

MDSD Workshop Universität Hamburg

Ralph Winzinger, Senacor

Agenda 09.01.2013

- ⌚ Vorstellung R. Winzinger, Senacor & MDSD Projekte
- ⌚ DSLs textuell & grafisch
- ⌚ Übungsblock 1: getting started with Xtext / xtend
- ⌚ Hello World
- ⌚ (Conway's Game of Life)
- ⌚ Q&A, Diskussion

Agenda 16.01.2013

- ⌚ Übungsblock 2: some more Xtext and Xtend
- ⌚ Editor: Validierungen, Hilfestellungen, Formatierung, ...
- ⌚ Xtend: Sprachkonstrukte, (Code-)Generierung
- ⌚ Build: Integration & Automatisierung
- ⌚ Best Practices
- ⌚ Q&A, Diskussion

Ralph Winzinger & Senacor

RALPH WINZINGER
PRINCIPAL ARCHITECT

 SENACOR

Beratungsschwerpunkte

- Enterprise Architektur
- Legacy-Integration/-Migration
- Software-Engineering
- JEE, SOA, Webservices, SAP Integration, OSGi, Frontendtechnologien, Mobile



Branchenfokus

- Banking

Ausbildung und berufliche Erfahrung

- Über 10 Jahre Erfahrung in der Anwendungsentwicklung, technischen Projektleitung und Architekturberatung im Bankenumfeld
- Architekt, Senacor Technologies AG (seit 2000)
- Freiberufliche Tätigkeit im Bereich Mobile Computing (1996 – 2002)
- Diplom in Informatik, Univ. Erlangen-Nürnberg

Warum DSLs?

Wer hat schon m DSLs gearbeitet?

Wer wusste vorher, dass  mit einer DSL arbeitet?

DSLs sind vielleicht nicht überall,
aber durchaus weit verbreitet.

Warum DSLs?

```
internalOnly (true|false) "false">

<!ELEMENT characteristic EMPTY>
<!ATTLIST characteristic
  name CDATA #REQUIRED
  value CDATA #REQUIRED>
]>

<processmodel>
  <domain name="Baufi" id="1">

    <module name="Vorgang" id="1">
      <state name="n.a." id="1">
        <transition name="externBearbeiten" state="extern in Bearbe...
        <transition name="bearbeiten" state="inBearbeitung"/>
        <transition name="internerVKBearbeiten" state="interner VK ...
      </state>

      <state name="extern in Bearbeitung" maps="vertrieb-inBearbeitu...
        <transition name="uebergeben" state="uebergeben"/>
        <transition name="interner VK in Bearbeitung" state="interner ...
      </state>

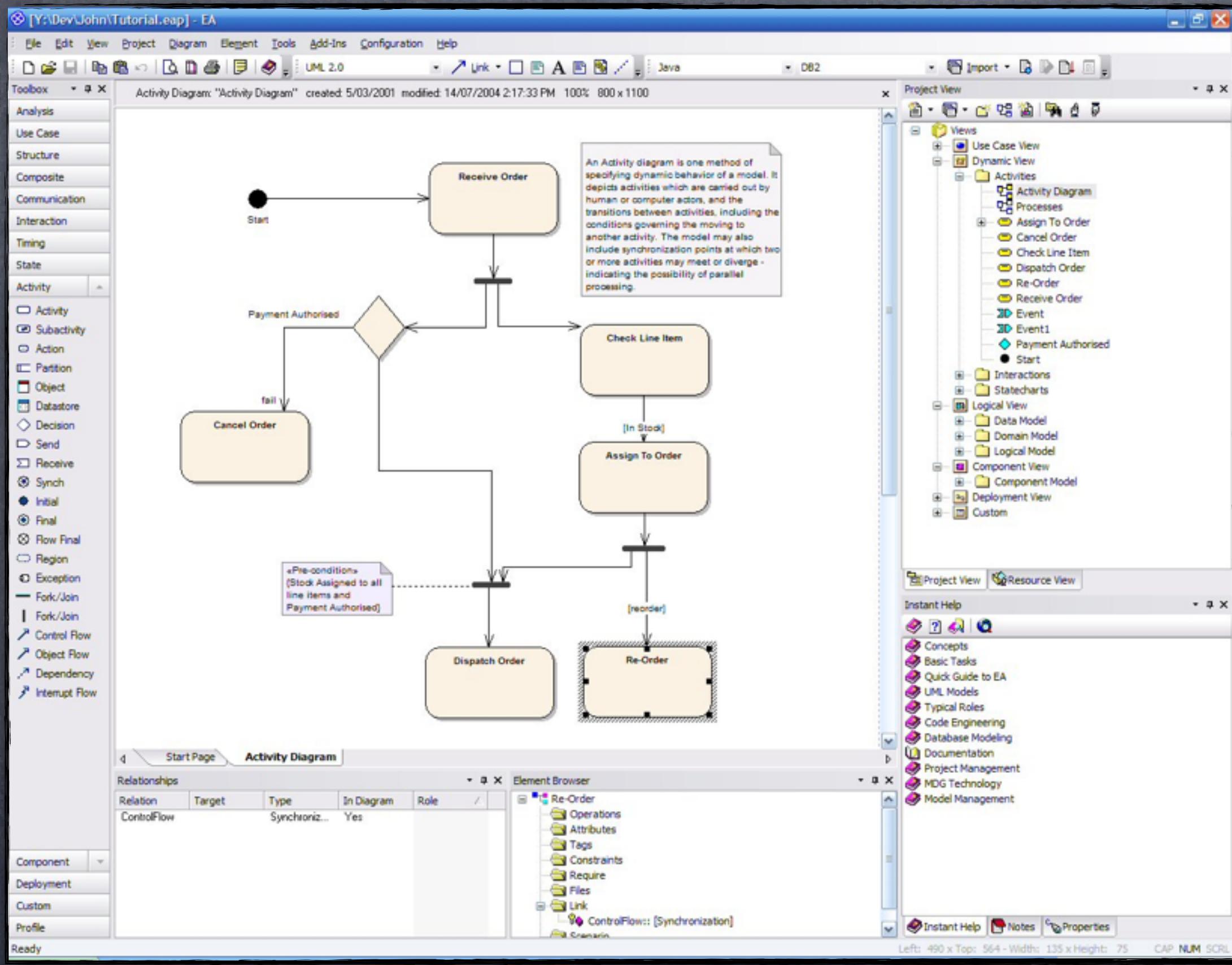
      <state name="uebergeben" id="3">
        <dependency module="PreScoring" state="vollstaendig"/>
        <transition name="bearbeiten" state="inBearbeitung"/>
      </state>

      <state name="frei" id="4">
        <transition name="bearbeiten" state="inBearbeitung"/>
        <transition name="zurueckstellen" state="zurueckgestellt"/>
      </state>

    </state>
    <transition name="zurueckstellen" state="zurueckgestellt"/>
    <transition name="durchsetzen" state="durchsetzen"/>
  </processmodel>
```

- ⌚ Abstraktion von eingesetzter Technologie
- ⌚ Annäherung an fachliches Vokabular, gemeinsame Diskussionsbasis

Warum DSLs?



