Software Engineering for Engineers

Lecture 1: Introduction

Yesterdays News

- 40 Millon Customers without access to their T-Mobile phones
- Full blackout of the T-Mobile network from 15:45 to 22:00
  - The black out took so long, because most of the technicians and service personal that maintain the T-Mobile network could not be reached on their mobile phones
- Reason for the black out:
  - Central computer responsible for mapping phone numbers to SIM cards had crashed.

Bootstrapping

- Bootstrapping a network
  - Bringing up a network using the network itself.
- Bootstrapping a compiler:
  - Writing a compiler for a computer language using the computer language itself to code the compiler.
- Booting a computer:
  - process of a simple computer system activating a more complicated computer system.
- Bootstrap funding:
  - Starting a business without external capital. Startups that fund their business and development of their company through their own internal cash flow.

Baron von Münchhausen

- "I was still a couple of miles above the clouds when it broke, and with such violence I fell to the ground that I found myself stunned, and in a hole nine fathoms under the grass, when I recovered, hardly knowing how to get out again.
- Looking down, I observed that I had on a pair of boots with exceptionally sturdy straps. Grasping them firmly, I pulled with all my might.
- Soon I had hoist myself to the top and stepped out on terra firma without further ado.“

From R. E. Raspe, *Singular Travels, Campaigns and Adventures of Baron Munchausen*, 1786.

Outline

- Organizational issues
  - Time and location
  - Grading criteria
  - Exercises
- The development challenge
- Software Engineering activities
- Lecture Schedule

- What is UML and why do we use it?

- UML Class Diagram
  - Associations
  - Inheritance
  - UML to Java

Module

- Intended Audience:
  - Engineers (Physics, Mechanical Engineering, ...)
  - Computer Science
  - CSE
- Prerequisites:
  - Modul IN1503: Introduction to Programming
  - Modul IN2157: Grundlegende Algorithmen (CSE)
  - Experience in Java
- Beneficial:
  - You have failed a software project
  - You have had practical experience with a large software system
  - You have already participated in a large software project

Time and location

- Lecture: Wednesday 10:15-12:00 MI 00.08.038
- Exercise: Wednesday 16:00-18:00 MI 00.08.038

- Change due to schedule collision

- Final Exam: July 30th (preliminary)

Grading Criteria

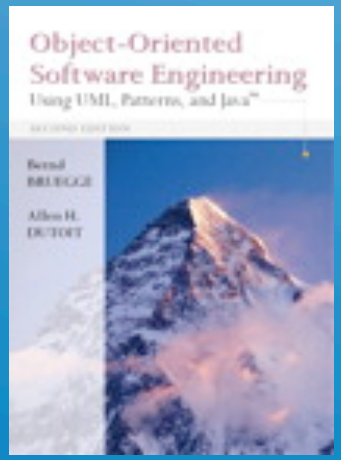
- Successful participation in the exercise is an admission requirement for the exam
- Your final grade is your final exam grade possibly improved by a bonus for your exercise participation
- If your participation in the exercises is excellent, we will give you a bonus of 1/3 on the final grade (For example, if your final grade is 2.3 you can improve it to 2.0)
  - The bonus cannot be granted if your grade is 4.3 or worse.
  - Information on the participation is available on the exercise portal.
- You pass the course if your final grade is 4.0 or better

Exercises

- Teams of 3 to 5 people (allocation today)
- Extending an existing system
- Practically teach Software Engineering activities
- Management and modeling
- Development Environment:
  - Java
  - Eclipse
  - Unicase
- Teaching assistants: Jonas Helming and Maximilian Kögel
  - helming@in.tum.de
  - koegel@in.tum.de

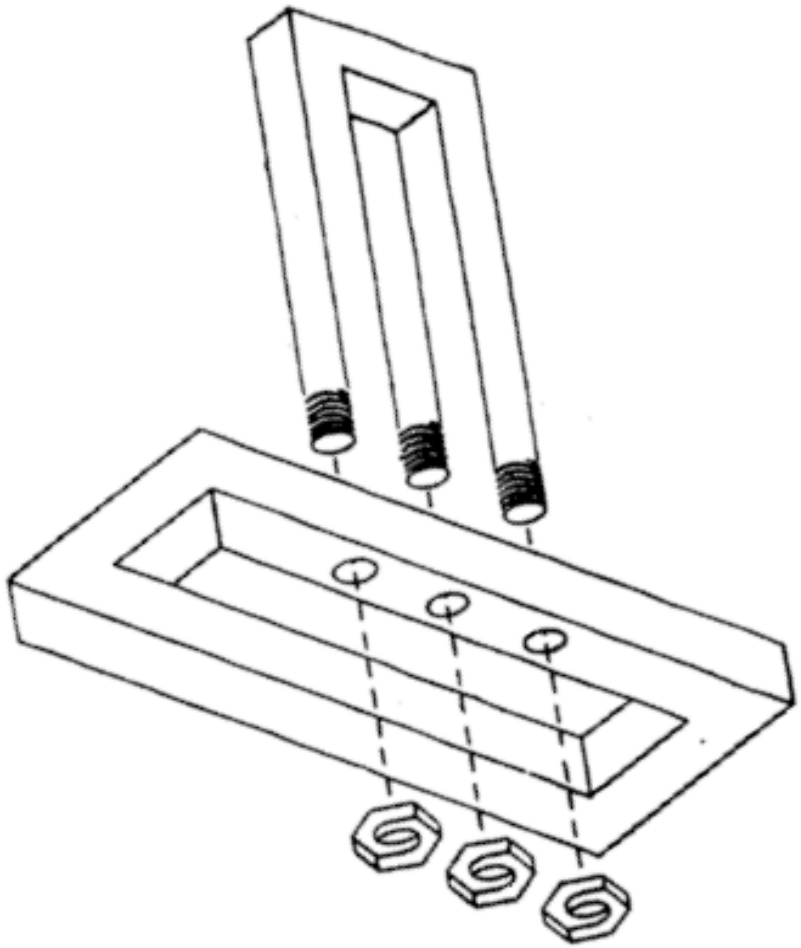
Readings

Bernd Bruegge, Allen H. Dutoit
Object-Oriented Software Engineering: Using UML, Patterns and Java, 2nd Edition
Publisher: Prentice Hall, Upper Saddle River, NJ, 2003;
ISBN: 0-13-047110-0

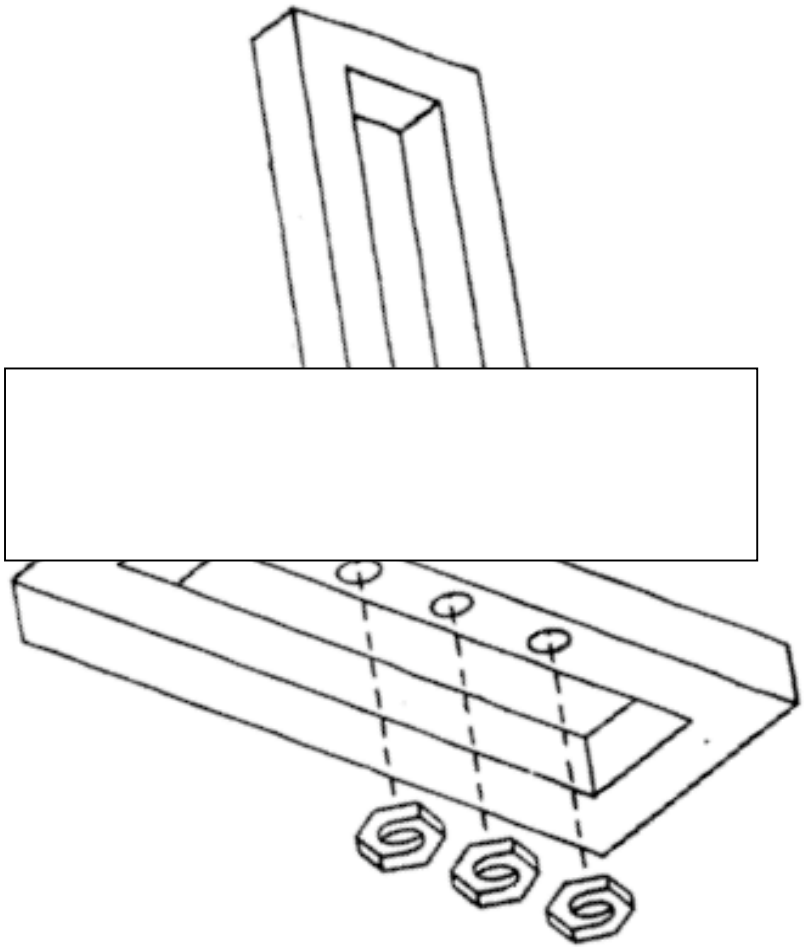


- German Version: “Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java, Pearson Education, Oktober 2004.
- Spanish Version: Ingeneria De Software Orientado a Objetos
- Chinese Version also available

