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

SYSTEM ANALYSIS AND DESIGN

- Nguyen Thanh Binh, Nguyen Quang Vu, Le Viet Truong, Vo Van Luong, Nguyen Ngoc Huyen Tran
- Faculty of Computer Science

- Chapter 2

An overview of Unified modeling language

An overview of Unified modeling language

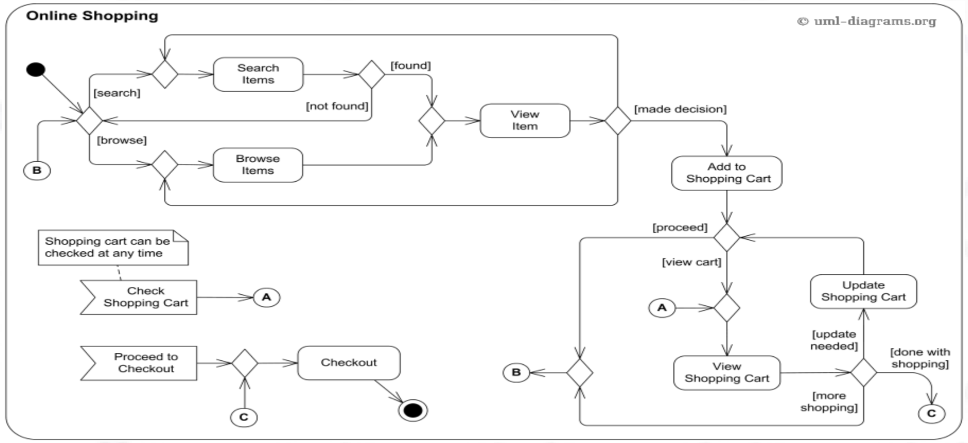
- Modeling concept
- Object-oriented modeling techniques
- History of UML
- Brief introduction to UML
    - Notions
    - Diagrams
    - Views

Model and modeling

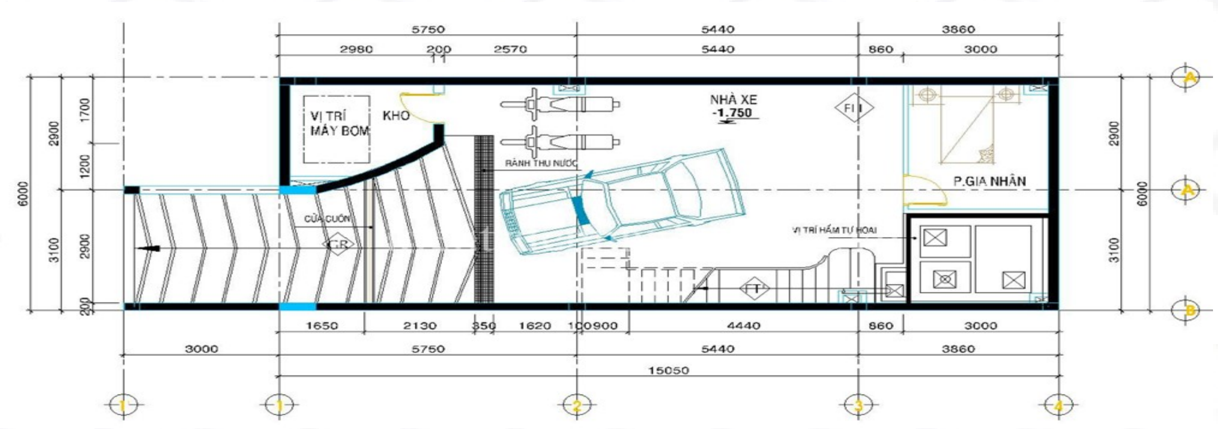
- A **model** is a simplification of reality. We build models so we can better understand the system we are developing.

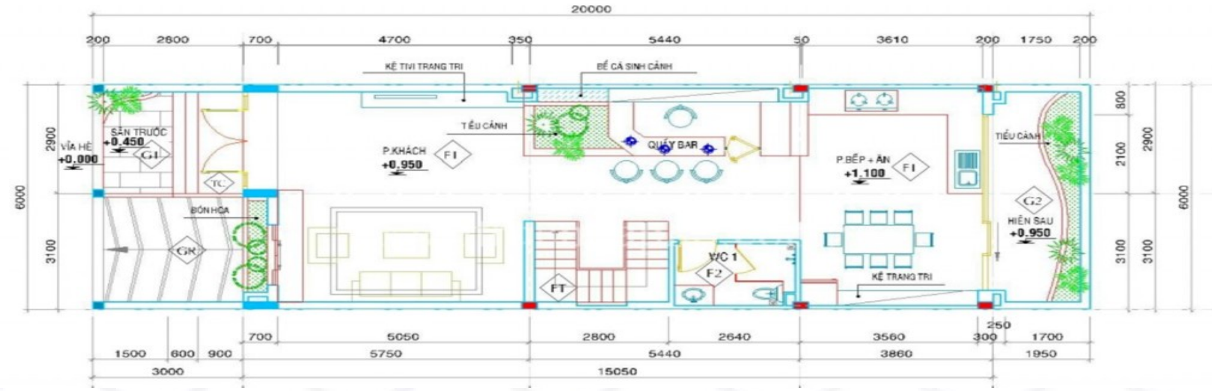
- **Modeling** is the process of building models to represent a system

