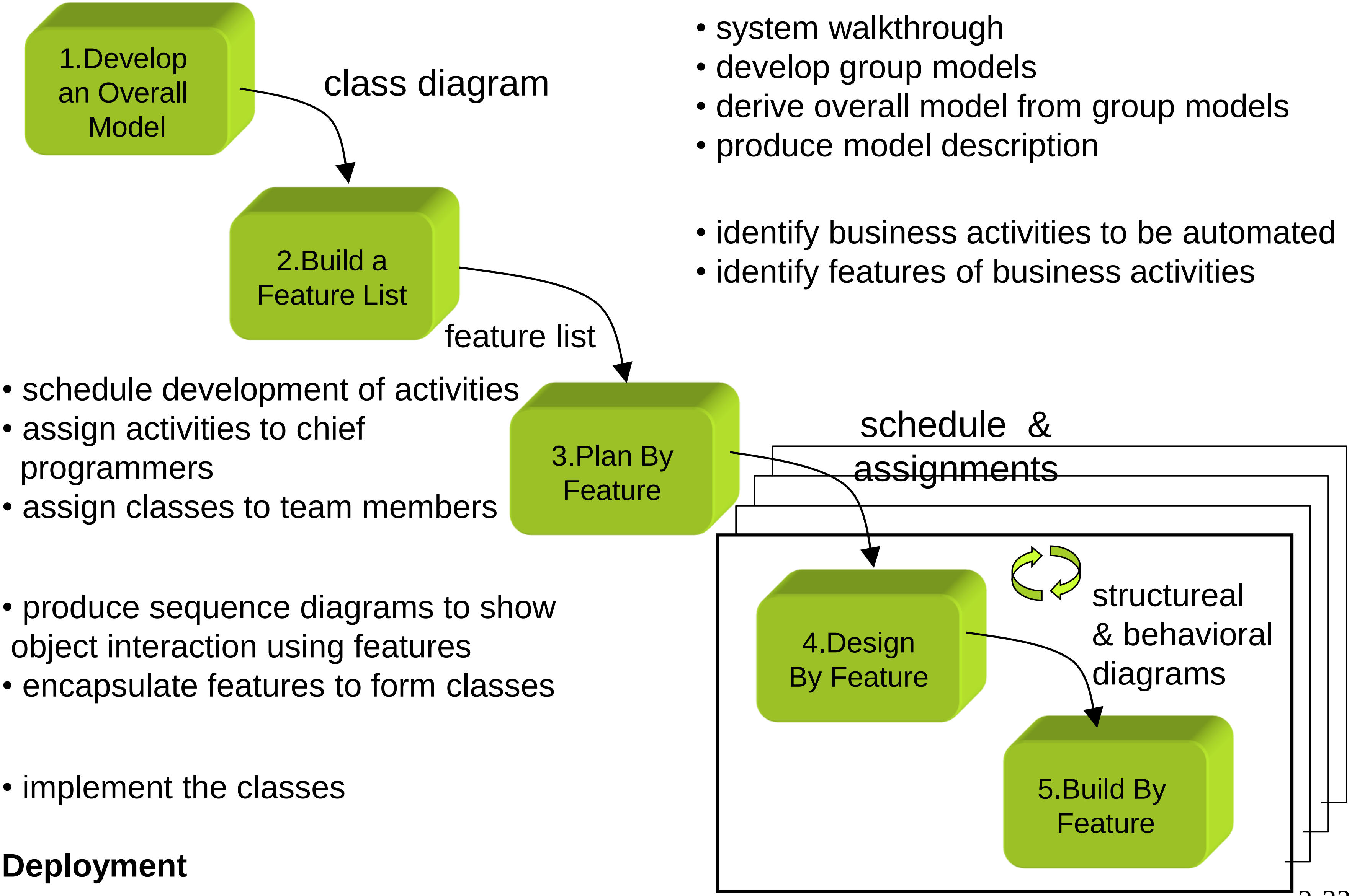
DSDM Lifecycle Activities



Feature Driven Development (FDD)

- Unique key features:
  - feature driven and model driven
  - configuration management, review and inspection, and regular builds
  - suitable for agile or plan-driven projects

FDD Lifecycle Activities



1.Develop an Overall Model

class diagram

- system walkthrough
- develop group models
- derive overall model from group models
- produce model description

2.Build a Feature List

- identify business activities to be automated
- identify features of business activities

feature list

- schedule development of activities
- assign activities to chief programmers
- assign classes to team members

3.Plan By Feature

schedule & assignments

4.Design By Feature

structureal & behavioral diagrams

- produce sequence diagrams to show object interaction using features
- encapsulate features to form classes