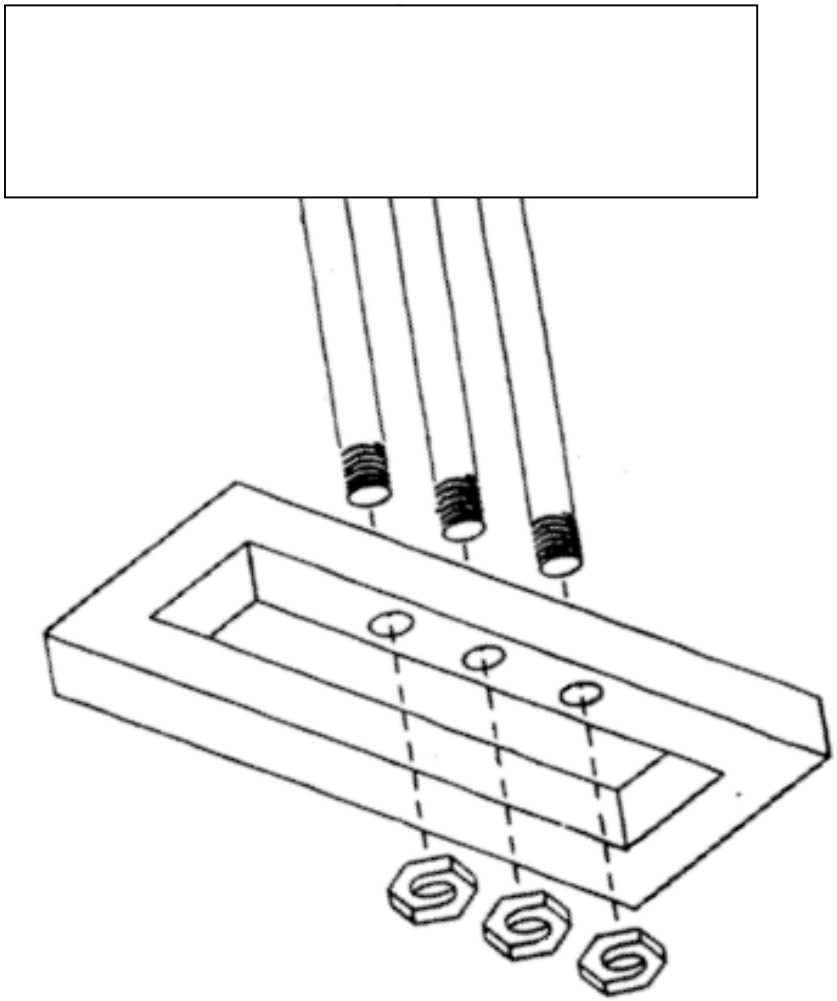
Can you develop this system?



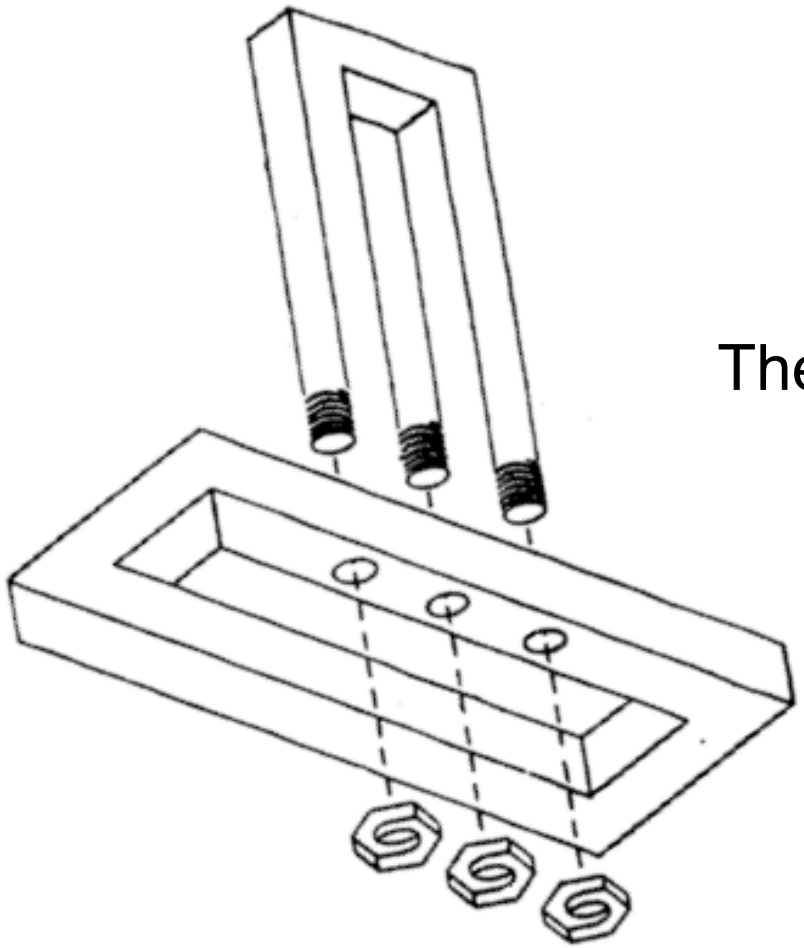
Can you develop this system?



Can you develop this system?

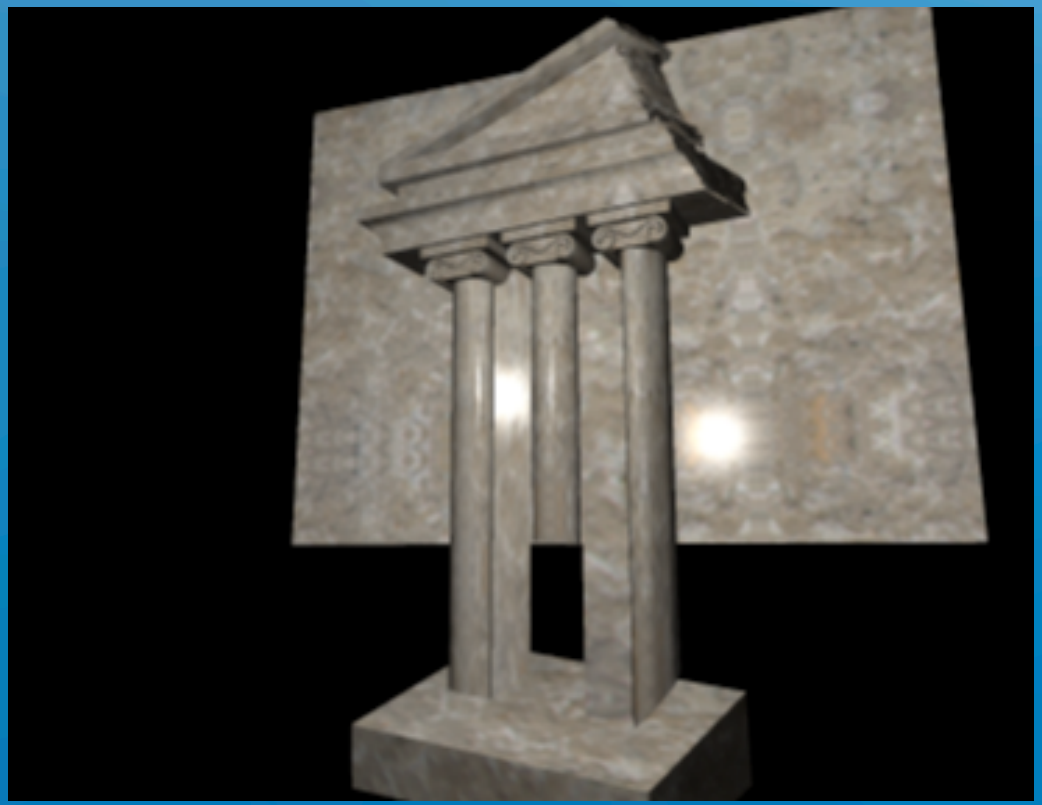


Can you develop this system?



