Software Engineering Essentials

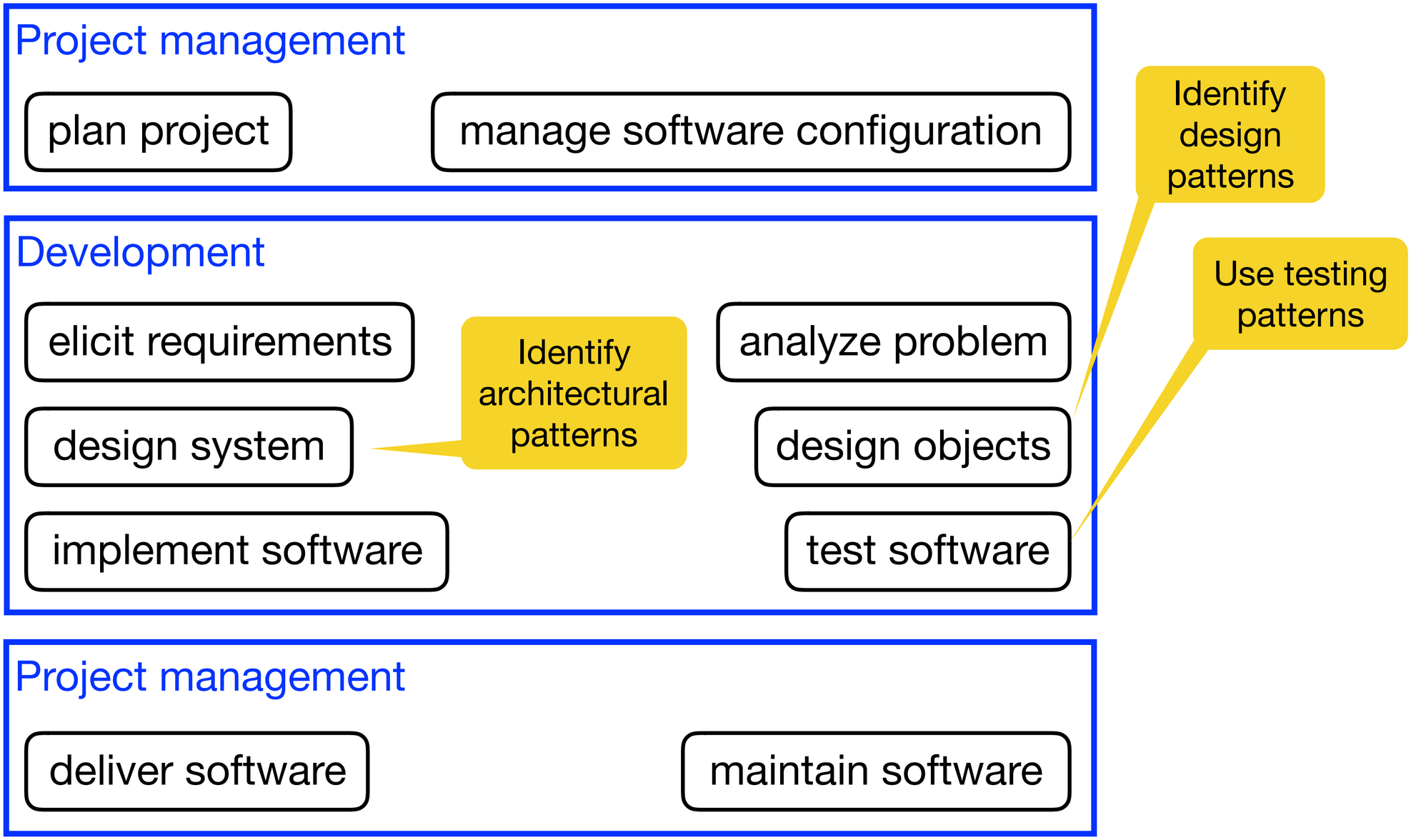
# Summary 1

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics

# Learning Goals

1) Review the main concepts of the course
2) Understand the transition from a problem statement to a software release
3) Remember the key facts about applied software engineering

