ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

SYSTEMS ANALYSIS AND DESIGN

Lecturers: Nguyen Thanh Binh – Nguyen Quang Vu – Le Viet Truong – Le Thi Bich Tra – Vo Van Luong – Nguyen Thi Hanh

Faculty of Computer Science

Vietnam - Korea University of Information and Communication Technology (VKU)

http://vku.udn.vn/

Chapter 2.
An overview of UML

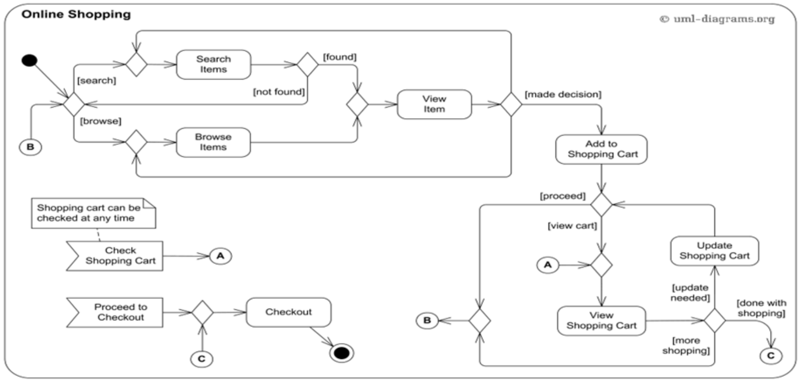
- Modelling
- Object-oriented modelling techniques
- History of UML
- Brief introduction to UML
  - Notions
  - Diagrams
  - Views

Model and Modelling

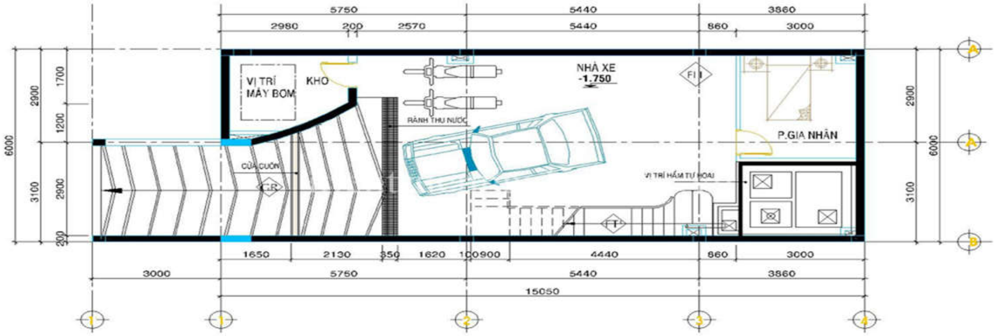
- A **model** is a simplification of reality. We build models so we can better understand the system we are developing.

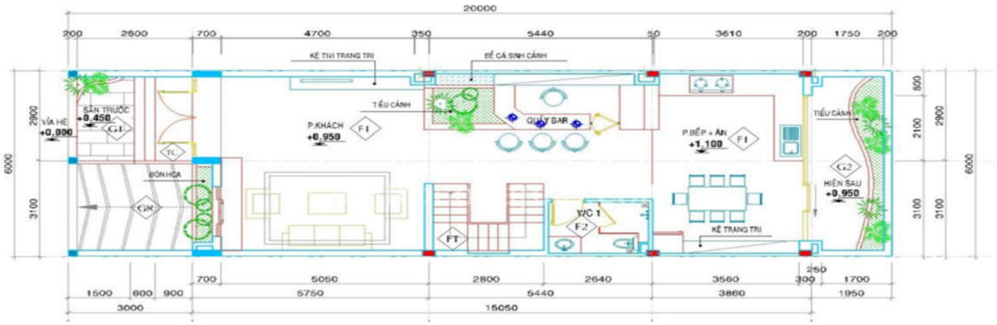
- **Modelling** is the process of building models to represent a system

- Modelling
  - helps us to visualize a system as it is or as we want it to be
  - allows us to specify the structure or behavior of a system
  - gives us a template that guides us in constructing a system
  - documents the decision we have made

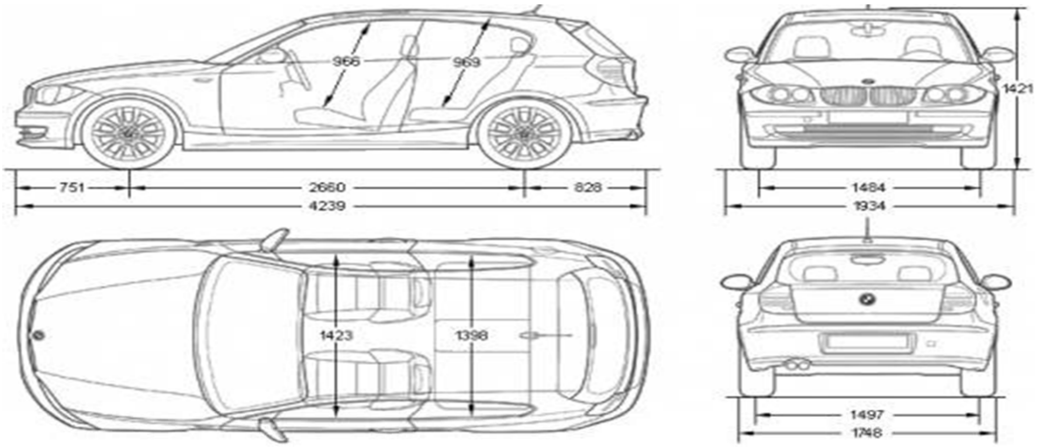


Model and Modelling: Example





Model and Modelling: Example



VDA1

Modelling

- A model is a simple representation of a part of the real system with a specific purpose
  - The actual system is complex, then it is necessary to simplify
  - Master the complexity of the system
- A model represents the system
  - at certain abstraction level,
  - according to a viewpoint,
  - by means of description (e.g., text, image, …)

Slide 6

VDA1 TODO: rework (inspired by “model & modelling” section of ABMS course

http://www.maxpellizzaro.com/tutorials/uml/modeling.pdf

VO Duc An, 26/05/2015

Modelling: why?

- Better understand the system
  - Facilitate the master the problem
- Ease the communication
  - Supply means of communication between developers
- Better complete the system
  - Ease the recognition of consistency between models and the needs to improve and complete the system
- Specify the structure and the behavior of the system
- Document the important decisions

Modelling

- Meta-model
  - is a representation of a model
  - can be used to
    - describe the syntax and the semantic of a model
    - manipulate models with tools
    - transform models
    - verify and maintain the coherence between models

Principles of modelling

- The choice of the appropriate model
  - Data view: entity-association model
  - Structural view: algorithm
  - Object-oriented view: classes and relations between them
- The models must represent the system at different levels of abstraction (according to the needs of the users)
- Models must be connected to the real world
  - Constructed models are close to real systems
  - Object approach > Procedural approach
- A system must be modelled by a set of models
  - A model is not sufficient
  - Describe different views of the systems: dynamic, static, installation, use, …

Modelling

- A **good model** should
  - use a standardized notation
  - be understandable for customers and users
  - allow software developers to understand the system
  - provide an abstract view of the software
  - be visual

Benefits of modelling

- Ease the revision and the evolution of the system
- Reduce errors by allowing to detect errors early in the stages of development
- Reduce development cost
- Reduce time-to-market
- Reduce complexity by mechanism of abstraction