The impossible Fork

Physical Model of the impossible Fork (Shigeo Fukuda)



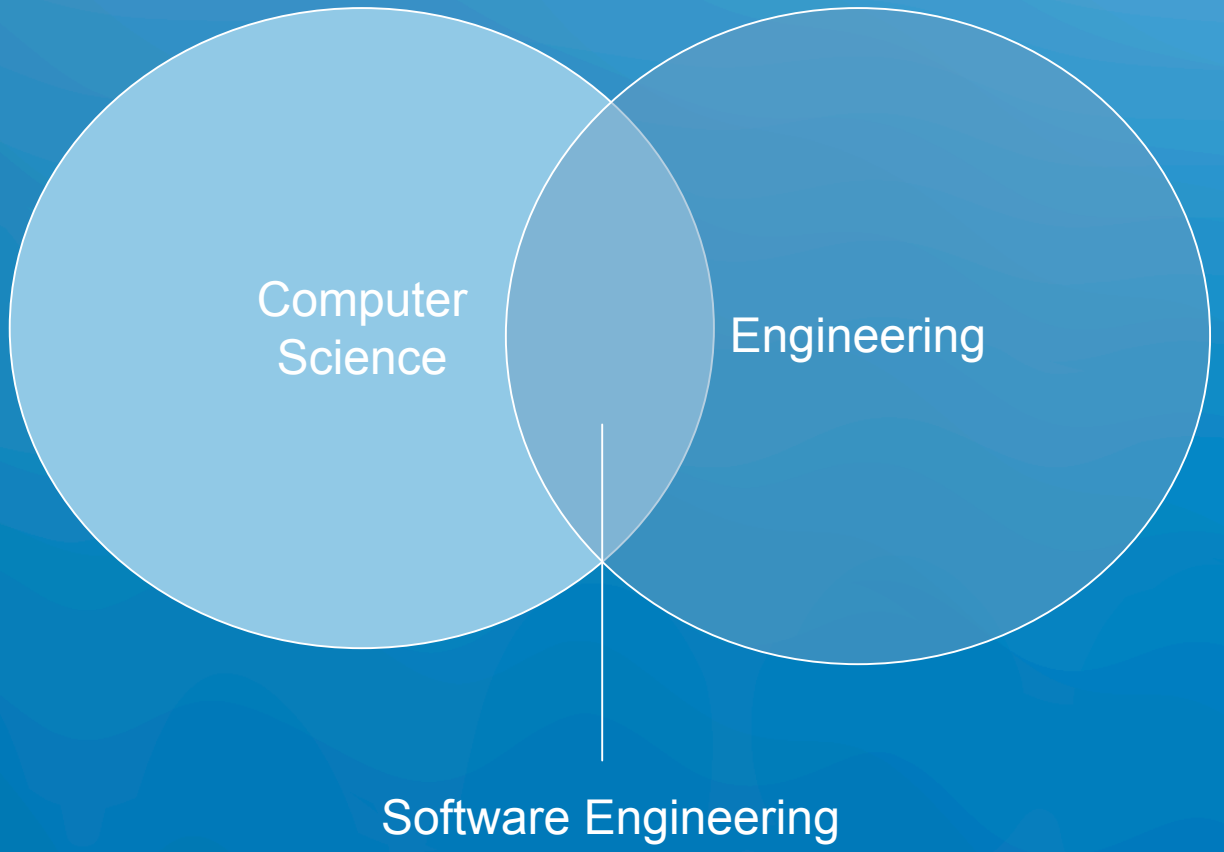
http://illusionworks.com/mod/movies/fukuda/DisappearingColumn.mov

Why is software developement difficult?

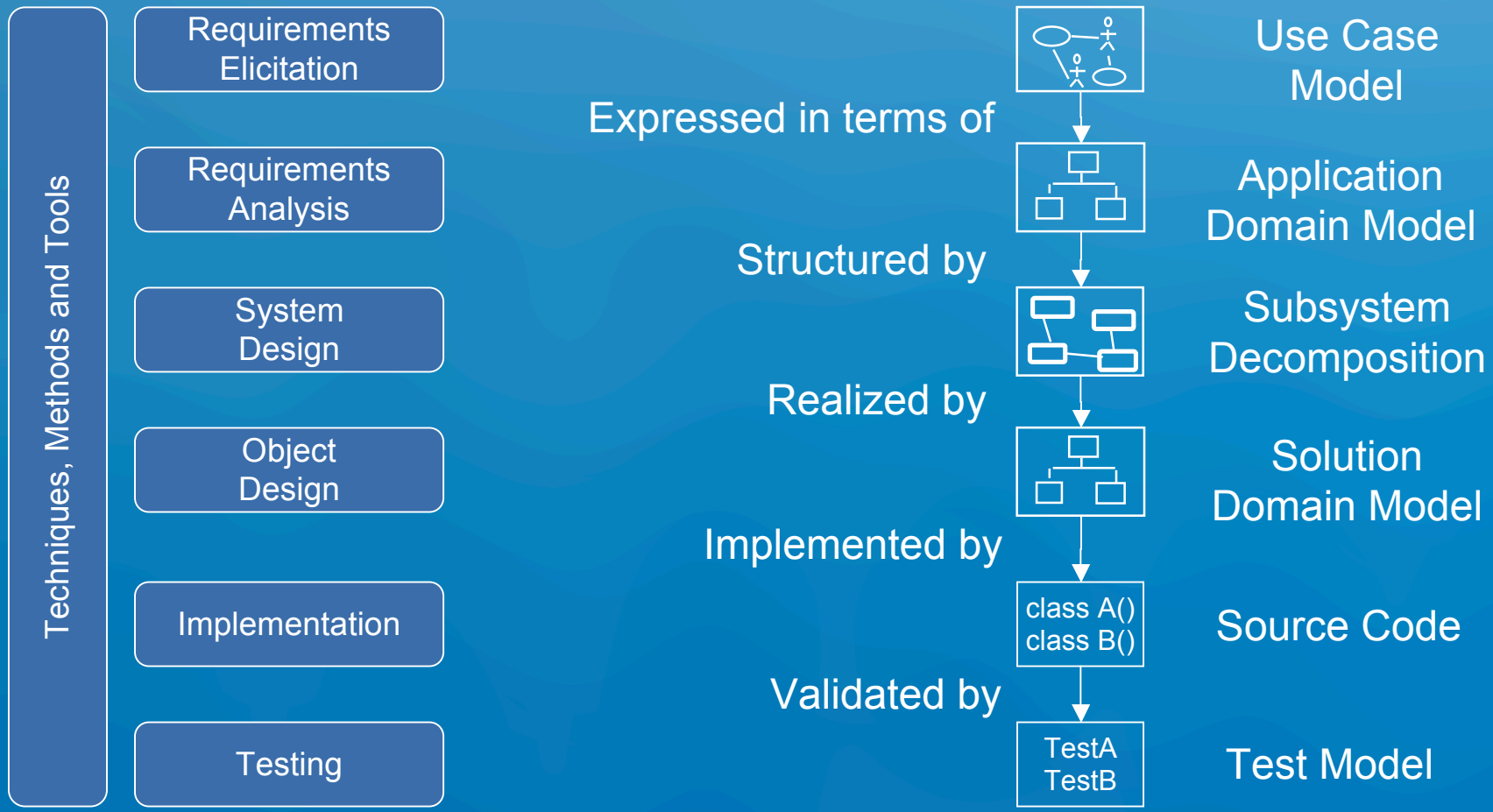
- The problem domain (also called application domain) is difficult
- The solution domain is difficult
- The development process is difficult to manage
- Software offers extreme flexibility
- Software is a discrete system
  - Continuous systems have no hidden surprises
  - Discrete systems can have hidden surprises! (Parnas)

David Lorge Parnas is an early pioneer in software engineering who developed the concepts of modularity and information hiding in systems which are the foundation of object oriented methodologies.