☞ Dokumentation

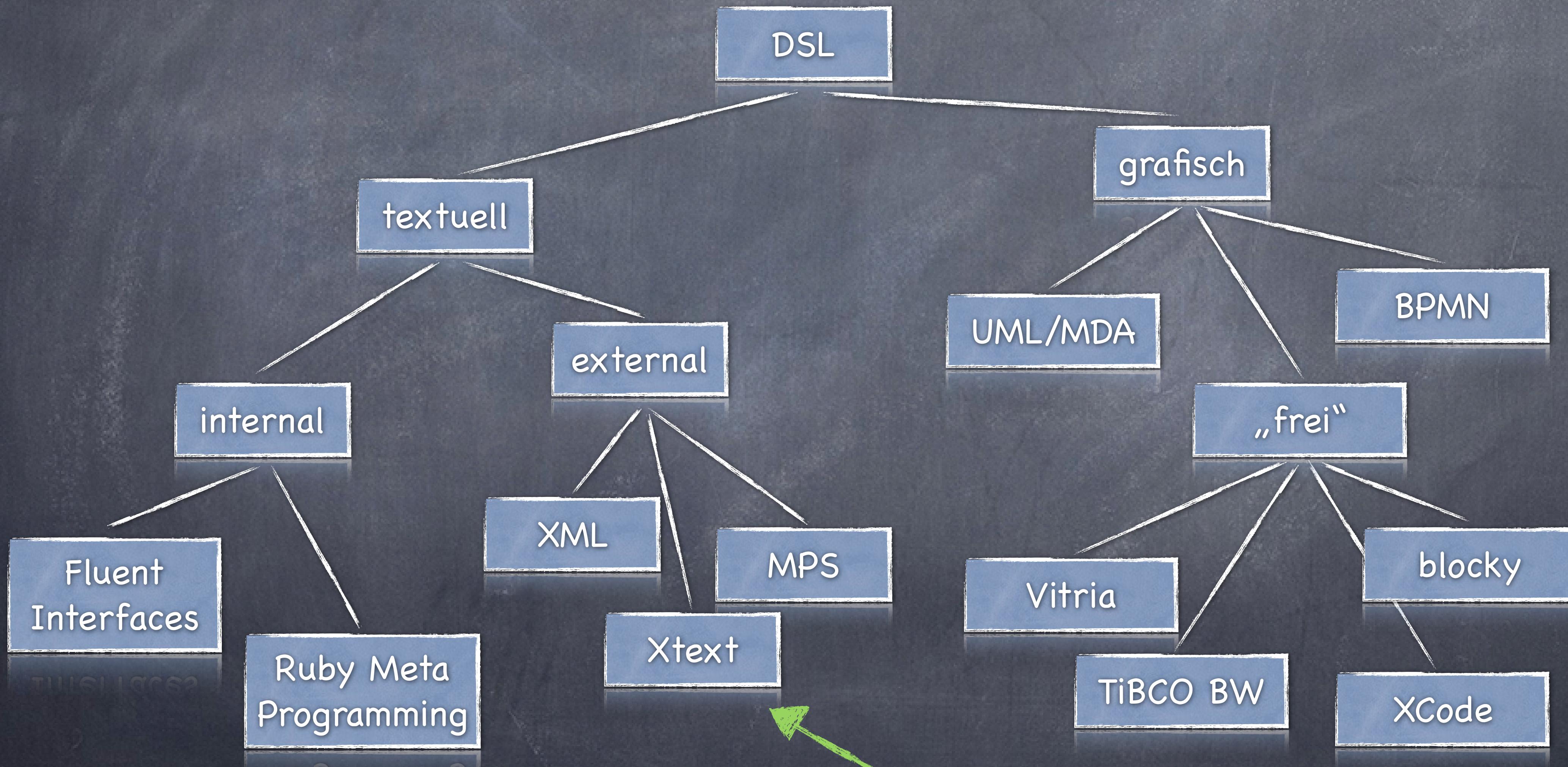
☞ Definition von
Sachverhalten mit
komplexen
Strukturen

Warum DSLs?



Dont
Repeat
Yourself

DSLs haben viele Gesichter



Welche DSL
und weshalb nun eigentlich?

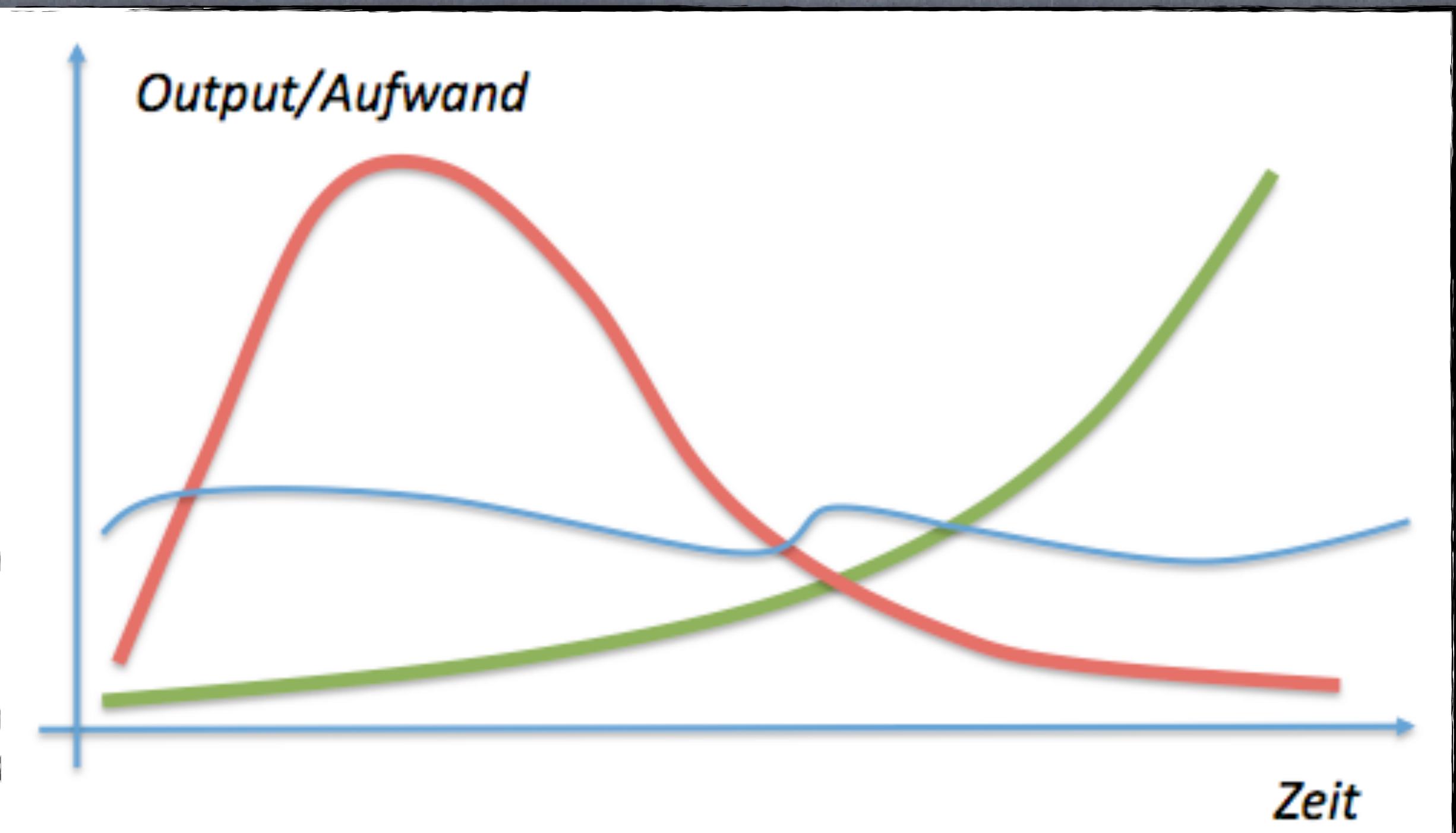
~~do obey „the law of the instrument“!~~

„do we have a problem here?“

- Lern- bzw. Effizienzkurve
- Problem von hinreichender Größe
- Problem von hinreichender Dauer
- Spricht Domain-/Projekt-/Sprachgröße für GPL oder DSL?

„do we have a problem here?“

- Lern- bzw. Effizienzproblem
- Problem von hinreichendem Output
- Problem von hinreichendem Aufwand
- Spricht Domain-/Problem aus?



561f

Grafisch oder Textuell?