# Main software engineering activities



# Project management and communication

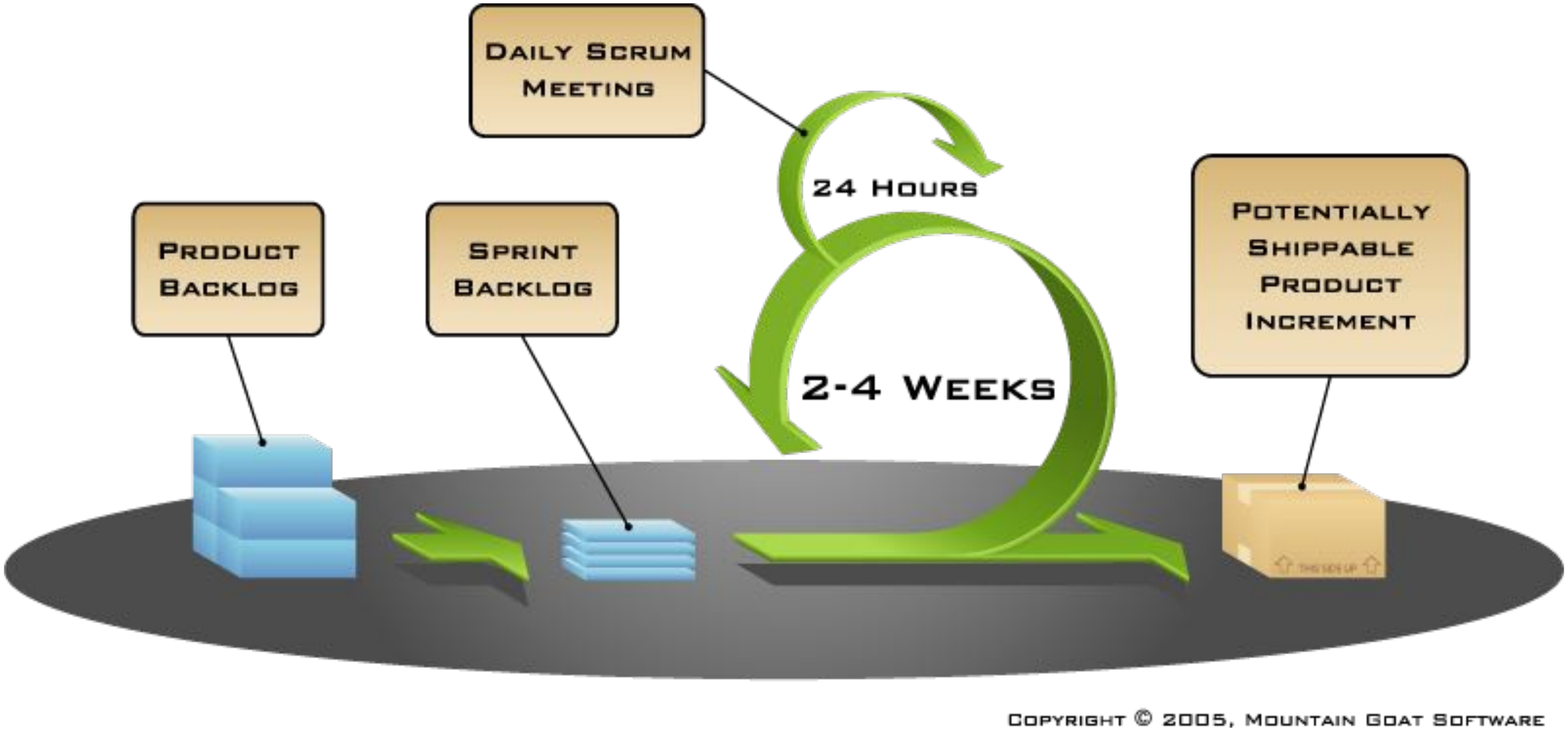
Collection of techniques, methodologies, tools and heuristics to develop

- A **high quality software** system
- With a **given budget**
- Before a **given deadline**
- **While change occurs**