Computer Science vs. Engineering



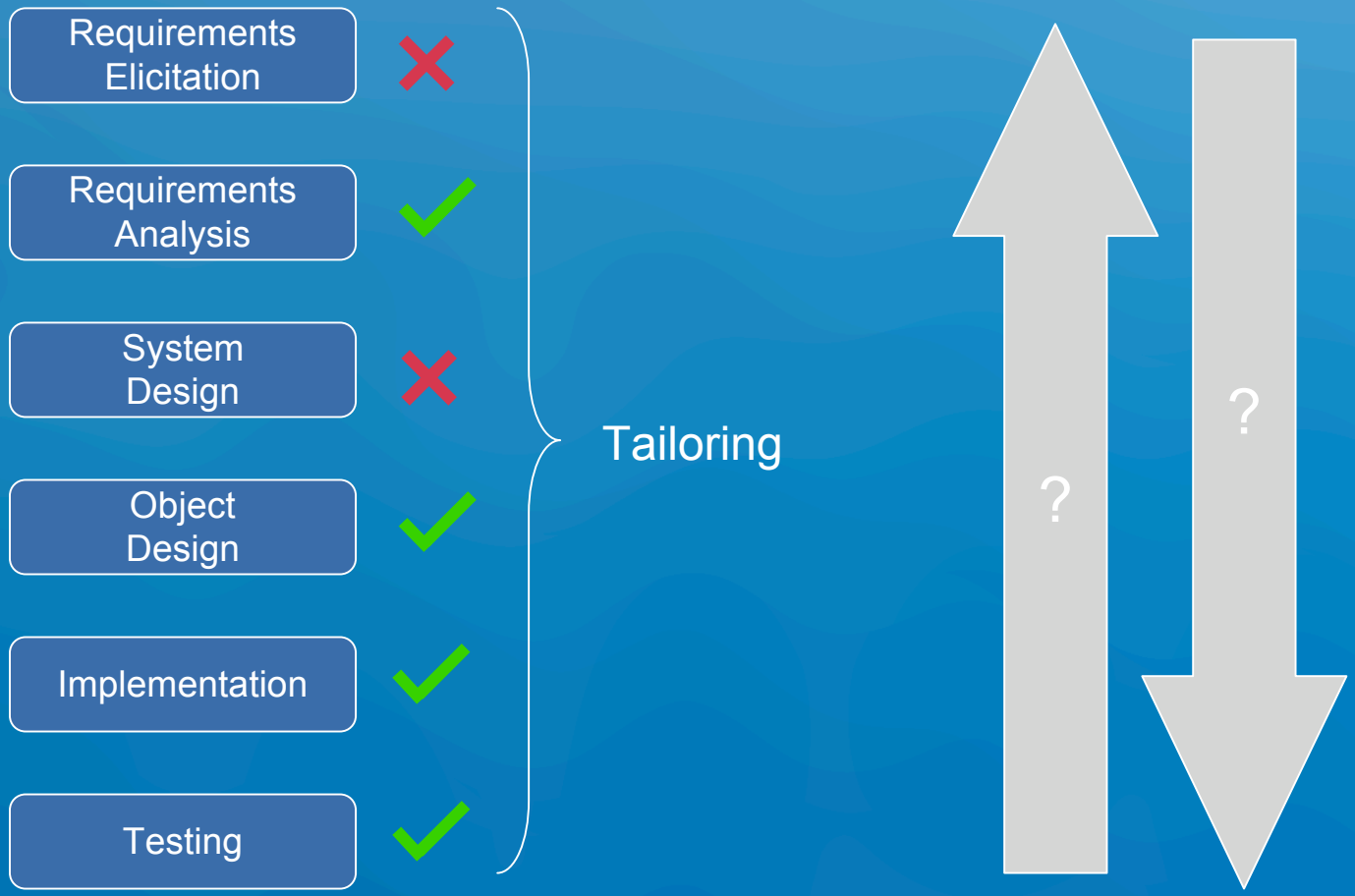
Software Engineering Activities



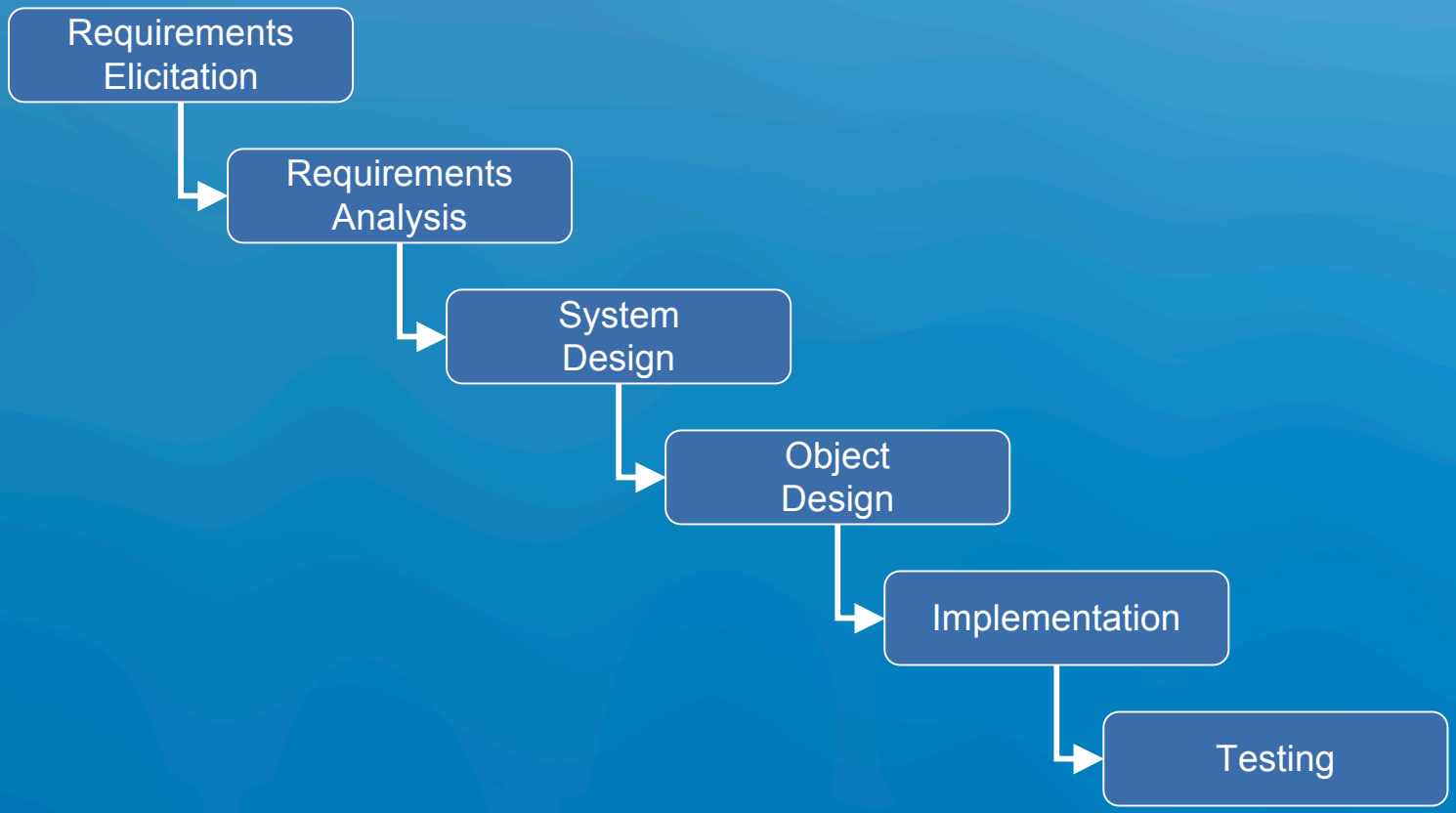
Techniques, Methods and Tools

- **Techniques or Methods :**
  - Formal procedures for producing results using some well-defined notation
  - UML = Unified Modeling Language
  - Number of repeatable steps involved in solving a specific problem
- **Tools:**
  - Instruments or automated systems to accomplish a technique
  - CASE = Computer Aided Software Engineering
- Methodology:
  - Consists of techniques, methods and Tools

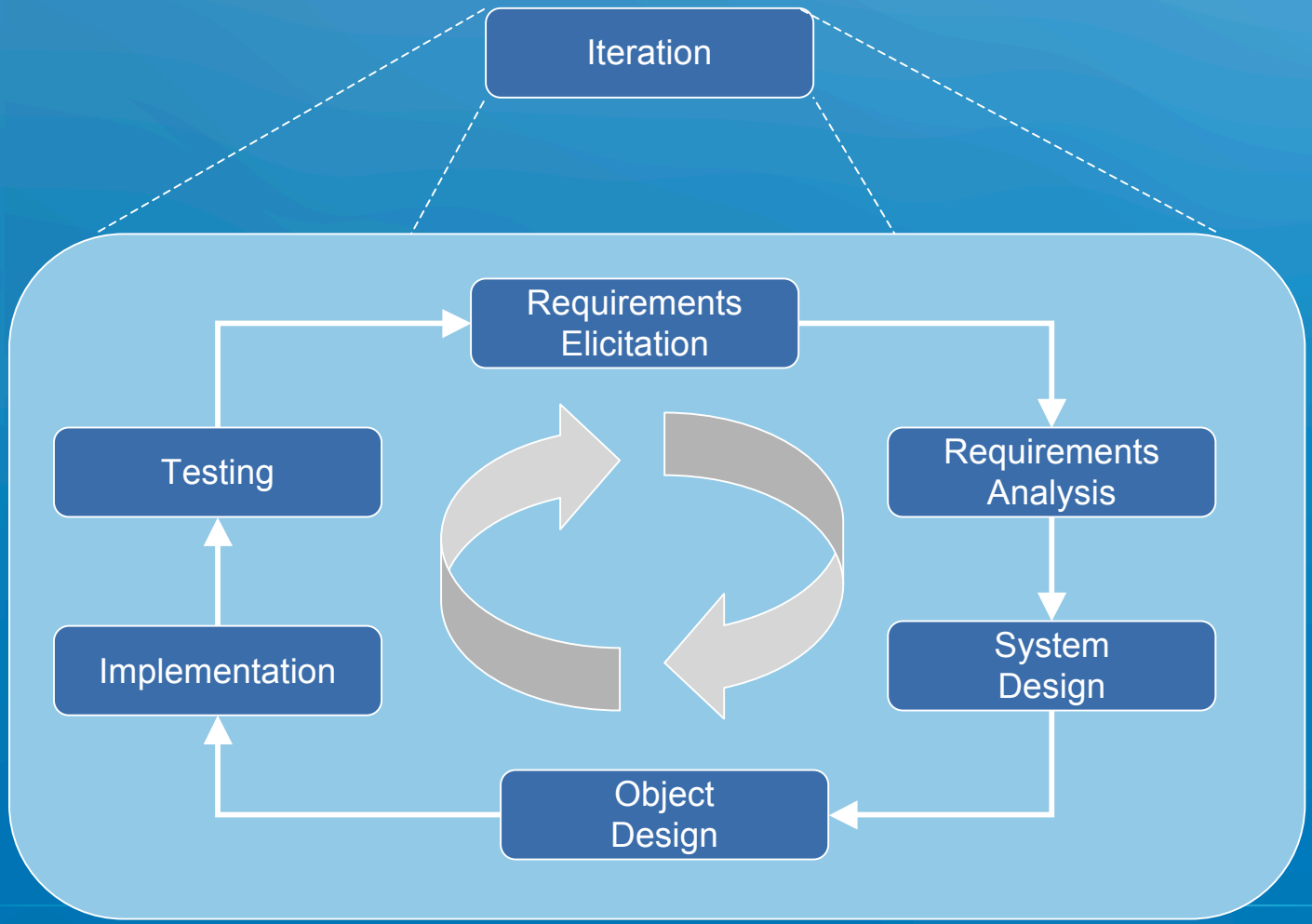
Software Life Cycle: which activities, which dependencies?



