# The Use-Case Construct in Object-Oriented Software Engineering

A Survey on a Paper from

Ivar Jacobson

# Overview

- Presentation of the Paper
  - Introduction to Use Cases
  - Use Case Modeling
  - Summary
- Critical Review
  - The Paper
  - The Use Case Construct today

# The Problem Domain

- It is important to have a shared Model for Users and Developers in the Requirements Analysis (see other lectures)
- You need a black box view of a System (which Developers can sell to customers)

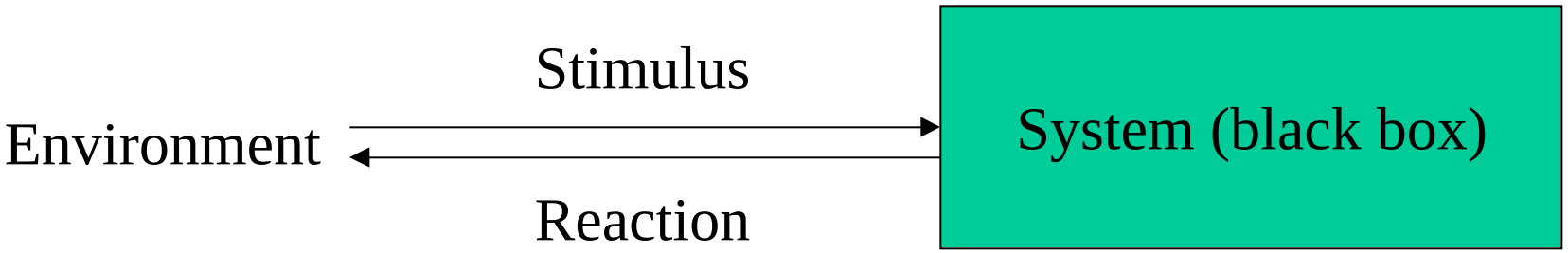
Use Case Construct

and the associated modeling techniques

# The Role of Use Cases (1)

Capture the **functional requirements** by describing the systems behavior:

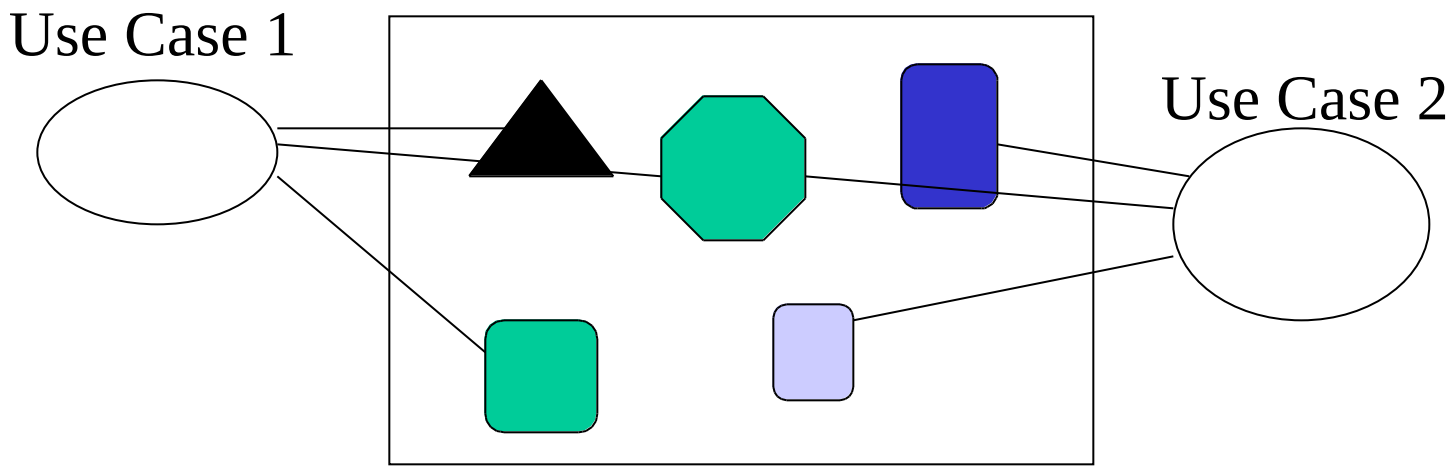
- It defines the systems environment
- It defines how the system reacts to input