- ⌚ Grafische DSL abstrahiert und vereinfacht - kann sie dem Problem gerecht werden?
- ⌚ Können komplexe Sachverhalte überhaupt sinnvoll dargestellt werden?
- ⌚ Können Lösungen effizient beschrieben werden?

Grafisch oder Textuell?

The image displays three distinct graphical interfaces used for programming or process definition:

- Top Left:** A logic simulation tool window titled "Common". It shows waveforms for signals HCLK, HSEL, HADDR, HTRANS, HREADY, and HREADYOUT. Below the waveforms is a Scratch-like script for a robot, starting with a "repeat while [true]" loop. Inside the loop, the robot moves forward if there is no wall to the left, turns left if there is a wall ahead and not to the right, turns right if there is a wall ahead, and turns right otherwise.
- Top Right:** A BPMN diagram titled "BusinessPartnerKeyProcess". It consists of a central rounded rectangle labeled "BusinessPartnerKeyProcess" connected to a "FromSapBpTriggerQ" input port at the top and a "ToSapBpQ" output port at the bottom. The "ToSapBpQ" port connects to another rounded rectangle labeled "BusinessPartnerProcess".
- Bottom Right:** A BPMN diagram titled "BusinessPartnerProcess". It features four parallel "FromSapBpQ" input ports (labeled Q1, Q2, Q3, and Q4) that converge on a single "BusinessPartnerProcess" rounded rectangle. This process then connects to a "BusinessPartnerStatusProcess" rounded rectangle and a "CompletionCallback" port.

Grafisch oder Textuell?

The screenshot illustrates the trade-off between graphical user interfaces and textual domain-specific languages (DSLs) in software development.

Left Side (Graphical User Interface):

- A screenshot of a web browser displaying a legal document (§ 102 Ergänzende Leistung). The content describes winter subsidies for employees.
- Below the document are links: "zum Seitenanfang" and "Datenschutz".
- The background shows a dark theme with "Datenscript" and "Schulungsvorstellung" visible.

Right Side (Textual DSL Definition):

- A screenshot of the DSL-Editor interface. The title bar reads "DSL-Editor v31.3.0 - BA-DSL-Perspektive - PRV_13.01.00.00 VERBIS/Funktionen/Unterfunktion/Personenverwaltung/Funktion_ANZb_AnrechnungszeitBearbeiten/Maske_ANZb_AnrechnungszeitBearbeiten.gui - DSL-Editor".
- The central area shows a textual DSL definition for a "Maske_ANZb_AnrechnungszeitBearbeiten" (Mask for Anrechnungszeit Bearbeitung). The code includes sections for "titel", "beschreibung", and "inhalt". It uses conditionals like "wenn #IstAnrechnungszeitAV = ja" and handles multiple cases for "Meldungsgrund".
- On the left, a tree view shows the project structure under "BA-DSL-Perspektive", including "Funktion_EGVd_Eingliederu", "Funktion_EGv", "Funktion_IPLE_EintraegeDes", and "Unterfunktion".
- At the bottom, there are tabs for "Fehler", "Protokoll", "Fortschritt", "Suchen", and "Maske_ANZb_AnrechnungszeitBearbeiten.html".

Tooling

- ⦿ Bei grafischen DSLs in der Regel vorgegeben, keine Anpassungsmöglichkeiten
- ⦿ Maturity
- ⦿ Benutzbarkeit (Zielgruppe!)
- ⦿ Integration in den Entwicklungsprozess
- ⦿ Refactoring

Collaboration & Versioning

- Sind Modellierer Teamplayer oder Einzelkämpfer?
- Diff & Merge, insbesondere bei grafischen DSLs
- Wird Versionierung angemessen unterstützt?
- Wird gewähltes Vorgehen unterstützt?

nicht zu kurz denken ...

- ⦿ Nicht nur primäre Artefakte modellieren/generieren!
- ⦿ Tests
- ⦿ Spezifikation, Dokumentation
- ⦿ Querschnittliche Aspekte
- ⦿ ...
- ⦿ Aber nur soweit sinnvoll!

<CODE>

Umgebung & Technologie



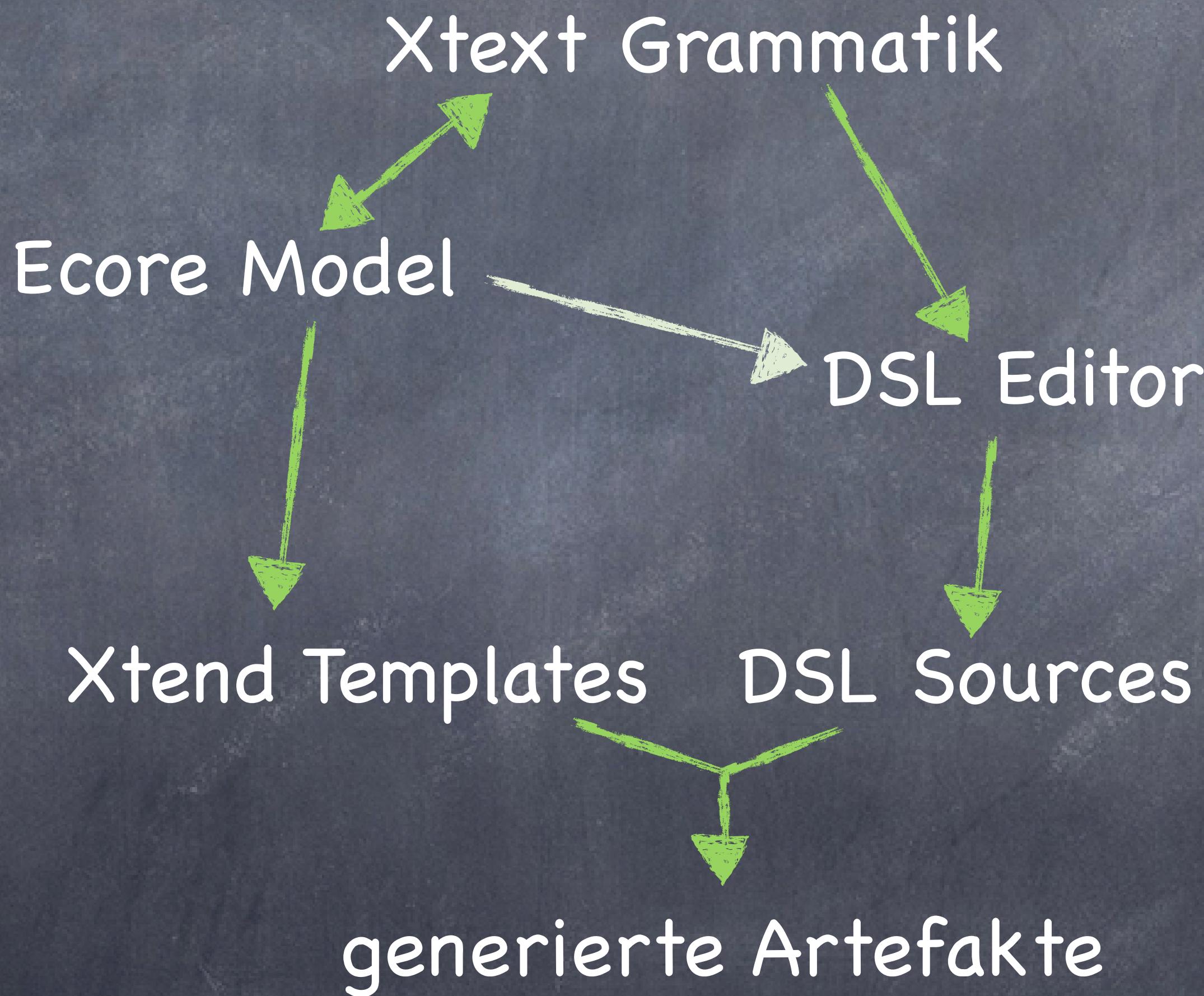
Umgebung & Technologie

- ⦿ Xtext & Xtend
- ⦿ Java basiert, Eclipse Projekte
- ⦿ Definition textueller DSL Grammatiken & Templating Engine
- ⦿ Automatisch generierter Editor für DSL Code
- ⦿ enge Integration mit weiteren Eclipse Projekten

Use Cases

- ⌚ Hello World
- ⌚ (nächste Woche) Game of Life

Xtext & Xtend Komponenten



Xtext Basics

Grammar:

```
'grammar' name=GrammarID
('with' usedGrammars+=[Grammar|GrammarID] (',' usedGrammars+=[Grammar|GrammarID])* )?
(definesHiddenTokens?= 'hidden' '(' (hiddenTokens+=[AbstractRule] (',' hiddenTokens
    +=[AbstractRule])* )? ')')?
metamodelDeclarations+=AbstractMetamodelDeclaration*
(rules+=AbstractRule)+
;
```

GrammarID returns ecore::EString:

```
ID ('.' ID)*;
```

```
AbstractRule : ParserRule | TerminalRule | EnumRule;
```

```
AbstractMetamodelDeclaration :
GeneratedMetamodel | ReferencedMetamodel;
```

usw. ...

