<table>
<thead>
<tr>
<th>Diagram type</th>
<th>Diagram</th>
<th>Phase</th>
<th>Application area</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use cases</strong></td>
<td>Use case diagrams</td>
<td>• Requirements</td>
<td>Business processes, general applications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Definition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Creation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Delivery</td>
<td></td>
</tr>
<tr>
<td><strong>Static Models</strong></td>
<td>Class diagrams Instance diagrams</td>
<td>• Definition</td>
<td>Everywhere, the class diagram is the most Important diagram in UML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Creation</td>
<td></td>
</tr>
<tr>
<td><strong>Dynamic Models</strong></td>
<td>Activity diagrams Interaction diagrams Collaboration diagrams</td>
<td>• Requirements</td>
<td>Shows the information flow and the time flow of the cooperation of the objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Definition</td>
<td>Timeflow structure with few messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Creation</td>
<td>Timeflow structure with few classes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Delivery</td>
<td></td>
</tr>
<tr>
<td></td>
<td>State diagrams</td>
<td>• Requirements</td>
<td>Representation of the dynamic behavior</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Definition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Creation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Delivery</td>
<td></td>
</tr>
</tbody>
</table>
Interaction diagrams

- Interaction diagrams represent a scenario graphically.
- Each of these interaction diagrams contains at least
  - **Life cycle data** (production, initializing, use, deletion) of objects
  - Communication pathways (links)
  - Reaction of receiving objects
  - Temporal order of messages
  - **Messages** along the links
- Benefits of interaction diagrams
  - Simple presentation allows quick review and reading of details
- Presentation through either
  - **Sequence diagrams** with emphasis on temporal progress or
  - **Collaboration diagrams** with emphasis on the collaboration of objects
Sequence diagrams are the most important interaction diagrams and show the timely run of a series of messages (method calls) between certain objects in a time-limited situation.

- The vertical time axis is therefore important. It is represented as a dashed life line for each object.
- The calls can be construed as method calls as synchronous or asynchronous messages, or also as their own methods calls along a life line. According to the desired degree of detail, call parameters and return values can also be given.
Sequence diagram:

- **Time axis**: runs vertically downwards and determines the temporary order of the events.

- **Objects**: are represented by UML object symbols. A vertical dashed line parallel to the time axis represents the lifeline of the object. If the object is active, its lifeline is covered by an activity bar.

- **Message**: each interaction between the objects is shown by a message, represented by a directed horizontal arrow between the objects.
Sequence Diagrams and their Graphic Notation (3)

- **Synchronous call**
  An arrow with a **filled peak** represents a synchronous method call. The call is made from source to target, the targeted class must implement the respective method, accordingly. The source waits until the target finishes the processing before continuing its activities. The synchronous calls are ended with a return arrow, that can be optionally labeled. **Return is shown by a dashed arrow with empty peak from target to source.**

- **Asynchronous call**
  Asynchronous calls are shown with an **arrow with empty peak**. The call goes from source to target. The source does not wait for the target to finish its processing, but continues with its activity. Asynchronous calls are well-suited for parallel processing.
• **Sequence diagram:**

  - **Ramification**
    Sequence diagrams can be branched to two targets, depending on a condition, that is shown inside square brackets. If the condition \(b==true\), message 1 is send (to object 2). If \(b==false\), message 2 is send (to object 3).

  - **Recursion**
    A recursive message calls on the source object one of the object’s methods.
Sequence diagrams and their graphical notation

3.2. object oriented modelling with UML

3.2.4. Dynamic Models

Object

Life line

Messages

Creation of new objects

Asynchronous messages

Destruction of an object

:Actor

New()

:Object1

New()

:Object2

Message2()

Message3() (return)
Example:

Describe with a sequence diagram the following scenario;

A student wants to recharge his mensa card. The following objects occur in the scene: MensaAutomat, Card, AutomatDisplay

Solution:

```
Student

mensaAutomat : MensaAutomat

moneyIn()

readCard()

showAmount()

receipe()

loadCard()

Card : MensaCard

AutomatDisplay : AutomatDisplay
```
Collaboration diagrams

- A collaboration diagram shows **which objects can work together by a specified system activity** - similar to a sequence diagram.
- The **timely progression of the calls** can be given here through a **prefixed order of numbers**.
- Collaboration diagrams are used prior to this to demonstrate which objects, if any, can work together in a specific situation.
- The **time axis** plays a **secondary role**.
• The objects are represented by rectangles. These are interlinked with association lines, that are labeled with messages. The label consists of continuous numbering, the name of the message(s) and answer(s) as well as their possible arguments.

• An arrow shows the direction of the message, from sender to receiver.

• Answers are shown as \[ \text{answer} := \text{message}() \].

