

# ETHZ-Kurs INTERLIS 2

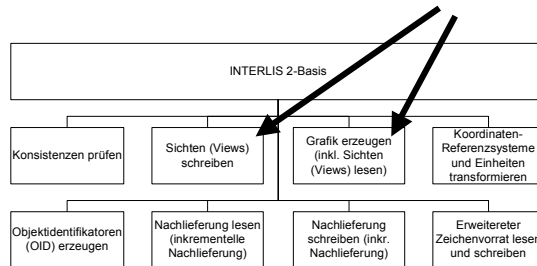
## 7. Teil: Datensichten und Grafikbeschreibungen

Prof. Stefan F. Keller  
Center int>e>gis am Institut ITA der  
HSR Hochschule für Technik Rapperswil, FHO  
sfkeller@hsr.ch, www.integis.ch

### Abschnitt Datensichten



## INTERLIS 2-Grafik und Sichten als spezielle Dienste (Spezifikationsteile)



## Datensichten

- ◆ **Datensichten (views) werden als Bestandteile von verschiedenen Diensten benötigt:**
  - **Grafik erzeugen (Grafikbeschreibung):**
    - **Welche Objekte** sollen dargestellt werden?
  - **Konsistenzbedingungen prüfen:**
    - **K.** ist eine Funktion auf Objekte, die wahr oder falsch zurückgibt: **welche Objekte?**
  - **Daten(-Ausschnitte) schreiben, bzw. abgeben**
    - evtl. nur ein Ausschnitt aus den Originaldaten, abgeleitete Daten: **Ausschnitt / abgeleitet von welchen Objekten?**

## Datensichten

- ◆ **Resultat einer Sicht ist eine Menge von nicht-  
originären Objekten: "Sicht-Objekte"**
- ◆ **Sicht-Objekte werden transferiert**
- ◆ **Aus der Perspektive des Datenempfänger werden  
Sicht-Objekte wie eigenständige Objekte  
aufgefasst (Sichten erscheinen wie Klassen)**
  - **OID-Problem: Nachlieferung von Sichten, die aus einer  
Kombination von Objekten besteht**

## Datensichten

- ◆ **Beispielanwendung:**
  - **Liegenschaftenbescrieb der AV zuhanden  
Grundbuch als Auszug aus Bodenbedeckung,  
Liegenschaften, Lokalisierung, etc.**
- ◆ **Viele Kombinationen möglich**
  - **SELECTION, JOIN, UNION, WHERE, etc.**
  - **Beispiel**

```
VIEW OurView =  
  JOIN OF B,C WHERE B:Attr1 == C:Attr1;  
ATTRIBUTE  
  ALL OF B; Attr3 = TEXT*40 := C.Attr1;  
END OurView;
```

## Datensichten

### Weitere Informationen

- ◆ vgl. INTERLIS-Referenzhandbuch, Kapitel 2.15
- ◆ sowie [www.integis.ch](http://www.integis.ch), sowie [www.interlis.ch](http://www.interlis.ch)

## Abschnitt Grafikbeschreibungen



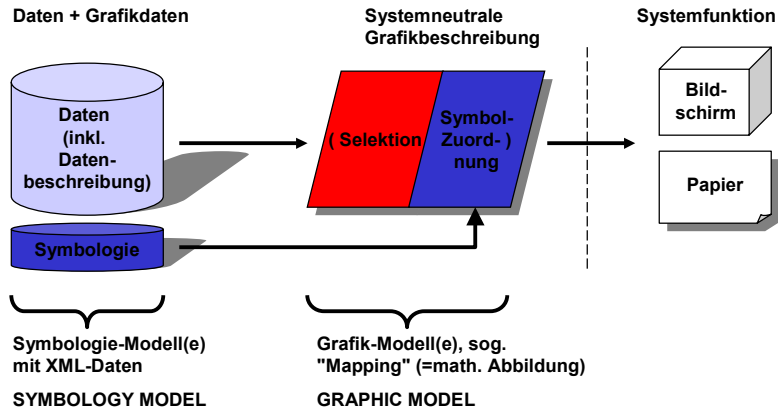
## Was Grafikbeschreibung meint...

- ◆ **Synonym: Grafikdefinition**
- ◆ **Darstellung des Inhalts und**
  - *nicht* automatische Darstellung (Textposition),
  - *nicht* Gestaltung des Kartenrandes oder Legenden

## Grafikbeschreibungen und modell-basierten Methode

- ◆ **Passt zum Prinzip**
  - Daten einmal erfassen, mehrfach nutzen und mehrfach darstellen
  - Ein Datenmodell ⇔ mehrere (karto-)grafische Modelle
- ◆ **Lösungsansatz**
  - system-neutral, beschreibend, deklarativ
- ◆ **Nutzen**
  - wie bekannt: nachvollziehbar dokumentiert / vergleichbar (Transparenz), für jede Aufgabe das geeignetste (verfügbare) System, offene Märkte...