EBNF

Xtext Basics

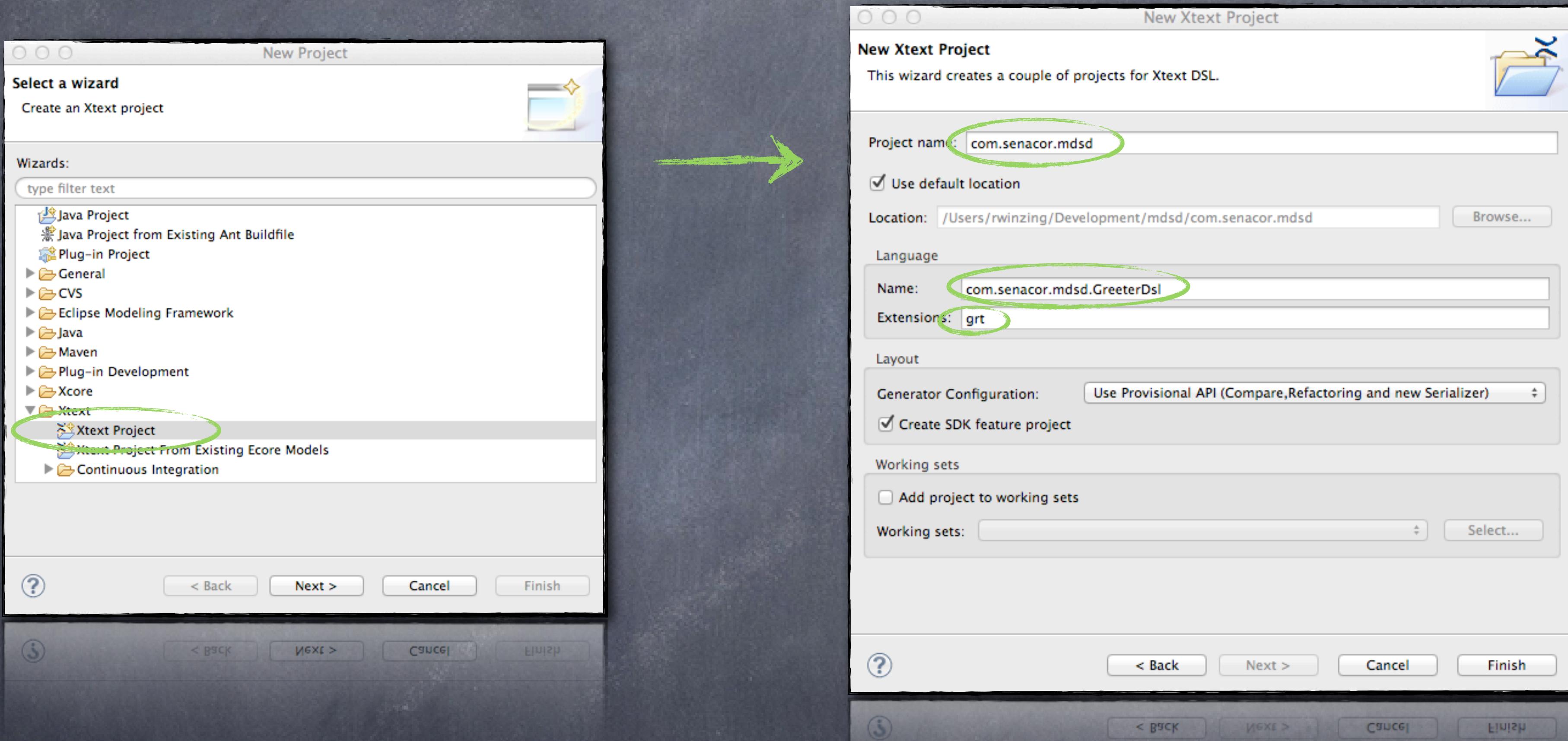
- ⌚ Rules
- ⌚ Terminal
- ⌚ DataType
- ⌚ Parser/Production/EObject
- ⌚ Keywords
- ⌚ Terminals
- ⌚ Assignments
- ⌚ Referenzen

```
Model:  
greetings+=Greeting*;  
  
Greeting:  
'Hello' name=ID '!';
```

Hello, World! (1)

- ➊ Projekt erzeugen
- ➋ Editor testen
- ➌ generierte Artefakte ansehen

Ein erstes Xtext Projekt ...



Hello, World! (2)

- etwas Struktur durch Klammern
- Name soll als String erfasst werden

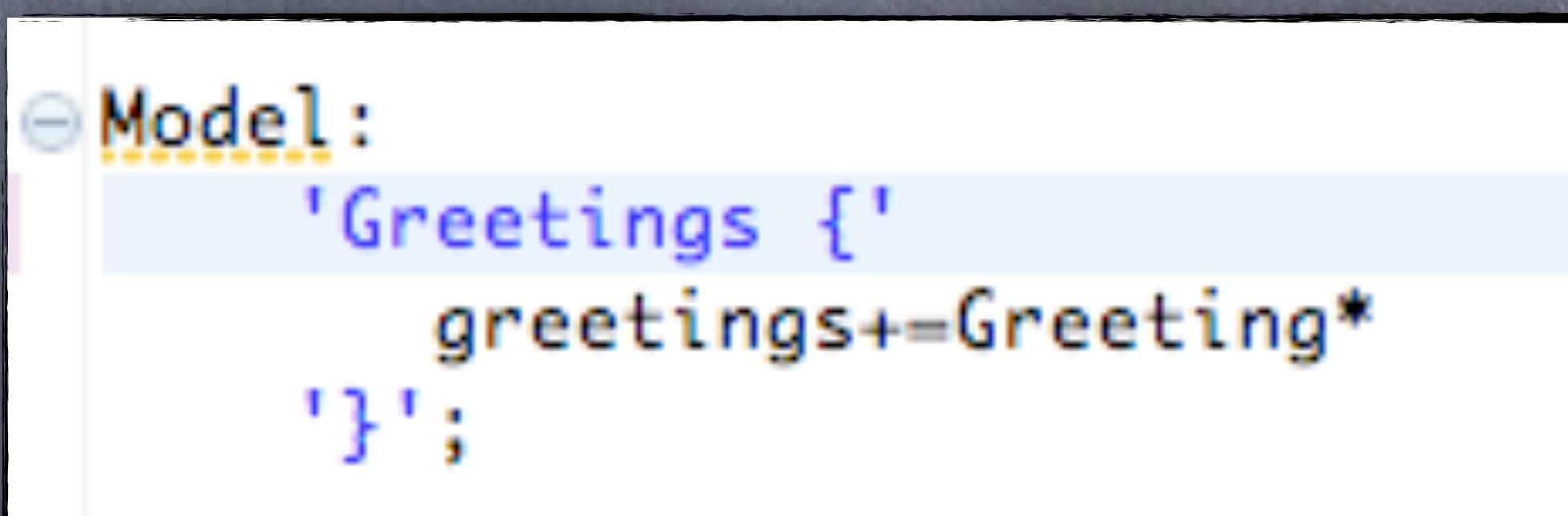


```
test.grt01 test.grt02
Greetings {
    Hello "World"!
}
```

```
test.grt01 test.grt02
Greetings {
    Hello "World"!
}
```

Hello, World! (2)

Wie wäre es mit



```
Model:
    Greetings {
        greetings+=Greeting*
    };
```

?

