

Software Engineering I: Software Technology

WS 2008/09

UML 2.0

Prof. Bernd Bruegge, Ph.D.

Florian Schneider

Applied Software Engineering

Technische Universitaet Muenchen

Outline for today

- Further Reading
- UML 1 to 2: A giant leap
- Overview of important changes covered in this lecture
 - Frames and nesting
 - Activity diagrams
 - Deployment diagrams
 - Sequence diagrams
 - Profiles, Stereotypes
 - And other changes

Further reading

- G. Booch, J. Rumbaugh, I. Jacobson
 - “The Unified Modeling Language User Guide”, 2nd Edition, Addison-Wesley Professional, 2005
 - For German readers: “Das UML Benutzerhandbuch. Aktuell zur Version 2.0”, Programmer’s Choice series, Addison-Wesley, 2006

- UML Superstructure, V2.1.2
 - <http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF>
 - For a list of all UML specifications, see <http://www.omg.org/spec/UML/>

Recent History of UML

March
2003:

- UML 1.5

July
2005:

- UML 2.0

November
2007:

- UML 2.1.2

A giant leap

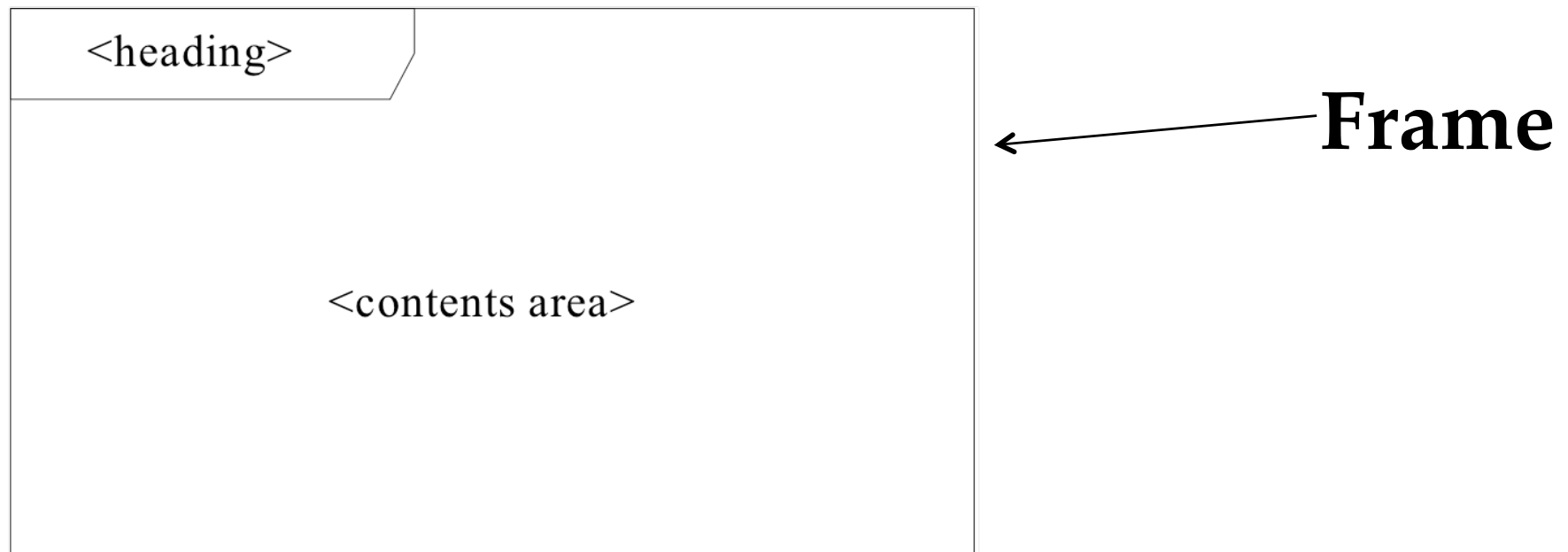
- UML now usable with Model Driven Architecture (MDA)
 - Better support for the automatic transformation of a Platform Independent Model (PIM) into a Platform Specific Model (PSM)
- UML based on Meta Object Facility (MOF)

Overview of important changes (non-comprehensive)

- Frames and nesting
- Changes in diagram notation:
 - Activity diagram
 - Deployment diagram
 - Sequence diagram
- Profiles and stereotypes
- New diagram types:
 - Composite structure diagrams
 - Timing diagrams
 - Not covered in this lecture

Common notation for all UML 2.0 diagrams

- Mandatory: **contents area**
- Optional: **frame with a heading**



$\langle heading \rangle ::= [\langle diagram\ kind \rangle] \langle name \rangle [\langle parameters \rangle]$

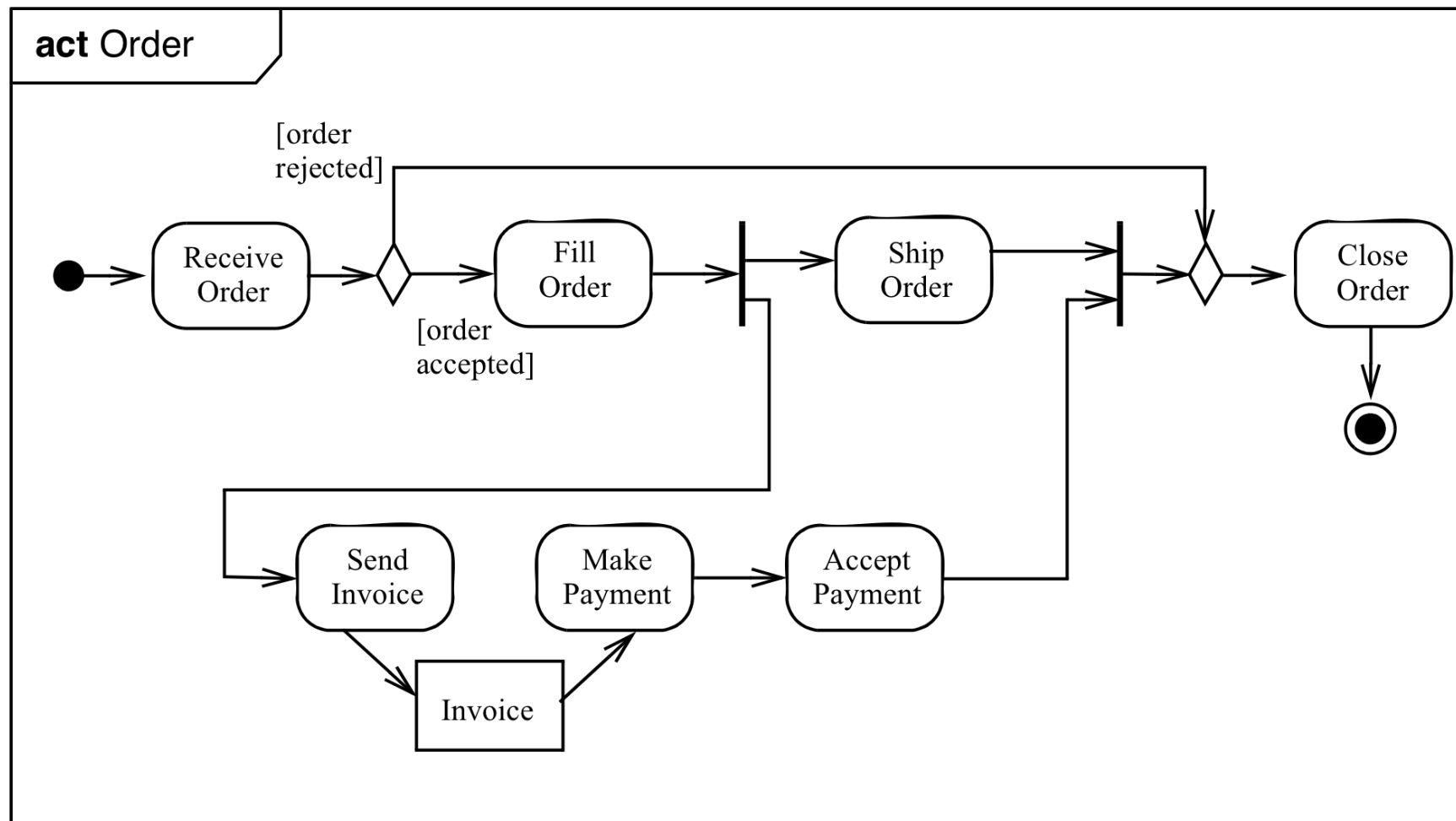
Keywords for Diagram Kinds

- activity
 - class
 - component
 - interaction
 - package
 - state machine
 - use case
- The following abbreviated forms are usually used:
 - **act** (for activity frames)
 - **cmp** (for component frames)
 - **sd** (for interaction frames)
 - **pkg** (for package frames)
 - **stm** (for state machine frames)
 - **uc** (for use case frames)