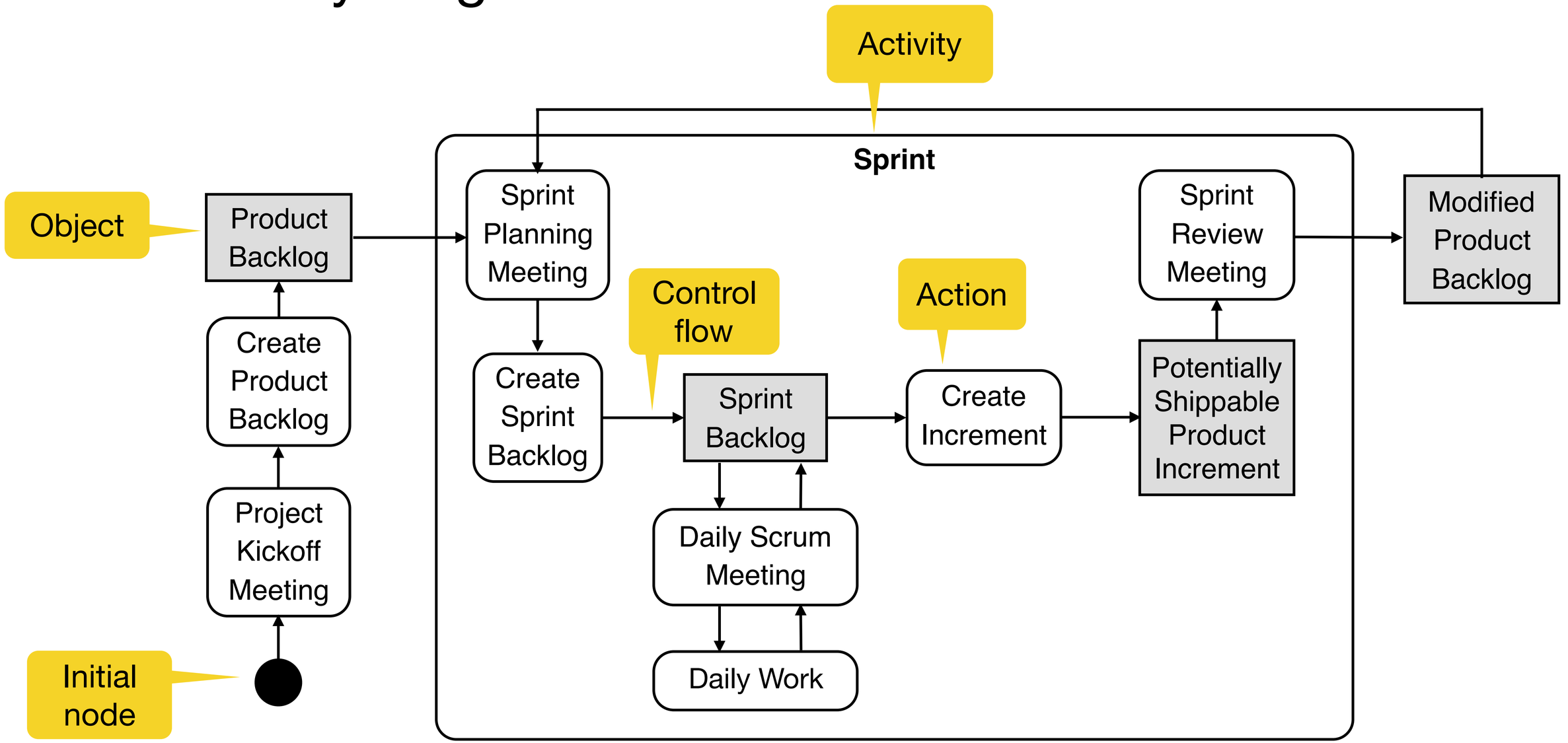
**Communication:** Important skill, clear and accurate communication is critical for the success of a project

**Meeting management:** procedure to plan and ensure productive meetings, typical roles: facilitator, minute take, time keeper

