Systems Development

Lecture 6: Functions

Contents

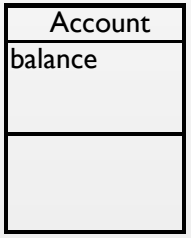
- Difficulties in exercises
- Summary of last lecture
- The Function activity
- Challenges in this activity
- Solution to Hand-in Assignment 1

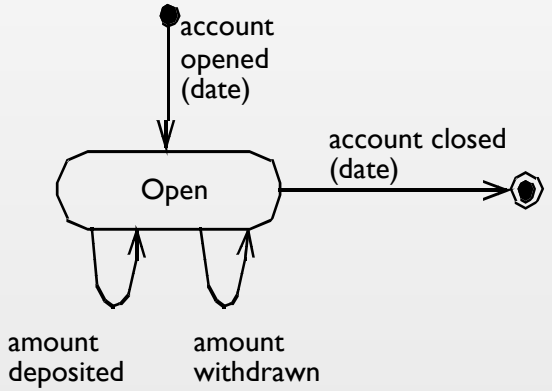
Contents

- Difficulties in exercises
  - Final state for an object
  - Iteration in statechart diagrams
  - Item-Descriptor pattern and the behaviour of the classes
  - Actor
- Summary of last lecture
- The Function activity
- Challenges in this activity
- Solution to Hand-in Assignment 1

Difficulties (1)

- What happens when an object gets to the final state? Does it die, disappear or what?
- In analysis:
  - The object can no longer perform or suffer events
  - The state of the object can still be read by a function
  - Example: a bank account object must still exist after we have closed it, so we can read it; e.g. to produce information to the customer and tax authorities at the end of the year





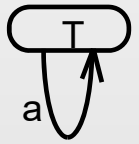
- In design:
  - Decide how long we want to keep objects who are in their final state
  - E.g. there are laws that define how long a bank account object that is in its final state (closed) must be kept

Difficulties (2)

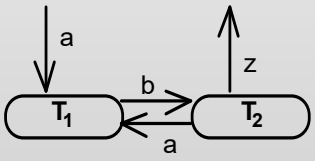
- Iteration in statechart diagrams
  - When should we put iteration in the event table and when is it not an iterative event?
  - How should we describe iteration in the statechart diagram? Should be through another state and back, or through a loop arrow?

	Customer	Assistant	Apprentice	Appointment	Plan
reserved	*	*		+	*
cancelled	*	*		+	
treated	*			+	
employed		+	+		
resigned		+	+		
graduated			+		
agreed		*	*		*

Iteration

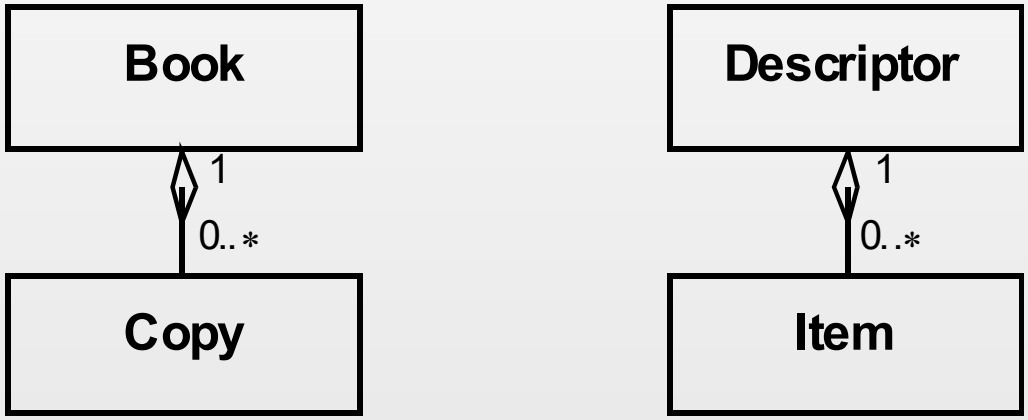


Also interation



Difficulties (3)

- Item-descriptor pattern - why and when should we use it; and how do we create behavioural patterns for the item and descriptor classes?



- Events:
  - Book (Descriptor): book included, book deleted
  - Common events: copy bought, copy destroyed
  - Copy (Item): copy borrowed, copy returned
- How will the behavioural patterns for Book and Copy be?

Difficulties (4)

- Examples: What is an actor (for a system) and what is not?
- General: An abstraction of a user or other system that interacts with the system we are developing
- Example:

	Actors			
Use Cases	Account owner	Creditor	Administrator	Liquidity Monitor
Payment	X	X		
Cash Withdrawal	X			
Money Transfer	X	X	X	
Account information	X		X	X
Credit information		X	X	
Registration			X	
Monitoring			X	
Fault processing			X	

Contents

- Difficulties in exercises
- Summary of last lecture
  - Application Domain analysis
  - The Usage activity
- The Function activity
- Challenges in this activity
- Solution to Hand-in Assignment 1

Application Domain Analysis: Results

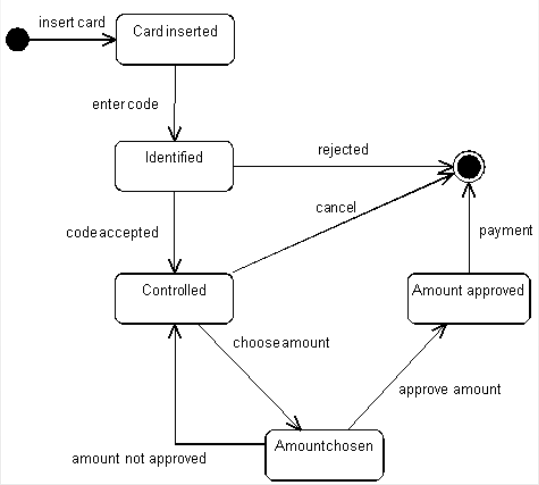
- Actors and Use cases

Account Owner

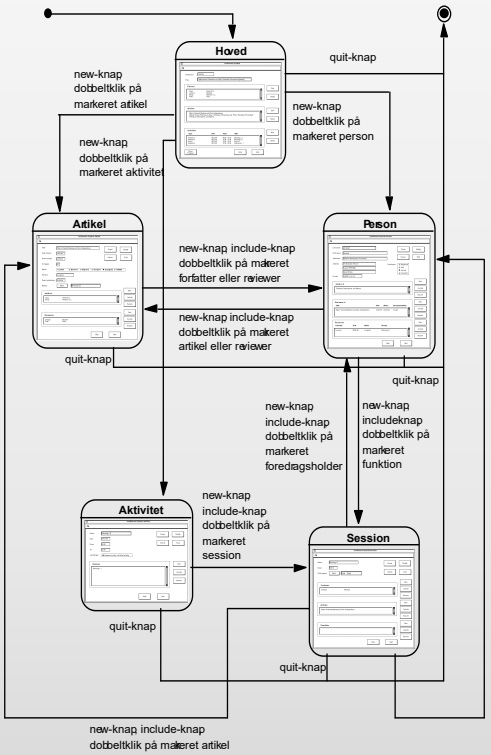
Purpose: A person who owns an account. The account owner's basic need is to be able to make payments with his plastic card.