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

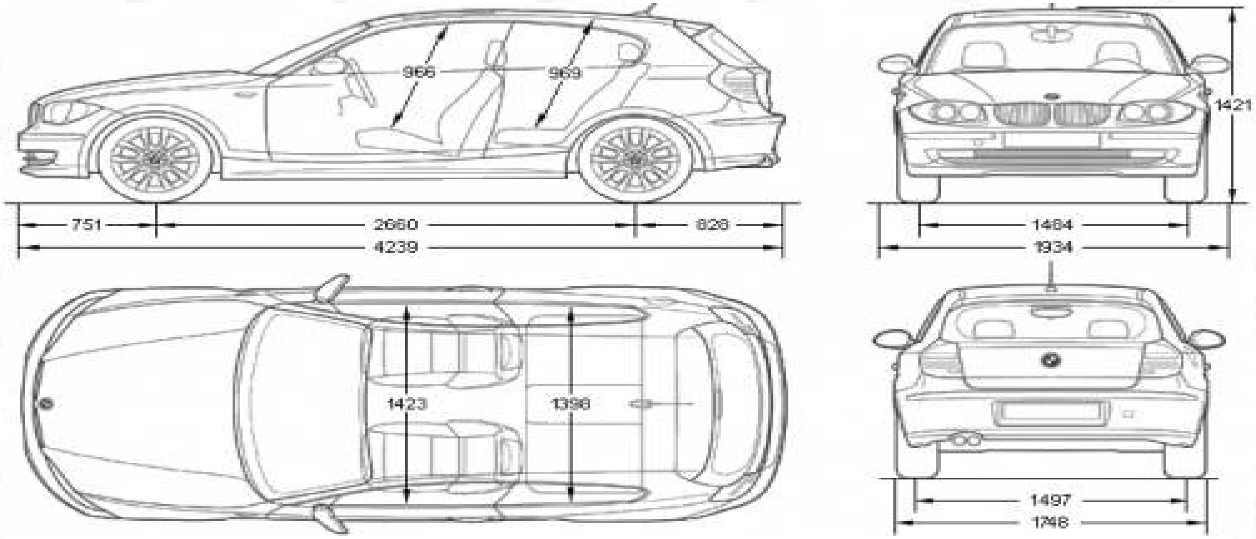


Model and modeling: Example





Model and modeling: Example



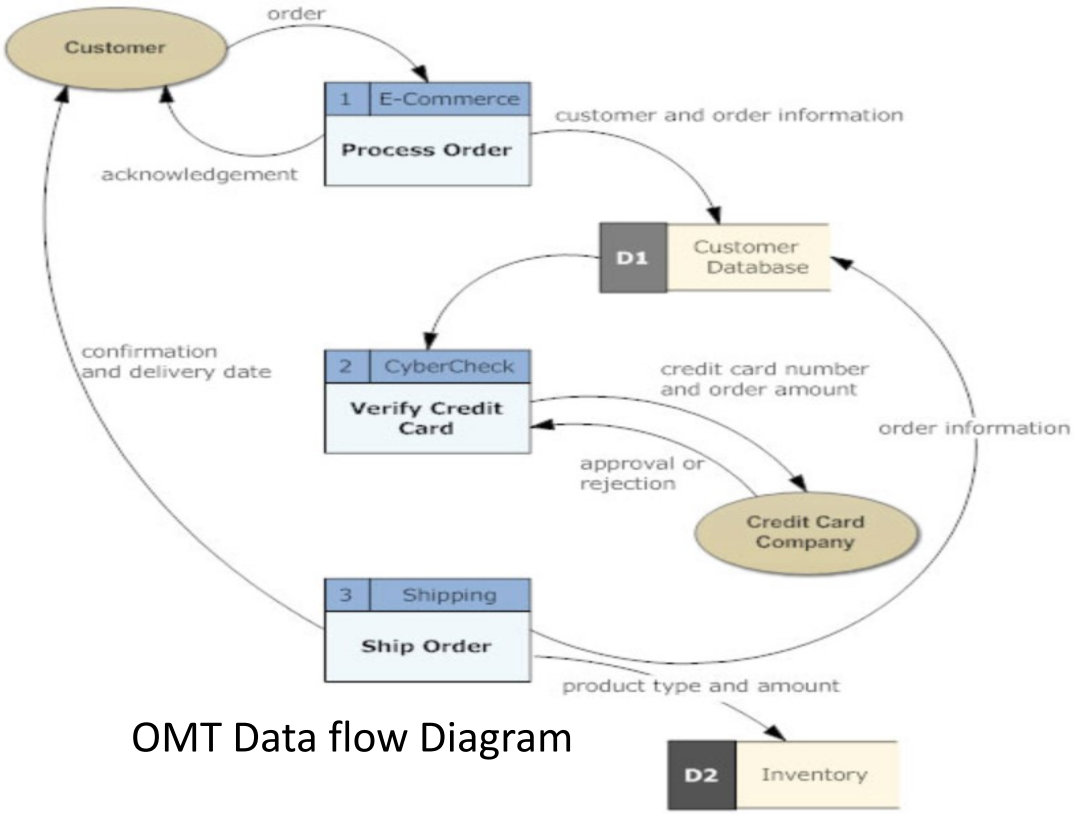
Object-oriented modeling techniques

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

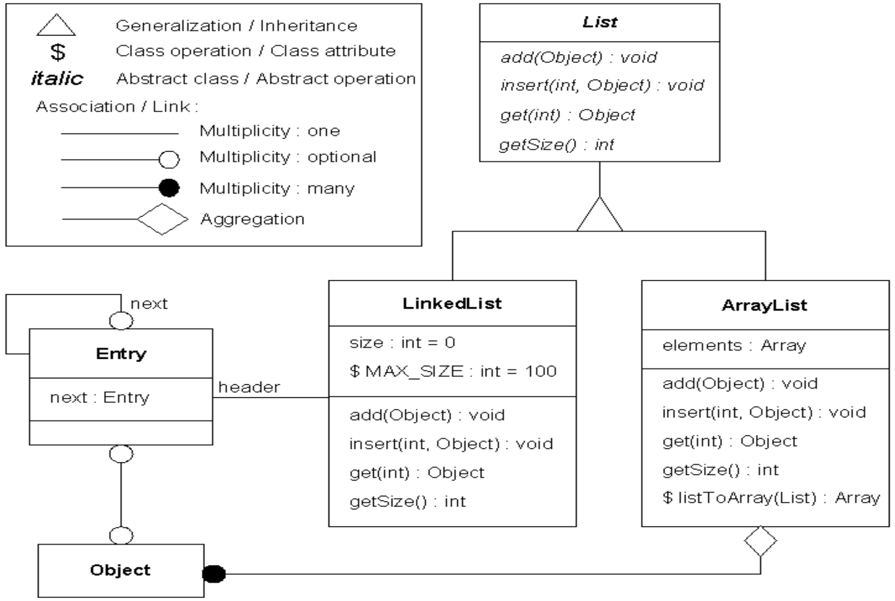
- Best-known techniques
  - OOD (Object-Oriented Design)
  - OOSE (Object-Oriented Software Engineering)
  - OMT (Object Modeling Technique)

OMT technique

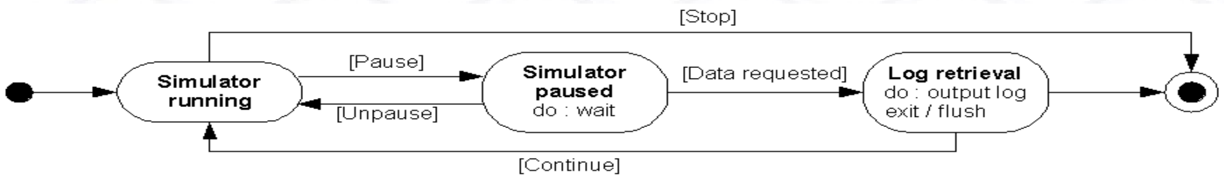
- Developed by Jim Rumbaugh (1991)
- Consists of 3 main types of models
  - Object model: Object diagram
  - Dynamic model: State diagram
  - Functional model: Data flow diagram