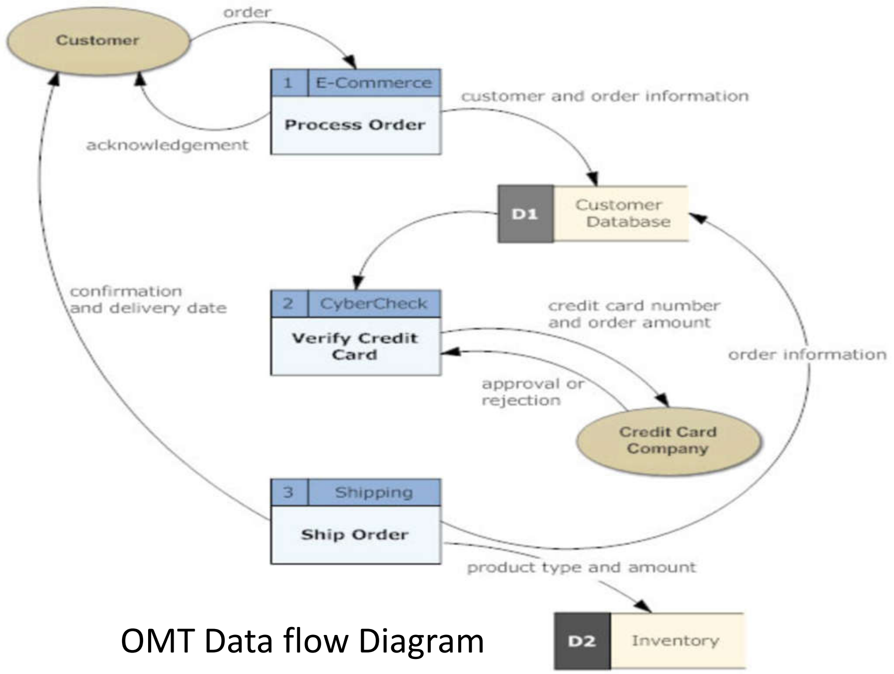
Object-oriented modelling techniques

- **Object-oriented modelling techniques** are processes/methodologies/approaches for software modelling and designing
  - 1975 - 1990: several object-oriented techniques are developed
  - 1990 - 1994: there are more than 50 object-oriented modelling techniques

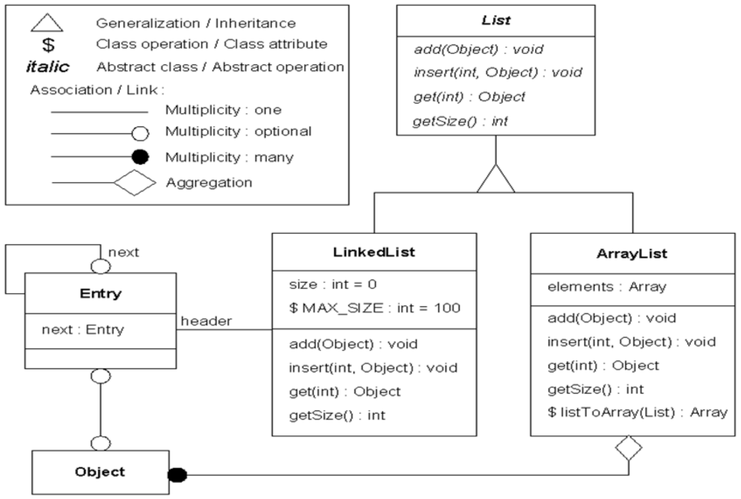
- Best-known techniques
  - OOD (Object-Oriented Design)
  - OOSE (Object-Oriented Software Engineering)
  - OMT (Object Modelling Technique)

OMT technique

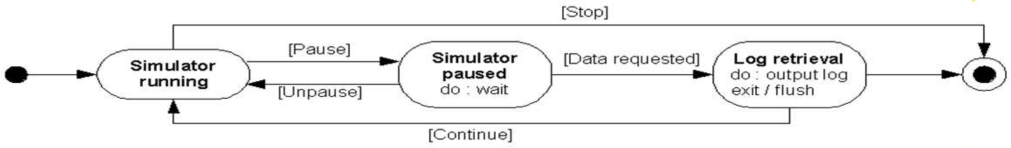
- Developed by Jim Rumbaugh (1991)
- Consists of 3 main types of models
  - Object model: Object diagram
  - Dynamic model: State diagram
  - Functional model: Data flow diagram



OMT Data flow Diagram



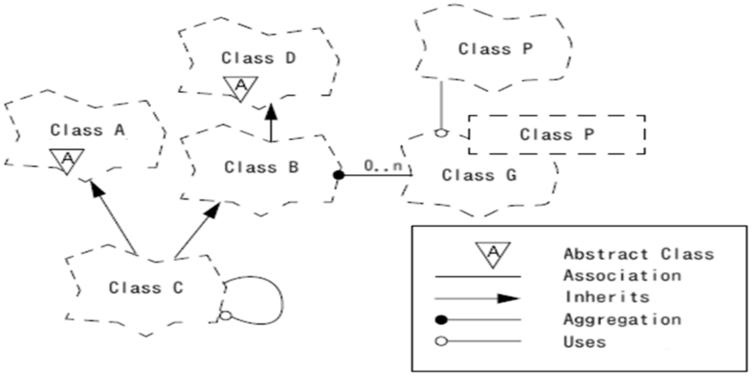
OMT Object Diagram

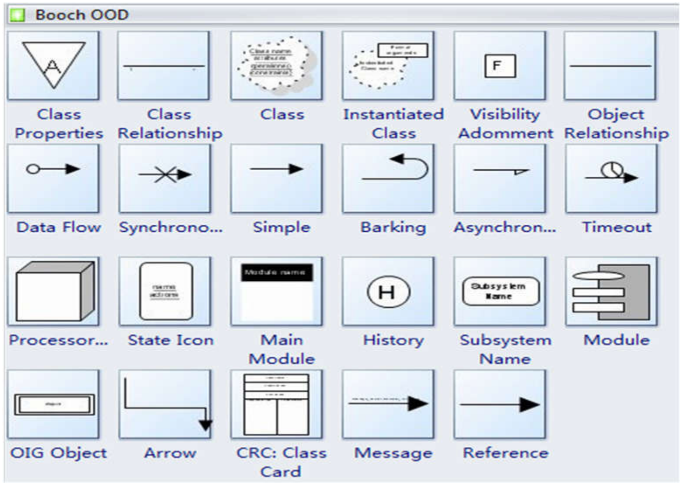


OMT State Diagram

OOD technique

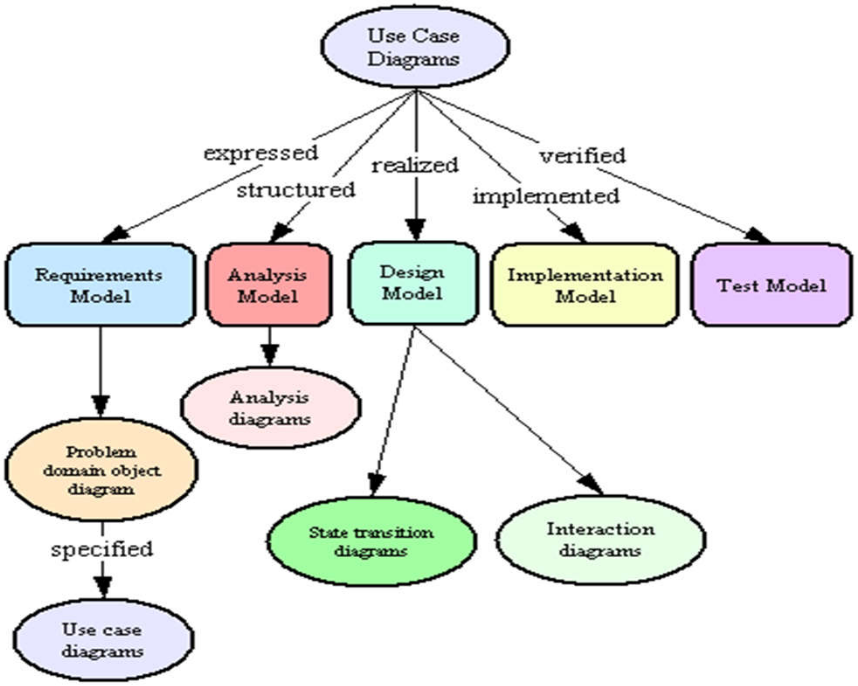
- Developed by Grady Booch (1991)
- Consists of
- Static view
    - Class diagram
    - Object diagram
    - Module diagram
- Dynamic view
    - State transition diagram
    - Process diagram
    - Interaction diagram





