

# Object-Oriented Analysis & Design (OOA&D)

Concepts, Principles & Methodology

## Chapter 1



# Systems development

- Methodology:
  - Guidelines for work processes (OOA&D)
  - Guidelines for documentation (UML)



# Systems development

## Analysis:

- Understand a system, its context, and the conditions for its implementation
- To determine system requirements

## Methodology:

- Guidelines for work processes (OOA&D)
- Guidelines for documentation (UML)

# Systems development

## ● Analysis:

- Understand a system, its context, and the conditions for its implementation
- To determine system requirements

## ● Design:

- Produce a system design without significant uncertainties

## ● Methodology:

- Guidelines for work processes (OOA&D)
- Guidelines for documentation (UML)



# Systems development

## ● Analysis:

- Understand a system, its context, and the conditions for its implementation
- To determine system requirements

## ● Design:


- Produce a system design without significant uncertainties

## ● Implementation:

- Realize a design on a technical platform

## ● Methodology:

- Guidelines for work processes (OOA&D)
- Guidelines for documentation (UML)



# A brief timeline of analysis methods in System Development

- function oriented (1970ties)
  - focus on data-processing, not data
- data oriented (1980ties)
  - all data available to all functions
- object oriented (1990ties-now)
  - encapsulation of data and functions
- process and service oriented (now->)
  - designing for computer supported work



# What is object-orientation?

- Object:
  - An entity with:
  - identity, state, and behavior
- An object belongs to a class
- Class:
  - A description of a collection of objects sharing:
  - structure, behavioral pattern, and attributes
- Each class contains a set of objects



# Objects in analysis and design

## Analysis:

- Phenomena outside the computer system
- Identity: identifies an object
- Behavior: the events an object have performed or suffered

## Design (and programming):

- Phenomena inside the computer system
- Identity: gets access to an object
- Behavior: the operations an object can perform on request and offers to other objects

