

Vielfalt vs. Abstraktion

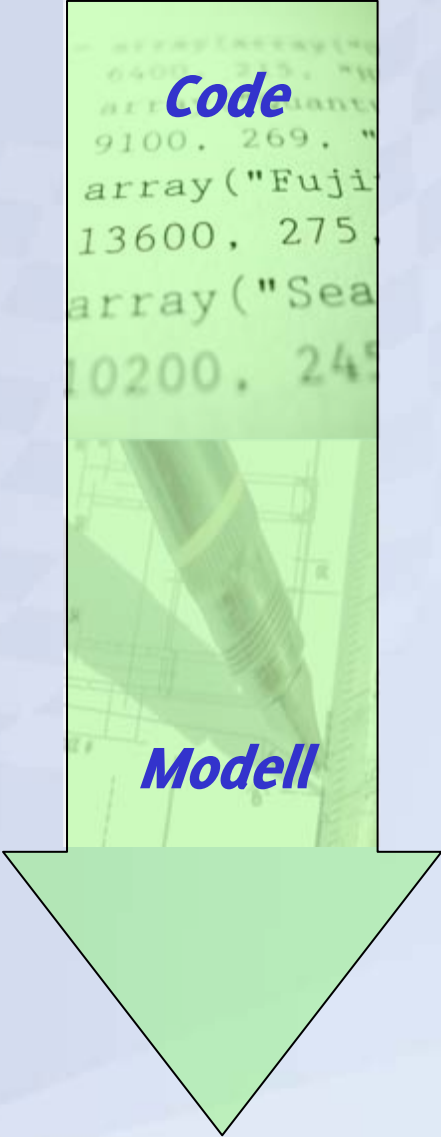
Der Jakobsweg der modellbasierten GUI-Entwicklung

Benjamin Schmid

ex|xcellent
solutions



Ansätze zur GUI Entwicklung



Code

Modell

- **Programmatisch**
- **GUI Designer**
- **Deklarative Sprachen**
 - XUL, XAML, XForms, JAXfront / KParts / ...
- **Annotated Business Model**
 - Naked Objects, Trails, JMatter, ...
- **Modellierungssprachen**
 - UML, UMLi, DiaMODL, ...
 - **MDA**

GUI-Entwicklung: **Programmatisch**

- ❌ Vorgehen & Struktur:
Technologiegebunden
- ❌ geringe Abstraktion
- ❌ keine Verwendung
von Modellwissen
- ✅ Hohe Flexibilität &
Detailsteuerung
- ✅ idR gute Basis für
aufbauende Konzepte

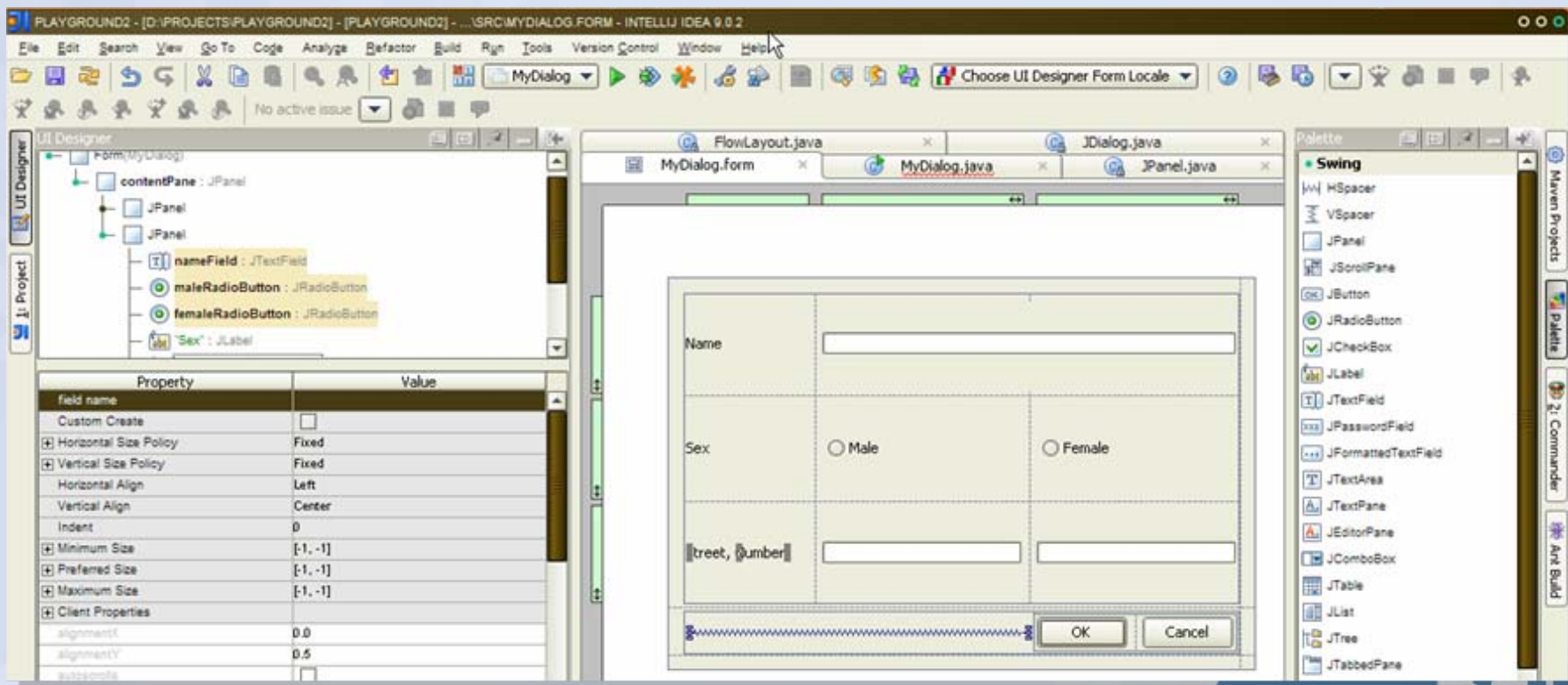
```
class HelloWorldSwing {  
    public static void main(String[] args) {  
        Runnable guiCreator = new Runnable() {  
            public void run() {  
                JFrame f = new JFrame  
                    ("Hallo Welt mit Swing");  
                f.setDefaultCloseOperation(JFra...  
  
                // Fügt den "Hallo Welt"-Text hinzu  
                JLabel label = new JLabel("Hallo Welt");  
                f.getContentPane().add(label);  
  
                // Zeigt das Fenster an  
                f.setSize(300, 200);  
                f.setVisible(true);  
            }  
        };  
        SwingUtilities.invokeLater(guiCreator);  
    }  
}
```

GUI-Entwicklung: **GUI-Designer**

oft: **Visueller Editor** für **Swing** mit **Code Generator**

✓ Werkzeug, niedrige Einstiegshürde, WYSIWYG

✗ behindert weitere Abstraktionen & komplexere UIs



GUI Entwicklung: **Deklarative Sprachen**

Vertreter: *XUL, XAML, XForms, JAXfront, KParts, ZUL, ...*

✗ Sehr technisch / konkret / gegenständlich

✗ Betrachtet in der Regel nicht:

- Modellwissen
- GUI-Verhalten & -Logik
- dynamische GUI-Strukturen

Beispiel: XUL (Mozilla)

