## Modelling Method Implementation Instruments

| Modelling Method Conceptualisation | Modelling Language | | | Modelling Procedure | Mechanisms and Algorithms | | |
|---|---|---|---|---|---|---|---|
| | Notation | Syntax | Semantic | | Basis Functions | Script Functions | External Access |
| Particulari-sation | PART - 1 | | | | | | |
| Implementation | Model Type: Method-GraphRep, Icons | Model Type: ADOxx Library Language | | | Functional Description Table | | |
| | Object: GraphRep | Object: ADOxx Library Language | Object: Class Hierarchy | | Configuration Description | Scripting and Expression | ADO Web-Service (ADOscript API, Java API) |
| | Attribute: GraphRep | Attribute: ADOxx Library Language | | | | | |





## Platform Specific Development Environment for the Implementation Process

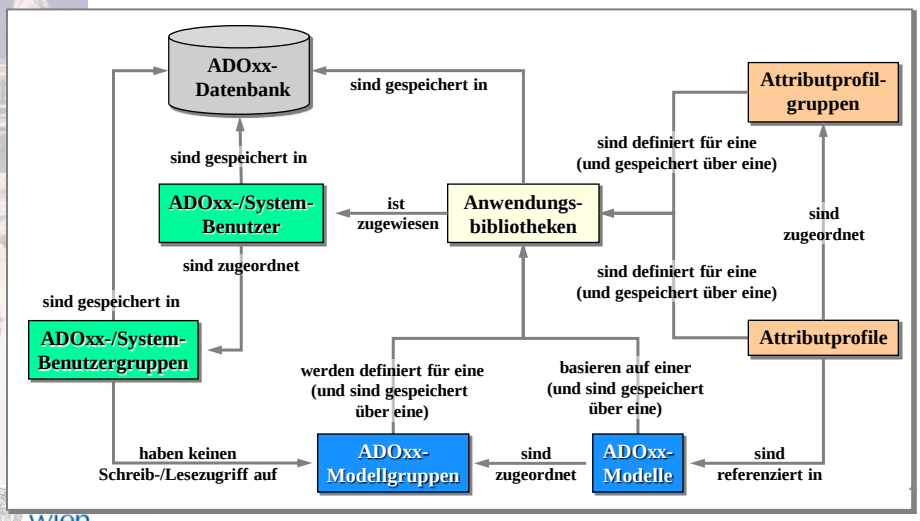
**ADOxx Platform**

ADOxx Method Development Environment
- Implementation in ADOxx Library Language (ALL)
- Implementation in ADOxx Method Development Tool

ADOxx Modelling Language Implementation
- ADOxx Model Hierarchy
- ADOxx Class Hierarchy
- ADOxx GraphRep

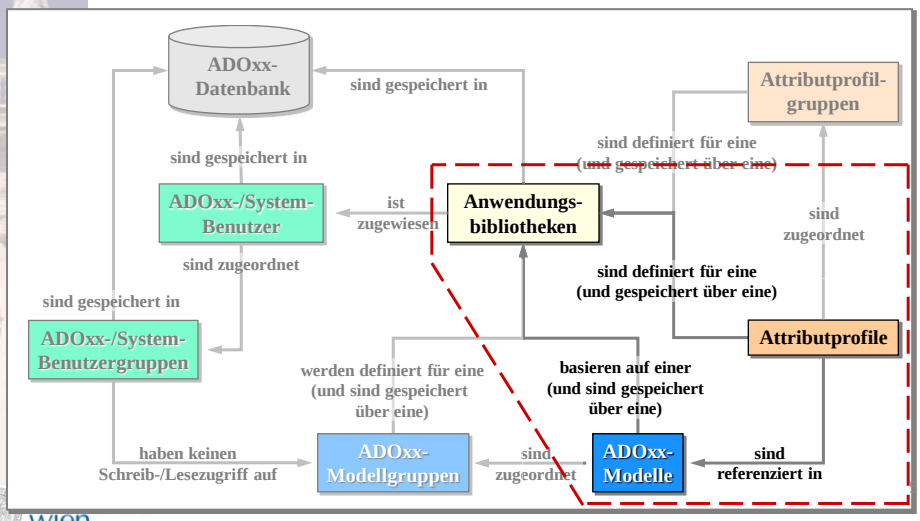
ADOxx Mechanisms and Algorithms
1. Basis Functionality
2. ADOscript
3. External Access
   1. ADOscript
   2. ADO Web-Service

## Zusammenhänge in ADOxx






## Auswirkungen des Customizings



## Was bedeutet ADOxx Customizing?

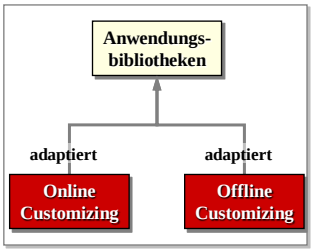
Unter Customizing versteht man die Tätigkeit, ADOxx an die Modellierungsmethode anzupassen. Diese Anpassungen sind ohne programmtechnische Änderung möglich. Man unterscheidet zwei verschiedene Arten des Customizings:

Online-Customizing:

- Verwalten von Klassen und Beziehungen
- Gestaltung der ADOxx-Notebooks
- Klassenkardinalitäten
- Vordefinierte Abfragen
- …

Offline-Customizing:

- Neue Klassen, Beziehungen und Attribute



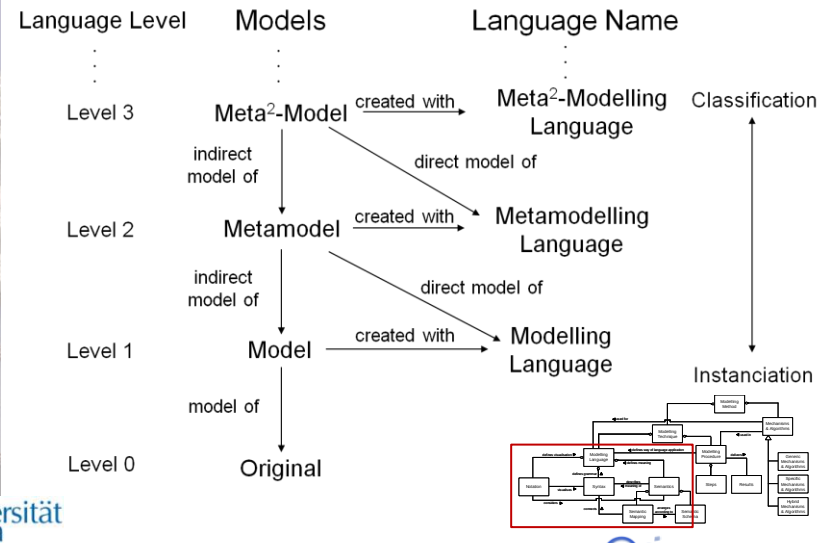



universität wien DKE



OPEN | MODEL Initiative www.openmodels.at

## Sprachen-Level: Die Modellhierarchie



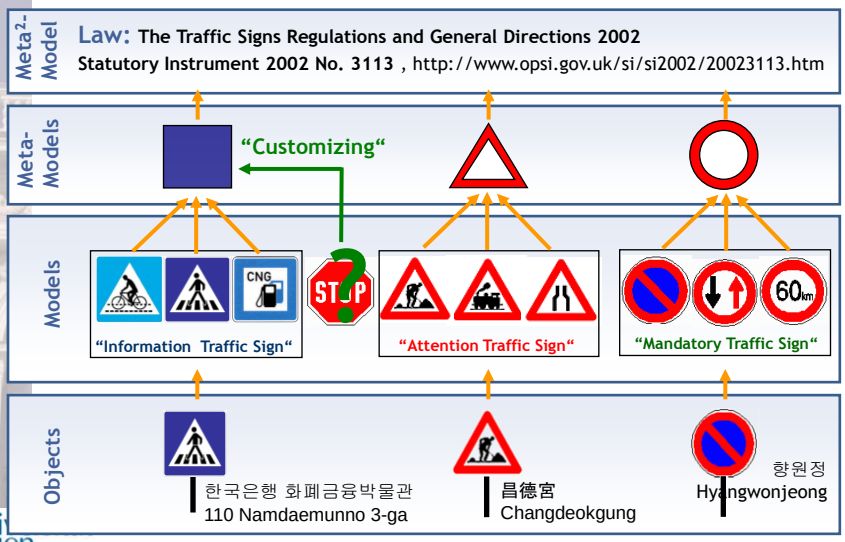



universität wien DKE

o.Univ.-Prof. Dr. Dimitris Karagiannis

OPEN | MODEL Initiative www.openmodels.at

## AN ILLUSTRATIVE EXAMPLE FOR METAMODELLING TRAFFIC SIGNS IN SEOUL

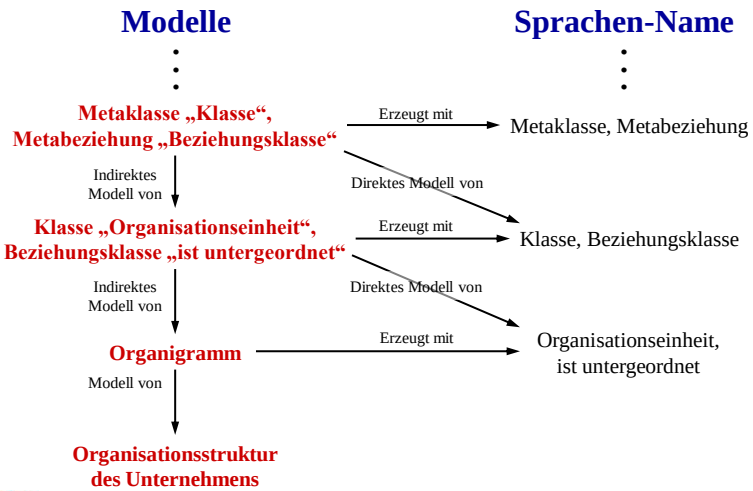




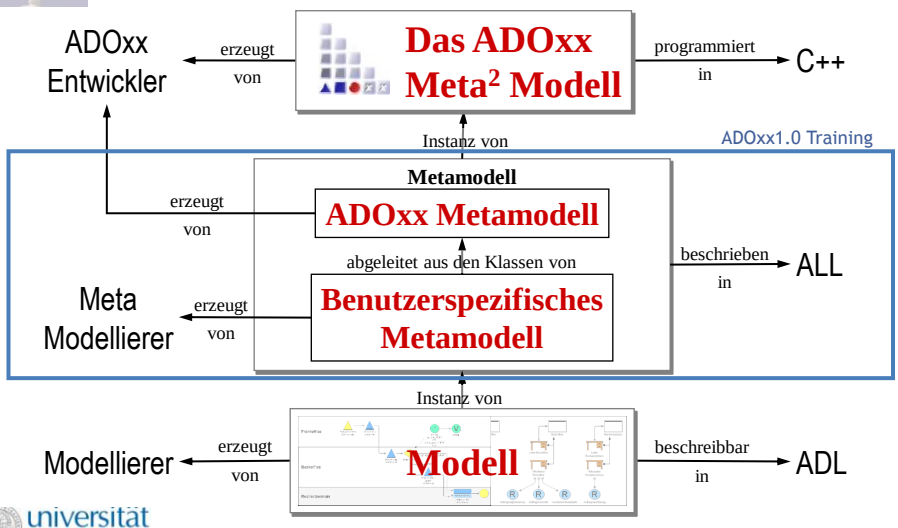





## Sprachen-Level: Ein Beispiel



## Die Modellhierarchie





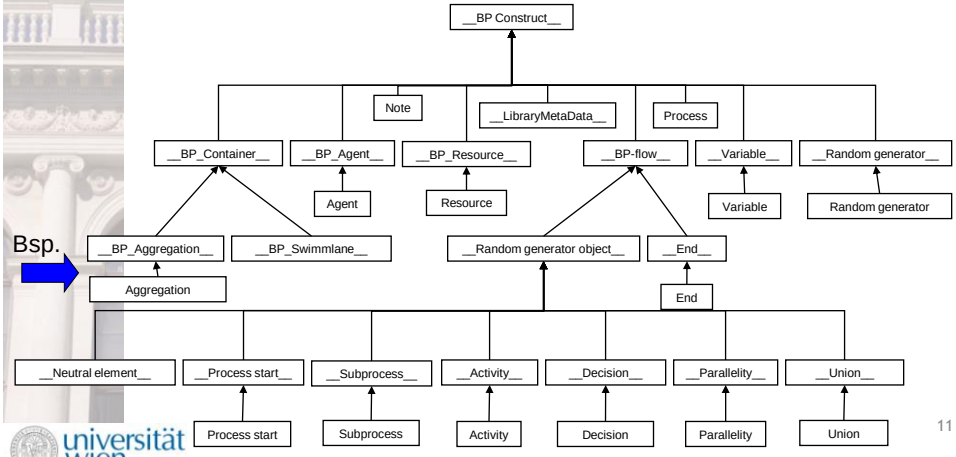

universität wien DKE



OPEN | MODEL Initiative www.openmodels.at

## Anpassung der Funktionalität der Plattform

Anpassen der Plattformfunktionalität durch Polymorphismus von zur Verfügung gestellten abstrakten Klassen. Dadurch erbt die neue Modellierungsklasse die Strukturdefinition und ist „annotiert“.