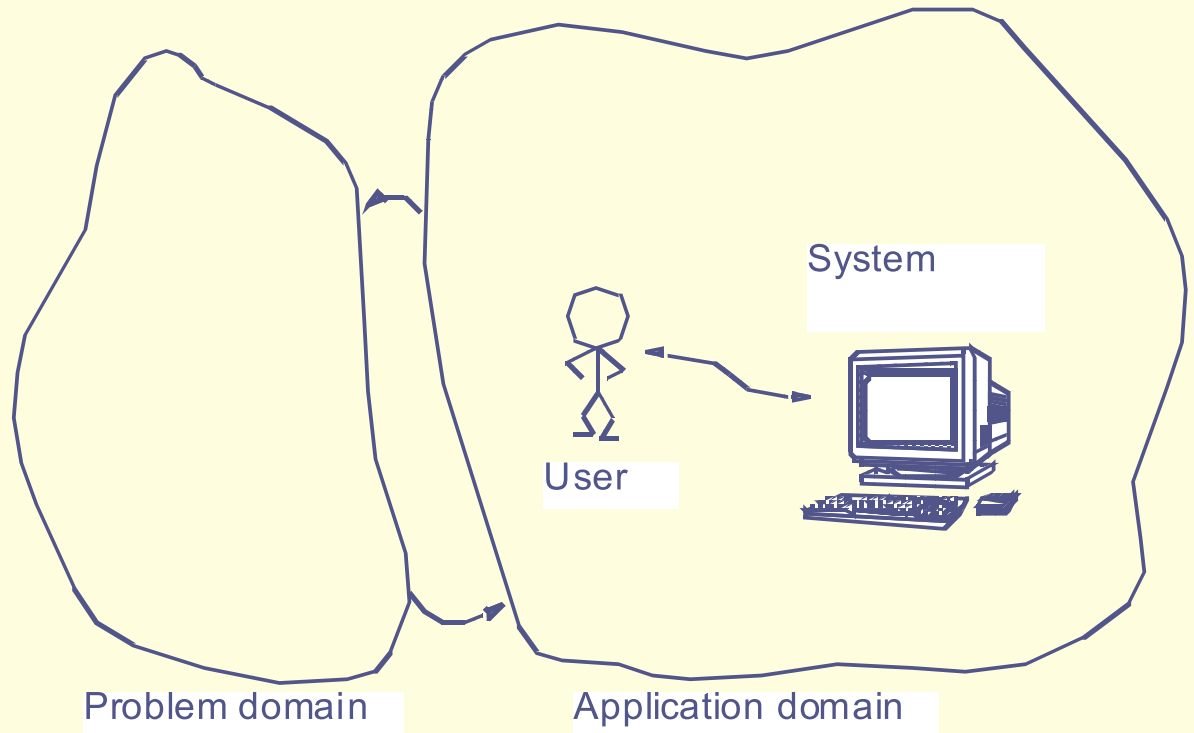




# Benefits of object-orientation

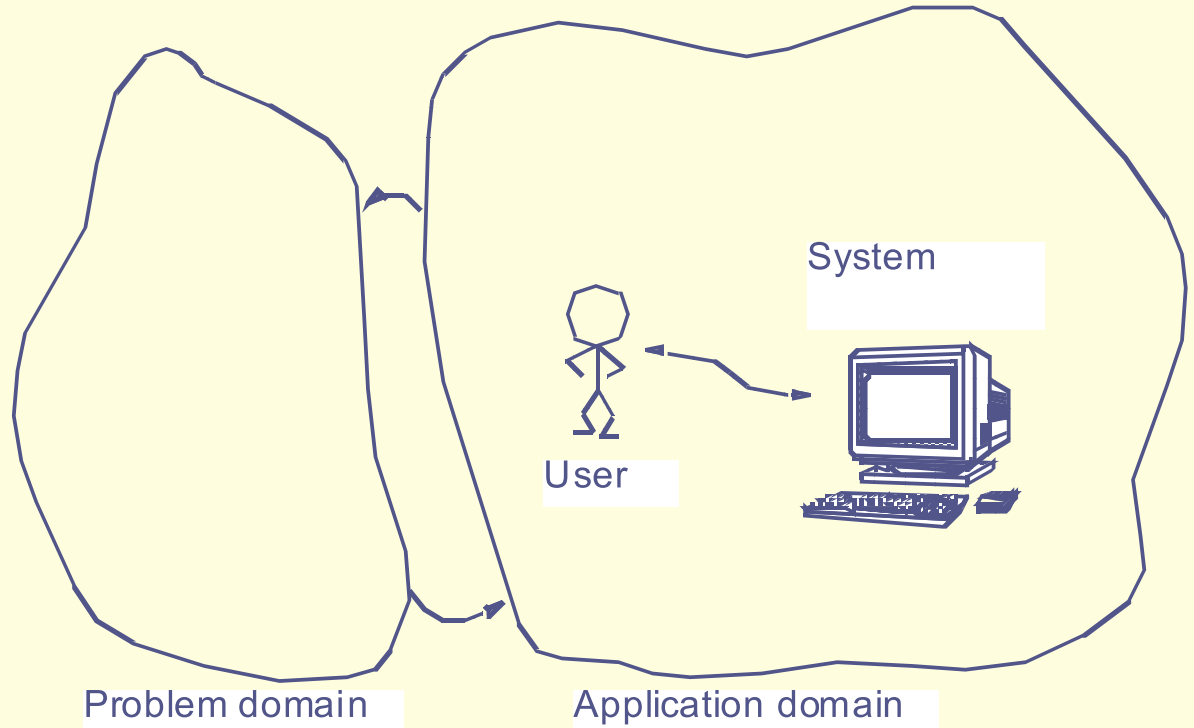
- Local cohesion: integrated description data and processing
- Activity cohesion: same concepts in analysis, design, programming and interfaces
- Global cohesion: reuse and cooperating systems