OOSE technique

- Developed by Ivar Jacobson (1992)
- Consists of 5 models
  - Requirements model: Problem domain diagram, Use-case diagram
  - Analysis model: Analysis diagram
  - Design model: State transition diagrams, Interaction diagrams
  - Implementation model
  - Test model

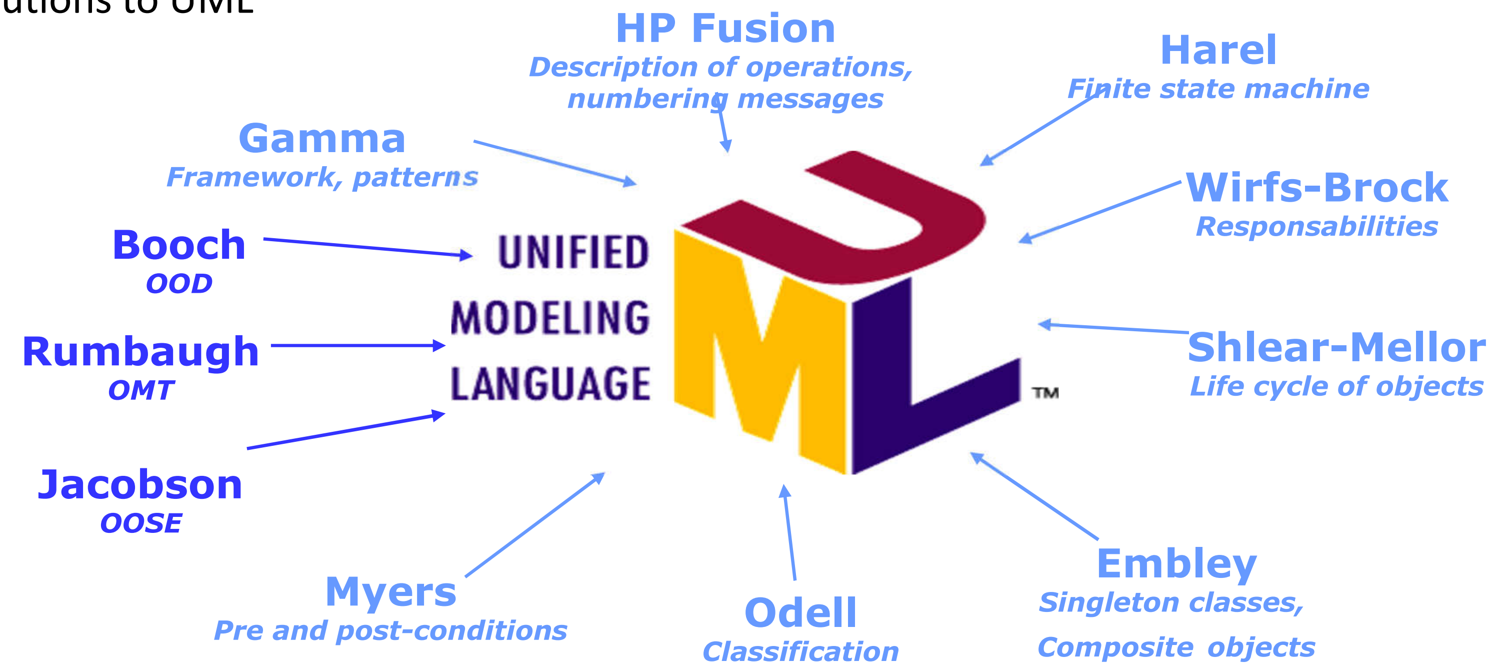


History of UML

- Too many object-oriented modelling techniques
  - Need for standardisation
    - Unification of modelling techniques
- In 1994
  - Rumbaugh and Booch unified their approaches for the UML project at Rational Software
- In 1995
  - The first version was released under the name “Unified Method” v0.8
- In 1996
  - Jacobson joined the team
- In 1997
  - The birth of UML v0.9 integrating OOSE
  - The first conference of the UML is organized
- In 2005, UML 2.0 is released
  - New diagrams, enhancement of existing diagrams
- In September, 2013, UML v.2.5 RTF - Beta 2
- In June, 2015, UML v.2.5

History of UML

- Contributions to UML



Introduction to UML

- **UML** (Unified Modelling Language) is **a modelling language**
  - consisting of the vocabulary, syntax and semantics
  - allowing to represent a system at different levels: conceptual, physical
  - consisting of vocabulary and rules to describe different models representing a system

- **UML**
  - is neither a methodology nor a process
  - allows freedom of design
  - can be combined with several development processes

Introduction to UML

- UML is **a language of visualisation**
  - using graphical representations
  - providing a better view of the system (thanks to graphical representations)
- UML is **a language of specification**
  - allowing to specify a system without ambiguity
  - allowing to specify a system at different stages: analysis, design, deployment
- UML is **a language of construction**
  - allowing to simulate the system
  - UML models are easily transformed into source code
- UML is **a language of documentation**
  - allowing to describe all the development stages of the system
  - Built models are complete documents of the system

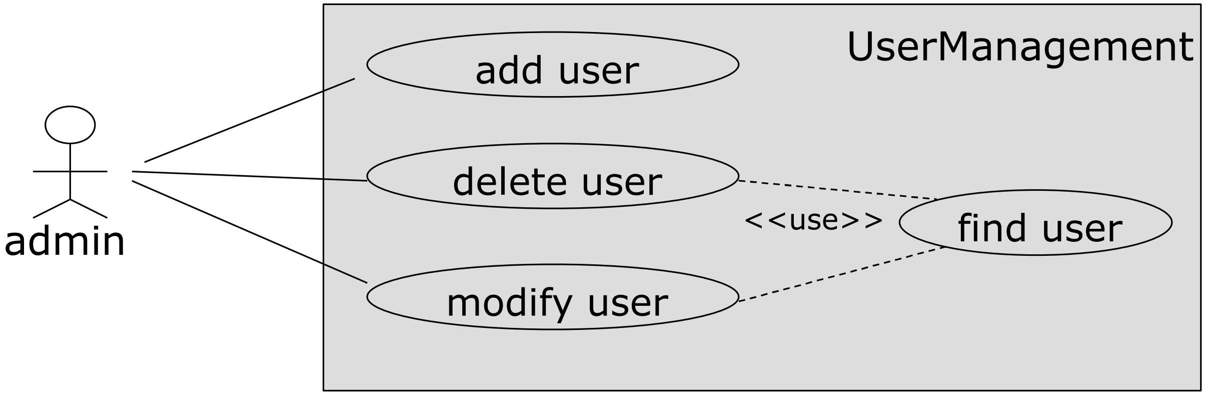
Introduction to UML: the diagrams

- Consisting of 10 main diagrams
  - **Requirements** modelling
    - Use-case diagrams
  - **Static structure** modelling
    - Class diagrams
    - Object diagrams
  - **Dynamic behavior** modelling
    - Interaction diagrams
      - Sequence diagrams
      - Collaboration diagrams
    - Activity diagrams
    - State diagrams
  - **Architectural** modelling
    - Package diagrams
    - Component diagrams
    - Deployment diagrams