Characteristic: The system's users include many and very different account owners

Examples: Account owner A is insecure in the use of a plastic card as a form of payment. A originally received his card because it was the only possibility…



- User interfaces



- Functions

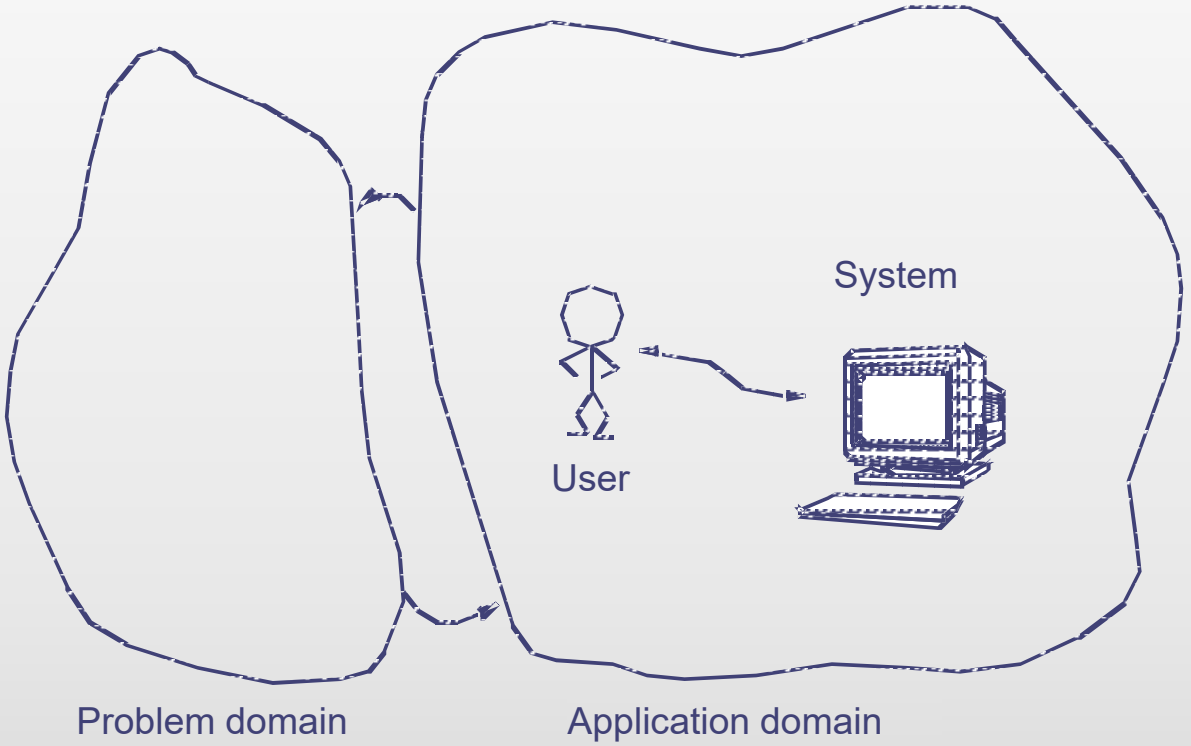
Lav plan	Særdeles kompleks	Opdatering
Konsekvensberegn plan	Kompleks	Signalering
Find kundeoplysninger	Middel	Aflæsning
Sæt indhold I plan	Kompleks	Opdatering
Slet plan	Simpel	Opdatering
Lav reservation	Middel	Opdatering
. . .	. . .	. . .

Application Domain Analysis: Key Concepts

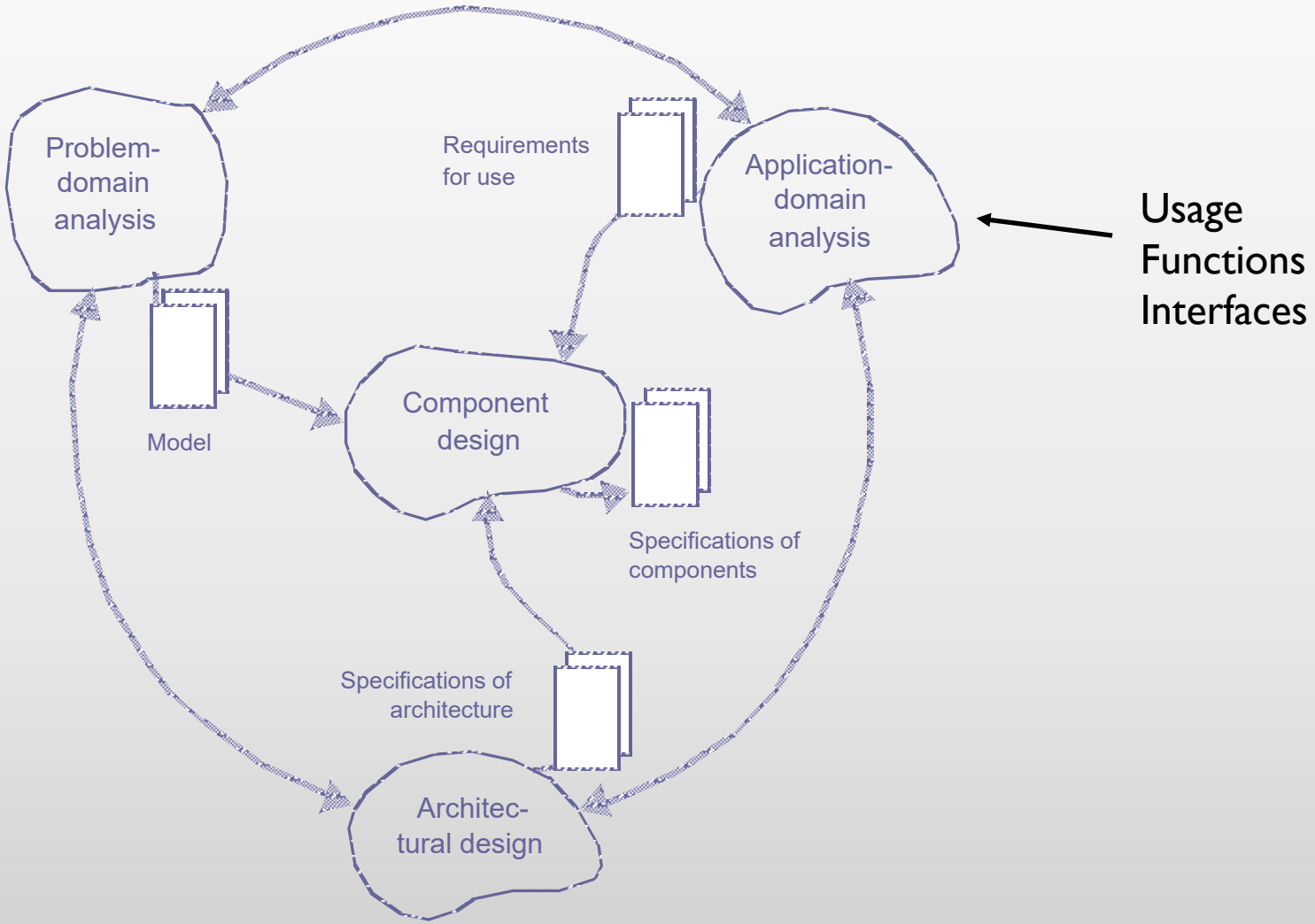
Application domain:
The organization that administrates, monitors, or controls a problem domain

