Test Data DSL
Erste Ergebnisse 08.04.2013

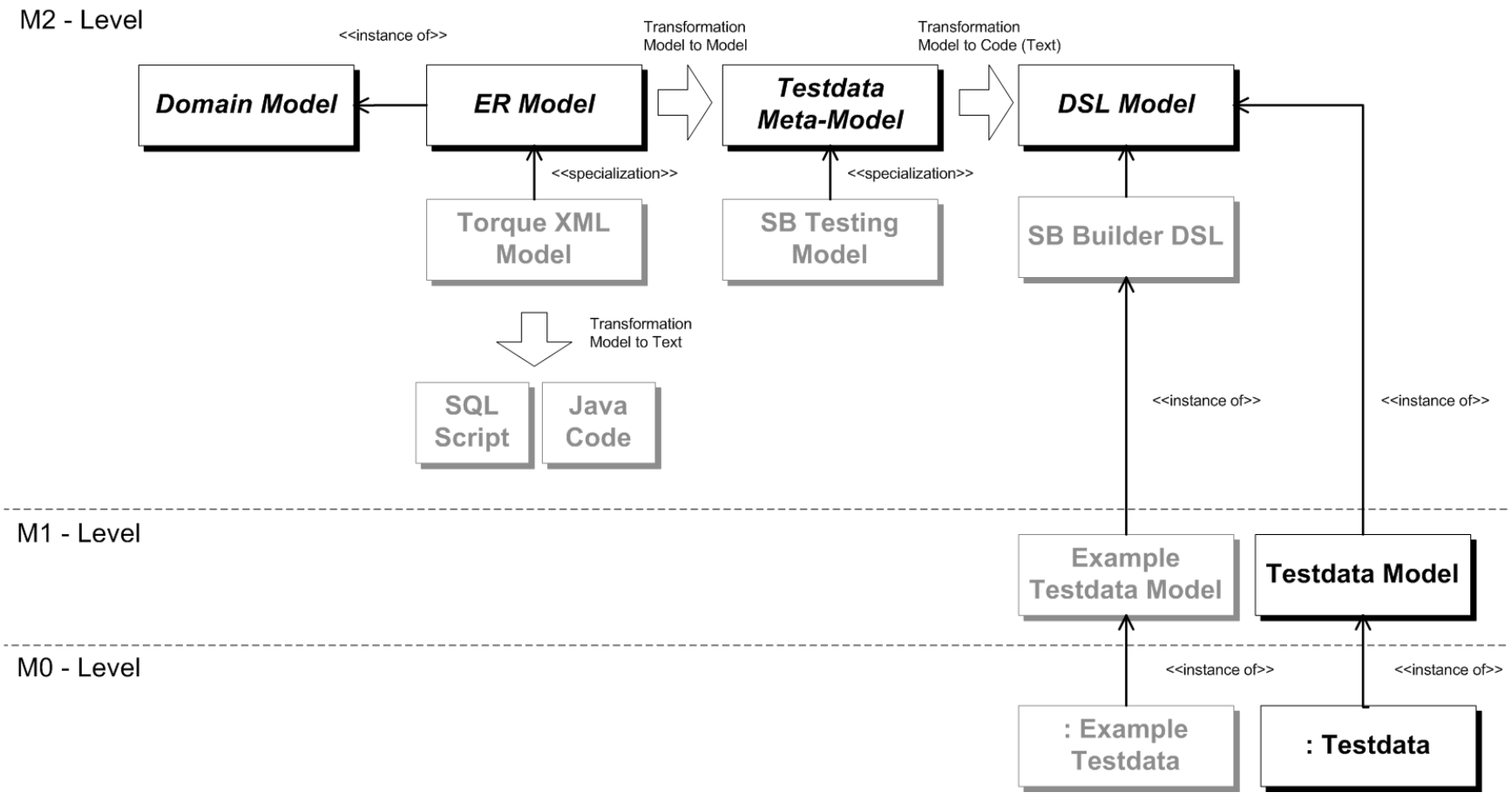
Nikolaus Moll

Christian Baranowski

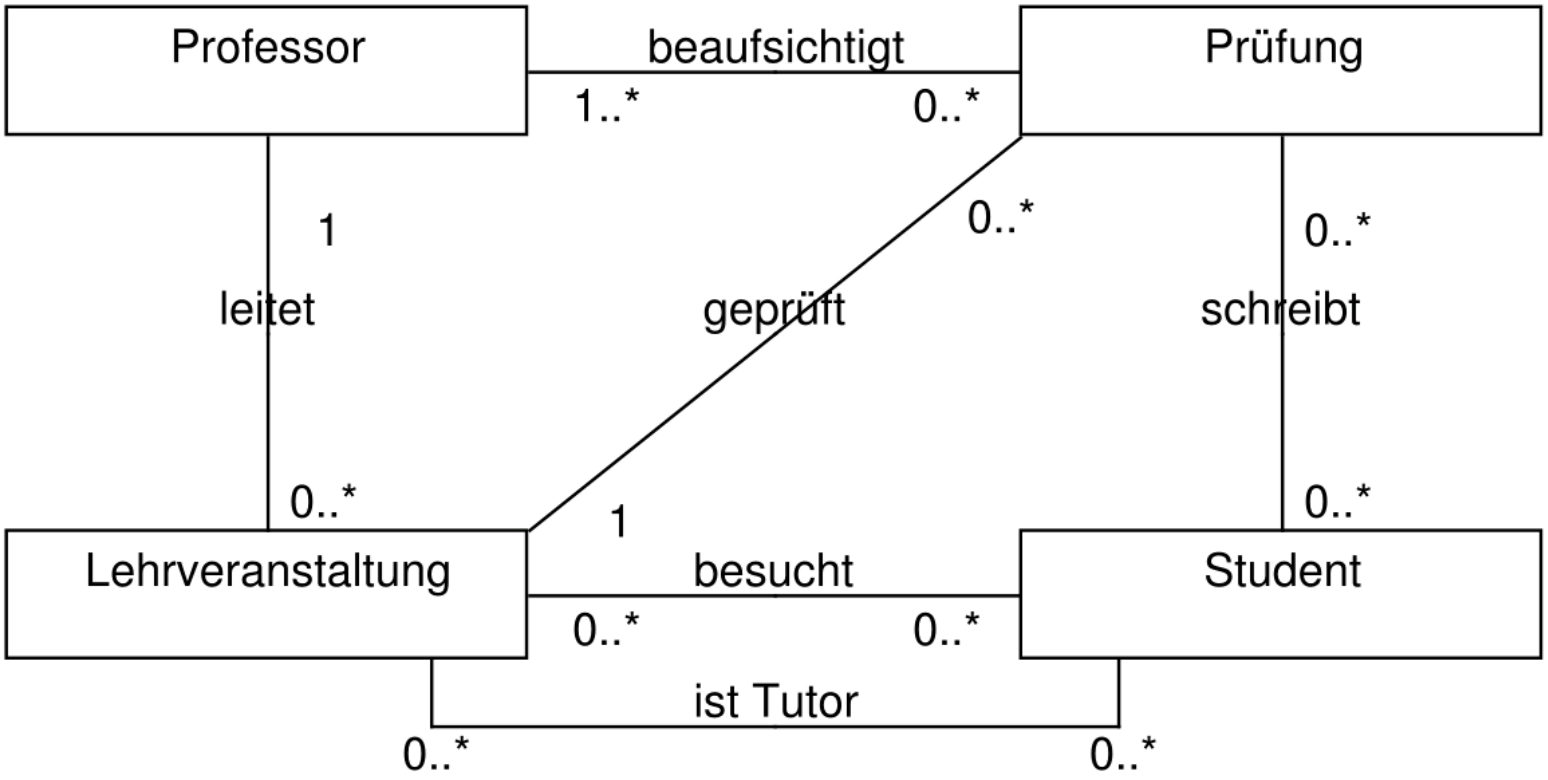
Agenda

- Modellierungsebenen
- Beispiel und Fragestellung vorstellen
- Problemstellung an Beispielen
  - DBUnit XML Data-Set
  - DBUnit Java Data-Set
  - SBTesting DSL (Builder Pattern) für DBUnit
- DSL Vorschläge und Diskussion
- Stand der Arbeit

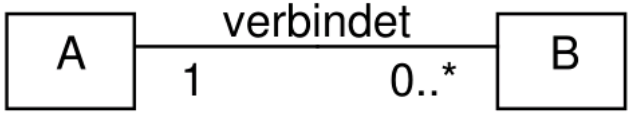
Modellierungsebenen



Beispiel Anwendung (M2)