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<window id="sample-window"
  title="Beispiel"
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <vbox>
    <checkbox label="CheckBox"/>
    <hbox>
      <spacer flex="1"/>
      <button label="OK"/>
      <button label="Abbrechen"/>
    </hbox>
  </vbox>
</window>
```



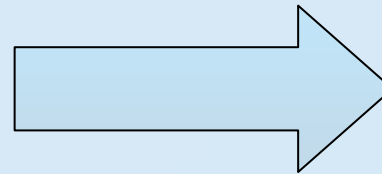
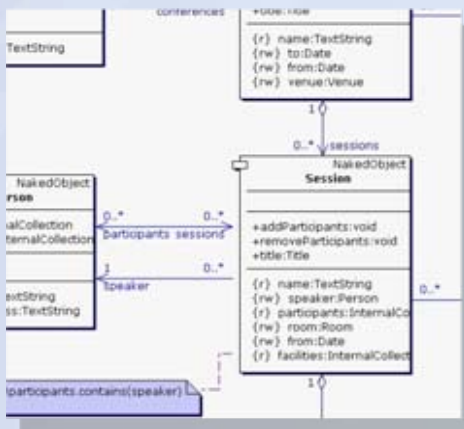
GUI-Entw.: UI aus Business Model

❌ Basisannahme: **B0-Modell : UI Funktion ~ 1:1**

- gute Basis für einfachste CRUD-Anwendungen
- Praxisbedürfnisse gehen schnell darüber hinaus (Dialogführung/-struktur/-verhalten/-gestaltung, etc.)

✓ Abstraktion und Verwendung modellierten Wissens

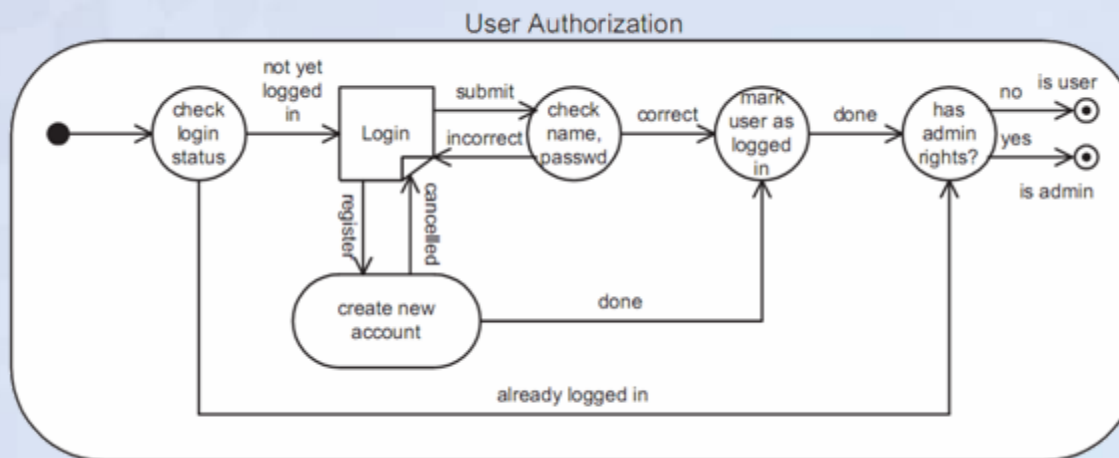
Reinform-Beispiel: „NAKED OBJECTS“-Architektur



GUI Entwicklung: Weitere UI Modelle

Dialogsteuerungs-Modelle

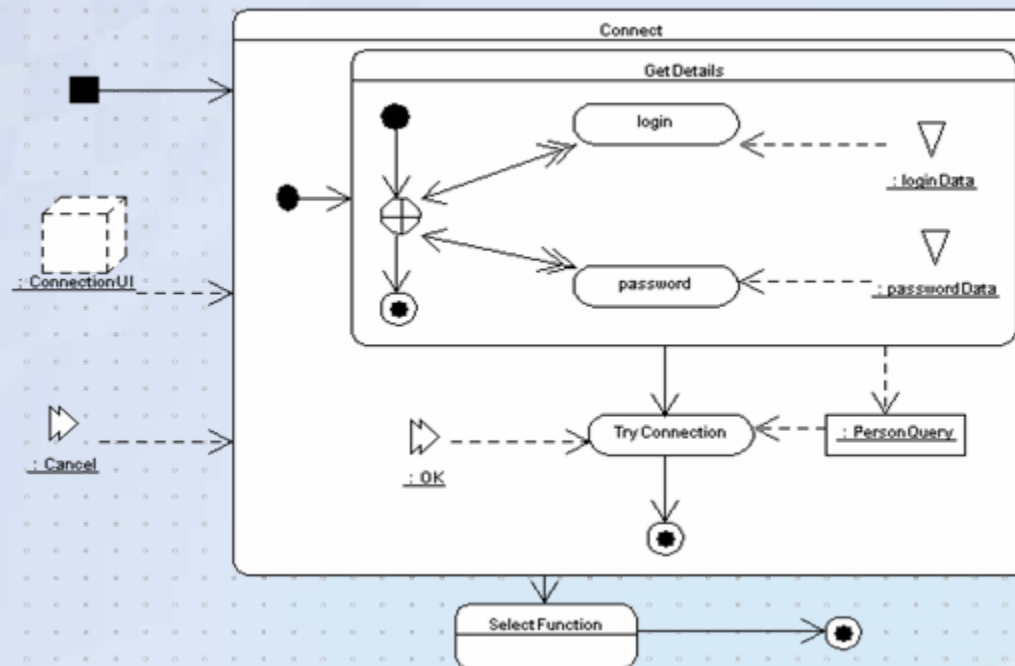
z.B. *Dialog Flow Notation (DFN)*



- Webseiten-/Dialogorientiert
- Dialogsteuerung auf Makroebene
- Richtung: Prozessflußsteuerung

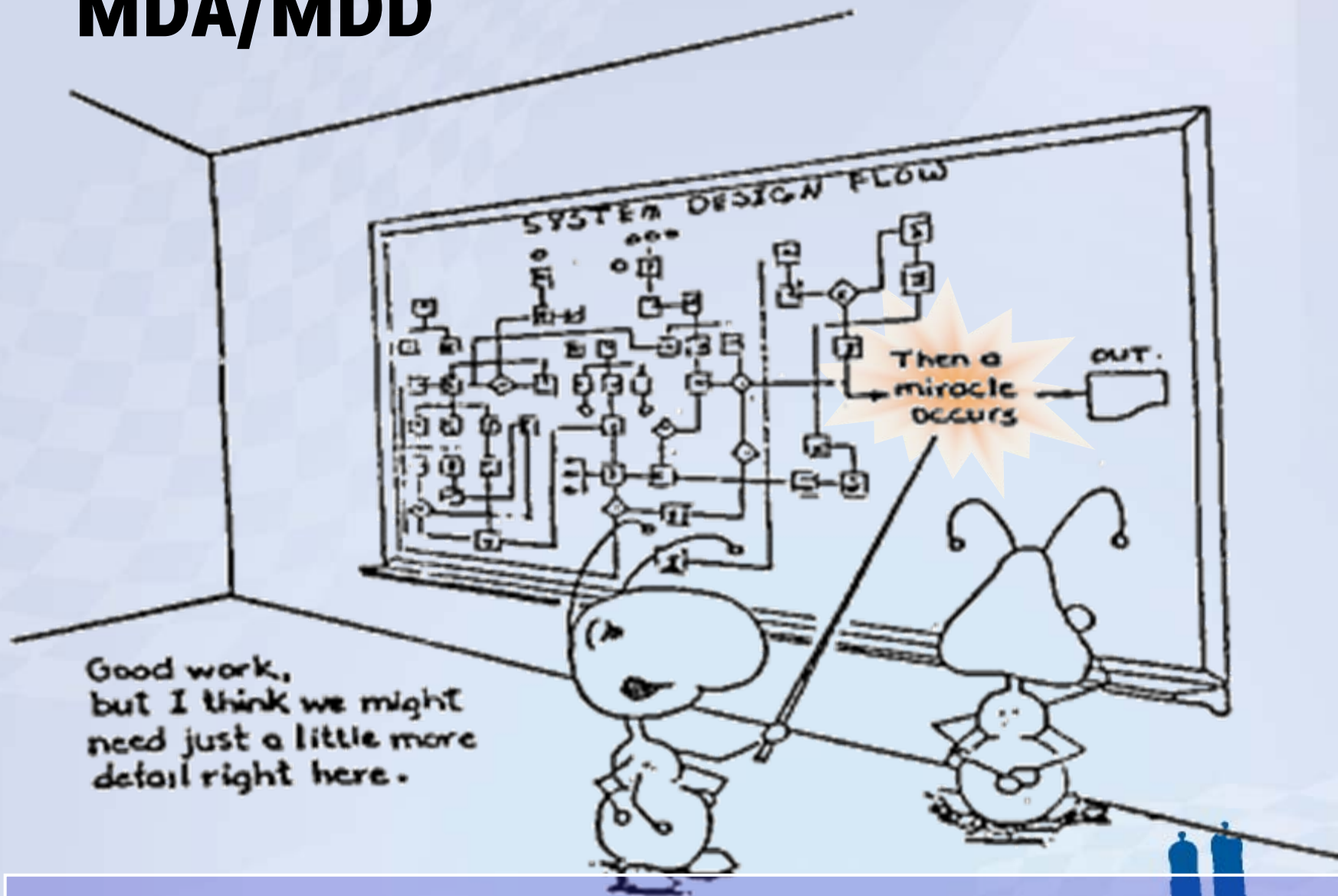
GUI Entwicklung: **MDA/MDD**

- GUIs konzeptionell **integraler Bestandteil** von **MDA**
- ... in UML direkt gar nicht betrachtet. Dafür **zahlreiche erfolglose Modell-Versuche** (UMLi, DiaMODL, ...)



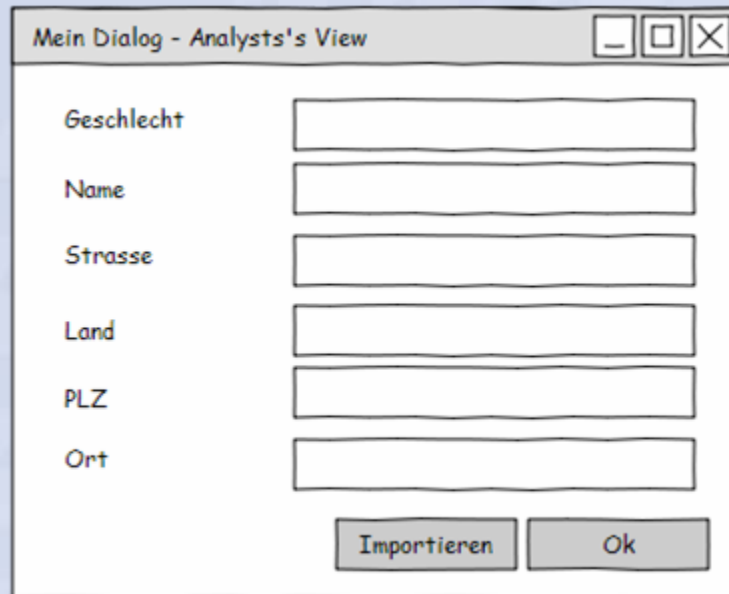
☹️ **Es fehlen:** Werkzeuge, praxistaugliche Lösungen, ...

MDA/MDD



MDA: PIM -> PSM -> Fertige Anwendung - mit GUI?

Dilemma: **Abstraktion** ...



Mein Dialog - Analysts's View

Geschlecht

Name

Strasse

Land

PLZ

Ort