Why are interaction frames titled “sd”?

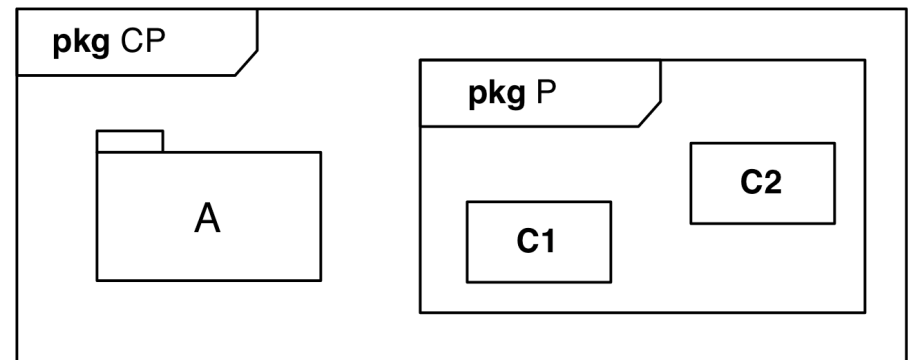
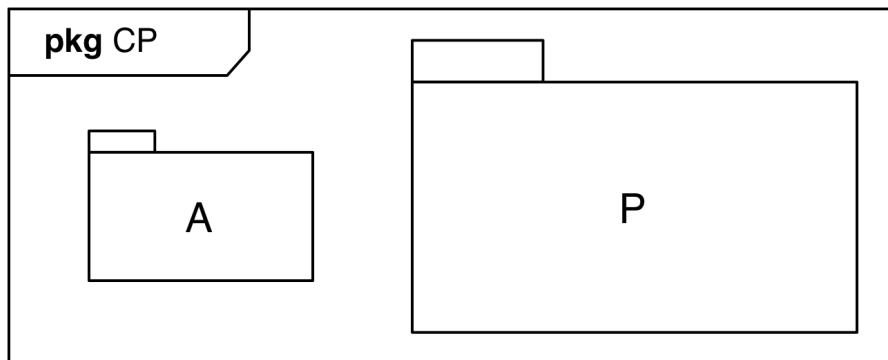
Historical: **s**equence **d**iagram

Example of Activity Diagram with Frame



UML 2.0 diagrams can be nested

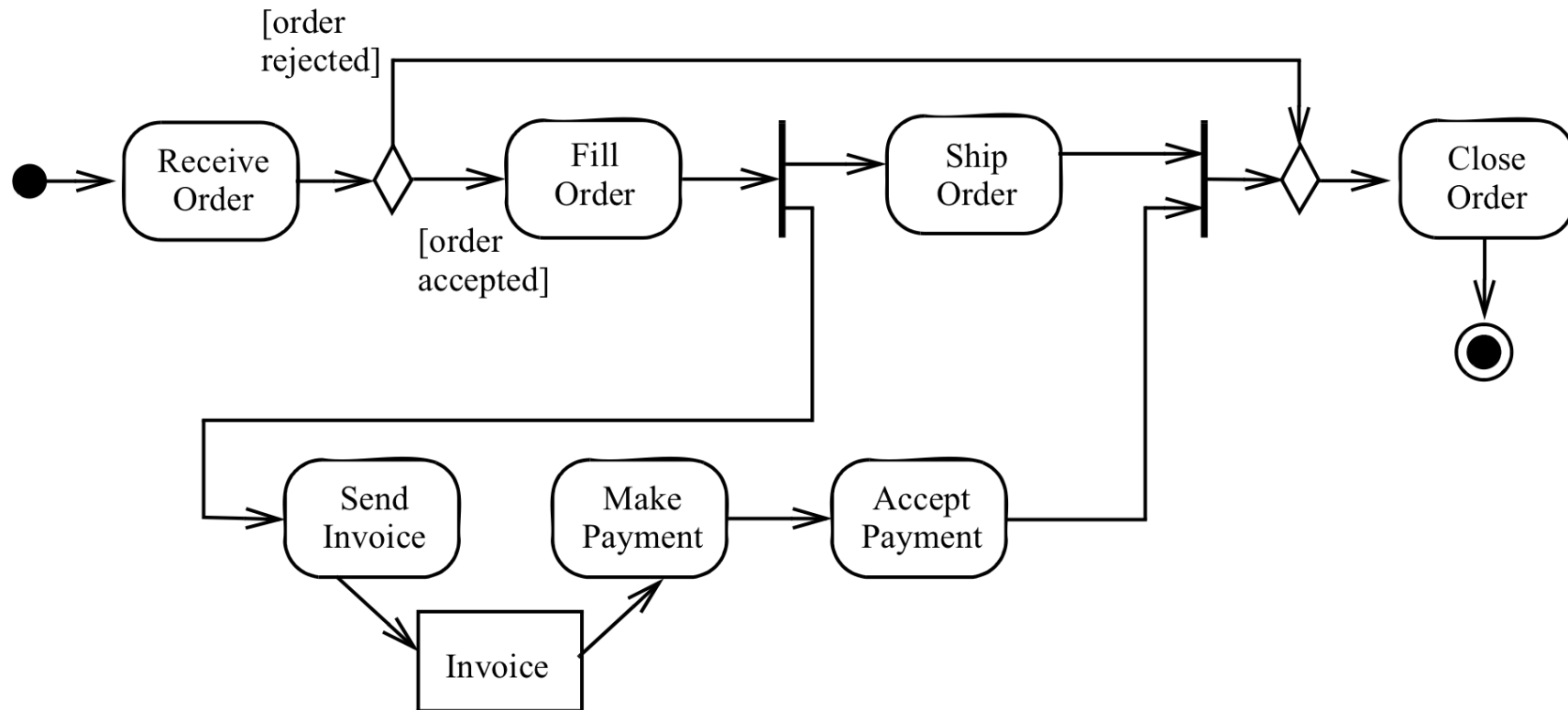
- UML 2.0 supports nested diagrams
 - e.g. an activity diagram inside a class
- The frame concept visually groups elements that belong to one diagram



Where are we now?