# Model the context



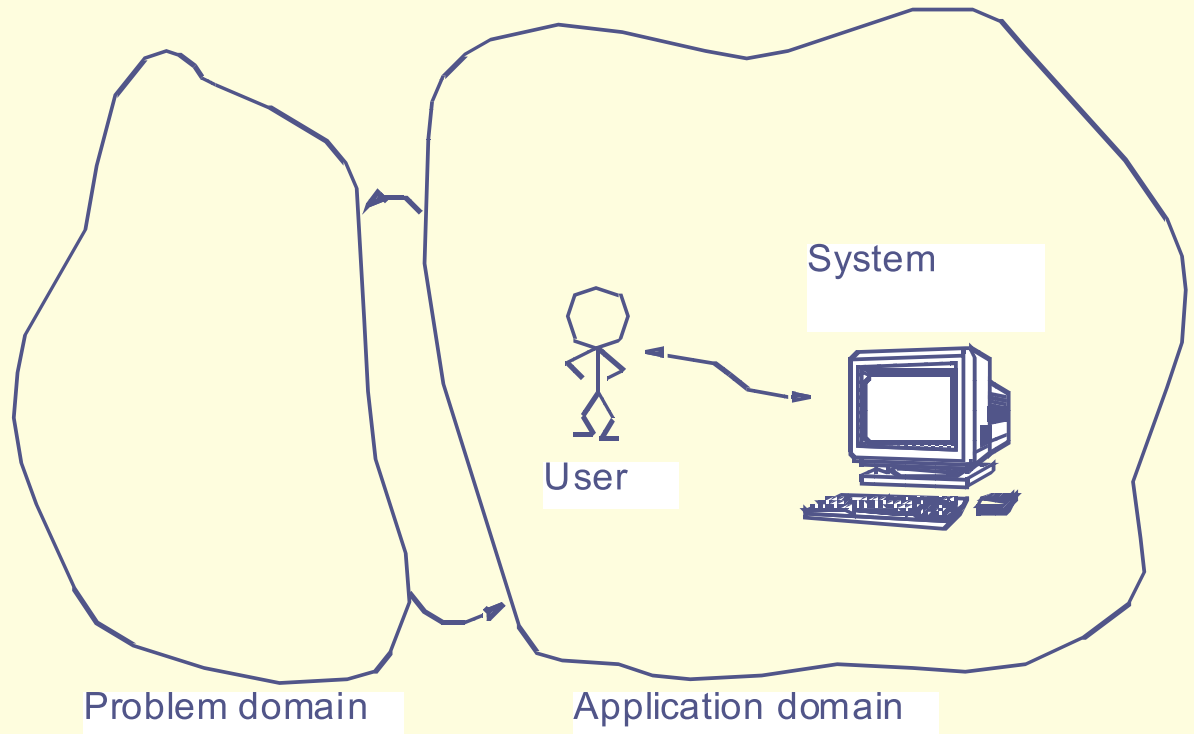
# Model the context

- Problem domain:  
That part of a context  
that is administrated,  
monitored, or  
controlled by a  
system

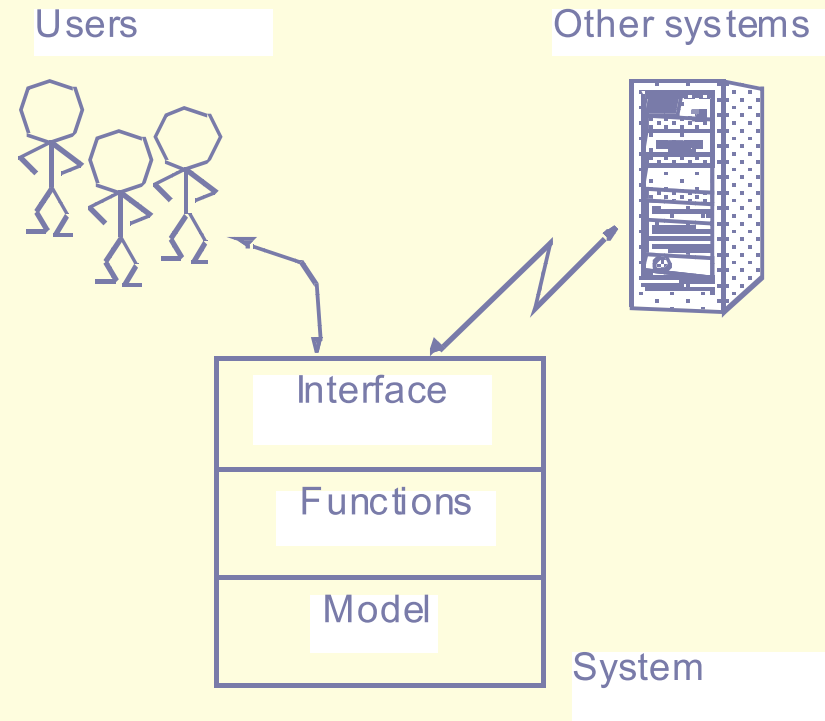


# Model the context

- Problem domain:  
That part of a context that is administrated, monitored, or controlled by a system
- Application domain:  
The organization that administrates, monitors, or controls a problem domain