Introduction to UML: Use-case diagram

- Showing the possible uses of a system
- Describing **the static view** of the system according to users perspective
- Being very important to understand the functions of the system

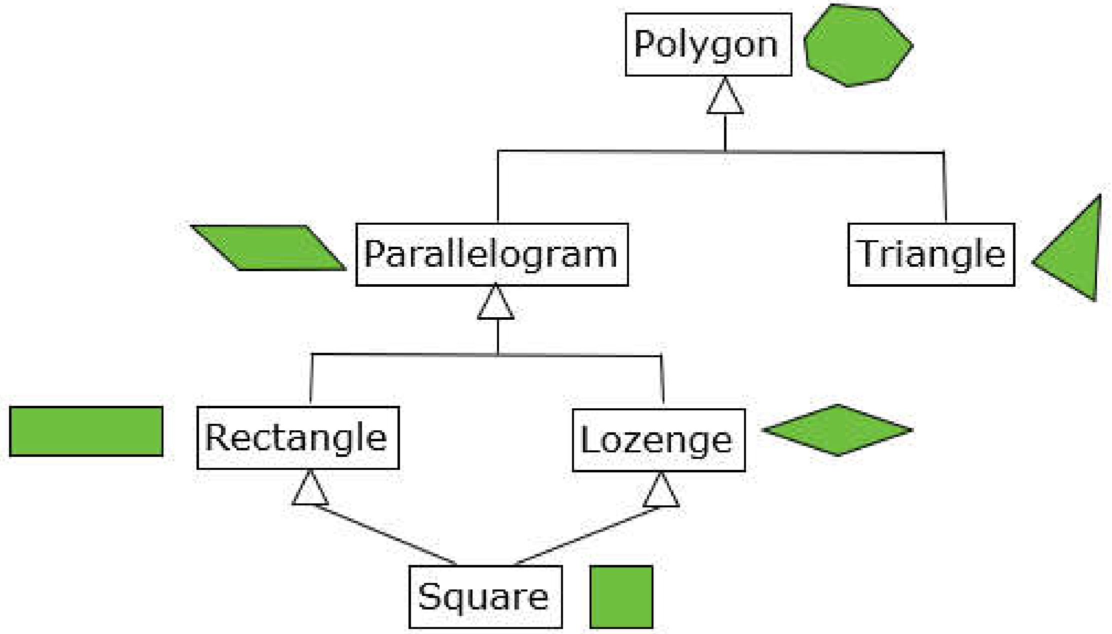
- Example



Introduction to UML: class diagram

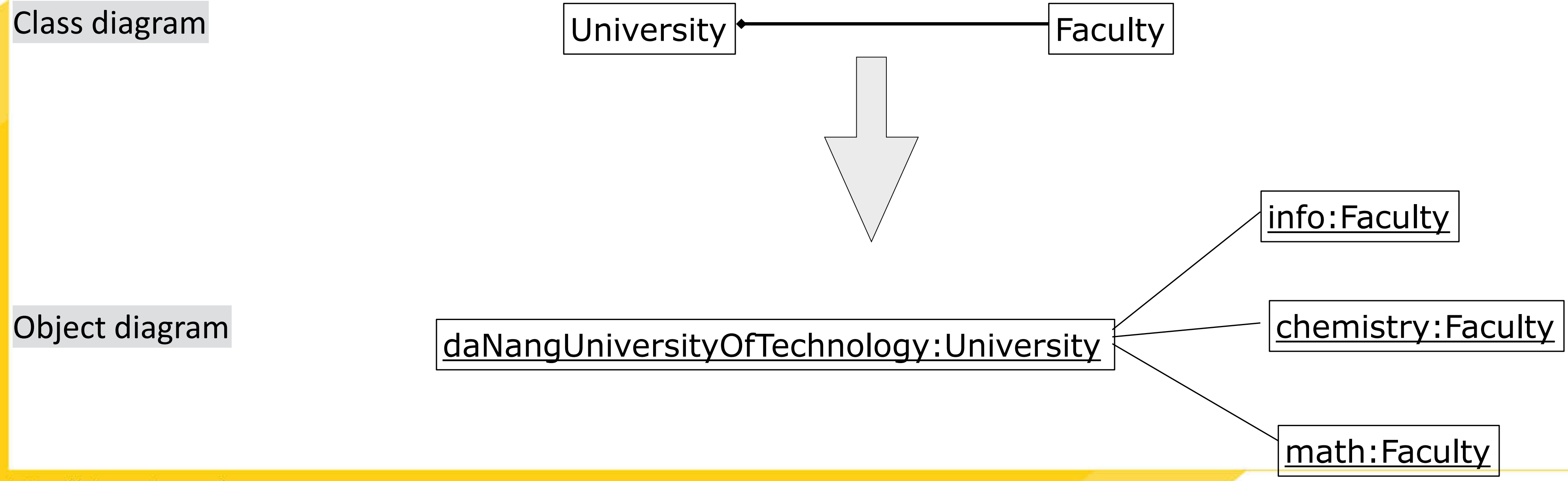
- Describing the classes and their relationship
- Describing **the static view** of the system

- Example



Introduction to UML: object diagram

- Describing a set of objects and their relationship
- An object diagram represents the same information that a class diagram but at the instance level of classes
- Describing **the static view** of the system
- Example