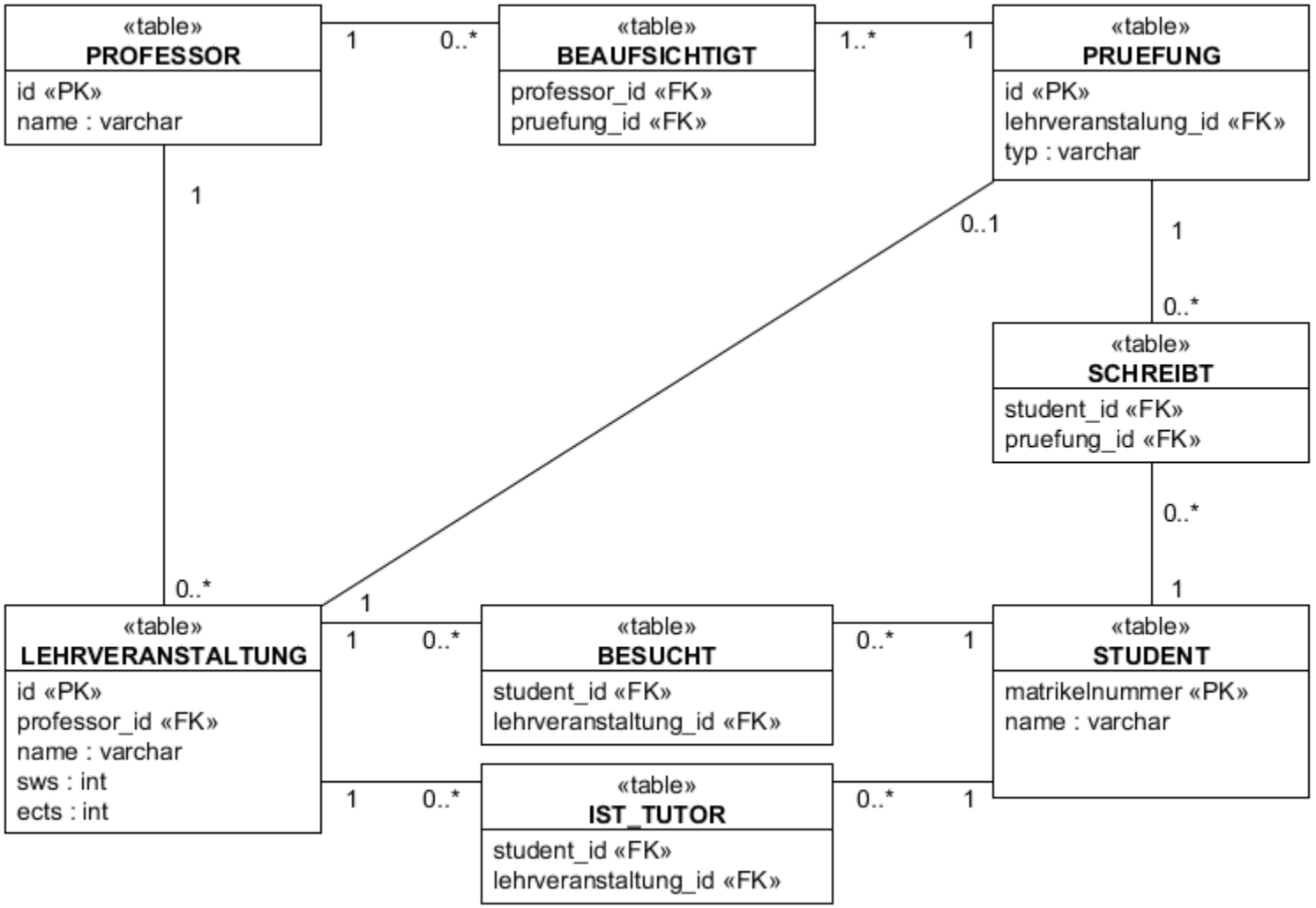


Legende



Ein A ist mit beliebig vielen B verbunden
Ein B ist genau mit einem A verbunden

Beispiel Anwendung (M2)



Beispiel-Fragestellung

1. Welcher Professor unterrichtet die meisten Studenten?
2. Welcher Student nimmt an den meisten Prüfungen teil?
3. Welcher Student ist Tutor und nimmt gleichzeitig an der Prüfung teil?
4. Welcher Professor macht die wenigste Aufsicht in Fremdveranstaltungen (Lehrveranstaltungen eines anderen Professors)?