SPACES ARE EVIL!

Hello, World! (3)

- Alternative Grußtexte
- Eine zusätzliche Rule



```
Greetings {
    Hello "World"!
    Hallo "Welt"!
}
```

Hello, World! (4)

- ⌚ Grußworte als Enum
- ⌚ Kommentare



```
Greetings {
    Hello "World"!
    Hallo /* comment */ "Welt"!
}
```

Agenda 16.01.2013

- ⌚ Recap
- ⌚ Übungsblock 2: some more Xtext and Xtend
 - ⌚ Editor: Validierungen, Hilfestellungen, Formatierung, ...
 - ⌚ Xtend: Sprachkonstrukte, (Code-)Generierung
 - ⌚ Build: Integration & Automatisierung
- ⌚ Best Practices
- ⌚ Q&A, Diskussion

Recap

Model:

```
'Greetings' '{'  
    greetings+=Greeting*  
'}';
```

Greeting:

```
('Hello' | 'Hallo') name=ID '!'  
;
```

:

```
(Hello | Hallo) name=ID !
```

Recap

Model:

```
'Greetings' '{'  
    greetings+=Greeting*  
'}';
```

Greeting:

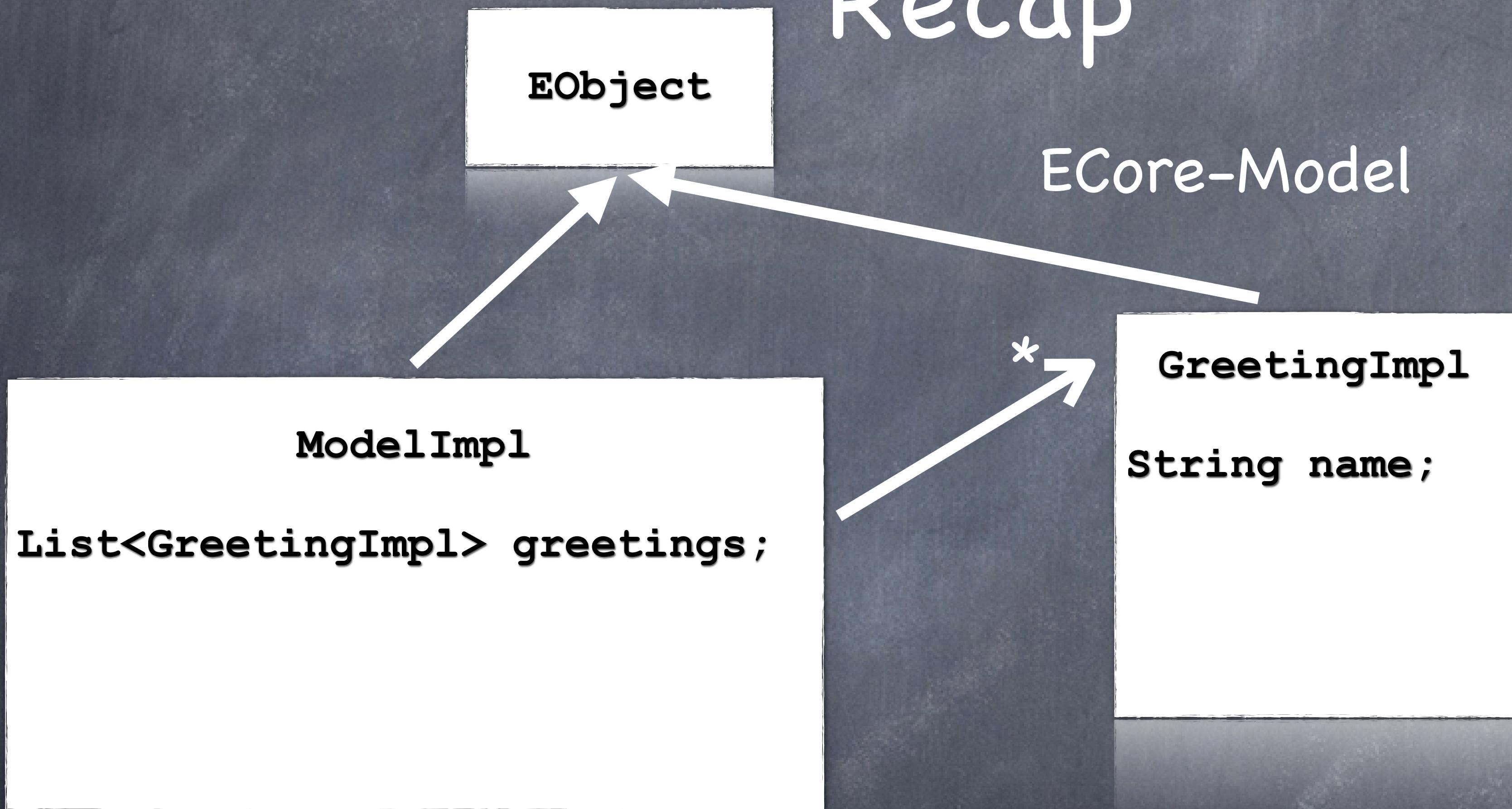
```
('Hello'|'Hallo') name=ID '!'  
;
```

:

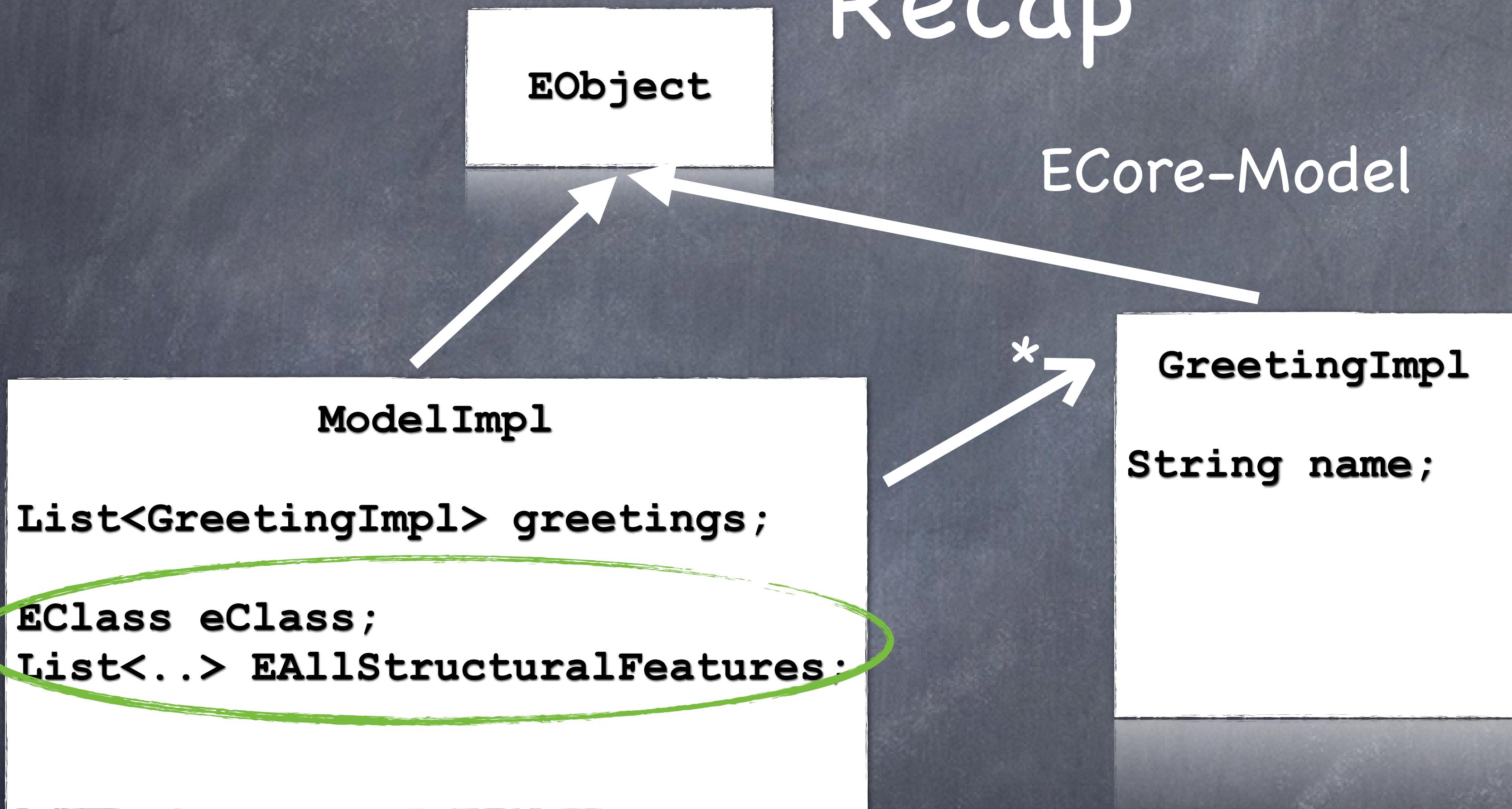
```
Greetings {  
    Hello Ralph!  
    Hallo Hamburg!  
}
```

```
public class GreetingImpl extends EClass  
{  
    protected String name = NAME_EDEFAULT;  
  
    protected GreetingImpl() {  
        super();  
    }  
  
    protected EClass eStaticClass() {  
        return GreeterDsl02Package.Literals.GREETING;  
    }  
  
    public String getName() {  
        return name;  
    }  
}  
}  
}
```

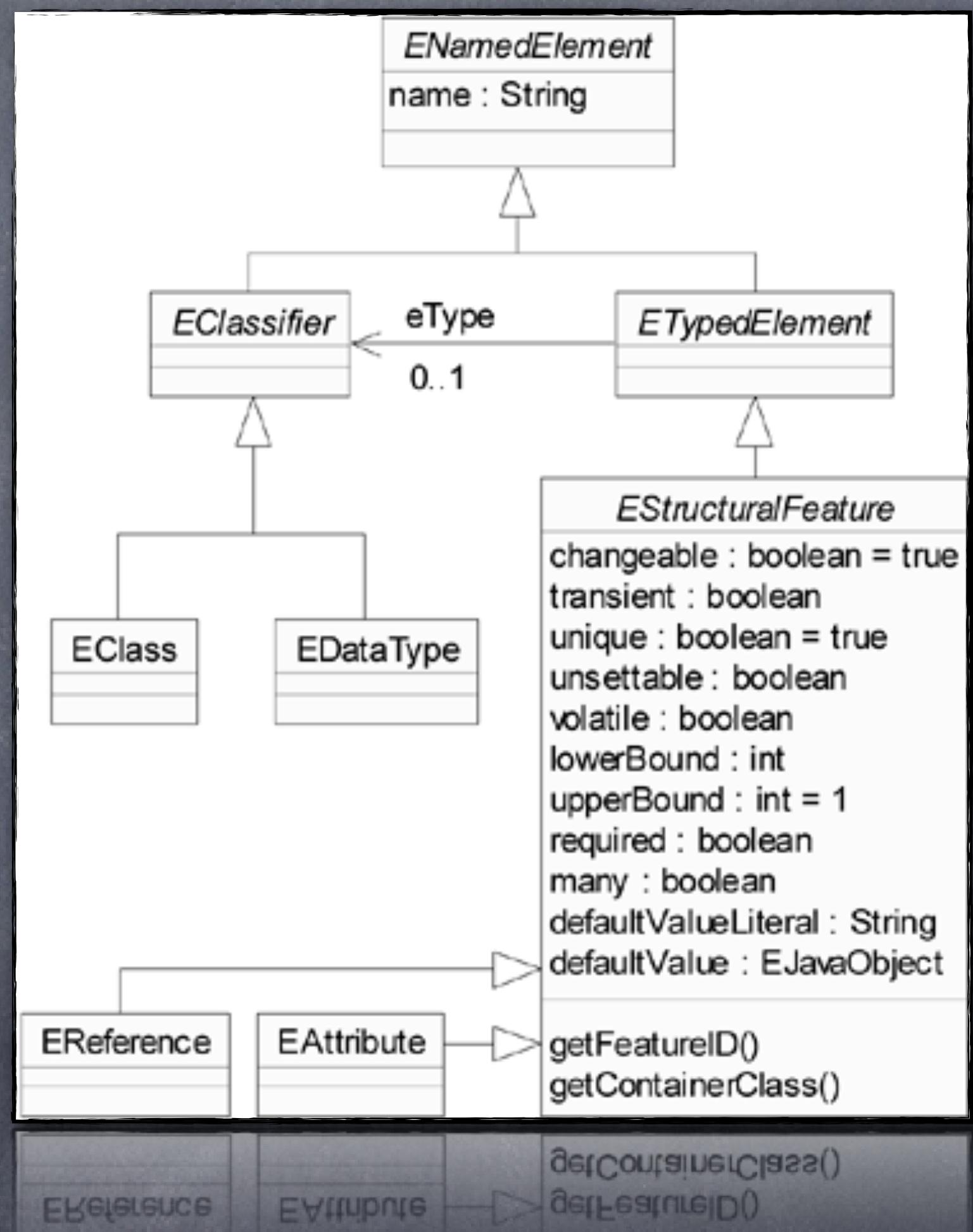
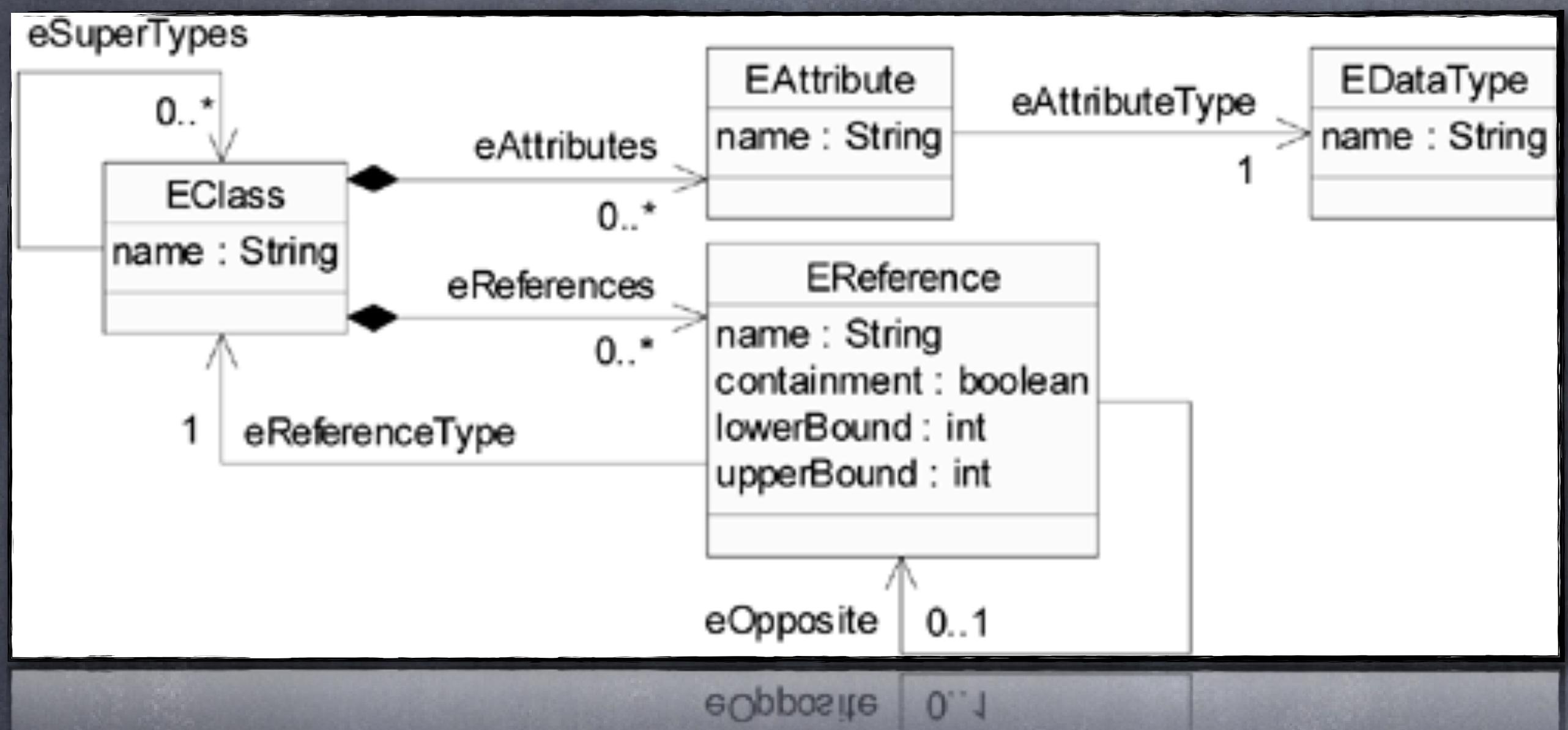
Recap