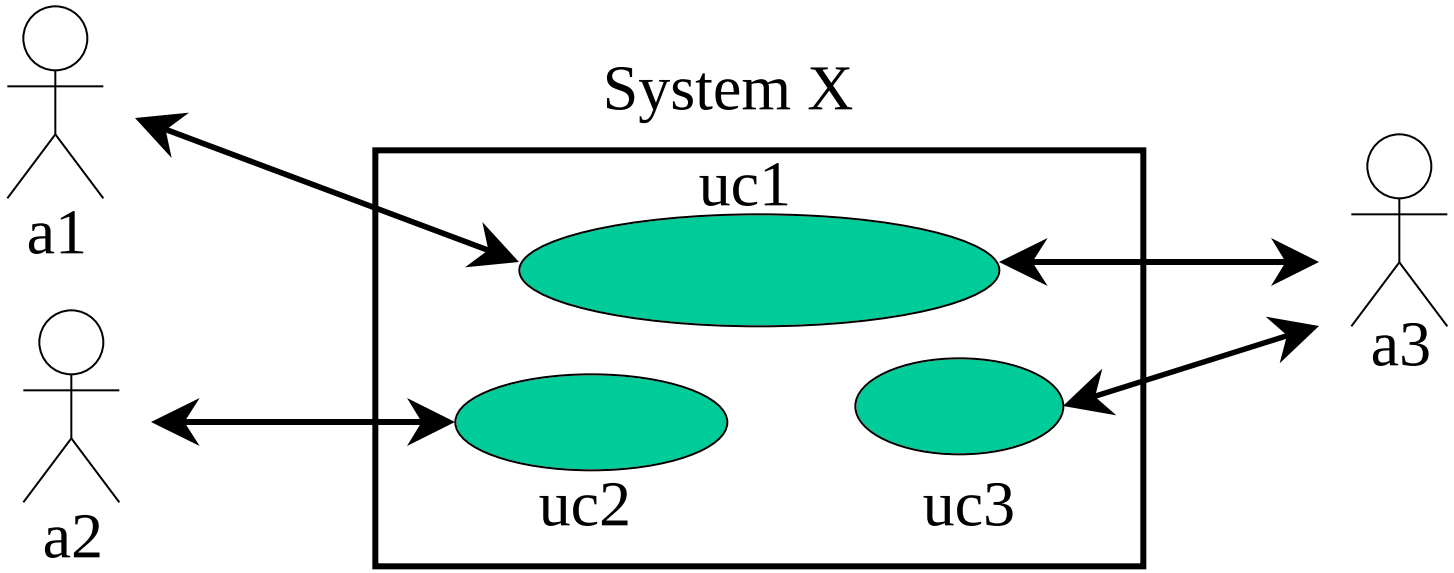
# The Role of Use Cases (2)

**Structure** object model into a managable view:

In Modelling a System with lots of Objects (100+) you can structure it after different Use Cases

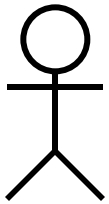


# Use-Case Modelling




- Use-Case Model is graph with two types of nodes
- Its name is the systems name

# Actors Pragmatics



- Actors are objects outside the system
- Only Actors can have an impact at the system
- Actors can be humans or other systems

# Difference Actor - User

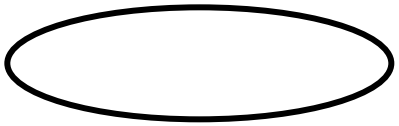
## User

- Is not a formal concept
- Is a human
- Can act as different instances of several different Actors

## Actor

- Is a formal concept
- Can be a human or a system
- Exists only if a user does something

# Use-Case Pragmatics (1)



- Is a Class, which can be instantiated
- When an Actor uses it, the system performs a use case
- All use cases are complete functionality of a system

# Use-Case Pragmatics (2)

What is a Good Use Case?

- Has a particular actor
- Has a measurable result of values

Where Do Use Cases Come From?