Importieren Ok

- **Inhalte**

- Modellierte **Attribute**
(Labels, Eingabefelder)
- Modellierte **Aktionen**
(Buttons)

- **Funktionen**

- Business Funktionen
(„Löschen“)
- Navigation

Dilemma: ... vs. Details

- **Inhalt**
- **Funktion**



- Anordnung
- Größen
- Ausrichtung
- Widgettypen
- Farben
- Interaktionsmuster
- Tooltips & Hints
- visuelle Gruppierungen
- ...

Mein Dialog - User's View

Stammdaten Details

☐ Herr ☐ Frau

Vorname*: Max

Nachname*:

Strasse: Herbert-Mayer-Str. 12a

PLZ/Ort: DE - 12345 Musterstadt

Staatsangeh.: Bajuware

Bitte geben Sie einen Nachnamen ein!

Importieren ... Abbrechen Ok

5px

350px

500px

5px

2px

Fehlerhintergrund: RGB(255,200,200)

rechtsbündig

Fehlermeldungsbereich

Länderliste aus Datenbankabfrage

5-Zeichen, numerisch

Tooltip / Pflichtfeld

GUI-Modellierung: **Ja oder nein?**

Modellorientierte UI-Entwicklung birgt viel Potential:

- Wiederverwendung **bestehenden Modellwissens**
 - Domain Model (Entitäten, Attribute)
 - Validierungen (Kardinalitäten, OCL)
 - Zusatz-Infos / DSLs (Namen, Rechteregeeln, ...)
- Für eng abgegrenzte Domänen (Rapid Prototyping, Dialogfluss,...) erscheinen auch zusätzliche, reine **UI Modelle** attraktiv

GUI-Modellierung: **Ja oder nein?**

Unmöglich **alle GUI-Aspekte** sinnvoll zu modellieren

- Erst **starke Reduktion** ermöglicht **sinnvolle Modellierung**
- Reine Modell-GUIs nur unter *sehr engen* Annahmen („Naked-Objects“)
- Vollwertige UIs erfordern immer **zusätzliche Detailsteuerung**
 - Implementierungszugriff/Code notwendig!