New Concepts:

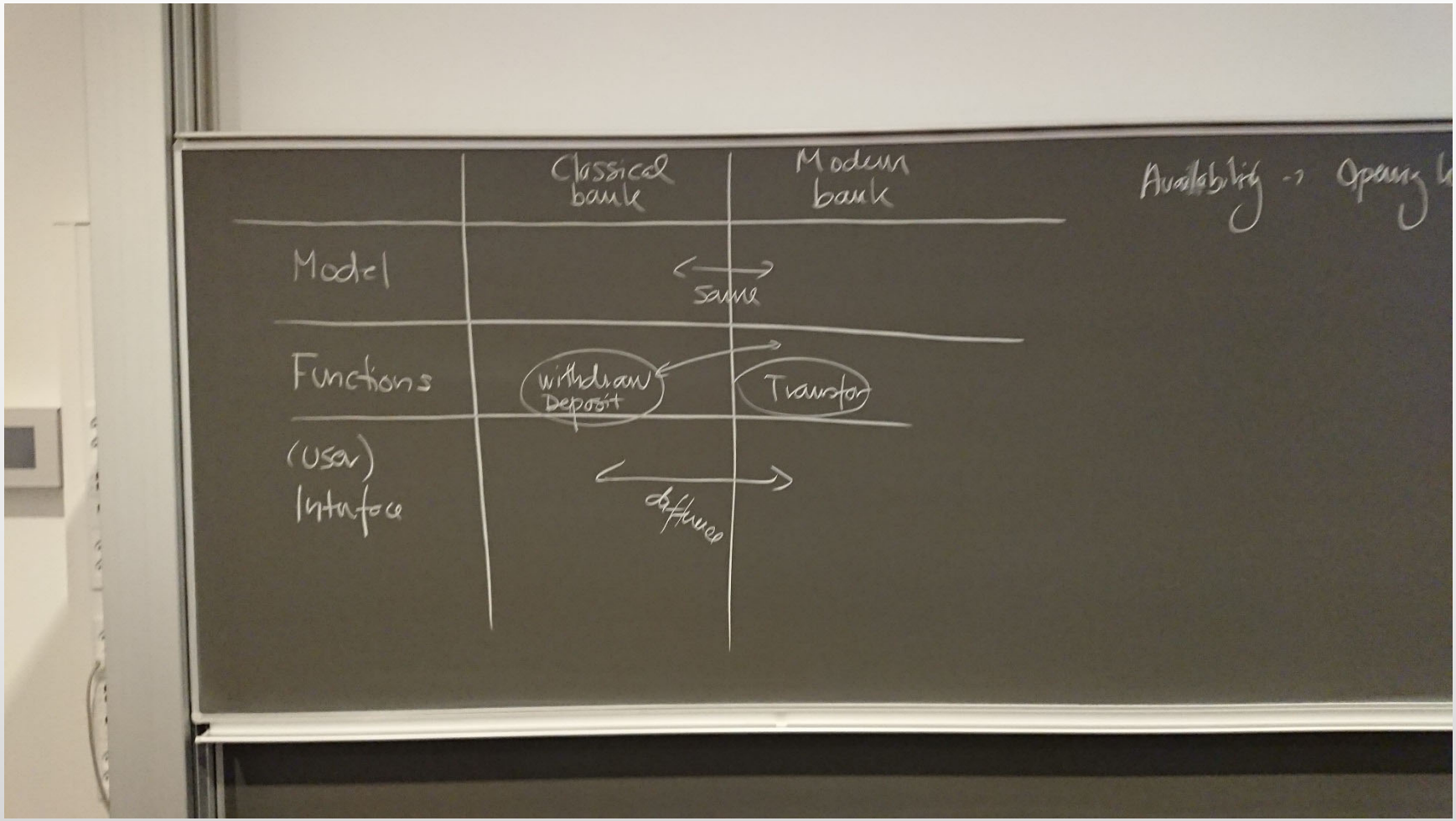
- Actors (users and other systems)
- Use cases
- Functions
- Interfaces



Application Domain Analysis: Activities



Stable and Transient Properties



Application Domain Analysis: Summary

Purpose	• To determine a system's usage requirements.
Concepts	• Application domain: An organization that administrates, monitors, or controls a problem domain. • Requirements: A system's externally observable behavior.
Principles	• Determine the application domain with use cases. • Collaborate with users.
Results	• A complete list of the system's overall usage requirements.