Class diagram

University

Faculty

Object diagram

daNangUniversityOfTechnology:University

info:Faculty

chemistry:Faculty

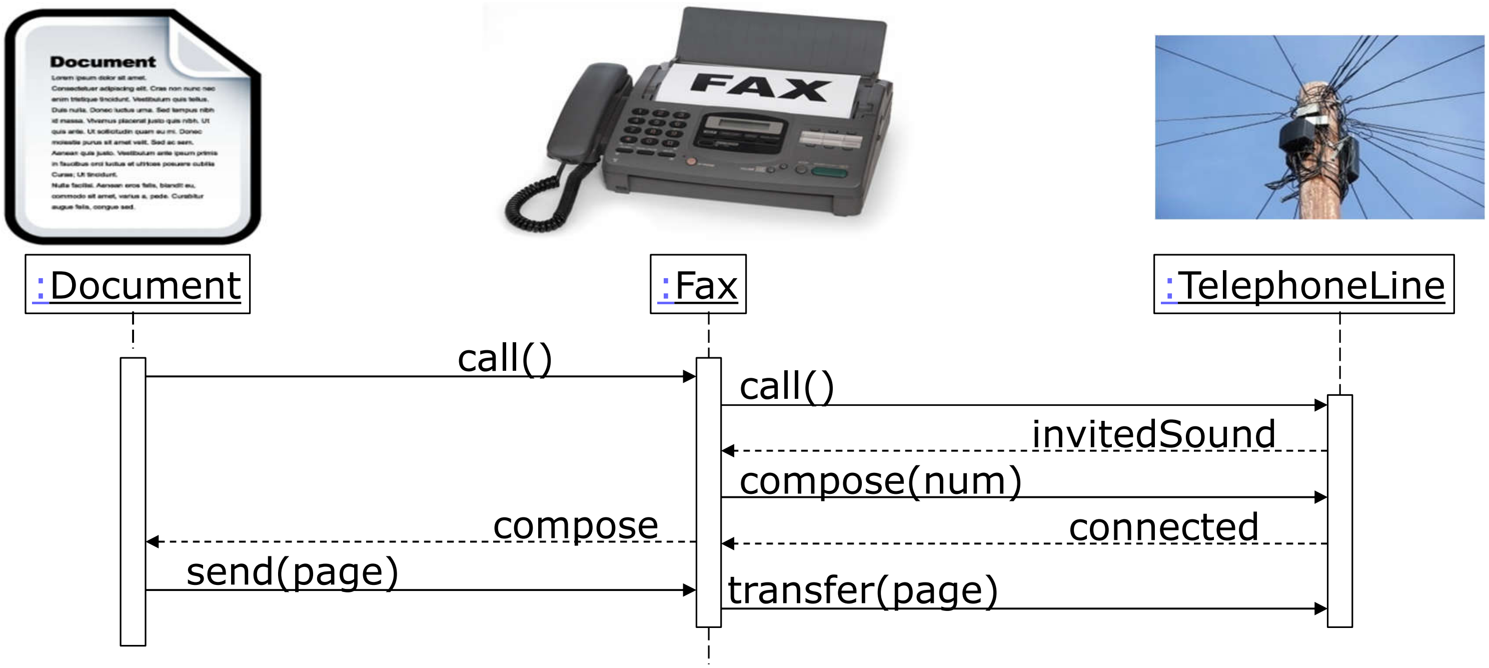
math:Faculty

Introduction to UML: interaction diagram

- Describing the behaviours of the system by the interactions between the composing objects
- Modelling **the dynamic view** of the system
- The interaction diagram is an extension of the object diagram by describing the interactions between objects
- Consisting of two types of diagrams
  - **Sequence Diagram** describes the interactions between objects with the emphasis on sequencing of messages
  - **Collaboration Diagram** describes the interactions between objects with the emphasis on the structure of objects

Introduction to UML: interaction diagram

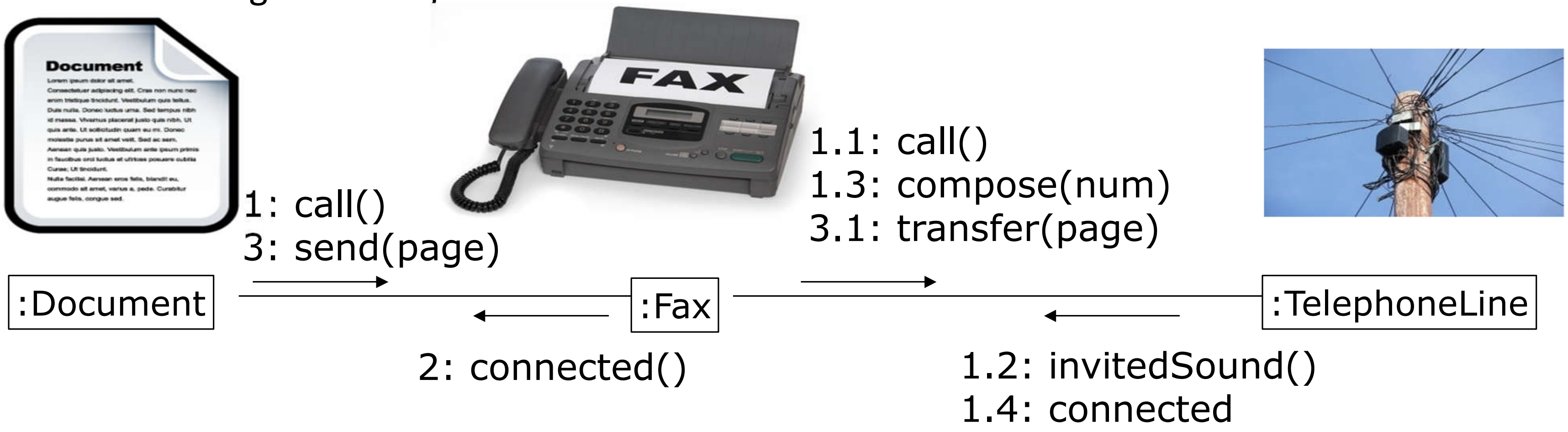
- Sequence Diagram example



“Sending Fax” Sequence Diagram

Introduction to UML: interaction diagram

- Collaboration diagram example



:Document

:Fax

:TelephoneLine

1: call()
3: send(page)

2: connected()

1.1: call()
1.3: compose(num)
3.1: transfer(page)