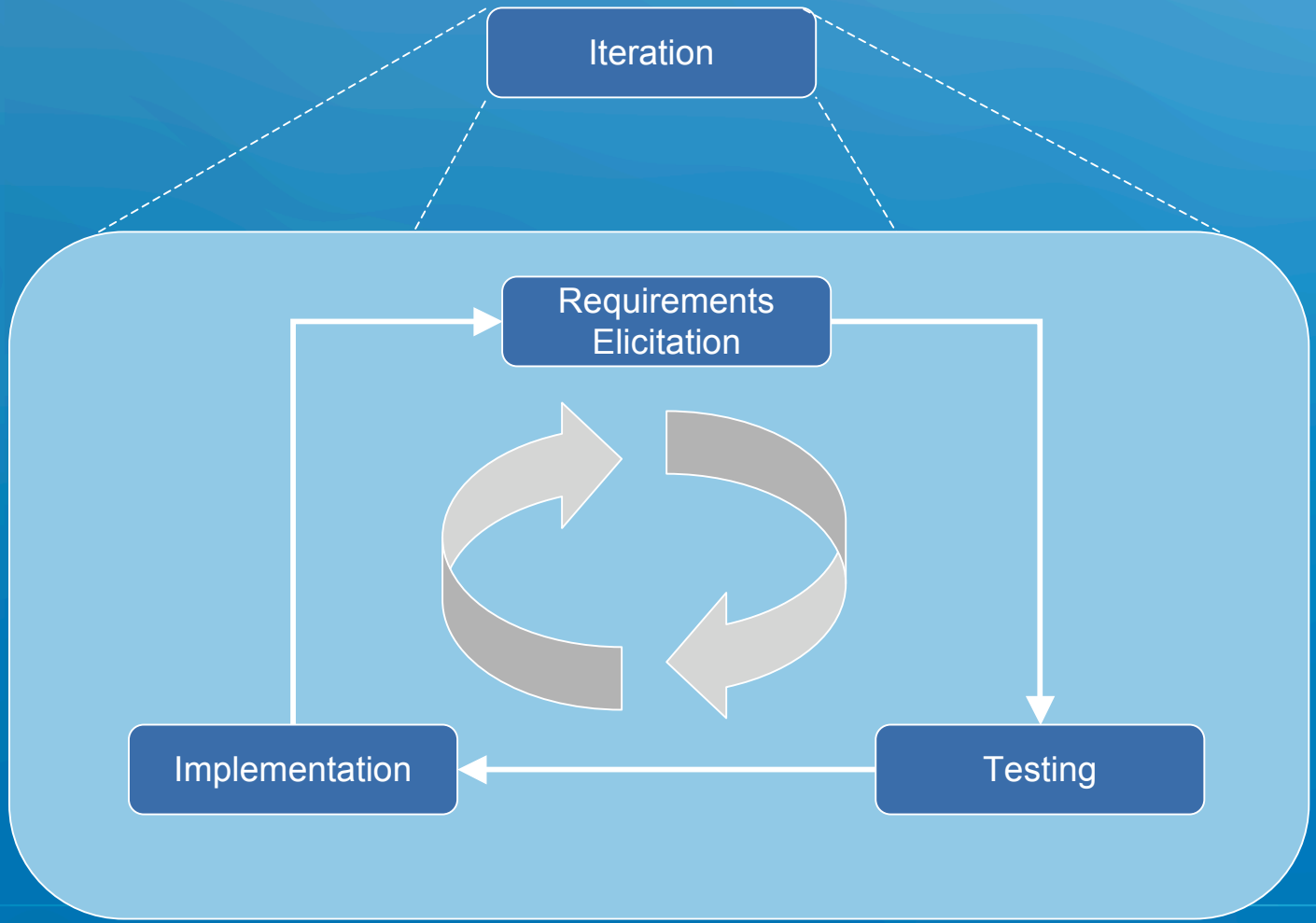
Sequential Model (Waterfall Model)



Iterative Model



Lightweight Lifecycle Model (XP)



Intermediate Summary

- There are different Software Engineering activities
- A lifecycle model describes:
  - Which activities to perform?
  - How do the different activities depend on each other?
- The activities are supported by techniques, methods and tools

 Next: Lecture Schedule

Schedule (1/2)

Week1 22.4.2009	UML Class Diagrams
Week2 29.4.2009	Testing
Week3 6.5.2009	Object Design
Week4 13.5.2009	Design Pattern 1
Week5 20.5.2009	Design Pattern 2
Week6 27.5.2009	Requirements Elicitation
Week7 3.6.2009	Analysis

Schedule (2/2)

Week8 10.6.2009	System Design 1
Week9 17.6.2009	System Design 2
Week10 24.6.2009	Testing 2
Week11 1.7.2009	Guest Speaker
Week12 8.7.2009	Methodologies
Week13 15.7.2009	XP and Scrum
Week144 22.7.2009	Putting everything together

Software Engineering for Engineers

UML Class Diagrams

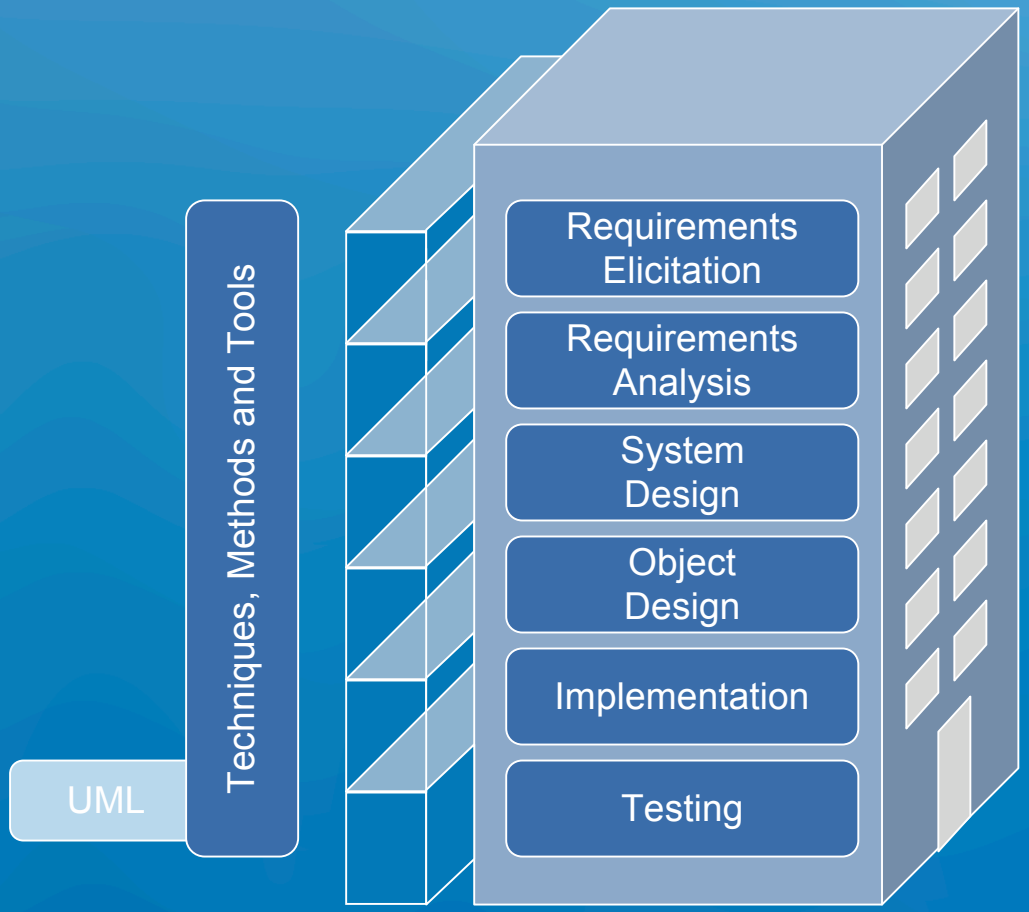
Outline

- ✓ Organizational issues
  - ✓ Time and location
  - ✓ Grading criteria
  - ✓ Exercises
- ✓ The development challenge
- ✓ Software Engineering activities
- ✓ Lecture Schedule

- What is UML and why do we use it?

- UML Class Diagram
  - Associations
  - Inheritance
  - UML to Java

Where are we?


