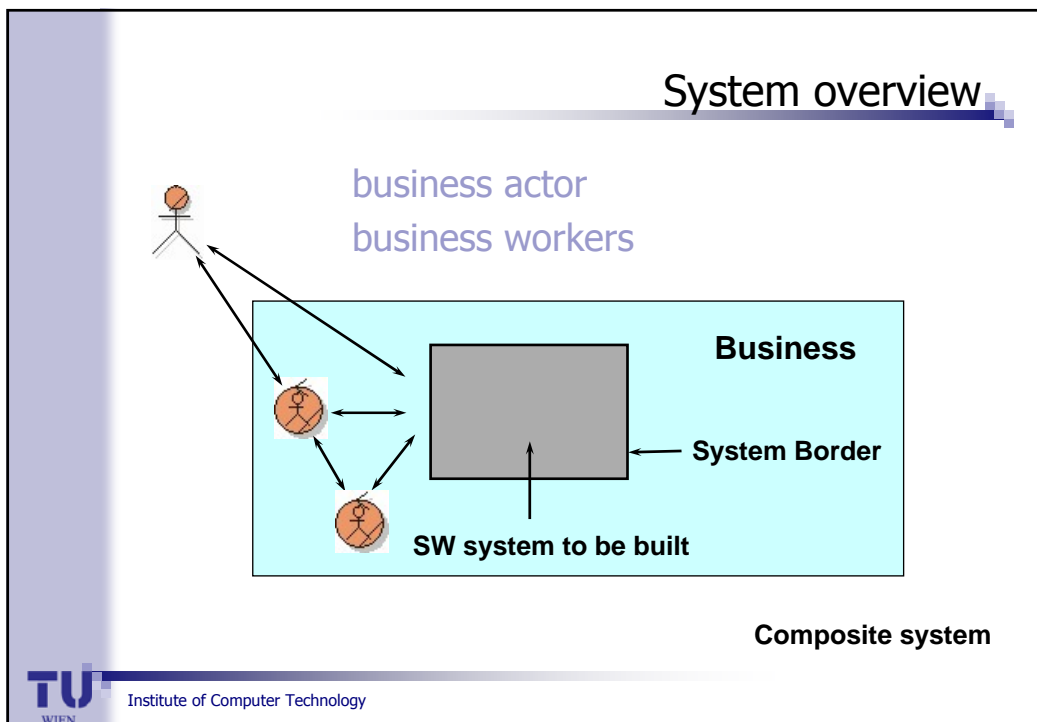


TU
WIEN

Model-based Transition from Requirements to High-level Software Design

Institut für Computertechnik
ICT
Institute of Computer Technology

Hermann Kaindl
Vienna University of Technology, ICT
Austria




Outline

- Background
- Functional requirements, goals and scenarios / use cases
- Requirements and UML models
- Transition to software design
- Model-based transformation?



What are requirements?

- User wishes / needs 
- *IEEE Standard:*
"A condition or capacity needed by a user to solve a problem or achieve an objective."
- "The <system> shall be able to ..."
 - system to be built
 - composite system
- *Example:* "The ATM shall accept a cash card."
- Requirements modeling



What are requirements? – In practice

- User requirements documents
- Software/system requirements documents
- Mostly descriptions in **natural language**
- Representation often unstructured
- Ad hoc process
- Communication problem
- Requirements and **use cases?**

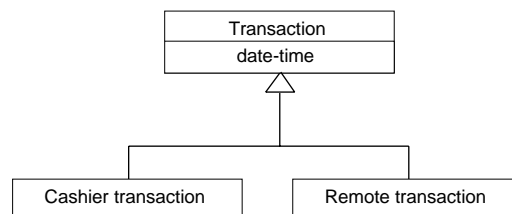


Class and generalization

Class in **UML** (Unified Modeling Language)
<http://www.omg.org>

Generalization / Specialization
(in UML notation)

Object – **instance**



Inheritance

Example: **attribute** date-time

Mechanism for information sharing

- Structure (variables, attributes)
- Behavior (methods, procedures)

Multiple inheritance

various theories

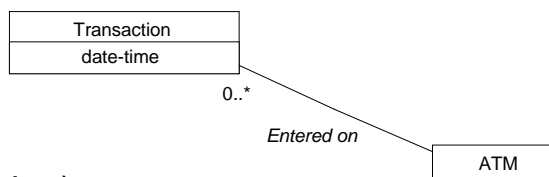


Institute of Computer Technology



Attributes and associations

Attribute for representation of property



Association
(in UML notation)

Relation for linking instances

Multiplicity: range of allowable cardinalities



Institute of Computer Technology

Methods and messages

Methods are procedures / functions

Interface protocol for indirect calls

Sending and receiving of **messages**

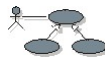
Actual method to be invoked determined through rules for processing a message, e.g., "dynamic binding"



Institute of Computer Technology

Use cases

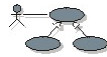
- "particular cases of how the system is to be used"
- Use-Case Report (according to Unified Process):
 1. Brief Description
 2. Flow of Events
 3. Special Requirements
 4. Pre-conditions
 5. Post-conditions
 6. Extension Points
 7. Relationships
 8. Use-Case Diagrams
 9. Other Diagrams



Institute of Computer Technology

Use-case diagram

- UML graphical notation



- Ellipse: use case



Name of use case

- Stick man: actor



Name of actor

- Connecting line: association



Institute of Computer Technology

Outline

- Background
- ■ Functional requirements, goals and scenarios / use cases
- Requirements and UML models
- Transition to software design
- Model-based transformation?



Institute of Computer Technology

Glossary

Functions: "effects achieved by some entity"

Goals: "partially specified states that the user considers as desirable"

Scenarios: "sequences of actions aimed at accomplishing some task goal"

Use cases: "particular cases of how the system is to be used", "classes of scenarios"



Institute of Computer Technology

Functional requirements

- Describe required functionality not yet available
- Functional user requirements may be high-level statements of what the system should be able to do.
- Functional software/system requirements should describe the functions of the software/system to be built in detail (but not yet its design or implementation).



Institute of Computer Technology

Scenarios – Stories and narratives

- For representation of
 - cultural heritage
 - explanations of events
 - everyday knowledge
- Human understanding in terms of specific situations
- Human verbal interactions by exchanging stories



Institute of Computer Technology

Scenarios – Video Store example

Rent Available Video:

1. A member of the video store identifies himself/herself to VSS (Video Store Software).
2. VSS shall check the identification.
3. If the identification is successful, VSS shall start a transaction and present a selection of video titles.
4. The member selects a video title that is available and indicates the intent to rent (a copy of) the video.



Institute of Computer Technology

Scenario – Video Store example (cont.)

5. VSS shall book this rental on the account of the member and ask the clerk to hand out a video copy to the member.
6. The clerk hands out a copy of the video title and acknowledges this to VSS.
7. VSS shall again present a selection of video titles.
8. The member does not select further titles, but initiates the termination of the transaction.
9. VSS shall issue a good-bye message and terminate the transaction.



Institute of Computer Technology

By-Function – Video Store example

1. A member of the video store identifies himself/herself to VSS (Video Store Software).

2. VSS shall check the identification.

By-Function: Member Identification Check

...

5. VSS shall book this rental on the account of the member and ask the clerk to hand out a video copy to the member.

By-Function: Video Rental Booking
Video Handing-out Request

...



Institute of Computer Technology

Functional requ. – Video Store example

Rent Available Video *By-Function* *Video Rental Booking*

Video Rental Booking:

VSS shall book the rental of a copy of a video title on the account of the member, and reduce the number of available copies of the video title by 1.