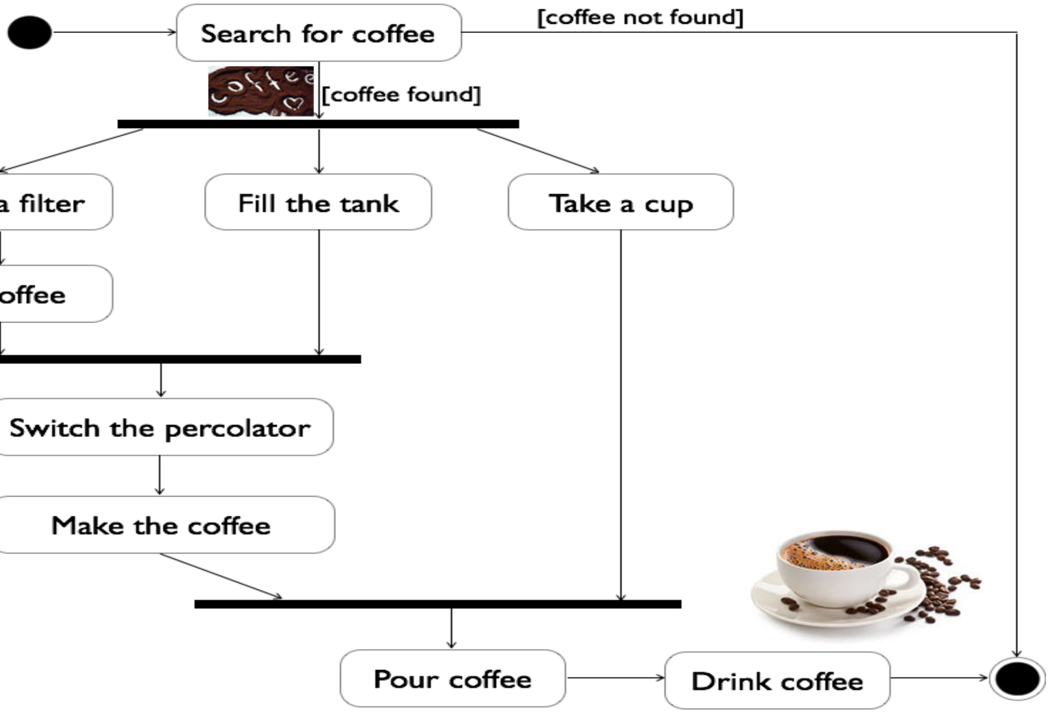
1.2: invitedSound()
1.4: connected

“Sending Fax” Collaboration Diagram

Introduction to UML: activity diagram

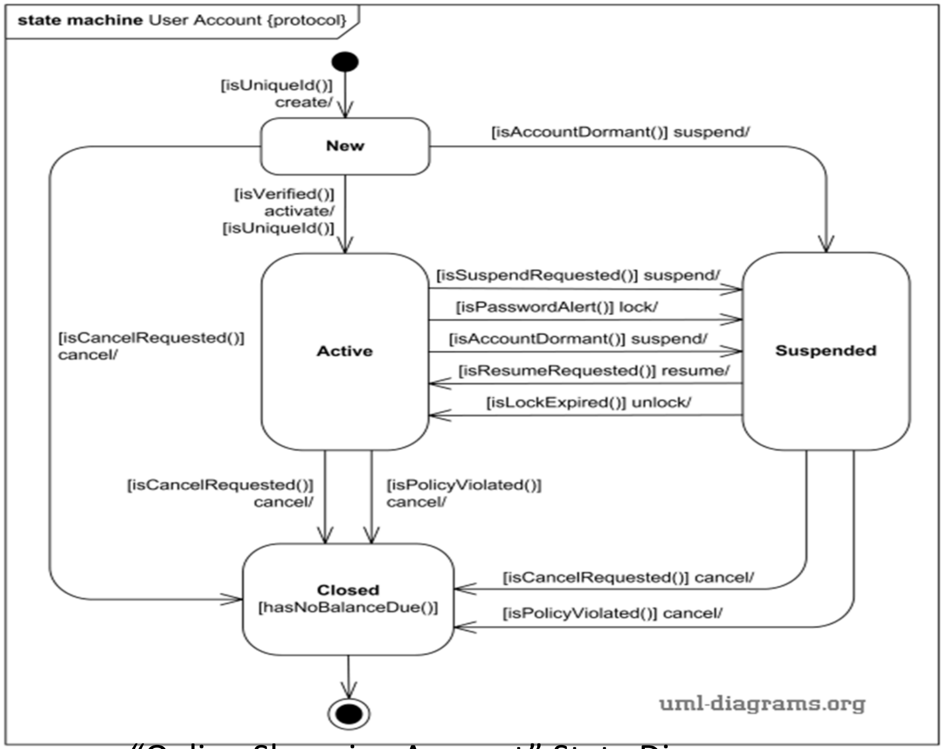
- Describing the information flows in the system
- Modelling **the dynamic view** of the system

- Example: Making coffee



Introduction to UML: state diagram

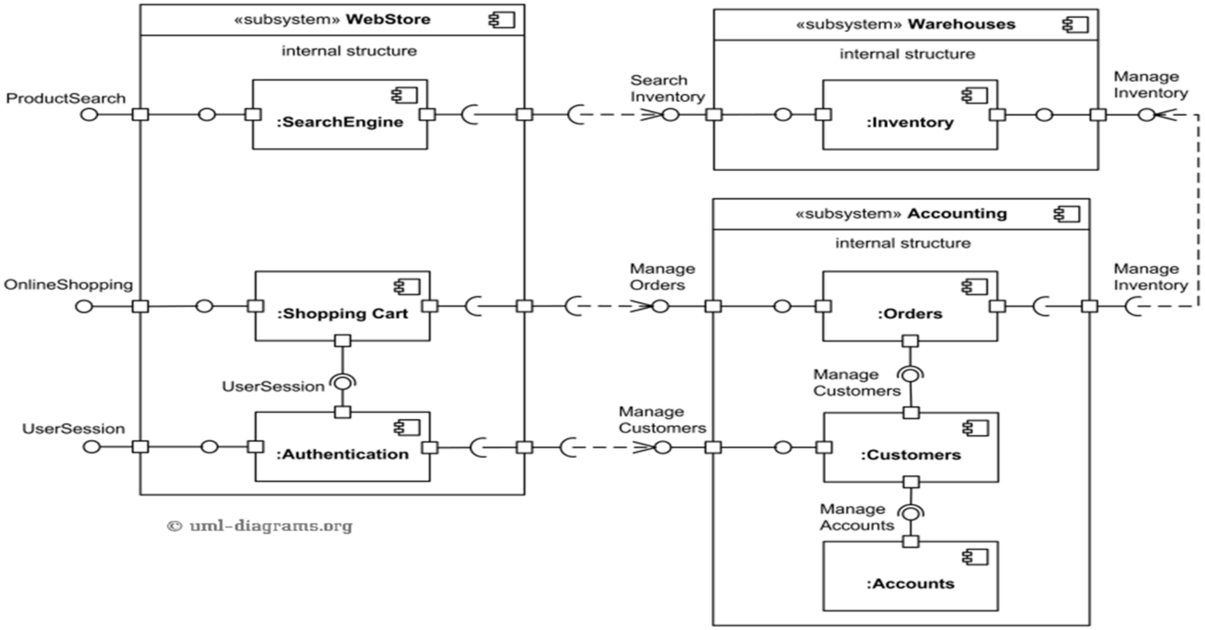
- Describing the internal behaviour of the system
- Modelling the **dynamic view** of the system
- Example


state machine User Account {protocol}
[isUniqueId()] create/
New
[isAccountDormant()] suspend/
[isVerified()] activate/ [isUniqueId()]
[isCancelRequested()] cancel/
Active
[isSuspendRequested()] suspend/
[isPasswordAlert()] lock/
[isAccountDormant()] suspend/
[isResumeRequested()] resume/
[isLockExpired()] unlock/
Suspended
[isCancelRequested()] cancel/
[isPolicyViolated()] cancel/
Closed [hasNoBalanceDue()]
[isCancelRequested()] cancel/
[isPolicyViolated()] cancel/
uml-diagrams.org

"Online Shopping Account" State Diagram

Introduction to UML: component diagrams

- Describe the organisation of different components of the system
- The **static view** of the organisation of the system
- Example



"Online Shopping Website" Component Diagram

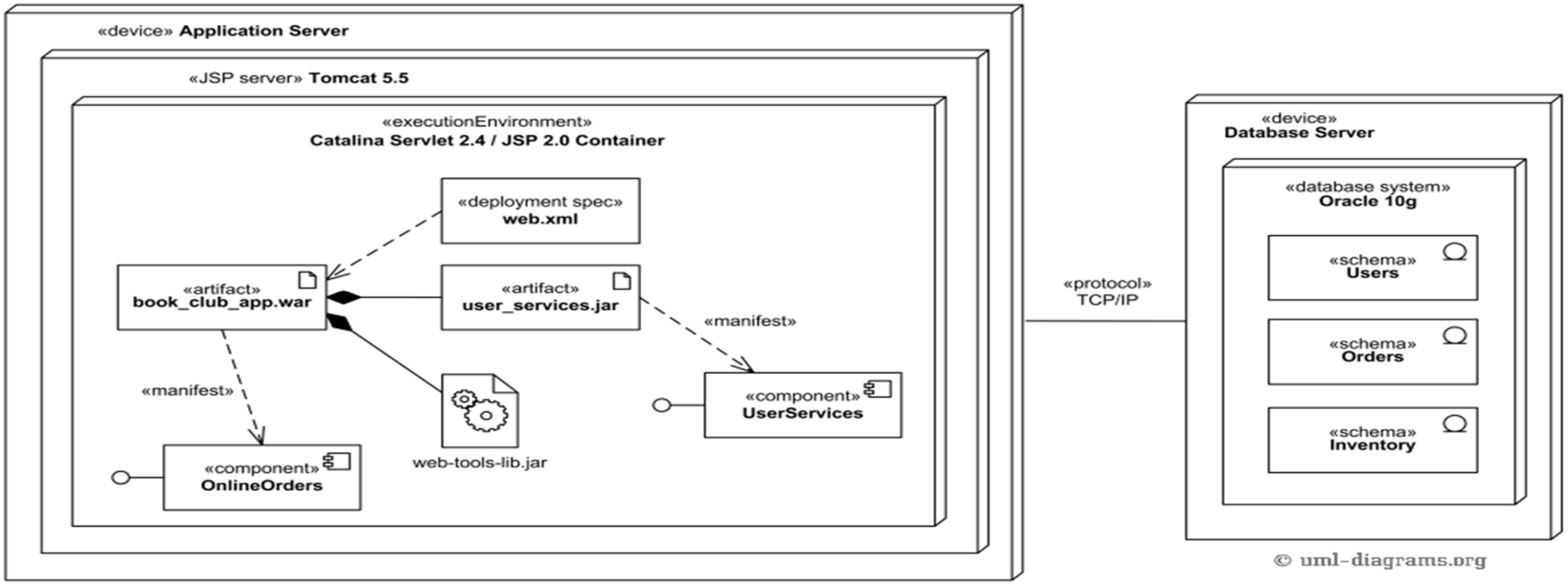
ebay

amazon

Alibaba.com

Introduction to UML: deployment diagrams

- Describing the physical organisation of different components (machines) of the system (material)