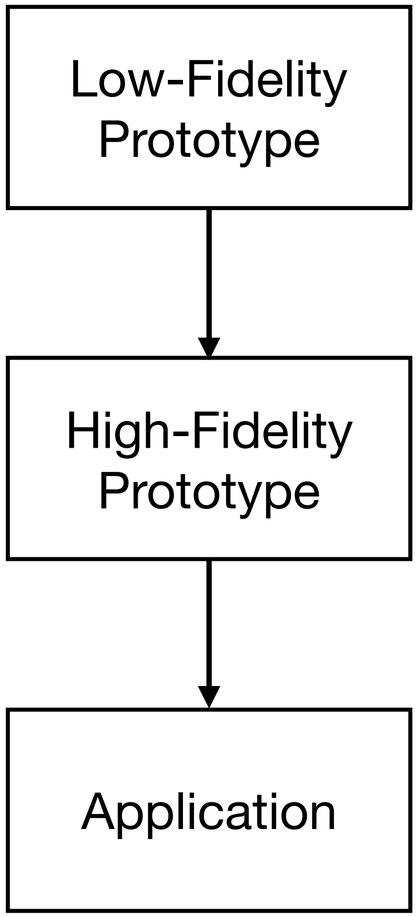
# Agile methods

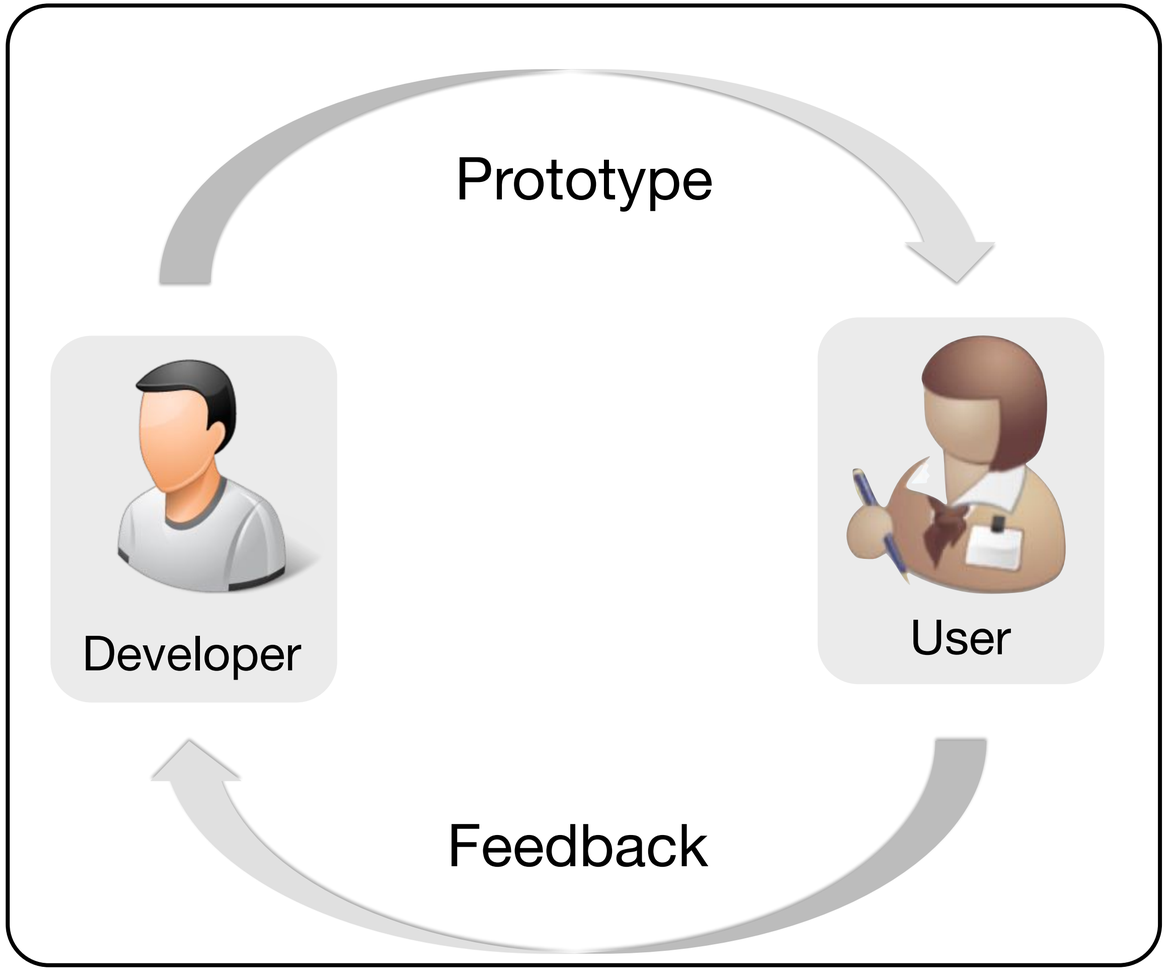


# UML activity diagram



# Prototyping: an iterative approach



# Software configuration management

## 1) Configuration item identification

- Modeling the system as set of evolving components

## 2) Promotion management

- Creation of versions for other developers

## 3) Build and release management

- Creation of versions for customers and end users

## 4) Change management

- Handling, approval & tracking of change requests

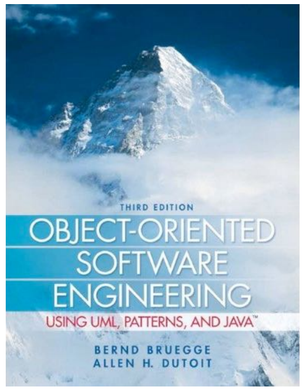
Covered in this course

## 5) Branch management

- Management of concurrent development

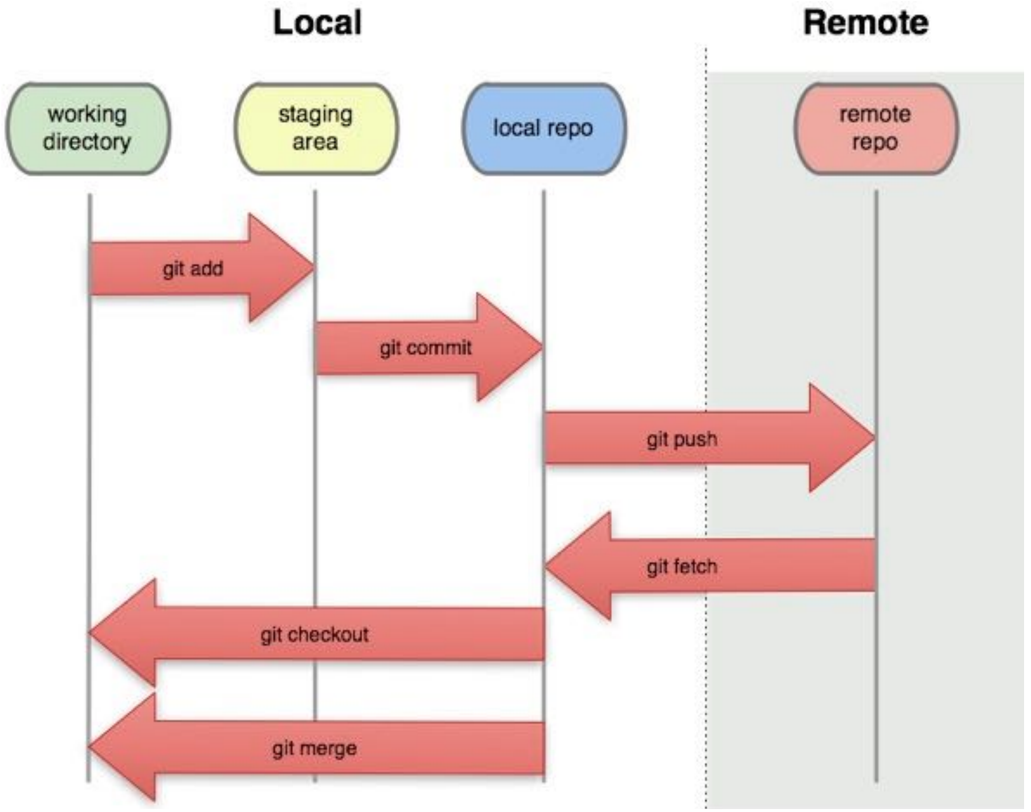
## 6) Variant management

- Management of coexisting versions




Bruegge, Dutoit: Object-Oriented Software Engineering Using UML, Patterns, and Java (Chapter 13)

# Distributed version control




Lightweight branching and offline commits