 - **Feinsteuerung** auf **allen Ebenen:** vom Querschnitt bis ins Detail

Anforderungen datenzentrischer UIs (1)



GUI-VISUALISIERUNG

Visualisierung

mit div. Basistechnologien

Dynamische Layouts

für „Formulare“

Regelabhängige Präsentation

aufgrund Rechte, Zustände, ...

Benutzerführung und Feedback

Anforderungen datenzentrischer UIs (2)

GUI-INHALTE

Datenbeschaffung

modellierte und modellfremder Inhalte

Eingabe-Validierungen

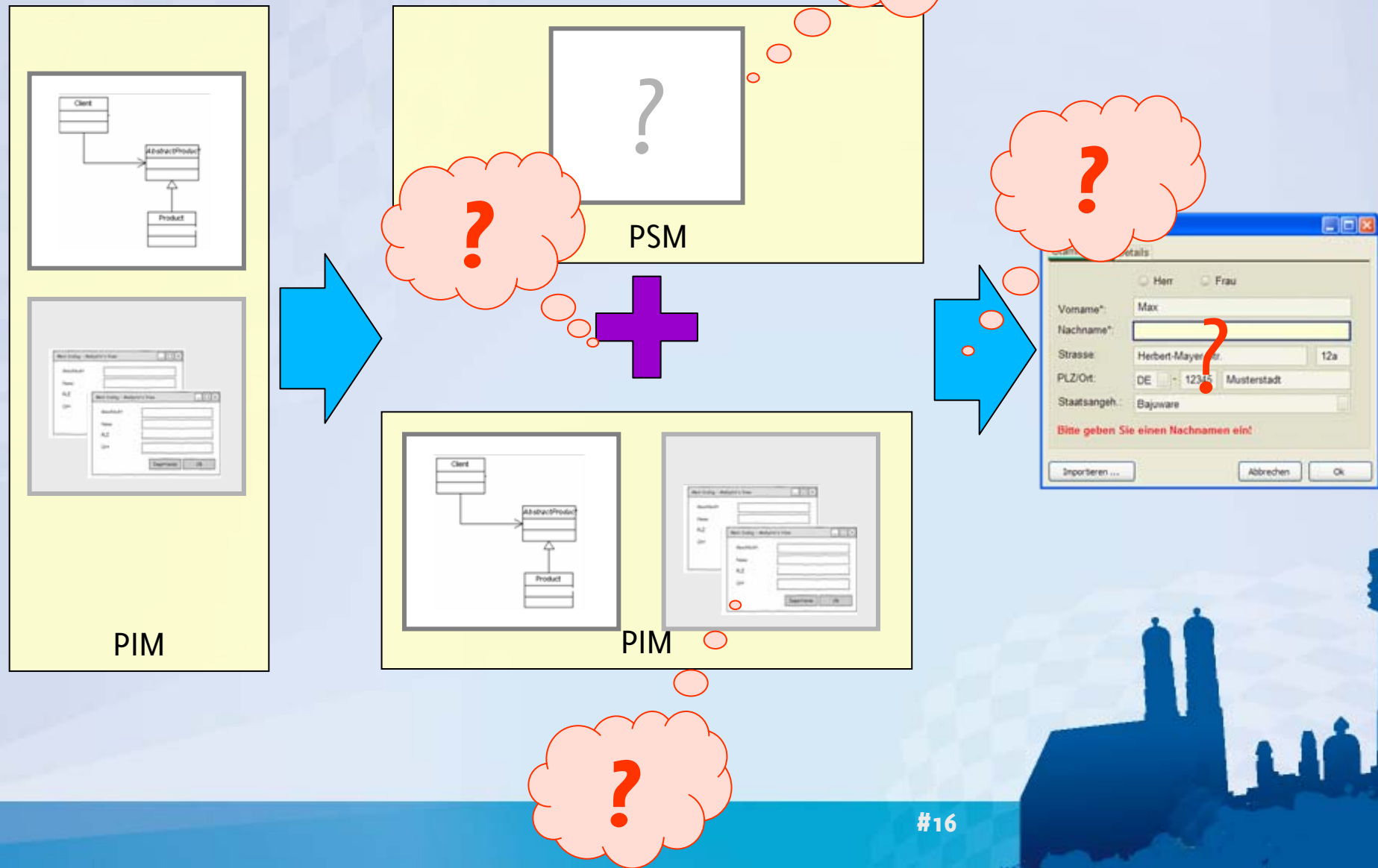
Attribut/Objekt-Prüfungen z.B. **via OCL (Modell)**

I18N

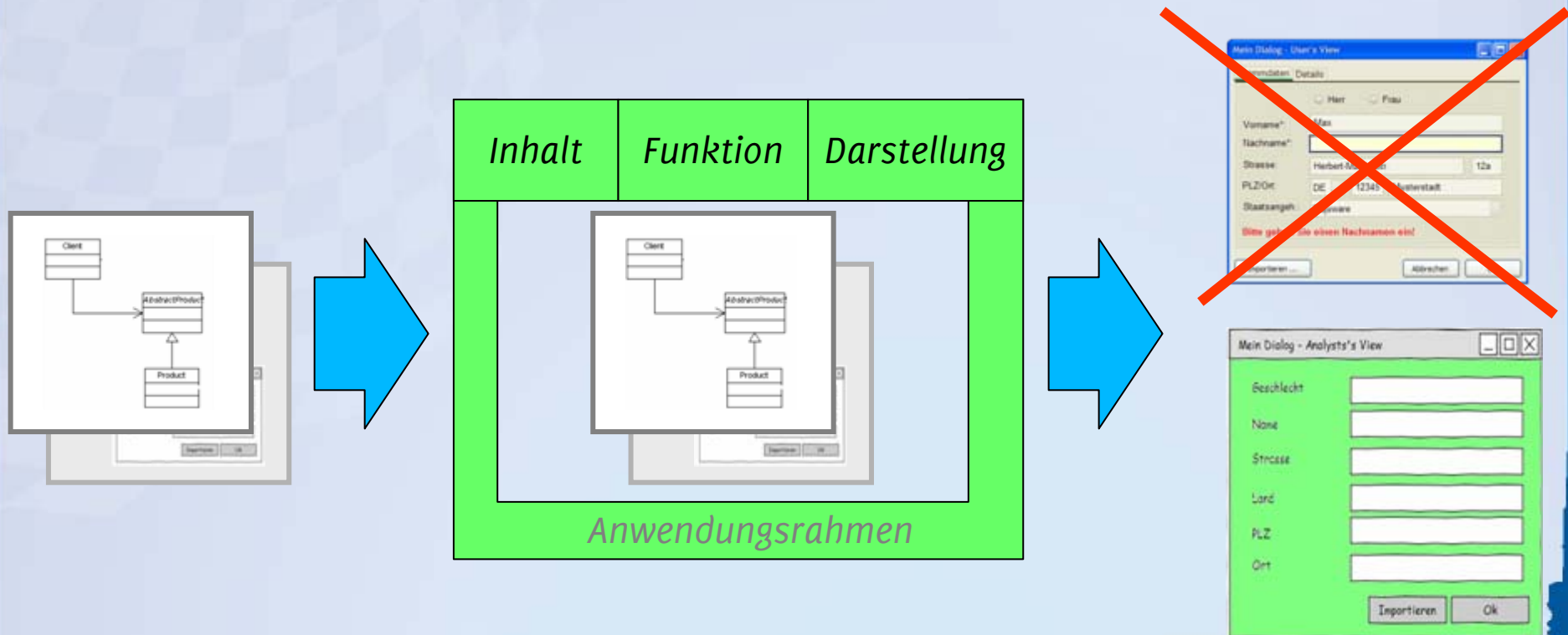
UI Ablaufsteuerung & Datenfluss



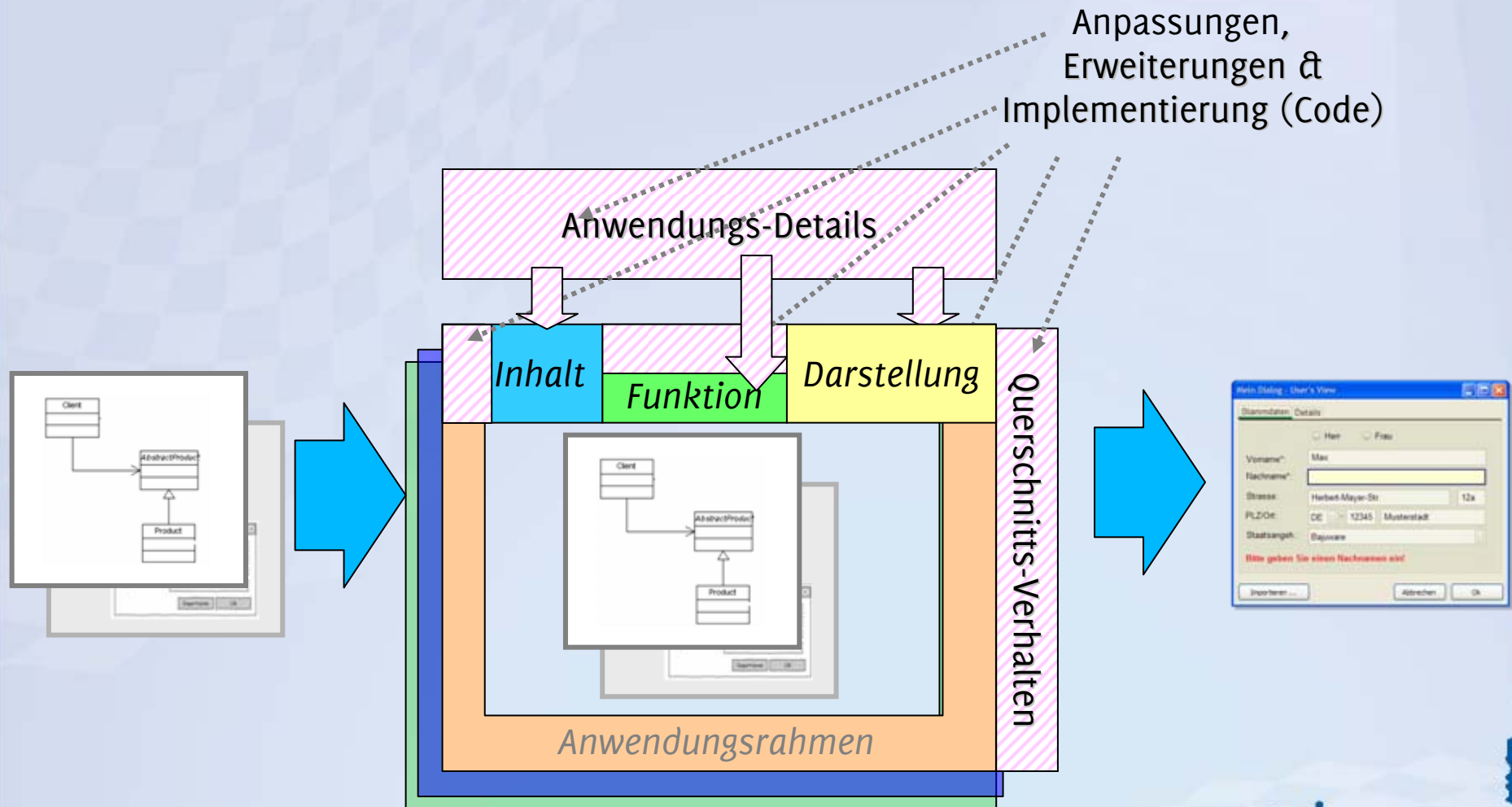
Rückblick: GUIs in der MDA



Rückblick: 100% GUI Modellierung



GUIs durch Modell + Code, aber wie?



Wir wollen: **Baukasten, aus-/anpassbare Einzelteile, Detailzugriff**

Was ist orchideo?



orchideo/suite

***Modellierungs-** und **Entwicklungsumgebung** für unsere Projekte auf Eclipse-Basis*



orchideo/views

*Modell-orientiertes Abstraktions**framework** zur effizienten Erstellung datenzentrischer GUIs auf unterschiedlichen GUI-Plattformen*

Der Weg...



1. Abstraktion
2. Modellnutzung
3. Deklaratives Vorgehen
4. Bausteine & -elemente



Der Weg...



1. Abstraktion

2. Deklaratives Vorgehen

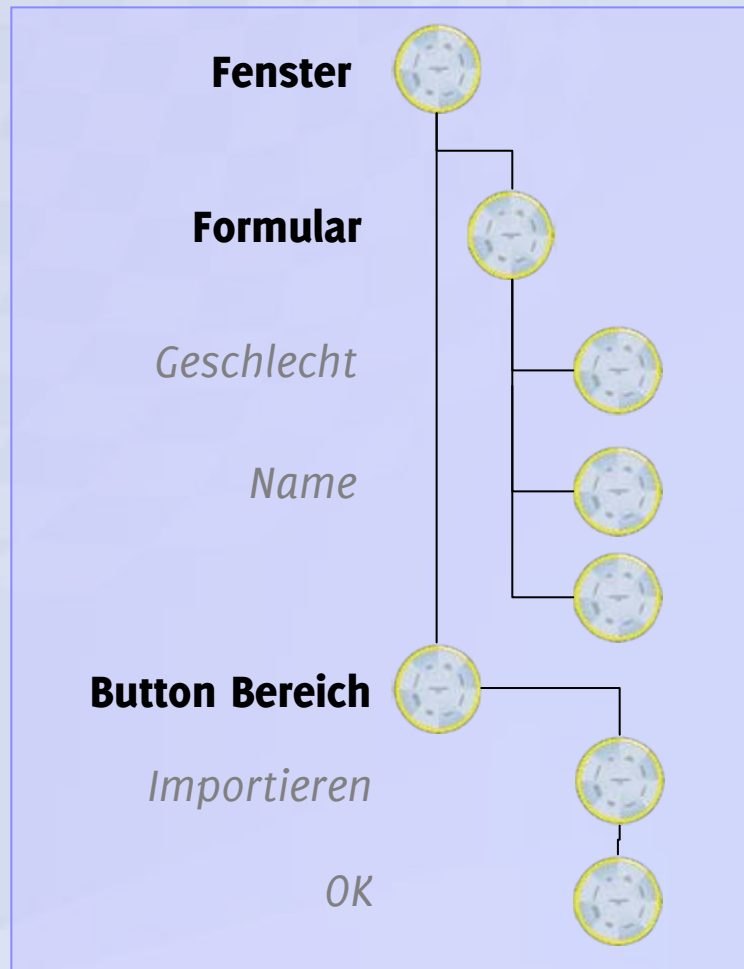
3. Delegation an Modellwissen

4. Bausteine & -elemente

- Zergliederung in gröbere Strukturen (Vereinfachung, DRY)
- Entkopplung (Technologie/Domänen)

Abstraktion: **Representation**

Representation : Darstellungsknoten mit „Features“