## Datenfluss: Von "darstellungslosen" Geodaten zu Grafikdaten




## Grafikbeschreibung: Ziel

Ziel ist, dass wir z.B. für einen Grenzpunkt und eine Beschriftung schreiben können:

```

GRAPHIC SimplePunktGrafik
  BASED ON Daten.Punkte.Punkt =
    Symbol OF SimpleSigns.Signs.Symbol: (
      Name := "Punkt";
      Pos := Lage);
END SimplePunktGrafik;


```



```

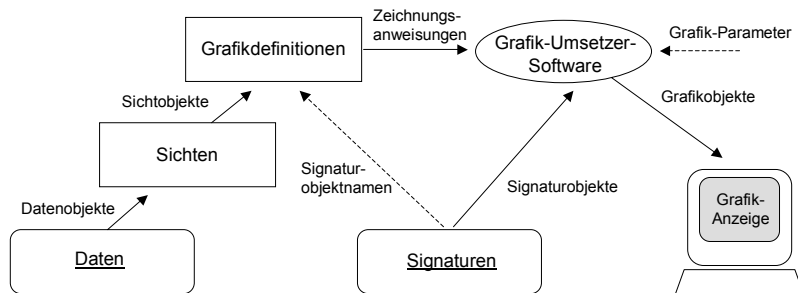
GRAPHIC PunktNummer BASED ON AV93CH_BB.BB.Einzelpunkt =
  Text OF ExtendedSymbology.ExtendedSigns.TextSign: (
    Name := "Vektorschrift";
    Pos := Geometrie;
    Text := Identifikator;
    !! Priority: ...);
END PunktNummer;

```



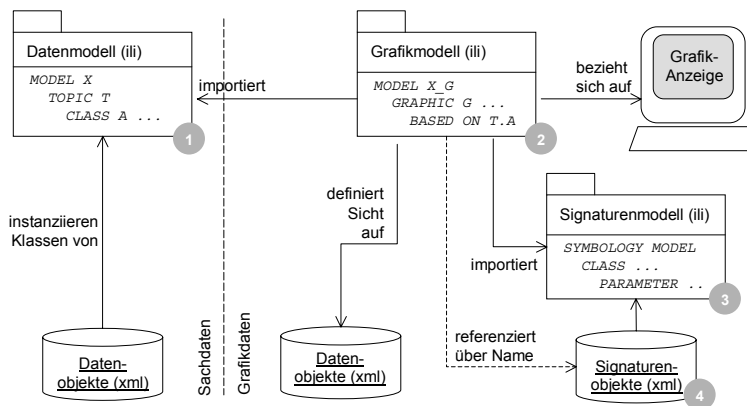
101

## Figur (aus ILI 2.1-Ref.-Hb.)



Figur 5: Grafikdefinitionen, die einerseits auf Daten und Sichten und andererseits auf Signaturen aufbauen, um daraus eine Grafik zu erzeugen (abstrahierte Darstellung).

## Modell- und Objekt-Abhängigkeiten bei der Grafikbeschreibung und -erzeugung



Legende:   
 ———> : starke Modellabhängigkeit (z.B. importieren oder instanzieren)   
 - - - - -> : sonstige Abhängigkeit (z.B. Referenzieren via Name)

(aus: figures\_misc.ppt)

# GRAPHIC MODEL und SYMBOLOGY MODEL

```

DATA MODEL Roads_Example =
:
CLASS PointObject =
  Type: MANDATORY (tree,...);
  Position: MANDATORY Point2D;
END PointObject;
    
```

```

GRAPHIC MODEL Roads_Graphics =
:
BASED ON <Data-Topic>.PointObject =
Tree OF <Symbology-Topic>.SimpleSign:
WHERE Type == #tree (
  Name := "Tree";
  Geometry:= Position;
  Priority := 130
);
    
```

```

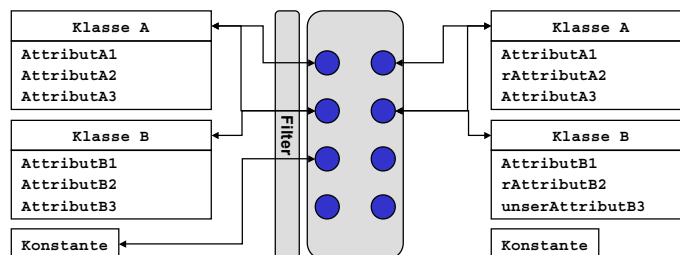
(INTERLIS vordefiniert)
:
CLASS SIGN (ABSTRACT) =
  Name: NAME;
PARAMETER
  Name: NAME;
END SIGN;

SYMBOLY MODEL BasicSymbology
:
CLASS SymbolSign (ABSTRACT) EXTENDS SIGN =
PARAMETER
  Geometry: MANDATORY Style_COORD2;
  Rotation: Style_Angle;
END SymbolSign

SYMBOLY MODEL ExtendedSymbology =
:
CLASS SymbolSign (EXTENDED)
  Symbol: MANDATORY -> FontSymbol;
  Scale: ES Float;
  Color: -> Color;
  Rotation: ES_Angle;
  ClipSymbol: -> FontSymbol;
PARAMETER
  Priority: MANDATORY ES_Priority;
END SymbolSign
    
```