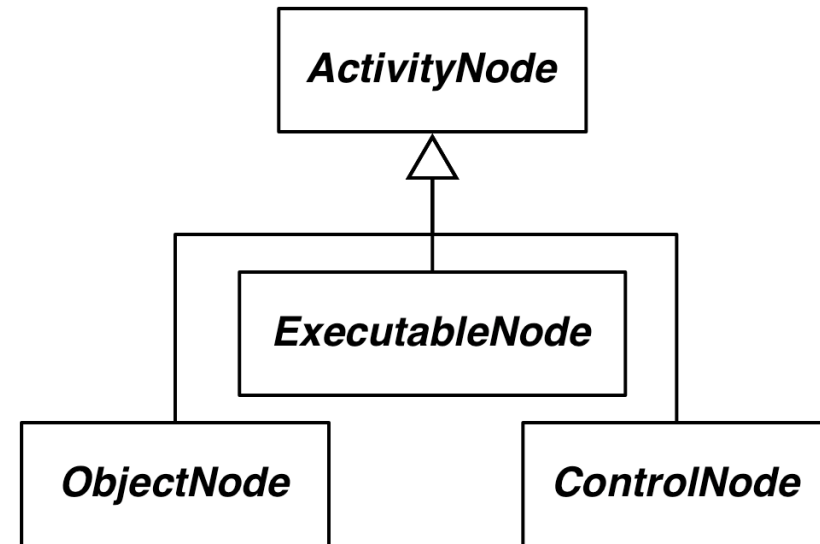
- ✓ Introduction
- ✓ Frames and nesting
- Changes in diagram notation:
 - Activity diagram
 - Deployment diagram
 - Sequence diagram
- Profiles and stereotypes
- Other changes

Again, the Activity Diagram Example



Activity Diagram Changes in UML 2.0

- New node taxonomy
 - See graphic:
- New node types
 - Loop node
 - Sequence node
 - Activity final node
 - Flow final node
 - Activity parameter node
 - Central buffer node
 - Data store node



Activity Diagram Changes: Action

- UML 1: Action is operation on transition of a state machine
- UML 2: Action
 - Is part of an activity
 - Has local pre- and post conditions

Activity Diagram: Activity Nodes & Edges

- An activity consists of activity **nodes**
- There are **three** types of nodes
 - Executable nodes
 - Most prominent: **Action**
 - Object nodes
 - E.g. a document
 - Control nodes
- An **edge** is a directed connection between nodes
 - There are two types of edges
 - Control flow edges
 - Object flow edges

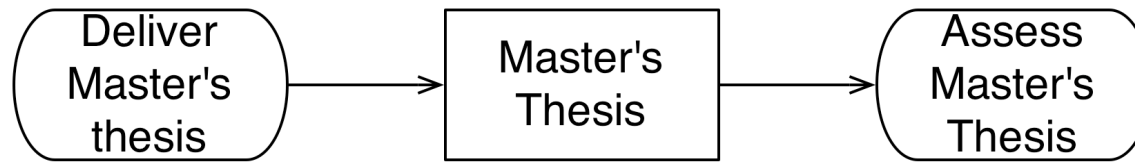
Activity Diagram: Control Nodes

- Initial node
- Final node
 - Activity final node
 - Flow final node
- Fork node
- Join node
- Merge node
- Decision node