- Designers are observers
- Users are observed and asked

# Difference Use Case - Scenario

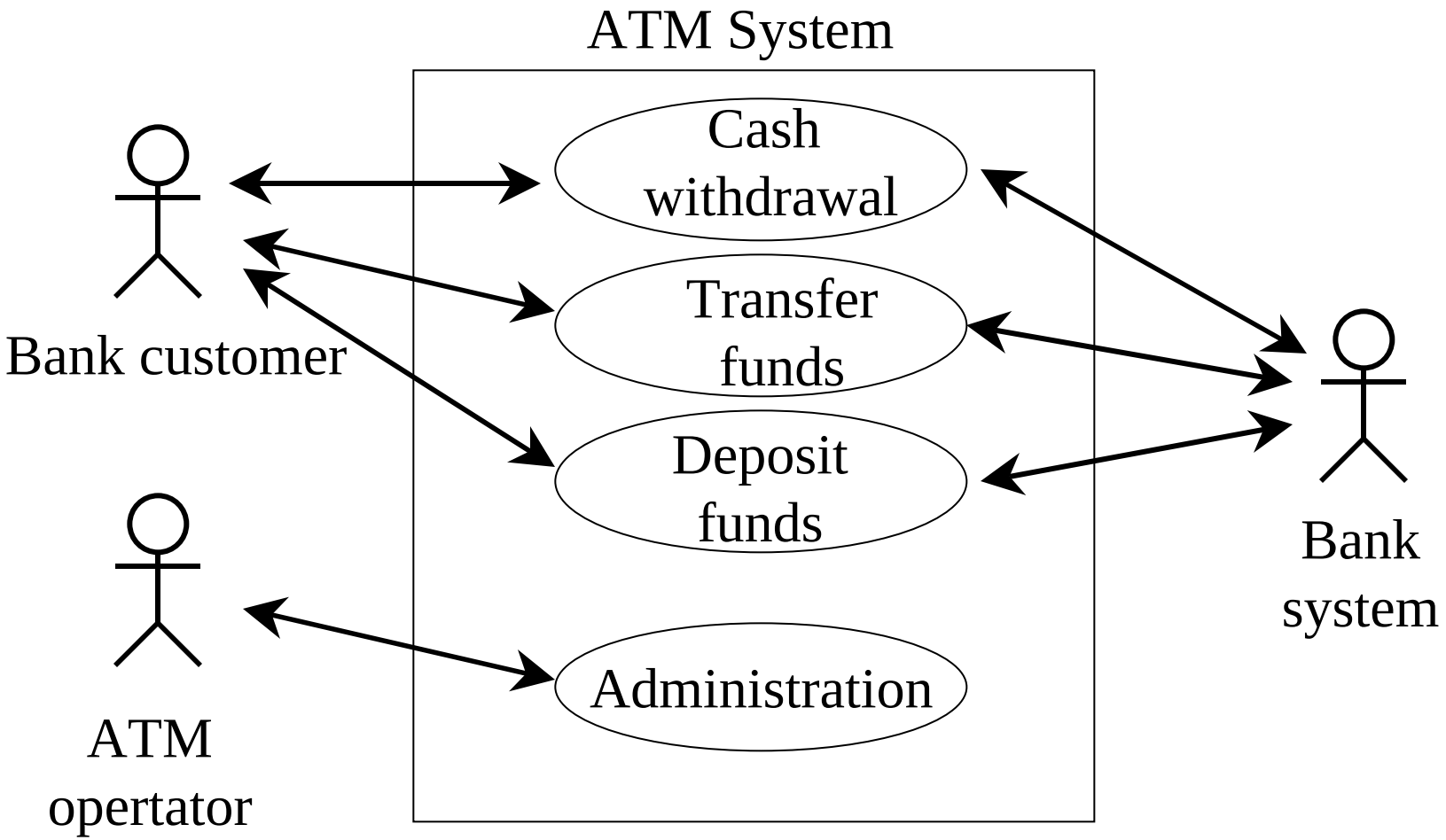
## Use Case

- Are described formally in a model of their own or in interactions between objects in different object models