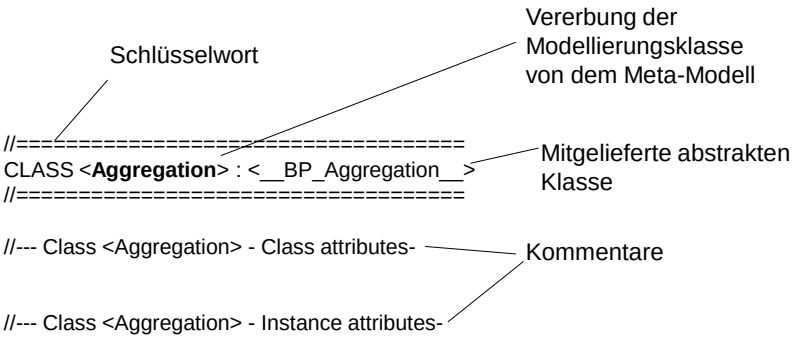



universität wien DKE

o.Univ.-Prof. Dr. Dimitris Karagiannis

OPEN | MODEL Initiative www.openmodels.at

## Definition einer Modellierungsklasse



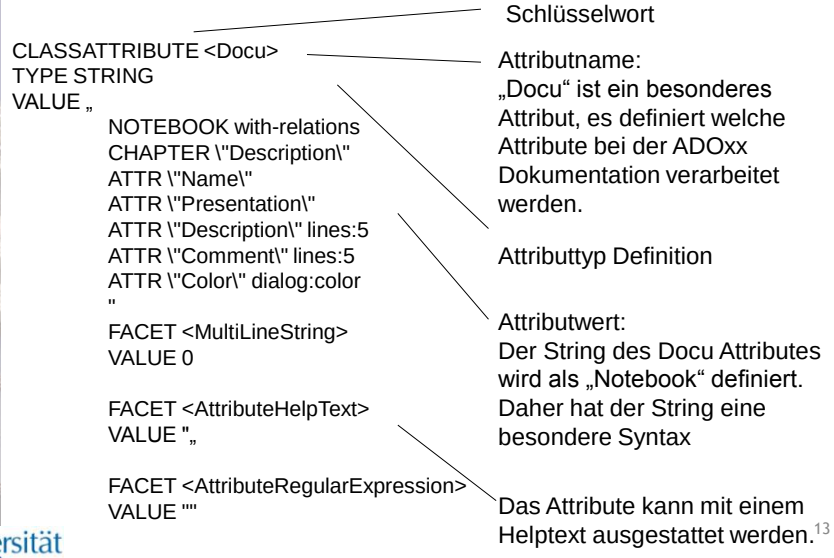



universität wien DKE



OPEN | MODEL Initiative www.openmodels.at

## Beispiel einer Klassenattribute Definition



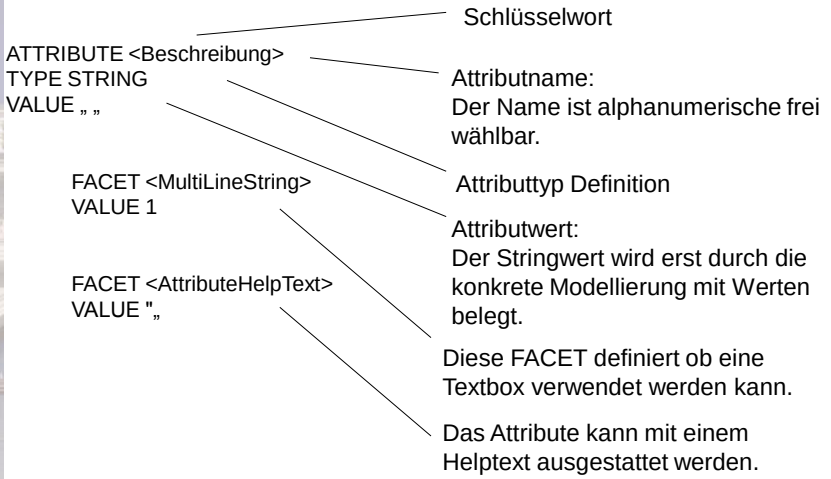



universität wien DKE

o.Univ.-Prof. Dr. Dimitris Karagiannis

OPEN | MODEL Initiative www.openmodels.at

## Beispiel eines Instanzattribut Definition






## Beispiel von MetaDaten

Attribute können definiert und mit beschreibenden Default Wert belegt werden. Damit der Benutzer diese Metadaten nicht verändern kann, dürfen diese in der Notebook Darstellung nicht angeführt werden. Somit sind sie „versteckt“ und nur durch Processing veränderbar.

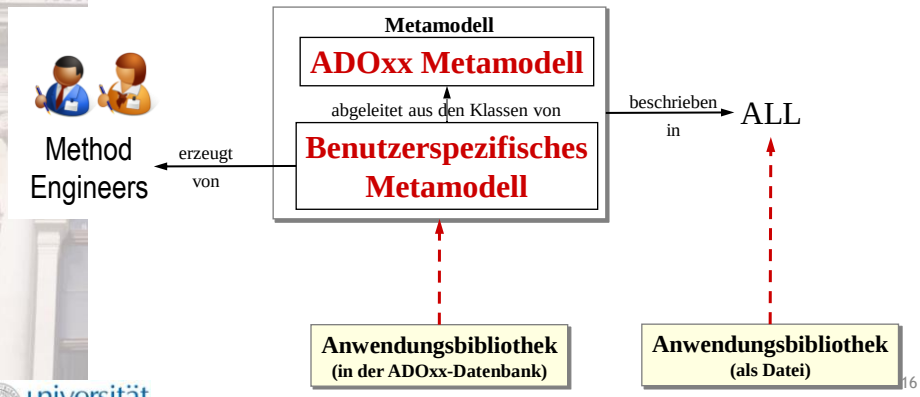
ATTRIBUTE <Verwendungszweck>
TYPE STRING
VALUE „Alle Objekte in dieser Aggregation gehören zusammen und müssen bei allen Funktionen als zusammengehörige Gruppe behandelt werden. "

FACET <MultiLineString>
VALUE 1

FACET <AttributeHelpText>
VALUE "Enter a description for documentation purposes."

FACET <AttributeRegularExpression>
VALUE ""

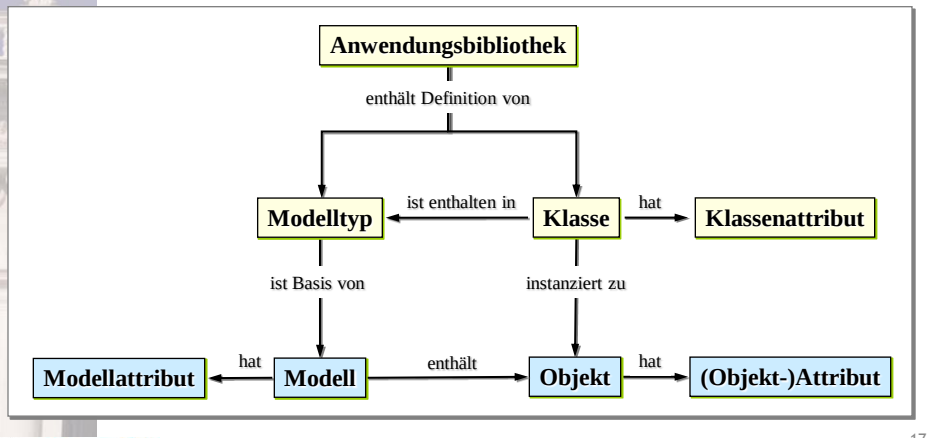
## Der Ansatzpunkt für das Customizing








## Die Methodenarchitektur in ADOxx



## Platform Specific Development Environment for the Implementation Process

ADOxx Platform

**ADOxx Method Development Environment**

- Implementation in ADOxx Library Language (ALL)
- **Implementation in ADOxx Method Development Tool**

ADOxx Modelling Language Implementation

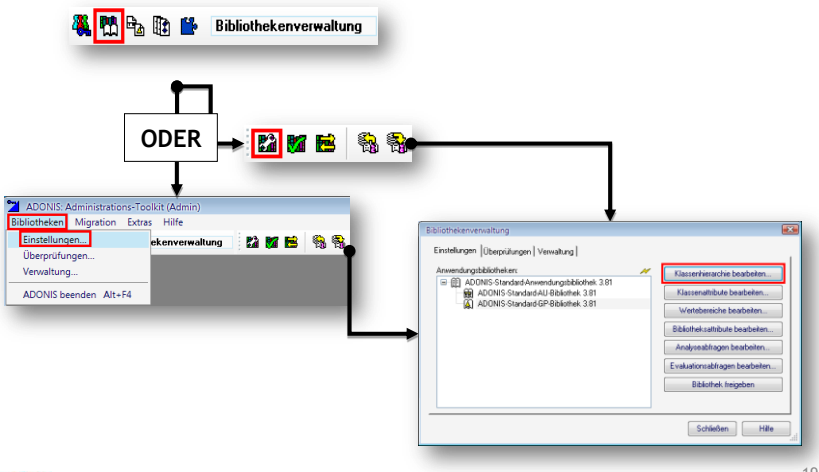
- ADOxx Model Hierarchy
- ADOxx Class Hierarchy
- ADOxx GraphRep

ADOxx Mechanisms and Algorithms

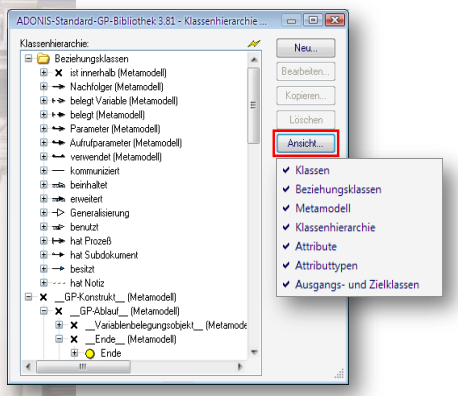
1. Basis Functionality
2. ADOscript
3. External Access
    1. ADOscript
    2. ADO Web-Service



## Einstieg in die Klassenhierarchieverwaltung



# Ansichten der Klassenhierarchie




***Klassen***

Es werden alle **sichtbaren** Klassen angezeigt

***Beziehungsklassen***

Es werden **alle** Beziehungsklassen angezeigt

***Metamodell***

Es werden **alle** Klassen angezeigt

***Klassenhierarchie***

Es werden alle Klassen innerhalb der **Vererbungshierarchie** angezeigt

***Attribute***

Es werden die **Attribute** der jeweiligen (Beziehungs-) Klassen angezeigt

***Attributtypen***

Es wird der zugehörige **Typ** des jeweiligen Attributs angezeigt

***Ausgangs- und Zielklassen***

Zeigt bei Beziehungsklassen an, zwischen welchen (Meta-) Klassen die jeweilige Beziehungsklasse **gezogen werden kann**.





