

## 0. Object Orientation: An Overview

---

**Subject/Topic/Focus:**

- Overview over this lecture

**Summary:**

- Lecturer, lecture, rooms, assistants, lab classes, credit points ...
- Need for systems analysis and software engineers
- Literature and recommended readings
- Topics covered by this lecture
  - Object-Oriented Development: Java
  - Object-Oriented Analysis and Design: UML

## Object-Oriented Analysis and Design

---

**Lecturer:**

Prof. Dr. J.W. Schmidt  
Software Systems (STS)  
Harburger Schloßstraße 20  
Phone: 040 / 42878 - 3460

**Subject:**

- Object-Oriented Development in Java
- Object-Oriented Analysis and Design using UML

**2 Exams:**

- Java: 2 credit points
- UML: 2 credit points

**Schedule:**

- **Lecture:** Tuesday, 12:15h-13:45h, ES38 - 007
- **Labs:** Monday, 15:15h-16:45h, Harburger Schloßstraße 20, Room 014, CIP Pool 2b

**Assistants:**

○ Claudia Niederee (Java)  
Phone: 42878-2929, Room: 223  
email: c.niederee@tu-harburg.de

○ Michael Skusa (UML)  
Phone: 42878-2795, Room: 207  
email: skusa@tu-harburg.de

<http://www.sts.tu-harburg.de/teaching/ws-99.00/OOA+D/entry.html>

## The Tar Pit



„No scene from prehistory is quite so vivid as that of the mortal struggles of great beasts in the tar pit.

Large-system programming has over the past decade been such a tar pit“

[Frederick P. Brooks. jr; The mythical man-month, Addison-Wesley, 1972]

## The Good News

- **Many jobs** available under labels such as systems analyst, database administrator, applications programmer, information officer,...
- List of employers offering such jobs includes **management consulting companies** -- who sell consulting, **development and maintenance services** -- and large organizations such as **banks, telephone companies, government departments**,... who run and depend on information services.
- Most IT jobs are with small companies -- the majority of the software companies are in application development or other **information system-related areas**.

[John Mylopoulos, Information Systems Analysis and Design, 1997]

## The Need for Systems Analysis <sup>(1)</sup>

---

### The age of innocence:

- Once upon a time, the information processes in an organization were simple and small - easy to handle with one or two people.
- As the organization grew, it took on additional tasks and had to handle information about more people or things important to the organization, for example, statistical data on production and sales, design data, financial data etc.
- As the organization grew, the information workers within the organization learned their information management tasks (classification, reference) well enough to add on more complex tasks.
- As the organization grew, it split into subunits to facilitate management, but this often acted as a barrier to information flow. Still, there was one organizational information system.
- As the organization evolved, so did its organizational information system, but this evolution was not documented.

## The Need for Systems Analysis <sup>(2)</sup>

---

### Fall from Eden:

- People who knew the existing organizational information system retired or took employment elsewhere.
- The external environment for the organization was constantly changing, affecting differently different units in the organization. Suddenly the organization's information system was no longer integrated.
- The external environment changed sufficiently so that the techniques used for managing and processing the information were no longer efficient.
- Suddenly, there was no one person who knew exactly what the information processes in the organization were all about.
- Documentation as to what the processes were was completely out of date or non-existent.
- The information was available but it was in the heads of many employees.
- The organization needed to respond to the external environment, e.g., its customers, in new ways because the environment had changed.

## The Bad News

---

- 30% of large software projects are cancelled before completion.
- 50% of software projects are overbudget by more than 200%.
- The majority of completed projects deliver 60% or less of prescribed functionality.
- Many delivered information systems are under-used because they do not meet user needs and/or expectations.
- Legacy systems are a serious and growing bottleneck to organizational evolution.

## Murphy's Law

---

**If something can go wrong it will.**

- Things are more complex as they seem.
- Things take longer than expected.
- Things cost more than expected.

Note also the corollary: **Murphy was an optimist.**

... applies directly to software development!

To make things get ~~worse~~ <sup>better</sup>, we need software-engineers!

## Literature and Recommended Readings <sup>(1)</sup>

---

Bruce Eckel:

Thinking in Java. Prentice-Hall 1998.