Institute of Computer Technology

Goal – Video Store example

Member Has Video for Rent *By-Scenario* *Rent Available Video*

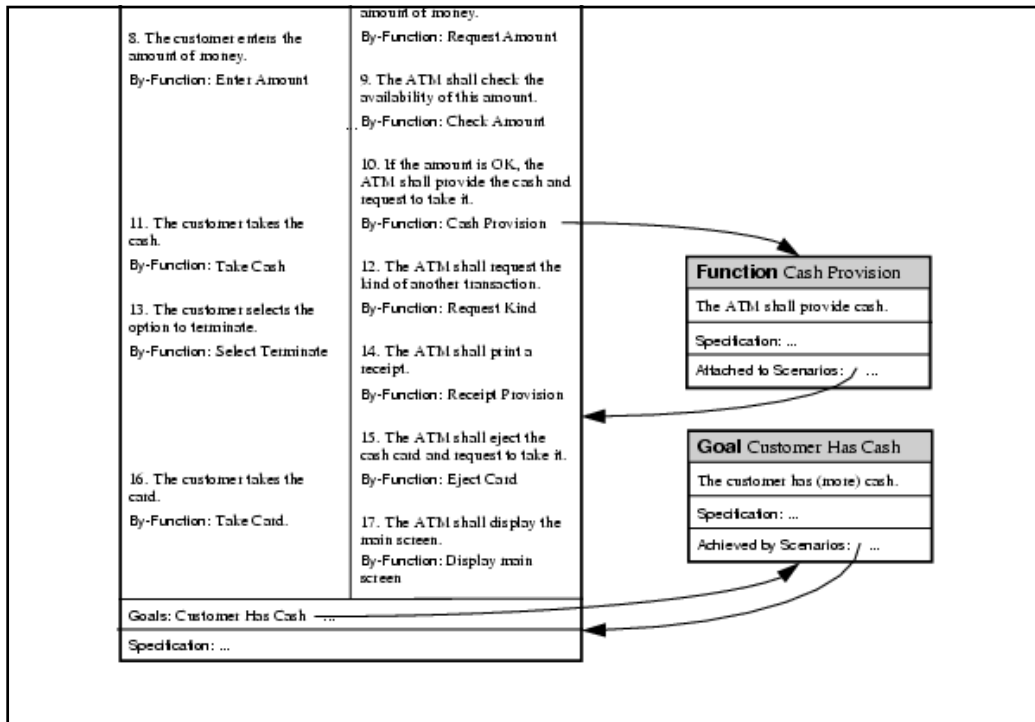
Member Has Video for Rent:

A member of the video store has a copy of a video title from the store for rent.



Institute of Computer Technology

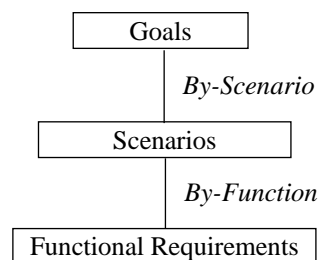
Model-based Transition from Requirements to High-level Software Design



Systematic process

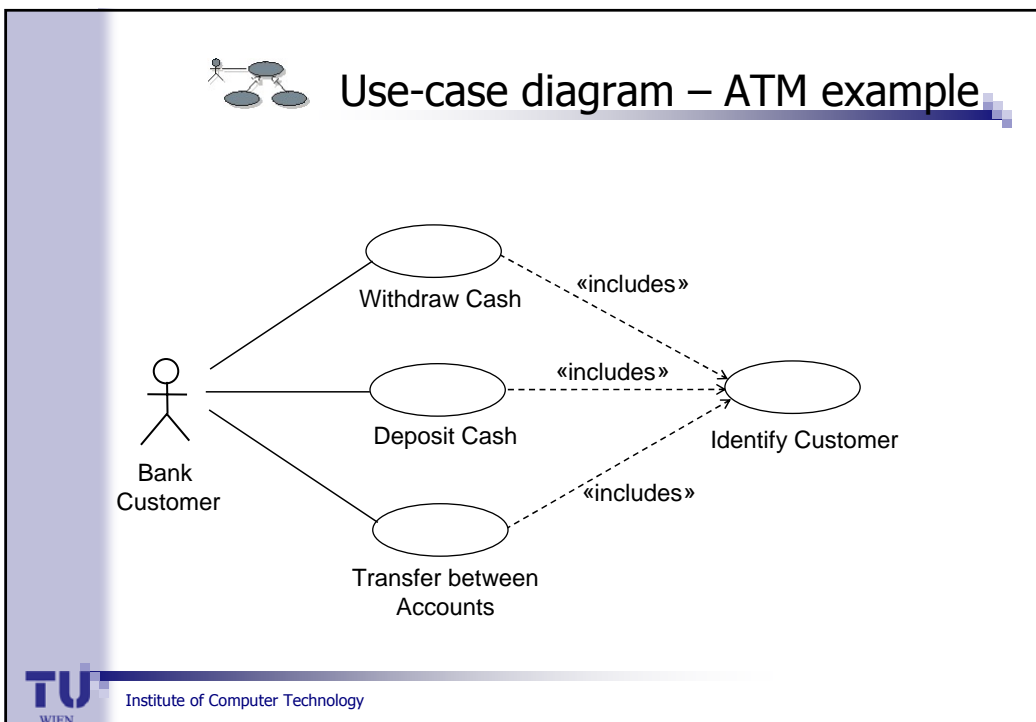
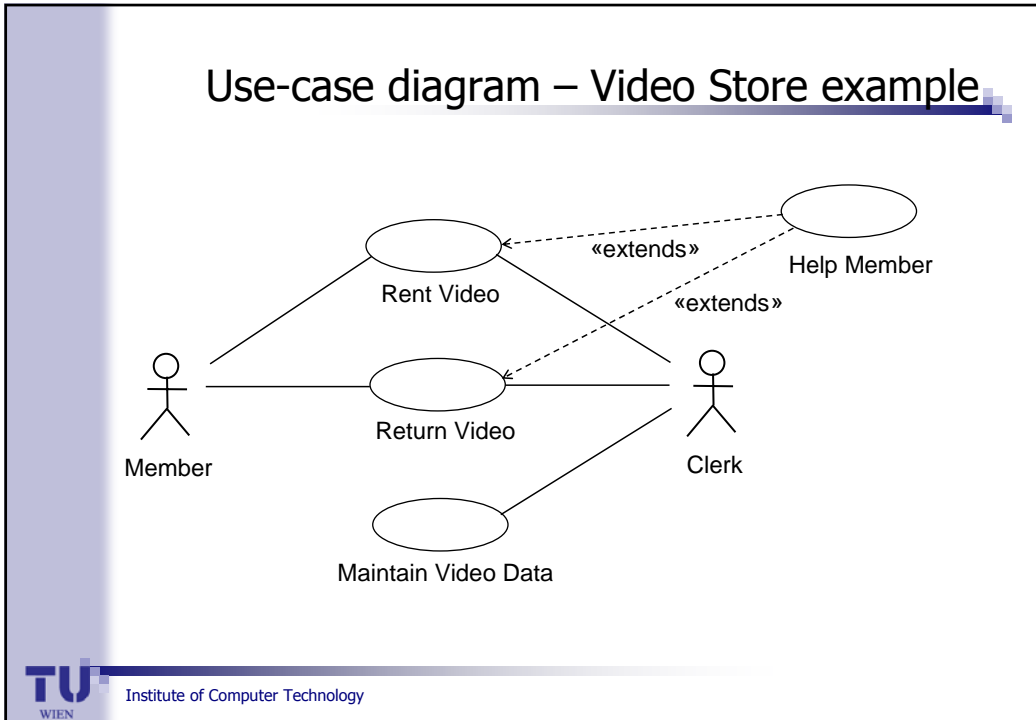
Idea: navigation in the metamodel graph

Excerpt:



What is known already?

Old system or system to be built?