# Icons der Klassenhierarchieverwaltung

Innerhalb der Klassenhierarchieverwaltung werden die nachfolgend aufgelisteten Icons verwendet:

**Klasse** (das Icon zeigt die grafische Definition des jeweiligen Objektes und kann daher variieren)

**Klasse** (ohne Definition der grafischen Darstellung)

**Attribut**

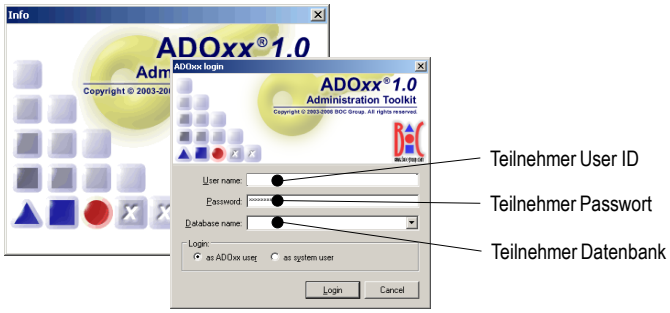
**Attribut** (abgeleitet von einer anderen Klasse)

**Klassenattribut**

**Klassenattribut (abgeleitet von einer anderen Klasse)**

# Übung

Öffnen der ADOxx Entwicklungsumgebung




1. Am Desktop sind 3 Icons angelegt:
   1. Admintoolkit mit einem user01
   2. Modelltoolkit mit einem user02 der auf die **StandardAB** eingetragen ist
   3. Modelltoolkit mit einem user03 der auf die **leere AB** eingetragen ist
2. Die Datenbank ist lokal und heißt adoxxdb
3. Der Admin hat das Kennwort adoxx die Kennwörter der user01 – 03 sind gleich dem Usernamen



# Platform Specific Development Environment for the Implementation Process

ADOxx Platform

ADOxx Method Development Environment
- Implementation in ADOxx Library Language (ALL)
- Implementation in ADOxx Method Development Tool

**ADOxx Modelling Language Implementation**
- **ADOxx Model Hierarchy**
- **ADOxx Class Hierarchy**
- **ADOxx GraphRep**

ADOxx Mechanisms and Algorithms
1. Basis Functionality
2. ADOscript
3. External Access
   1. ADOscript
   2. ADO Web-Service

## Modelling Method Implementation Instruments

| Modelling Method / Conceptualisation | Modelling Language | | | Modelling Procedure | Mechanisms and Algorithms | | |
|---|---|---|---|---|---|---|---|
| | Notation | Syntax | Semantic | | Basis Functions | Script Functions | External Access |
| Design | | | | | | | |
| Implementation | Model Type: Method-GraphRep, Icons | Model Type: ADOxx Library Language | | | Functional Description Table | | |
| | Object: GraphRep | **Object: ADOxx Library Language** | **Object: Class Hierarchy** | | Configuration Description | Scripting and Expression | ADO Web-Service (ADOscript API, Java API) |
| | Attribute: GraphRep | Attribute: ADOxx Library Language | | | | | |





## Klassenhierarchieverwaltung

Ansichtsmodi

Vererbung

Anlegen von Klassen (Objekte)

Anlegen von Relationen (Beziehungsklassen)

Klassenattribute vs. Instanzattribute

Datentypen und Attribute (Integer, String, Boolean etc.)

Anlegen von Attributen

## Vererbung

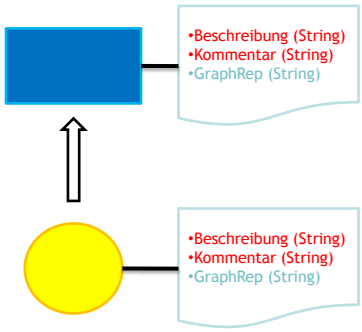
Ein wesentlicher Punkt bei der Erstellung / Erweiterung einer Modellierungsmethode ist die **Vererbung von Datenstrukturen**.

Über Vererbung können **Klassen**, **Relationen**, **Records** und **Attributprofile** erstellt / erweitert werden

Bei der Vererbung erhält die abgeleitete Klasse **alle Eigenschaften** (Attribute) der übergeordneten Superklasse.

Die Attributdefinitionen der beiden Klassen sind identisch und **können nicht geändert** werden.

Einzig die Werte der geerbten **Klassenattribute können unterschiedlich sein**.



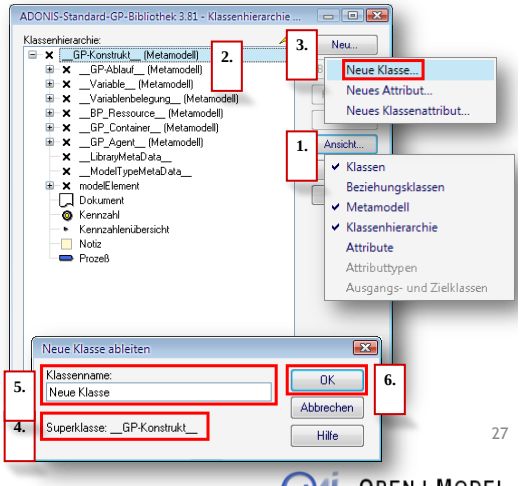





## Erstellung einer (abgeleiteten) Klasse

Um eine neue Klasse zu erstellen, sind die nachfolgend angegebenen Schritte notwendig:

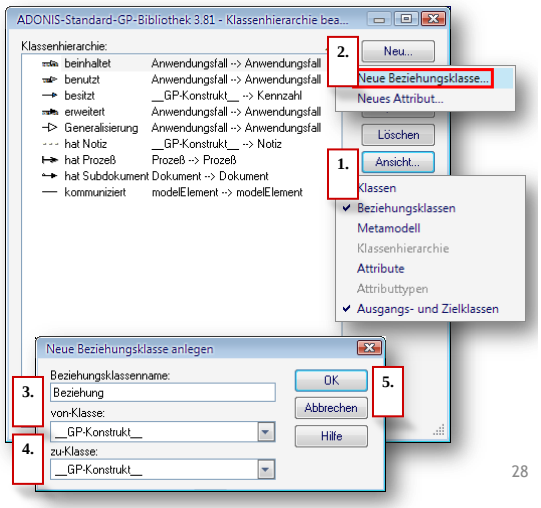
1. Wählen Sie unter „**Ansicht**“ die Optionen „**Klassen**“, „**Metamodell**“ und „**Klassenhierarchie**“ aus.
2. Wählen Sie die Klasse, deren Eigenschaften die neue Klasse erben soll, aus.
3. Unter „**Neu**“ wählen Sie die Option „**Neue Klasse**“.
4. Überprüfen Sie ob die richtige **Superklasse** ausgewählt wurde (s. Punkt 2).
5. Vergeben Sie einen neuen **Namen** für die neue Klasse.
6. Bestätigen Sie Ihre Eingabe durch Klicken des **OK-Buttons**.



## Erstellung einer Beziehungsklasse

Um eine neue Beziehungsklasse zu erstellen, sind die nachfolgend angegebenen Schritte notwendig

1. Wählen Sie unter „**Ansicht**“ die Optionen „**Beziehungsklassen**“ und „**Ausgangs- und Zielklassen**“ aus.
2. Unter „**Neu**“ wählen Sie die Option „**Neue Beziehungsklasse**“.
3. Vergeben Sie einen **Namen** für die neue Beziehungsklasse.
4. Wählen Sie die beiden Klassen aus, zwischen denen die neue Beziehungsklasse angelegt werden darf. Die „**von-Klasse**“ gibt hierbei die Ausgangsklasse und die „**zu-Klasse**“ die Zielklasse an.
5. Bestätigen Sie Ihre Eingabe durch Klicken des **OK-Buttons**.








## Attribute

ADOxx unterscheidet zwei Dimensionen von Attributen

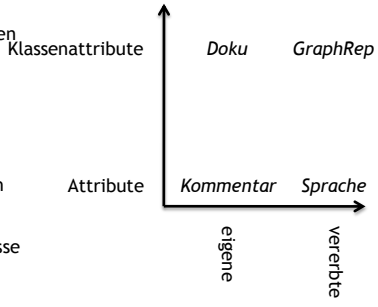
1. Attribute oder Klassenattribute
2. Vererbte Attribute oder eigene Attribute

Attribute sind Datencontainer die den Instanzen einer Klasse zur Verfügung stehen um Informationen zu speichern. Die enthaltenen Informationen können von Instanz zu Instanz variieren.

Klassenattribute sind Datencontainer deren Wert für alle Instanzen einer Klasse gleich sind (z. B. AttrRep, GraphRep etc.).

Vererbte Attribute sind Attribute die eine Klasse von einer Superklasse erhalten hat. Die Definition kann nicht verändert werden. Nur vererbte Klassenattribute können unterschiedliche Werte haben

Eigene Attribute sind nicht vererbt. Sie können unterschiedliche Definitionen haben



## Attributtypen

Die nachfolgenden Attributtypen stehen zur Auswahl:

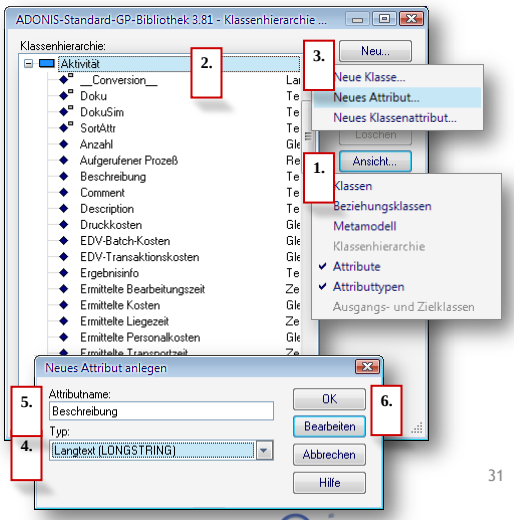
- **Attributprofilreferenz** - Integration von Attributprofilen
- **Aufzählung** - Einfache Auswahl aus einer vorgegebenen Werteliste
- **Aufzählungsliste** - Mehrfache Auswahl aus einer vorgegebenen Werteliste
- **Ausdruck** - Integration von Ausdrücken
- **Datum** - Eingabe von Datumswerten
- **Datum** und Zeit - Eingabe von Datums- und / oder Zeitwerten
- **Ganzzahl** - Eingabe von ganzen Zahlen
- **Gleitkommazahl** - Eingabe von Fließkommazahlen
- **Langtext** - Text mit einer max. Länge von 32.000 Zeichen
- **Programmaufruf** - Integration externer Dateien und / oder Programme
- **Referenz** - Verweise auf andere Modelle / Objekte
- **Tabelle** - Integration von Records
- **Text** - Text mit einer max. Länge von 3.200 Zeichen
- **Zeit** - Eingabe von Zeitwerten



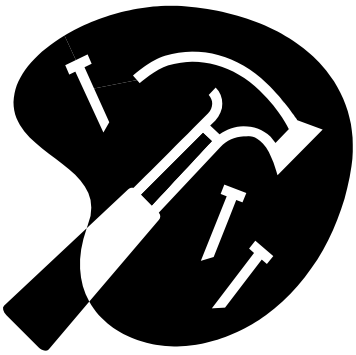
## Anlegen von (Klassen-) Attributen

Um ein neues Attribut zu erstellen, sind die nachfolgend angegebenen Schritte notwendig:

1. Wählen Sie unter „**Ansicht**“ die Optionen „**Klassen**“, „**Attribute**“ und „**Attributtypen**“ aus.
2. Wählen Sie die Klasse aus, der Sie das neue Attribut hinzufügen wollen.
3. Unter „**Neu**“ wählen Sie die Option „**Neues Attribut**“ oder „**Neues Klassenattribut**“.
4. Vergeben Sie einen neuen **Namen**
5. Wählen Sie den gewünschten Attributtyp aus.
6. Bestätigen Sie Ihre Eingabe durch Klicken des **OK-Buttons**.