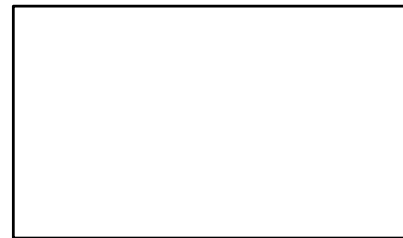


Activity Diagram: Action Nodes and Object Nodes

- Action Node

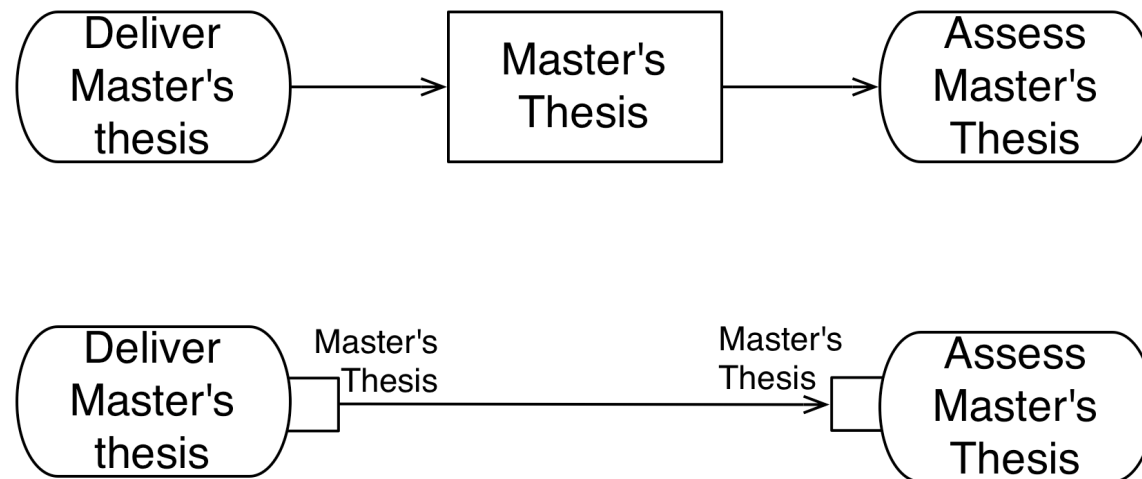


- Object Node

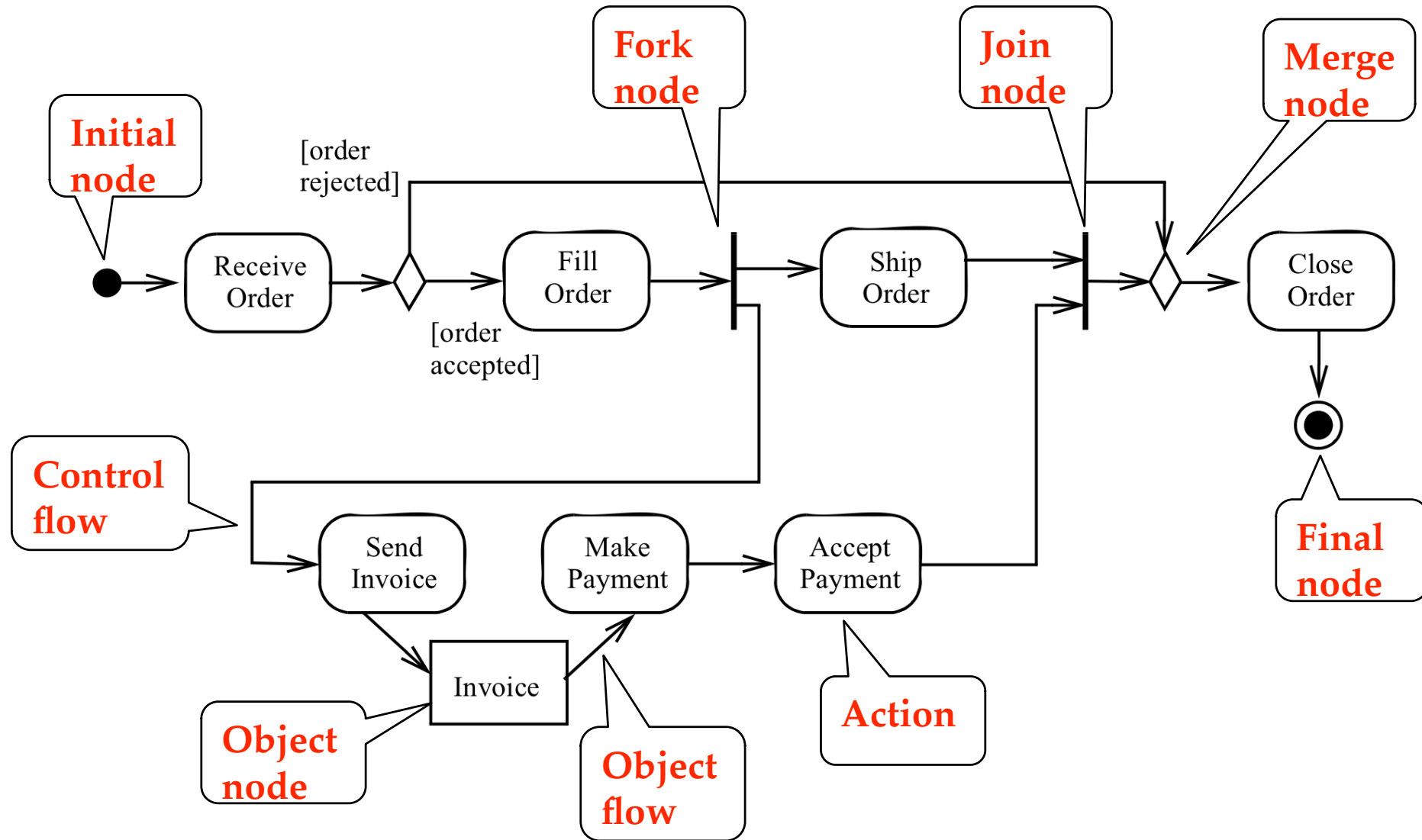


Activity Diagram: Object Nodes vs. Pins

- Different notations with same semantics
- Pins: abbreviated notation for object node
- Both define object flow in an activity



Activity Diagram Example

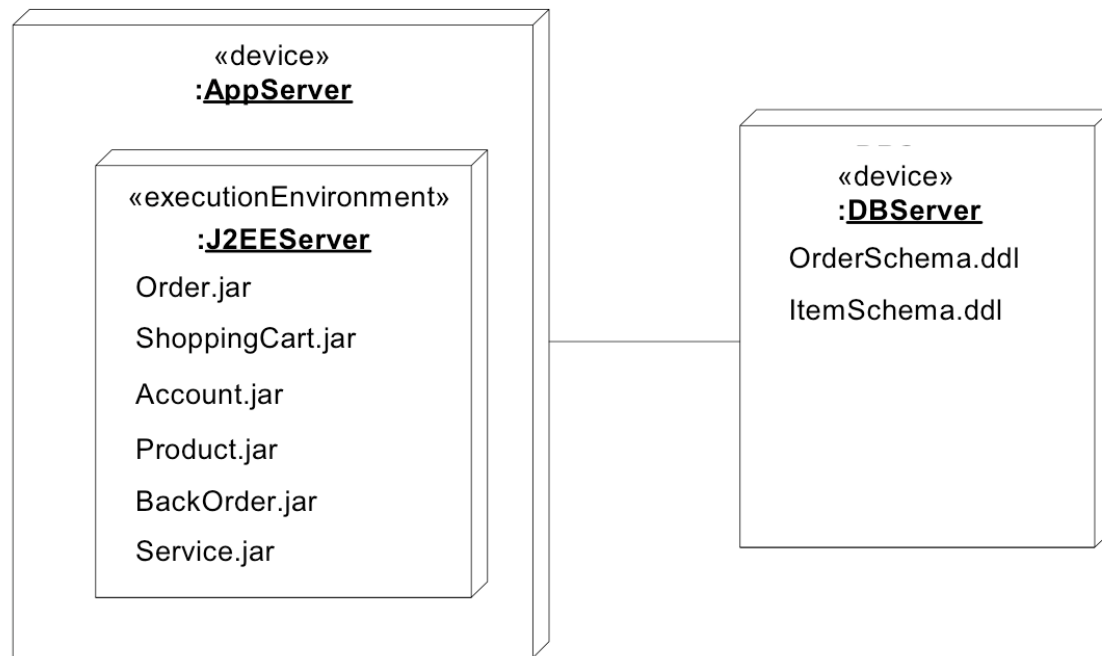


Where are we now?

- ✓ Introduction
- ✓ Frames and nesting
- Changes in diagram notation:
 - ✓ Activity diagram
 - Deployment diagram
 - Sequence diagram
- Profiles and stereotypes
- Other changes

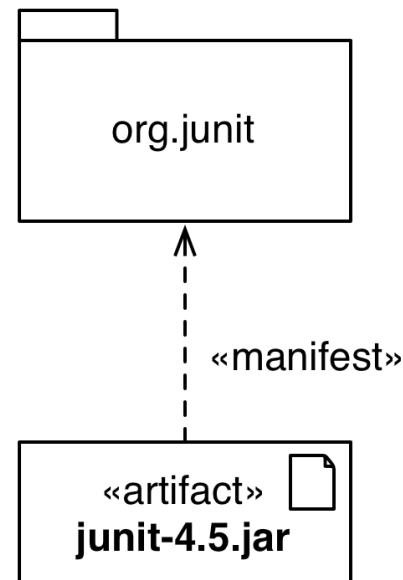
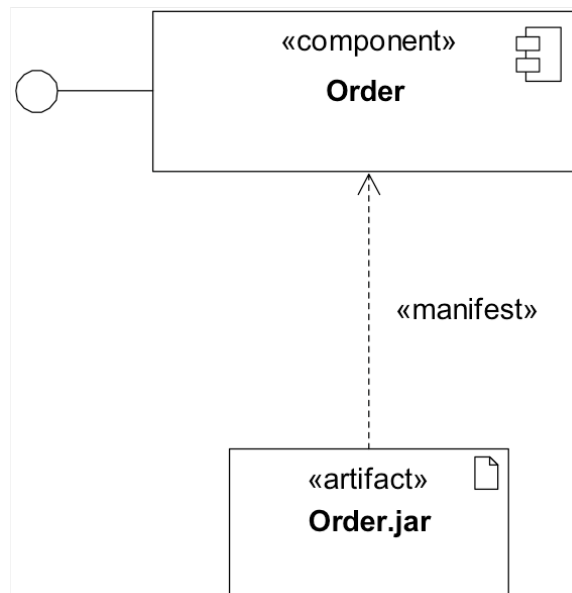
Deployment Diagram Changes I