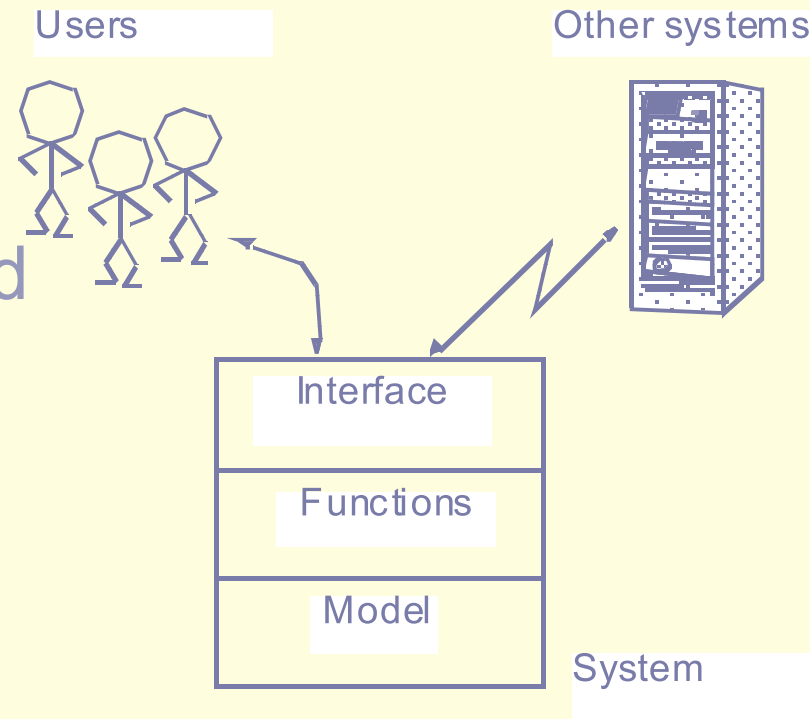


# Emphasize the architecture



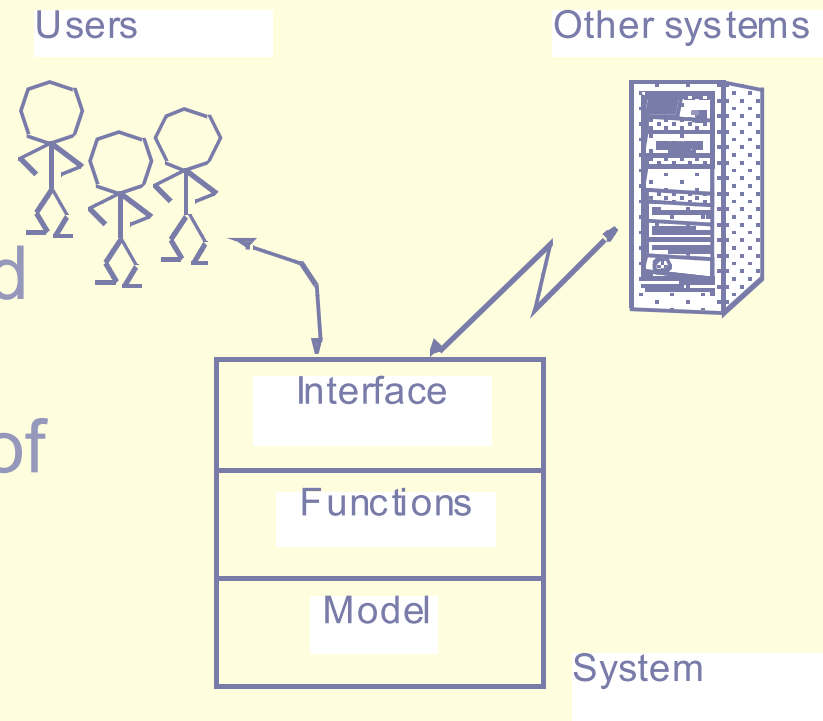
# Emphasize the architecture

- Architecture:  
A general structure  
that is later developed  
further



# Emphasize the architecture

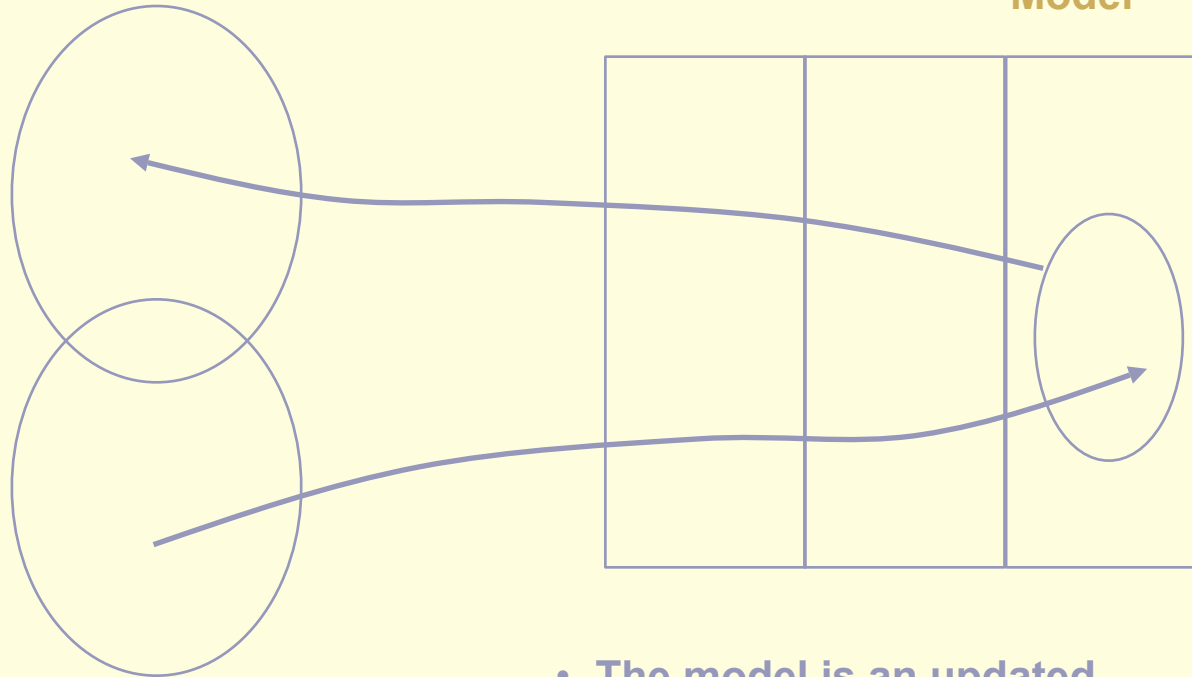
- Architecture: A general structure that is later developed further