## Übung





## Vorbereitung - Anwendungsbibliothek

1. Wechseln Sie in der „Bibliothekenverwaltung“ zu den „Verwaltung“ (letztes Icon der rechten Icongruppe).
2. Wählen Sie die „ADOxx-Standard-Anwendungsbibliothek“ aus.
3. Klicken Sie auf „Exportieren“.
4. Wählen Sie ein Zielverzeichnis und einen Dateinamen.
5. Achten Sie darauf, dass die Option „Inklusive zugehöriger Dateien“ ausgewählt ist und klicken Sie auf „exportieren“
6. Bestätigen Sie die Erfolgsmeldung durch anklicken des OK-Buttons.
7. Klicken Sie auf „Importieren“ und wählen Sie die zuvor exportierte Datei aus.
8. Klicken Sie auf „Importieren“ und bestätigen Sie die folgende Meldung mit „Ja“.
9. Vergeben Sie einen neuen Namen (z. B. durch Anhängen von „-Schulung“)
10. Wiederholen Sie die Schritte 8 und 9 weitere zwei mal.
11. Lassen Sie Modell- und Attributprofilgruppen automatisch erstellen (Abfrage mit „Ja“ bestätigen).
12. Beenden Sie den Vorgang durch Klicken auf den Schließen-Button des Importprotokolls.

## Aufgaben - Klassenhierachie

Erstellen Sie eine neue Klasse

- Name: Task
- Superklasse: __GP-Konsrukt__
- Attribute:
  - Beschreibung (Longstring, Standardwert „“)
  - Kommentar (Longstring, Standardwert „“)
  - Reihenfolge (Ganzzahl, Standardwert 0)
  - Relevanter Prozess (Modellreferenz, Modelltyp „Geschäftsprozeßmodell“, Max. Referenzen 1)
  - Externe Referenz (Programmauruf, Standardwert „“)



## Aufgaben - Klassenhierachie

Erstellen Sie eine neue Klasse

- Name: Step
- Superklasse: Task
- Attribute:
  - Relevante Aktivität (Objektreferenz, Modelltyp „Geschäftsprozeßmodell“, Klasse „Aktivität“. Max. Referenzen 1)

## Aufgaben - Klassenhierachie

Erstellen Sie eine neue Beziehungsklasse

- Name: besteht aus
- Grafische Darstellung:



- Von: Task
- Nach: Step
- Attribute: -



## Aufgaben - Klassenhierachie

Erstellen Sie eine neue Beziehungsklasse

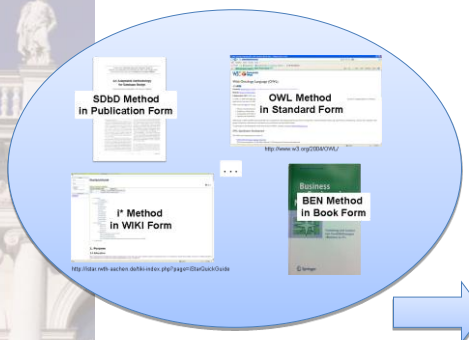
- Name: verbindet
- Grafische Darstellung:



- Von: Task, Step
- Nach: Task, Step
- Attribute: -

## Aufgaben - Meine Methode

**Startpunkt**




**Aufgabe:**

- Erstellen der benötigten Klassen
- Erstellen der benötigten Beziehungen
- Erstellen der benötigten Attribute

**Tipp:**
Verwendung der **leeren** Anwendungsbibliothek





## Modelling Method Implementation Instruments

| Modelling Method / Conceptualisation | Modelling Language | | | Modelling Procedure | Mechanisms and Algorithms | | |
|---|---|---|---|---|---|---|---|
| | Notation | Syntax | Semantic | | Basis Functions | Script Functions | External Access |
| Design | | | | | | | |
| Implementation | Model Type: Method-GraphRep, Icons | Model Type: ADOxx Library Language | | | Functional Description Table | | |
| | **Object: GraphRep** | Object: ADOxx Library Language | Object: Class Hierarchy | | Configuration Description | Scripting and Expression | ADO Web-Service (ADOscript API, Java API) |
| | **Attribute: GraphRep** | **Attribute: ADOxx Library Language** | | | | | |

## Ausgewählte Klassenattribute

AttrRep

GraphRep

Modellzeiger

Klassenkardinalität

Conversion



## Customizing von Klassenattributen

ADOxx Anwendungsbibliotheken enthalten eine Auswahl von Klassen, die gemeinsam das Metamodell abbilden:

- **Metaklassen**
- **Objektklassen**
- **Beziehungsklassen**
- **Tabellenklassen** (optional)
- **Attributprofilklassen** (optional)

Die Klassen selbst und ihre definierten Attribute können im Rahmen des allgemeinen Online-Customizings nicht verändert werden. Die meisten Attribute stehen dem ADOxx-*Modellierer* für seine Arbeit zur Verfügung. Einige Attribute haben jedoch Inhalte, die nur von ADOxx-*Administratoren* modifiziert werden können (z.B. das Aussehen der Klasse). Diese werden Klassenattribute genannt.

# Die Klassenattribute in ADOxx

Folgende Klassenattribute stehen für das Customizing bereit:

- **AttrRep:** Notebook-Definition (alle Klassen)
- **GraphRep:** Aussehen (Objekt- und Beziehungsklassen)
- **Modellzeiger:** Beziehungen zu anderen Modellen (Objektklassen)
- **Klassenkardinalität:** Beziehungsregeln (Objektklassen)
- **__Conversion__:** Überführung eines Objekts in ein anderes (nur ausgewählte Objektklassen)

**Hinweis:**
Nur die Klassenattribute *instanzierbarer* Klassen können mittels Online-Customizing verändert werden.



# Klassenattributfunktion starten

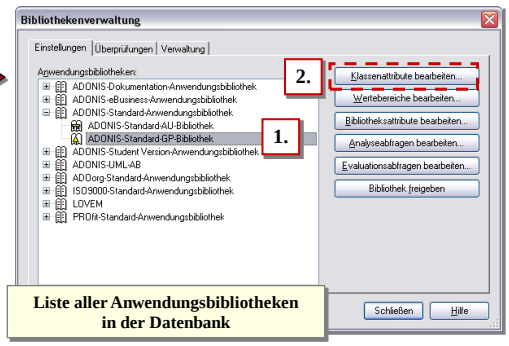
Zum Bearbeiten der Klassenattribute stehen Auswahldialoge in der Bibliothekenverwaltung zur Verfügung.

- SmartIcon

**oder**

- Menü **„Bibliotheken“**
- Menüpunkt **„Einstellungen“**

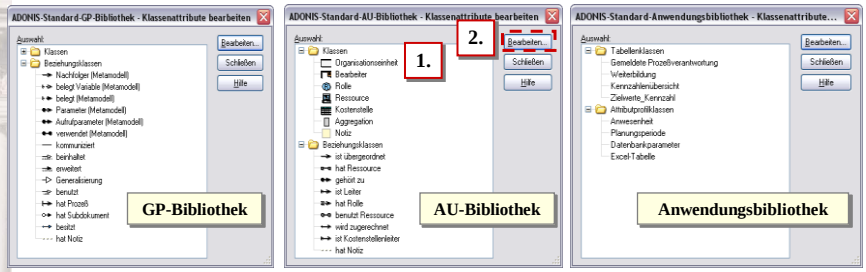
1. **Bibliothek oder Bibliothek auswählen**
2. Button **„Klassenattribute bearbeiten“**




**Liste aller Anwendungsbibliotheken in der Datenbank**

# Die Klassenlisten

Je nach selektierter Anwendungsbibliothek / Bibliothek sieht der Klassenauswahldialog unterschiedlich aus:




Um die Klassenattribute einer beliebigen Klasse zu bearbeiten:

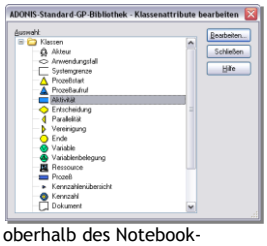
1. **Klasse selektieren**
2. **Button „Bearbeiten“**





# Klassenattribute bearbeiten

Die Klassenattribute werden in einem ADOxx-Notebook angezeigt.





Rechts oberhalb des Notebook-eintrags befindet sich ein Smart-Icon, mit dem eine Eingabeunterstützung für das jeweilige Klassenattribut geöffnet werden kann.

# AttrRep – Grundlagen

Das Klassenattribut „AttrRep“ kontrolliert das Vorhandensein und die Strukturierung des ADOxx-Notebooks dieser Klasse. Ist AttrRep nicht belegt, verfügt die Klasse über kein Notebook.

Im Notebook stehen folgende Elemente zur Verfügung:

- **Kapitel:** Jedes Notebook muss mindestens ein Kapitel haben, damit Attribute angezeigt werden können. Kapitel werden im fertigen Notebook durch die Kapitellaschen (Reiter) repräsentiert.
- **Attribute:** In die Kapitel werden die anzuzeigenden Attribute eingebettet. Verteilung und Reihenfolge werden ebenfalls in AttrRep definiert.
- **Gruppen:** Innerhalb eines Kapitels lassen sich Attribute zu Gruppen zusammenfassen.



# Die AttrRep-Befehle

`NOTEBOOK`

- Die Notebook-Definition muss mit diesem Befehl eingeleitet werden, um gültig zu sein. Dieser Befehl hat keine Parameter.

`CHAPTER „chapterName“`

- Mit diesem Befehl wird ein neues Notebook-Kapitel begonnen. Das Kapitel hat die Bezeichnung <*chapterName*> (Anm.: Ein Befehl „**`ENDCHAPTER`**“ ist nicht nötig).

`ATTRIBUTE „AttrName“`

- An dieser Stelle wird das Attribut <*AttrName*> in das Notebook eingefügt. Einige Attributtypen erlauben verschiedene Parameter, um die tatsächliche Darstellung anpassen zu können.

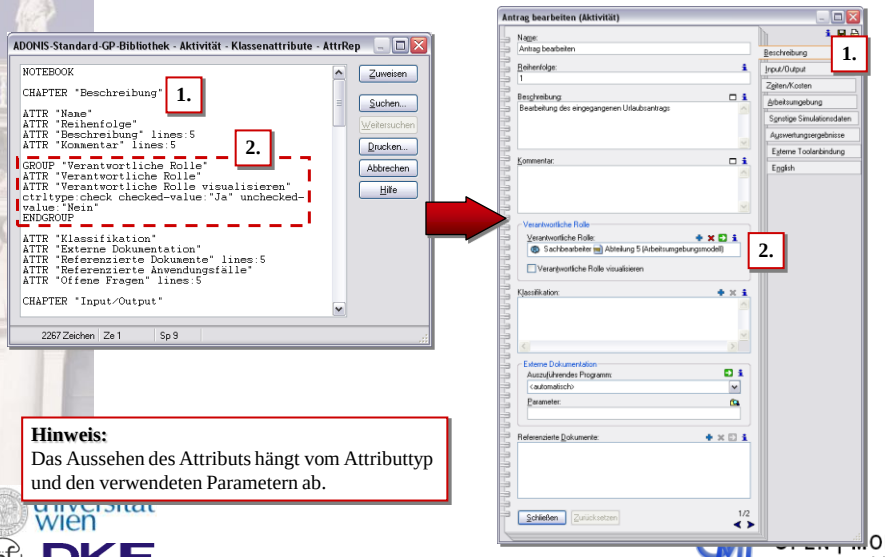
`GROUP „groupName“ / ENDGROUP`

- Die zwischen diesen beiden Befehlen aufgelisteten Attribute werden im Notebook von einer Gruppenbox umschlossen, welche die Bezeichnung <*groupName*> trägt.