## Grafikdefinition als "Schema Mapping"

- ◆ **Schema Mapping/Integration: Zuordnung von Objekten aus Schema A nach Schema B**
  - z.B. von Objekt-orientiertem (konzeptionellen) Schema nach relationalem (internem) Schema
- ◆ **Typische Zuordnungen:**





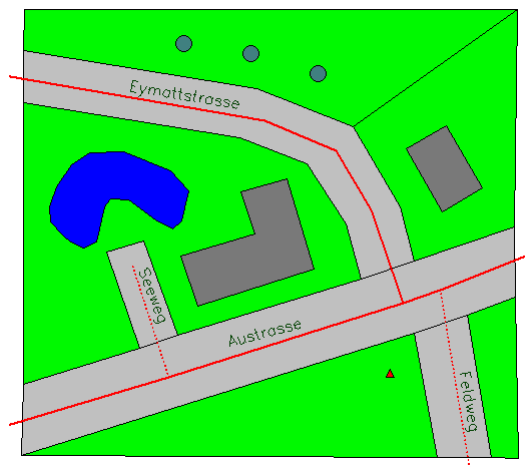
## Grafikdefinitionen: Erste Näherung

- ◆ Vorgegeben ist **CLASS SIGN** (vgl. S. 49/50)

```
CLASS SIGN (ABSTRACT) =  
  Name: NAME;  
  PARAMETER  
    Name: NAME;
```

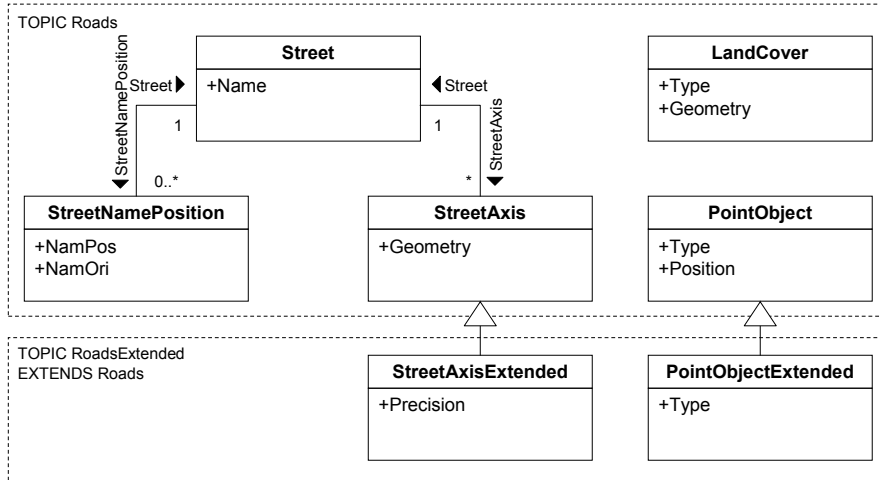
- ◆ auf **SIGN** aufbauend
  - das **SYMBOLLOGY MODEL**
- ◆ das (Daten-) **MODEL**
  - mit den darzustellenden Objekten
- ◆ dann (endlich)
  - die Grafikdefinition in einem (Grafik-) **MODEL**

## Ein kleines Beispiel Roads



[Abbildung 8 aus dem  
INTERLIS 2-Ref.-Hb.]

# Datenmodell des Beispiels Roads



[aus INTERLIS 2-Ref.-Hb.]

# Grafikdefinition mit dem Roads-Beispiel

```

MODEL RoadsExgm2ien_10 (en)
IMPORTS RoadsExdm2ien_10;
IMPORTS ExtendedSymbology;
...
GRAPHIC Point_Graphics
  BASED ON RoadsExdm2ien_10.Roads.PointObject =
  Tree OF ExtendedSymbology.ExtendedSigns.SymbolSign:
  (
    Sign := Tree;
    Geometry := Position;
  );
END Point_Graphics;
...
END RoadsExgm2ien_10.