- System: A collection of components that implements modeling requirements, functions, and interfaces.



# A model of the problem domain

Application domain

Model



Problem domain

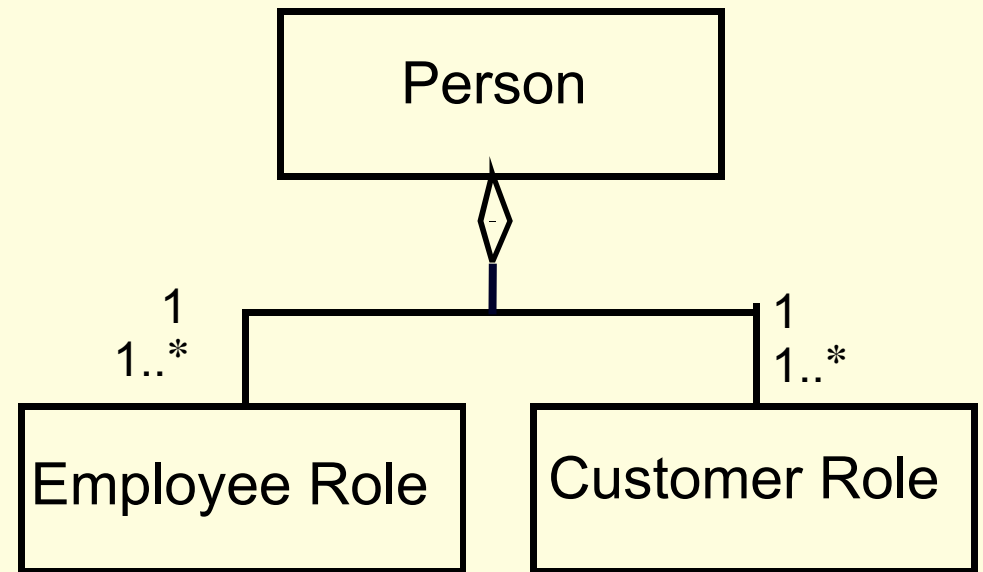
- The model is an updated representation of the state in the problem domain.
- The users get information about the problem domain mediated through the model.