Most important git commands

- add: add changed files to the staging area
- commit: commit selected changed files of the staging area to your local repository
- push: upload local commits to a remote repository
- pull (fetch & merge): download and merge remote commits into your working copy
- clone (fetch & checkout): clone a complete repository into a new working directory

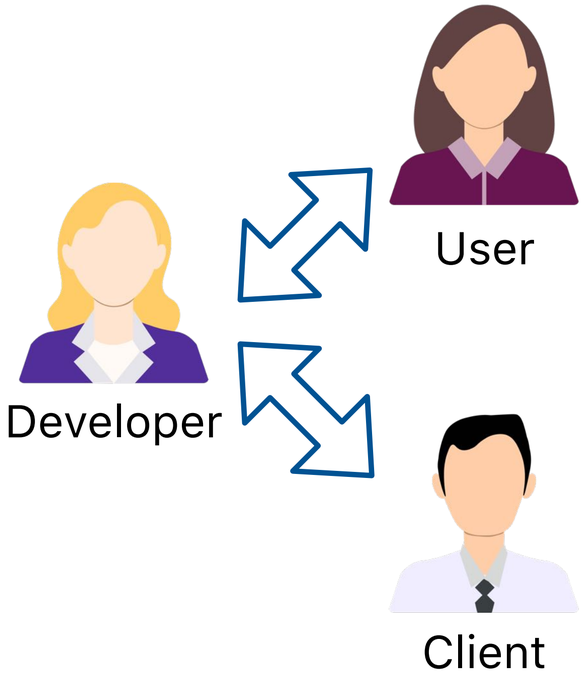
# Object oriented programming

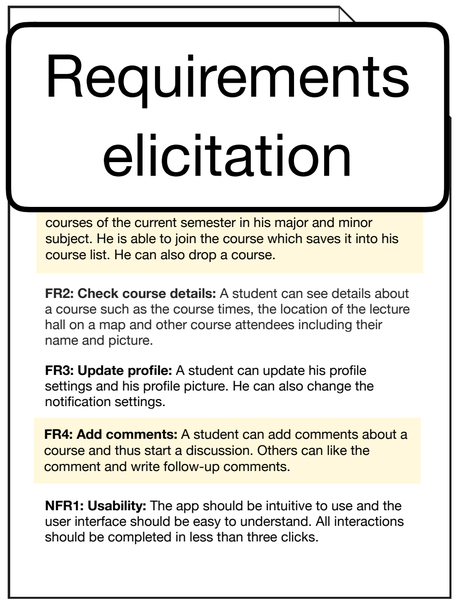
## 4 major principles

1) **Encapsulation:** objects define their structure through attributes and their functionality through methods
2) **Inheritance:** objects reuse structure and functionality of a super-class
3) **Polymorphism:** objects override functionality of a super-class
4) **Abstraction:** abstract classes and interfaces define structure and functionality to be shared among sub-classes, but cannot be instantiated