5.Build By Feature

- implement the classes

Deployment

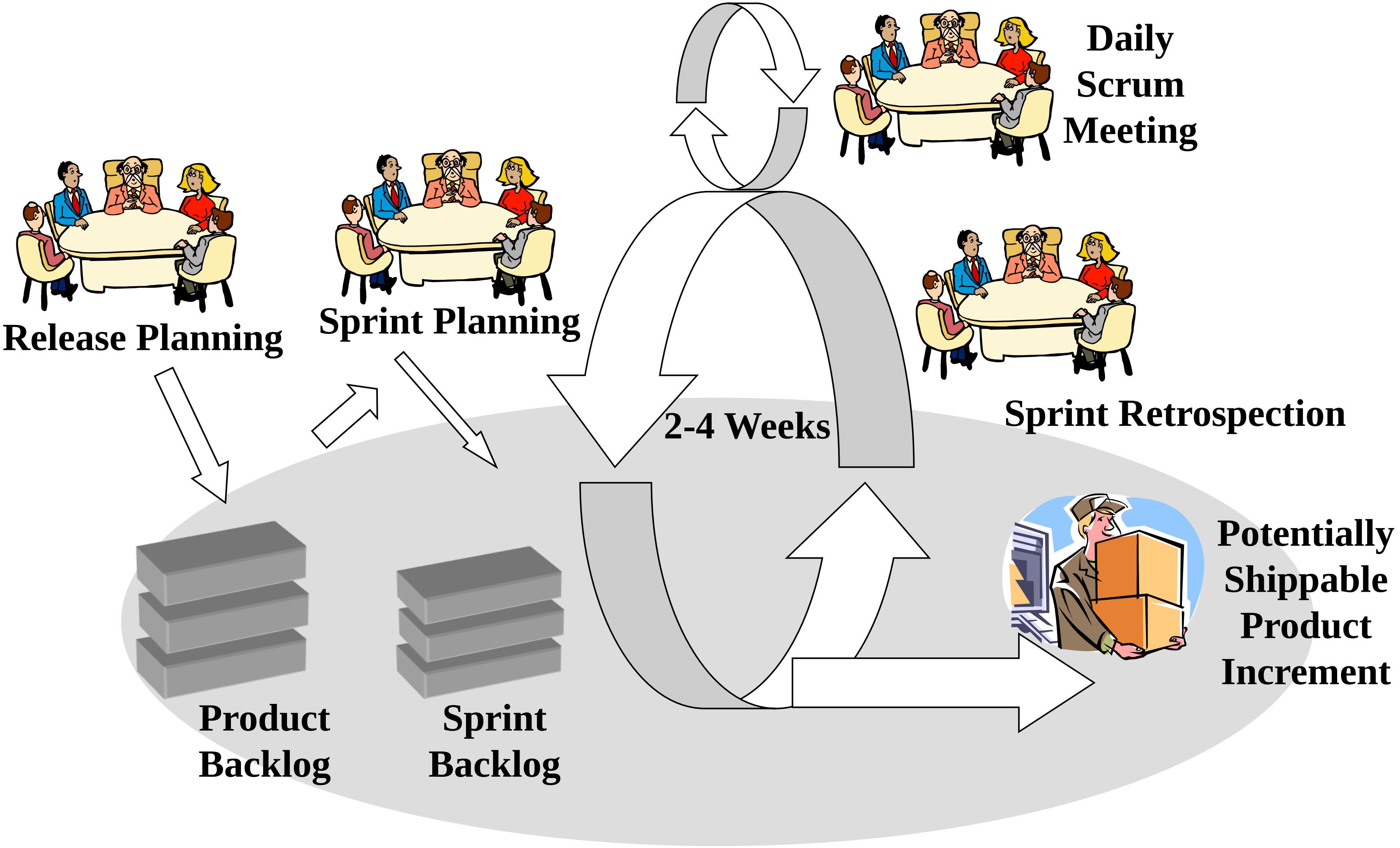
Scrum

- Unique key features:
  - include Scrum Master, Product Owner, and Team roles
  - 15 minute daily status meeting to improve communication
  - team retrospect to improve process

Scrum Lifecycle Activities

- Release planning meeting
  - identify and prioritize requirements, called product backlog
  - identify top priority requirements to be delivered within an increment, called a sprint
  - identify sprint development activities
- Sprint iteration
  - sprint planning meeting to determine what and how to build next
  - daily Scrum meeting to exchange status
- Sprint review meeting
  - increment demo
  - team retrospection
- Deployment

Scrum Process



Extreme Programming (XP)

- Unique Key Features:
  - Anyone can change any code anywhere at any time
  - Integration and build many times a day whenever a task is completed
  - Work ≤ 40 hours a week