OMT Data flow Diagram



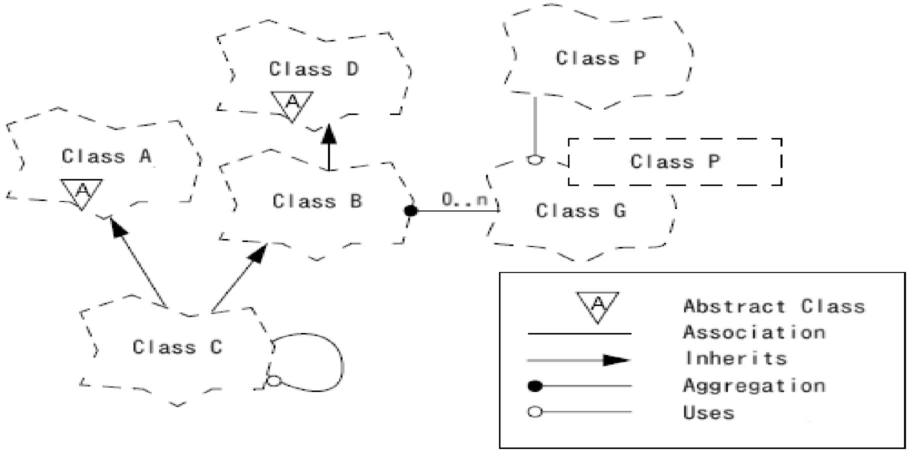
OMT Object Diagram

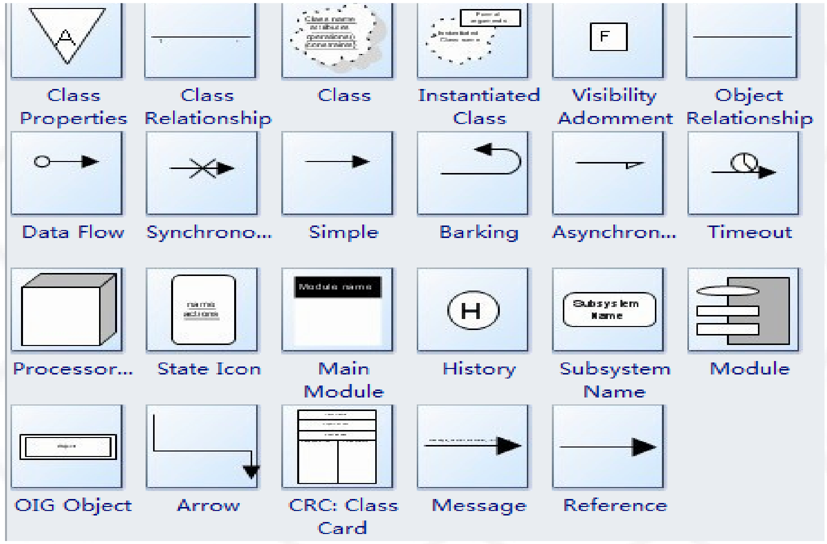


OMT State Diagram

OOD technique

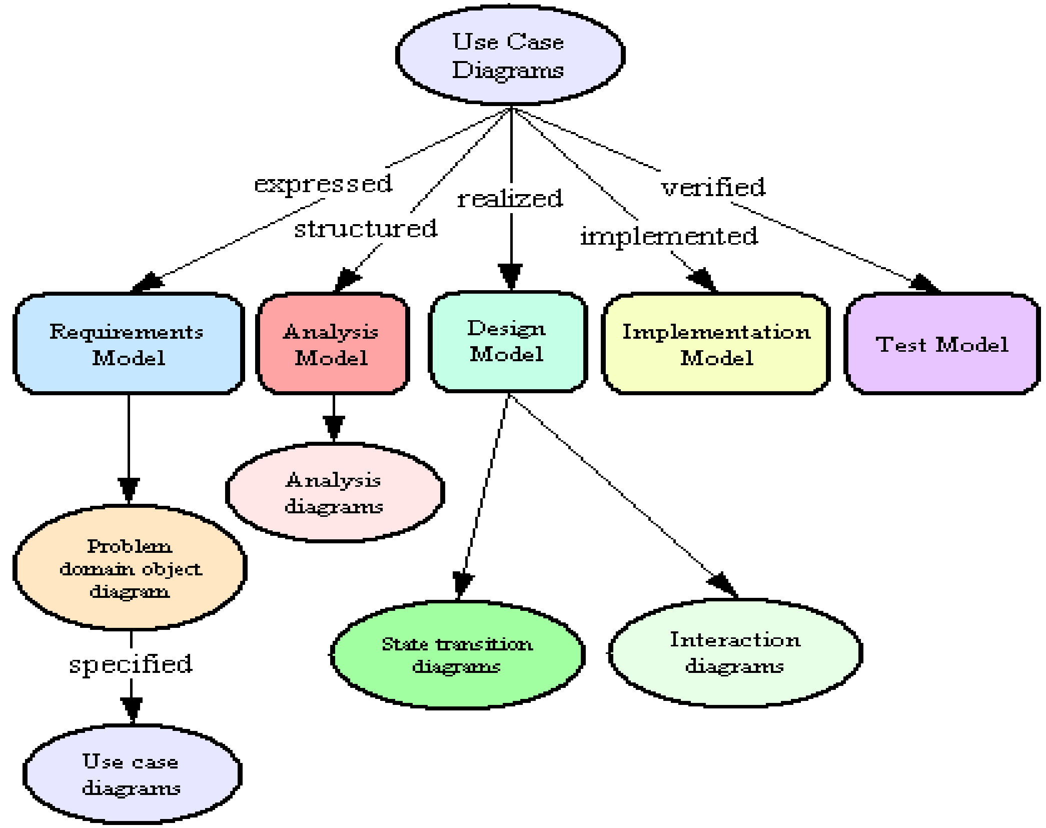
- Developed by Grady Booch (1991)
- Consists of
- Static view
  - Class diagram
  - Object diagram
  - Module diagram
- Dynamic view
  - State transition diagram
  - Process diagram
  - Interaction diagram





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, Inter
  - Implementation model
  - Test model

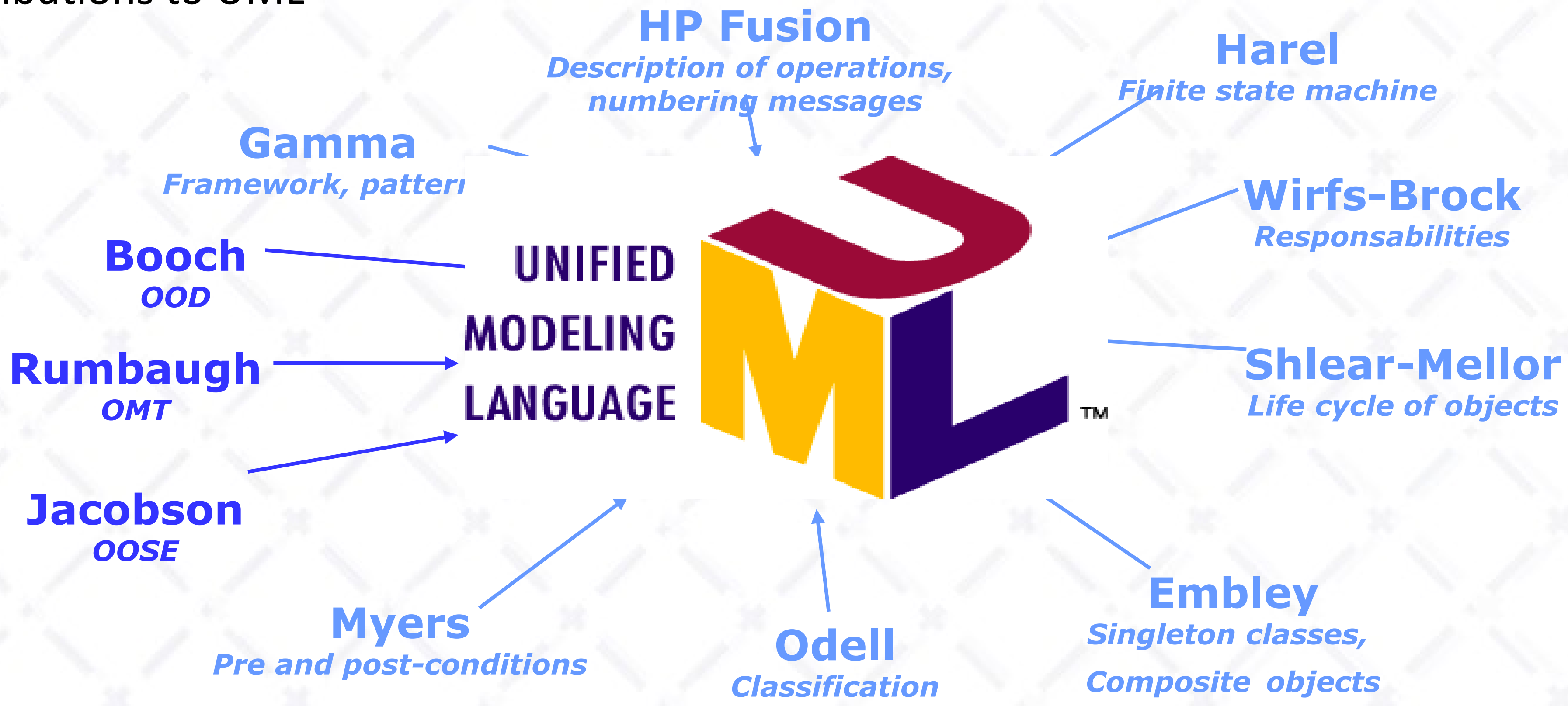


History of UML

- Too many object-oriented modeling techniques
  - Need for standardisation
    - Unification of modeling 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



HP Fusion
Description of operations, numbering messages

Harel
Finite state machine

Gamma
Framework, patter

Wirfs-Brock
Responsabilities

Booch
OOD

Rumbaugh
OMT

Shlear-Mellor
Life cycle of objects

Jacobson
OOSE

Myers
Pre and post-conditions

Odell
Classification

Embley
Singleton classes,
Composite objects

Introduction to UML

- **UML** (Unified Modeling Language) is **a modeling 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

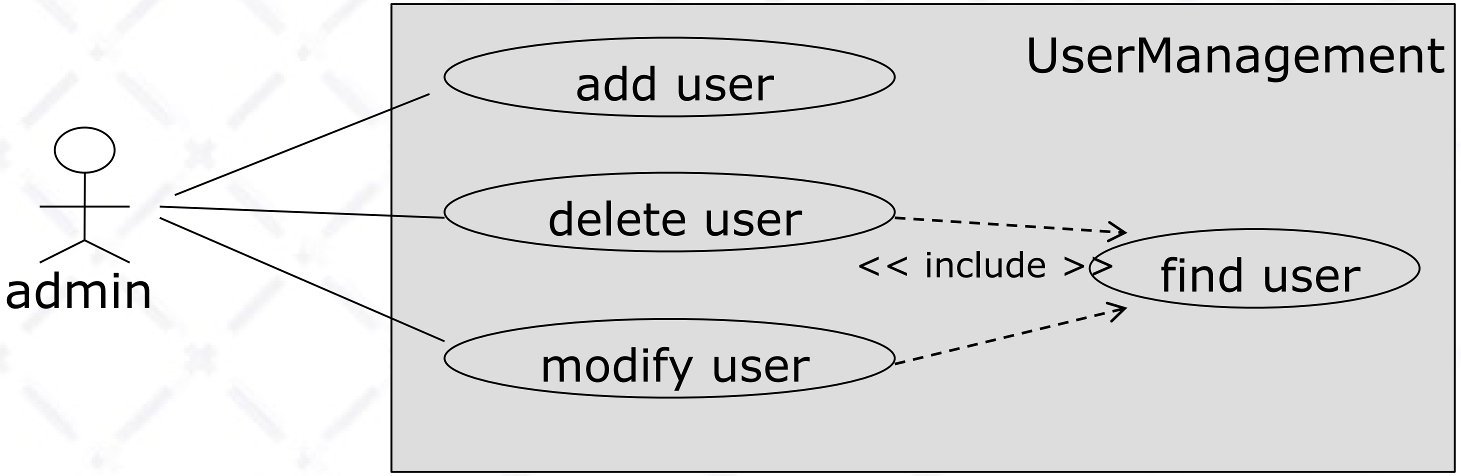
- Consisting of 10 main diagrams
  - **Requirements modeling**
    - Use-case diagrams
  - **Static structural modeling**
    - Class diagrams
    - Object diagrams
  - **Dynamic behavioral modeling**
    - Activity diagrams
    - State diagrams
    - Interaction diagrams
      - Sequence diagrams
      - Communication diagrams
  - **Architectural modeling**
    - Package diagrams
    - Component diagrams
    - Deployment diagrams