Recap



ecore Model



Hello, World! (5)

- Personen & Grüße
- Cross-References



The screenshot shows a text editor with two tabs: `test.grt01` and `*test.grt02`. The `*test.grt02` tab is active, displaying the following XML code:

```
Persons {  
    rw firstname Ralph lastname Winzinger  
}  
  
Greetings {  
    Hallo rw!  
}
```

]

Hello, World! (5)

```
grammar com.senacor.mdsd.greeter02.GreeterDsl02 with org.eclipse.xtext.common.Terminals
```

```
generate greeterDsl02 "http://www.senacor.com/mdsd/greeter02/GreeterDsl02"
```

Model:

```
'Persons' '{'  
    persons+=Person*  
'}'
```

```
'Greetings' '{'  
    greetings+=Greeting*  
'}';
```

Person:

```
name=ID 'firstname' firstname=ID 'lastname' lastname=ID gender=( 'w' | 'm' );
```

Greeting:

```
word=Greetword person=[Person] '!' ;
```

enum Greetword:

```
DE='Hallo' | EN='Hello' | FR='Bonjour' ;
```

```
DE='Hallo' | EN='Hello' | FR='Bonjour' ;
```

enum Greetword:

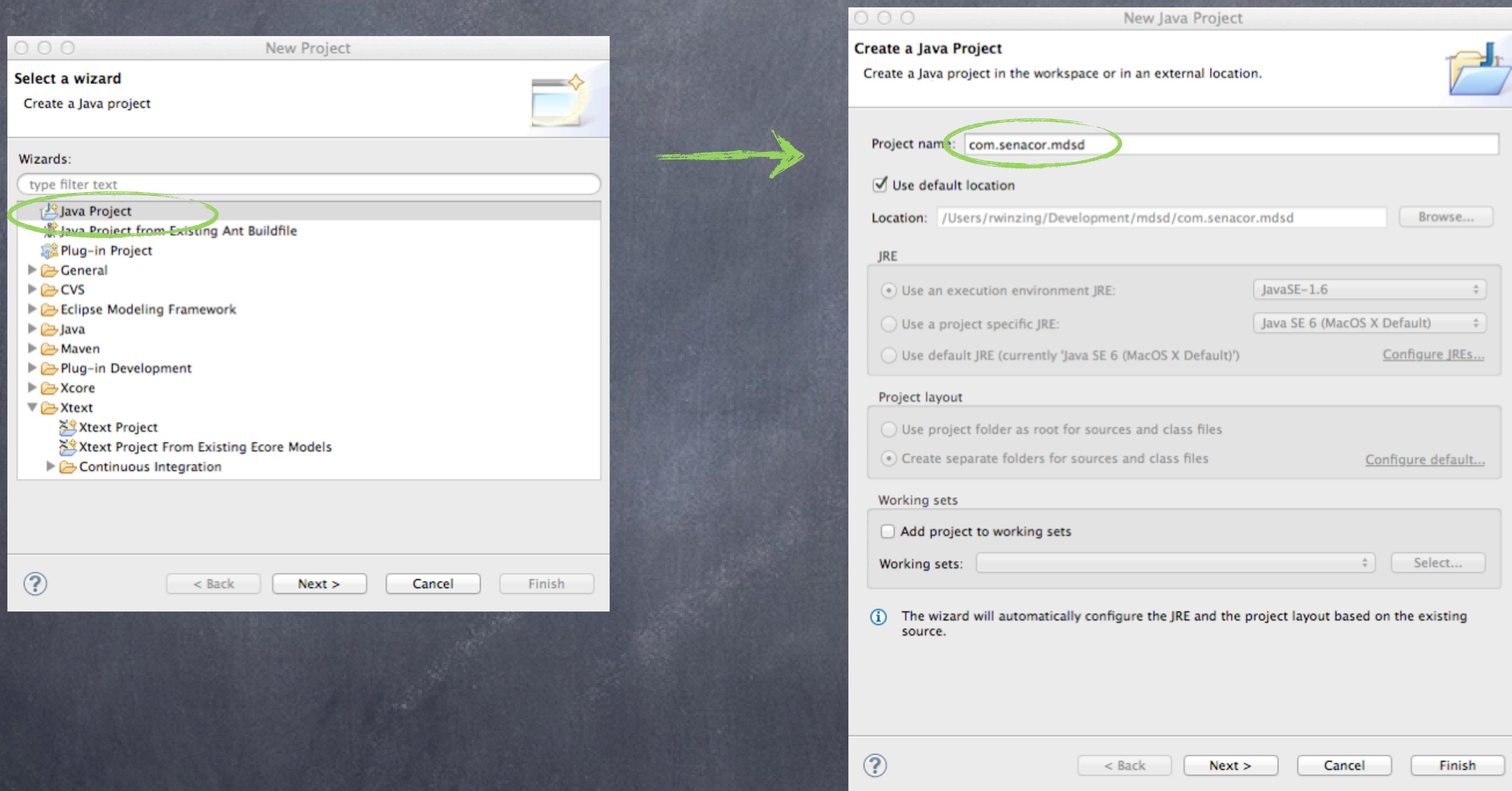
Xtend Basics

- ⦿ Java Erweiterung (kompiliert wieder zu Java-Quellcode)
- ⦿ Type extensions
- ⦿ Type inference
- ⦿ Keywords def, dispatch, val & var
- ⦿ Lambdas
- ⦿ Templates

Hello, World! (wieder mal)