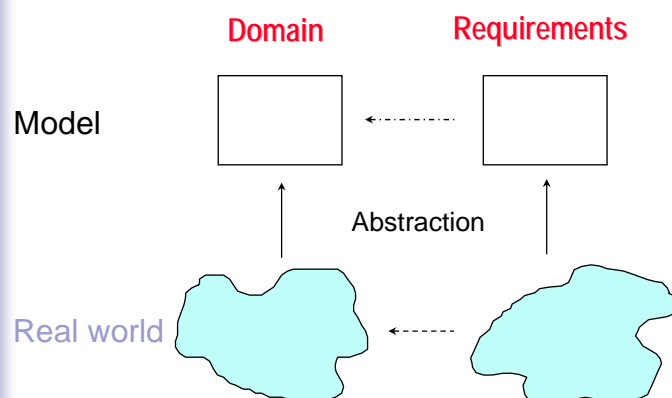


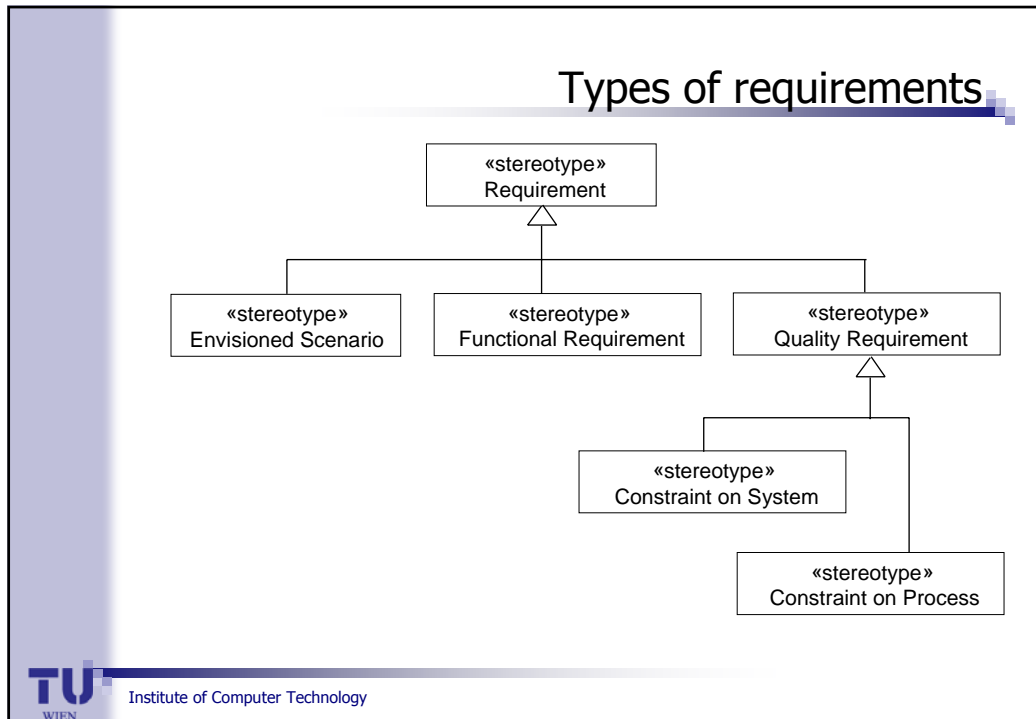
Outline

- Background
- Functional requirements, goals and scenarios / use cases
- ➔ ■ Requirements and UML models
- Transition to software design
- Model-based transformation?



Requirements and object-oriented models





- ### Types of requ. – Constraints on system
- Performance
 - Reliability
 - Security
 - Safety
 - Portability
 - Maintainability
 - Reusability
 - Interface
 - Usability
- TU WIEN Institute of Computer Technology

Types of requ. – Constraints on process

- Specific development process to follow
- Specific programming language for implementation
- Specific tools to be used
- Specific hardware to be used
- Political issues
- Time to market
- Terms of delivery
- Cost



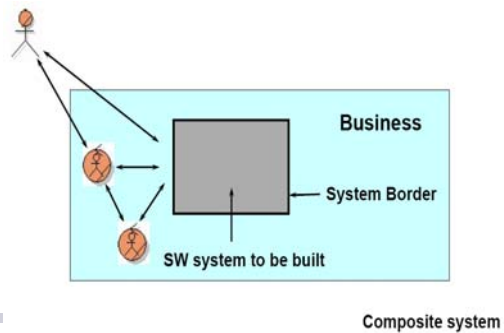
Conflicts between Quality Requirements

- VSS example
 - VSS shall allow direct access to member data.
 - VSS shall protect member data from illegal access.
- Usability vs. Security
- Trade-off
- Common in complex systems

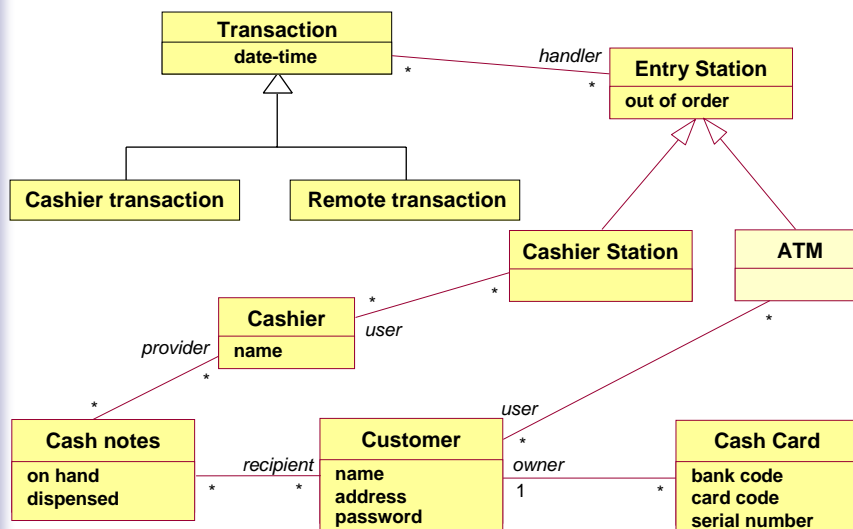


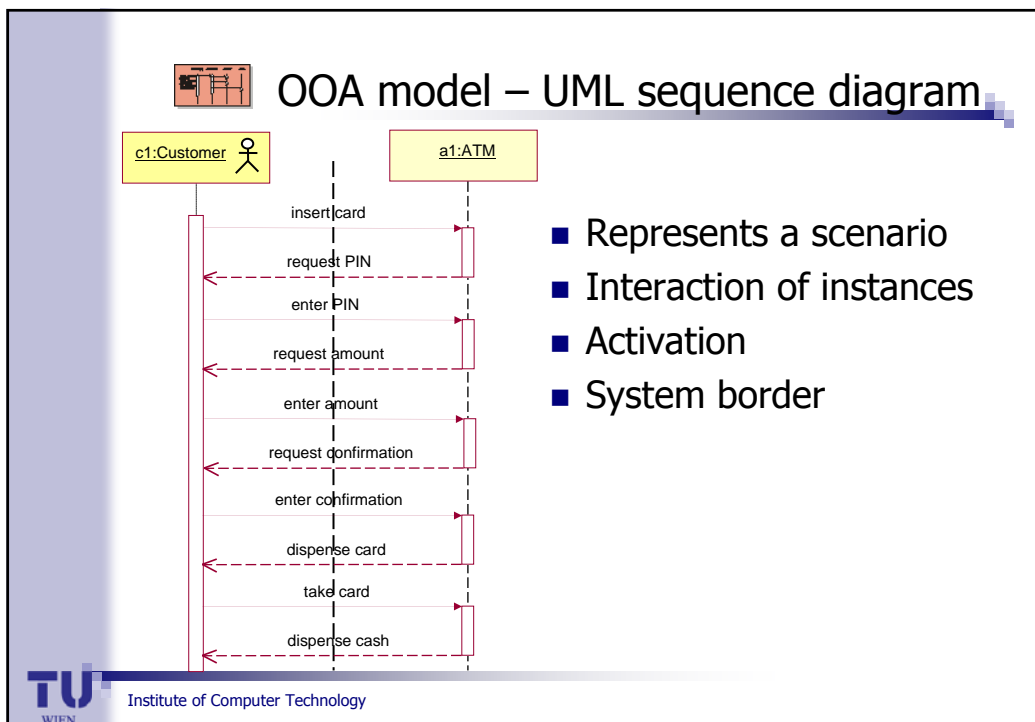
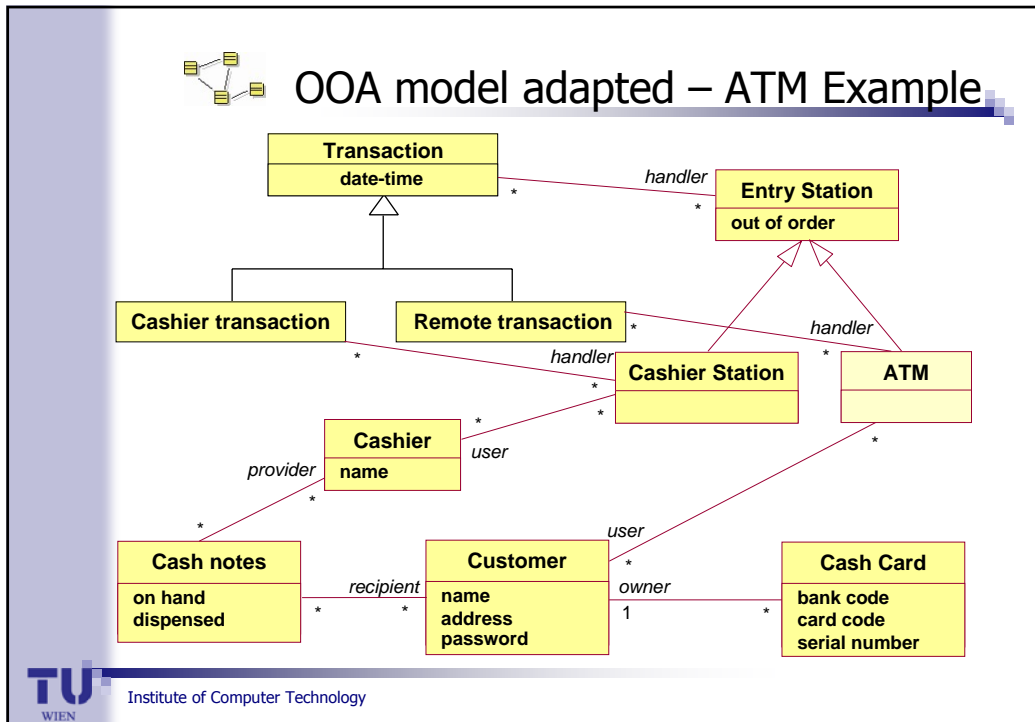
Types of requ. – Which “system”?