XP Lifecycle Activities

Exploration

1. Collect information about the application
2. Conduct feasibility study

Planning

1. Determine the stories for the next release
2. Plan for the next release

Iteration to 1st Release

1. Define/modify architecture
2. Select and implement stories for each iteration
3. Perform functional tests by customer

Productionizing

1. Evaluate and improve system performance
2. Certify and test system for production use

Maintenance

1. Improve the current release
2. Repeat process with each new release

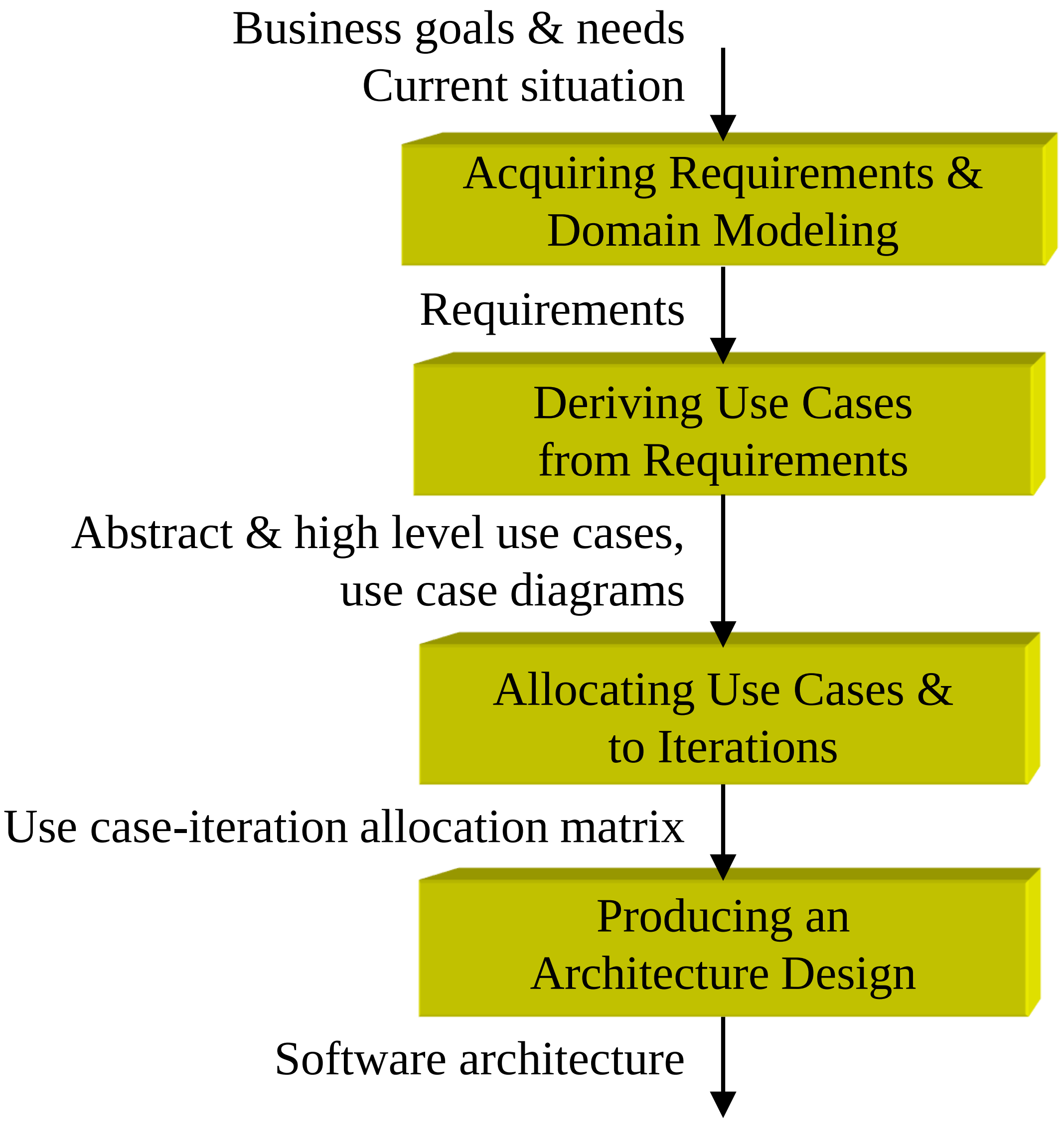
Death

1. Produce system documentation if project is done, or
2. Replace system if maintenance is too costly

The Methodology Presented in This Book

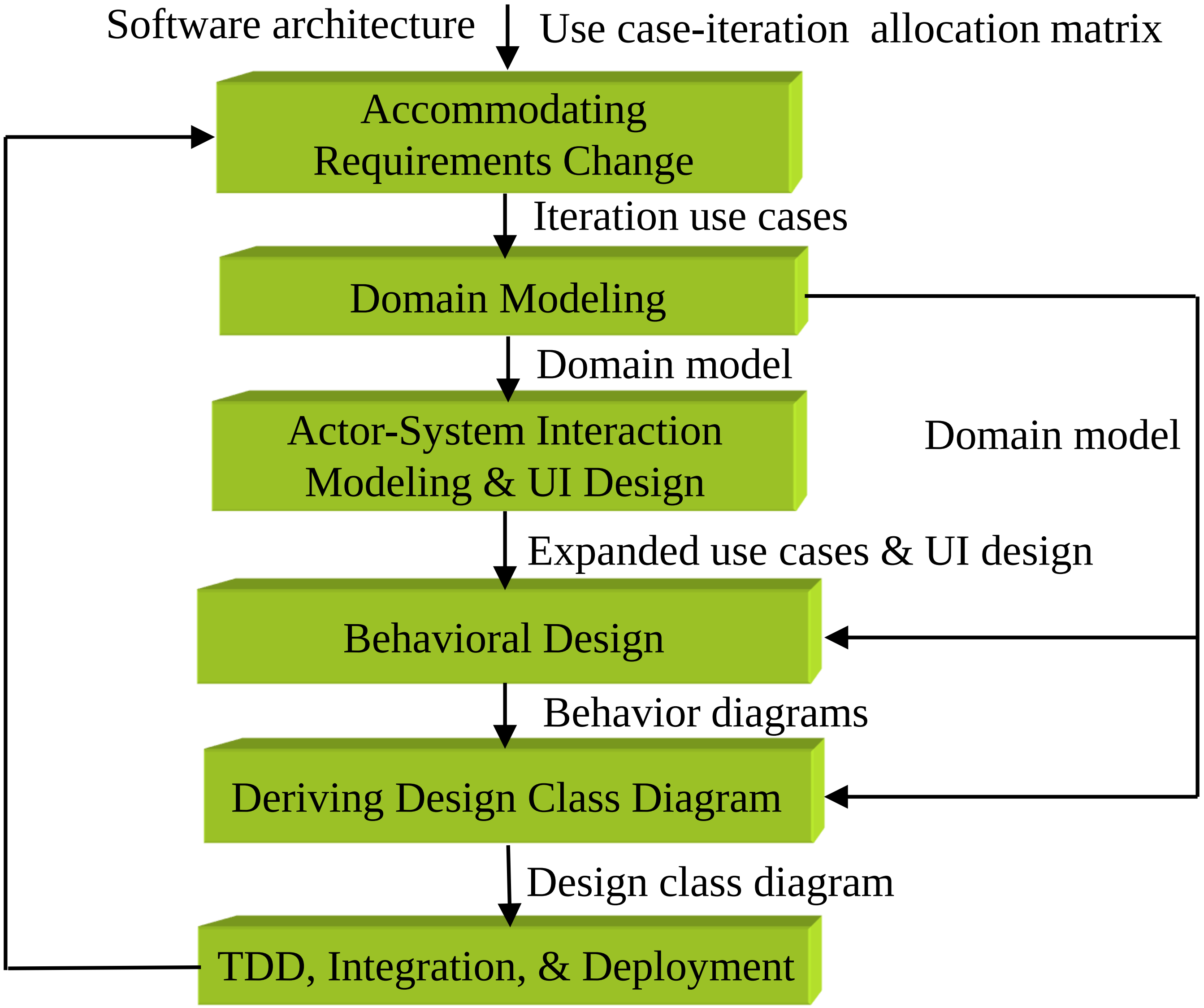
- It is designed for beginners as well as seasoned developers.
- It is aimed at educating software architects and systems analysts.
- It can be applied to agile as well as plan-driven projects. It has been applied to sponsored as well as industrial projects.
- Team members should work together from project start to completion.
- Many students continue practicing the methodology after graduation.

Methodology Overview – Planning Phase



Business goals & needs
Current situation
Acquiring Requirements & Domain Modeling
Requirements
Deriving Use Cases from Requirements
Abstract & high level use cases, use case diagrams
Allocating Use Cases & to Iterations
Use case-iteration allocation matrix
Producing an Architecture Design
Software architecture

Methodology Overview – Iterative Phase



Software architecture

Use case-iteration allocation matrix

Accommodating Requirements Change

Iteration use cases

Domain Modeling

Domain model

Actor-System Interaction Modeling & UI Design

Domain model

Expanded use cases & UI design

Behavioral Design

Behavior diagrams

Deriving Design Class Diagram

Design class diagram

TDD, Integration, & Deployment