`SET_ACCESS usergroup: userGroupSpec`

- Die dem Befehl folgenden Attribute werden nur für die Benutzergruppe <*userGroupSpec*> ange-zeigt. Diese Einschränkung wird mit „**`SET_ACCESS usergroup: all`**“ wieder aufgehoben.

# Beispiel für ein AttrRep




**Hinweis:**
Das Aussehen des Attributs hängt vom Attributtyp und den verwendeten Parametern ab.

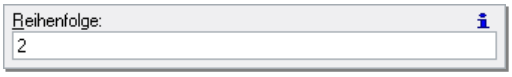




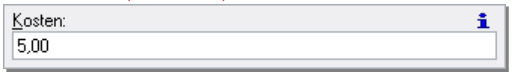
## Einige Attributtypen und ihre Darstellung

# Numerische Attribute

Ganzzahl (INTEGER):



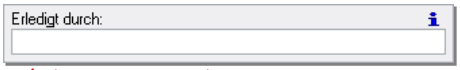

Gleitkommazahl (DOUBLE):



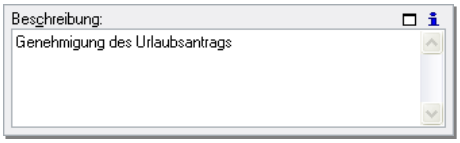

- Die Anzahl der Kommastellen wird bereits bei der Attributdefinition festgelegt und kann über das einfache Onlinecustomizing nicht mehr geändert werden.

## Einige Attributtypen und ihre Darstellung
# Textattribute

Text (STRING)




Langtext (LONGSTRING):



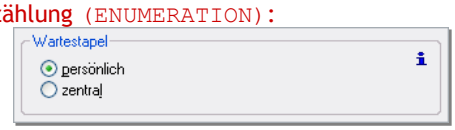

- Manche Textattribute werden bereits bei der Bibliotheksdefinition als **mehrzeilig** definiert. Über den Parameter **`lines`** wird festgelegt, mit wie vielen Zeilen ein solches Attribut im Notebook dargestellt wird.
- Mittels Parameter **`dialog`** können statt des Standard-Bearbeitungsdialogs verschiedene spezielle Eingabeunterstützungen aufgerufen werden.



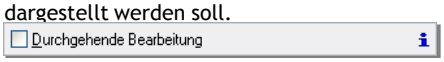


## Einige Attributtypen und ihre Darstellung
# Aufzählung / Aufzählungsliste

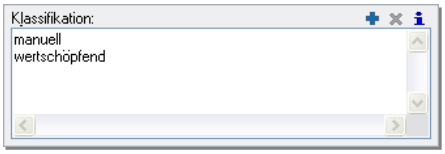
Aufzählung (ENUMERATION):




- Über den Parameter **`ctrltype`** wird festgelegt, ob die Aufzählung als Drop-down-Liste, Radiobuttons oder Checkbox (falls nur zwei Werte) dargestellt werden soll.




Aufzählungsliste (ENUMERATIONLIST):

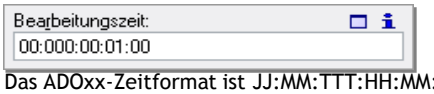


## Einige Attributtypen und ihre Darstellung
# Datum / Zeit

Datum `(DATE)`:

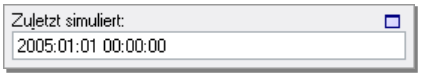

- Das ADOxx-Datumsformat ist JJJJ:MM:TT

Zeit `(TIME)`:




- Das ADOxx-Zeitformat ist JJ:MM:TTT:HH:MM:SS

Datum und Zeit `(DATETIME)`:





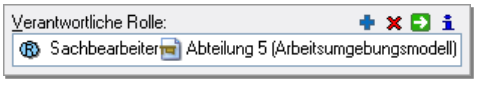

universität wien DKE



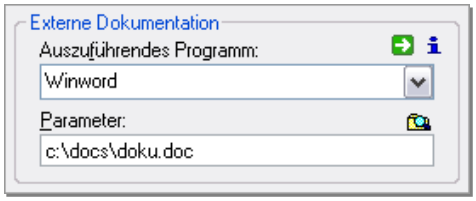
OPEN | MODEL Initiative www.openmodels.at

## Einige Attributtypen und ihre Darstellung
# Referenz / Programmaufruf

Referenz `(INTERREF)`:




Programmaufruf `(PROGRAMCALL)`:






universität wien DKE

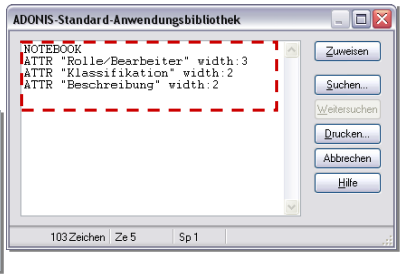


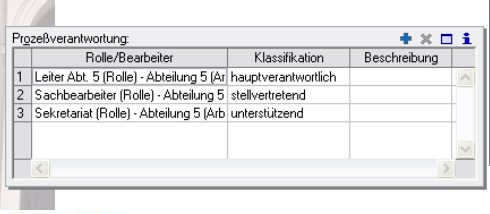
OPEN | MODEL Initiative www.openmodels.at

## Einige Attributtypen und ihre Darstellung
# Tabelle

Tabelle (TABLE):

- Tabellen werden in Notebooks entsprechend der Definition der dazugehörenden Tabellenklasse dargestellt.
- Im AttrRep der Tabellenklasse selbst lassen sich folgende Einstellungen treffen:
  - welche Spalten angezeigt werden
  - in welcher Reihenfolge
  - wie breit (relative Breite):
    Parameter **width**



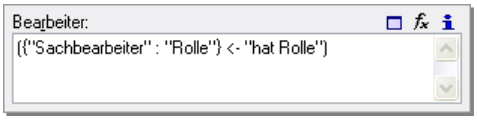








## Einige Attributtypen und ihre Darstellung
# Ausdruck / Attributprofilreferenz

Ausdruck (EXPRESSION):

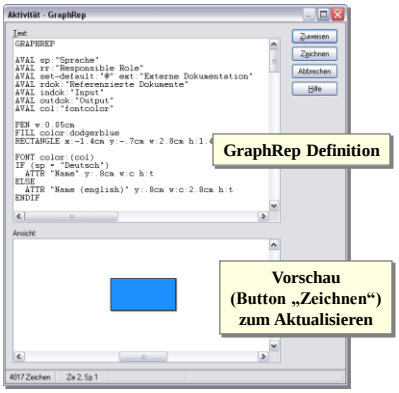



Attributprofilreferenz (ATTRIBUTEPROFILEREFEREN):

# GraphRep – Grundlagen

Das Klassenattribut „GraphRep“ kontrolliert das Aussehen einer Objekt- oder Beziehungsklasse für das ADOxx-Geschäftsprozess-management-Toolkit. Anders als AttrRep muss GraphRep für alle instanzierbaren Modellierungsklassen belegt werden.

Für das GraphRep steht ein eigener Unterstützungsdialog mit Vorschaufenster bereit:








# Einige GraphRep-Befehle (1)

GRAPHREP

- Die GraphRep-Definition muss mit diesem Befehl eingeleitet werden, um gültig zu sein. Der Parameter **layer** legt fest, ob ein Objekt auf der Zeichenfläche oberhalb oder unterhalb anderer Objekte zu liegen kommt, **sizing** ob das Objekt größenveränderbar ist.

SHADOW

- Bestimmt, ob die Modellierungsklasse einen Schatten erhält oder „flach“ gezeichnet wird.

PEN

- Bestimmt die Strichstärke (des Konnektors bzw. des Objektrahmens).

FILL

- Definiert Füllfarbe, Füllstil und Transparenz der Füllung von Objekten.

ATTR

- Zeigt einen Attributwert auf der Zeichenfläche an (z.B. einen Objektnamen).

## Einige GraphRep-Befehle (2)

POINT
- Zeichnet einen Punkt.

LINE / POLYLINE
- Eine gerade Linie (**LINE**) bzw. eine einmal oder mehrfach geknickte Linie (**POLYLINE**).

CURVE / ARC
- Zeichnet eine Kurve nach einer mathematischen Funktion / einen Ellipsenbogen.

POLYGON
- Zeichnet ein Vieleck aus geraden Linien, bei dem jeder Eckpunkt einzeln definiert werden muss.

RECTANGLE / ROUNDRECT / ELLIPSE / PIE
- Ein Rechteck, ein Rechteck mit abgerundeten Ecken, eine Ellipse, ein Ellipsensegment.

COMPOUND
- Eine ausgefüllte, zusammengesetzte Form (aus **LINE**, **POLYLINE** und **CURVE**-Elementen).



## Einige GraphRep-Befehle (3)

TEXT
- Ermöglicht das Darstellen von feststehendem Text (Buchstaben, Symbole …).

FONT
- Bestimmt die Schrifttype von eingefügtem Text.

BITMAP
- Ermöglicht das Einbinden einer Grafik (*.BMP-Format) in die Darstellung.

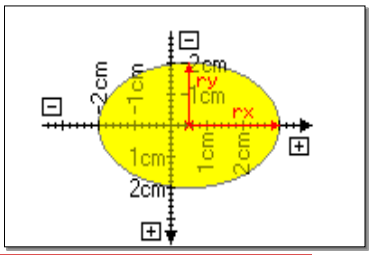
TABLE
- Erstellt eine Tabelle für eine strukturierte Attributdarstellung auf einem Objekt.

**Hinweis:**
Grafische Elemente können für komplexe Grafiken beliebig kombiniert werden!

# Das GraphRep-Koordinatensystem

Um die exakte Position von GraphRep-Darstellungselementen festlegen zu können, steht ein Koordinatensystem zur Verfügung. Dieses ist folgendermaßen aufgebaut:

- Der Nullpunkt des Koordinaten-systems liegt in der Mitte
- Links und oben (!) liegen die negativen Wertebereiche
- Rechts und unten befinden sich die positiven Wertebereiche



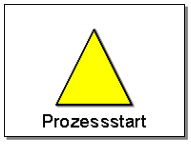

**Hinweise:**
- Die **Maßeinheit** bei Positionen und Ausmaßen (cm oder pt) ist unbedingt anzugeben. Maße in Pixel sind nicht möglich.
- Der Drehsinn des Koordinatensystems verläuft **gegen** den Uhrzeigersinn!





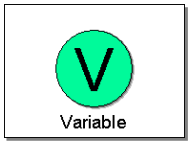
# GraphRep-Beispiele: Basisformen

```
GRAPHREP
PEN w:0.05cm
FILL color:yellow
POLYGON 3 x1:-.7cm y1:.7cm x2:.7cm y2:.7cm x3:0cm y3:-.7cm
ATTR "Name" y:.8cm w:c:2.8cm h:t
```




```
GRAPHREP
PEN w:0.05cm
FILL color:dodgerblue
RECTANGLE x:-1.4cm y:-.7cm w:2.8cm h:1.4cm
ATTR "Name" y:.8cm w:c h:t
```


```
GRAPHREP
FILL color:mediumspringgreen
ELLIPSE rx:0.70cm ry:0.70cm
ATTR "Name" y:0.8cm w:c:1.4cm h:t
FONT "Arial" h:32pt color:black
TEXT "V" y:0.13cm w:c h:c
```



# GraphRep-Steuerungsbefehle

SET
- Belegt Variablen wahlweise mit Konstanten oder mit Ausdrücken, die wiederum Variablen enthalten können.

AVAL
- Belegt Laufzeitvariablen mit Attributwerten der aktuellen Klasse.

IF / ELSIF / ELSE / ENDIF
- Ermöglicht – in Zusammenhang mit Variablen – bedingte grafische Darstellungsformen.