- Requirements for proposed **system to be built**
Software (and hardware) system
- Requirements for **composite system**
Including human users and business artifacts




OOA model – ATM Example







OOA model – RUP

- **Analysis Model:**
“An object model describing the realization of use cases, and which serves as an abstraction of the Artifact: Design Model. The Analysis Model contains the results of use case analysis, instances of the Artifact: Analysis Class.” 
- “Analysis classes represent an early conceptual model for ‘things in the system which have responsibilities and behavior’.”



OOA model – UP Larman

- “The **Analysis Model** is perhaps not ideally named, as it is actually a kind of design model. In conventional usage, ..., an analysis model suggested essentially a domain object model—an investigation and description of domain concepts. But the UP “Analysis Model” is an early version of the UP Design Model—it describes collaborating software objects with responsibilities.”



OOA model – SEM

■ Purpose

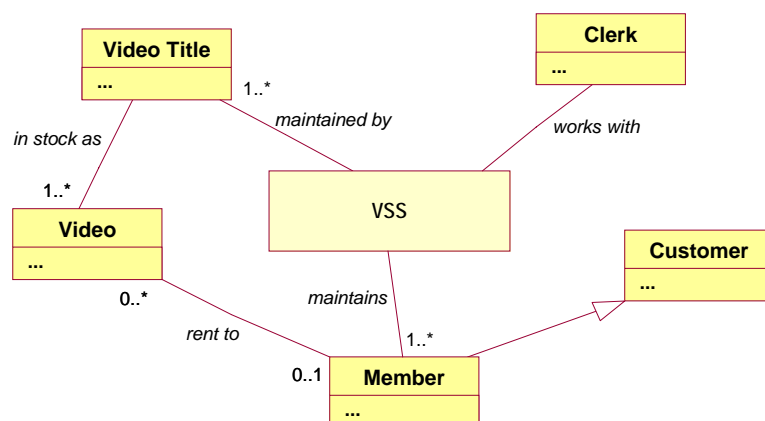
“An OOA model facilitates a better understanding of the application domain as it shall be after the implementation of the product. In this sense, it shall contribute to the specification of the problem and thus of the requirements.”

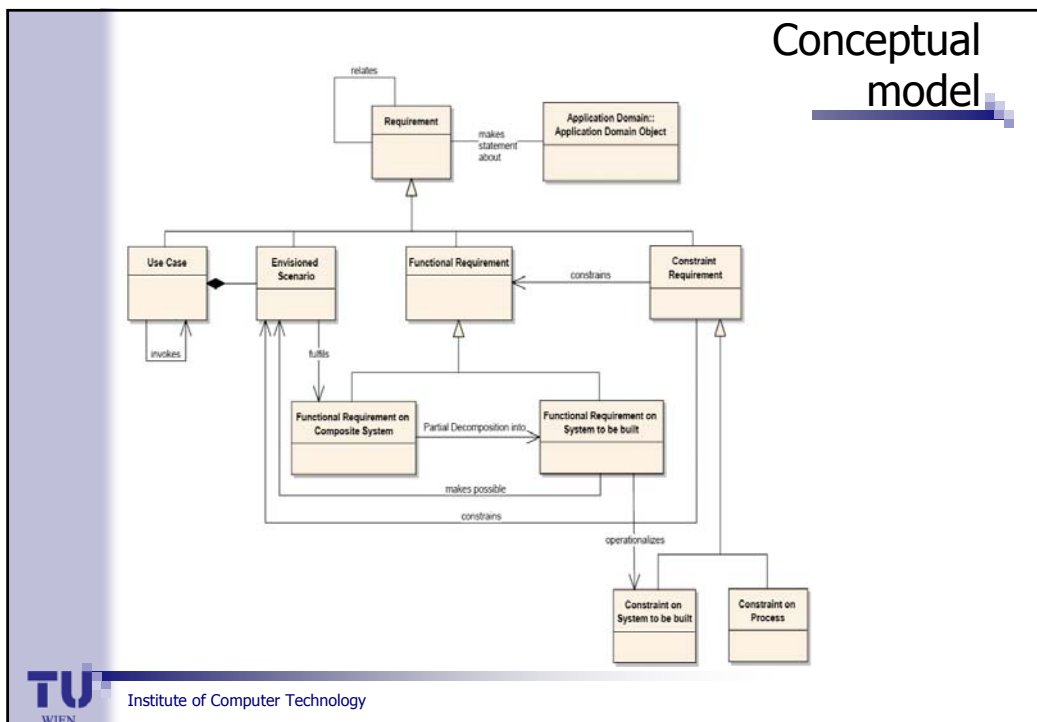
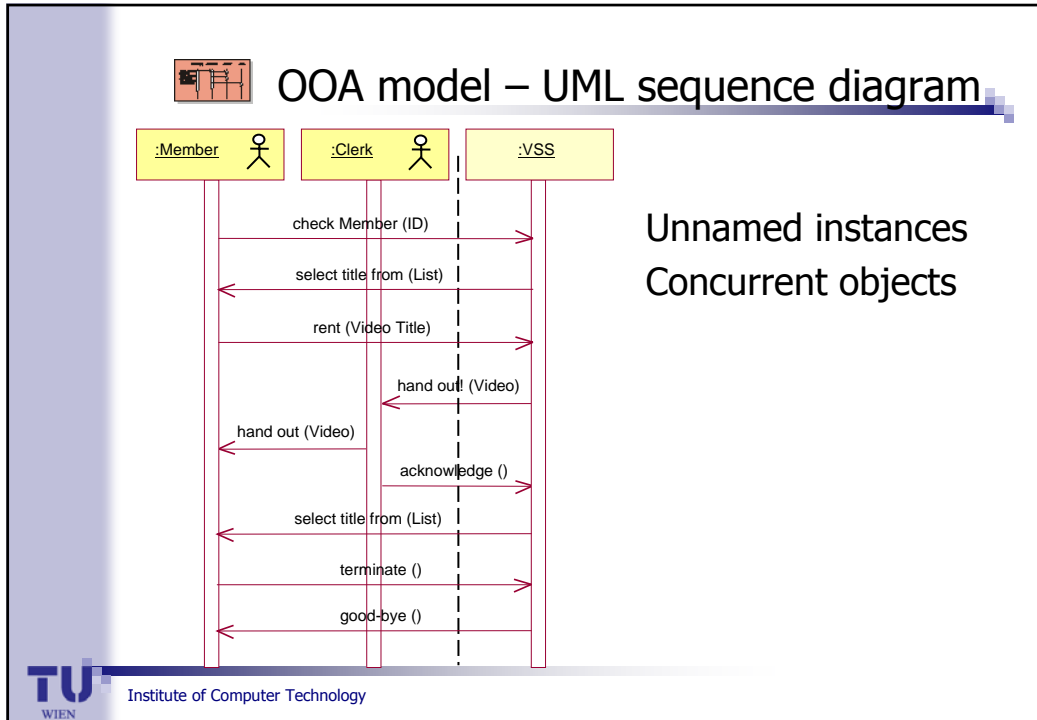
■ Content

“In an OOA model the to-be situation of the real world is modeled, where the product to be built will be integrated.”