```

- Legende:**
- Grafikmodell-Elemente
  - Datenmodell-Elemente
  - Symbologiemodell-Elemente
  - Symbolbibliothek-Objektnamen

## Grafikdefinition mit dem Roads-Beispiel (II)

```

MODEL RoadsExgm2ien_10 (en) =
  CONTRACT ISSUED BY Unknown;
  IMPORTS RoadsExdm2ien_10;
  IMPORTS ExtendedSymbology;
  SIGN BASKET ExtendedSymbology
    ~ ExtendedSymbology.ExtendedSigns;

  TOPIC Graphics DEPENDS ON RoadsExdm2ien_10.Roads =
    GRAPHIC Text_Graphics
      BASED ON RoadsExdm2ien_10.Roads.StreetNamePosition =
        StreetName OF ExtendedSymbology.ExtendedSigns.TextSign: (
          Sign := Linefont 18; !! oder Arial
          Txt := Street.Name;
          Geometry := NamPos; ); !! Rotation:=NamOri; Priority:=120;
        END Text_Graphics;

    GRAPHIC Point_Graphics
      BASED ON RoadsExdm2ien_10.Roads.PointObject =
        Tree OF ExtendedSymbology.ExtendedSigns.SymbolSign: (
          Sign := Tree;
          Geometry := Position; ); !! Priority:=130;
        END Point_Graphics;

  END Graphics;
END RoadsExgm2ien_10.

```

Legende:  
 - Grafikmodell-Elemente  
 - Datenmodell-Elemente  
 - Symbologiemodell-Elemente  
 - Symbolbibliothek-Objektnamen

## Grafikdefinition mit dem Roads-Beispiel (III)

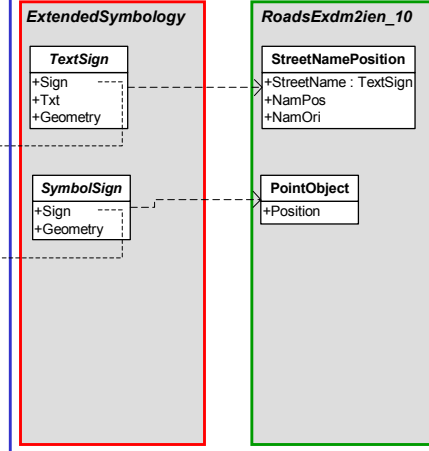
RoadsExgm2ien\_symbols.xml

```

ExtendedSymbology
<SymbolSign TID="2002"> <!--
  <Name>Text</Name><!--
  <Symbol>102</Symbol>
  <Scale>1.0</Scale>
  <Color>2</Color>
</SymbolSign>
<FontSymbol TID="102"> <!--
  <Name>Circle</Name>
  <Geometry>
    <FontSymbol_Surface>
      <Geometry>
        <SURFACE>
          <BOUNDARY>
            <POLYLINE>
              <P C1="-0.5" C2="0.0"/>
              <A C1="0.0" C2="0.5" R="0.5"/>
              <P C1="0.5" C2="0.0"/>
              <A C1="0.0" C2="-0.5" R="0.5"/>
              <P C1="-0.5" C2="0.0"/>
            </POLYLINE>
          </BOUNDARY>
        </SURFACE>
      </Geometry>
    </FontSymbol_Surface>
  </Geometry>
</FontSymbol>

```

RoadsExgm2ien.iii



## Zusammenfassung Grafikdarstellungen

### ◆ Grafikdarstellungen sind...

- Mathematisch gesehen:
  - Grafik := Funktion ( Daten, Parameter)
- Datenbank-technisch gesehen:
  - Externe Modelle zu konzeptionellem Modell in 3-  
Schema-Architektur (extern-konzeptionell-intern)

### ◆ INTERLIS-Textbeispiele:

- INTERLIS 2-Referenzhandbuch:
  - SimpleGrafik (siehe Kap. 2.14)
  - Roads (siehe roads-Verzeichnis und Anhang C)
- Übung

## Grafikbeschreibungen: Demo

### ◆ Demo \xml-xsl

- tds02d2\_inkr\_xsl.xml, ruft tds02d.xsl auf
- Direkt im Internet Explorer >5.5 interpretiert
- erzeugt (intern) VML, das direkt dargestellt wird

### ◆ Weitere INTERLIS-SW, die Grafik erzeugt:

- INTERLISViewer (INTERLIS 1)
- InterlisStudiio (INTERLIS 1)
- GeoShop
- jedes GIS, das INTERLIS integriert...

## Weitere Projekte und Informationen

### ◆ Projekte

- IXS-Tool: siehe [www.integis.ch/projekte.html](http://www.integis.ch/projekte.html)

### ◆ Links speziell zur INTERLIS-Grafik

- Das INTERLIS-Portal (de/fr/en): [www.interlis.ch](http://www.interlis.ch)
- Center interlis an der HSR: [www.integis.ch](http://www.integis.ch)