The concrete representation is a dialog box titled "Mein Dialog - Analysts's View". It contains the following fields and buttons:

- Geschlecht**: Text input field
- Name**: Text input field
- Strasse**: Text input field
- Land**: Text input field
- PLZ**: Text input field
- Ort**: Text input field
- Importieren**: Button
- Ok**: Button

Abstraktion: **Features**

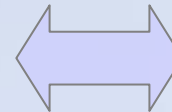
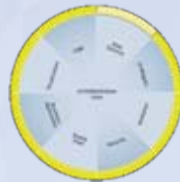
Feature : **Austauschbare** Querschnittsfunktionen aller Representations



- im Querschnitt
(Swing -> Echo3)
- als auch im Einzelnen
(Button -> Grafik-Button)
- Themen
 - Visualisierung
 - Data Binding
 - Validierung
 - Rechte & Editierbarkeiten
 - UI Ereignisse, ...

Überblick: **Abstraktionsebenen**

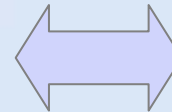
REPRESENTATION &
FEATURE



Importieren

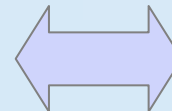
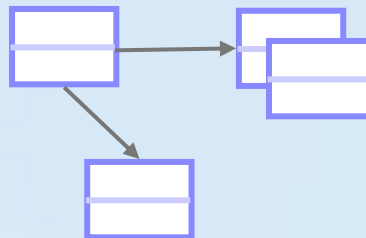
PROCESS & VIEW

Process
View



New Entry - Analyze's View	
Search:	
Name:	
Street:	
Land:	
PLZ:	
City:	
Importieren OK	

FLOW



Der Weg...

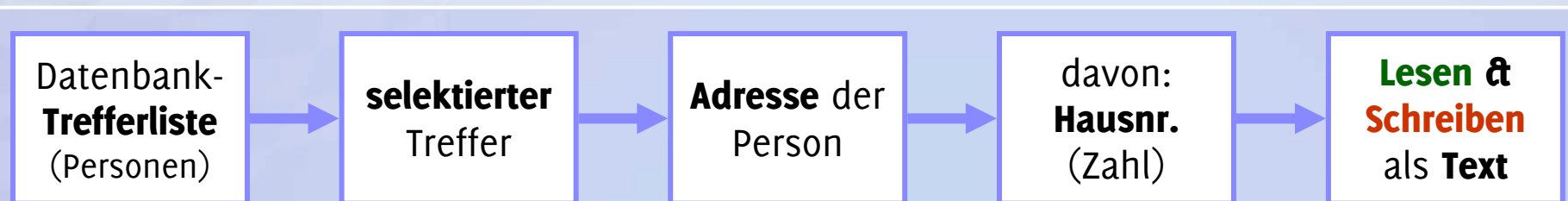


1. Abstraktion
- 2. Deklaratives Vorgehen**
3. Delegation an Modellwissen
4. Bausteine & -elemente

- Beschreibung der GUI via **Regeln** statt konkreter **Einzelbefehle**
- Tor zur **Modellinterpretation**

Deklarative Formulierung von

- **Datenanbindung** → Lazy Loading, Transformation, etc.
- **GUI-Layout** → Dynamische Formulare
- **Darstellungs-Modi** → Regelbasierte Rechteabbildung
- **Dialogablauf**