OOA model – Video Store example





Further development

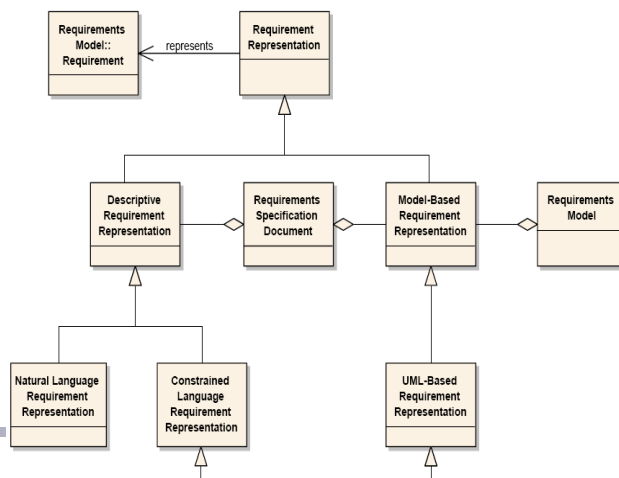
- European Union **ReDSeeDS** project
 - Requirements Driven Software Development System
 - Contract number IST-2006-033596
 - www.redseeds.eu
- Scenario-driven development method
- Reuse and tool support
- Case-based approach



Institute of Computer Technology

Requirements vs. requirements representation

- Reuse of requirements **representation** only
- Distinction between
 - **descriptive** and
 - **model-based**
- **Descriptive:** need described
- **Model-based:** abstraction of what the system should look like



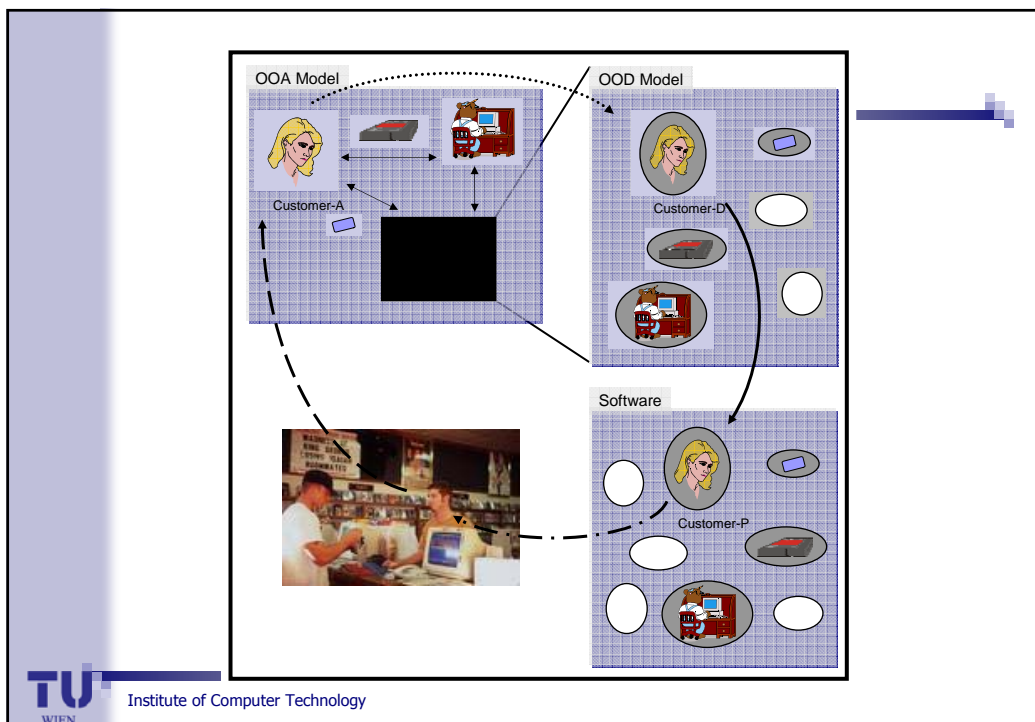
Institute of Computer Technology

Outline

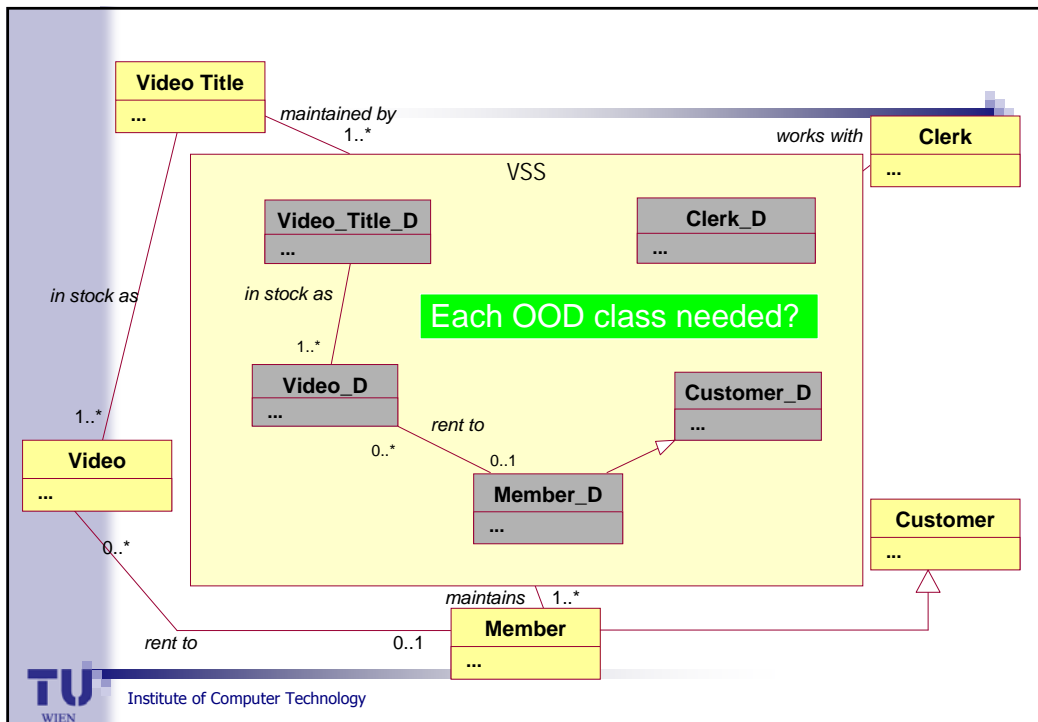
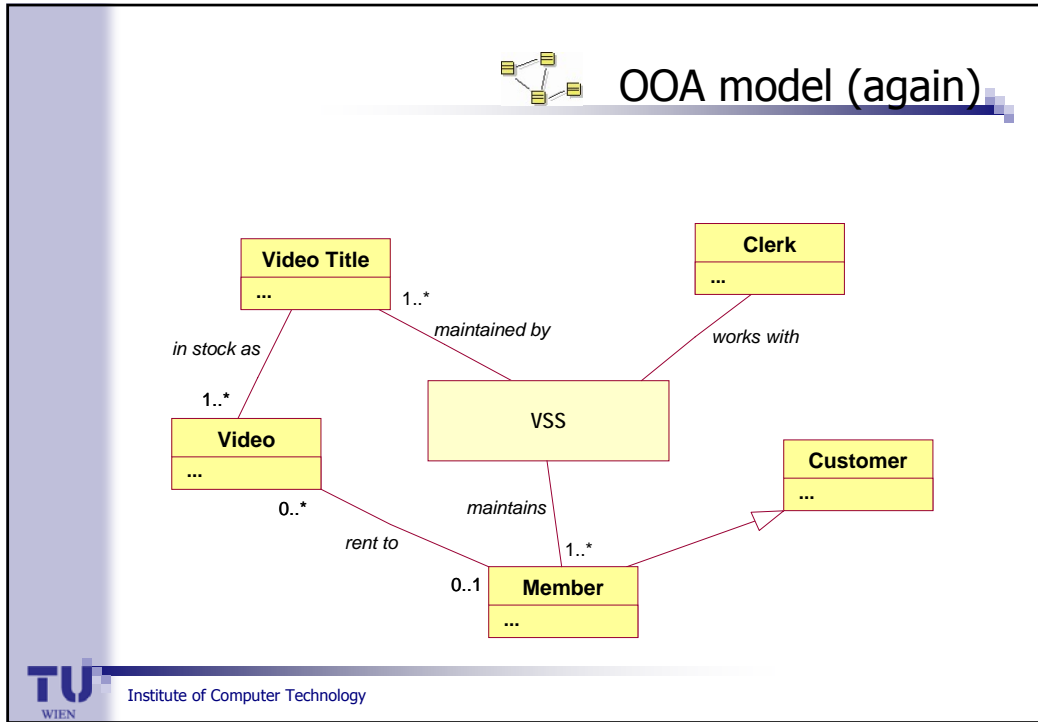
- Background
- Functional requirements, goals and scenarios / use cases
- Requirements and UML models
- ➔ ■ Transition to software design
- Model-based transformation?