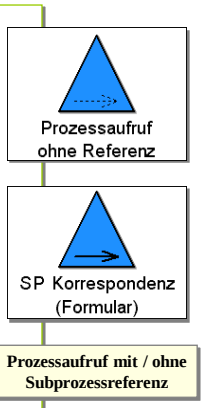
BITMAPINFO
- Liest die Höhe und Breite einer Bitmap-Datei aus, um sie korrekt darstellen zu können.

TEXTBOX / ATTRBOX
- Variablenzuweisungselemente mit den gleichen Parametern wie die korrespondierenden grafischen Elemente TEXT und ATTR. Der Unterschied ist, dass hier nichts gezeichnet wird, sondern dass die Parameter bestimmten Variablen zugewiesen werden.



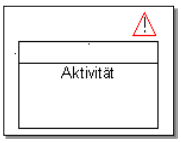
# GraphRep-Beispiele: Bedingte Darstellung (1)

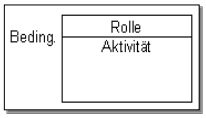
```
GRAPHREP
AVAL col:"fontcolor"
AVAL set-default:"x" p:"aufgerufener Prozeß"
AVAL sub:"aufgerufener Prozeß"
AVAL i:"Reihenfolge"
AVAL sn:"Subprocessname"
FILL color:dodgerblue
PEN w:0.05cm
POLYGON 3 x1:-.7cm y1:.7cm x2:.7cm y2:.7cm x3:0cm y3:-.7cm
SHADOW mode:off
IF (NOT LEN p)
  PEN style:dot
ENDIF
LINE x1:-.4cm y1:.5cm x2:.4cm y2:.5cm
LINE x1:.1cm y1:.4cm x2:.4cm y2:.5cm
LINE x1:.1cm y1:.6cm x2:.4cm y2:.5cm
FONT color:(col)
IF (sub = "")
  ATTR "Name" y:.8cm w:c:2.8cm h:t
ELSE
  FONT "Arial" h:8pt bold
  ATTR "aufgerufener Prozeß" y:(texty2 + .1cm) w:c:2.8cm h:t format:"%m"
  FONT
ENDIF
```



## GraphRep-Beispiele:
# Bedingte Darstellung (2)

```
GRAPHREP
AVAL set-default:"Modellierung abgeschlossen" b:"Status"
SHADOW off
FILL style:null
POLYGON 4 x1:-1.54cm y1:0.92cm x2:1.54cm y2:0.92cm
          x3:1.54cm y3:-0.98cm x4:-1.54cm y4:-0.98cm
LINE x1:-1.54cm y1:-0.50cm x2:1.54cm y2:-0.50cm
IF (b = "Modellierung nicht abgeschlossen")
LINE x1:1.25cm y1:-1.5cm x2:1.25cm y2:-1.3cm
LINE x1:1.25cm y1:-1.22cm x2:1.25cm y2:-1.18cm
PEN color:red
POLYGON 3 x1:1cm y1:-1.1cm x2:1.25cm y2:-1.6cm
x3:1.50cm y3:-1.1cm
ENDIF
```




**Bedingung erfüllt / nicht erfüllt**





## GraphRep-Beispiele:
# Tabellen

```
GRAPHREP
sizing:asymmetrical
SHADOW off
PEN color:black
FILL style:null
TABLE x:-3.5cm y:-2cm w:7cm h:4cm
cols:3 rows:4
w1:1.3cm w2:50% w3:50%
h1:1cm h2:0.5cm h3:0.5cm h4:100%
```

| Stabstelle | | |
|---|---|---|
| | Name der Person | |
| Nr | Planstellen | Iststellen |
| Mitarbeiter1<br>Mitarbeiter2 | | |

**Tabelle mit 4 Zeilen und 3 Spalten**

**Hinweise:**
Beim händischen Verändern der Tabellengröße ändert sich nur die Größe der Felder **mit Prozentangaben**. Felder mit absoluten Werten bleiben immer gleich groß.

## GraphRep-Beispiele:
# Tabellenrahmen

Tabellen können mit oder ohne Rahmen gezeichnet werden. Rahmen werden als Linien definiert, die von einem Eckpunkt einer Tabellenzelle zu einem anderen gezogen werden.

Dabei hat beispielsweise die linke obere Ecke der Tabelle die Koordinate tabx0, taby0, die rechte obere Ecke der ersten Zelle tabx1, taby0 etc.

```
LINE x1:(tabx0) y1:(taby0) x2:(tabx3) y2:(taby0)
LINE x1:(tabx0) y1:(taby1) x2:(tabx3) y2:(taby1)
LINE x1:(tabx0) y1:(taby2) x2:(tabx3) y2:(taby2)
LINE x1:(tabx0) y1:(taby3) x2:(tabx3) y2:(taby3)
LINE x1:(tabx0) y1:(taby4) x2:(tabx3) y2:(taby4)

LINE x1:(tabx0) y1:(taby0) x2:(tabx0) y2:(taby4)
LINE x1:(tabx1) y1:(taby1) x2:(tabx1) y2:(taby3)
LINE x1:(tabx2) y1:(taby2) x2:(tabx2) y2:(taby3)
LINE x1:(tabx3) y1:(taby0) x2:(tabx3) y2:(taby4)
```

| Stabstelle | | |
|---|---|---|
| | Name der Person | |
| Nr | Planstellen | Iststellen |
| Mitarbeiter1 | | |
| Mitarbeiter2 | | |

**Tabelle mit 4 Zeilen und 3 Spalten; Linien nur teilweise ausgeführt**





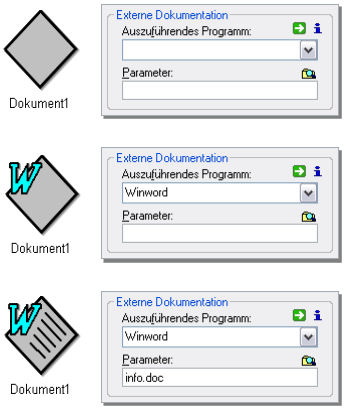
## GraphRep-Beispiele:
# Komplexe, attributabhängige Darstellung

```
GRAPHREP
AVAL a:"Externe Dokumentation"
PEN w:0.1cm
FILL r:200 g:200 b:200
POLYGON 4 x1:0cm y1:-1cm x2:1cm y2:0cm
          x3:0cm y3:1cm  x4:-1cm y4:0cm
ATTR "Name" y:1.2cm w:c:2.8cm h:t
IF (search(lower(a),"winword",0) >= 0)
  PEN w:0.07cm
  FILL r:0 g:255 b:255
  ...

  IF (search(lower(a),".doc",0) >=0)
    ...
  ENDIF
ELSIF (search(lower(a),"powerpnt",0) >= 0)
  ...
ENDIF
```

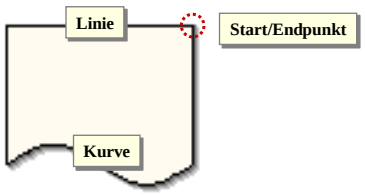
**Suche nach Zeichenmustern**

**Verschachtelte Bedingungen**



## GraphRep-Beispiele:
# Zusammengesetzte Darstellung

```
GRAPHREP
COMPOUND 2
LINE x1:1.0cm y1:-.7cm x2:-1.0cm y2:-.7cm
CURVE "t" f:(t) g:(-.2*sin(3.14*(t+1))+.7) from:-1 to:1
```




**Hinweise:**
- Die Darstellung ist zusammengesetzt **aus *einer* Linie und *einer* Kurve**.
- Der Endpunkt des vorhergehenden Elements ist der Anfangspunkt des folgenden.
- Bei Bedarf wird die **Verbindung** zwischen den Elementen in einem COMPOUND-Objekt **automatisch** gezogen (Zeichenreihenfolge ist wichtig!).

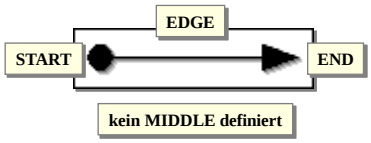


# GraphRep-Definition für Beziehungsklassen

Prinzipiell ist die GraphRep-Definition von Objekt- und Beziehungsklassen gleich. Für sichtbare Beziehungen („Konnektoren“) stehen jedoch zusätzliche Schlüsselworte zur Verfügung:

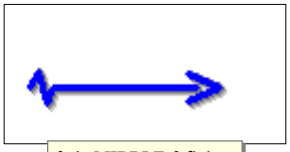
`EDGE`
- Bestimmt das Aussehen der Konnektorkante (Linie).

`START / MIDDLE / END`
- Diese Befehle definieren das Aussehen der markanten Konnektorpunkte. Ist **`MIDDLE`** definiert, kann der Konnektormittelpunkt auch verschoben werden.



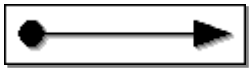
## GraphRep-Beispiele:
# Konnektoren

```
GRAPHREP
PEN color:lightblue w:.08cm
EDGE start-trans:-.3cm end-trans:-.3cm
START
POLYLINE 4 x1:0cm y1:0cm x2:-.1cm y2:.18cm
x3:-.2cm y3:-.18cm x4:-.3cm y4:0cm
END
POLYLINE 3 x1:-.4cm y1:.15cm x2:0cm y2:0cm
x3:-.4cm y3:-.15cm
```



**kein MIDDLE definiert**

```
GRAPHREP
START
FILL color:black
ELLIPSE x:-.1cm rx:.1cm ry:.1cm
END
LINE x1:-.3cm y1:.1cm x2:0cm y2:0cm
LINE x1:-.3cm y1:-.1cm x2:0cm y2:0cm
```

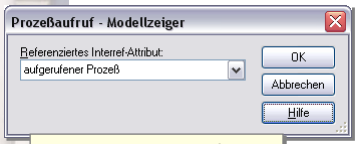




# Das Klassenattribut „Modellzeiger“

Das Klassenattribut „Modellzeiger“ (Typ: `INTERREF`) steuert die Fähigkeit von ADOxx, über ein Objekt in einem Modell direkt zu einem anderen Modell zu gelangen (z.B. beim Aufrufen eines Subprozesses oder eines Dokumentenpools).

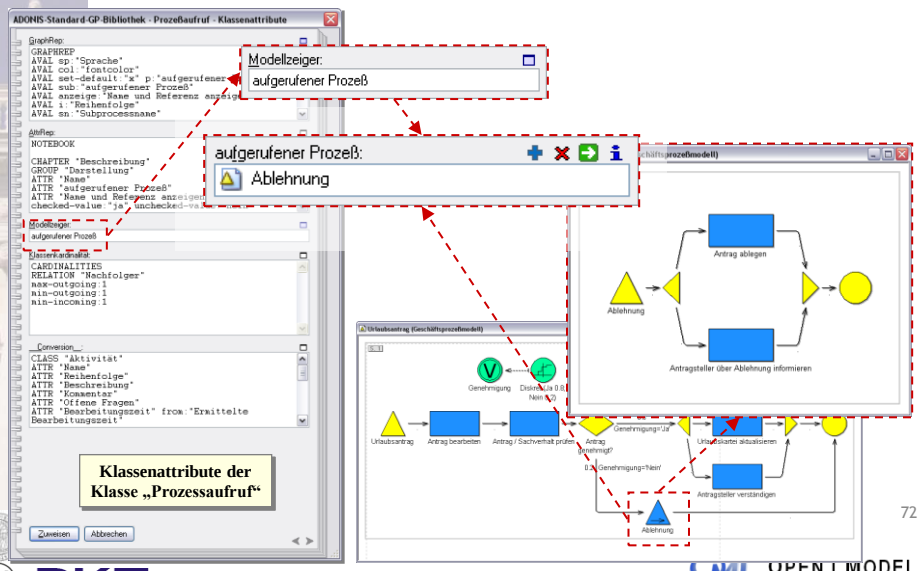
Im Modellzeiger-Attributfeld wird der Name jenes Attributs eingetragen, dessen Wert eine Referenz auf ein anderes Modell bzw. ein Objekt in einem anderen Modell darstellt.