• The messages are numerated sequenced starting with 1, the start message is not numbered.
Collaboration Diagrams and their Graphical Notation (2)

Actor

Start message() → Object1

Object2

1.2:answer := message()

1.1:message()

Object3

[Condition1]

Messages, comprised of timed numbering

answer

Associations line

[Condition2]

Start messages contains no numbering

3.2.4. Dynamic Models

3.2. object oriented modelling with UML
Example of a sequence and collaboration diagram (1)

- The assignment acquisition window sends a message „prepare“ to the assignment
- The assignment then sends the „prepare“ to every assignment position of the assignment
- Every assignment position checks the indicated article in stock
  - When this examination delivers „true“, the assignment position deletes the corresponding quantity of articles in stock from the warehouse
  - Otherwise, the quantity of the articles in stock falls under the boundaries of the order, and the article in stock requires a new delivery
Example of a sequence and collaboration diagram (2)

3.2. object oriented modelling with UML

3.2.4. Dynamic Models

:An Assignment Acquisition Window

:An Assignment

:An Assignment Position

:An Item In Stock

Object

Message

Iteration

Preparation

Available

Verification

Iteration condition

Return

Condition

Self-delegation

Repeat order

Repeat order necessary

New

Remove

Necessary

Prepare

Verify

New

A Repeat Order Item

A Deliverable Item
Example of a sequence and collaboration diagram (3)

1.: prepare()

2.: *[for each assignment position]: prepare()

3.: available := check()

4.: remove()

5.: repeat order := repeated orderNecessary( )

6.: new()

7.: new()

Object

Message

Sequence Number

Self-Delegation

: An Assignment Window

: An Assignment

: An Assignment Position

: A Deliverable Item

: A Repeat Order Item

: An item in Stock

1.2. object oriented modelling with UML

3.2.4. Dynamic Models
Each message processing corresponds to a transition in the state diagram.
2.2. Objektorientierte Modellierung mit UML

2.2.4. Dynamische Modelle

- Operations change the attribute values of the instance
- This means a transition into another state in the state diagram

Class Diagram

<table>
<thead>
<tr>
<th>Kl</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
</tr>
<tr>
<td>a2</td>
</tr>
</tbody>
</table>

/ z: State

<table>
<thead>
<tr>
<th>op1</th>
</tr>
</thead>
<tbody>
<tr>
<td>op2</td>
</tr>
<tr>
<td>op3</td>
</tr>
</tbody>
</table>

State Diagram

- op1
- op3
- opX
Correlation: Class and Sequence Diagrams

3.2. object oriented modelling with UML

3.2.4. Dynamic Models

Class diagram

**K11**

**K12**

Instances

```
 ob2:K11
```

```
 ob1:K11
```

```
 ob3:K12
```

Sequence diagram
Outline

Lecture Content

1. Preface
2. Basics
3. Object orientation
   3.1. Introduction and basic concepts
      3.1.1. Introduction
      3.1.2. Objects and classes
      3.1.3. Attributes of object orientation
   3.2. Object-oriented modelling with UML
      3.2.1. Intro
      3.3.2. Use Case models
      3.2.3. Static models
      3.2.4. Dynamic models
      3.2.5. Summary
   3.3. Methods of the OO Analysis and Design
Weaknesses of UML

- Semantics presently described only textually, only informal, *ambiguities possible*;
- Methodology of utilization of UML insufficiently researched (note: multiplicity of methodologies possible, use case specific)
- Little support for real-time applications
- Support of semantics-containing refining
Typical Problems with Graphical Modeling

- Relationships between diagrams:
  - Overlap can lead to inconsistencies,
  - Separation can lead to incompletion
- Various interpretation of diagrams by developers/clients
- Step-by-step refining of diagrams cannot be viewed. The addition of details often does not preserve the semantics
- Scalability of graphic notation often problematic, hierarchical draft or other structuring necessary
- Tabular representation sometimes clearer
Summary

- Notation for the description of the structure of object oriented programs
- Based on UML, entire program fragments can be generated
  - MDA (Model Driven Architecture)
- Different layers describing the same circumstances
- Dynamic and static diagram types, describing both the structure and the behavior
- CASE Tools (Computer Added Software Engineering) can generate source code from UML diagrams and vice versa.
  - Code $\rightarrow$ UML = Reverse Engineering
  - Code $\rightarrow$ UML $\rightarrow$ Code = Roundtrip Engineering
- Overview on CASE / UML Tools: [http://jeckle.de/umltools.html](http://jeckle.de/umltools.html)
Recommended Literature

- Dr. Christian Prehofer: Vorlesung Objektorientierung, TU München, SS 2001
- Dr. Uwe Aßmann, Vorlesung Informatik II, Institut für Programmstrukturen und Datenorganisation, Universität Karlsruhe, SS 2000
- Martin Fowler / Kendall Scott: UML konzentriert, Addison-Wesley, 2te Auflage, 2000
- G. Krüger: Handbuch der Java-Programmierung, Addison-Wesley, 2000
- [http://www.wikipedia.de](http://www.wikipedia.de)
Recommended Literature


[HiKa99] M. Hitz, G. Kappel: „UML@Work“, dpunkt.verlag, 1999


Recommended Literature