- New node types:
 - Device
 - Execution environment



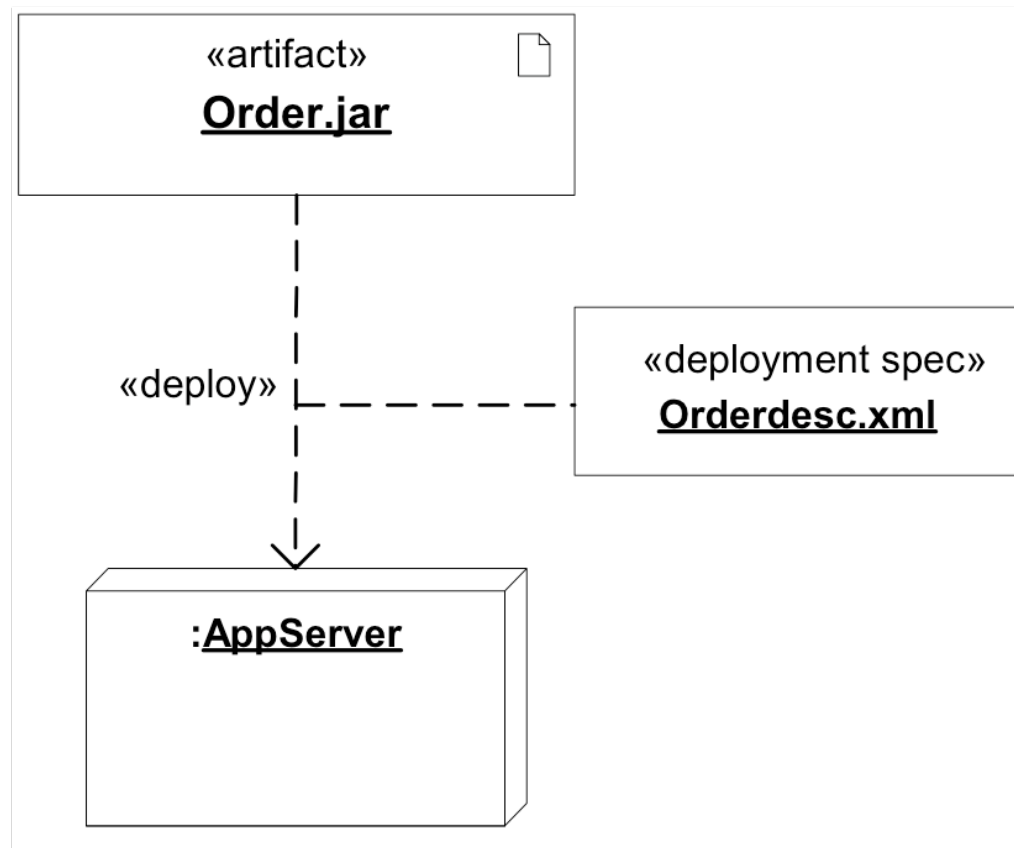
Deployment Diagram Changes II

- Artifacts can now manifest any packageable element, not just components
- Manifestation is shown by a dependency with keyword «manifest»



Deployment Diagram Changes III

- A deployment now has a deployment specification



Sequence Diagram Changes

- New concept of **interaction fragments**

- Before going into detail with interaction fragments, we cover the concept of an **interaction**

Interaction

- An Interaction is a concept providing a basis for interaction diagrams:
 - Sequence diagrams
 - Communication diagrams
 - Interaction overview diagrams
 - Timing diagrams
- An **interaction** is a unit of behavior that focuses on the observable exchange of information between connectable elements.
- We only focus on the impact of interactions on sequence diagrams

Usage of Interactions

- UML Interactions are used to get a better grip of an interaction situation
- Interactions are also used during the detailed design phase if precise inter-process communication must be set up according to formal protocols
- When testing is performed, the traces of the system can be described as interactions and compared with those of the earlier phases.

Example of an Interaction: Sequence Diagram

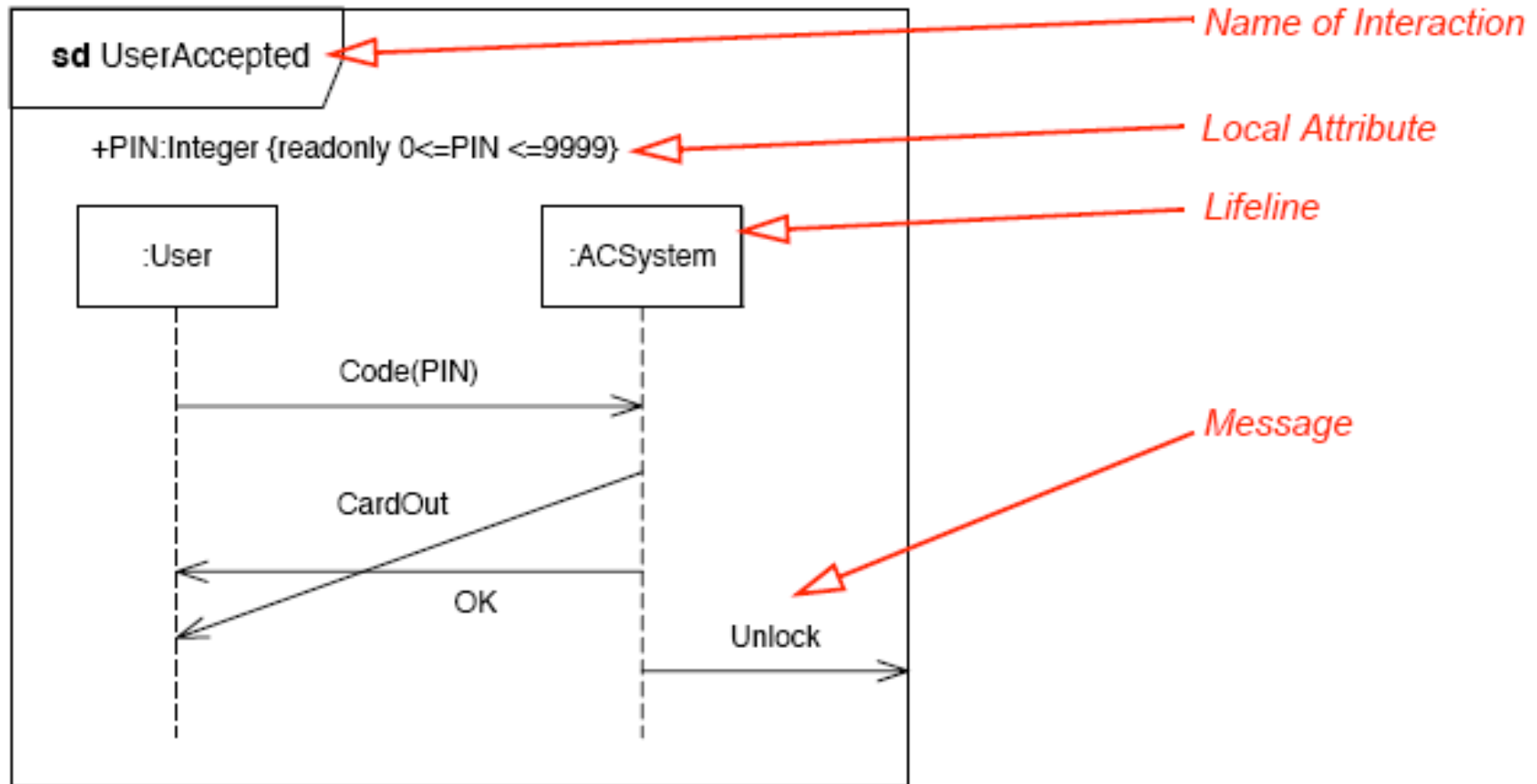


Figure 14.16 - An example of an Interaction In the form of a Sequence Diagram

Interaction Fragment

- Interaction Fragment
 - Is a piece of an **interaction**
 - Acts like an interaction itself
- Combined Fragment
 - Is a subtype of interaction fragment
 - defines an expression of interaction fragments
 - defined by an interaction operator and corresponding interaction operands