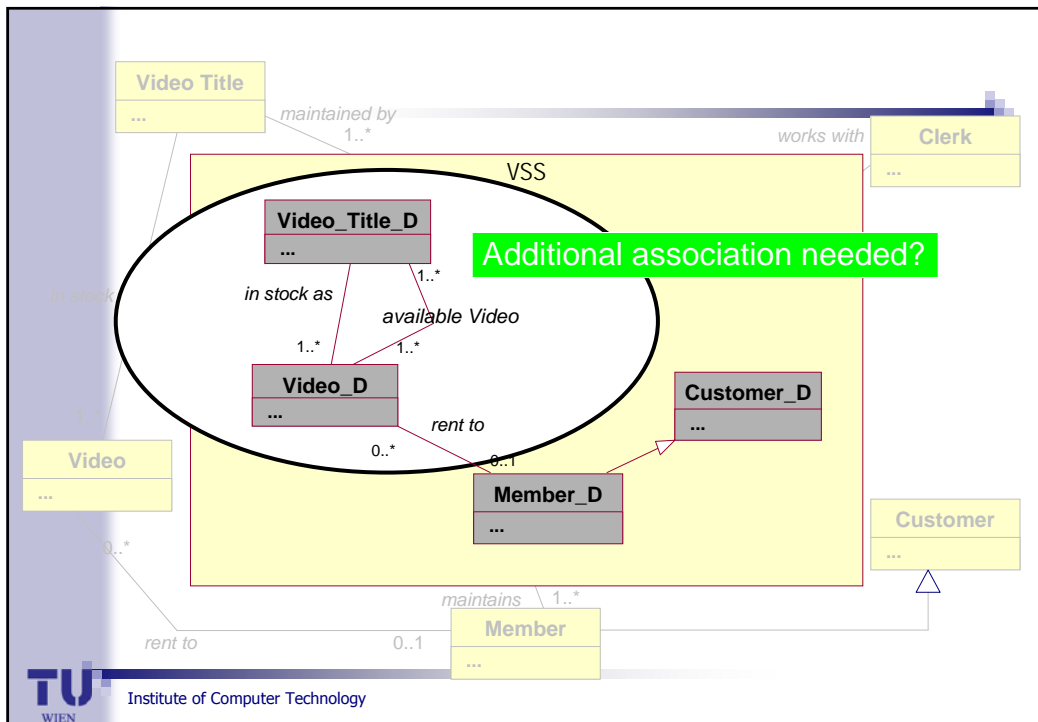
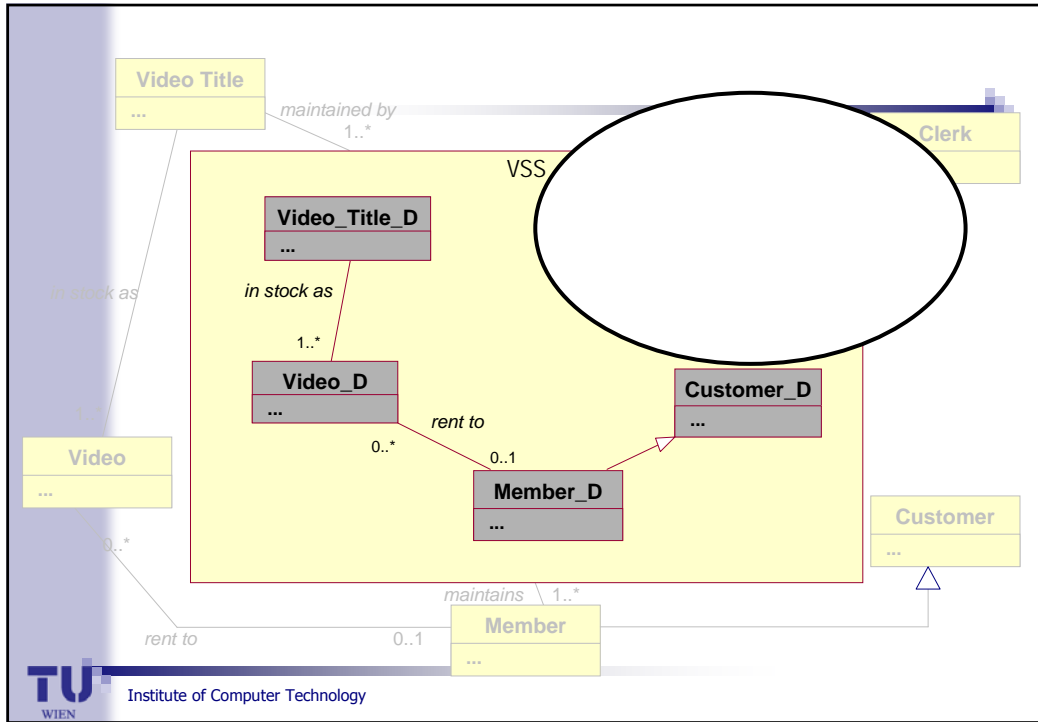
Institute of Computer Technology

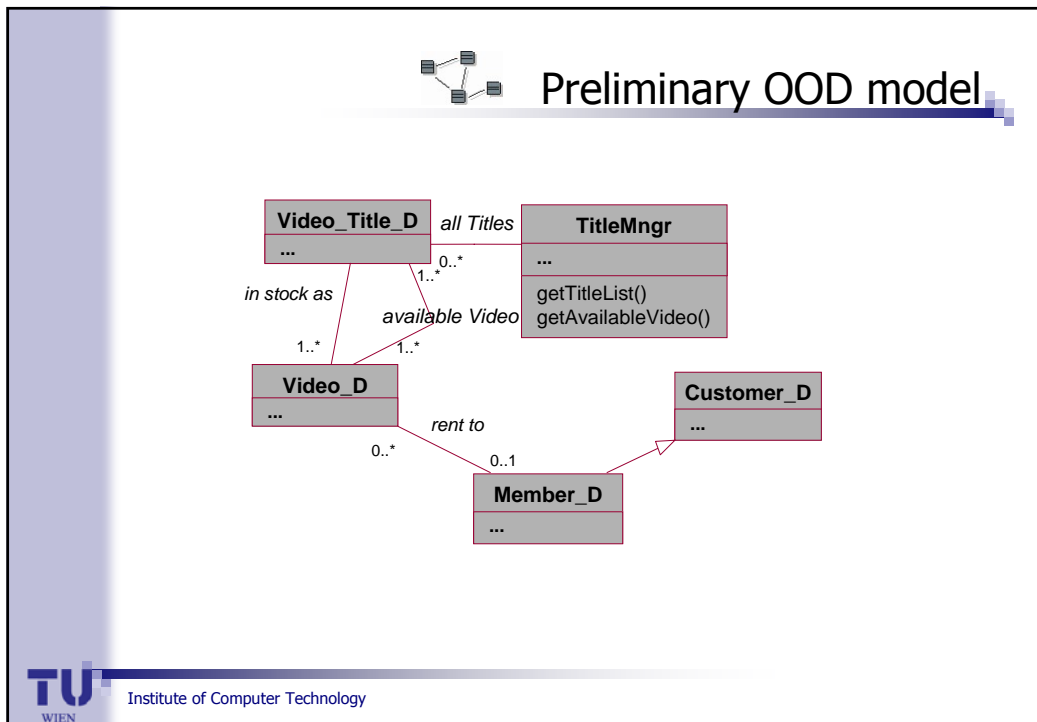
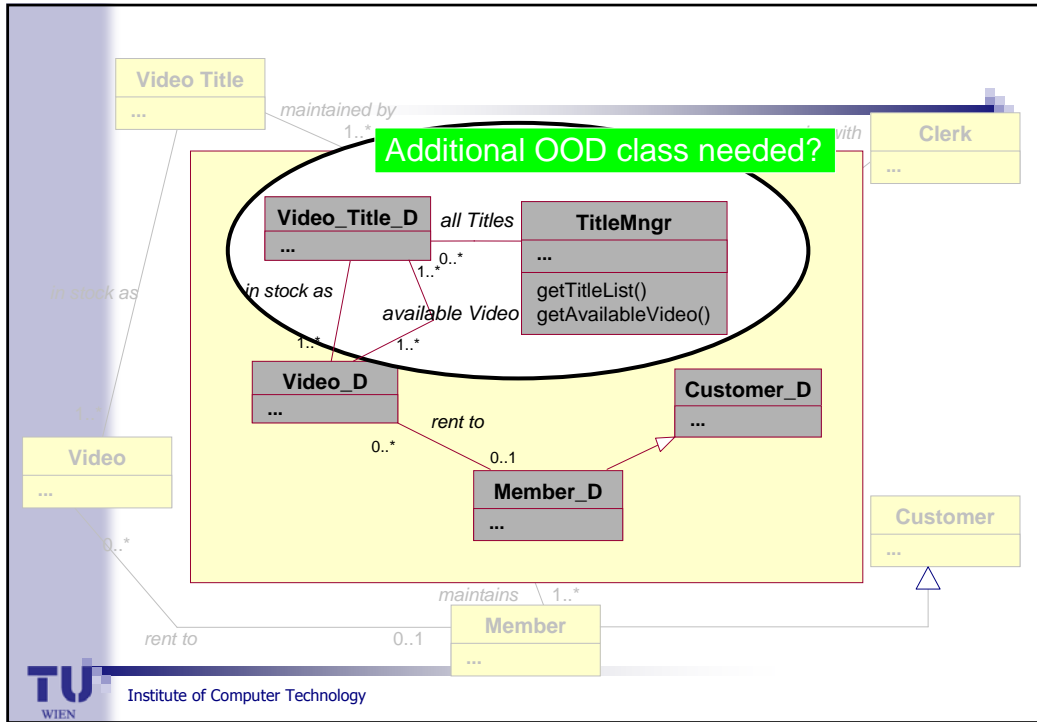