UNIFIED
MODELING
LANGUAGE

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

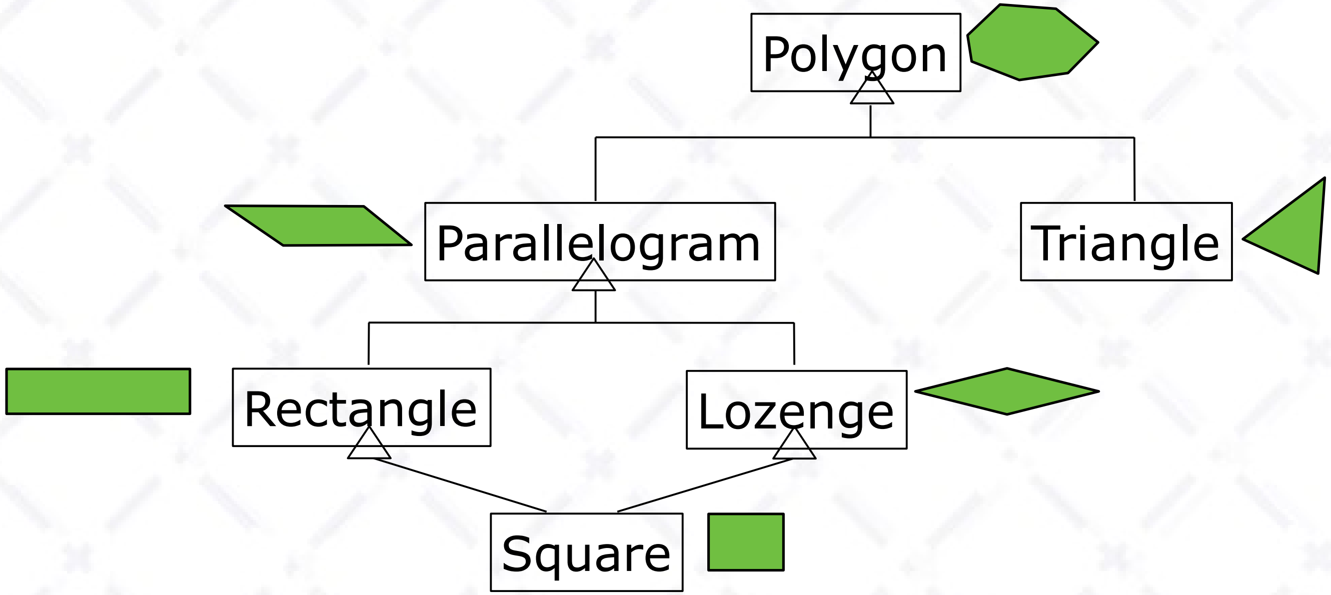
- Example



Introduction to UML: class diagram

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

- Example

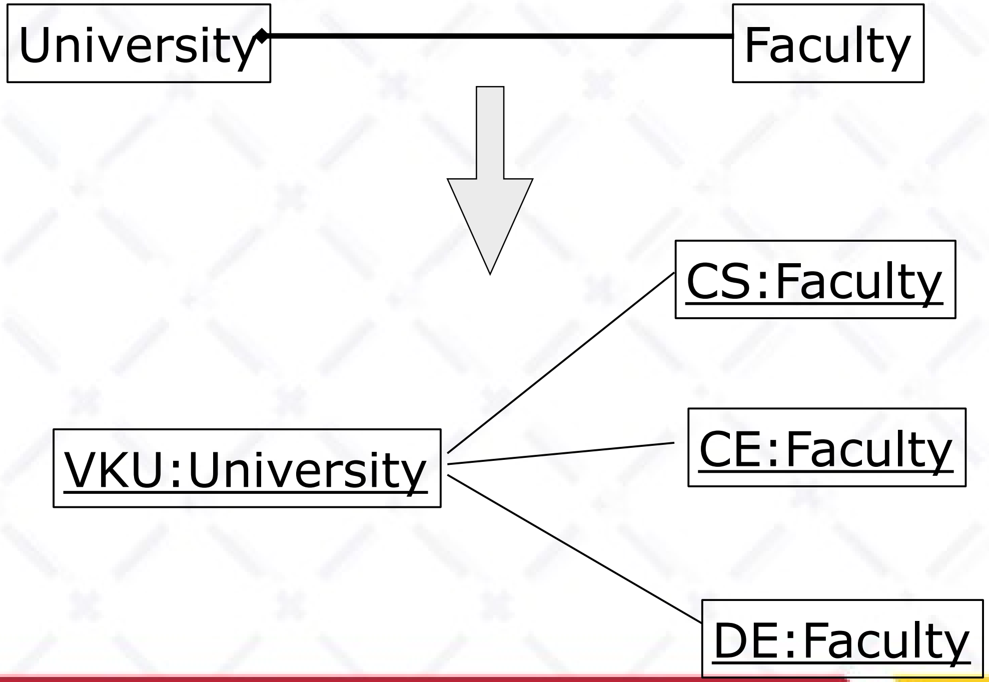


Introduction to UML: object diagram

- 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

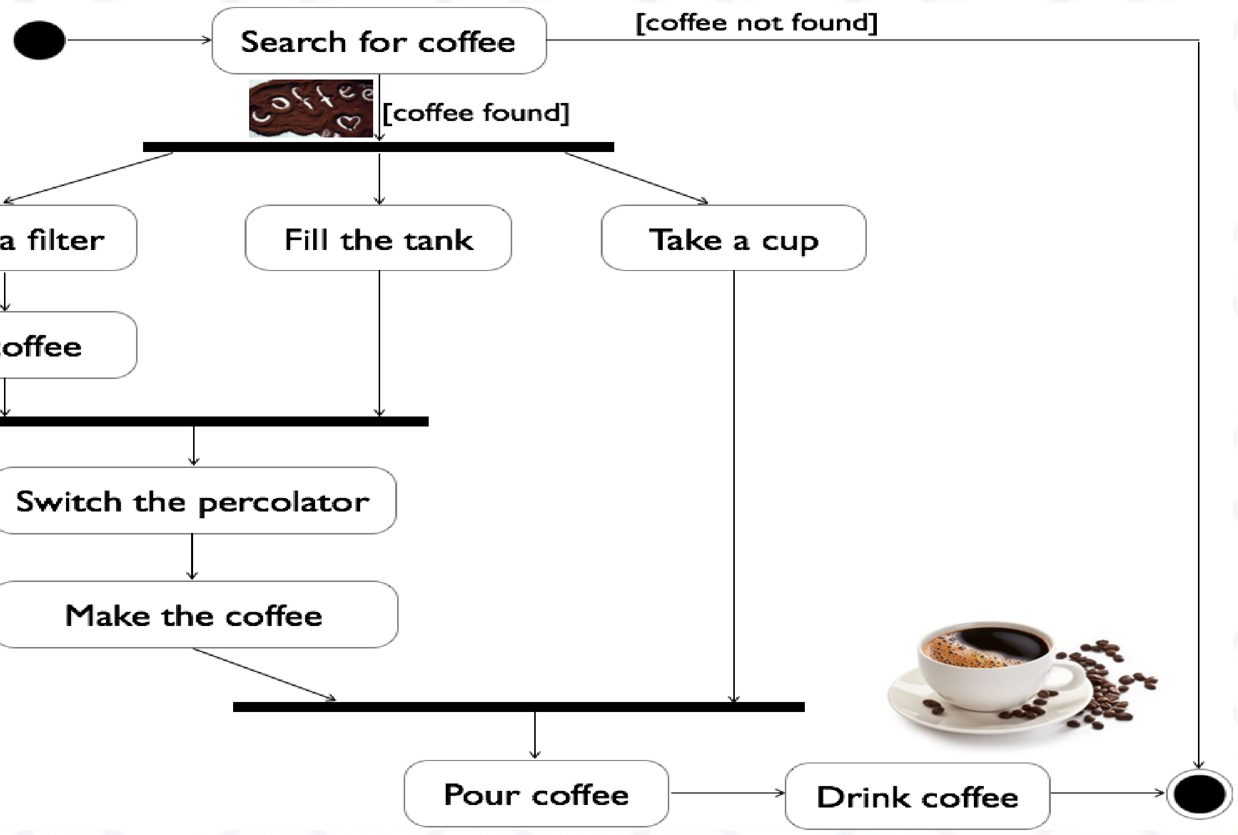
Object diagram



Introduction to UML: activity diagram

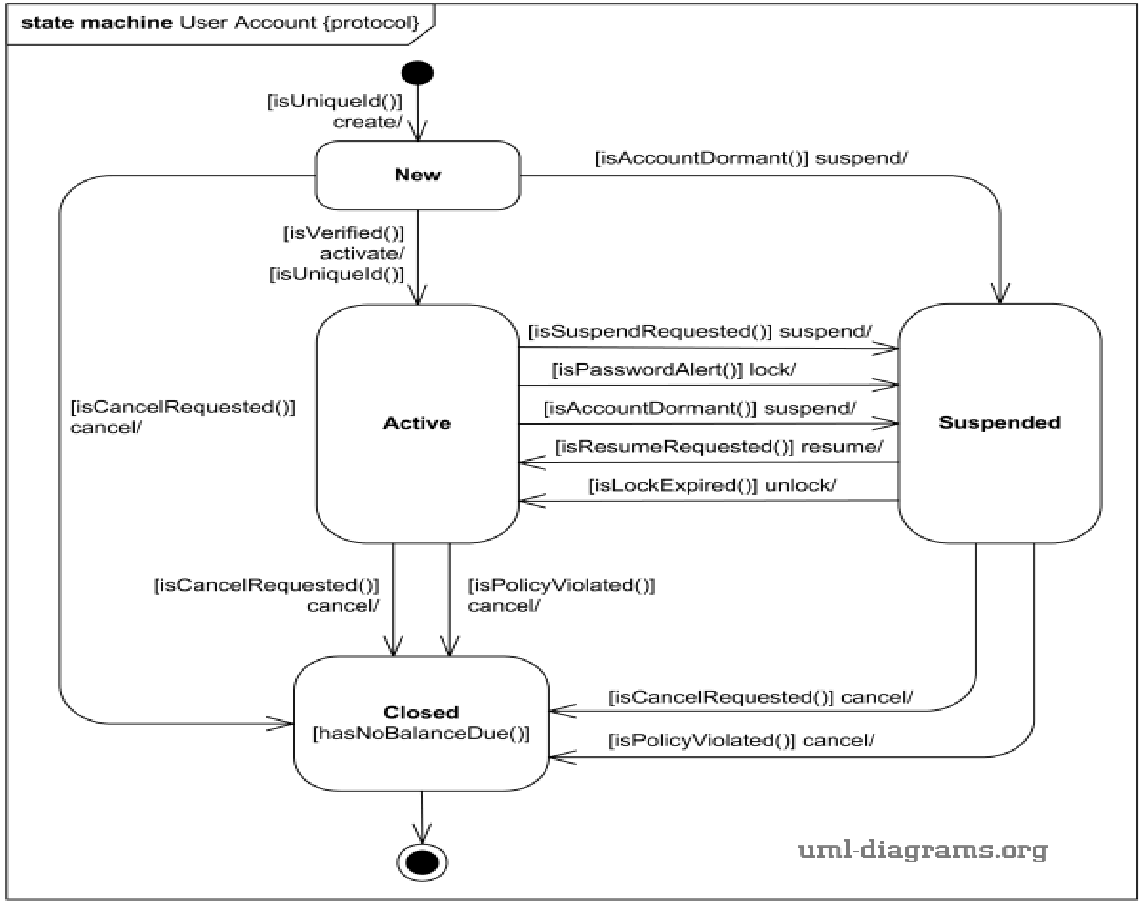
- Describing the information flows in the system
- Modeling **the dynamic view** of the system