Techniques, Methods and Tools
UML
Requirements Elicitation
Requirements Analysis
System Design
Object Design
Implementation
Testing

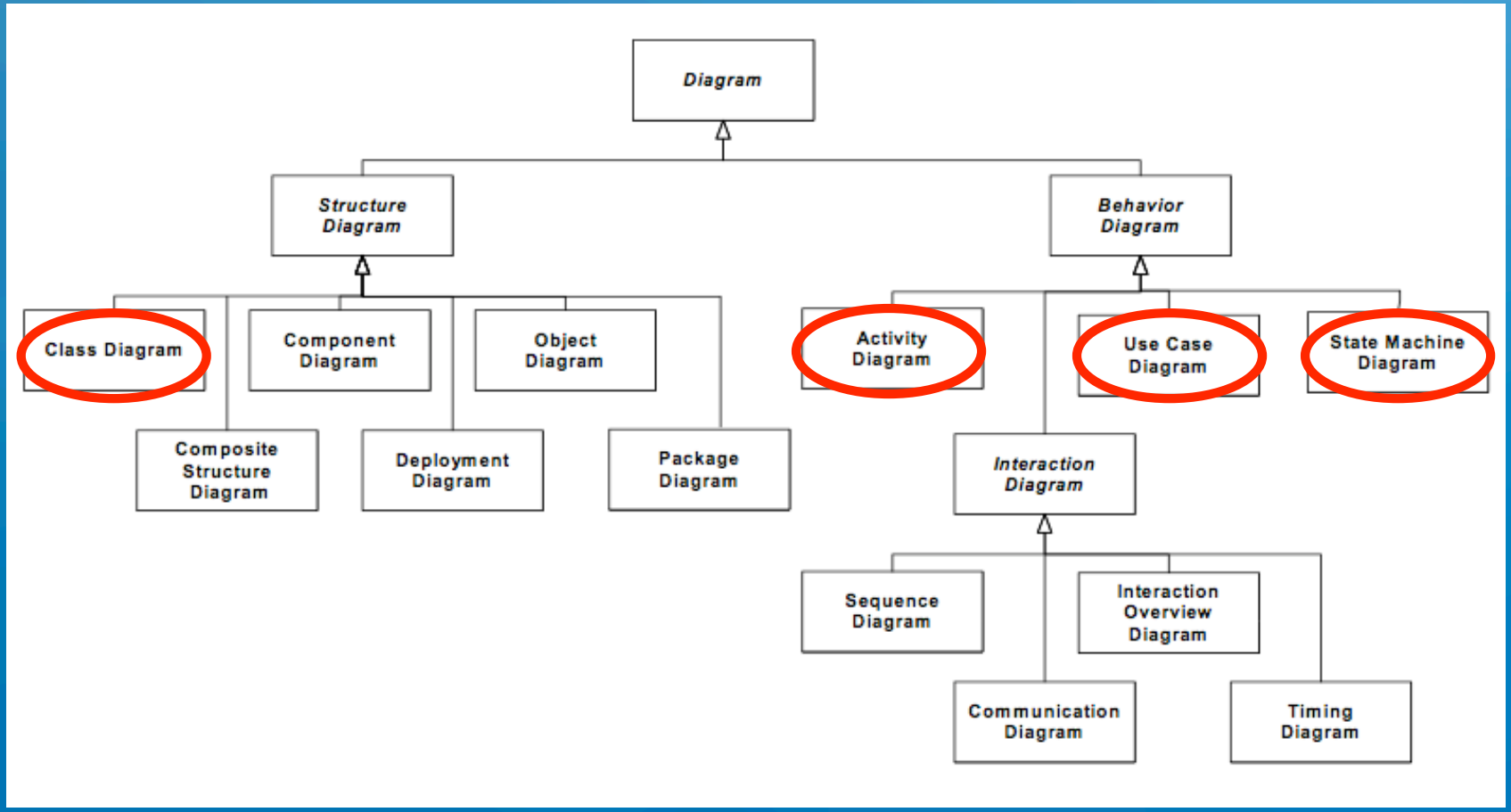
What is UML?

- UML (Unified Modeling Language)
  - Convergence of notations used in object-oriented methods
    - OMT (James Rumbaugh and colleagues)
    - Booch (Grady Booch)
    - OOSE (Ivar Jacobson)
- Current version 2.1.2
  - Information at the UML portal http://www.uml.org/
- Commercial CASE tools: Rational Rose (IBM), Together (Borland), Visual Architect (business processes, BCD)
- Open Source CASE tools: ArgoUML, StarUML, Umbrello, Unicase
- Commercial as well as Open Source: PoseidonUML (Gentleware)

We use Models to describe Software Systems

- **System model:** Object model + functional model + dynamic model
- **Object model:** What is the structure of the system?
  - UML Notation: Class diagrams
- **Functional model:** What are the functions of the system?
  - UML Notation: Use case diagrams
- **Dynamic model:** How does the system react to external events?
  - UML Notation: Sequence, State chart and Activity diagrams

Another view on UML Diagrams



Where are we now?

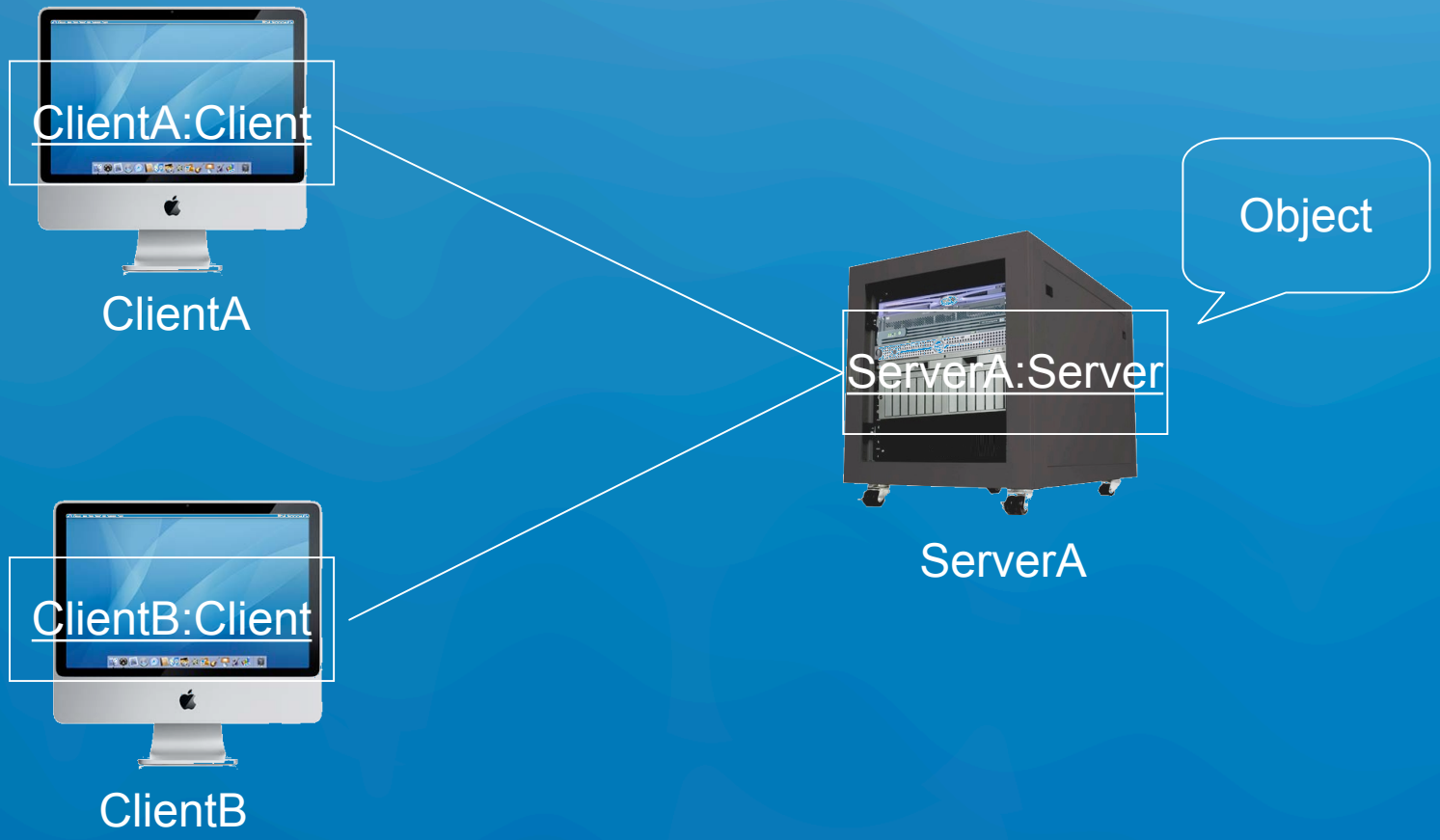
- ✓ What is UML and why do we use it?