# Requirements elicitation

**Main goal:** create the requirements specification by talking to the client, end-users and other stakeholders



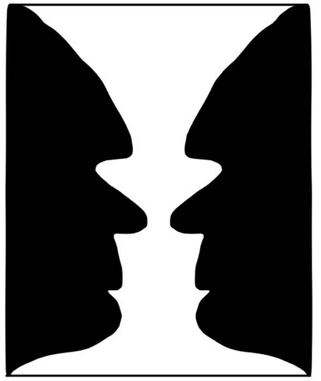

Problem Statement

Requirements specification
Non-functional requirements
Functional model

**Functional model:**
- Functional requirements
- Actors
- Scenarios
- Use cases

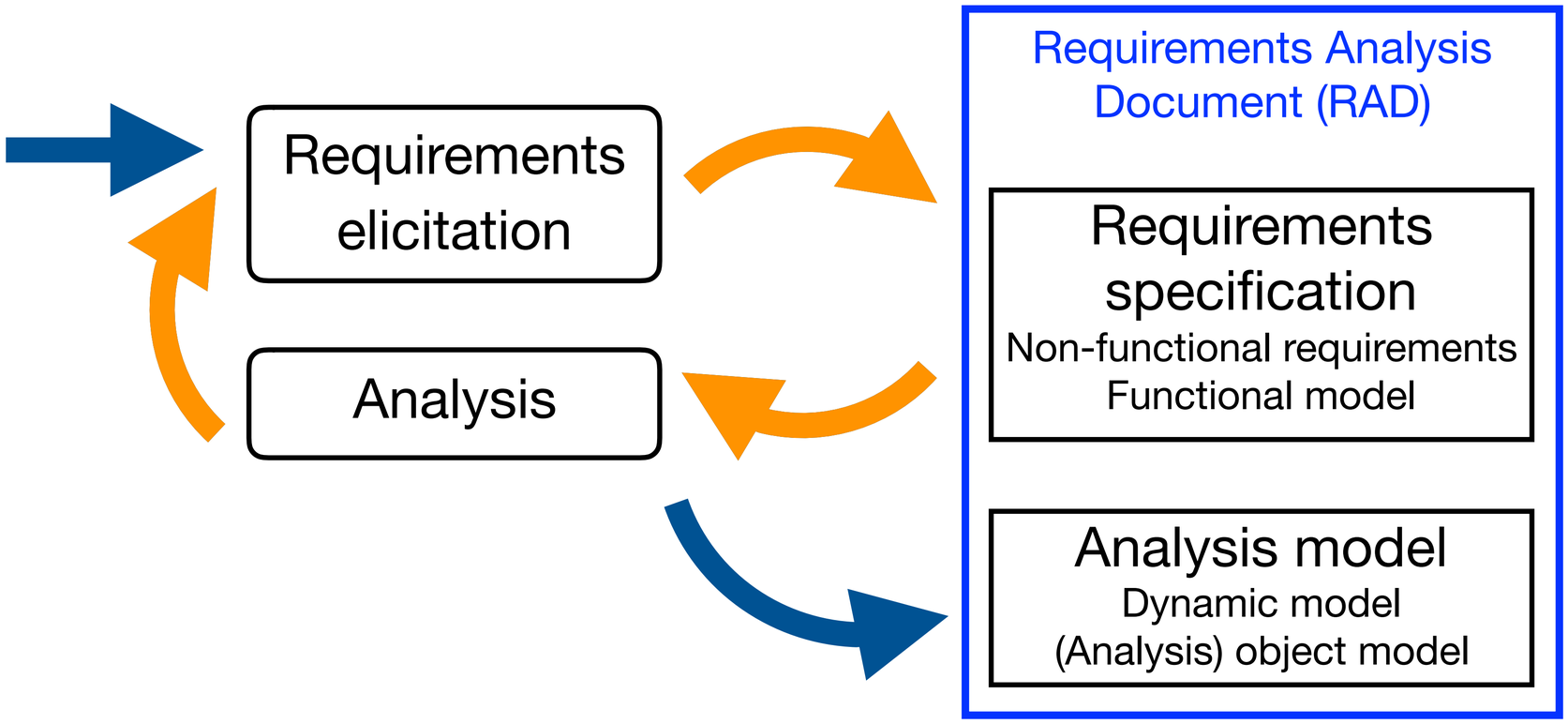
# Analysis

- Create the analysis model
- It needs to be correct, complete, consistent, and verifiable