Usage: Results

Actor table

	Actors			
Use Cases	Account owner	Creditor	Administrator	Liquidity Monitor
Payment	X	X		
Cash Withdrawal	X			
Money Transfer	X	X	X	
Account information	X		X	X
Credit information		X	X	
Registration			X	
Monitoring			X	
Fault processing			X	

Actor

Account Owner

Purpose: A person who owns an account. The account owner's basic need is to be able to make payments with his plastic card.

Characteristic: The system's users include many and very different account owners

Examples: Account owner A is insecure in the use of a plastic card as a form of payment. A originally received his card because it was the only possibility…

Use case

Cash Withdrawal

Use Case: Cash withdrawal is started by the account owner, when he wishes to use his credit card to withdraw cash from an ATM. The account owner inserts his card in an ATM, and is then requested via the screen to type his PIN code. The screen will either show a polite denial, the card will be rejected from the ATM and the process will be cancelled; or the screen will show a menu requesting the account owner to choose an amount of money by typing on the ATM's keyboard. A new screen requests the account owner to approve the transaction. If the transaction is not approved the account owner is again requested to type an amount. Otherwise the use case ends by the ejection of the card, and the desired amount of money being paid.

Objects: (to be added later)

Functions: (to be added later)

Key Concepts: Actor

- Identify actors
  - Determine the distribution of roles of the works tasks related to the system
  - Consider human actors
  - Consider other systems as actors
- Describe actors
  - Make actor specification

Account Owner

Purpose: A person who owns an account. The account owner's basic need is to be able to make payments with his plastic card.

Characteristic: The system's users include many and very different account owners

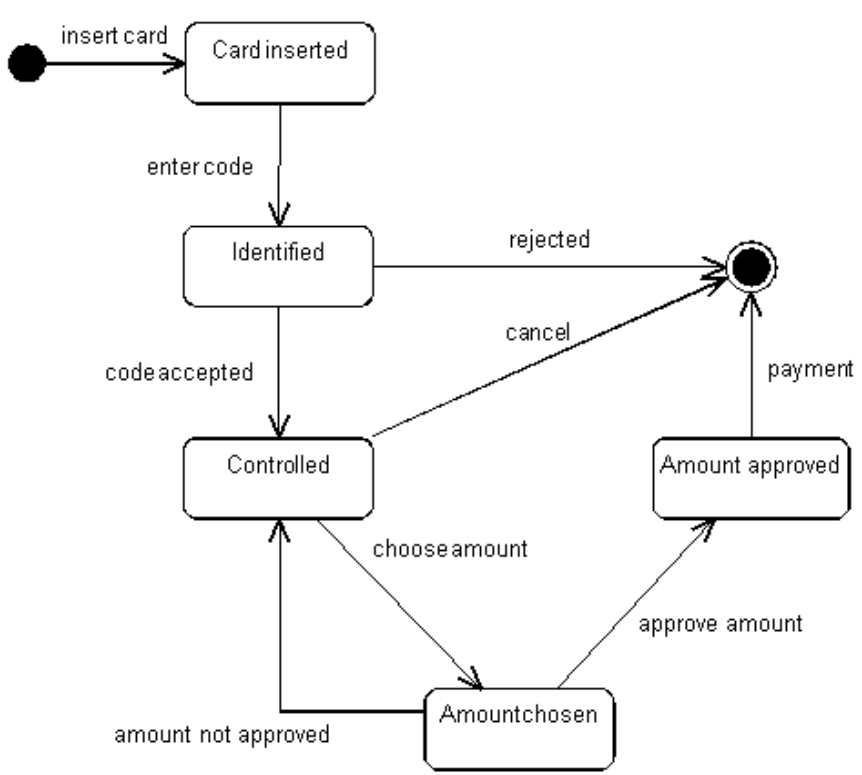
Examples:

Account owner A is insecure in the use of a plastic card as a form of payment. A originally received his card because it was the only possibility for getting an ID card for his checks. A only withdraws money from the ATM in emergency situations.

Account owner B is technologically curious and uses the system often, optimally, and to the limit of its abilities. B has never had major problems in understanding the possibilities of the system, and B also examines the possibilities that are not obviously available.