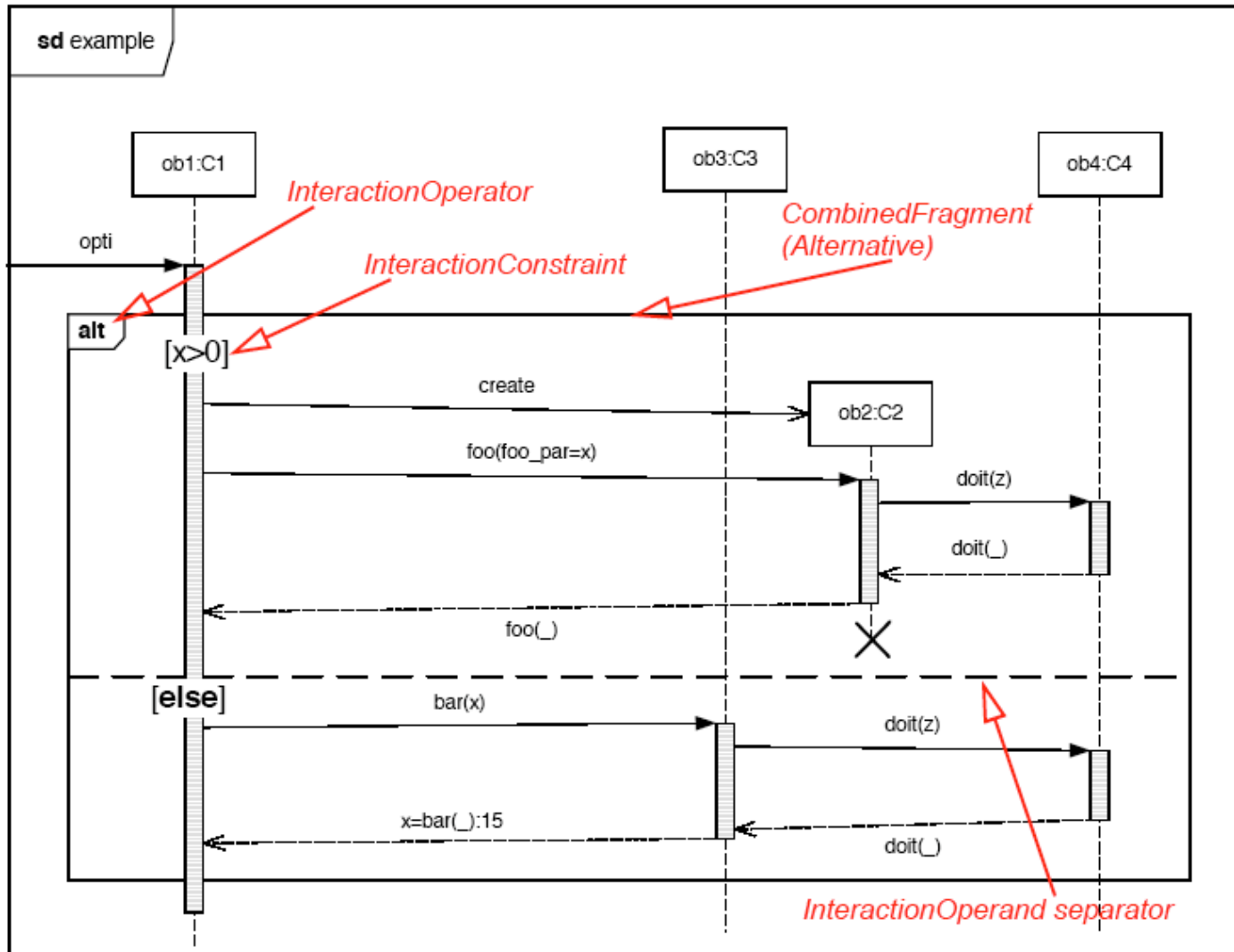
Interaction Operators

- A combined fragment defines an expression of interaction fragments. The following operators are allowed in an combined fragment expression:
 - Alt
 - Opt
 - Par
 - Loop
 - Critical
 - Neg
 - Assert
 - Strict
 - Seq
 - Ignore
 - Consider

Alt and Else Operators

- The interaction operator **alt** designates that the combined fragment represents a choice of behavior.
 - At most one of the operands will be chosen. The chosen operand must have an explicit or implicit guard expression that evaluates to true at this point in the interaction. An implicit true guard is implied if the operand has no guard.
 - The set of traces that defines a choice is the union of the (guarded) traces of the operands.
- An operand guarded by **else** designates a guard that is the negation of the disjunction of all other guards in the enclosing combined fragment.
 - If none of the operands has a guard that evaluates to true, none of the operands are executed and the remainder of the enclosing interaction fragment is executed.

Example of a Combined Fragment



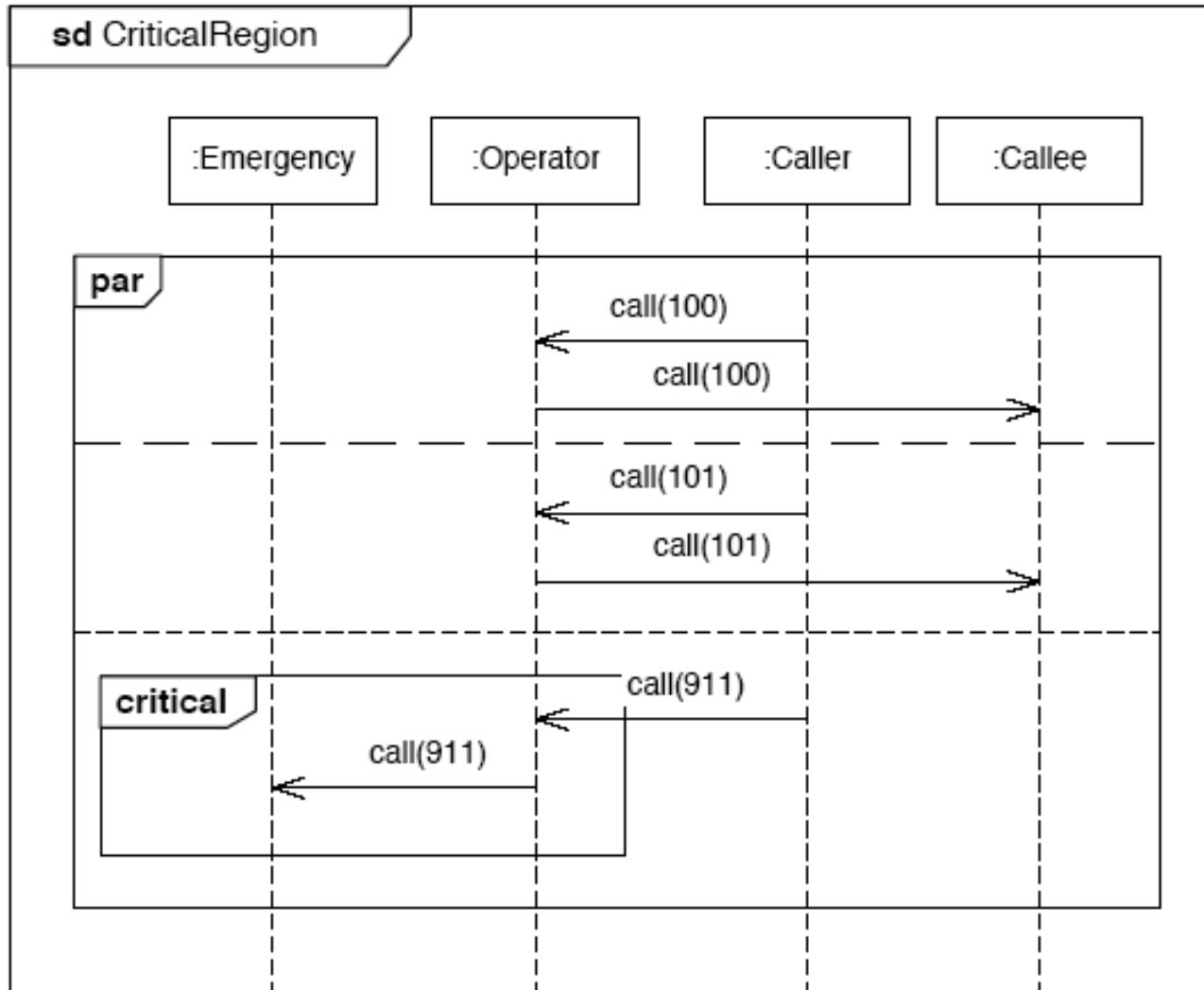
Opt and Break Operators

- **Option:** The interaction operator **opt** designates a choice of behavior where either the (sole) operand happens or nothing happens.
- **Break:** The interaction operator **break** represents a breaking scenario: The operand is a scenario that is performed instead of the remainder of the enclosing interaction fragment.
 - A break operator with a guard is chosen when the guard is true
 - When the guard of the break operand is false, the break operand is ignored and the rest of the enclosing interaction fragment is chosen.
 - The choice between a break operand without a guard and the rest of the enclosing interaction fragment is done non-deterministically.