## Scenarios

- Are Use Case instances
- Described as interactions between objects

# Example of a Use Case Model



# What cannot be modeled?

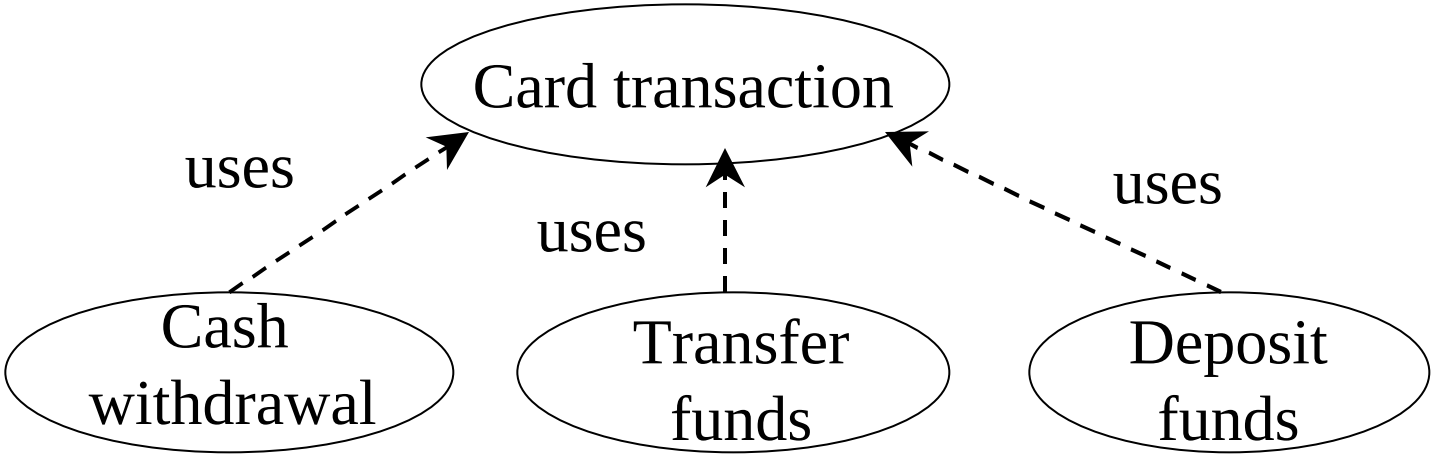
- Internal communication inside the system (no interfaces between use cases)
- Conflicts between use-case instances (to be modeled in a object model)
- Concurrency between use cases

# Associations between use cases

- Big systems may have hundreds of use cases
- There is a need for an intelligible use case model to avoid redudancy and get a layered description

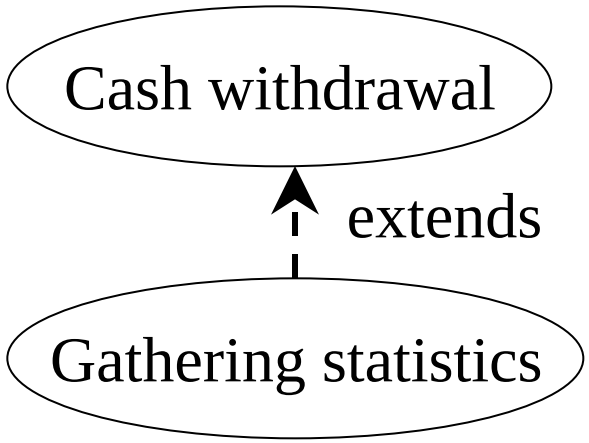
Association arcs: „uses“ and „extends“

# The Uses Association




- Uses association is like „inheritance“
- Only concrete use cases can be instantiated
- Uses association to more than one use case is possible

# The Extends Association




- Optional parts of use cases
- Complex and alternative courses
- Subsequences (execution in certain cases)

# The Uses versus the Extends Association

## Uses

- Is good for using abstract use cases and so get different layers
- Is bad for extensions because you need a use case for each possible use case

## Extends

- Is good because it offers flexibility
- Is bad if you have to much extensions because it gets unmanageable

# The Use-Case Model Versus Object Models

„Traceability“ between Different Models

For each use case make

- One Object Diagram to identify candidates for objects and their interdependencies
- One Interaction Diagam for describing stimuli and dynamic behavior in a instantiated use case

# Summary

Use-case modeling is:

- Basis for communication between different people (orderers, users, designer and testers)
- Important tool to develop an outside view of a system

# Overview

- Presentation of the Paper
  - Introduction to Use Cases
  - Use Case Modeling
  - Summary
- **Critical Review**
  - The Paper
  - The Use Case Construct in the UML

# Critical Review

- It is nearly impossible to describe the use-case construct in one chapter
- When the author wrote this paper it was and is until today an object for research
- You cannot see the use case contruct as something isolated

# The papers goal

To describe the use case construct and different modeling techniques associated with use cases during different activities in the development work.