Key Concepts: Use Case

- Describe use cases
  - As text
  - As a statechart diagram



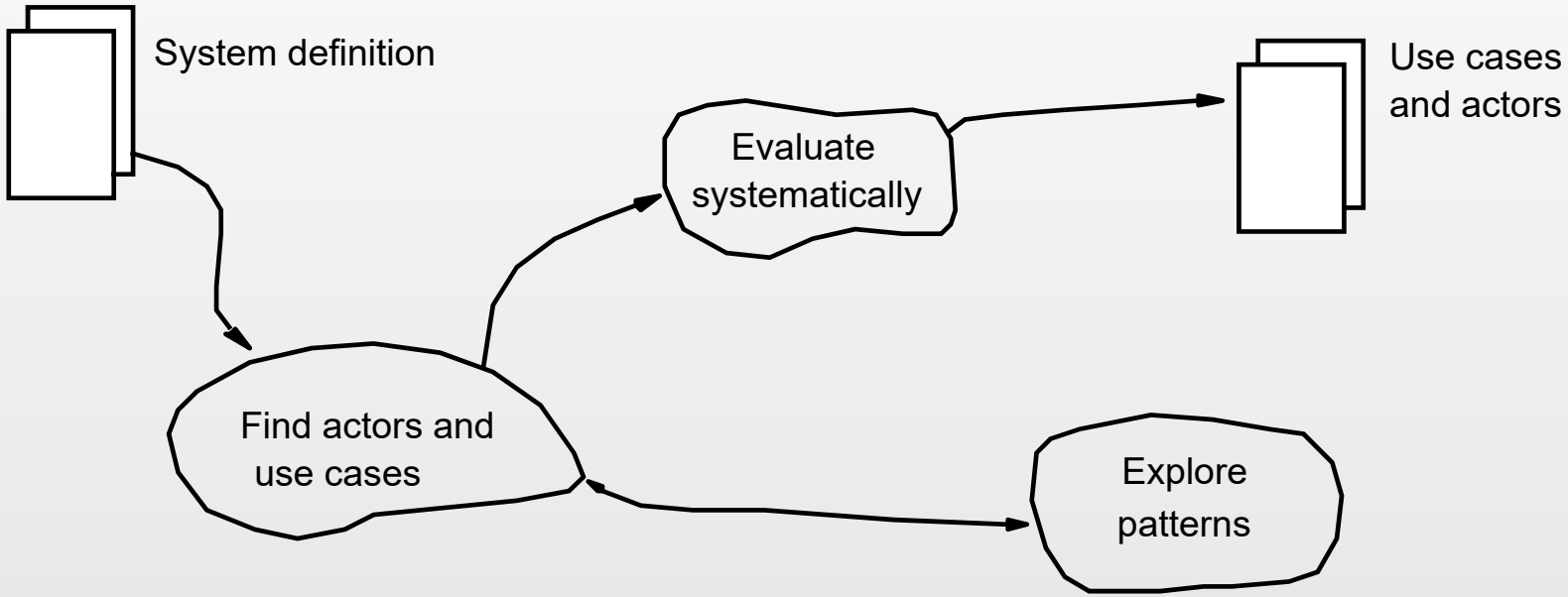
Cash Withdrawal

Use Case: **Cash withdrawal is started by the account owner, when he wishes to use his credit card to withdraw cash from an ATM. The account owner inserts his card in an ATM, and is then requested via the screen to type his PIN code. The screen will either show a polite denial, the card will be rejected from the ATM and the process will be cancelled; or the screen will show a menu requesting the account owner to choose an amount of money by typing on the ATM's keyboard. A new screen requests the account owner to approve the transaction. If the transaction is not approved the account owner is again requested to type an amount. Otherwise the use case ends by the ejection of the card, and the desired amount of money being paid.**

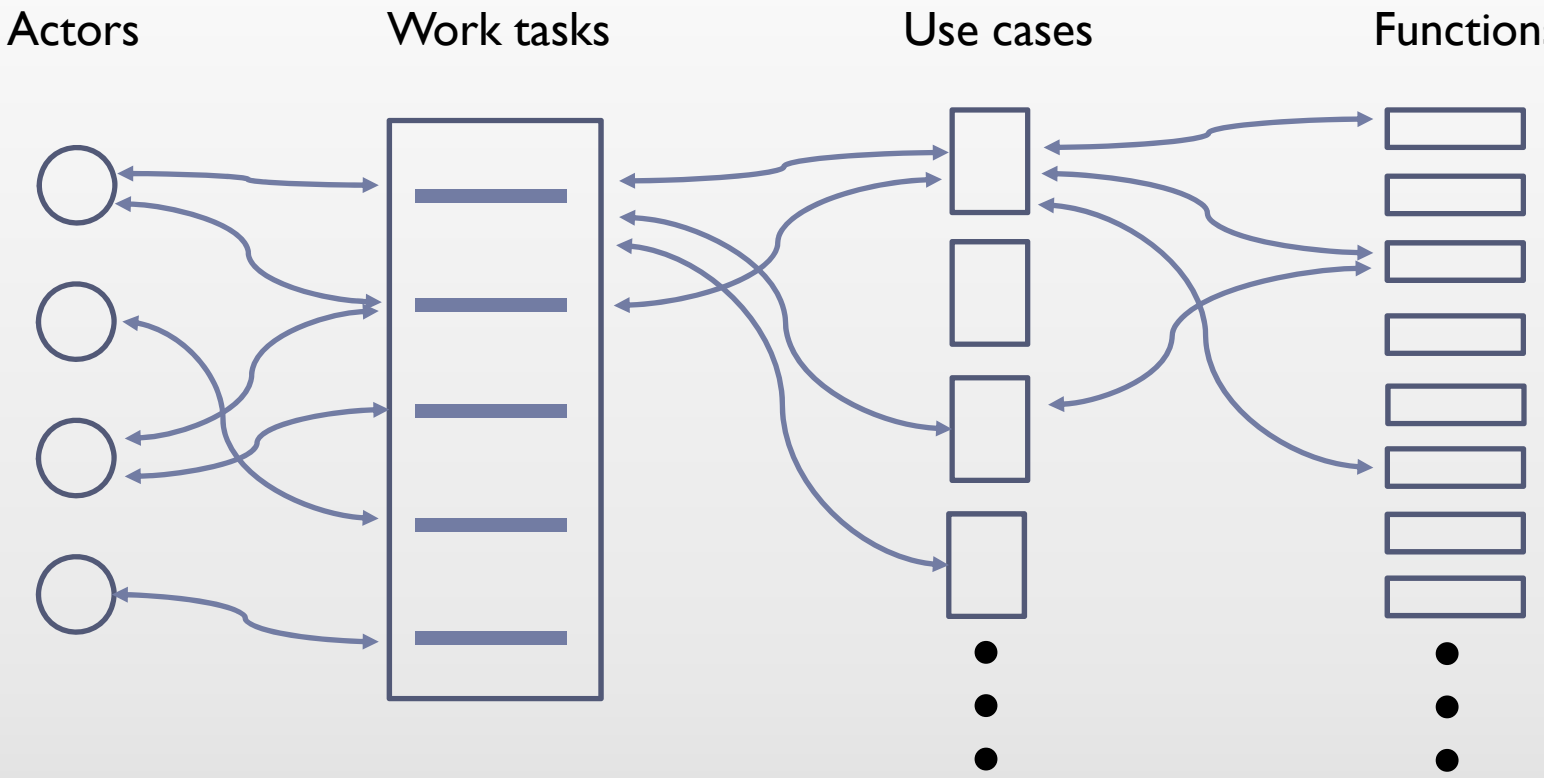
Objects: **(to be added later)**

Functions: **(to be added later)**

Usage: Activities



How Do We Get Started?



1. Describe work tasks
2. Define actors for work tasks
3. Identify use cases as elements of work tasks

Patterns

- Procedural
- Material

Usage: Summary

Purpose	• To determine how actors interact with a system.
Concepts	• Actor: An abstraction of users or other systems that interact with the target system. • Use case: A pattern for interaction between the system and actors in the application domain.
Principles	• Determine the application domain with use cases. • Evaluate use cases in collaboration with users. • Assess social changes in the application domain.
Results	• Descriptions of all use cases and actors.