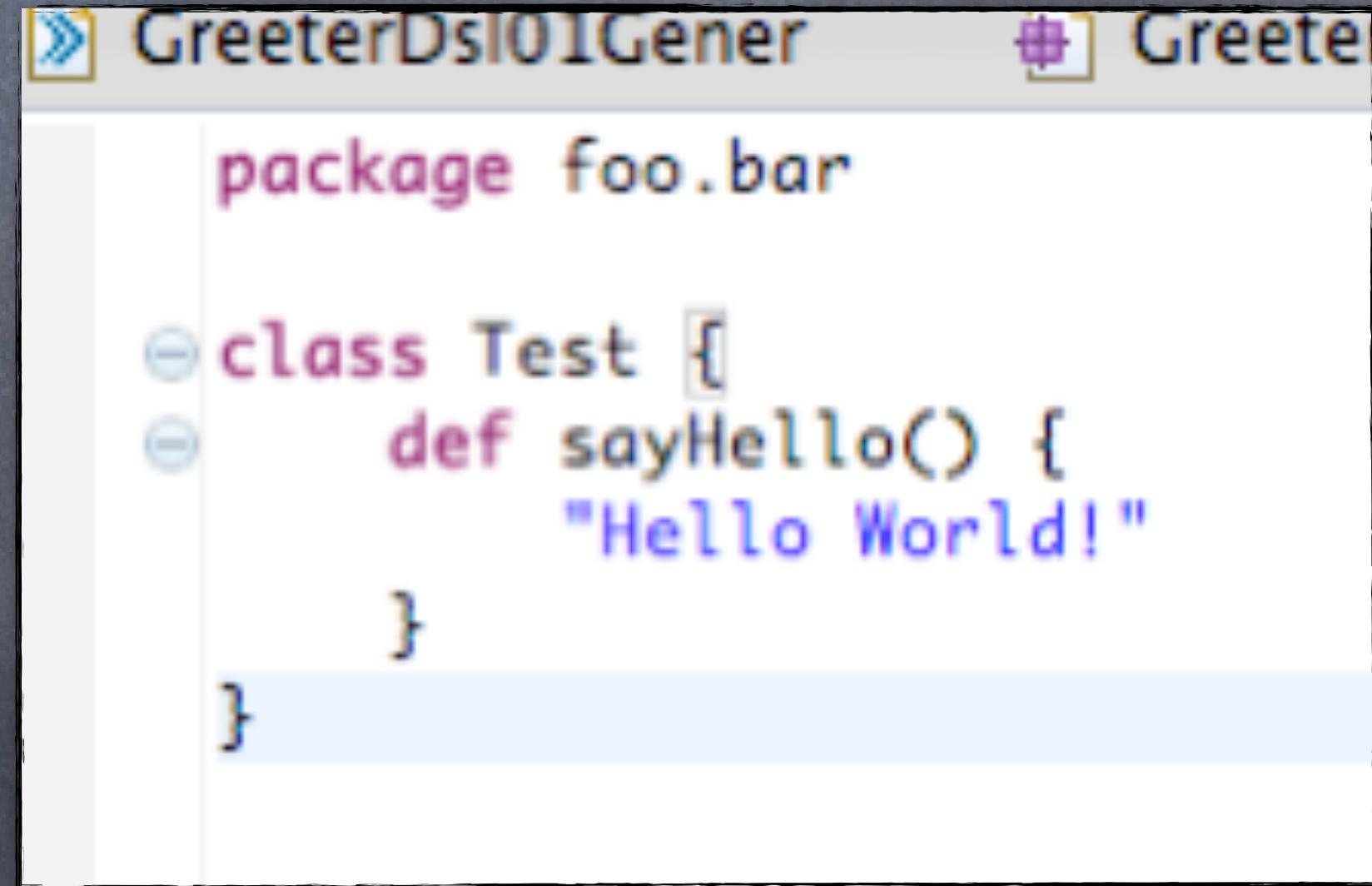
- ❶ Projekt erzeugen
- ❷ Xtend Klasse anlegen & speichern
- ❸ generierte Artefakte ansehen

Ein erstes Xtend Projekt ...



def, var & val

- „def“ deklariert eine Methode
- „var“ und „val“ deklarieren Variablen



```
GreeterDsl01Generator Greete
package foo.bar

class Test {
    def sayHello() =
        "Hello World!"
}
```

Transparenter Zugriff zwischen Java & Xtend

• „::“ für Zugriff auf statische Elemente



```
GreeterDSL01.ec GreeterDSL02.xt

package foo.bar

class Test {
    def printHello() {
        Util:::dump(sayHello)
    }

    def sayHello() {
        "Hello World!"
    }
}
```

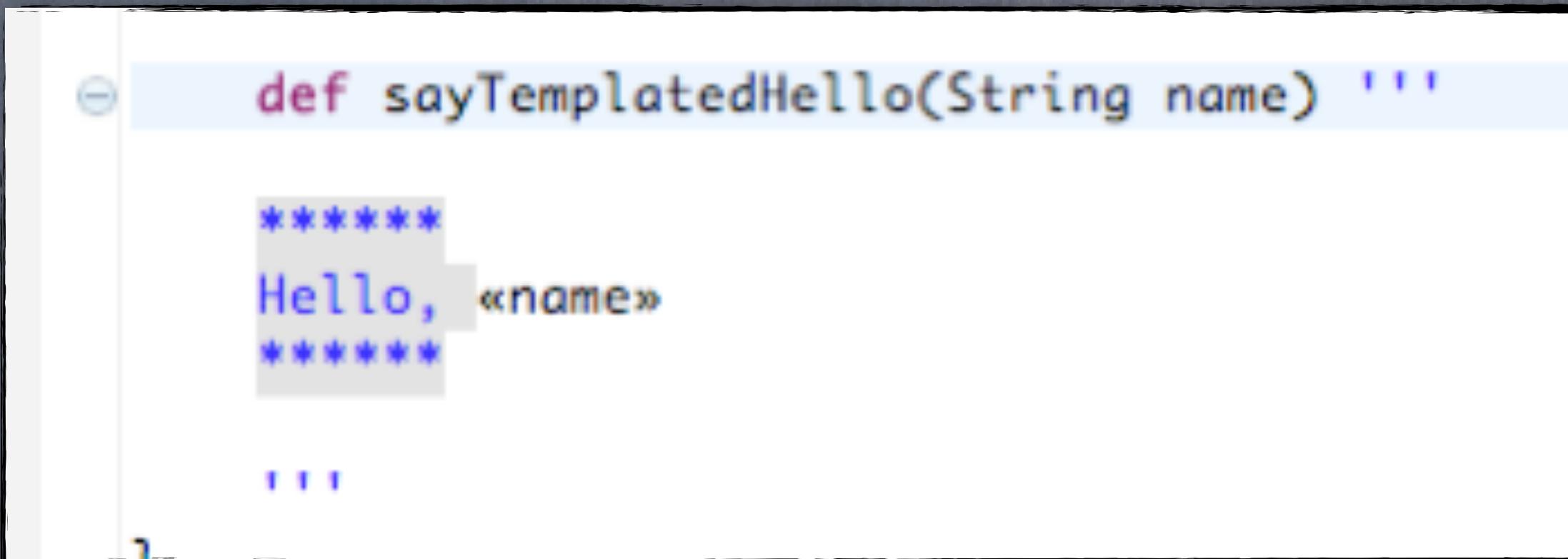
Weshalb eigentlich „Xtend“?

foo(a, b) kann man schreiben als a.foo(b)

```
def dumpToConsole(String s) {  
    System.out.println(s);  
}  
  
dumpToConsole(„hello, world!“);  
  
„hel  
    for (gw: resource.allContents.toIterable.filter(typeof(GW))) {  
        generateFile(gw, fsa);  
    }
```

Templates

- Triple-Quotes leiten ein Template ein
- String-Ausdruck
- IF / ENDIF
- FOR / ENDFOR
- BEFORE / SEPARATOR / AFTER



A screenshot of a code editor showing a Python function definition. The code is:

```
def sayTemplatedHello(String name) ***  
    ****  
    Hello, «name»  
    ****  
    ...  
]
```

The placeholder «name» is highlighted in blue, indicating it is a template variable.

putting stuff together ...

- xtext und xtend verbinden
- und ein wenig generieren

<CODE>