Parallel and Critical Operator

- **Parallel:** The interaction operator **par** designates a parallel merge between the behaviors of the operands of a combined fragment.
 - The event occurrences of the different operands can be interleaved in any way as long as the ordering imposed by each operand is preserved.
 - A parallel merge defines a set of traces that describes all the ways that event occurrences of the operands may be interleaved.
- **Critical:** The interaction operator **critical** designates that the combined fragment represents a critical region.
 - The traces of the region cannot be interleaved by other event occurrences (on the Lifelines covered by the region). This means that the region is treated atomically by the enclosing fragment.

Example of a Critical Region



The Operator must make sure to forward a 911-call to the Emergency object before doing anything else. Normal calls can be freely interleaved.

Time Constraint with Messages

- **In this example**, time constraints are associated with the duration of a Message and the duration between two event occurrences.

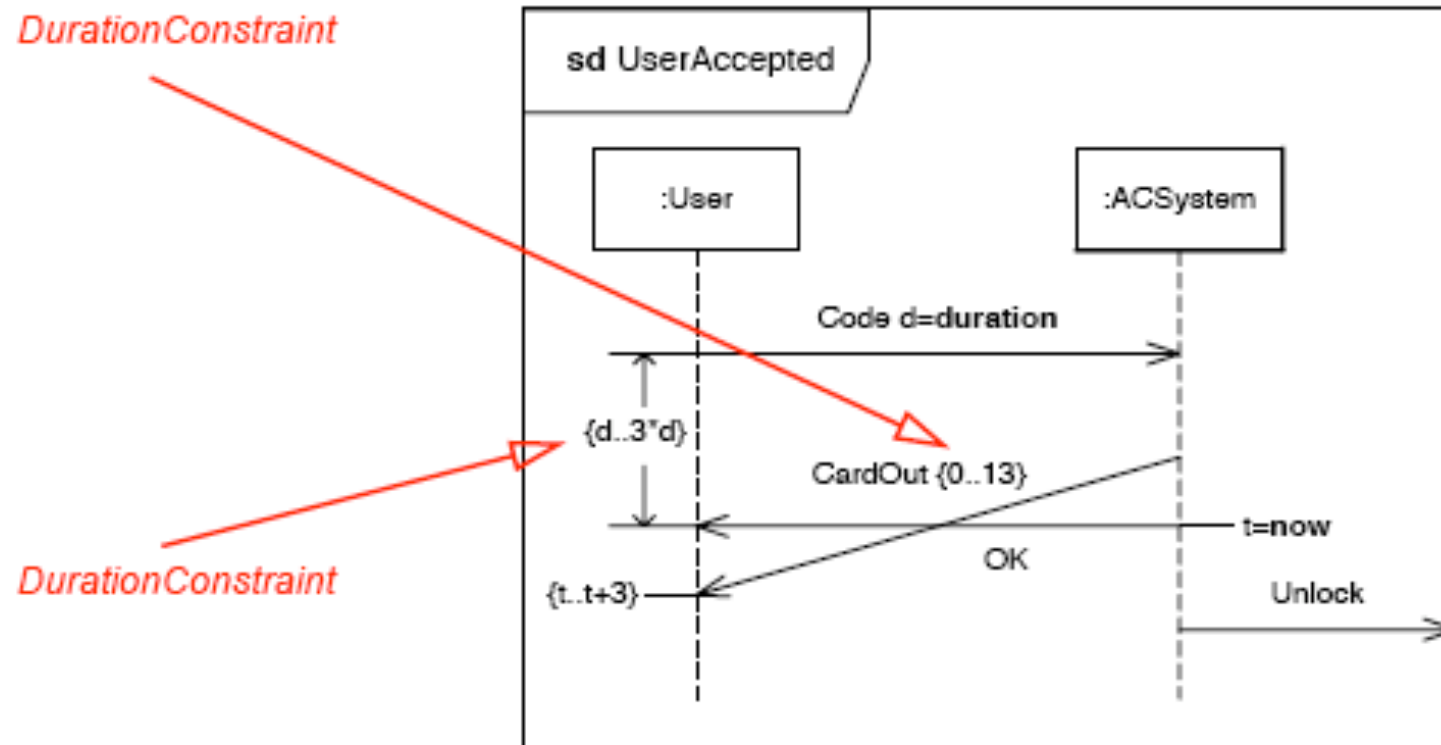


Figure 13.15 - DurationConstraint and other time-related concepts

Where are we now?

- ✓ Introduction
- ✓ Frames and nesting
- ✓ Changes in diagram notation:
 - ✓ Activity diagram
 - ✓ Deployment diagram
 - ✓ Sequence diagram
- Profiles and stereotypes
- Other changes

Excursion: Meta Modeling

- **Meta model:** A model describing a model
- **Meta class:** Part of the meta model, describing the structure of a model element

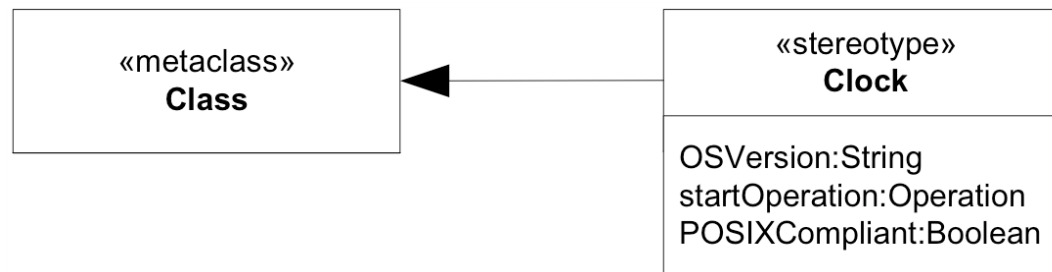
- Why are we talking about this?
 1. Meta models can be used to explain UML notation
 2. More about this in Friday's lecture
 3. We need the term meta class on the following slides

UML Profiles

- Lightweight extension mechanism for UML
- Concepts partially present in earlier versions
 - Stereotypes
 - Tagged Values
- Established as a specific meta-modeling technique in UML 2.0
 - Contains mechanisms that allow meta classes from existing meta models to be extended
 - ability to tailor the UML meta model for different platforms or domains
 - consistent with the OMG Meta Object Facility (MOF)
 - MOF will be covered on Friday

Stereotype

- Defines how an existing meta class may be extended
- Can only be used in conjunction with a meta class
- A meta class may be extended by one or more stereotypes



- Why should I want to do this?
 - You may want to use platform or domain specific terminology