- UML Class Diagram
  - Associations
  - Inheritance
  - UML to Java

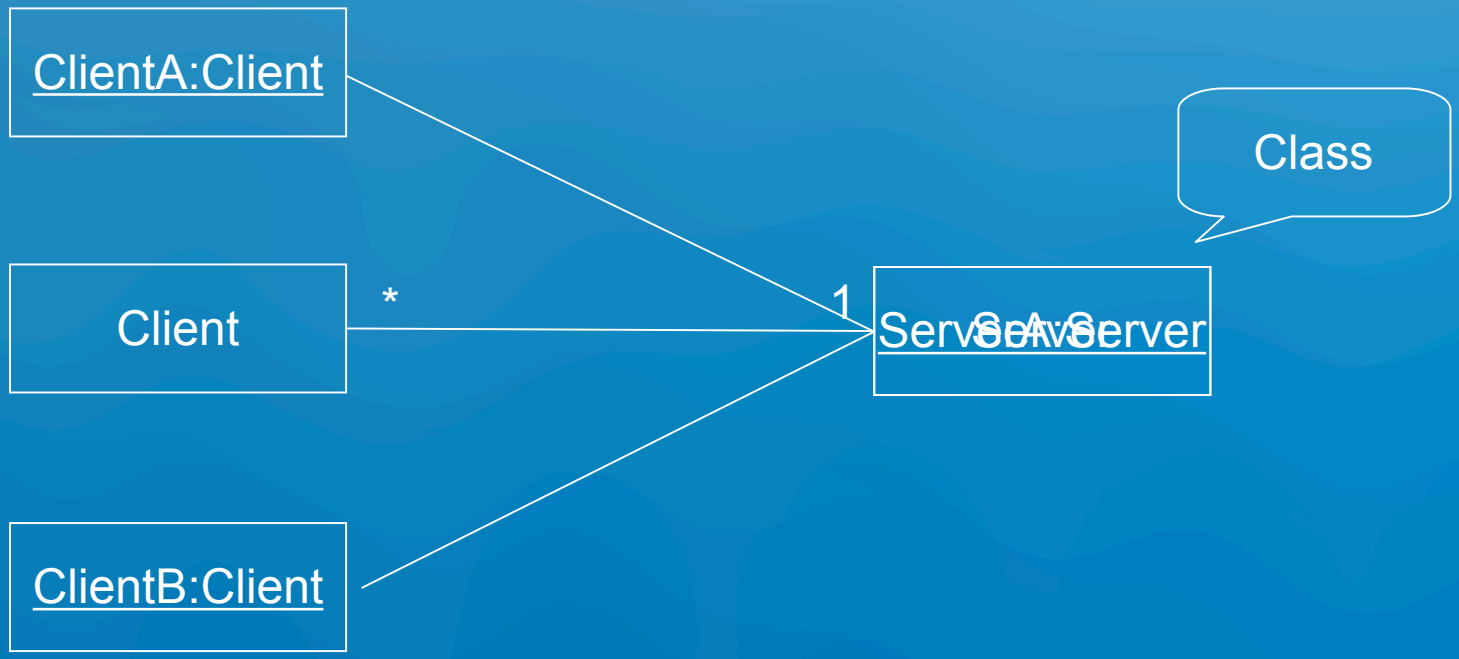
Models are abstractions

- Different audiences require different abstractions
- In Software Engineering we call different members of audience stakeholders
- Example for stakeholders:
  - Clients
  - End-users
  - Venture Capitalist
  - Developers
  - **CASE-Tools**

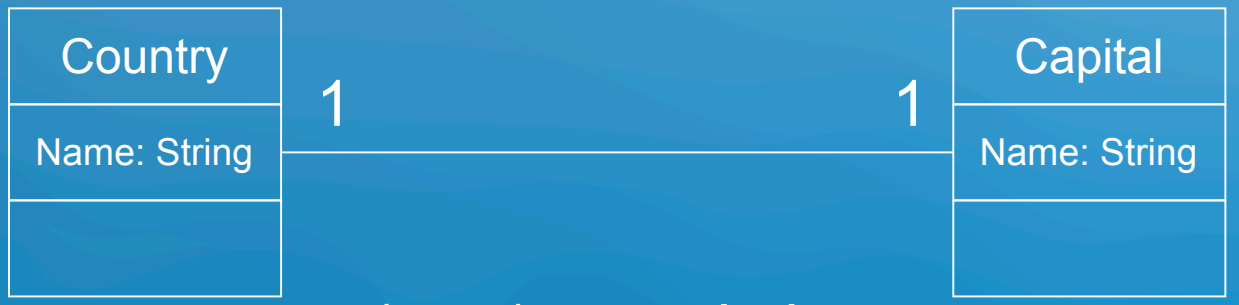
Modeling a computer network for a cluster



Modeling a computer network for a developer



1-to-1 and 1-to-many Associations



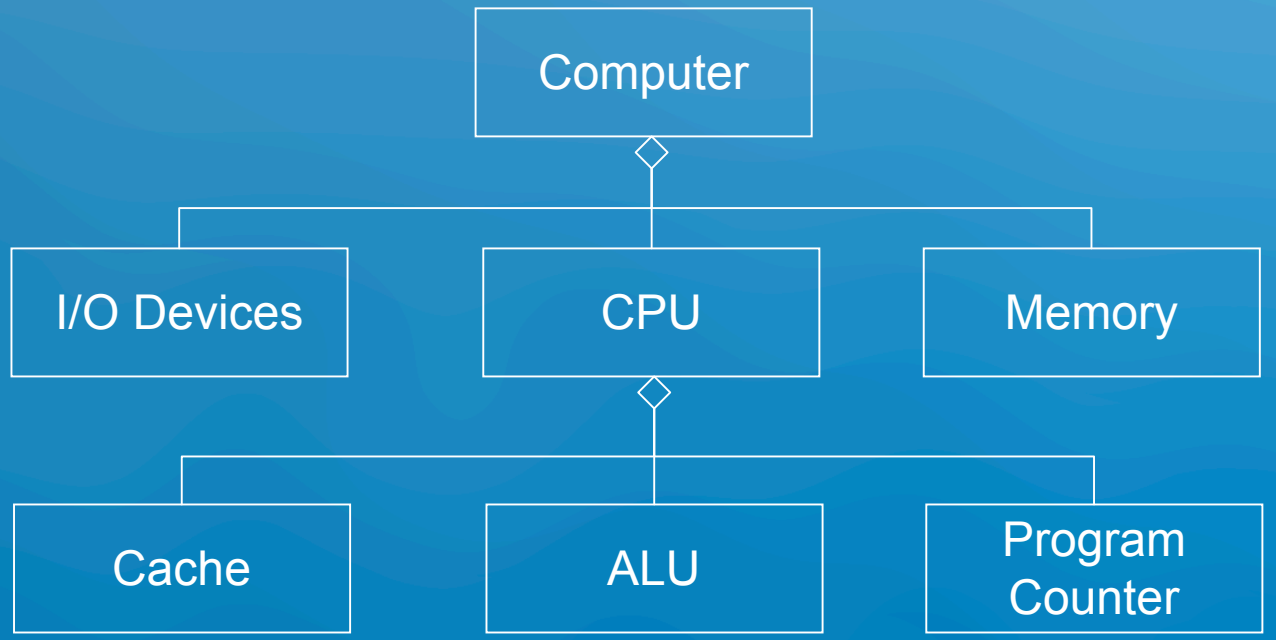
1-to-1 association

1-to-many association

Many-to-many Associations

- A stock exchange lists many companies.
- Each company is identified by a ticker symbol

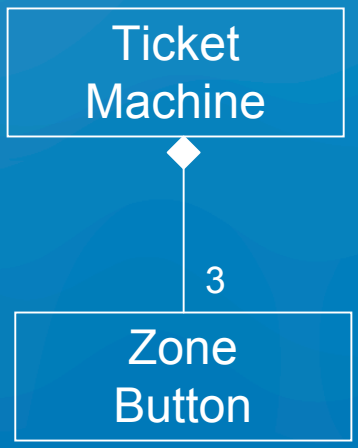
Part-of Hierarchy (Aggregation)