Appreciate the Difference: Actors and Classes

	Application domain	Problem domain
Static	Actors	Classes and Structure
Dynamic	Use cases	Behavioral patterns

Quiz 5

Quiz 5

Average

3.96 (of 6.00) of 77 finished attempts (of 159)

Best result (0.67-1.00)

2 (0.84) Which of the following are actors in the system?
1 (0.67) Problem domain and application domain objects

Middle result (0.34-0.66)

5 (0.65) You can evaluate your actor and use case candidates by:
6 (0.63) How many objects are there
4 (0.61) The purpose of the Procedural Pattern is to
3 (0.57) A use case ...

Worst result (0.00-0.33)

None

Quiz: 5 Answers

Question 1

(Exam, January 2014, question 3b)

Consider this system definition

An IT system to keep track of bicycles, their owners, and reports of stolen bicycles. The system shall increase the number of solved cases of bike theft by making information about bicycles and their insurance status, owners, and stolen bicycle reports available to the general public, insurance companies, the police, and municipal workers responsible for cleaning streets and bicycle parking lots.

The system is based on a centralized register with information about bicycles etc., and computers and mobile devices (tablets, smartphones) to register and search for information. All bicycles in the system will have barcode labels to supplement the mandatory frame number. The system must protect information about bicycle owners from unauthorized access.

Which of the following belong in the Problem Domain (PD), the Application Domain (AD) (or both) for the system

Citizen	AD
Bicycle	PD
Insurance company	Both PD and AD
Bicycle owner	Both PD and AD
Bicycle thief	Neither PD nor AD
Smartphone	AD
Police officer	AD
Bicycle shop	Neither PD nor AD
Label with barcode	PD

Question 3

A use case ...

Select one or more:

- ☐ a. Describes the actors using the system
- ☐ b. Is a behavioral pattern
- ☑ c. Describes a user's interaction with the system
- ☑ d. Describes the interaction between the system and another system
- ☐ e. Is an event trace
- ☐ f. Is a special type of function

Quiz 5: Answers

- Question 4

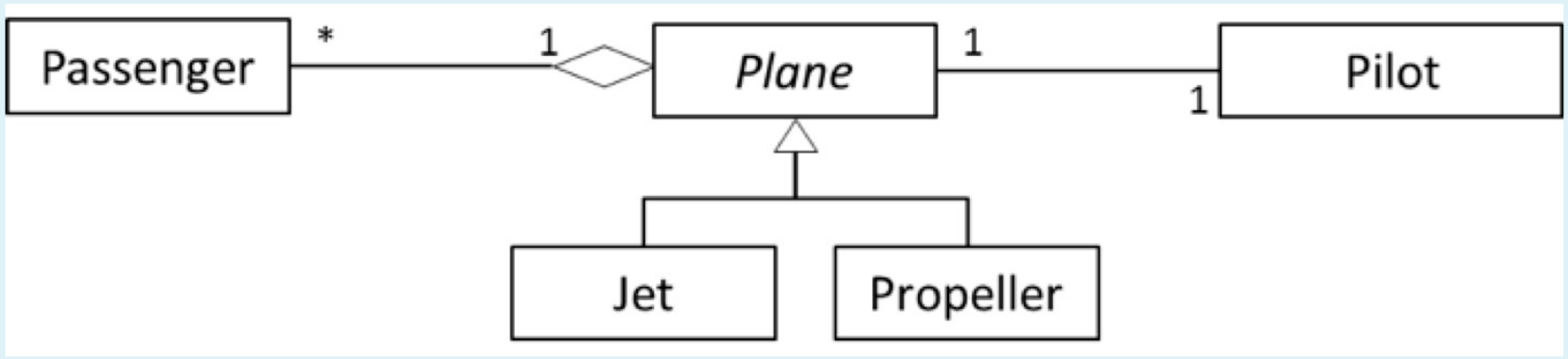
The purpose of the Procedural Pattern is to

Select one or more:

- ☐ a. Describe structure
- ☑ b. Describe usage
- ☐ c. Ensure that every action can be done almost in any order
- ☑ d. Ensure business rules are observed
- ☐ e. Describe behavior

Quiz 5: Answers

- Question 6



There is one object of the class pilot.

Select one or more:

- ☐ a. There is one object of the class 'Jet'
- ☑ b. There is either one object of the class 'Jet' or one of the class 'Propeller'
- ☑ c. All passengers are indirectly connected to a pilot
- ☐ d. There is one object of the class 'Passenger'
- ☐ e. There is one object of the class 'Plane'
- ☑ f. There may be several objects of the class 'Passenger'
- ☐ g. There are two objects of the class 'Plane'
- ☐ h. There is one object of the class 'Propeller'

Contents

- Difficulties in exercises
- Summary of last lecture
- The Function activity
  - Results
  - Key concept
  - Activities
- Challenges in this activity
- Solution to Hand-in Assignment 1

Functions: Results

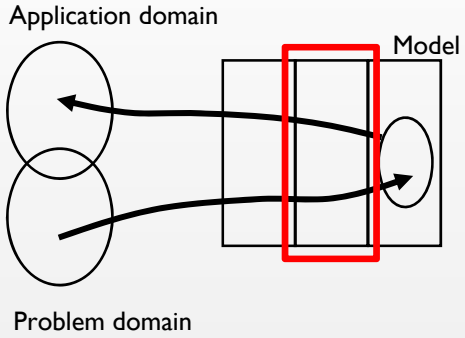
- Primary: a complete list of functions

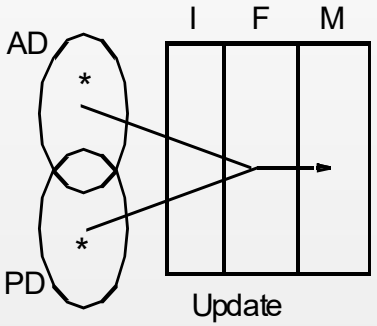
Planning		
Make schedule	Very complex	Update
Calculate schedule consequences	Complex	Signal
Find working hours from previous period	Medium	Read
Enter contents into schedule	Complex	Update
Erase schedule	Simple	Update
Query earlier schedules	Medium	Read
Make appointment	Medium	Update
Cancellation	Simple	Update
Query possible appointments	Complex	Read
Register treatment	Simple	Update
Create customer	Simple	Update
Query customer information	Medium	Read
Employment	Simple	Update
Retirement	Simple	Update
Update apprentice information	Simple	Update