DBUnit XML Data-Set (0..*) (M2 & M1)

```
<table name="professor">
  <column>ID</column>
  <column>name</column>
  <row>
    <value>1</value>
    <value>Jürgen Wäsch</value>
  </row>
  <row>
    <value>2</value>
    <value>Oliver Haase</value>
  </row>
</table>
```

```
<table name="lehrveranstaltung">
  <column>ID</column>
  <column>professorID</column>
  <column>name</column>
  <row>
    <value>1</value>
    <value>2</value>
    <value>Verteilte Systeme</value>
  </row>
  <row>
    <value>2</value>
    <value>2</value>
    <value>Design Patterns</value>
  </row>
</table>
```

DBUnit Java Data-Set (0..*) (M2 & M1)

```
DefaultTable professor = new DefaultTable(
  "professor",
    new Column[] {
      new Column("id", DataType.INTEGER),
      new Column("name", DataType.VARCHAR),
    }
  );

professor.addRow(new Object[] {
    Parameters.Professor.WAESCH_ID,
    "Jürgen Wäsch"
  });
professor.addRow(new Object[] {
    Parameters.Professor.HAASE_ID,
    "Oliver Haase"
  });
dataSet.addTable(professor);
```

```
DefaultTable lehrver = new DefaultTable(
  "lehrveranstaltung",
    new Column[] {
      new Column("id", DataType.INTEGER),
      new Column("professorid", DataType.INTEGER),
      new Column("name", DataType.VARCHAR),
    }
  );

lehrver.addRow(new Object[] {
    Parameters.Lehrveranstaltung.VERTEILTE_SYSTEME_ID,
    Parameters.Professor.HAASE_ID,
    "Verteilte Systeme"
  });
lehrver.addRow(new Object[] {
    Parameters.Lehrveranstaltung.DESIGN_PATTERNS_ID,
    Parameters.Professor.HAASE_ID,
    "Concurrency and Design Patterns"
  });
dataSet.addTable(lehrver);
```

SB Testing DSL mit DBUnit (0..*) (M1)

```
table_Professor
    .insertRow()
          .setId(Parameters.Professor.HAASE_ID)
          .setName("Oliver Haase")
    .insertRow()
          .setId(Parameters.Professor.WAESCH_ID)
          .setName("Jürgen Wäsch");

table_Lehrveranstaltung
    .insertRow()
          .setId(Parameters.Lehrveranstaltung.VERTEILTE_SYSTEME_ID)
          .setProfessorId(Parameters.Professor.HAASE_ID)
          .setName("Verteilte Systeme")
    .insertRow()
          .setId(Parameters.Lehrveranstaltung.DESIGN_PATTERNS_ID)
          .setProfessorId(Parameters.Professor.HAASE_ID)
          .setName("Design Patterns");
```

SB Testing DSL mit DBUnit (0..*) (M1)

```
RowBuilder_Professor haase =
  table_Professor
    .insertRow()
      .setName("Oliver Haase");

RowBuilder_Professor waesch =
  table_Professor
    .insertRow()
      .setName("Jürgen Wäsch");

RowBuilder_Lehrveranstaltung vsys =
  table_Lehrveranstaltung
    .insertRow()
      .setName("Verteilte Systeme")
      .refProfessorId(haase);

RowBuilder_Lehrveranstaltung design_patterns =
  table_Lehrveranstaltung
    .insertRow()
      .setName("Design Patterns")
      .refProfessorId(haase);
```

Mögliche DSL

```
WAESCH = professor {
    name           "Jürgen Wäsch"
    beaufsichtigt P_VSYS
}

HAASE = professor {
    name           "Oliver Haase"
    beaufsichtigt P_DESIGN_PATTERNS
    leitet         VSYS, DESIGN_PATTERNS
}

VSYS = lehrveranstaltung {
    name           "Verteilte Systeme"
}

DESIGN_PATTERNS = lehrveranstaltung {
    name           "Concurrency and Design Patterns"
}
```