© uml-diagrams.org

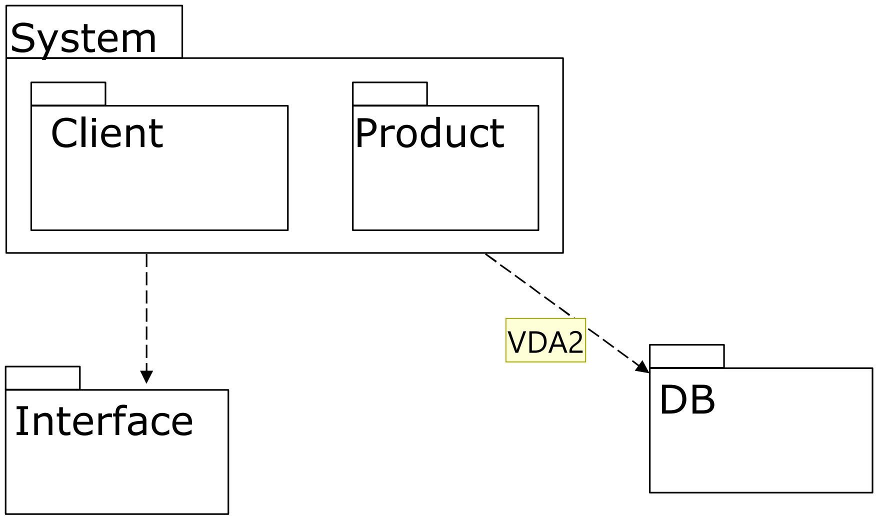
An example of deployment diagram of JEE web application

Introduction to UML: extension mechanism

- Built-in extension mechanism
  - Stereotypes
  - Tagged values
- Notes
- Constraints
  - OCL (Object Constraint Language) textual language

Introduction to UML: general mechanisms

- Packages
  - Allow to structure class diagrams
  - Build a dependence structures between packages
  - Example



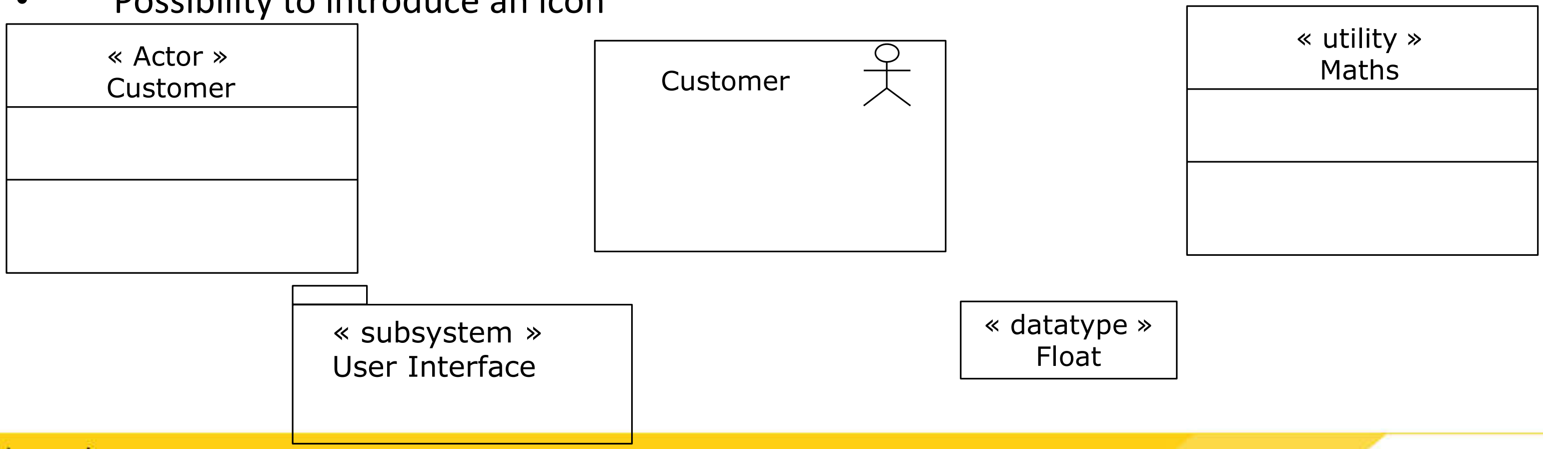
Slide 32

VDA2 “package” diagram?

VO Duc An, 17/06/2015

Introduction to UML: general mechanisms

- Stereotype
  - is a built-in extension mechanism
  - expands the vocabulary of UML
  - is used to create new types of UML elements that derive from the existing kinds but which are adapted to a given problem
  - there are predefined stereotypes in UML
  - Notation
    - “name of stereotype”
    - Possibility to introduce an icon



« Actor »
Customer

Customer

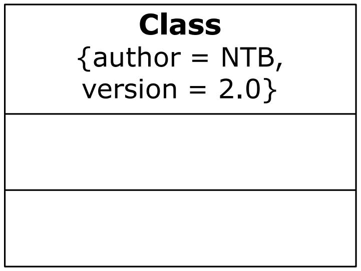
« utility »
Maths

« subsystem »
User Interface

« datatype »
Float

Introduction to UML: general mechanisms

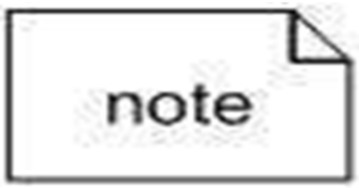
- Tagged values
    - Another extension mechanism
    - Provide additional information on the elements of UML
    - Pairs of type {name = value}
    - Example