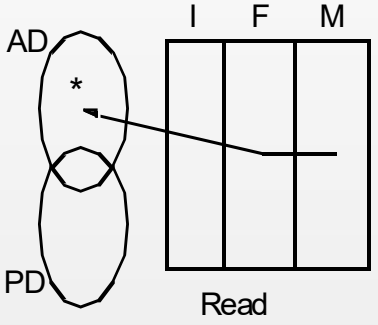
- Secondary: specifications of complex functions

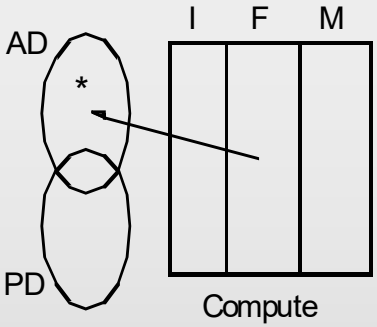
Query possible reservations:
given time or date or employee-name
search objects in time period-available and select those
who belong to employee-name, if known
have date, if known
cover point in time, if known
result objects of time period-available that fulfill the criteria

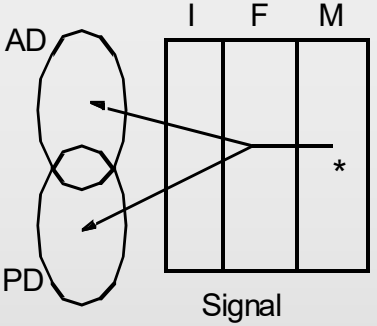
Key Concepts: Functions and Function Types





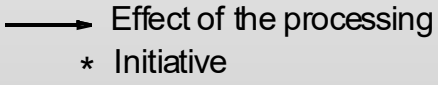




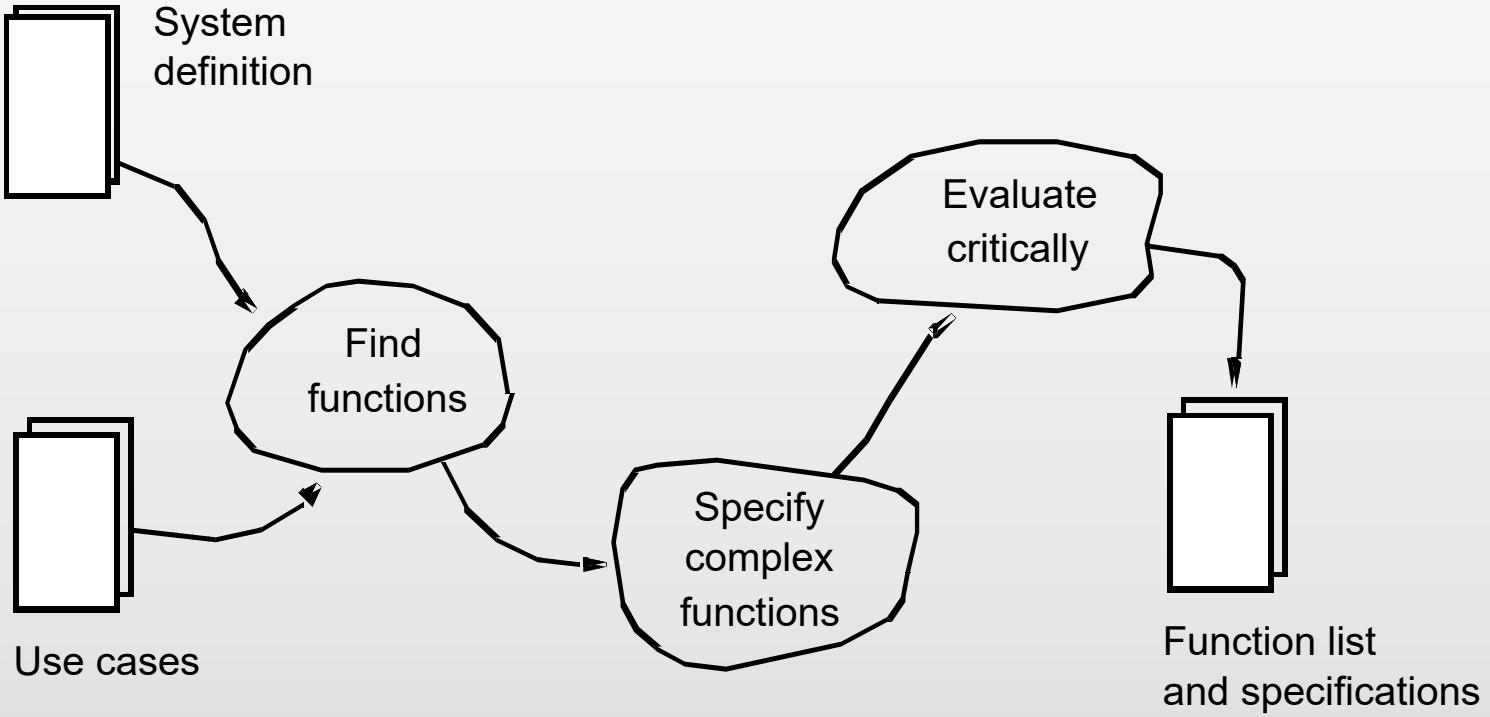


Function:
A facility for making a model useful for actors:

- A ressource for actors
- Uses the model component in order to support use cases
- Give an example of each function type