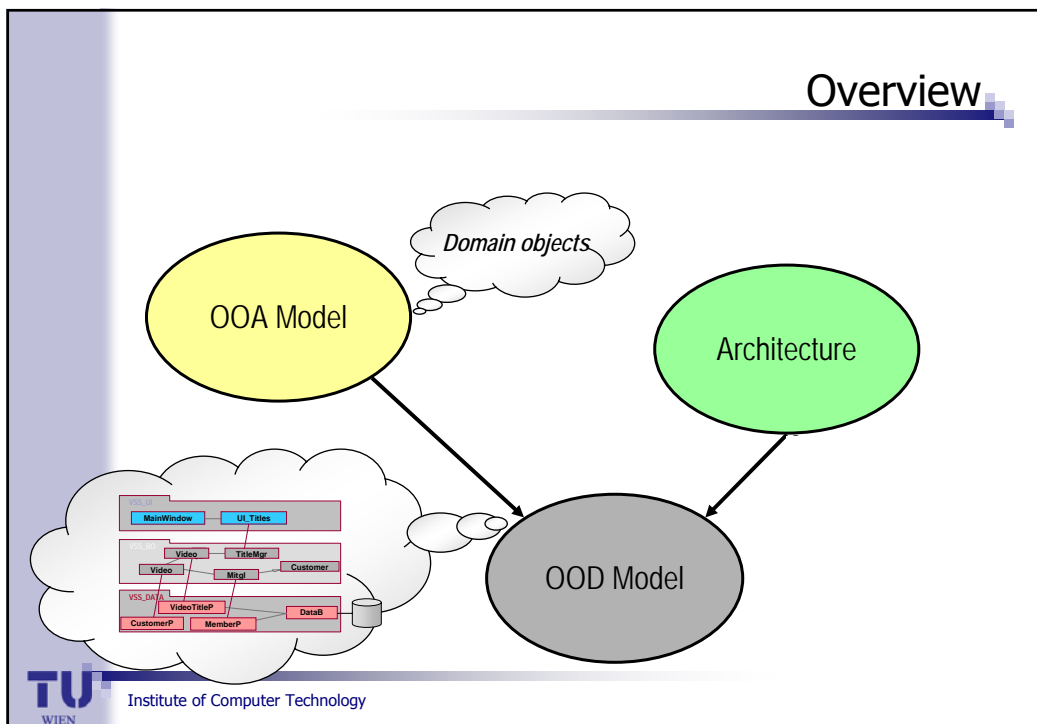
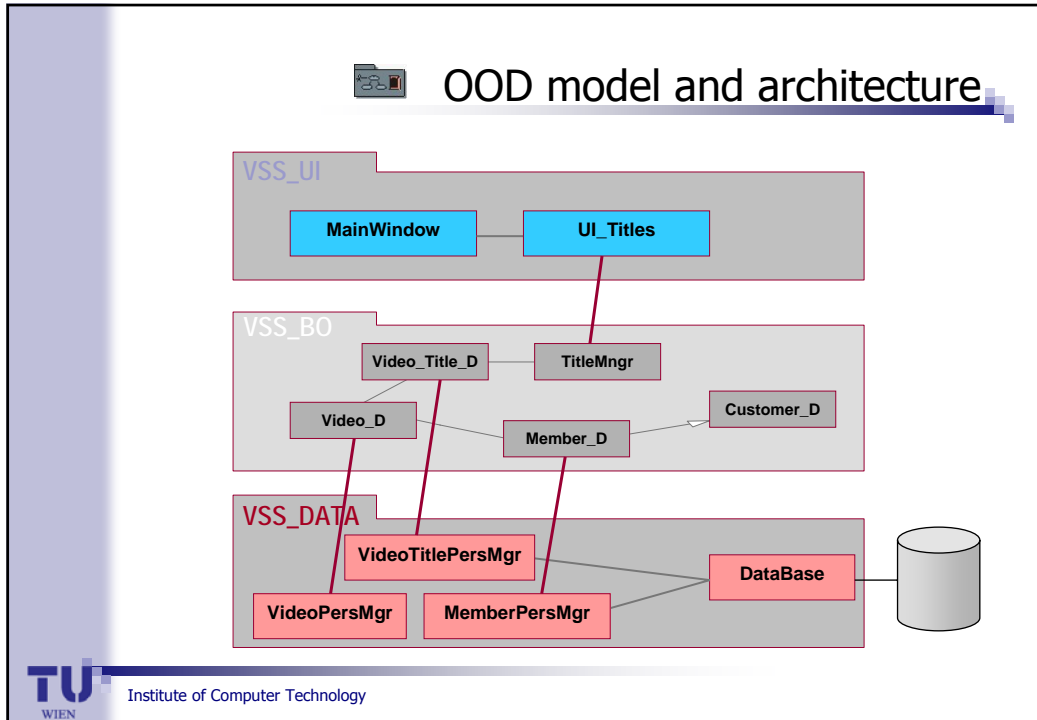
Institute of Computer Technology



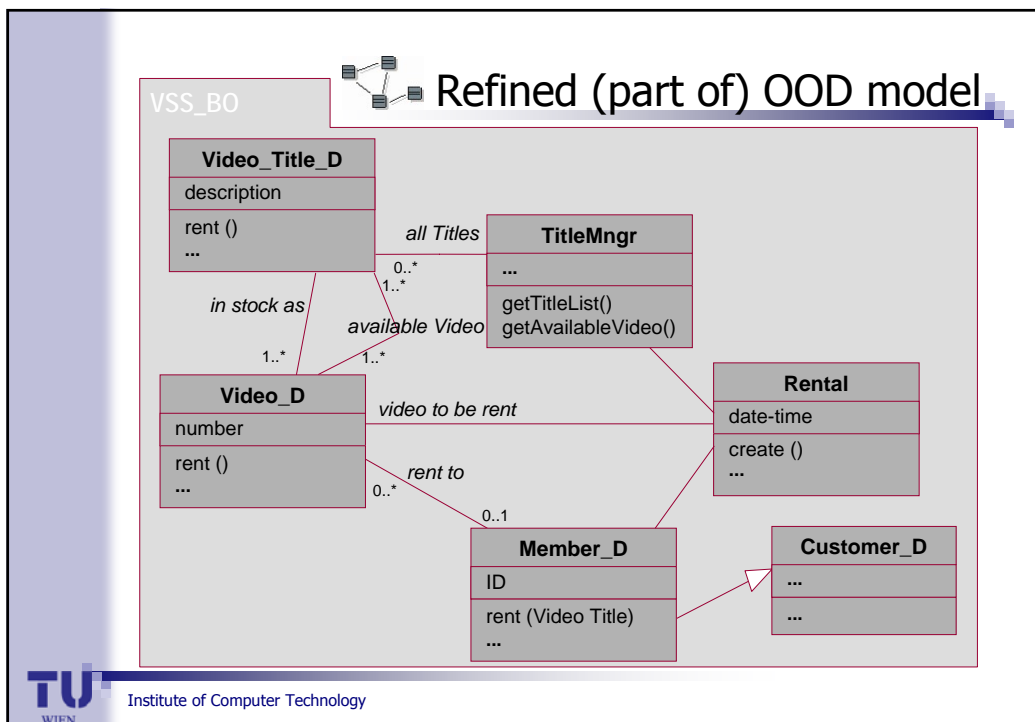
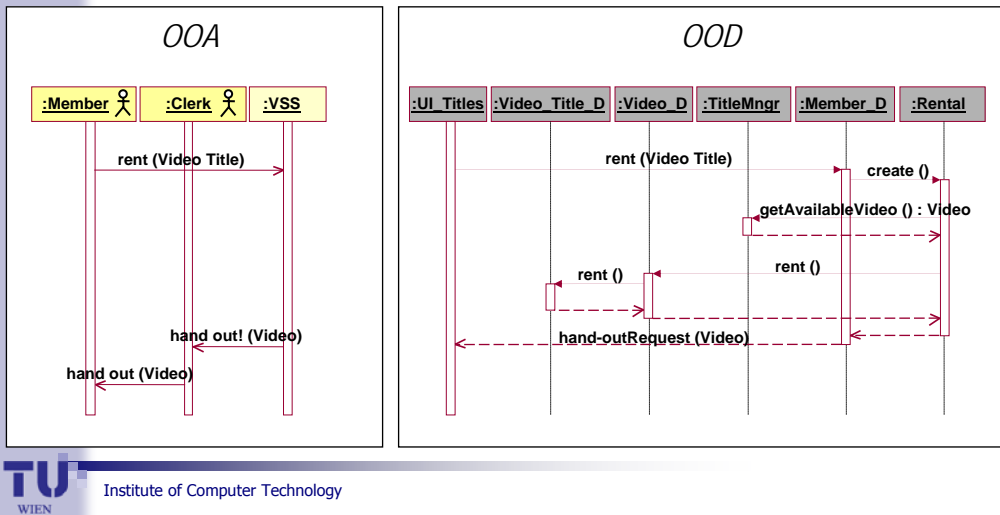
Model-based Transition from Requirements to High-level Software Design