Stereotype Notations

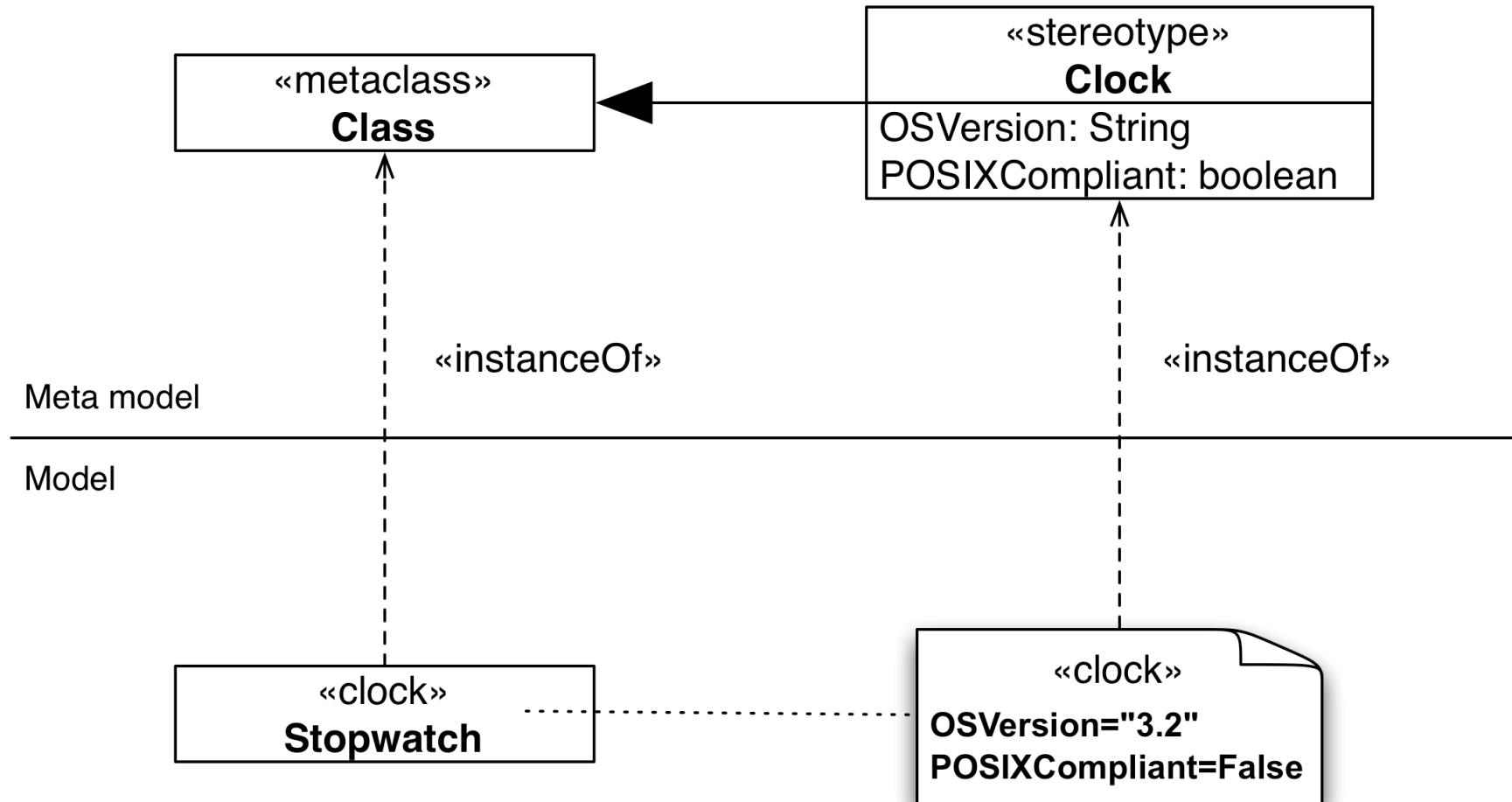
- The application of a stereotype is shown with a String enclosed in guillemets before the classifier name
 - E.g. «boundary»
 - Stereotypes you might already know:
 - «file» extends meta class Artifact
 - «framework» extends meta class Package
- Stereotypes can additionally have an image
 - The image may replace the standard notation of the element the stereotype has been applied to



Stereotype application I

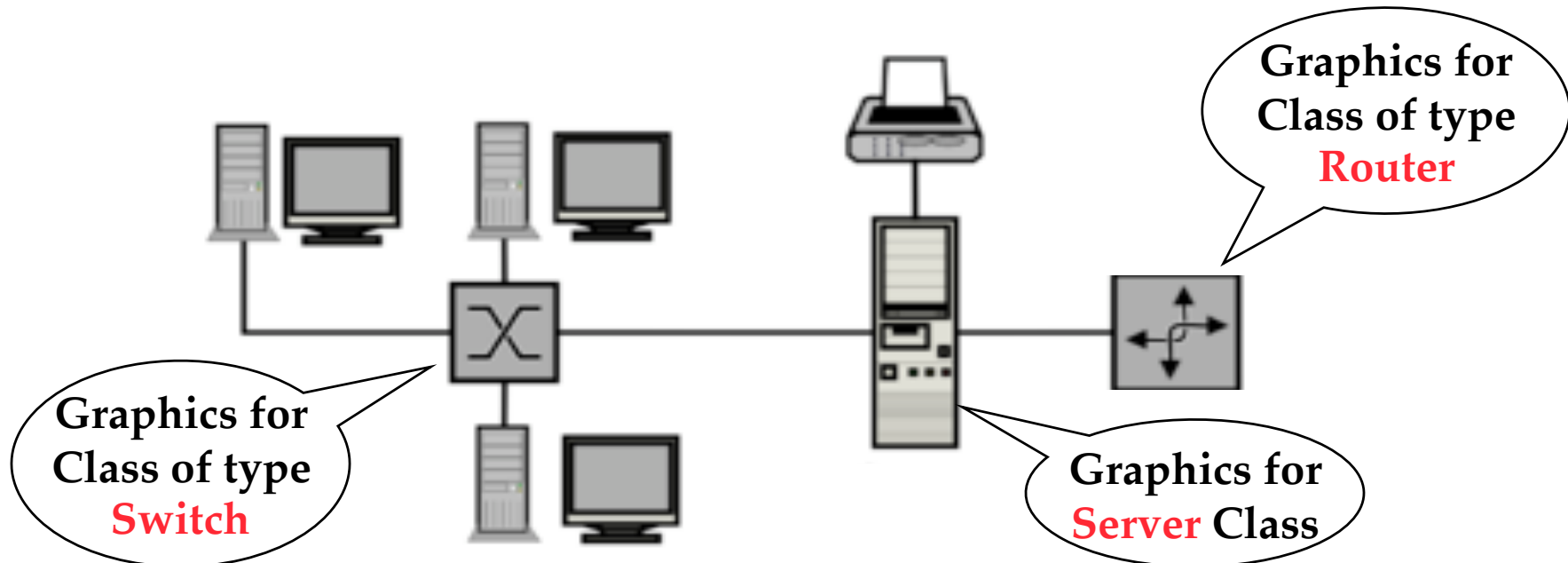
- The attributes of a stereotype are called **tagged values**
- Applying a stereotype to an instance of a meta class means instantiating the stereotype
 - If the stereotype has attributes, you have to provide values for them
- Confused? See following slide...

Stereotype application II



Graphics for Stereotypes

- One can also use **icons** or **graphical symbols** to identify a stereotype
 - When the stereotype is applied to a UML model element, the graphic replaces the standard notation for the model element.
- **Example:** When modeling a network, define graphics for representing classes of type **Switch**, **Server**, **Router**, **Printer**, etc.



Pros and Cons of Stereotype Graphics

- Advantages:
 - UML diagrams may be easier to understand if they contain graphics and icons for stereotypes
 - This can increase the readability of the diagram, especially if the client is not trained in UML
 - And they are still UML diagrams!
- Disadvantages:
 - If developers are unfamiliar with the symbols being used, it can become much harder to understand what is going on
 - Additional symbols add to the burden of learning to read the diagrams
- If you end up applying stereotypes all over your model, you should think of defining a profile.

Stereotypes vs. Keywords

- Not to be mixed up with keywords
 - Same notation (String enclosed in guillemets)
 - «interface» is no stereotype!
 - «extend» is no stereotype!
- See Annexes B and C of the UML Superstructure specification.

Other notable Changes in UML 2.0

- Composite Structure Diagrams
- Timing diagrams
- See further reading for more details!

- Exercise Sheet: Friday (tomorrow)
- Friday's lecture: The UML 2.0 Meta model

Thank you!

Explanation of the Interaction on the previous slide

- The example on the previous slide shows three messages between two anonymous lifelines of types User and ACSystem: CodePin, CardOut and OK
 - The message CardOut overtakes the message OK in the way that the receiving event occurrences are in the opposite order of the sending event occurrences.
 - Such communication may occur when the messages are asynchronous.
- A fourth message Unlock is sent from the ACSystem to the environment
 - Through a gate with the implicit name out_Unlock.