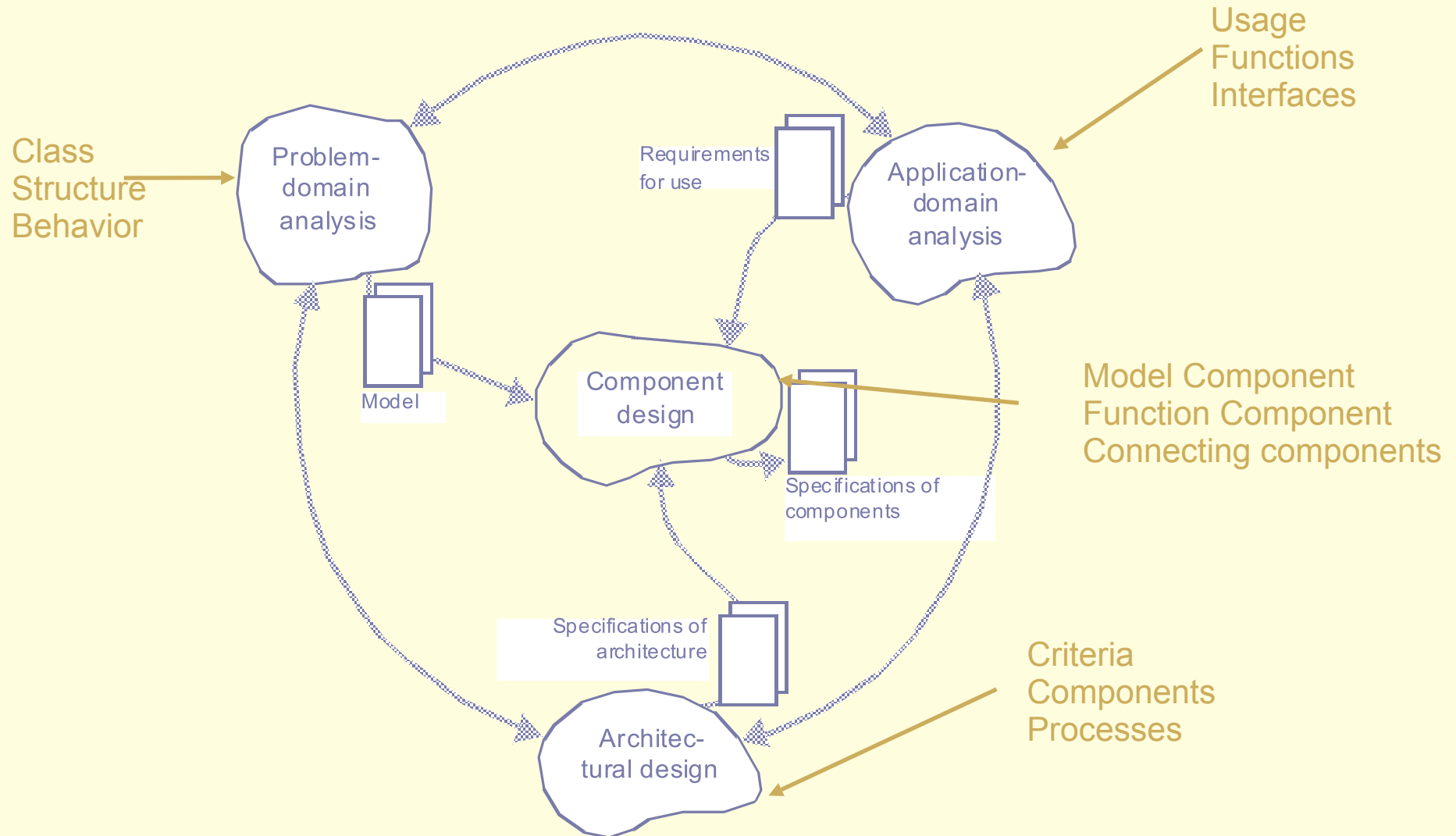
# Reuse patterns

## Example: Role

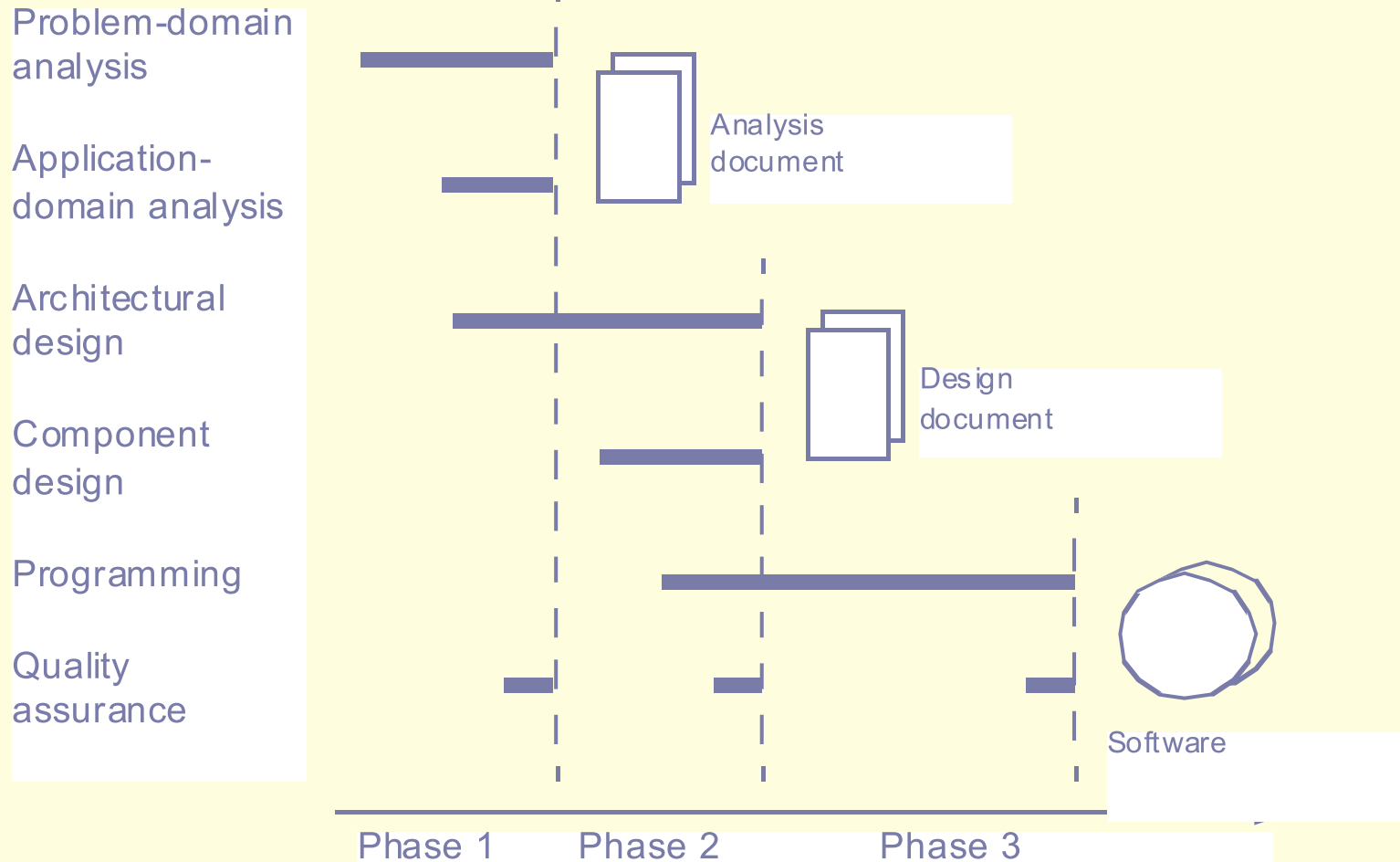
- Problem: A person has various roles shifting dynamically over time.
- Solution: To have an object aggregating a collection of objects each representing a role.