```
Source<String> hnr = Provider.of(result).at(selectHandle).get(Person.ADRESS)  
                    .get(Adress.HNR).as(String.class);
```

```
hnr.get();           // Lesen  
hnr.set(„7“);       // Schreiben
```

Beispiel: Deklarative Datenanbindung

Warum nun deklarative Struktur?

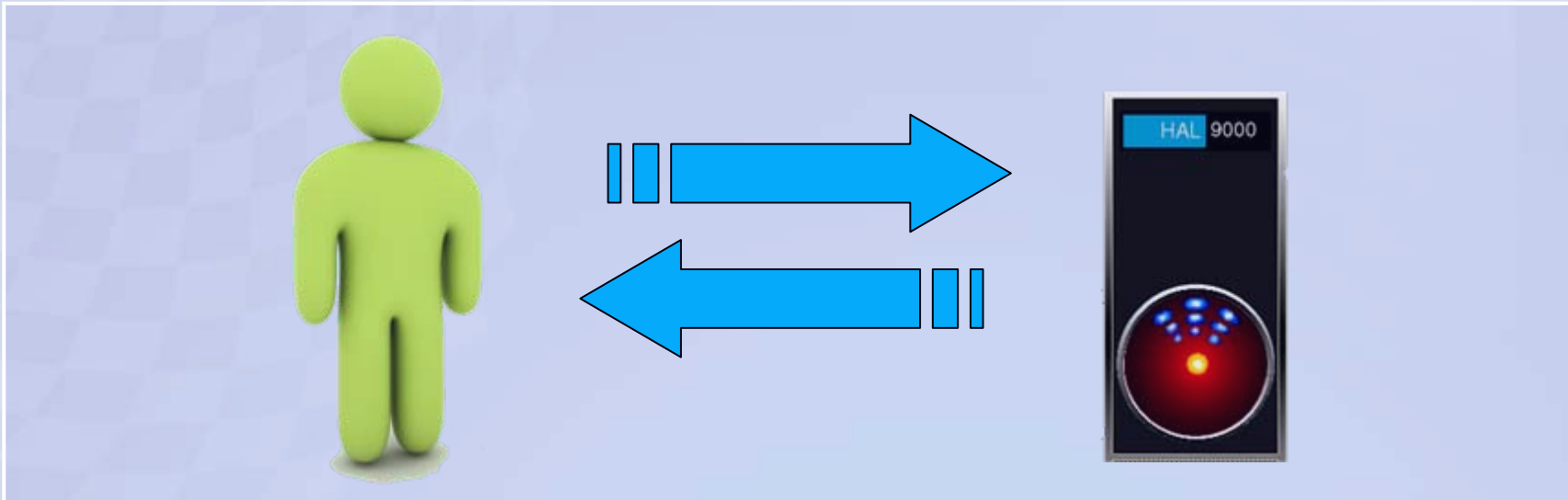
Beschreibung einer abstrakten **Regel** anstatt konkreter **Einzelbefehle**

- Einfacher
- Robuster & Effizienter
- ermöglicht Beschreibung
 - im Code
 - in einem Modell!



Strukturierung: **Lebenszyklus & Phasen**

Interaktions-Phasen



- Abbild des Dialogmodus von **Benutzeraktion** und **Systemreaktion**
- **Trigger** für die Interpretation der **deklarierten** Regeln

Der Weg...

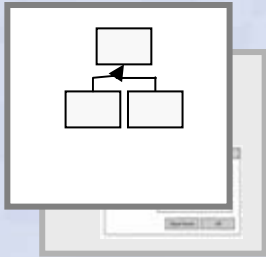


1. Abstraktion
2. Deklaratives Vorgehen
- 3. Delegation an Modellwissen**
4. Bausteine & -elemente

- *Modell-Abbildung*
geeigneter Aspekte
- GUI Deklaration:
Modellreferenz statt Code

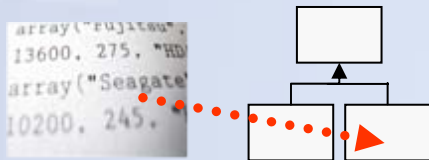
GUIs & Modelle?

Modellierung



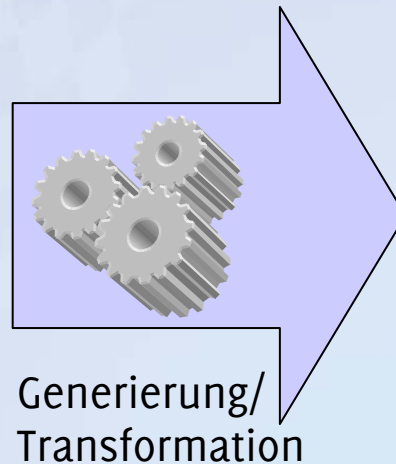
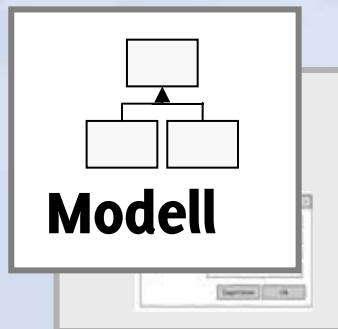
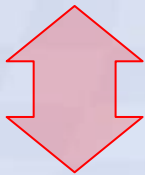
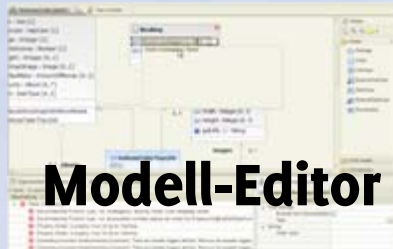
- Der **BO-Modelle**
- zusätzlicher **GUI Inhalte** & Modelle, wo **sinnvoll** → **DSLs**

Realisierung der GUIs

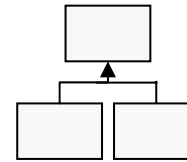


- Basierend auf den Modell-Inhalten...
... aber auch abweichend davon
- Generativ & Interpretativ

Modell-Artefakte & Nutzung



Laufzeit



Ladbares
**Laufzeit-
Modell**

Entwicklung & Laufzeit

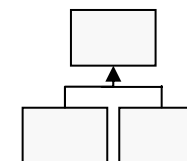


Business
Model Code



**GUI-Gerüst/
Prototyp**

Entwicklung



Code Modell-Referenzen