**Eingabeunterstützung für das Klassenattribut „Modellzeiger“**

**Hinweise:**
Der Sprung zum referenzierten Modell kann mittels *<Strg>+Doppelklick* auf das Objekt durchgeführt werden. Diese Funktion ist unabhängig von eventuell beim Objekt dargestellten Hyperlinks.

# Modellzeiger: Beispiel










# Das Klassenattribut „Klassenkardinalität“

Das Klassenattribut „Klassenkardinalität“ enthält die Kardinalitätsdefinition der aktuellen Objektklasse. Die Kardinalität einer Klasse beschreibt

- **die minimal/maximal erlaubte Anzahl von Objekten einer Klasse** und
- **die minimale/maximale Anzahl von Beziehungen eines Beziehungstyps**, welche zu einem Objekt hin- bzw. von ihm wegführen dürfen.

Sind keine Kardinalitäten definiert, unterliegt die Klasse auch keiner Beschränkung.

**Hinweise:**

- Je nach Customizing kann eine Kardinalitätenprüfung im Toolkit bei jedem Speichern eines Modells oder aber nur beim händischen Aufrufen der Funktion durchgeführt werden.
- Eine detaillierte Beschreibung der Kardinalitätsdefinition entnehmen Sie bitte dem ADOxx-Handbuch Band IV (Methodendefinition und Administration).

# Die Befehle der Kardinalitätsdefinition

`CARDINALITIES`

- Die Kardinalitätsdefinition muss mit diesem Befehl eingeleitet werden, um gültig zu sein.
  Dieser Befehl hat keine Parameter.

`RELATION „`*`RelationName`*`"`

- Schränkt die nachfolgenden Regel-Parameter auf Konnektoren der Klasse <*RelationName*> ein.

`FROM_CLASS „`*`ClassName`*`" / TO_CLASS „`*`ClassName`*`"`

- Schränkt die nachfolgenden Regel-Parameter auf Beziehungen zur Klasse <*ClassName*> ein.





# Die Parameter der Kardinalitätsdefinition

`min-objects / max-objects`

- Gibt an, wie viele Objekte dieser Klasse minimal/maximal im Modell vorhanden sein dürfen.

`min-relations / max-relations`

- Definiert die minimale/maximale Anzahl von Konnektoren, welche mit diesem Objekt verbunden sein dürfen.

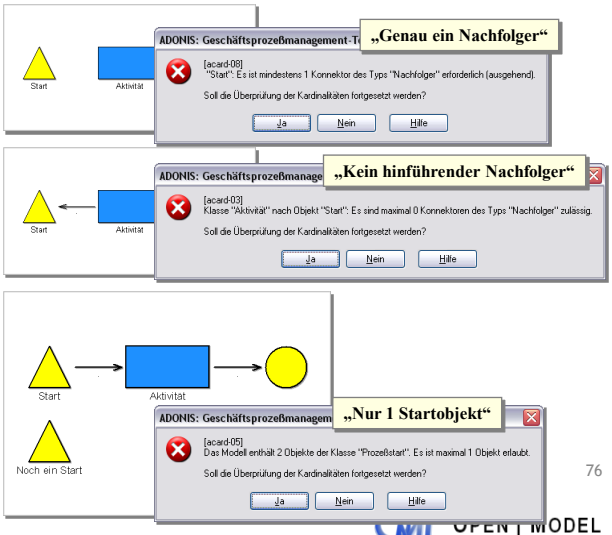
`max-outgoing / min-outgoing / max-incoming / min-incoming`

- Schränkt die erlaubte Anzahl ein-/ausgehender Konnektoren ein; wahlweise:
  - generell oder
  - mit vorangestelltem Befehl **`RELATION`** nur für eine bestimmte Beziehungsklasse oder
  - mit vorangestelltem Befehl **`FROM_CLASS`** bzw. **`TO_CLASS`** nur für eine bestimmte verbundene Objektklasse.

# Klassenkardinalität: Beispiele und Fehlermeldungen

```
CARDINALITIES
max-objects:1
min-objects:1
RELATION "Nachfolger"
max-outgoing:1
min-outgoing:1
max-incoming:0
```

**Kardinalitätsdefinition der Klasse „Prozessstart“**






# Das Klassenattribut „__Conversion__“

Das Klassenattribut „Conversion“ definiert und steuert die Umwandlung eines Modellierungsobjekts von einer Klasse in eine andere Klasse.

Bei der Umwandlung wird zunächst ein neues Objekt der definierten Klasse erzeugt. Anschließend werden automatisch jene Attributwerte in das neue Objekt übertragen, welche im Conversion-Attribut definiert sind. Zuletzt wird das alte Objekt gelöscht und das neue nimmt seinen Platz ein.

**Hinweise:**

- Die prinzipielle Möglichkeit zur Umwandlung muss zuvor im Metamodell selbst definiert worden sein, um später per Customizing darauf zugreifen zu können.
- Der Modellierer erhält dann einen entsprechenden Kontextmenüeintrag im ADOxx-GPM-Toolkit .

# Die Conversion-Befehle und Parameter

`CLASS „ClassName"`

- Legt fest, dass die aktuelle Objektklasse in die Klasse <*ClassName*> umgewandelt werden kann. Mehrere Zielklassen zu definieren ist dabei möglich.

`ATTR „AttrName"`

- Definiert, welche Attributwerte bei der Umwandlung auf die Zielklasse übertragen werden sollen.

`from`

- Dieser Parameter kommt zur Anwendung, wenn Information vom Ausgangsobjekt auf das Zielobjekt übertragen werden soll, die beiden Attribute aber verschiedene Namen haben. Dabei definiert **`from`** den Namen des Ausgangsattributs.

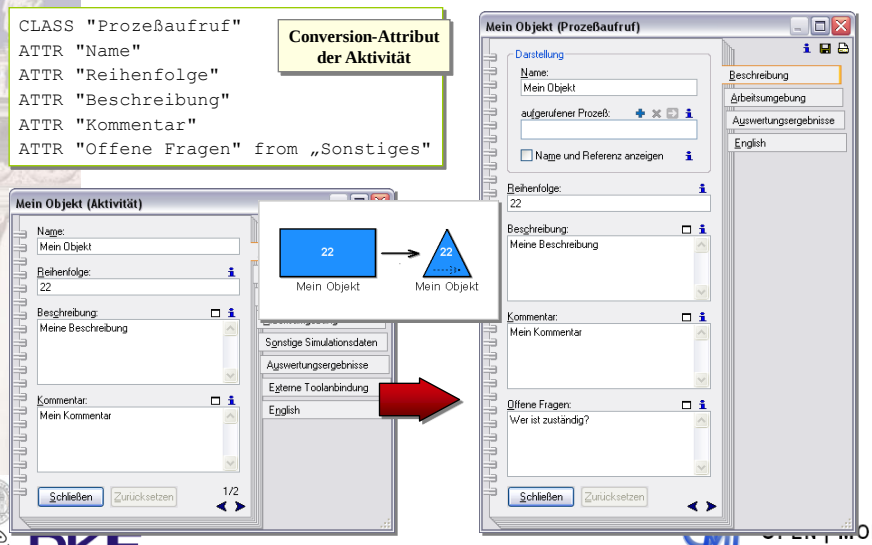
**Hinweis:**
Eine detaillierte Beschreibung der Conversion-Grammatik entnehmen Sie bitte dem ADOxx-Handbuch Band IV (Methodendefinition und Administration).



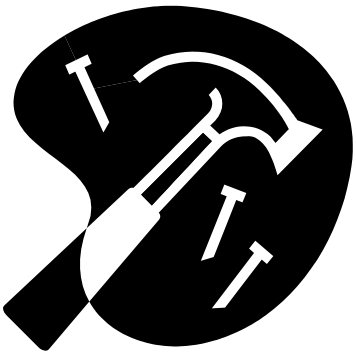
## Conversion-Beispiel:
# Umwandlung Aktivität ⇨ Prozessaufruf

```
CLASS "Prozeßaufruf"
ATTR "Name"
ATTR "Reihenfolge"
ATTR "Beschreibung"
ATTR "Kommentar"
ATTR "Offene Fragen" from „Sonstiges"
```

**Conversion-Attribut der Aktivität**



## Übung







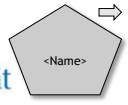
## Aufgaben - Klassenattribute

HINWEIS: Sofern nicht anders angegeben, soll das Attribut in der Standarddarstellung angezeigt werden.

Erstellen Sie für die Klasse Task ein neues Notebook

- Kapitel: Beschreibung
  - Attribut „Name“
  - Attribut „Beschreibung“ (Zeilen: 5)
  - Attribut „Kommentar“ (Zeilen: 5)
  - Attribut „Reihenfoge“
- Kapitel: Referenzen
  - Attribut „Relevanter Prozess“
  - Attribut „Externe Referenz“

Adaptieren Sie die grafische Darstellung entsprechend nachstehender Grafik:




**Anmerkung:** Der Pfeil ist nur sichtbar, wenn das Attribut „Relevanter Prozess“ befüllt wurde. Diese Pfeil soll dann einen Hotspot auf den referenzierten Prozess beinhalten.

## Aufgaben - Klassenattribute

HINWEIS: Sofern nicht anders angegeben, soll das Attribut in der Standarddarstellung angezeigt werden.

Erstellen Sie für die Klasse Step ein neues Notebook

- Kapitel: Beschreibung
  - Attribut „Name“
  - Attribut „Beschreibung“ (Zeilen: 5)
  - Attribut „Kommentar“ (Zeilen: 5)
  - Attribut „Reihenfoge“
- Kapitel: Referenzen
  - Attribut „Relevante Aktivität“
  - Attribut „Externe Referenz“

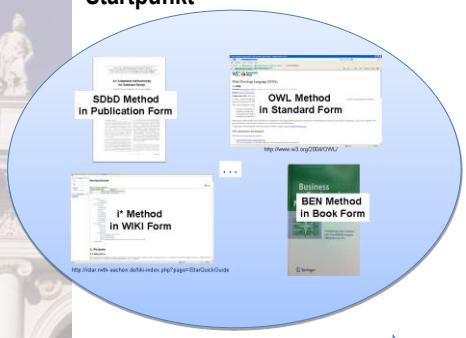
Adaptieren Sie die grafische Darstellung entsprechend nachstehender Grafik:


**Anmerkung:** Der Pfeil ist nur sichtbar, wenn das Attribut „Relevante Aktivität“ befüllt wurde. Diese Pfeil soll dann einen Hotspot auf die referenzierte Aktivität beinhalten.



## Aufgaben - Meine Methode

**Startpunkt**




**Aufgabe:**

- Erstellen der Notebooks
- Erstellen der Notation
- Benutzerfreundliches Verweisen auf Modell (Modellzeiger)
- Hinzufügen von Semantik (Kardinalitäten, Conversion)

## Modelling Method Implementation Instruments

| Modelling Method Conceptualisa... | Modelling Language | | | Modelling Procedure | Mechanisms and Algorithms | | |
|---|---|---|---|---|---|---|---|
| | Notation | Syntax | Semantic | | Basis Functions | Script Functions | External Access |
| Design | | | | | | | |
| Implementation | Model Type: Method-GraphRep, Icons | Model Type: ADOxx Library Language | | | Functional Description Table | | |
| | Object: GraphRep | Object: ADOxx Library Language | Object: Class Hierarchy | | Configuration Description | Scripting and Expression | ADO Web-Service (ADOscript API, Java API) |
| | Attribute: GraphRep | Attribute: ADOxx Library Language | | | | | |





## Modelltypdefinition

Modell-GraphRep

Modelltypen

Modi (Dokumentation, Modellierung)

# Das Bibliotheksattribut „Modi“

Im Bibliotheksattribut „Modi“ können Modelltypen, Modelltypen-Gruppen und (Ansichts-) Modi für Modelltypen definiert werden.