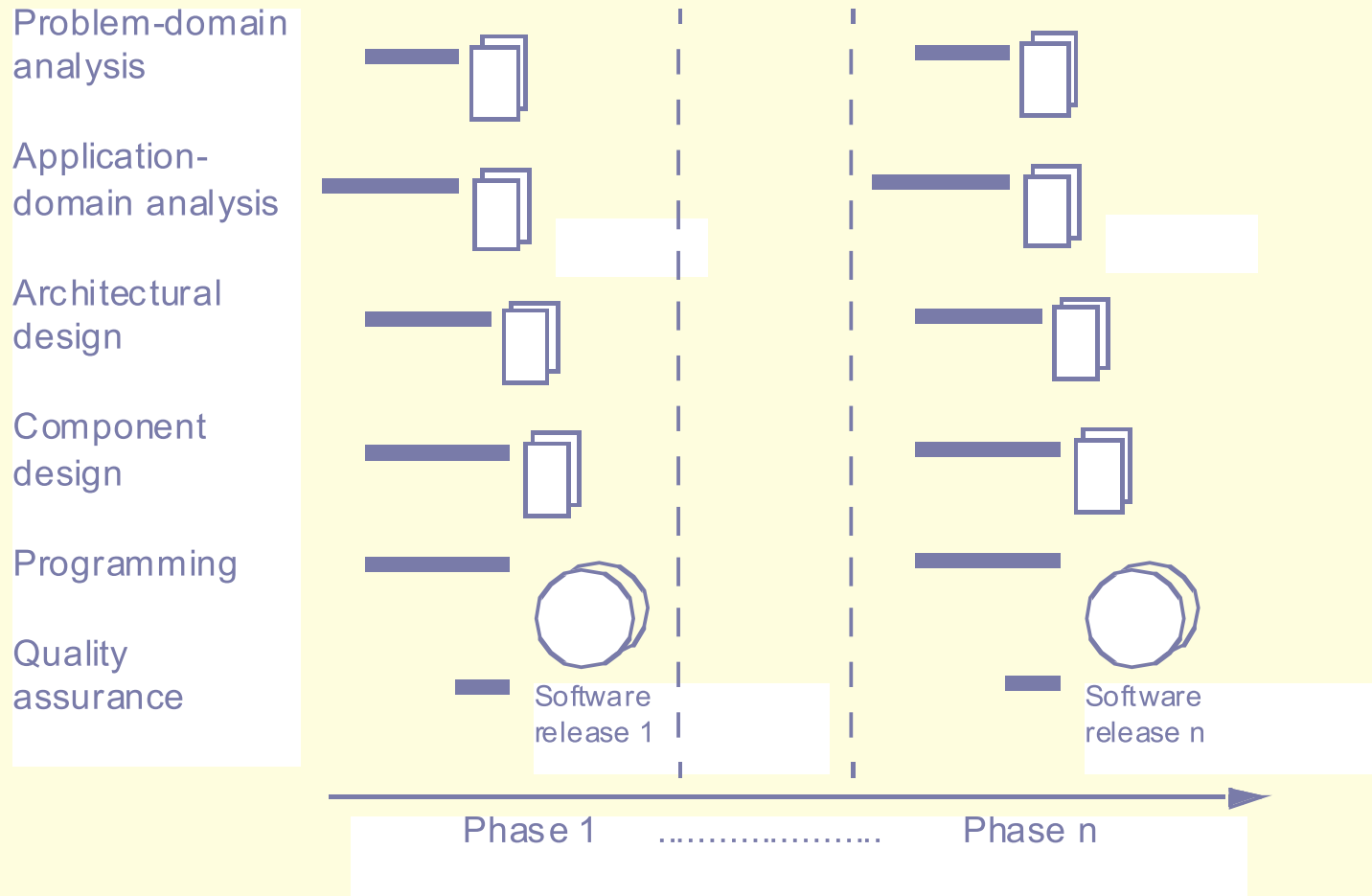
# Tailor the methodology



# Traditional, top-down approach



# Use-case driven, architecture-centric, and incremental approach



# The methodology 'OOA&D'

<b>Purpose</b>	<ul style="list-style-type: none"><li>• To determine system requirements.</li><li>• To produce a system design without significant uncertainties.</li><li>• To understand a system, its context, and the conditions for its implementation.</li></ul>
<b>Concepts</b>	<ul style="list-style-type: none"><li>• Object. An entity with identity, state, and behavior</li><li>• Class: A description of a collection of objects sharing structure, behavioral pattern, and attributes.</li><li>• Problem domain: That part of a context that is administrated, monitored, or controlled by a system.</li><li>• Application domain: The organization that administrates, monitors, or controls a problem domain.</li><li>• System: A collection of components that implements modeling requirement functions, and interfaces</li></ul>
<b>Principles</b>	<ul style="list-style-type: none"><li>• Model the context.</li><li>• Emphasize the architecture.</li><li>• Reuse patterns.</li><li>• Tailor the method to suit specific projects.</li></ul>
<b>Results</b>	<ul style="list-style-type: none"><li>• An analysis document and a design document.</li></ul>