# What it did

- It gives a first overlook and is good as a first reading about this theme
- Explains use case modeling in a very fresh and understandable way
- Explains what Actor, Use Case class and use case associaions are

# Where it lacks (1)

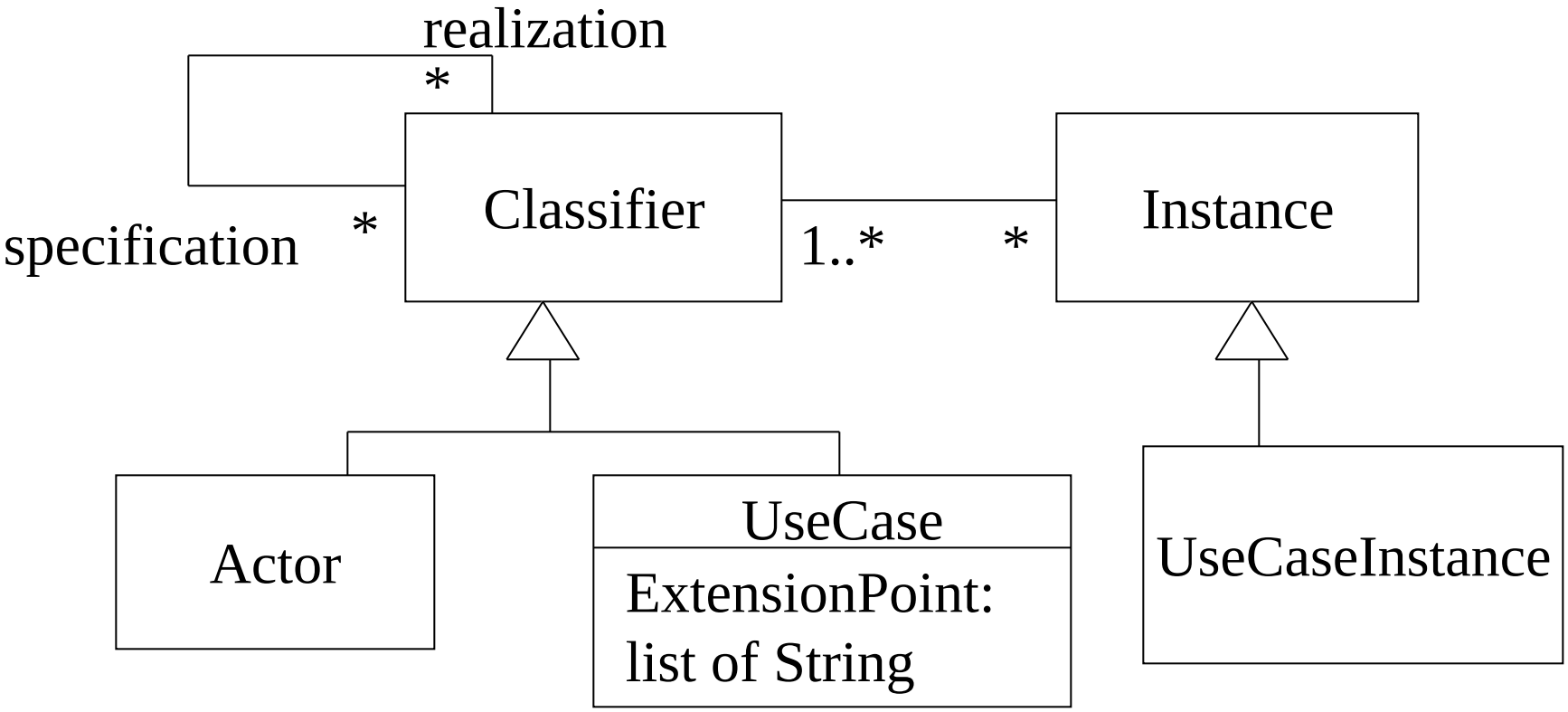
- The example is bad chosen
  - An ATM System is too complex and must lead to a feeling that things are missing in the model
  - The example is not complete nor absolutely clear (e.g. no security mechanism)
  - Too much about object and interaction diagram (he should have skipped this )