```
array("Fujitsu",  
13600, 275, "HD",  
array("Seagate",  
10200, 245, "
```

Modellwissen

Gut an das Modell delegierbar:

- Datenanbindung
- Präsentationdetails (Namen, Typen, ...)
- Eingabevalidierungen
- Fachlichkeiten
z.B. Rollenabhängige Schreibrechte

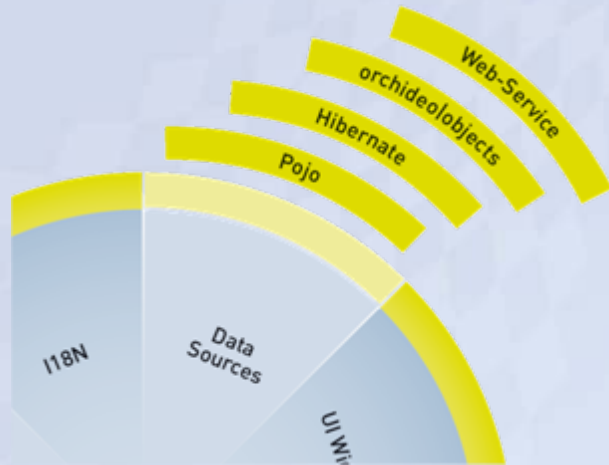
Der Weg...



1. Abstraktion
2. Deklarative Struktur
3. Delegation an Modellwissen
- 4. Bausteine & -elemente**

- *Flexibilität:*
Vom Grundpfeiler bis ins Detail
- *Effizienz und Wiederverwendung*

Technologie- & Lösungs-„Kits“



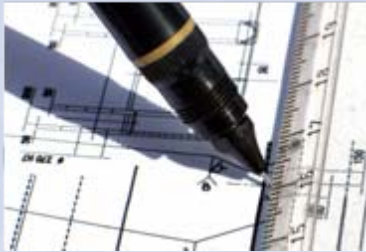
FRAMEWORK-KERN definiert
Abstraktionen & Strukturen

KITS liefern
Implementierungs-**Einzelteile**

- Technologie/Modell-Adapter
- Lösungsbausteine



Die Summe der Teile



EFFIZIENT?



Setups

- sind Templates
- Definieren Annahmen
 - für die Interpretation des Modells
 - in der Gestaltung der Projekt-API
- Liefern den sinnvollen ‚Standard‘fall



...und was ist nun mit den **Details**?

- **Rahmen**code in der Hand
- **Tauschbare** & **konfigurierbare** Einzelteile
- API-,Durchgriff' auf **Technologie**