- Example: Making coffee



Introduction to UML: state diagram

- Describing the internal behaviour of the system
- Modeling 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

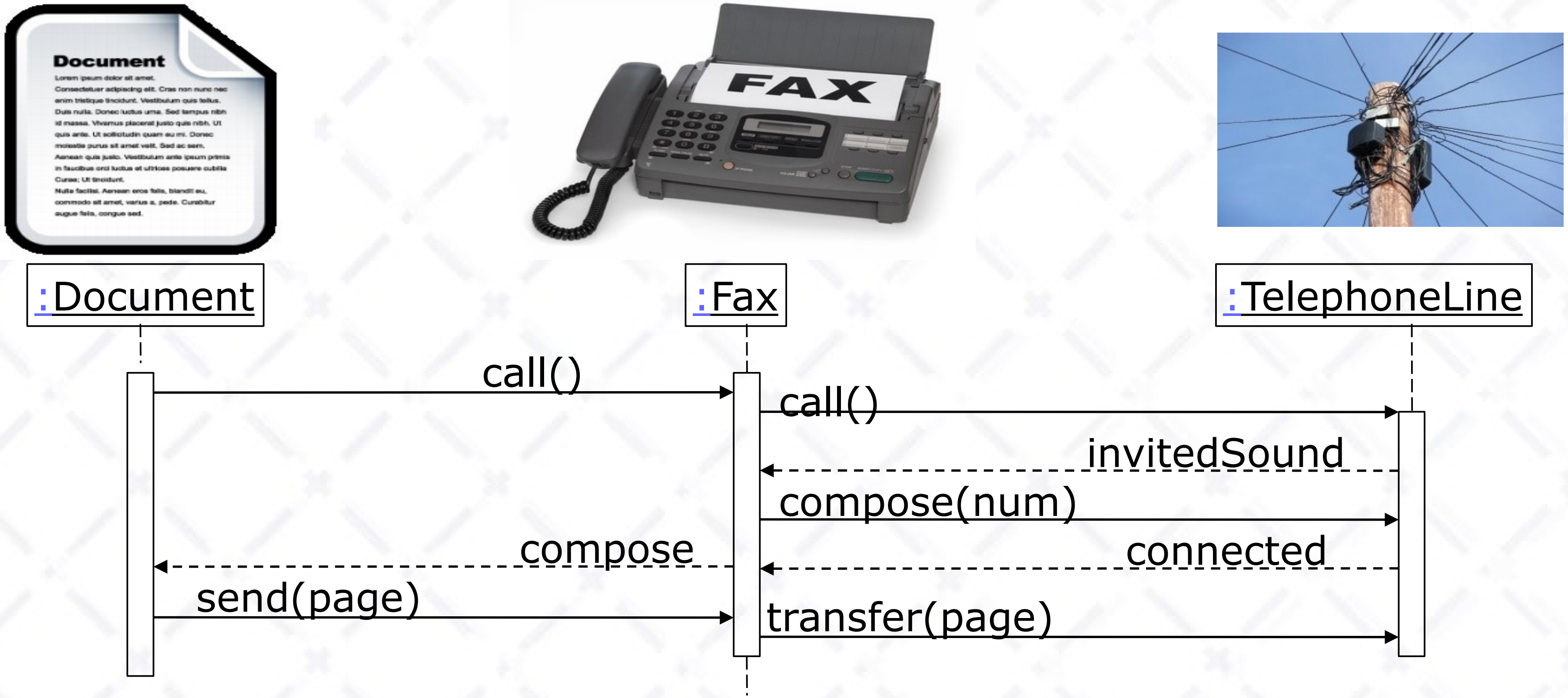
“Online Shopping Account” State Diagram

Introduction to UML: interaction diagram

- Describing the behaviours of the system by the interactions between the composing objects
- modeling **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
  - **Communication Diagram** describes the interactions between objects with the emphasis on the structure of objects