# Where it lacks (2)

- No direct feedback from users or developers
- It gives no real guide for modeling (methods seem to be unconcrete)
- He should have written more about problems and their solutions which a designer meets every day

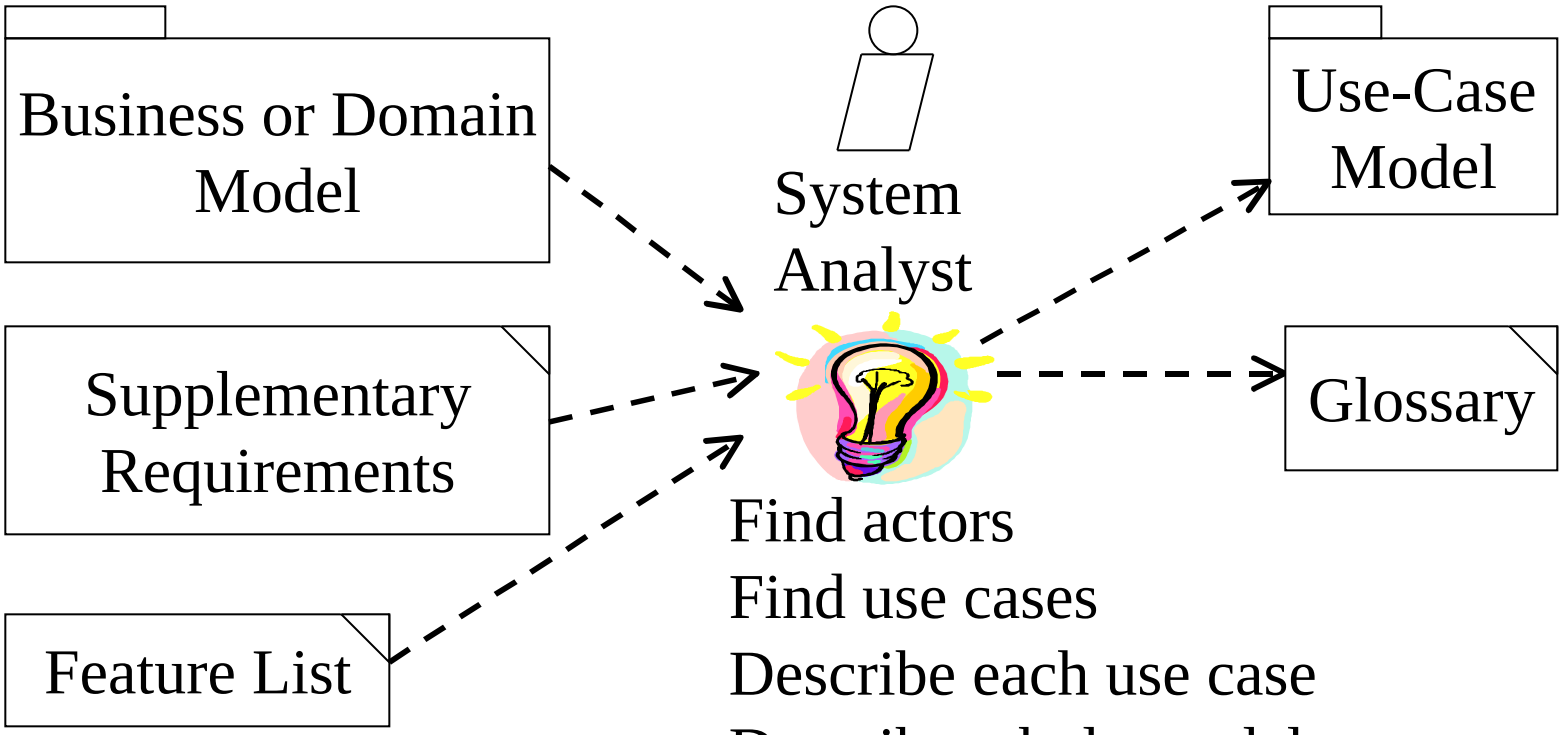
# State of the Art (1)

Use Case Construct defined in the UML



# State of the Art (2)

From Requirements to Use Cases
( from Unified Process)



# Summary

- With the UML it could probably get a standard for capturing requirements
- Interesting new applications are still object of research (like automatic testing)
- The software industry will be able to better fulfil the requirements of the society