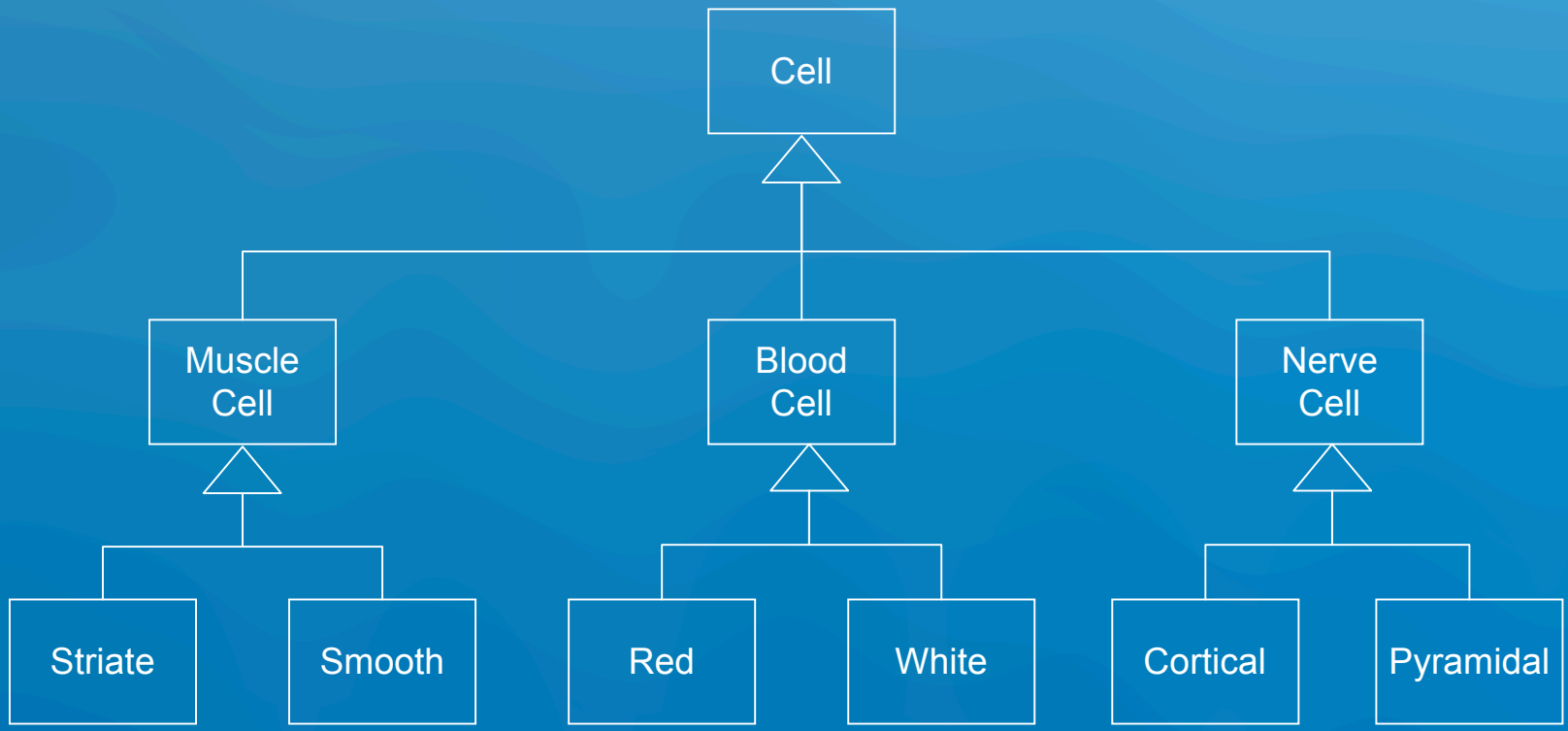
- An aggregation is a special case of association denoting a “consists-of” hierarchy
- The aggregate is the parent class, the components are the children classes

Composition

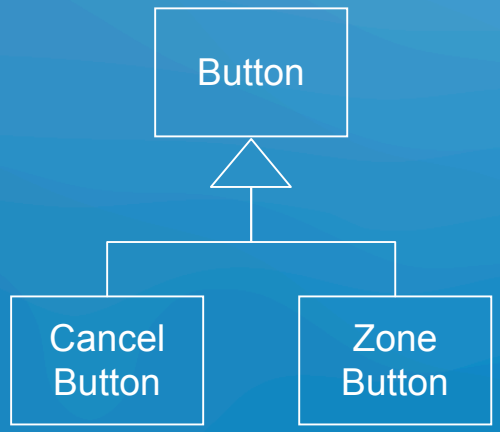
- A solid diamond denotes composition: A strong form of aggregation where the life time of the component instances is controlled by the aggregate (“the whole controls/destroys the parts”)



Is-Kind-of Hierarchy (Taxonomy)

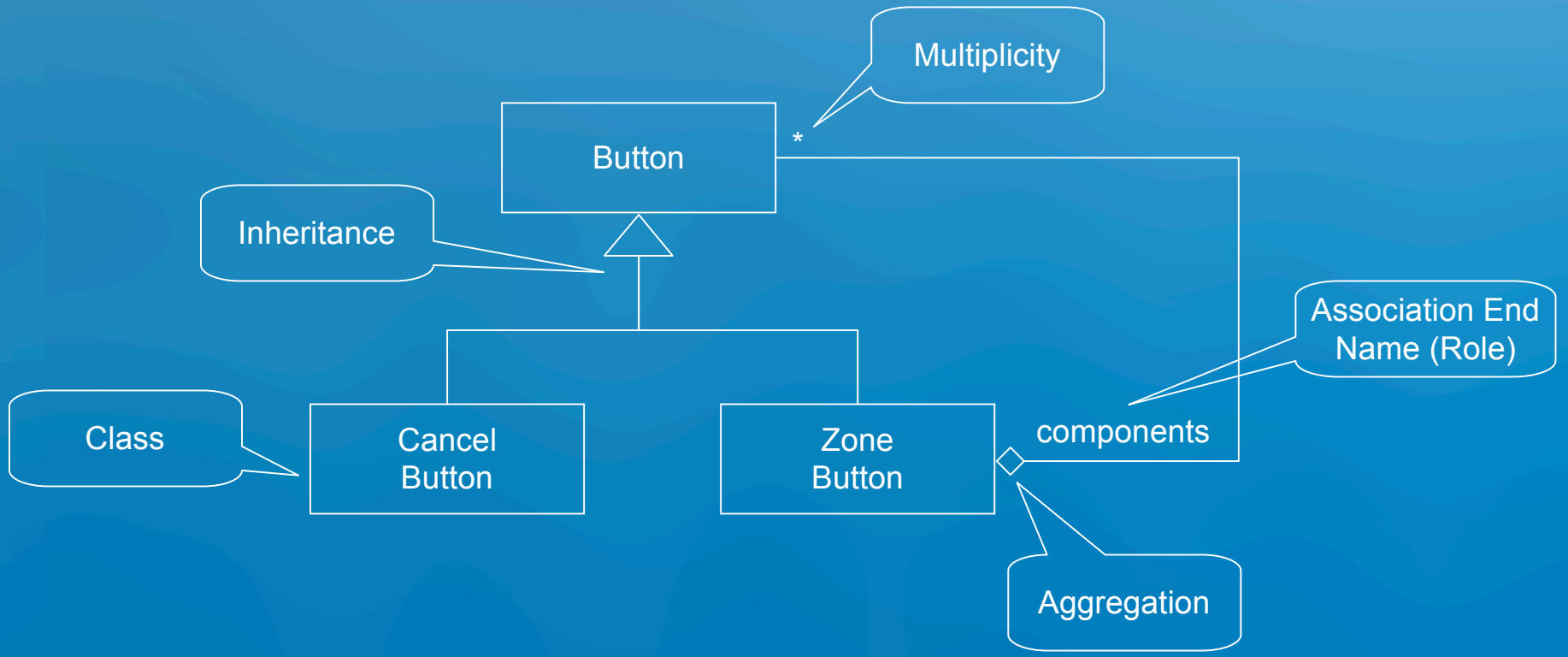


Inheritance



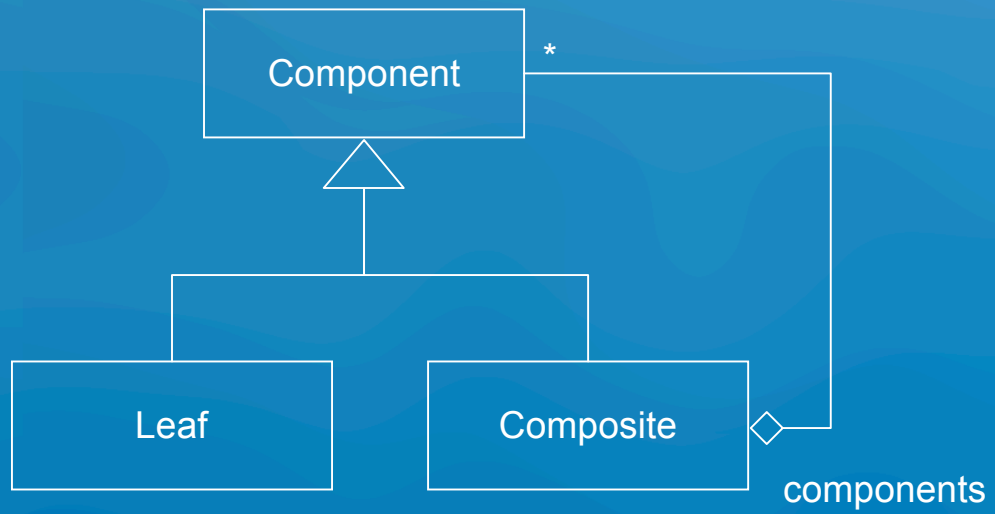
- ***Inheritance*** is another special case of an association denoting a “kind-of” hierarchy
- Inheritance simplifies the analysis model by introducing a taxonomy
- The **children classes** inherit the attributes and operations of the **parent class.**

Class diagram: Basic Notations



Class diagrams represent the structure of the system

Code Generation from UML to Java I




```
public class Component{  }

public class Leaf extends
   Component{  }

public class Composite extends
   Component{
   private Collection<Component>
   components;
   …
}
```

Excursion: Packages

- Packages help you to organize UML models to increase their readability
- We can use the UML package mechanism to organize classes into subsystems

- Any complex system can be decomposed into subsystems, where each subsystem is modeled as a package.