Introduction to UML: interaction diagram

- Sequence Diagram example



:Document

:Fax

:TelephoneLine

call()

call()

invitedSound

compose(num)

compose

connected

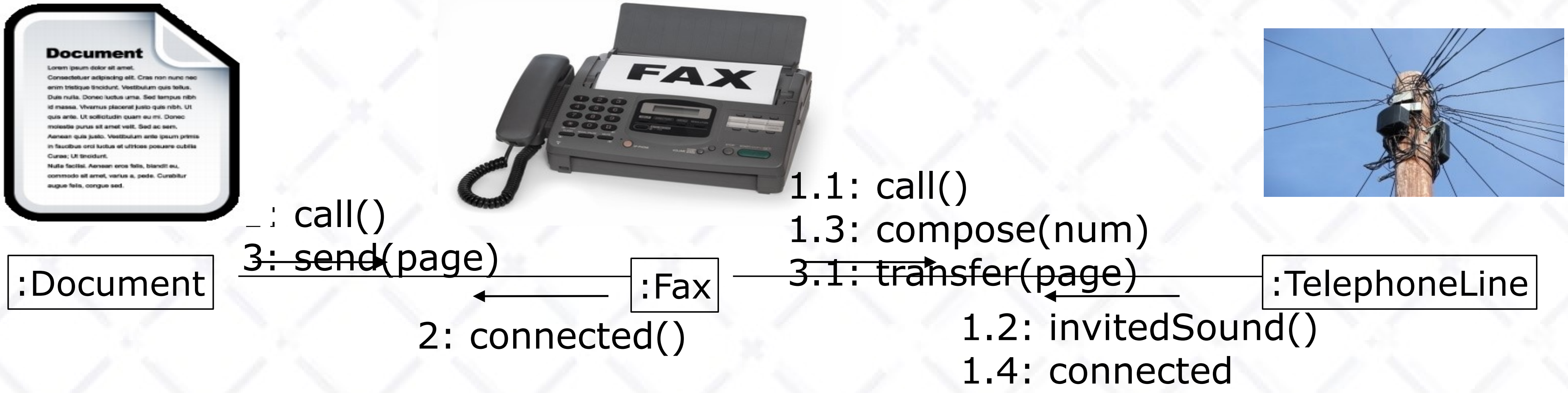
send(page)

transfer(page)

“Sending Fax” Sequence Diagram

Introduction to UML: interaction diagram

- Communication diagram example



:Document

1: call()
3: send(page)

:Fax

2: connected()

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

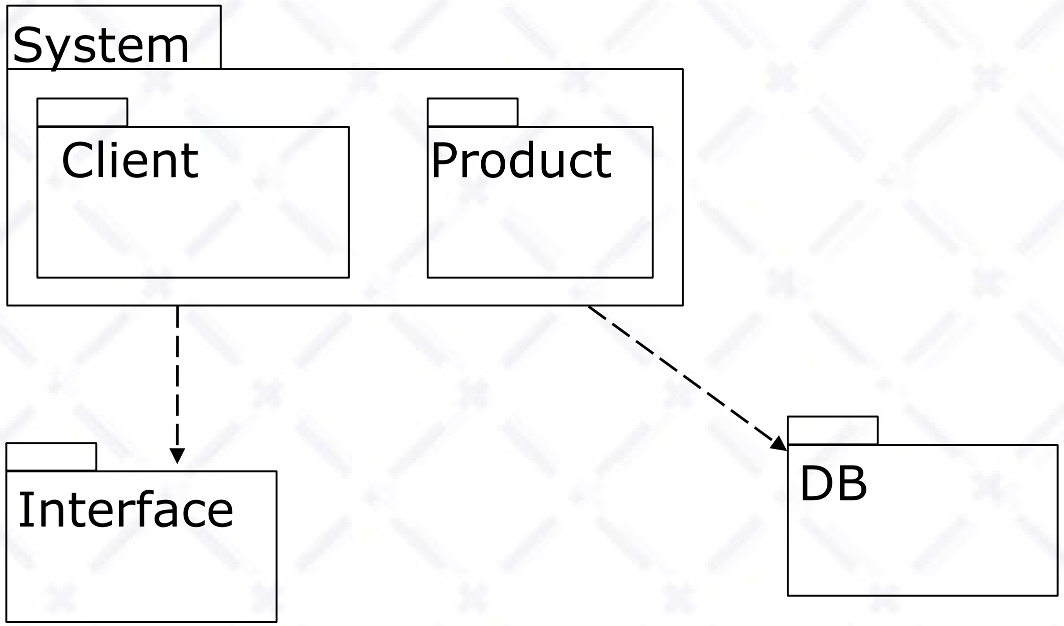
:TelephoneLine

1.2: invitedSound()
1.4: connected

“Sending Fax” Communication diagram

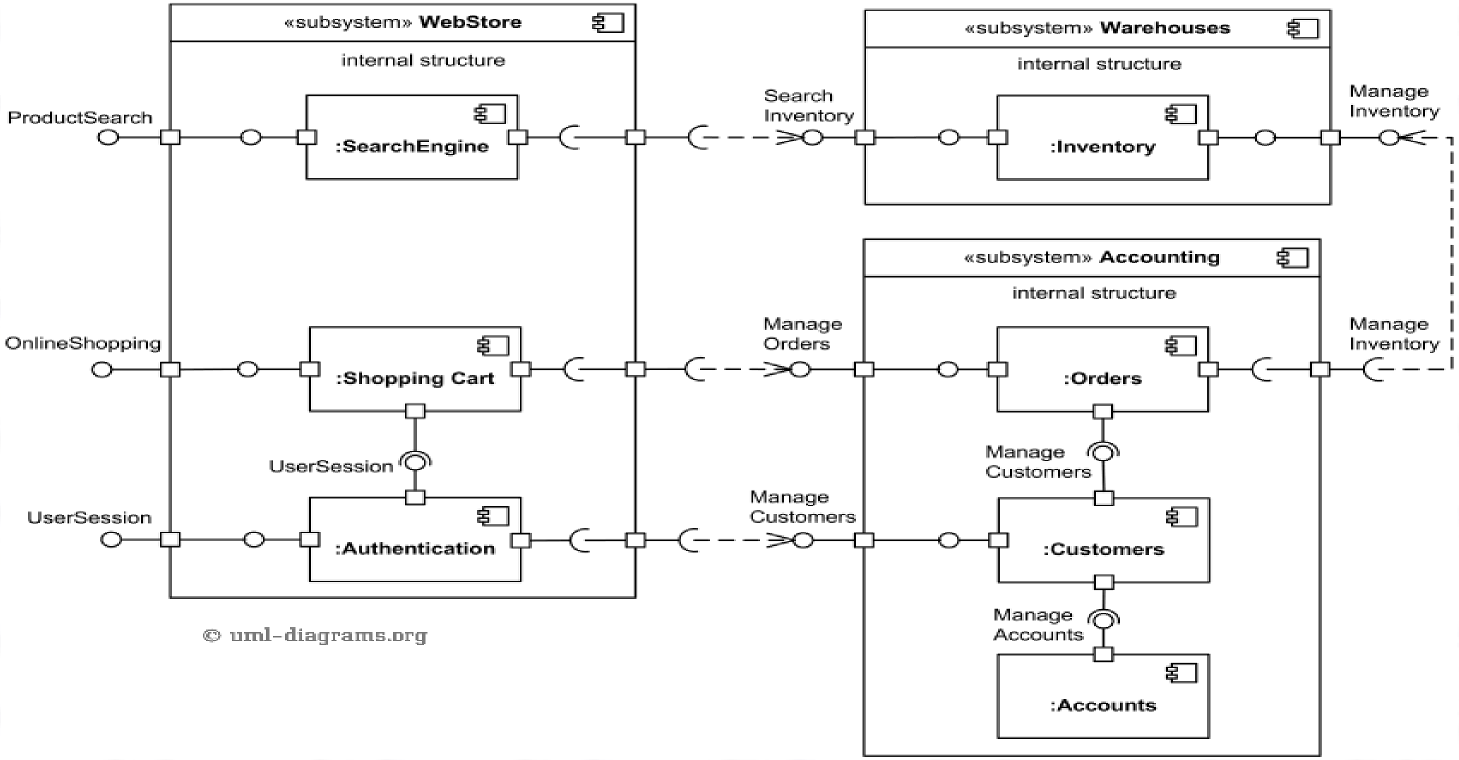
Introduction to UML: Package diagrams

- Packages
  - Universal element is used to organize classes, or some other group into a group
- Package diagrams
  - Displays packages and the relationships between packages.
  - Example