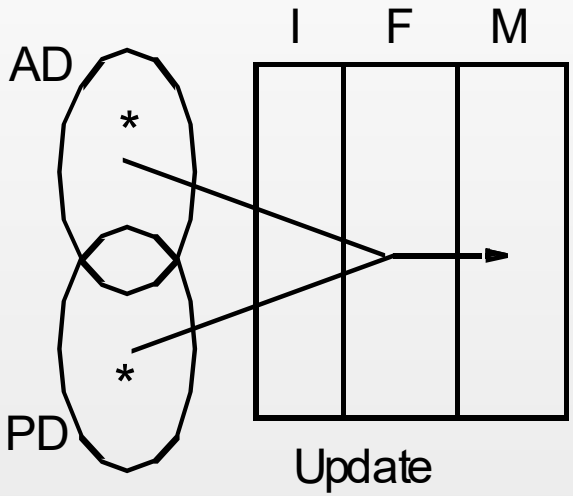


Functions: Activities



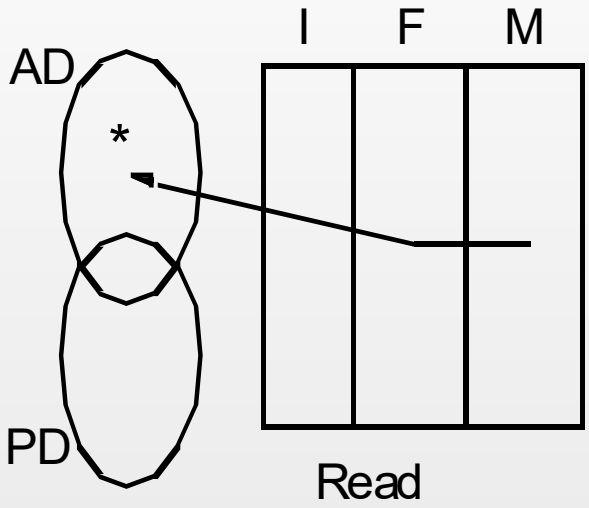
Find functions: Update

- Updating is related to events in the problem domain
- For each event ask:
  - How is the event observed, and how is it registered? In which use cases does this happen?
  - How should the use cases be supported by update functions?
  - Which objects, attributes, and object structures are affected by the event, and what requirements does this impose on the update functions?



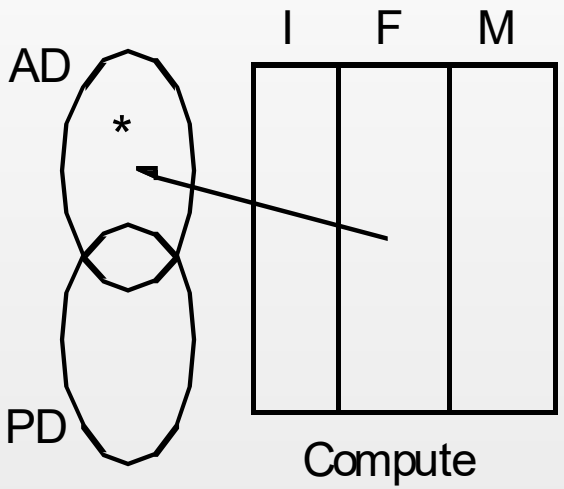
Find functions: Read

- Reading reflects a need for information in the application domain about the problem domain
- Actor/Use case perspective
  - Given the work of the actors, what do the actors need to know about the state of the model?
  - Which read functions do actors give rise to?
  - Which read functions do we need in each use case?
- Model perspective
  - Given the model, which objects and structures will the actors need information about?
  - What read functions does this give rise to?



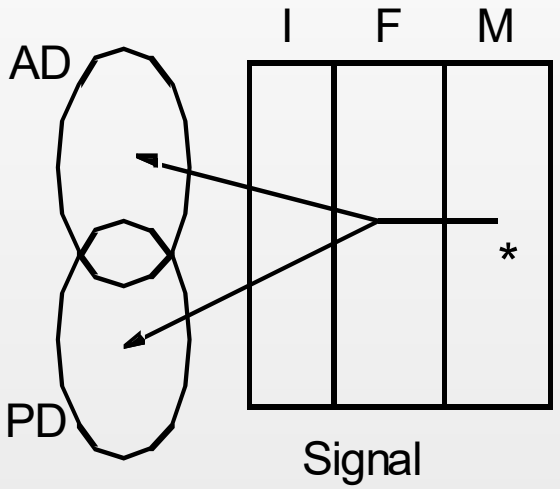
Find functions: Compute

- Computation is used to generate further information
- Take your point of departure in actors and use cases:
  - Which computations (not necessarily based on the model) do the actors need to have carried out?
  - Does the computational basis come from the actors, the model, or both?
  - Which computations form complete wholes in the use cases?



Find functions: Signal

- Signaling is related to critical states in the model
- Examine the model of the problem domain carefully:
  - What are the critical states for the model?
  - What is the significance of these critical states? What are the consequences when they occur?
  - How does a signal function register that the model has entered a critical state?
  - What signals does each critical state give rise to? How reliable and strong do the signals have to be?



Assess the Complexity of Each Function

- Identify functions where there is uncertainty = complex
- Complex functions need description in more detail
- Simple and medium functions don't need more description
- This saves a considerable amount of time
- Assessing complexity:
  - Simple: sets or reads the value of an attribute in an existing object
  - Medium: creates a new object and connects it by object structure(s) to other objects
  - Complex: reads or creates several objects (from different classes)
- Give an example of each

Specify Complex Functions

- Mathematical expression: u = f(i)

- Algorithm

```
Query possible reservations:
        given time or date or employee-name
        search objects in time period-available and select those
            who     belong to employee-name, if known
                    have date, if known
                    cover point in time, if known
        result objects of time period-available that fulfill the criteria
```

- Decomposition

Planning		
Make schedule	Very complex	Update
• Create six-week period	• Simple	
• Create standard distribution for an employee	• Medium	
• Use standard distribution for an employee	• Simple	
• Adjust the distribution of employees in a week	• Medium	

Evaluate Systematically

- Completeness:
  - Let the users review the list of functions
  - Use the questions for each function type to exhaust that category
  - Compare with the system definition, the model, and the use cases
- Make experiments and prototypes
  - To check the use cases
  - To check the set of function

Functions: Summary

Purpose	• To determine the system's information processing capabilities.
Concept	• Function: A facility for making a model useful for actors.
Principles	• Identify all functions • Specify only complex functions • Check consistency with uses cases and the model
Result	• A complete list of functions with specification of the complex functions.

Contents

- Difficulties in exercises
- Summary of last lecture
- The Function activity
- Challenges in this activity
- Solution to Hand-in Assignment 1

Appreciate the Difference: Event – Use Case – Function

- A source of confusion
- All three describe dynamic aspects
- They are connected

but …

- They belong to separate domains
- They are all needed because they emphasize different parts of the requirements to the system
- but keep them separate

- Example: order processing system
  - Event
    Ordered – a customer has entered into a legally binding agreement at a point in time
  - Use case
    Enter order – a user in the application domain applies the system to make an order for a customer
  - Function
    Create order – in the system's model, an object of the class Order is created

Work in Exercises for this Activity

- Find functions by means of function types
  - Do it systematically function type by function type
  - For each type ask the questions in the slides
- Specify one complex function
- Make the function list

Contents

- Difficulties in exercises
- Summary of last lecture
- The Function activity
- Challenges in this activity
- Solution to Hand-in Assignment 1