```
myButton.getVisualizer().getUI().setWidth(..)
```

Core

Feature

Echo3 Kit

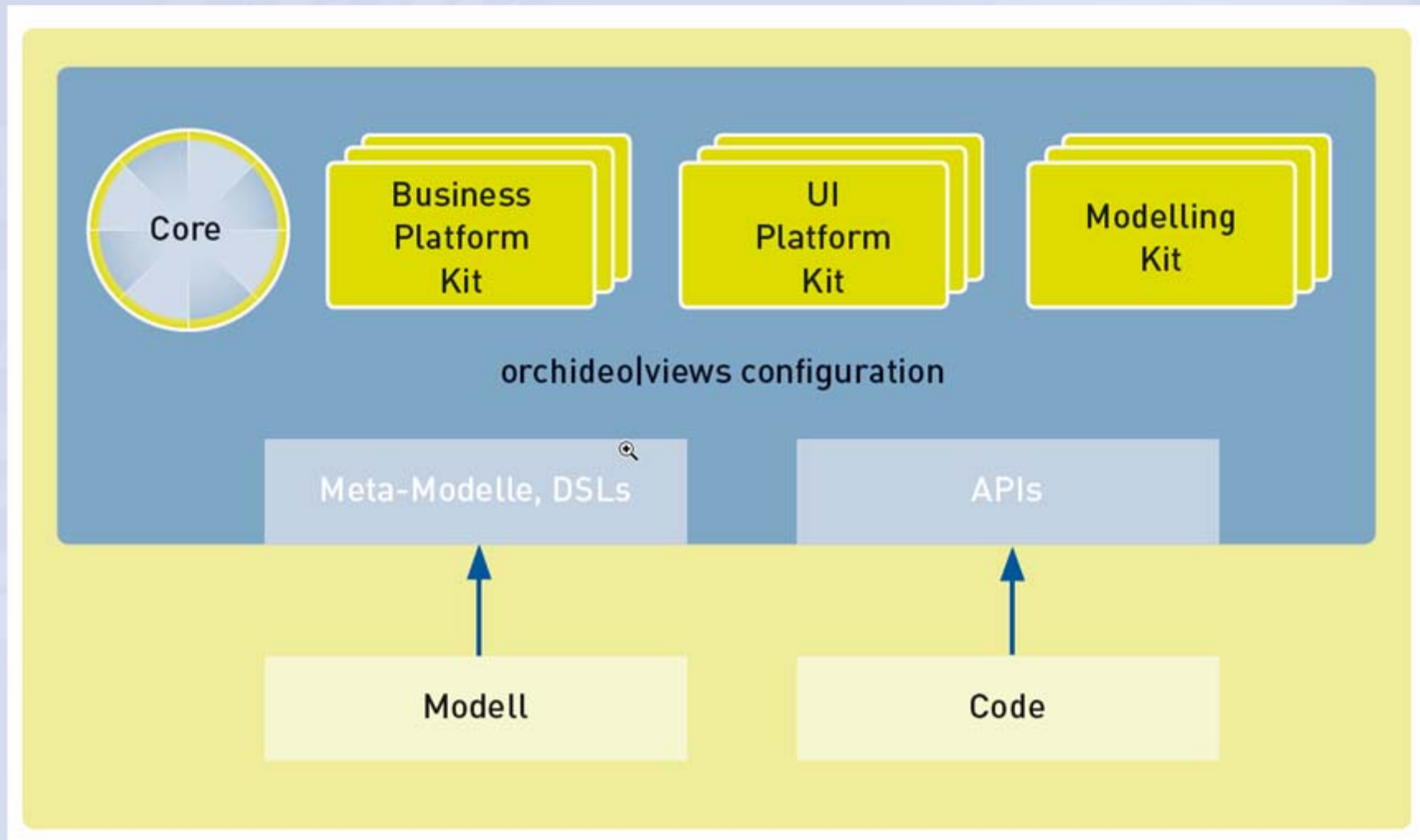
Echo3

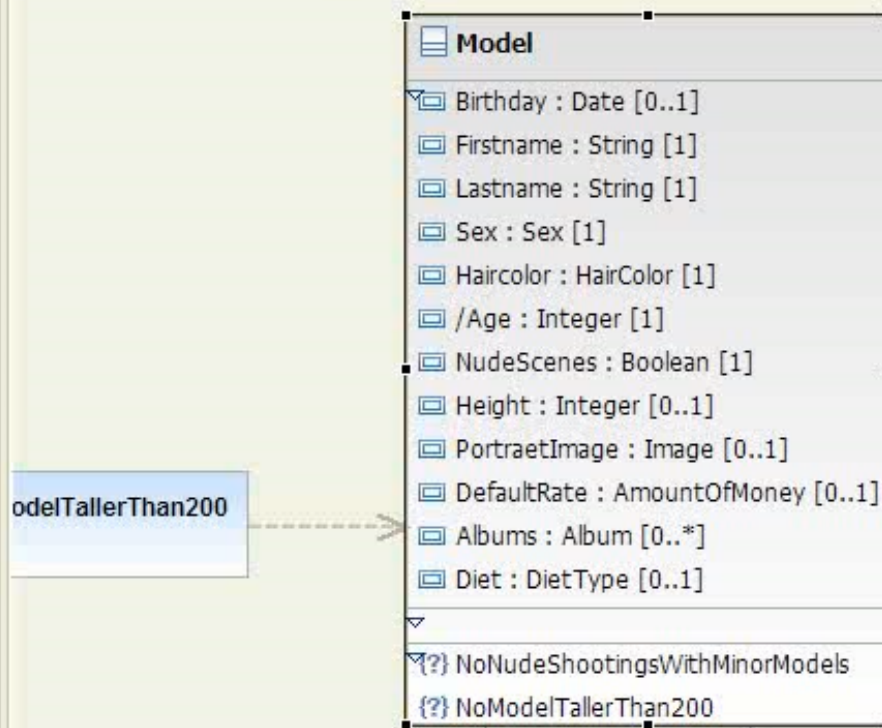
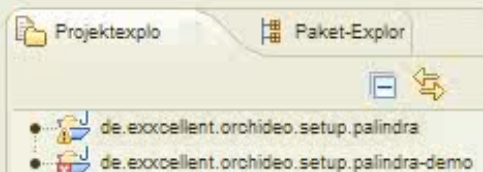
A screenshot of a Java Swing dialog box titled "Mein Dialog - Analysts's View". It contains several text input fields for "Geschlecht", "Name", "Strasse", "Land", "PLZ", and "Ort". At the bottom right, there are two buttons: "Importieren" and "Ok".



A screenshot of a Java Swing dialog box titled "Mein Dialog - User's View". It has a tabbed interface with "Stammdaten" and "Details" tabs. The "Details" tab is active, showing radio buttons for "Herr" and "Frau", and text fields for "Vorname*", "Nachname*", "Strasse", "PLZ/Ort", and "Staatsangeh.". The "Nachname*" field has a red error message below it: "Bitte geben Sie einen Nachnamen ein!". At the bottom, there are buttons for "Importieren ...", "Abbrechen", and "Ok".

Anwendung mit orchideo/views





Booking

Model : Model
DailyRate : AmountOfMoney

Image

RelativePosition : Integer
Width : Integer
Height : Integer
getURL() : String

Palette

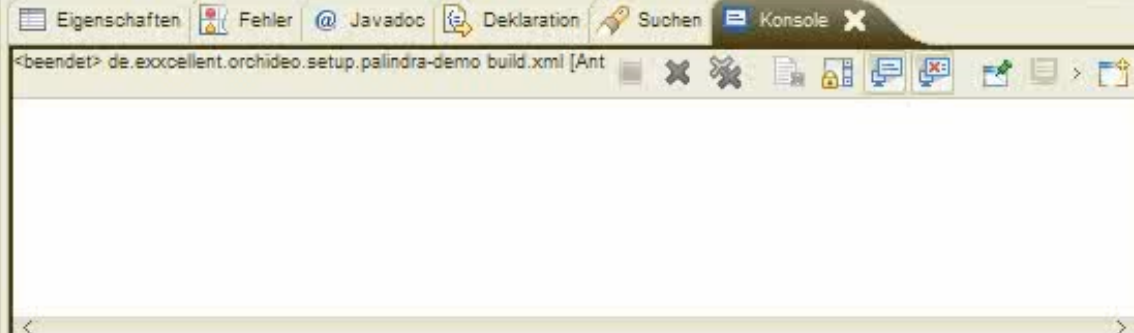
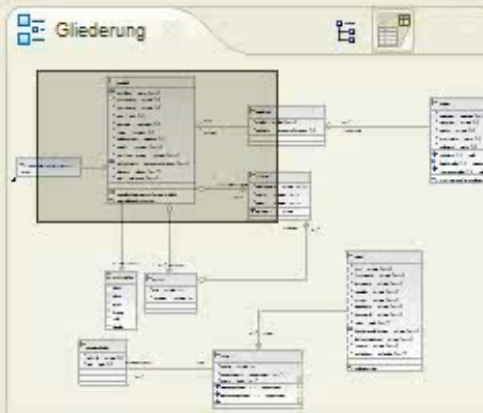
Nodes

- Package
- Class
- Interface
- ExternalInterface
- DataType
- ExternalDataTransfer
- Enumeration

Child Nodes

- Property
- Operation
- Constraint
- EnumLiteral
- StateMachine

Connections





Vielen Dank für Ihre Aufmerksamkeit!

Referenzen:

Naked Objects: <http://www.nakedobjects.org/>
Dialog Flow Notion: <http://bit.ly/9XPg1c>

XUL Tutorial: <https://developer.mozilla.org/en/XUL>
orchideo: <http://www.exxcellent.de/>