Sequence diagrams – Video Store Example



OOD recommendations


- Imagine **zooming in** on the black box representing the proposed system in the OOA model.
- Develop an **architectural vision** of the proposed system.
- Will the proposed program need **information about real-world objects** (from the OOA model)?
- Will all of the OOA object's **attributes** be needed?
- Will **additional attributes** be needed?
- Will **additional associations** be needed?
- Define the **architecture** and "additional" **object classes**.
- Assign **responsibilities** to OOD objects.



Institute of Computer Technology



Software architecture

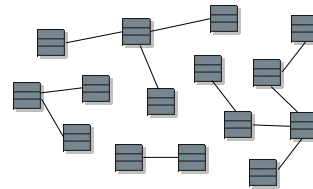
- *IEEE Standard Glossary of SW Engineering Terminology*
"The organizational structure of a system or component."
- *UML Specification (Glossary)* 
"The organizational structure and associated behavior of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems."



Institute of Computer Technology

Software architecture – Why?

- Granularity of objects too small
- High-level structure
- Big picture
- Strategic design decisions
- Divide-and-conquer: decomposition and modularization
- System-internal interfaces
- How to deal with constraints on system (requ.)



Selection depending on constraint requirements

- Performance
Large rather than fine-grain components for minimizing communication
- Security
Layered architecture with critical assets in the inner layers
- Safety
Localization of safety-critical features in a small number of sub-systems
- Availability
Redundant components and mechanisms for fault tolerance
- Maintainability
Fine-grain, replaceable components with low coupling

Outline

- Background
- Functional requirements, goals and scenarios / use cases
- Requirements and UML models
- Transition to software design
- ■ Model-based transformation?



Institute of Computer Technology

Differences between MD Transformation and Mapping

- *MD transformation:*
mathematical function that constructively and uniquely maps some input in one or more models to some output in one or more models
- *MD mapping:*
mathematical relation
- Transition from requirements to architecture?
- “Magic” problem solver?
- MD transformation possibly in hindsight, but not computable for most practical problems

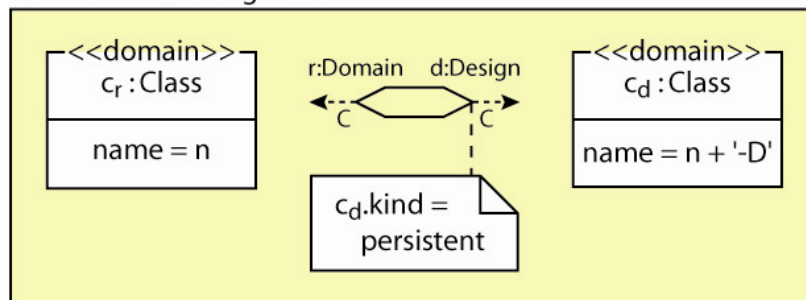


Institute of Computer Technology

From Domain Objects to Design Objects?

- MD mapping between domain classes and design classes by using the graphical notation of QVT

DomainClass2DesignClass

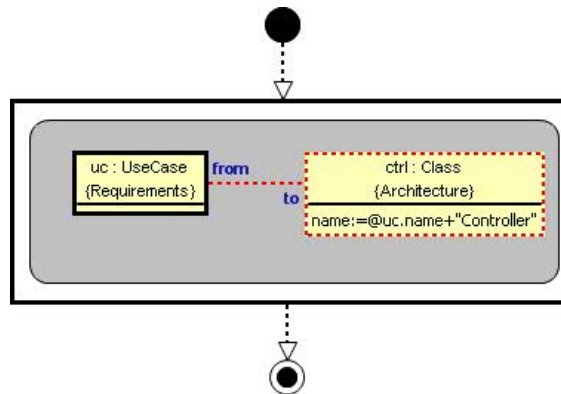


From Domain Objects to Design Objects? (cont.)

- Problem of universal quantification
- How about an MD transformation?
- Substitution of 'C' at the right through 'E' (for enforcement): Generates an object class in the OOD model for each object class in the OOA model.
- Manual modifications needed

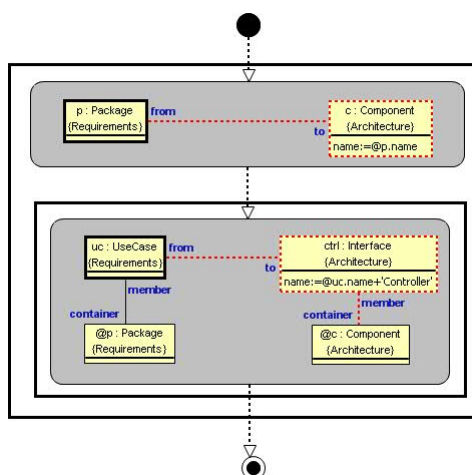
From Use Cases and Scenarios to Components?

- MD transformation for generating a controller class for each use case using the graphical notation of MOLA



From Use Cases and Scenarios to Components? (cont.)

- MD transformation for generating a controller class for each use case package, including an interface for each use case and an association to the corresponding component
- Tacit assumption: decomposition and grouping in the design (solution space) shall be the same as in the requirements (problem space)



From Use Cases and Scenarios to Components? (cont.)

- Probably results in less than optimal design
- Use cases in the requirements model should be grouped according to usage and goals.
- A good software design should group controllers according to low coupling and high cohesion.
- Refactoring needed after transformation
- Having an architecture with components and interfaces exactly corresponding to the use cases, their scenarios and contained functions may restrict the use of the system.
- Functions exposed directly may allow for unanticipated scenarios.



Summary and Conclusion

- OO business and domain modeling is useful.
- Objects, goals, scenarios / use cases and functional requirements can be combined.
- OO modeling can be used seamlessly from requirements to software design.
- Some mapping and transformation rules may be useful for certain systematic correspondences.
- Major manual adaptations and enhancements through the software architect will be needed.



Selected work of this tutorial presenter

- Kaindl, H., A Practical Approach to Combining Requirements Definition and Object-Oriented Analysis, *Annals of Software Engineering*, 3, 1997, pp. 319–343.
- Kaindl, H., Difficulties in the Transition from OO Analysis to Design, *IEEE Software*, Sept./Oct. 1999, pp. 94–102.
- Kaindl, H., A Design Process Based on a Model Combining Scenarios with Goals and Functions, *IEEE Transactions on Systems, Man, and Cybernetics (SMC) Part A* 30(5), 2000, pp. 537–551.
- Kaindl, H., Is Object-Oriented Requirements Engineering of Interest?, *Requirements Engineering*, 10, 2005, pp. 81–84.
- Kaindl, H., A Scenario-Based Approach for Requirements Engineering: Experience in a Telecommunication Software Development Project, *Systems Engineering*, 8, 2005, pp. 197–210.