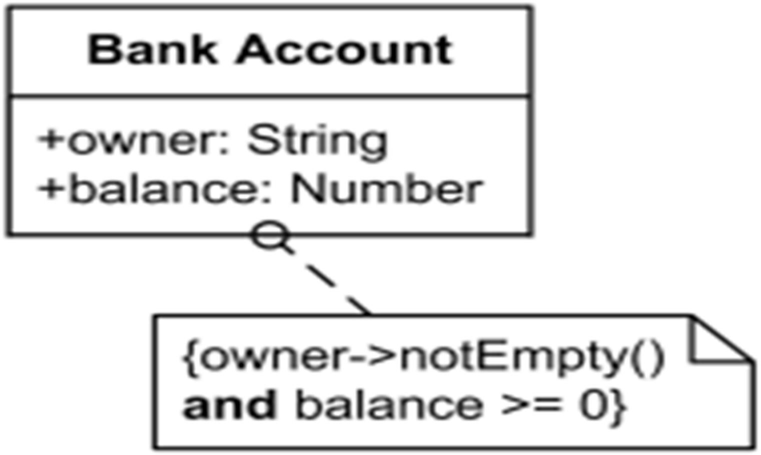
Class
{author = NTB,
version = 2.0}

Introduction to UML: general mechanisms

- Notes
  - are comments attached to one or more modelling elements
  - provide additional information on modelling elements
  - belong to the view, not the models



note



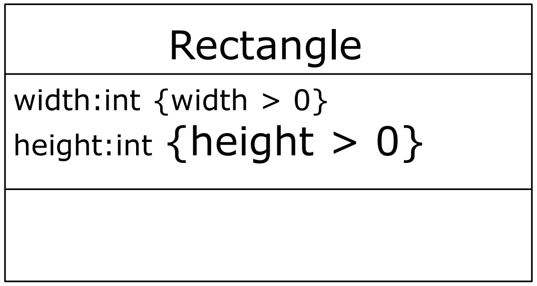
Bank Account

+owner: String
+balance: Number

{owner->notEmpty()
and balance >= 0}

Introduction to UML: general mechanisms

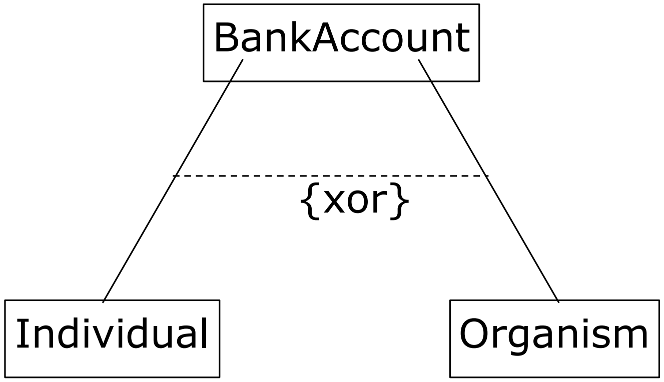
- Constraints
  - are restrictions that limit the use of an element or the element semantic
  - are expressed in natural language
  - are expressed in OCL (Object Constraint Language)
  - Example



Rectangle

width:int {width > 0}

height:int {height > 0}



BankAccount

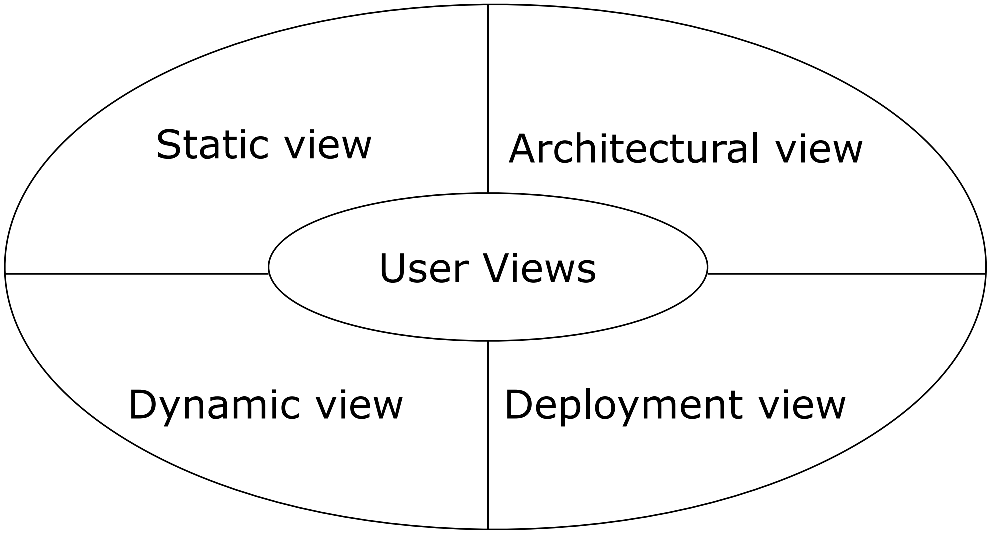
{xor}

Individual

Organism

Introduction to UML: views

- A system is modelled by 5 different views in the UML



Static view

Architectural view

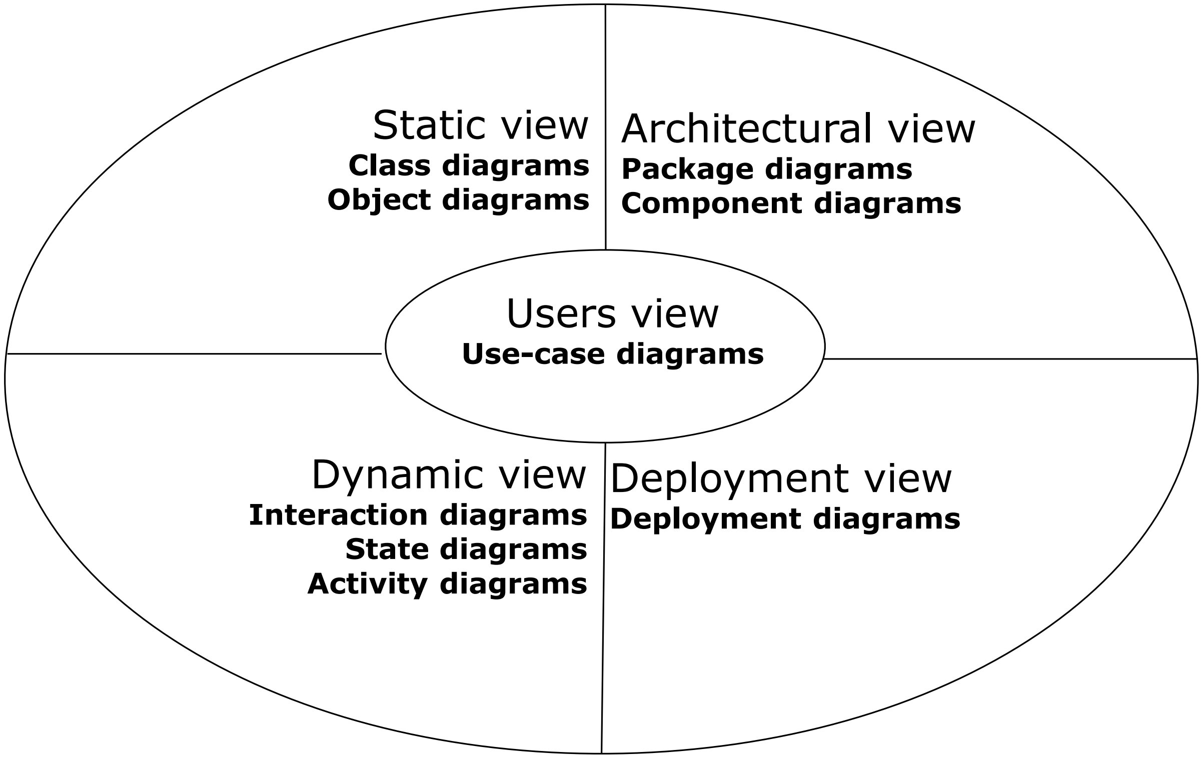
User Views

Dynamic view

Deployment view

Introduction to UML: views

- Diagrams and views



Static view
Class diagrams
Object diagrams

Architectural view
Package diagrams
Component diagrams

Users view
Use-case diagrams

Dynamic view
Interaction diagrams
State diagrams
Activity diagrams

Deployment view
Deployment diagrams

Chapter 2.
An overview of UML