Introduction to UML: component diagrams

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

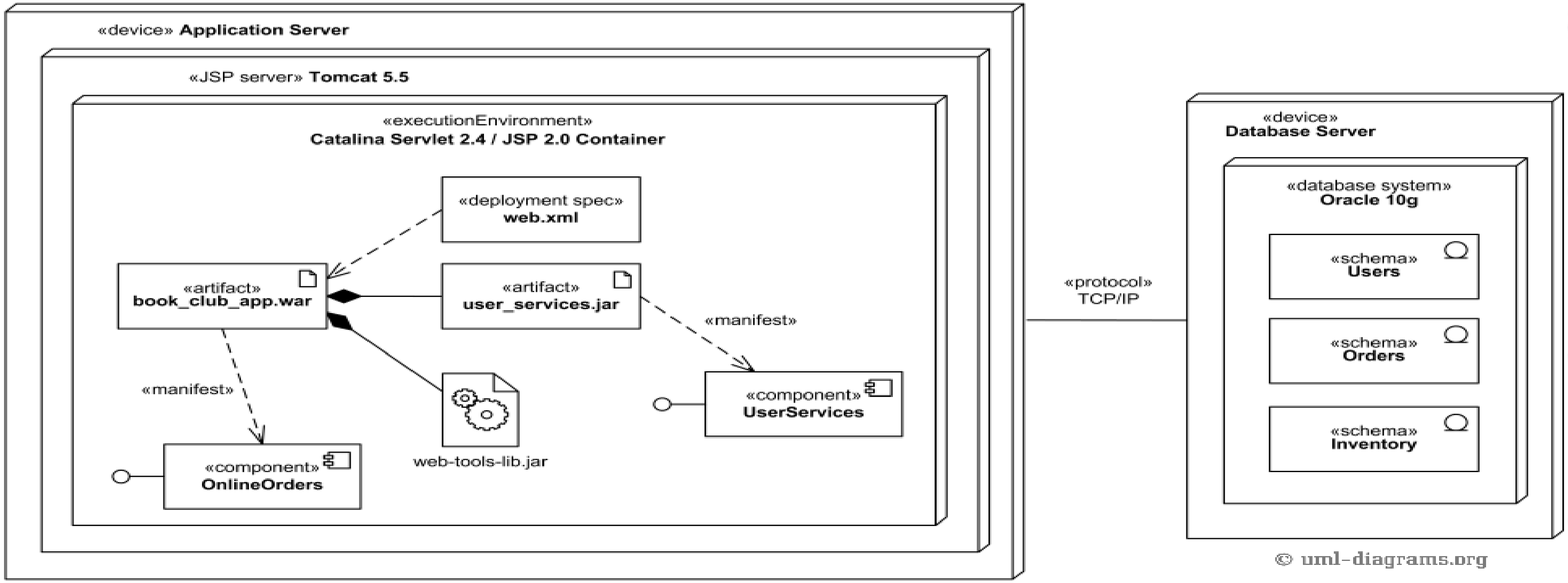


© uml-diagrams.org

“Online Shopping Website” Component Diagram

Introduction to UML: deployment diagrams

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



© uml-diagrams.org

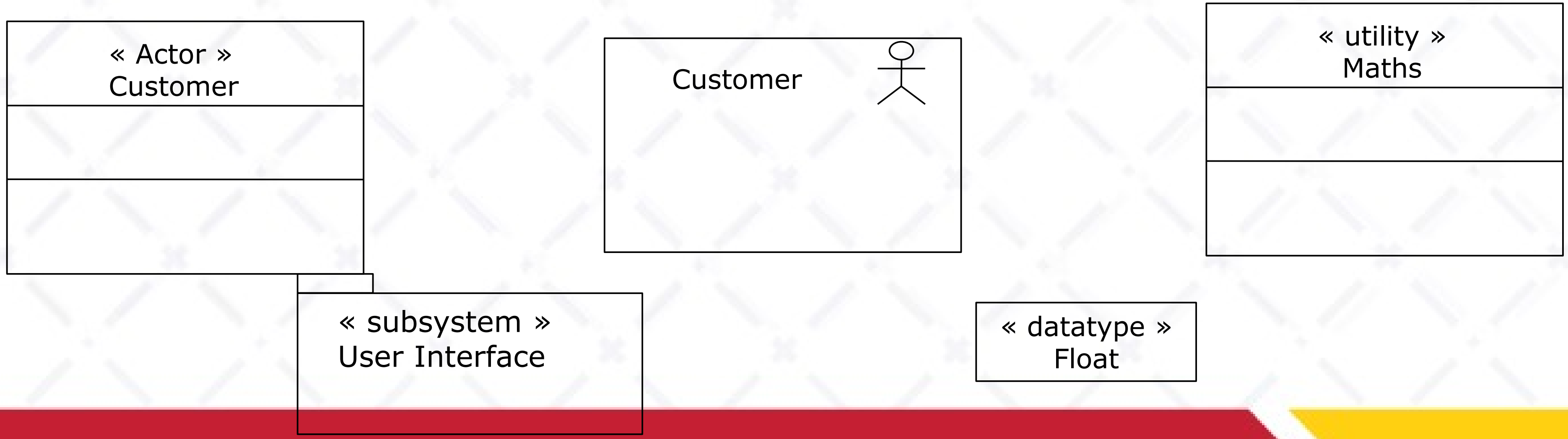
An example of deployment diagram of JEE web application

Introduction to UML: extension mechanism

- Built-in extension mechanism
  - Stereotypes
  - Tagged values
- Notes
- Constraints
  - OCL textual language

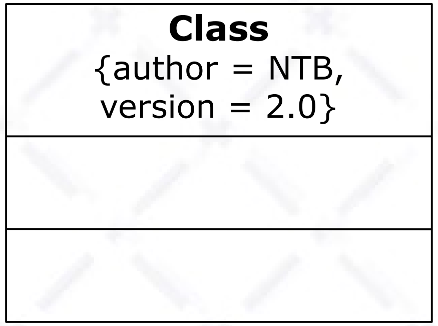
Introduction to UML: general mechanisms

- 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



Introduction to UML: general mechanisms

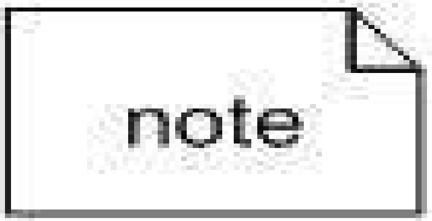
- Tagged values
  - Another extension mechanism
  - Provide additional information on the elements of UML
  - Pairs of type {name = value}
  - Example

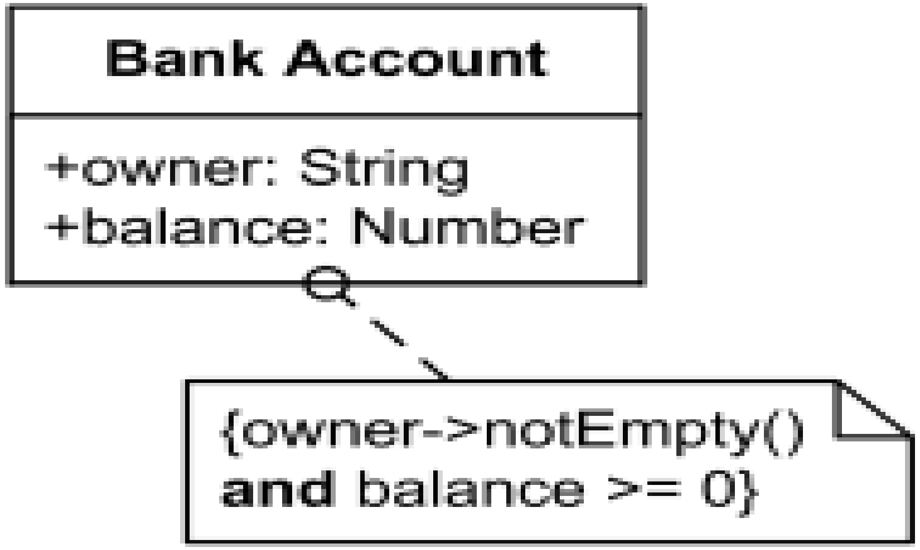


Class
{author = NTB,
version = 2.0}

Introduction to UML: general mechanisms

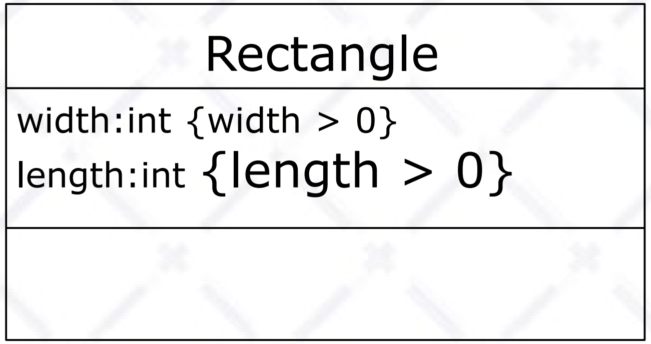
- Notes
  - are comments attached to one or more modeling elements
  - provide additional information on modeling elements
  - belong to the view, not the models