- Ein **Modelltyp** legt eine **Teilmenge aller instanzierbaren Klassen und Beziehungen** fest. Jedes Modell ist von einem bestimmten Modelltyp, der nach dem Anlegen des Modells nicht mehr geändert werden kann.
- **Modelltyp-Gruppen** sollten definiert werden, falls die Anwendungsbibliothek aus vielen verschiedenen Modelltypen besteht. Damit werden **verwandte Modelltypen zusammengefasst** und die Übersichtlichkeit erhöht.
- Ein **Modus** ist eine **weitere Einschränkung eines Modelltyps**. Er definiert eine Teilmenge der dem Modelltyp zugewiesenen Modellierungsklassen und vereinfacht so die Modellierung durch Ausblenden nicht benötigter Klassen. Im Gegensatz zum Modelltyp kann der Modus eines Modells jederzeit gewechselt werden.

**Hinweis:**
Eine detaillierte Beschreibung der Modi-Grammatik entnehmen Sie bitte dem ADOxx-Handbuch Band IV (Methodendefinition und Administration).



# Einige Modi-Befehle: Allgemeines

`GENERAL order-of-classes: OrderOfClasses`

- Legt fest, ob die Reihenfolge der Modellierungsklassen in der Modellierungsleiste vom Metamodell übernommen (<*OrderOfClasses*> = „**`default`**“) oder bei der Definition jedes Modelltyps händisch festgelegt werden soll („**`custom`**“).

`METHOD graphrep: „attrName“`

- Leitet ein Methodendiagramm ein.

`GROUP „GroupName“`

- Definiert eine Gruppe von Modelltypen mit dem Namen <*GroupName*>.

`graphrep: „attrName“`

- Bestimmt die grafische Gestaltung des Methodendiagramms. <*attrName*> definiert ein Attribut, dessen Inhalt wiederum das Aussehen des Diagramms festlegt (in der ADOxx GraphRep-Sprache).

**Hinweis:**
Die GraphRep-Sprache und ihre Grammatik ist Teil der Trainingseinheit „Klassenattribute“.

## Modi-Beispiele:
# Methodendiagramm

```
GENERAL order-of-classes:custom
METHOD graphRep:"Method GraphRep"
{
  GROUP "Simulation"
  {
    MODELTYPE "Geschäftsprozeßmodell"
    MODELTYPE "Arbeitsumgebungsmodell"
  }
  GROUP "Alle Modelltypen"
  {
    MODELTYPE "Geschäftsprozeßmodell"
    MODELTYPE "Arbeitsumgebungsmodell"
    MODELTYPE "Prozeßlandkarte"
    MODELTYPE "Dokumentenmodell"
    MODELTYPE "Anwendungsfalldiagramm"
  }
}
```

**Methodendiagramm mit zwei Methoden für die ADOxx-Standard-Anwendungsbibliothek**



## Einige Modi-Befehle:
# Modelltypen

`MODELTYPE „modelTypeName" from MTSource`

- Dieser Befehl definiert einen Modelltyp <*modelTypeName*> und übernimmt dazu die Klassen und Beziehungsklassen von der Quelle <*MTSource*> (**`all`**, **`none`** oder ein anderer Modelltyp)

`plural: „modelTypePluralName"`

- Definiert den Mehrzahl-Namen des Modelltyps.

`bitmap: „fileName"`

- Bestimmt ein grafisches Symbol für Auswahllisten (<*fileName*> = Pfad und Dateiname; Backslashes müssen mit Backslashes maskiert werden).

`attrrep: „attrName"`

- Verleiht dem Modelltyp das in der Bibliothek definierte Notebook <*attrName*> mit Modellattributen.

`INCL / EXCL`

- Fügt dem mit **`MODELTYPE`** angelegten Modelltyp Klassen hinzu (außer bei **`all`**) bzw. entfernt diese (außer bei **`none`**).

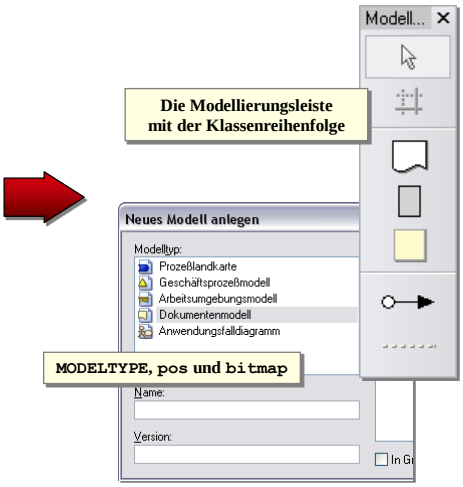
`pos / not-simulateable`

- Bestimmt die Position in der Reihenfolge der Modelltypen / schließt den Modelltyp von der Simulation aus.

## Modi-Beispiele:
## Modelltypen

```
MODELTYPE "Dokumentenmodell"
  from:none
  plural:"Dokumentenmodelle"
  pos:3
  not-simulateable
  bitmap:"db:\\doc_mod.bmp"
  attrrep:"BP Model Attributes"
INCL "Dokument"
INCL "Aggregation"
INCL "Notiz"
INCL "hat Subdokument"
INCL "hat Notiz"
```

**Modelltyp „Dokumentenmodell“ aus der ADOxx-Standard-Anwendungsbibliothek**




**Die Modellierungsleiste mit der Klassenreihenfolge**

**`MODELTYPE`, `pos` und `bitmap`**





## Einige Modi-Befehle:
## Ansichtsmodi

`MODE „modeName“ from: „modeSource“`

- Dieser Befehl definiert einen Ansichtsmodus mit dem Namen <*modeName*>. Gemeinsam mit diesem Befehl muss eine Liste von Modellierungsklassen definiert werden (wahlweise absolut oder relativ, wie oben beschrieben). MODE kann durch verschiedene Parameter erweitert werden.

`from: „modeSource“`

- Übernimmt die Klassen und Beziehungsklassen von der Quelle <*modeSource*> (**`all`**, **`none`** oder ein anderer Modus). **„`all`“** bezieht sich hierbei natürlich nur auf den Modelltyp, nicht das Metamodell.

`no-modeling`

- Der betroffene Modus ist für die Modellierung nicht verwendbar und wird daher auch nicht im Menüpunkt „Modi“ in der Modellierungskomponente angezeigt.

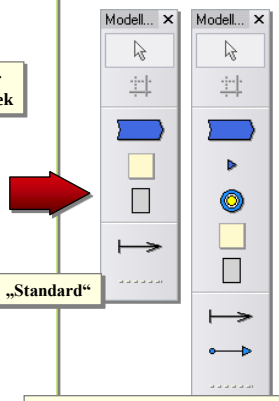
`no-documentation`

- Damit wird ein Modus bestimmt, welcher nicht für Dokumentationsgenerierung verwendet werden kann.

## Modi-Beispiele:
# Ansichtsmodi

```
MODELTYPE "Prozeßlandkarte" from:none plural:"Prozeßlandkarten"
  pos:0 not-simulateable bitmap:"db:\\compmap.bmp"
  attrrep:"BP Model Attributes"
INCL "Prozeß"
INCL "Aggregation"
INCL "Notiz"
INCL "hat Prozeß"
INCL "ist innerhalb"
INCL "hat Notiz"
INCL "Kennzahlenübersicht"
INCL "Kennzahl"
INCL "besitzt"
MODE "Standard" from:all
  EXCL "Kennzahlenübersicht"
  EXCL "Kennzahl"
  EXCL "besitzt"
MODE "Standard - mit Kennzahlen" from:"Standard"
  INCL "Kennzahlenübersicht"
  INCL "Kennzahl"
  INCL "besitzt"
MODE "Alle Modellierungsobjekte" from:"Standard"
  INCL "Kennzahlenübersicht"
  INCL "Kennzahl"
  INCL "besitzt"
MODE "Dokumentation" from:all no-modeling
  INCL "Prozeß"
  INCL "Kennzahlenübersicht"
  INCL "Kennzahl"
```

**Modelltyp „Prozesslandkarte“ aus der ADOxx-Standard-Anwendungsbibliothek**




**Modus „Standard“**

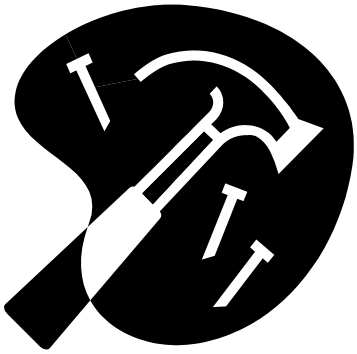
**Modi „Standard – mit Kennzahlen“ und „Alle Modellierungsobjekte“**







## Übung



## Aufgaben - Modelltypdefinition

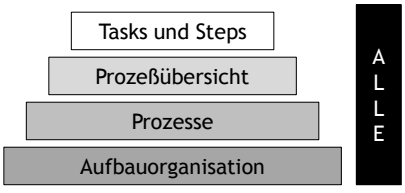
Erstellen Sie einen neuen Modelltyp

- Name: Task- und Stepmodell
- Klassen (in dieser Reihenfolge)
  - Task
  - Step
  - Notiz
  - Aggregation
  - besteht aus
  - verbindet
- Ansichtsmodus: Standard
  - alle Klassen
- Ansichtmodus: Tasksicht
  - Task
  - Aggregation
  - verbindet
- Ansichtsmodus: Stepsicht
  - Step
  - Aggregation
  - verbindet



## Aufgaben - Modelltypdefinition

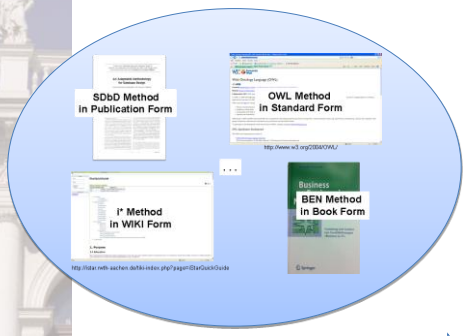
Erstellen Sie nachfolgende Modelltypgrafik




- Bei Klick auf „Tasks und Steps“ wird nur der Modelltyp „Task- und Stepmodell“ angezeigt und die Auswahl in der Grafik hervorgehoben (bspw. durch Fettdruck der Schrift).
- Bei Klick auf „Prozeßübersicht“ wird nur der Modelltyp „Prozeßlanskarte“ angezeigt und die Auswahl in der Grafik hervorgehoben (bspw. durch Fettdruck der Schrift).
- Bei Klick auf „Prozesse“ wird nur der Modelltyp „Geschäftsprozeßmodell“ angezeigt und die Auswahl in der Grafik hervorgehoben (bspw. durch Fettdruck der Schrift).
- Bei Klick auf „Alle“ werden alle Modelltypen angezeigt und die Auswahl in der Grafik hervorgehoben (bspw. durch Fettdruck der Schrift).

## Aufgaben - Meine Methode

**Startpunkt**




**Aufgabe:**

- Erstellen der Modelltypen
- Erstellen der Modelltypgrafik





## Modelling Method Implementation Instruments

| Modelling Method / Conceptualisation | Modelling Language | | | Modelling Procedure | Mechanisms and Algorithms | | |
|---|---|---|---|---|---|---|---|
| | Notation | Syntax | Semantic | | Basis Functions | Script Functions | External Access |
| Design | PART - 1 | | | | | | |
| Implementation | Model Type: Method-GraphRep, Icons | Model Type: ADOxx Library Language | | | Functional Description Table | | |
| | Object: GraphRep | Object: ADOxx Library Language | Object: Class Hierarchy | | Configuration Description | Scripting and Expression | ADO Web-Service (ADOscript API, Java API) |
| | Attribute: GraphRep | Attribute: ADOxx Library Language | | | | | |

## Material

1. ADOxx Methoden Handbuch
2. Open Models – Developer Wiki

   http://www.openmodels.at/web/omi/wiki