Martin Fowler with Kendall Scott:

UML Distilled, Second Edition, Addison-Wesley, 1999.

James Rumbaugh, Ivar Jacobson, Grady Booch:

The Unified Modeling Language Reference Manual. Addison-Wesley, 1999.

Grady Booch, James Rumbaugh, Ivar Jacobson:

The Unified Modeling Language User Guide. Addison-Wesley, 1998.

Craig Larman:

Applying UML and Patterns, Prentice-Hall, 1997.

## Additional Literature

---

Ivar Jacobson, Grady Booch, James Rumbaugh:

The Unified Software Development Process. Addison-Wesley, 1999.

I. Jacobson et.al.:

Object-Oriented Software Engineering - A use case driven approach,  
Addison-Wesley 1996 .

I. Sommerville:

Software Engineering, Addison-Wesley 1995 (5th ed.).

H. Rumbaugh, M. Blaha, W. Premarlani, F. Eddy, W. Lorensen:

Object-Oriented Modelling and Design, Prentice-Hall, 1995.

G. Booch:

Object-Oriented Analysis and Design with Applications,  
Addison-Wesley, 1994 (2nd ed.).

Bertrand Meyer:

Object-oriented Software Construction, Prentice Hall, 1988.

Frederick P. Brooks, jr:

The Mythical Man - Month, Addison-Wesley, 1972.

## Prerequisites

---

### User experience in

- application systems (Microsoft Office, ...)
- operating systems (Unix, ...)
- programming tools (compiler, editor, ...)

### Programming experience

- small projects (at least one week of work, approx. 500 lines of code)
- in any programming language (Pascal, Modula-2, C, LISP, ...)

### Basic object-oriented knowledge

- classes, methods, inheritance
- in any programming language (Java, C++, Eiffel, ...)

will be taught in this course

## Overview

---

### Object-Oriented Software Development in Java

- Language Fundamentals
- System Fundamentals
- Information Hiding
- Reusing Classes
- Polymorphism

**+ Exercises**

### Object-Oriented Analysis and Design using the Unified Modeling Language (UML)

- Software Development Process
- Modeling the System Usage
- Modeling the Static Structure
- Modeling the Dynamic Behavior
- Structuring the System

**+ Exercises**

## Object-Oriented Development in Java

### Language Fundamentals

- Base types
- Variables
- Control structures

### System Fundamentals

- Main method
- Working with the JDK

### Information Hiding

- Class Definition
- Data Members & Methods
- Object Creation
- Packages, Scopes
- Modifiers

### Reusing Classes

- Composition and Inheritance
- Constructors and Subclasses
- Method Overriding

### Polymorphism

- Substitutability
- Abstract Classes
- Interfaces
- Inner Classes

## Object-Oriented Analysis and Design using UML

### UML = Unified Modeling Language

#### ○ Software Development Process

- Inception
- Elaboration
- Construction
- Iteration

#### ○ Modeling the System Usage

- Actors
- Use Cases

#### ○ Modeling the Static Structure

- Classes and Attributes
- Inheritance
- Association

#### ○ Modeling the Dynamic Behavior

- Methods
- States
- Activities
- Interactions

#### ○ Structuring the System

- Packages
- Patterns

## OOAD: Overview

---

0. Object Orientation: An Overview	19. October 1999
1. Object-Oriented Programming with Java	
1.1 Introduction to Objects & Language Fundamentals	26. October 1999
1.2 Objects, Classes and Information Hiding	2. November 1999
1.3 Reusing Classes	9. November 1999
1.4 Polymorphism	16. November 1999

## OOAD: Overview

---

2. Analysis, Design, and Implementation	30. November 1999
3. Object-Oriented Modeling using UML	
3.0 Overview	} 7. December 1999
3.1 Use Cases	
3.2 Class Diagrams in Analysis	14. December 1999
3.3 Class Diagrams in Design	4. January 2000
3.4 Activity Diagrams	} 11. January 2000
3.5 Interaction Diagrams	
3.6 State Diagrams	} 18. January 2000
3.7 Architecture Diagrams	
3.8 Applying UML	→ Exercises
4. Software Project Management	} 25. January 2000 & 1. February 2000
5. Conclusions & Feedback	