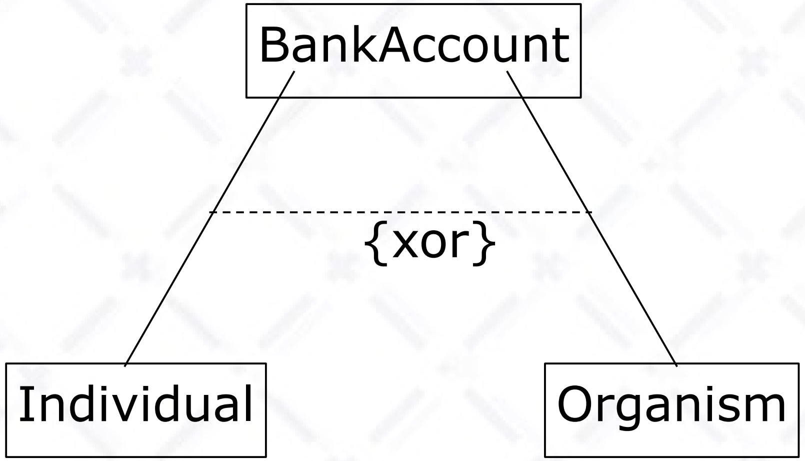




Introduction to UML: general mechanisms

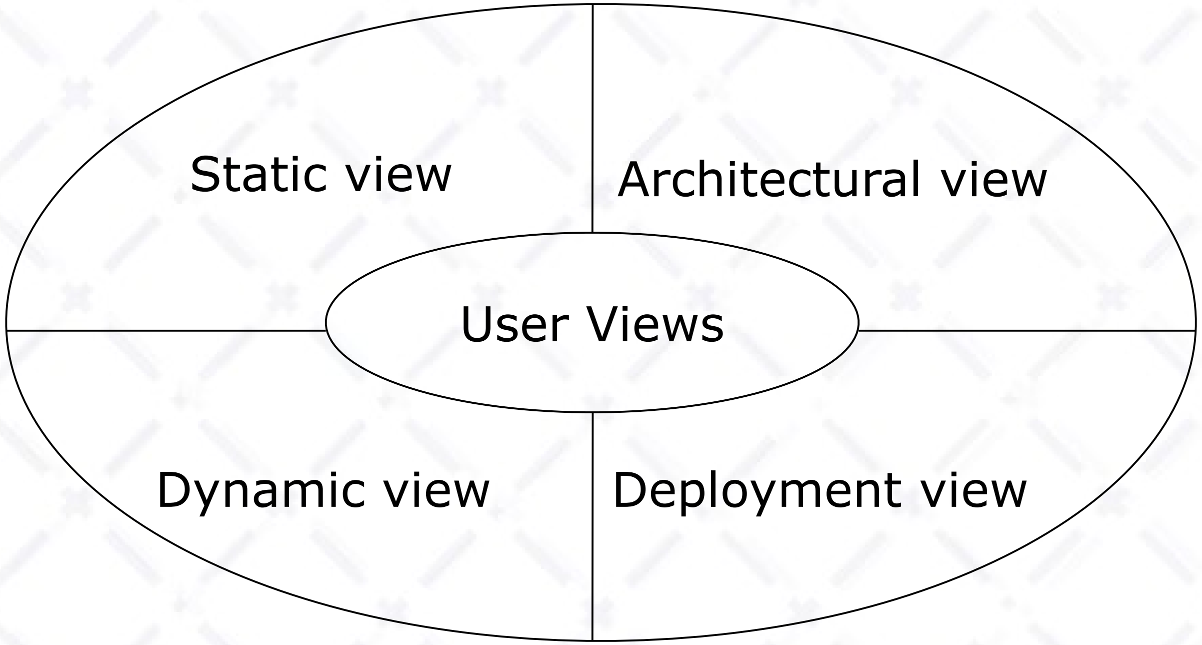
- Constraints
  - are restrictions that limit the use of an element or the element semantic
  - are expressed in natural language
  - are expressed in OCL (Object Constraint Language)
  - Example





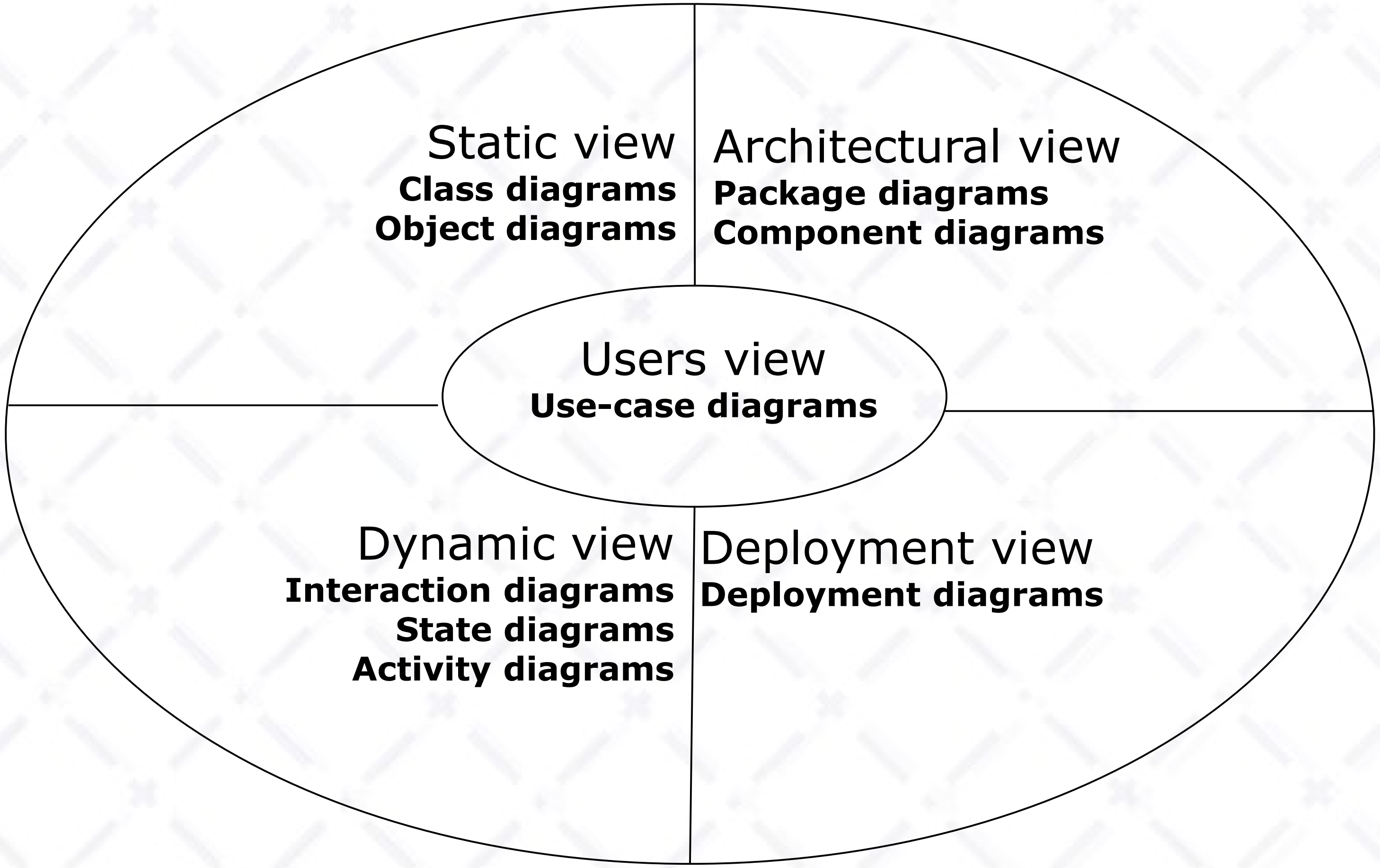
Introduction to UML: views

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



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