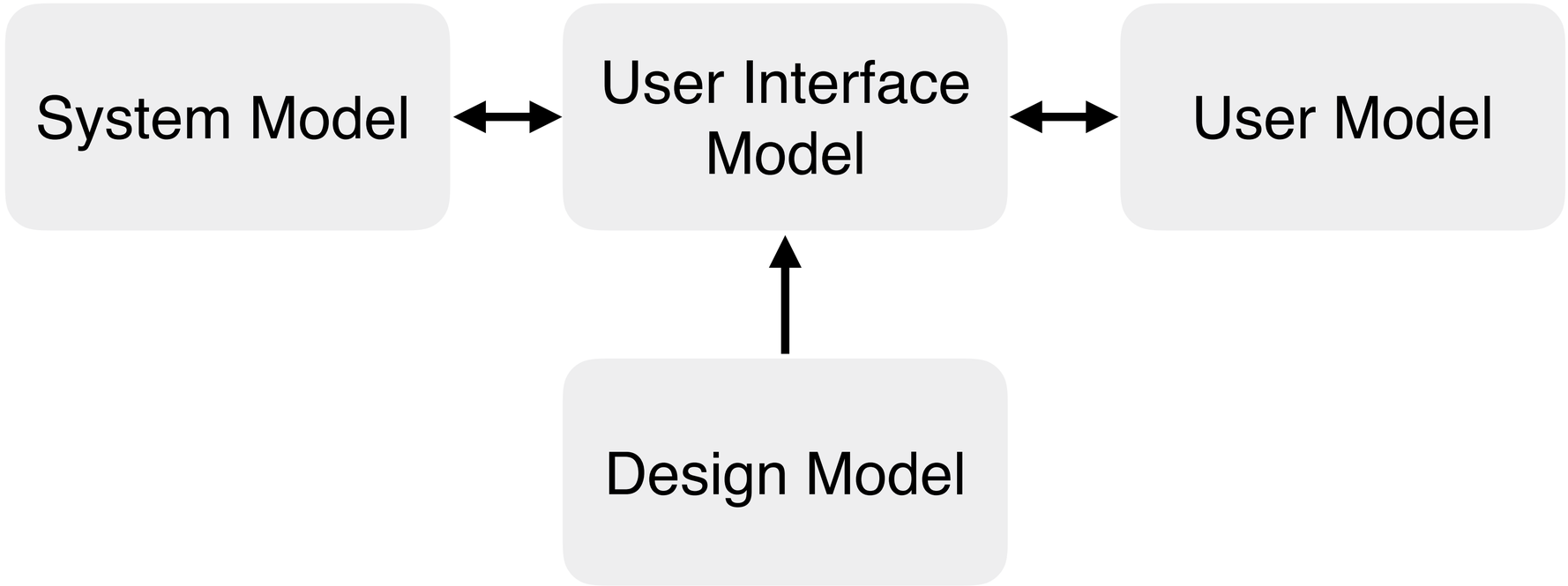
An optical Illusion

Ambiguity in drawings; We need to be consistent in our models

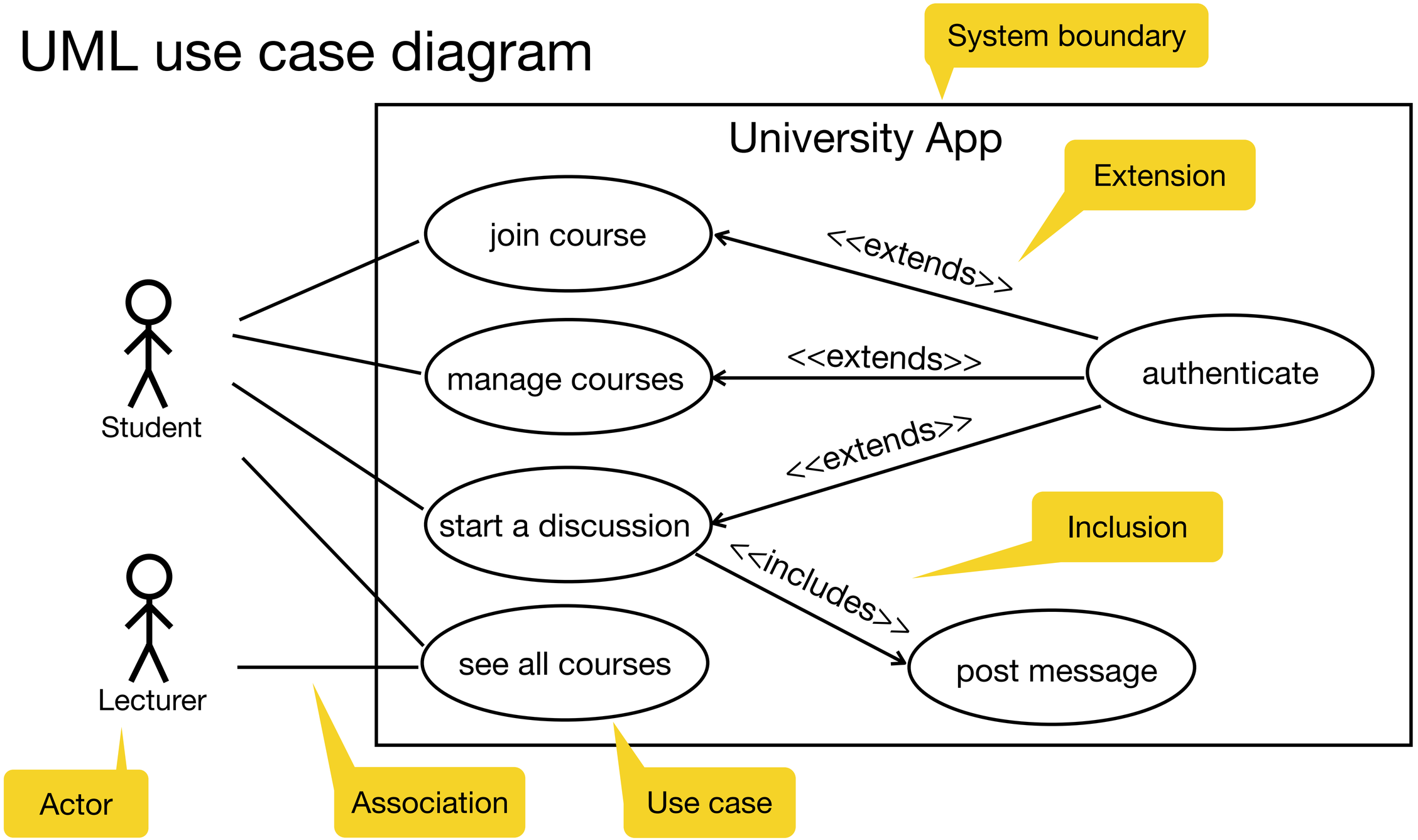


# Usability

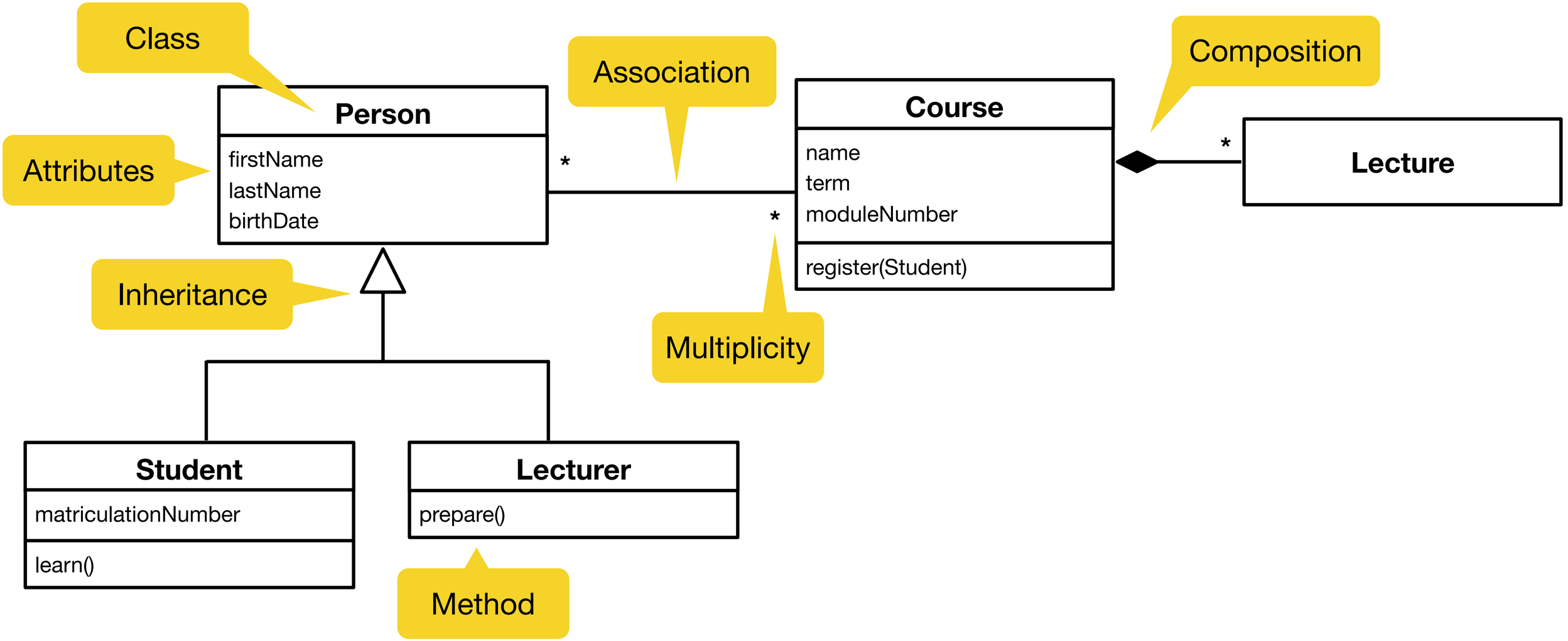
**Multidimensional non-functional requirement:** learnability, efficiency, memorability, error handling, satisfaction



# UML use case diagram



# Analysis object model (UML class diagram)



Software Engineering Essentials

# Summary 1

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics