

# MDSL in der Praxis

Ralph Winzinger, Senacor

# Agenda 06.01.2016

- 🕒 Vorstellung R. Winzinger, Senacor & MDSD Projekte
- 🕒 DSLs textuell & grafisch
- 🕒 Hands-On: getting started with Xtext / Xtend
  - 🕒 Grammatikdefinition mit Xtext
  - 🕒 Codegenerierung mit Xtend
  - 🕒 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

Senacor Technologies AG

DA Siegelabendpost 10/2008

# Senacor & MDSD

# Warum DSLs?



# Warum DSLs?

Wer hat schon mal mit DSLs gearbeitet?



# Warum DSLs?

Wer hat schon mal mit DSLs gearbeitet?

Wer wusste vorher, dass er mit einer DSL arbeitet?

# Warum DSLs?

Wer hat schon mal mit DSLs gearbeitet?

Wer wusste vorher, dass er 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="interno...
      </state>

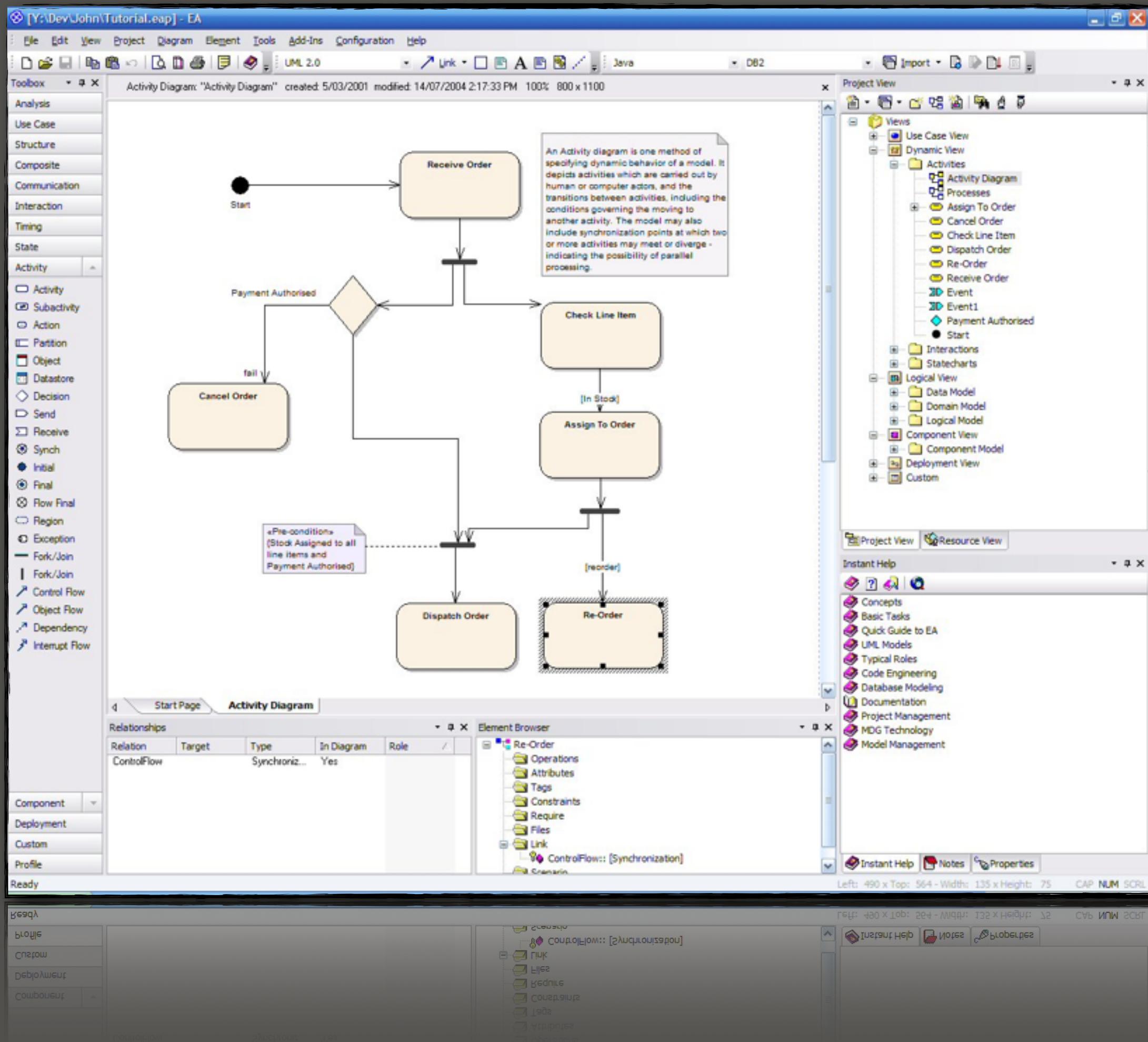
      <state name="uebergeben" id="3">
        <dependency module="PreScoring" state="vollstaendig"/>
        <transition name="bearbeiten" state="inBearbeitung"/>
      </state>

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

    </state>
    <!-->
    <!-->
    <!-->
  </processmodel>
```

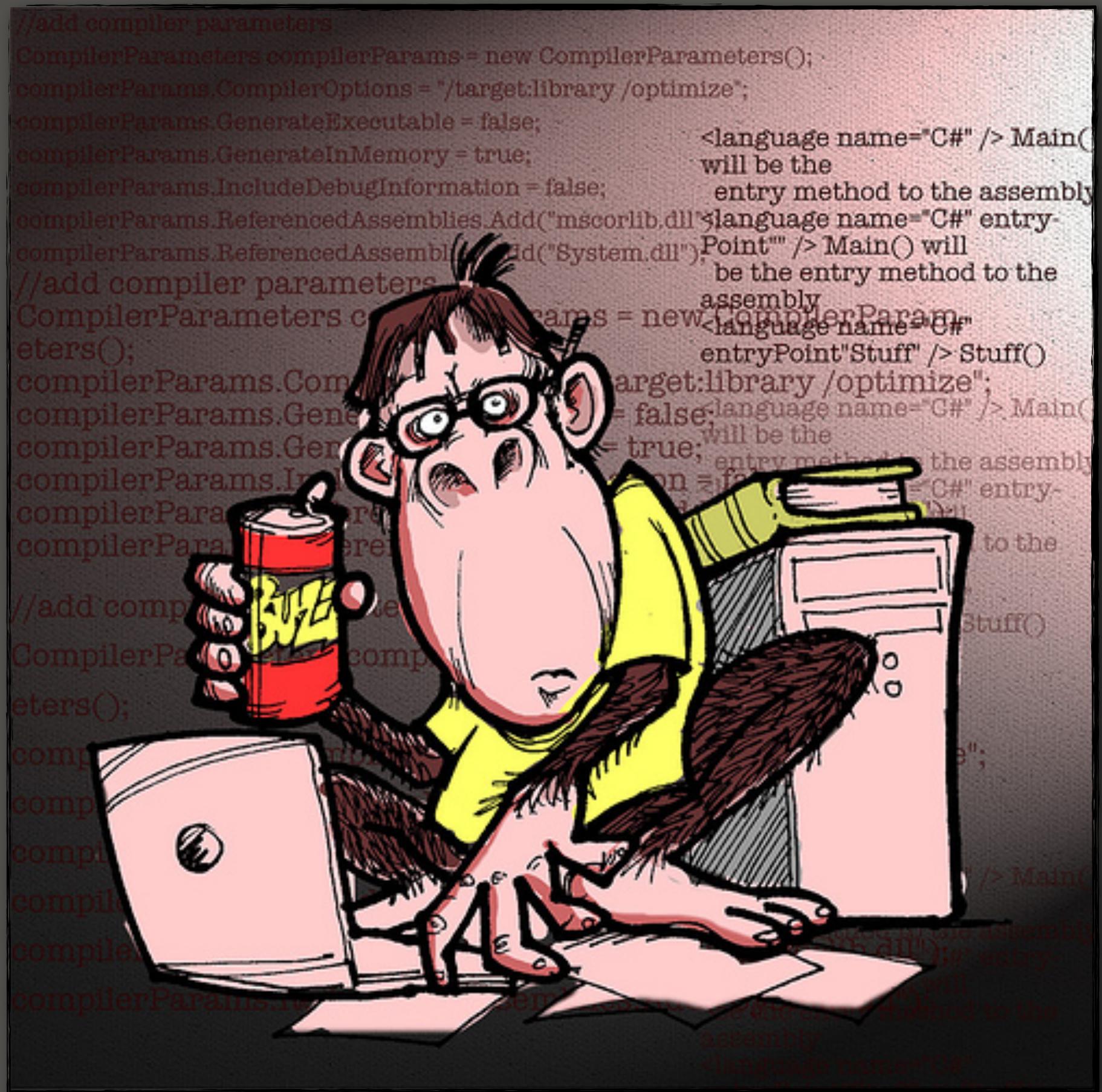
- ⌚ Abstraktion von eingesetzter Technologie
- ⌚ Annäherung an fachliches Vokabular, gemeinsame Diskussionsbasis

# Warum DSLs?



- ocular icon Dokumentation
- ocular icon Definition von Sachverhalten mit komplexen Strukturen

# Warum DSLs?

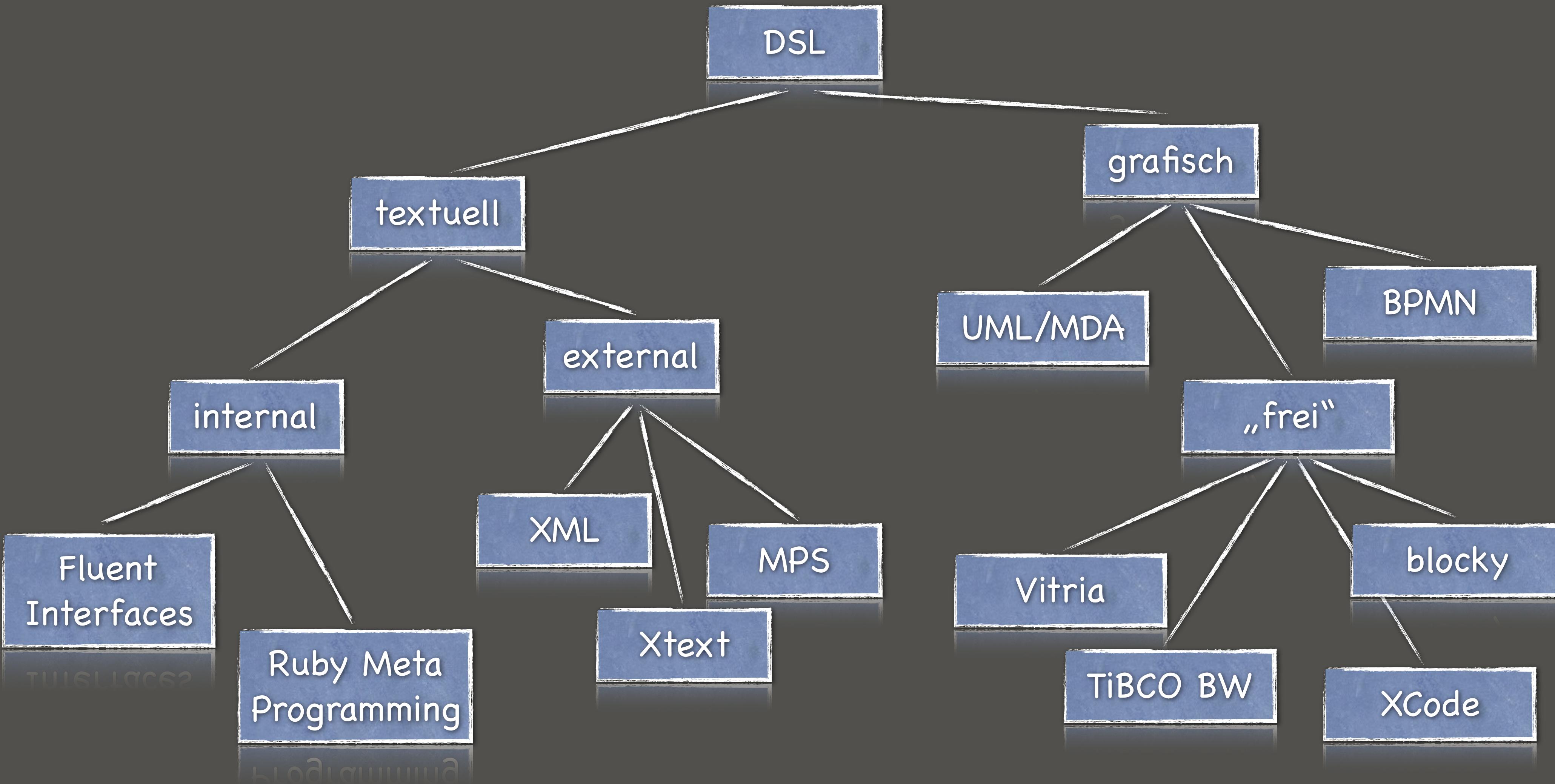


Dont  
Repeat  
Yourself

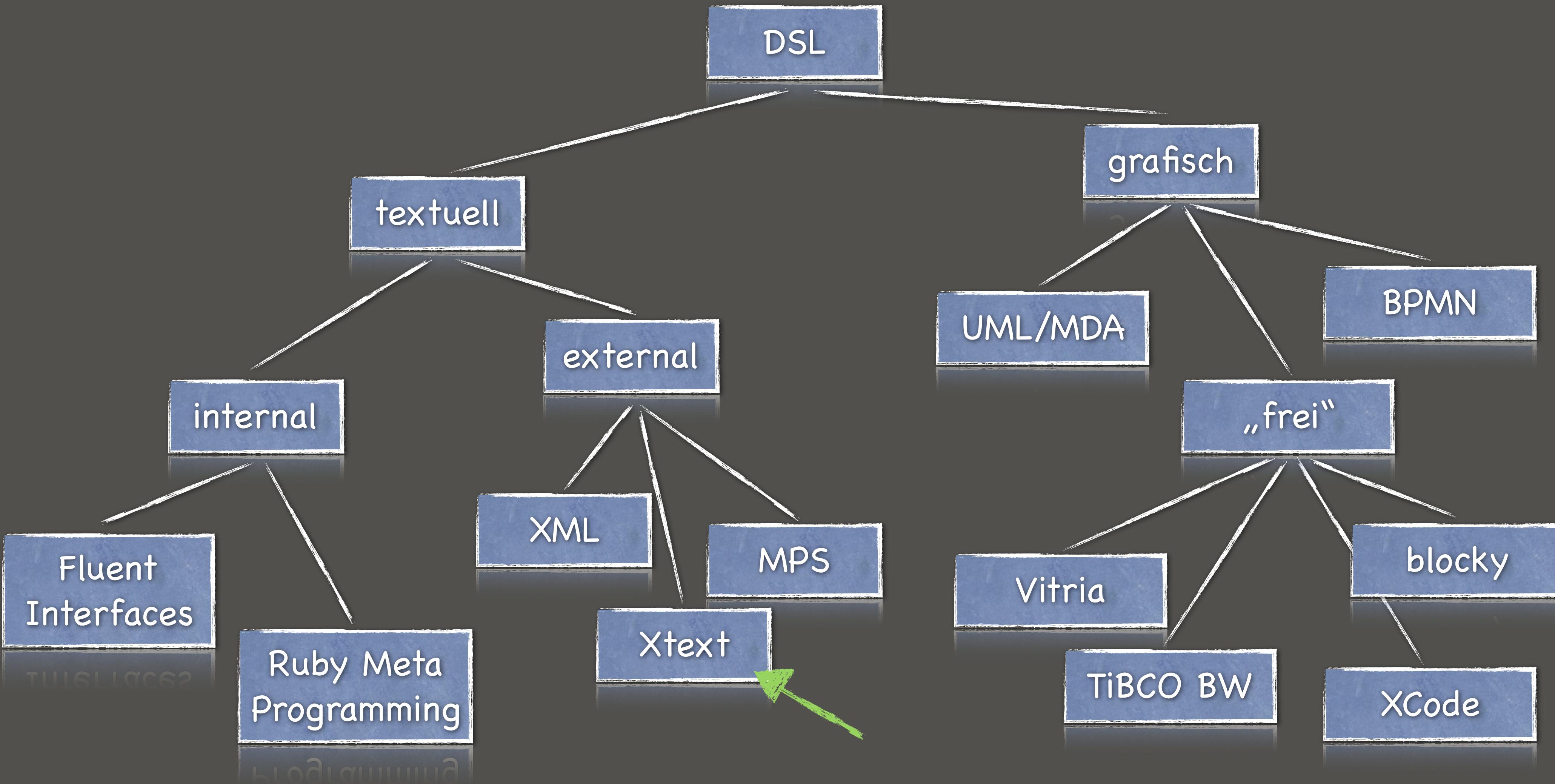
# DSLs haben viele Gesichter



# DSLs haben viele Gesichter



# DSLs haben viele Gesichter



Welche DSL  
und weshalb nun eigentlich?

Welche DSL  
und weshalb nun eigentlich?

do obey „the law of the instrument“!

Welche DSL  
und weshalb nun eigentlich?

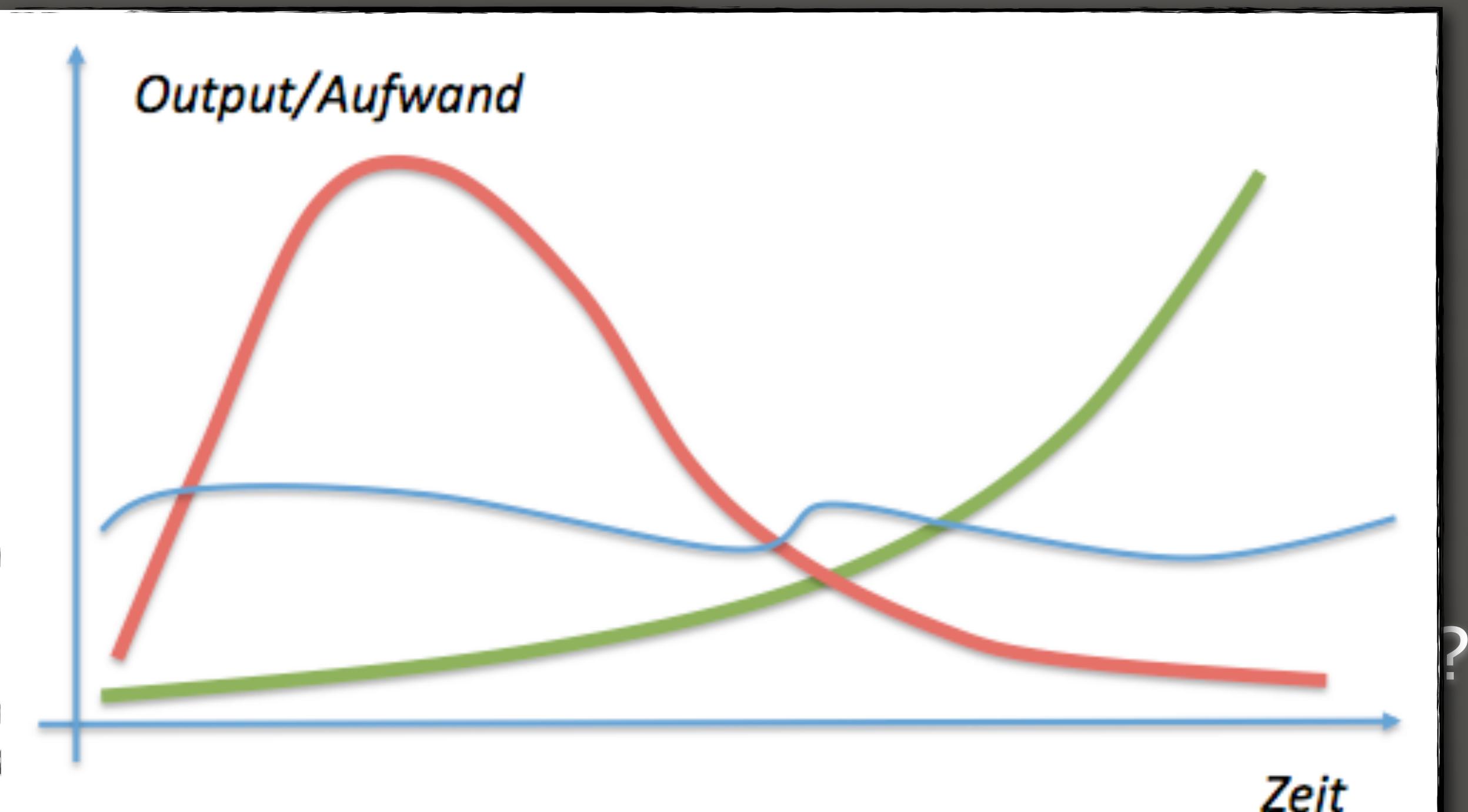
do obey „the law of the instrument“!

# „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?“

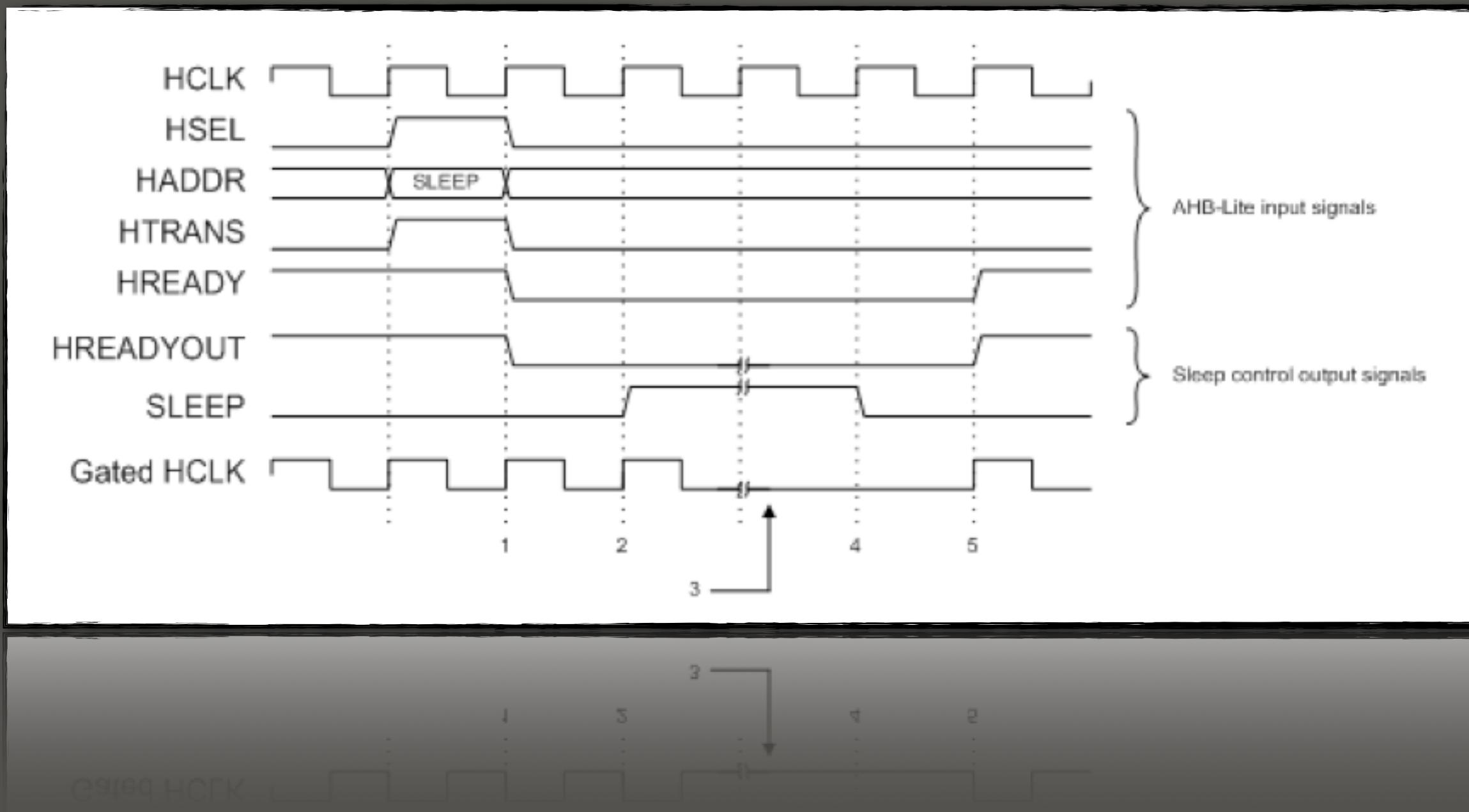
- ⌚ Lern- bzw. Effizienzproblem
- ⌚ Problem von hinteren Marktsegmenten
- ⌚ Spricht Domain-/Market-Manager?



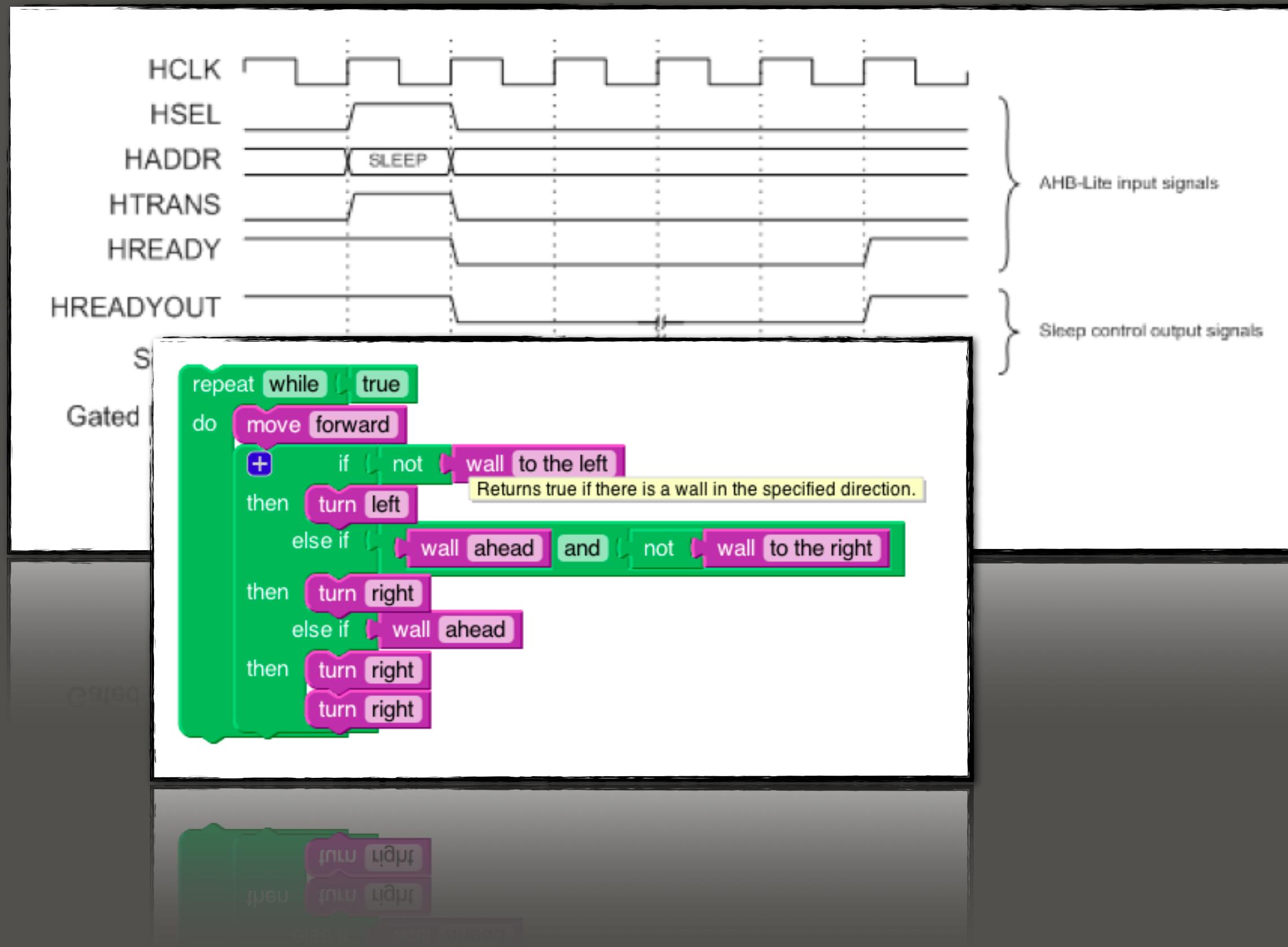
# Grafisch oder Textuell?

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

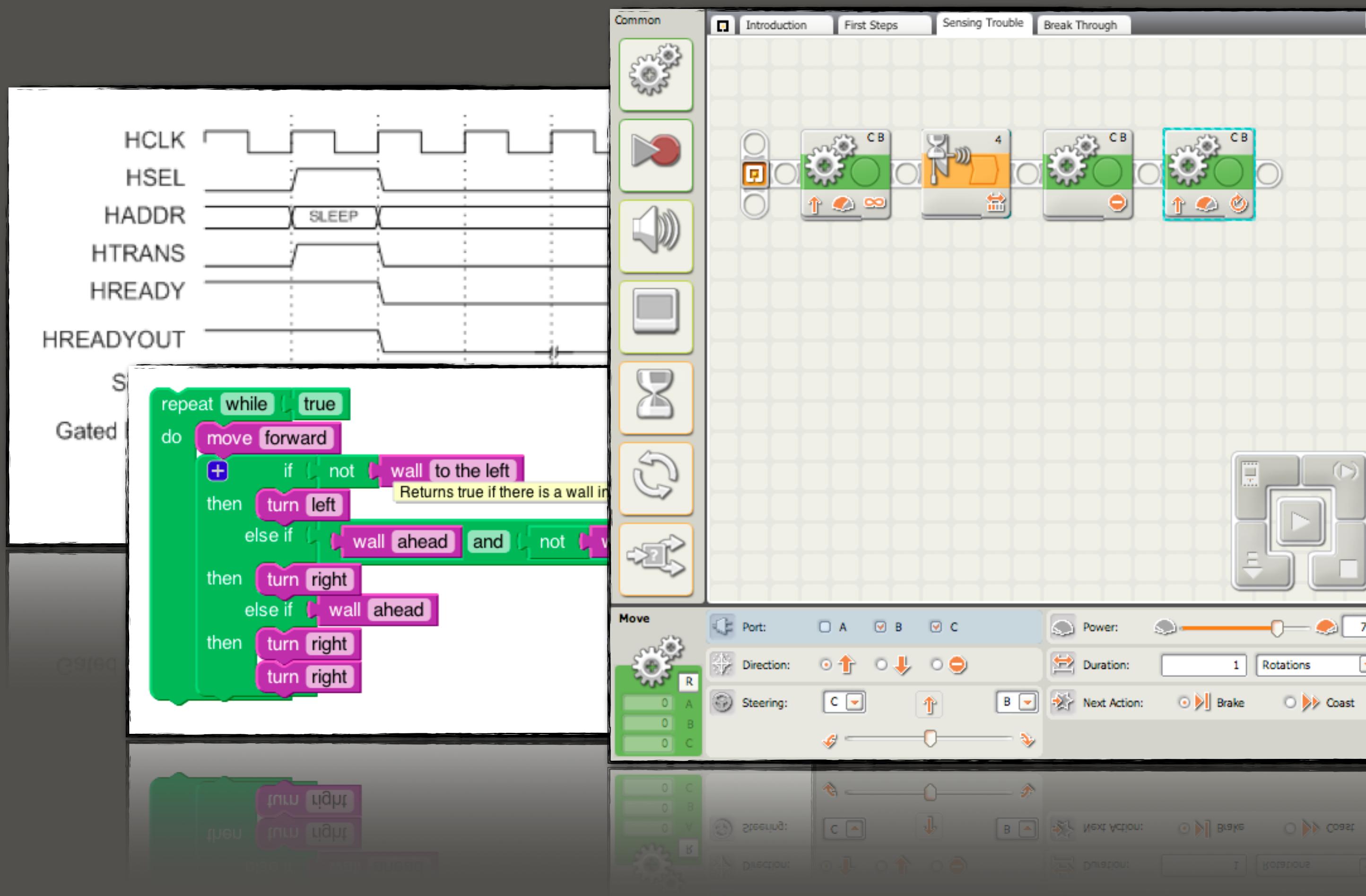
# Grafisch oder Textuell?



# Grafisch oder Textuell?



# Grafisch oder Textuell?



# Grafisch oder Textuell?

The image displays two distinct programming environments side-by-side, illustrating the trade-offs between graphical and textual representations.

**Left Side (Scratch-like Environment):**

- Top Panel:** Shows waveforms for system buses: HCLK, HSEL, HADDR (with a 'SLEEP' segment), HTRANS, HREADY, and HREADYOUT.
- Middle Panel:** A Scratch-style script editor titled "Gated". It contains a "repeat while [true]" loop with the following steps:
  - do [move [forward v] [100]]
  - + if [not [wall to the left v]] then [turn [left v] [90]]
  - else if [wall ahead v] and [not [wall to the right v]] then [turn [right v] [90]]
  - else if [wall ahead v] then [turn [right v] [90]]
  - turn [right v]
- Bottom Panel:** A "Move" block configuration window with settings for Port (B), Direction (Up), Steering (C), and a steering slider.

**Right Side (Business Logic Diagram):**

```
graph TD; SAPQ[FromSapBpTriggerQ] --> BPKey[BusinessPartnerKeyProcess]; BPKey --> SAPQ1[FromSapBpQ1]; BPKey --> SAPQ2[FromSapBpQ2]; BPKey --> SAPQ3[FromSapBpQ3]; BPKey --> SAPQ4[FromSapBpQ4]; BPProcess[BusinessPartnerProcess] --> SAPQ1; BPProcess --> SAPQ2; BPProcess --> SAPQ3; BPProcess --> SAPQ4; BPStatus[BusinessPartnerStatusProcess] -- CompletionCallback --> BPProcess;
```

This diagram illustrates a business process flow involving four parallel tasks (FromSapBpQ1 to FromSapBpQ4) triggered by a BusinessPartnerKeyProcess, which in turn is triggered by a BusinessPartnerProcess. A completion callback connects back to the BusinessPartnerProcess.

# Grafisch oder Textuell?

[Nichtamtliches Inhaltsverzeichnis](#)

**§ 102 Ergänzende Leistungen**

(1) Arbeitnehmerinnen und Arbeitnehmer haben Anspruch auf Wintergeld als Zuschuss-Wintergeld und Mehraufwands-Wintergeld und Arbeitgeber haben Anspruch auf Erstattung der von ihnen zu tragenden Beiträge zur Sozialversicherung, soweit für diese Zwecke Mittel durch eine Umlage aufgebracht werden.

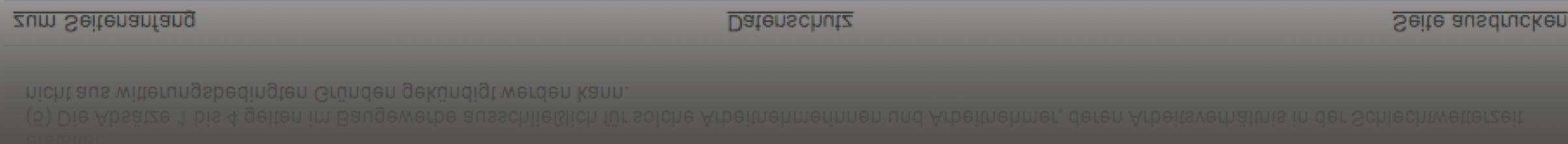
(2) Zuschuss-Wintergeld wird in Höhe von bis zu 2,50 Euro je ausgefallener Arbeitsstunde gezahlt, wenn zu deren Ausgleich Arbeitszeitguthaben aufgelöst und die Inanspruchnahme des Saison-Kurzarbeitergeldes vermieden wird.

(3) Mehraufwands-Wintergeld wird in Höhe von 1,00 Euro für jede in der Zeit vom 15. Dezember bis zum letzten Kalendertag des Monats Februar geleistete berücksichtigungsfähige Arbeitsstunde an Arbeitnehmerinnen und Arbeitnehmer gezahlt, die auf einem witterungsabhängigen Arbeitsplatz beschäftigt sind. Berücksigungsfähig sind im Dezember bis zu 90 Arbeitsstunden, im Januar und Februar jeweils bis zu 180 Arbeitsstunden.

(4) Die von den Arbeitgebern allein zu tragenden Beiträge zur Sozialversicherung für Bezieherinnen und Bezieher von Saison-Kurzarbeitergeld werden auf Antrag erstattet.

(5) Die Absätze 1 bis 4 gelten im Baugewerbe ausschließlich für solche Arbeitnehmerinnen und Arbeitnehmer, deren Arbeitsverhältnis in der Schlechtwetterzeit nicht aus witterungsbedingten Gründen gekündigt werden kann.

[zum Seitenanfang](#) [Datenschutz](#) [Seite ausdrucken](#)



# Grafisch oder Textuell?

The screenshot displays the DSL-Editor interface with two main panes illustrating different modeling approaches for a function.

**Left Pane (Graphical Representation):**

- Nichtamtliches Inhaltsverzeichnis:** A table of contents for "§ 102 Ergänzende Leistung".
- Text Content:**

(1) Arbeitnehmerinnen und Arbeitnehmer haben Anspruch auf Wintergeld als Zuschuss-Wintergeld. Anspruch auf Erstattung der von ihnen zu tragenden Beiträge zur Sozialversicherung, soweit für die (2) Zuschuss-Wintergeld wird in Höhe von bis zu 2,50 Euro je ausgefallener Arbeitsstunde gezahlt. Die Inanspruchnahme des Saison-Kurzarbeitergeldes vermieden wird.  
(3) Mehraufwands-Wintergeld wird in Höhe von 1,00 Euro für jede in der Zeit vom 15. Dezember bis zum 15. Februar berücksichtigungsfähige Arbeitsstunde an Arbeitnehmerinnen und Arbeitnehmer gezahlt, die auf Berücksichtigungsfähig sind im Dezember bis zu 90 Arbeitsstunden, im Januar und Februar jeweils 45 Arbeitsstunden.  
(4) Die von den Arbeitgebern allein zu tragenden Beiträge zur Sozialversicherung für Bezieherinnen erstattet.  
(5) Die Absätze 1 bis 4 gelten im Baugewerbe ausschließlich für solche Arbeitnehmerinnen und Arbeitnehmer, die nicht aus witterungsbedingten Gründen gekündigt werden kann.
- Links:** "zum Seitenanfang" and "Datenschutz".
- Bottom:** "Datenbeschreibung" section with placeholder text.

**Right Pane (Textual Representation):**

- Project Explorer:** Shows the project structure with nodes like "Funktion\_EGVd\_Eingliederu", "Funktion\_EGp\_Eingliederu", "Funktion\_IPL\_EintraegeDes", "Einschaltungerverwaltung", "Faehigkeitensprofilverwaltung", "Fallmanagement", "Lebenslaufeintraegeverwaltung", "Maßnahmenverwaltung", "Personenbetreuungsverwaltung", "Personenkurzuebersicht", "Personenverwaltung", "Profiling", "Rehaverwaltung", "Stellengesucheverwaltung", "VermittlungsUndLeistungsdaten", "Unterfunktion", "PRV\_13.01.00.0 Symbole [vamspezifikation]", and "PRV\_13.01.00.0 VERBIS [vamspezifikation/]."
- Code Editor:** Displays the textual representation of the function "Maske\_ANZb\_AnrechnungszeitBearbeiten". The code includes sections for "maske", "abschnitt", and "daten". It defines fields like "titel", "beschreibung", and "status" based on conditions such as "#IstAnrechnungszeitAV = ja".
- Output View:** Shows the generated HTML file "Maske\_ANZb\_AnrechnungszeitBearbeiten.html" containing a form for "Anrechnungszeit Berufsberatung" with fields for "Beginn MAZ-Zeitraum", "Ende MAZ-Zeitraum", and "Meldungsgrund".

# 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?

# Modell- & Templatedefinition

- Top down, nicht auf der grünen Wiese
  - mit dem Fachbereich DSL anhand tatsächlicher Problemstellungen erarbeiten
  - Templates entwickeln, nachdem hinreichend viele Use-Cases prototypisch umgesetzt wurden und daraus generische Codeanteile extrahiert werden können

# nicht zu kurz denken ...

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

# <CODE>

# Umgebung & Technologie

```
1. grammar org.xtext.example.mydsl with
2.           org.eclipse.xtext.common.Terminals
3.
4. generate myDsl "http://www.xtext.org/example/mydsl/MyDsl"
5.
6. Model:
7.   greetings+=Greeting*;
8.
9. Greeting:
10.  'Hello' name=ID '!';
```

```
7.  val movies = new FileReader('data.csv').readLines.map [ line |
8.    val segments = line.split(' ').iterator
9.    return new Movie(
10.      segments.next,
11.      Integer::parseInt(segments.next),
12.      Double::parseDouble(segments.next),
13.      Long::parseLong(segments.next),
14.      segments.toSet
15.    )
16.  ]
```

# Umgebung & Technologie



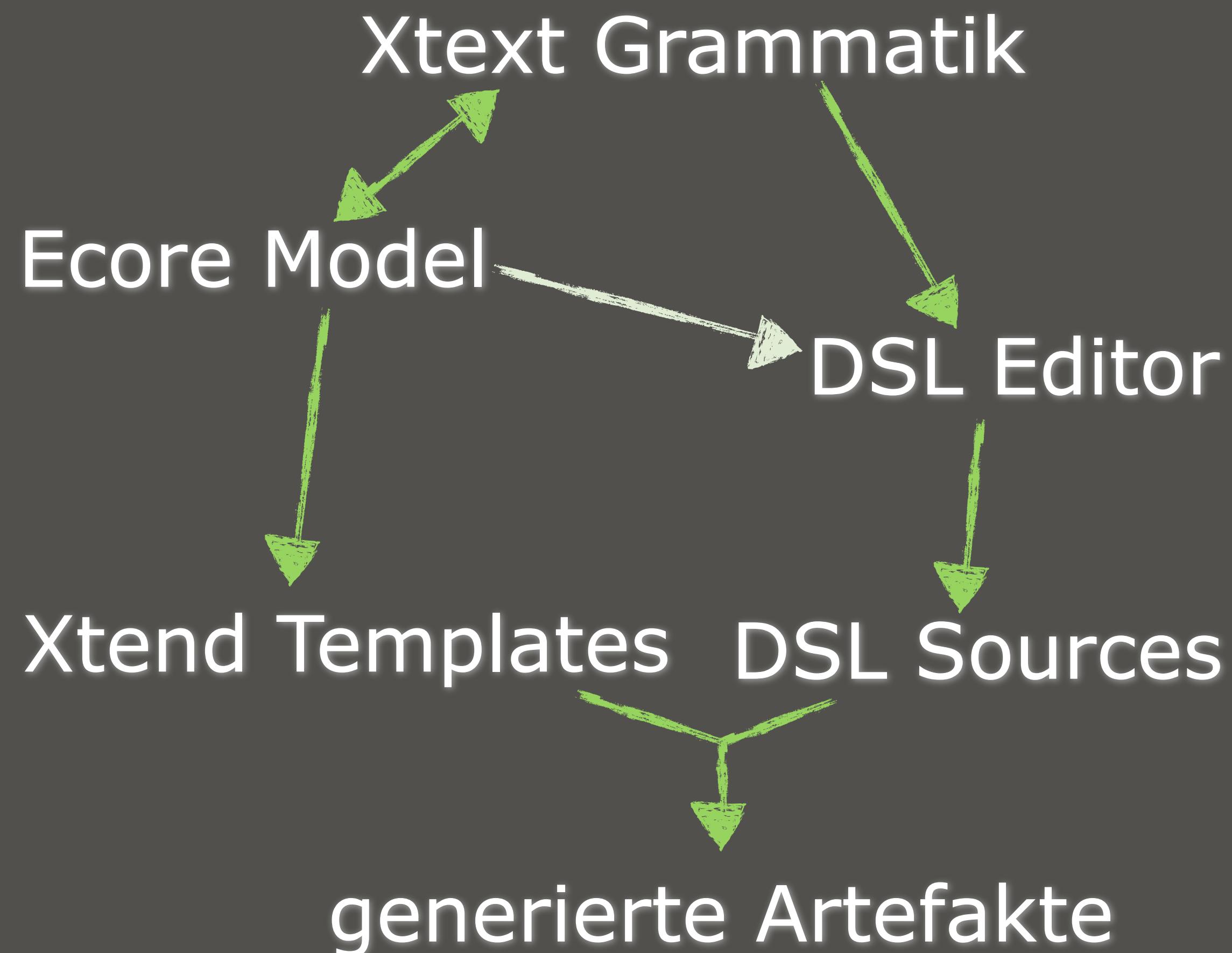
# Umgebung & Technologie

- 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
- Service/Logic/Entity

# 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

- Assignments

- Referenzen

# Xtext Basics

## ⌚ Rules

### ⌚ Terminal

### ⌚ DataType

### ⌚ Parser/Production/EObject

### ⌚ Keywords

### ⌚ Assignments

### ⌚ Referenzen

Model:

```
greetings+=Greeting*;
```

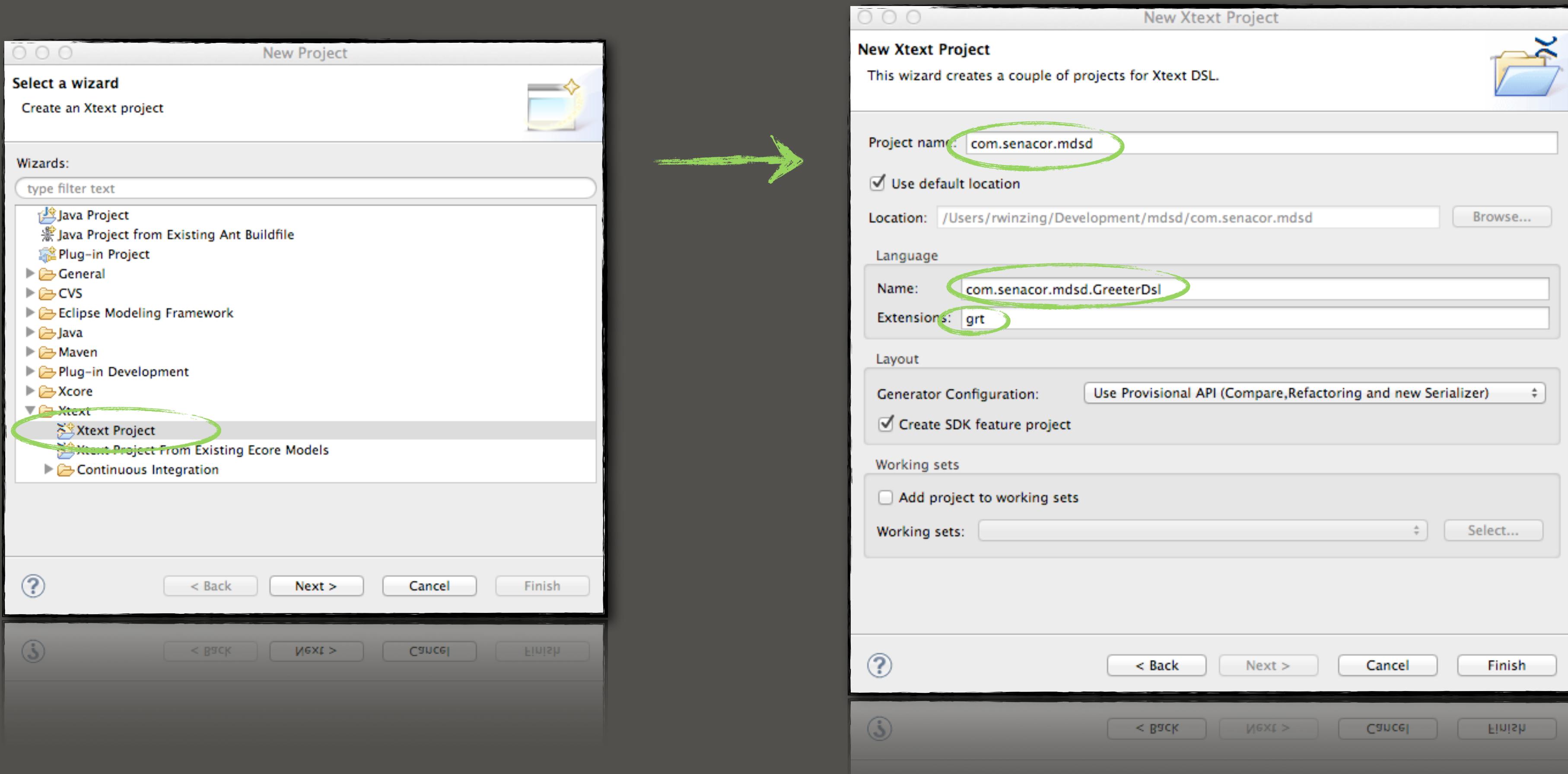
Greeting:

```
'Hello' name=ID '!';
```

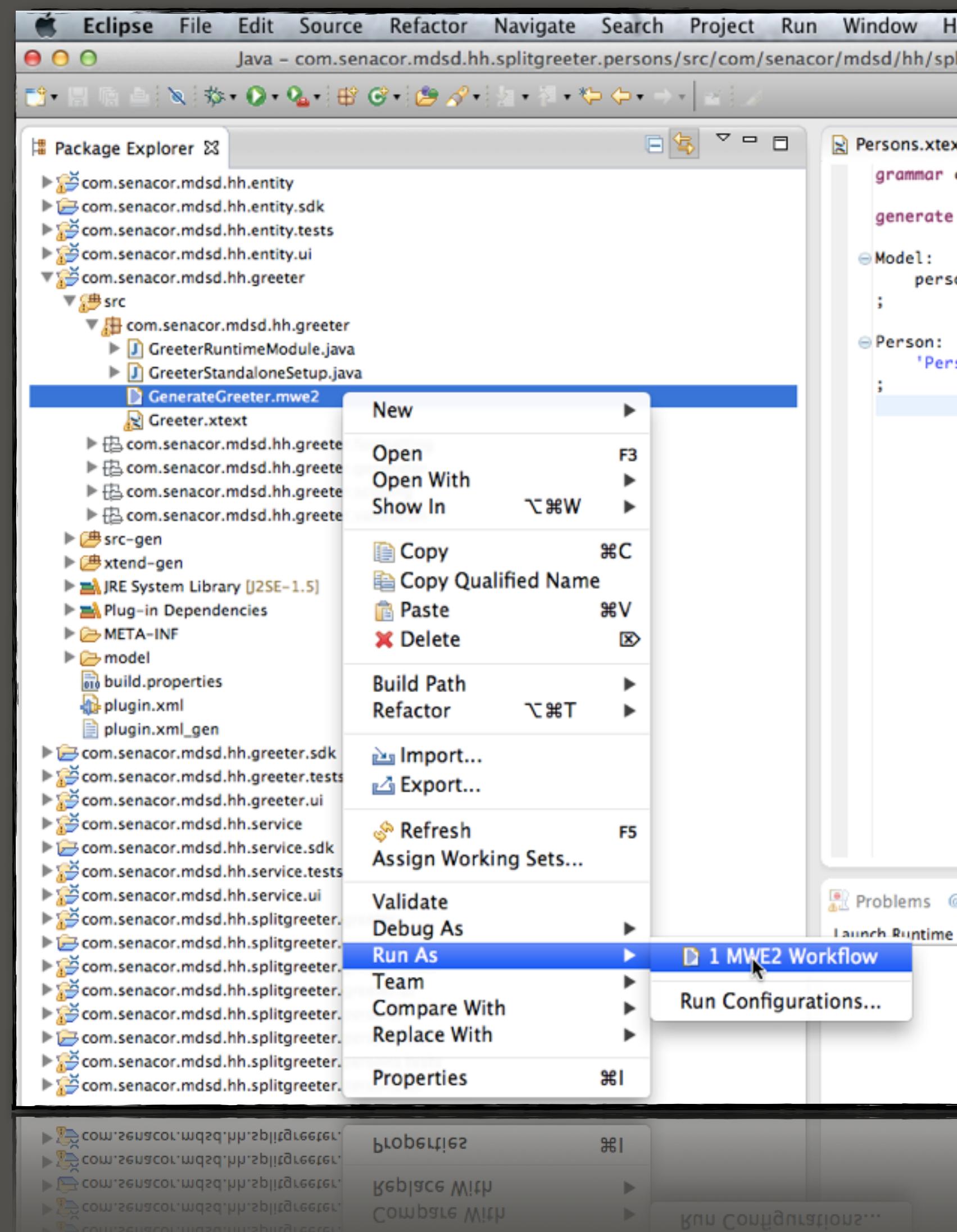
# Hello, World! (1)

- 🕒 Projekt erzeugen
- 🕒 Editor testen
- 🕒 generierte Artefakte ansehen

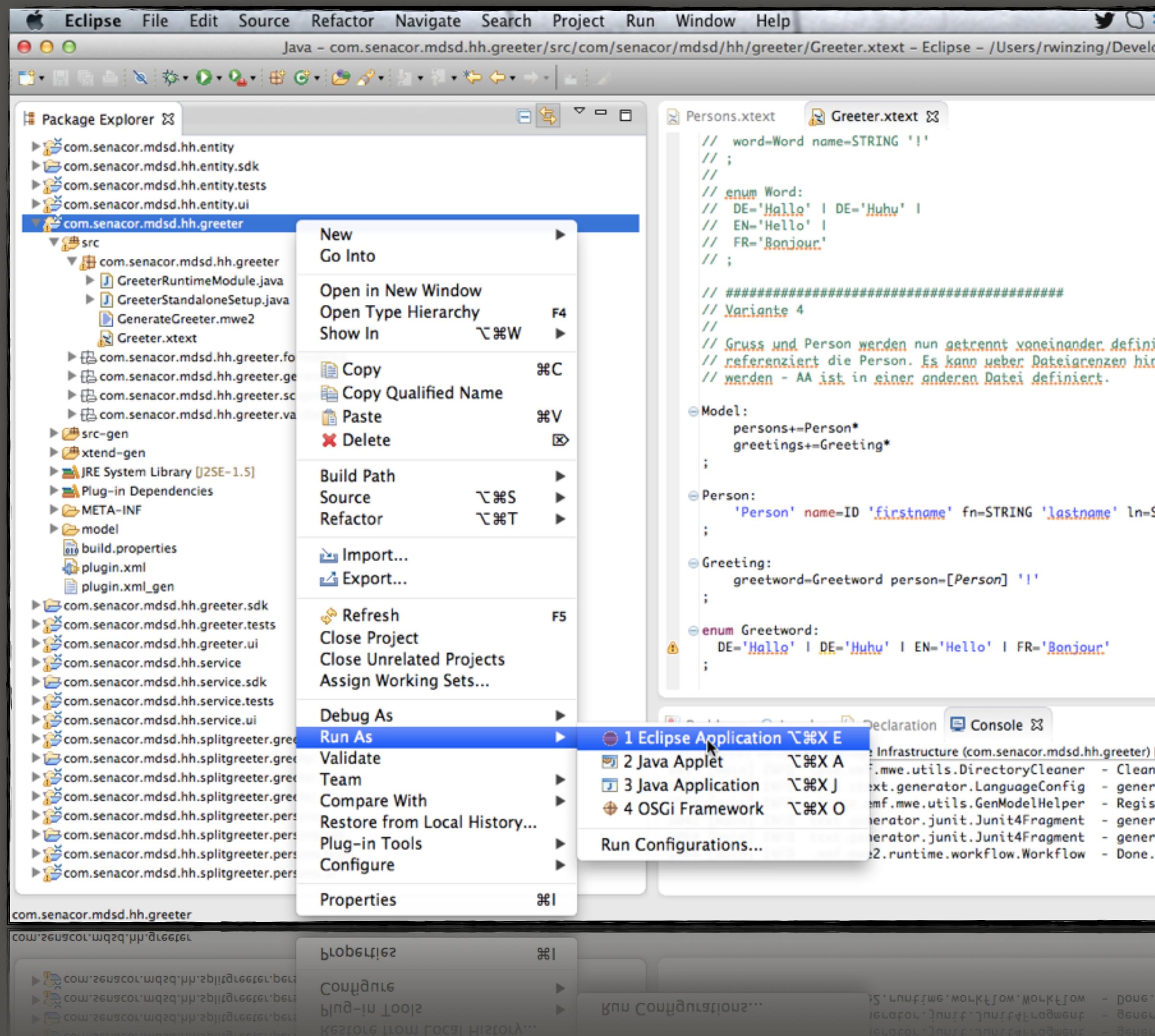
# Ein erstes Xtext Projekt ...



# ... Artefakte generieren ...



# ... und Editor starten.



# Hello, World! (2)

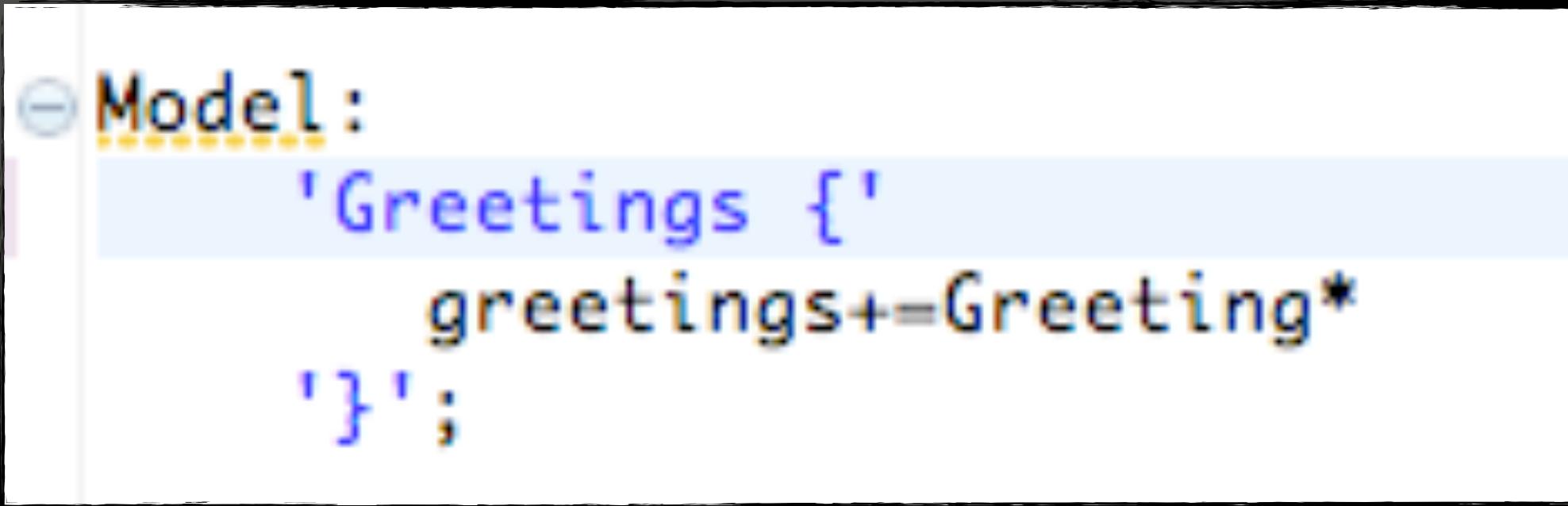
- etwas Struktur durch Klammern
- Name soll als String erfasst werden



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

# Hello, World! (2)

Wie wäre es mit



The screenshot shows a code editor with a Java file open. The file contains a class named 'Model' with a single method. The code is as follows:

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

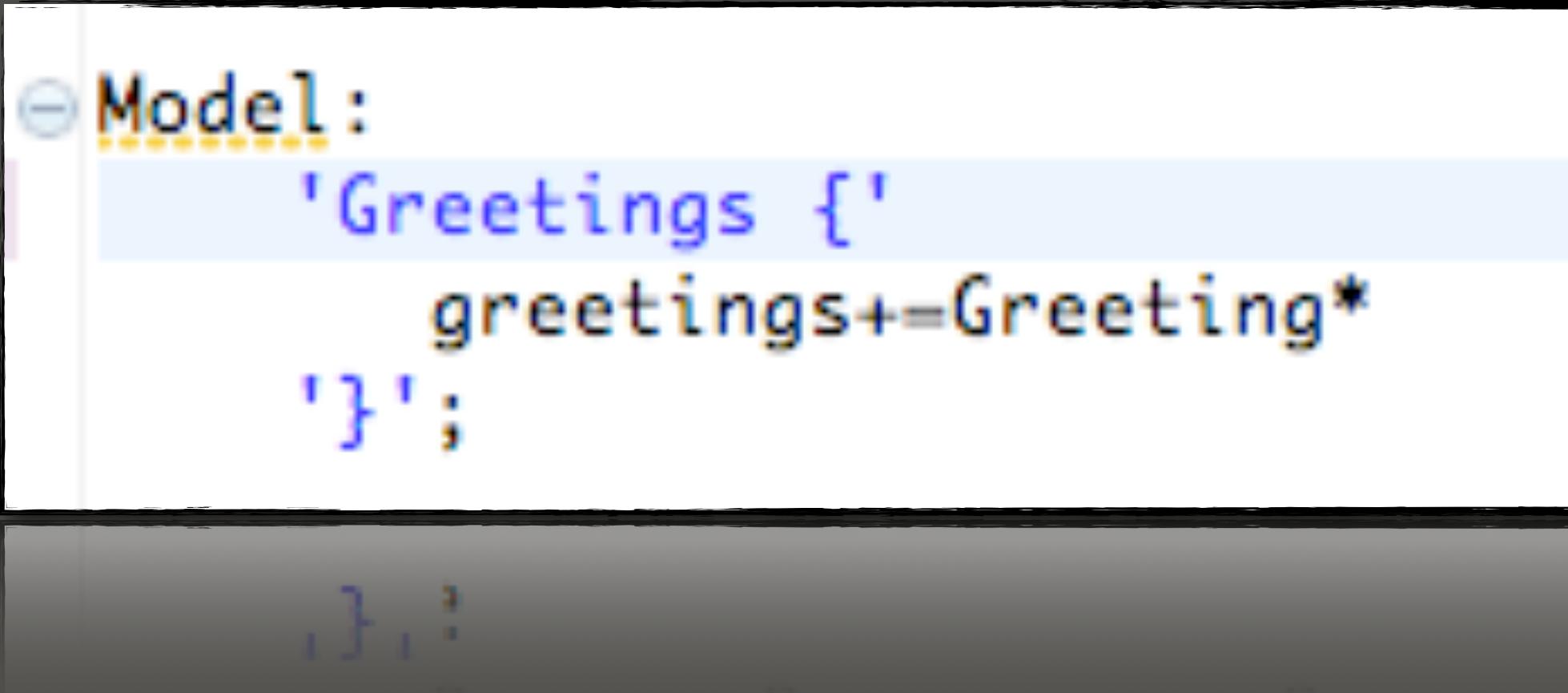
The code editor has a dark theme with syntax highlighting. The word 'Model' is in blue, 'Greetings' is in green, and 'greetings' is in red. The code is displayed in a light-colored box.

At the bottom of the slide, there is a small portion of another line of code, showing a closing brace '}' followed by a question mark '?'.

?

# Hello, World! (2)

Wie wäre es mit



The screenshot shows a code editor window with a dark theme. A tooltip or callout box is overlaid on the code area, highlighting a section of the code. The code is as follows:

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

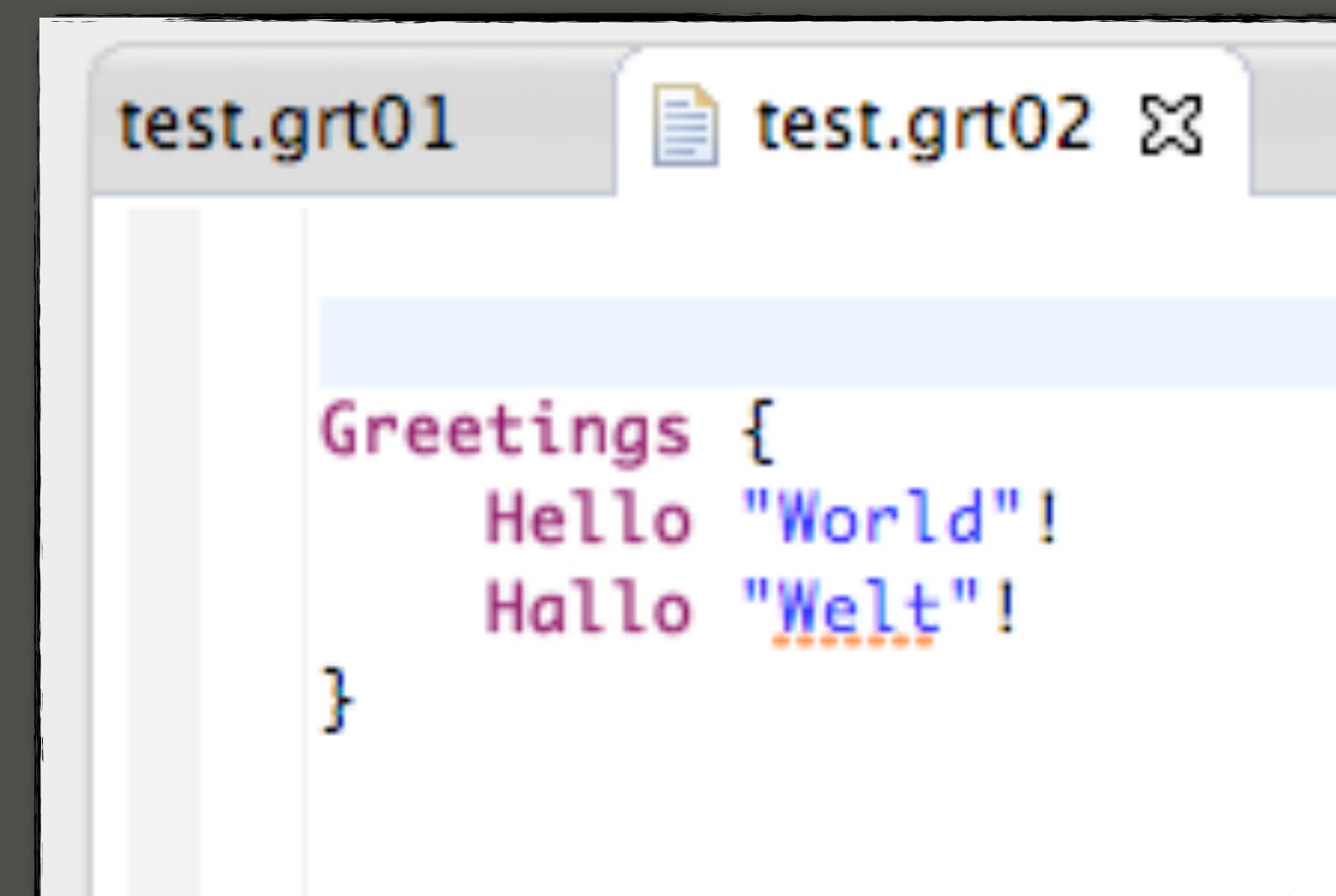
The word "Model" is underlined with a blue dotted line, indicating it is a variable or field name. The code uses inconsistent indentation: the opening brace of the block is aligned with the start of the block, while the closing brace is aligned with the start of the "greetings" line.

?

**SPACES ARE EVIL!**

# Hello, World! (3)

- ⌚ Alternative Grußtexte
- ⌚ Eine zusätzliche Rule



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

# Hello, World! (4)

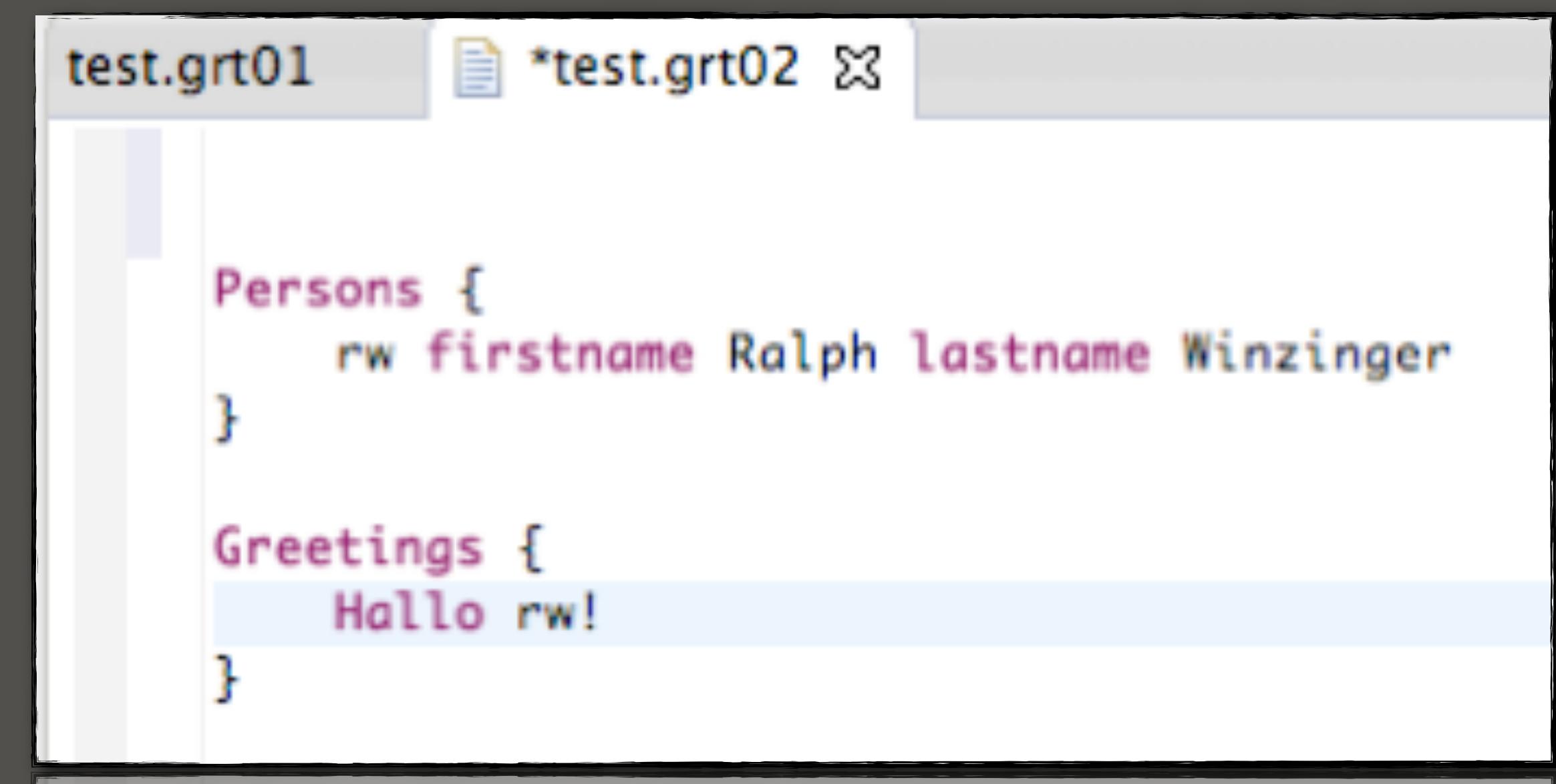
- ⌚ Grußworte als Enum
- ⌚ Kommentare



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

# Hello, World! (5)

- ☛ Personen & Grüße
- ☛ Cross-References



The screenshot shows a text editor interface with two tabs at the top: "test.grt01" and "\*test.grt02". The content of "test.grt01" is as follows:

```
Persons {
    rw firstname Ralph lastname Winzinger
}

Greetings {
    Hallo rw!
}
```

}

HOTTO LMJ

# 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:

# Hello, World! (6)

- Mixins - Grammatiken importieren
- Cross-References über Datei- und Grammatikgrenzen hinweg
- Grammatik in.xtext-Datei importieren
- Manifest anpassen - Plugin-Dependency einführen

# Hello, World! (6)

```
grammar com.senacor.mdsd.hh.splitgreeter.greeter with org.eclipse.xtext.common.Terminals

import "http://www.senacor.com/mdsd/hh/splitgreeter/persons/Persons" as Persons

generate greeter "http://www.senacor.com/mdsd/hh/splitgreeter/greeter/Greeter"

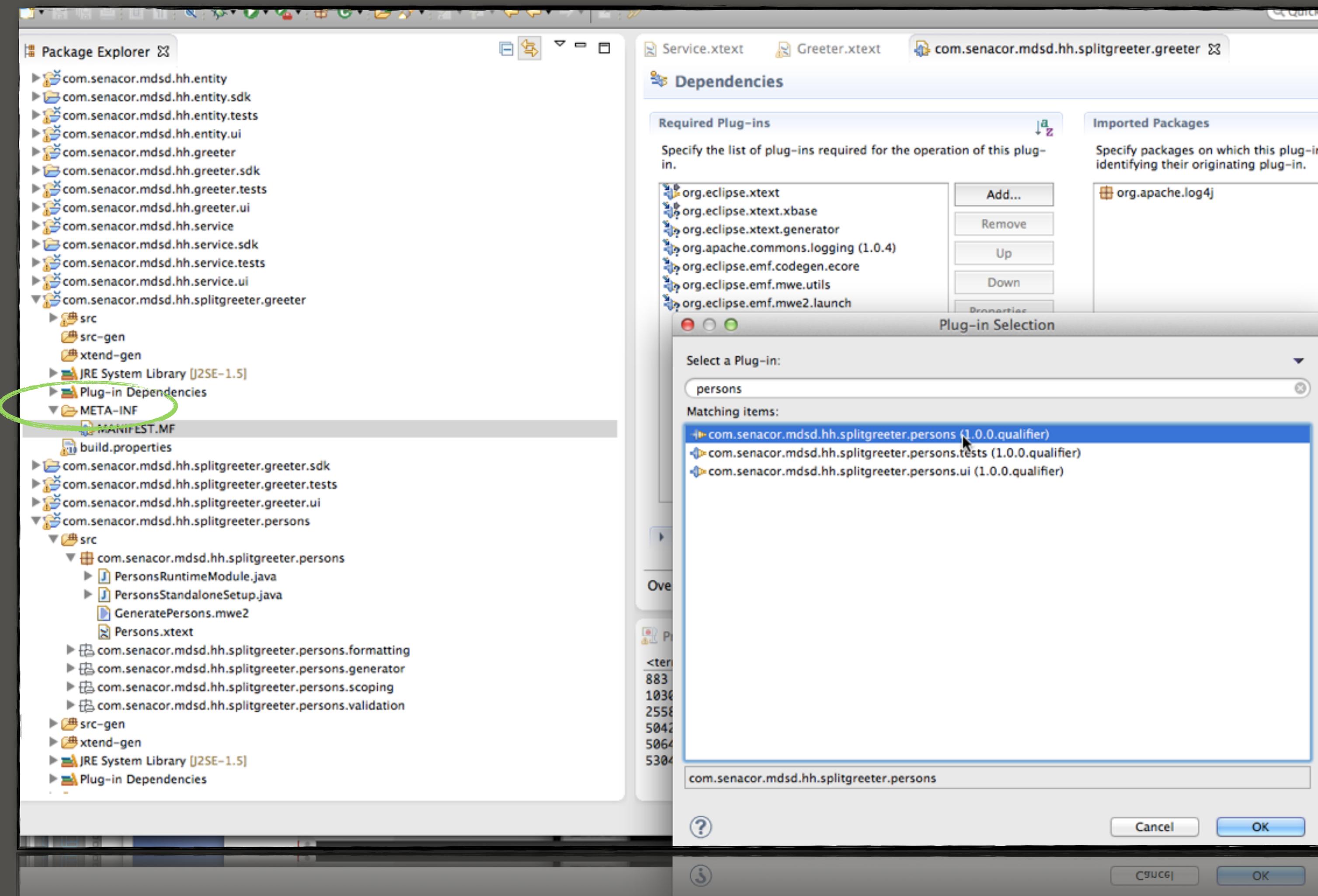
@Model:
    greetings+=Greeting*
;

@Greeting:
    greetword=Greetword person=[Persons::Person] !!
;

@enum Greetword:
    DE='Hallo' | DE='Huhu' | EN='Hello' | FR='Bonjour'
;
;
```

The screenshot shows an Eclipse IDE interface with several tabs at the top: Service.xtext, Greeter.xtext, com.senacor.mdsd.hh.splitgreeter.greeter, and Persons.xtext. The Greeter.xtext tab is active and displays an Xtext grammar. Two specific parts of the code are highlighted with green ovals: the import statement 'import "http://www.senacor.com/mdsd/hh/splitgreeter/persons/Persons" as Persons' and the 'person=[Persons::Person]' part of the Greeting rule's 'greetword' field.

# Hello, World! (6)



# Hello, World! (6)

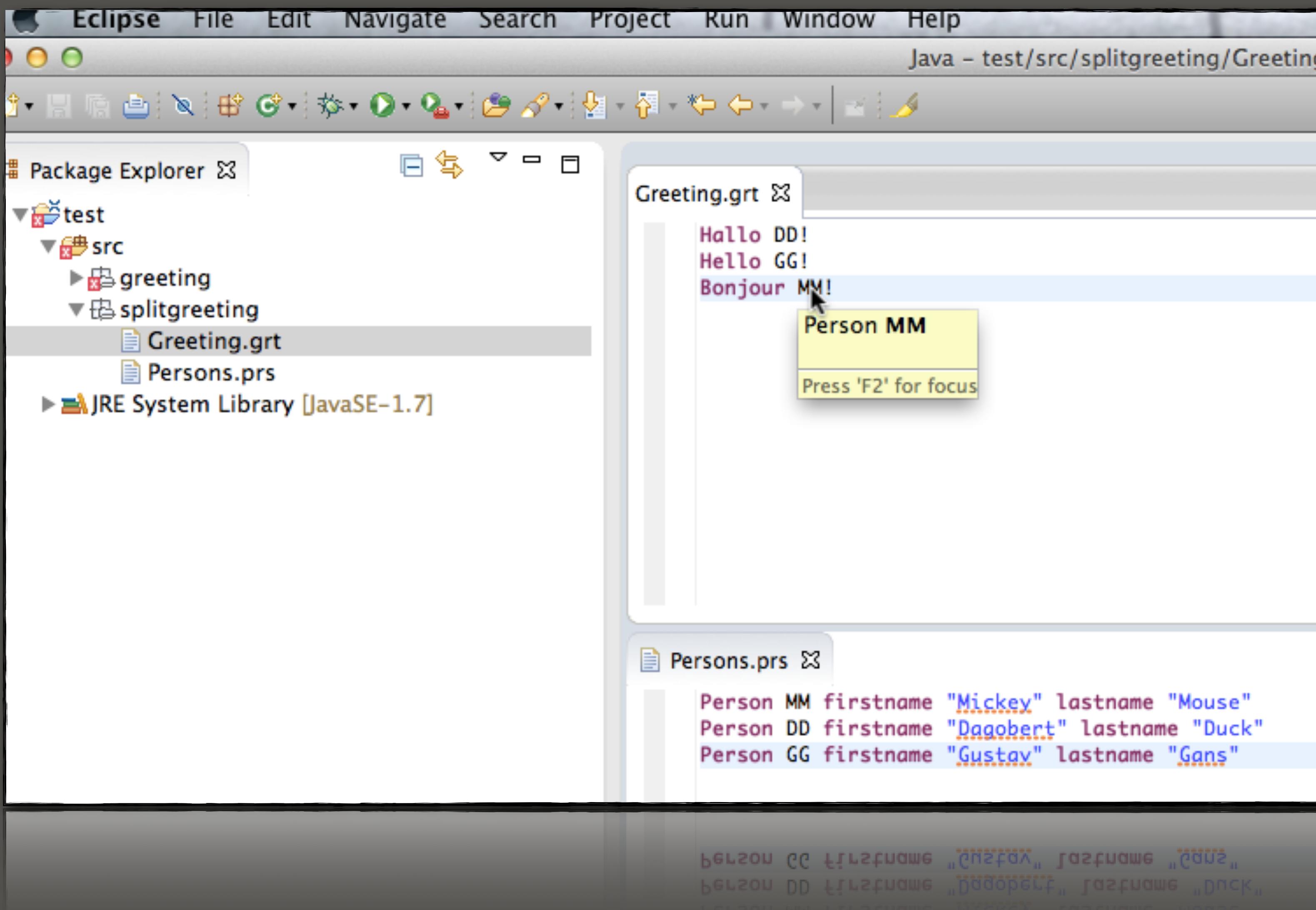
```
Workflow {
    bean = StandaloneSetup {
        scanClassPath = true
        platformUri = "${runtimeProject}..."
        // The following two lines can be removed, if Xbase is not used.
        // registerGeneratedEPackage = "org.eclipse.xtext.xbase.XbasePackage"
        // registerGenModelFile = "platform:/resource/org.eclipse.xtext.xbase/model/Xbase.genmodel"

        // register splitgreeter persons
        registerGeneratedEPackage = "com.senacor.mdsd.hh.splitgreeter.persons.persons.PersonsPackage"
        registerGenModelFile = "platform:/resource/com.senacor.mdsd.hh.splitgreeter.persons/model/generated/Persons.genmodel"
    }
}
```

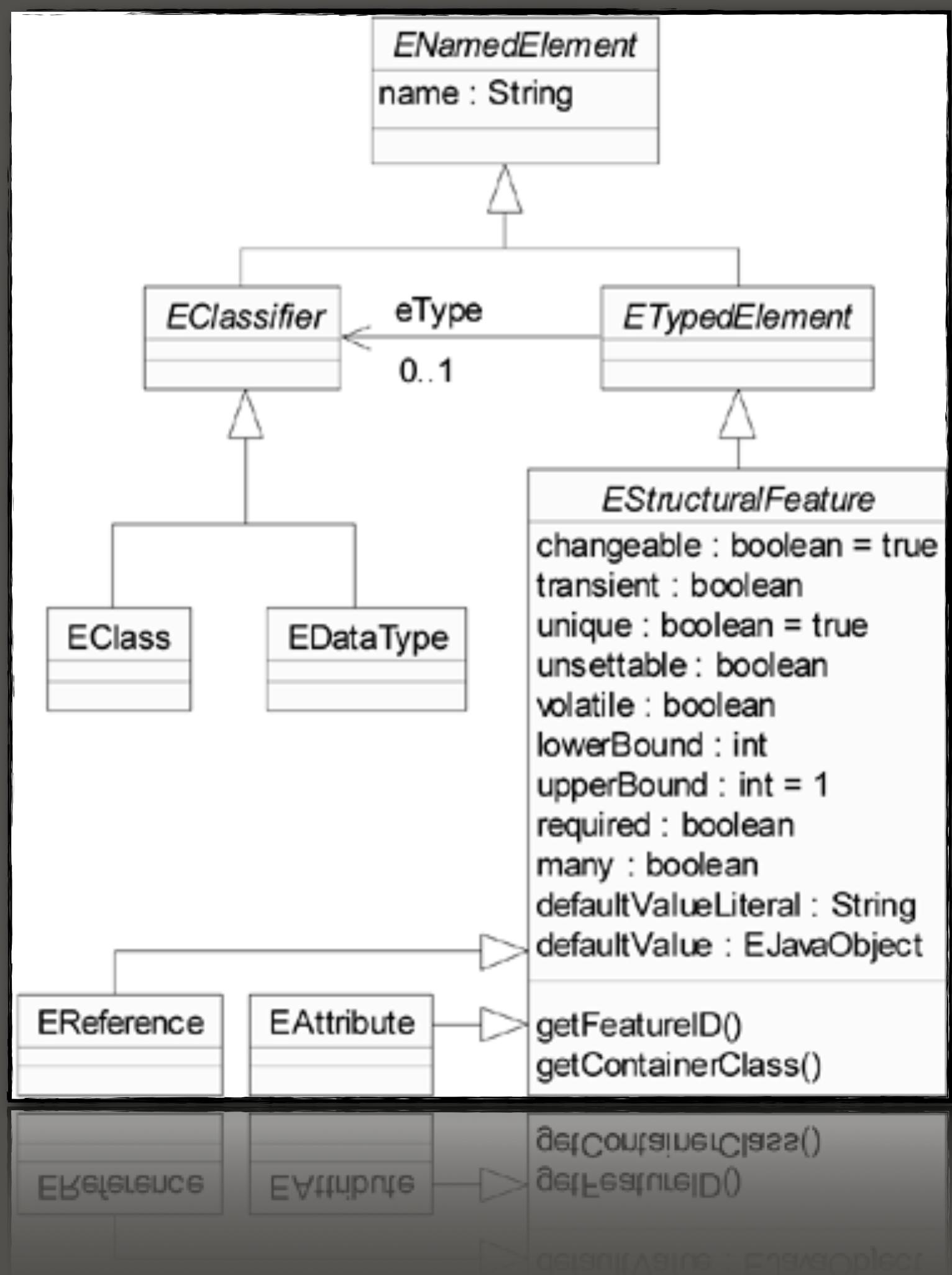
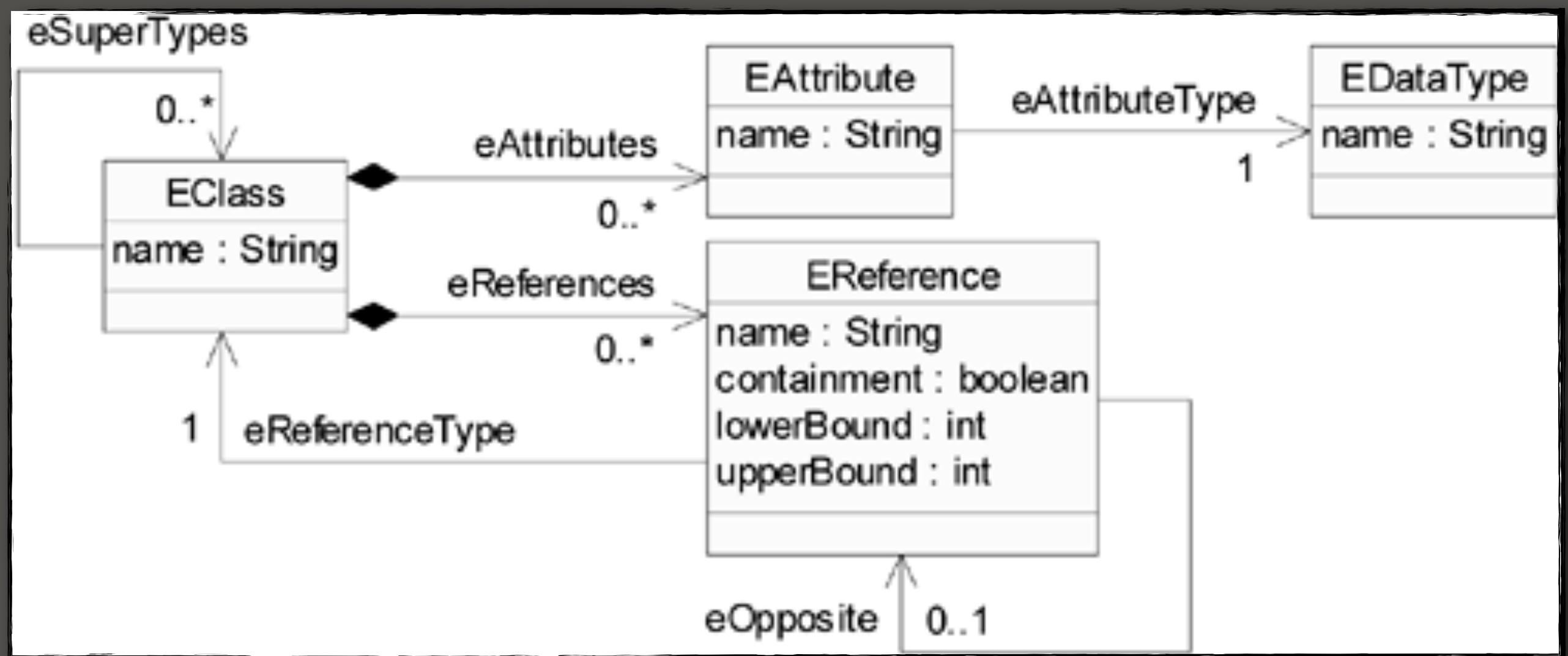
```
Workflow {
    bean = StandaloneSetup {
        scanClassPath = true
        platformUri = "${runtimeProject}..."
        // The following two lines can be removed, if Xbase is not used.
        // registerGeneratedEPackage = "org.eclipse.xtext.xbase.XbasePackage"
        // registerGenModelFile = "platform:/resource/org.eclipse.xtext.xbase/model/Xbase.genmodel"

        // register splitgreeter persons
        registerGeneratedEPackage = "com.senacor.mdsd.hh.splitgreeter.persons.persons.PersonsPackage"
        registerGenModelFile = "platform:/resource/com.senacor.mdsd.hh.splitgreeter.persons/model/generated/Persons.genmodel"
    }
}
```

# Hello, World! (6)



# ecore Model



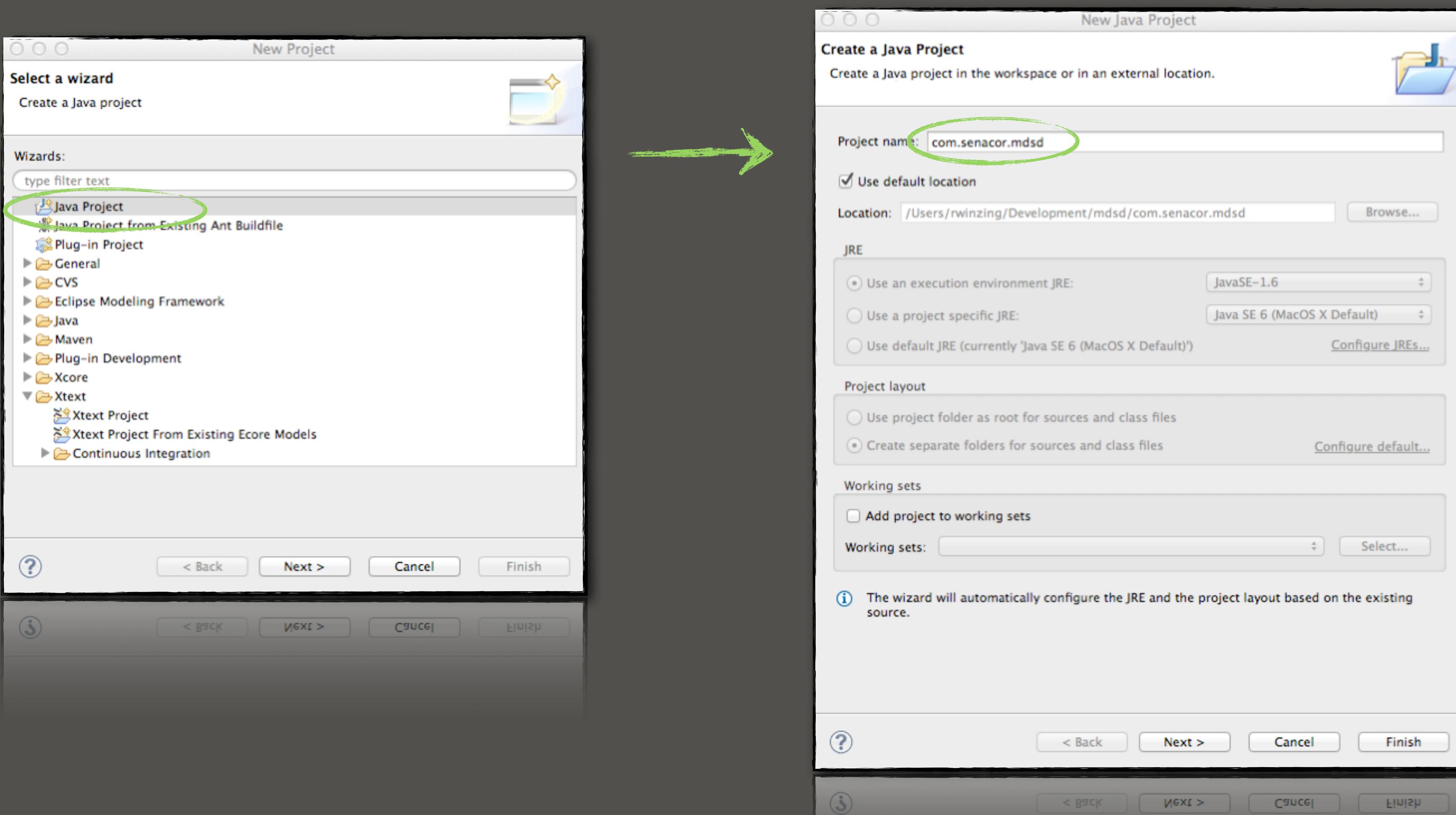
# Xtend Basics

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

# Hello, World! (wieder mal)

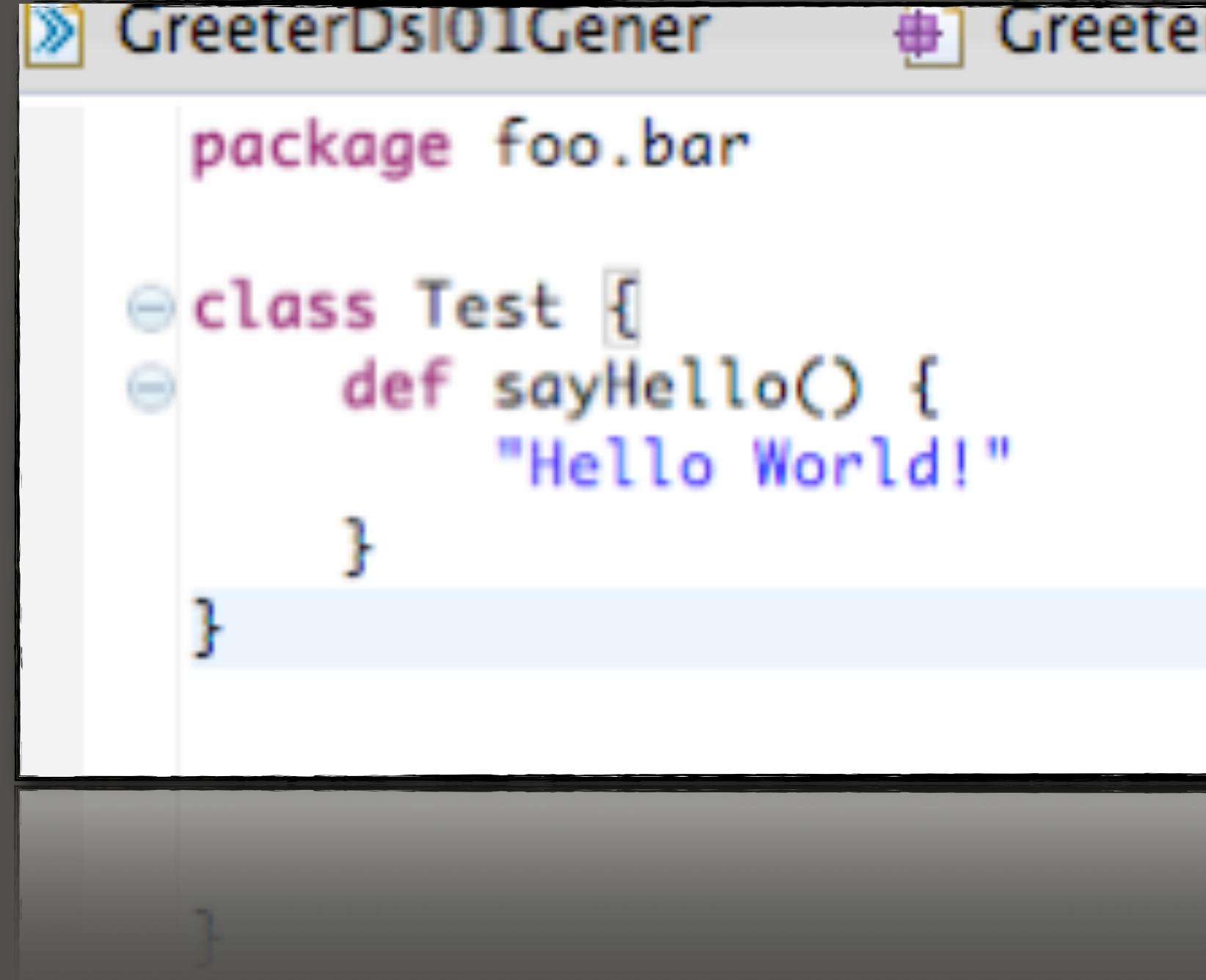
- ⌚ Projekt erzeugen
- ⌚ Xtend Klasse anlegen & speichern
- ⌚ generierte Artefakte ansehen

# Ein erstes Xtend Projekt ...



# def, var & val

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



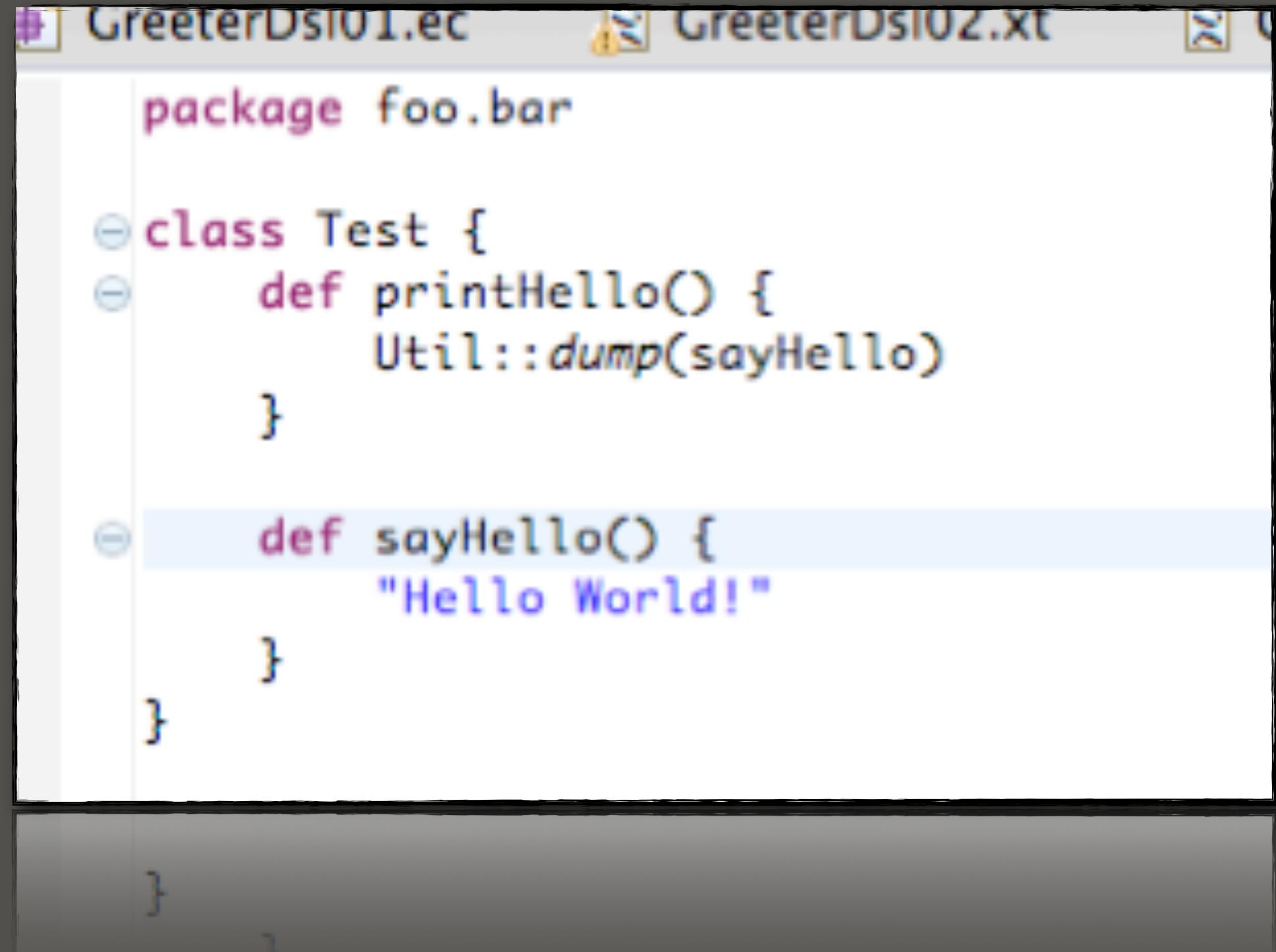
```
GreeterDSL01Generator GreeterDSL01Generator

package foo.bar

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

# Transparenter Zugriff zwischen Java & Xtend

„::“ für Zugriff auf statische Elemente



The screenshot shows an IDE interface with two tabs at the top: "GreeterDSI01.ec" and "GreeterDSI02.xt". The "GreeterDSI02.xt" tab is active, displaying Xtend code. The code defines a class "Test" with a method "printHello()". Inside "printHello()", there is a call to "Util:::dump(sayHello)". Below it, there is another method "sayHello()" which returns the string "Hello World!". The code is color-coded, with "Util" being blue and "::" being orange.

```
package foo.bar

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

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

# Weshalb eigentlich „Xtend“?



# Weshalb eigentlich „Xtend“?

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



# Weshalb eigentlich „Xtend“?

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

```
def dumpToConsole(String s) {  
    System.out.println(s);  
}
```

# Weshalb eigentlich „Xtend“?

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

```
def dumpToConsole(String s) {  
    System.out.println(s);  
}
```

```
dumpToConsole(„hello“);
```

# Weshalb eigentlich „Xtend“?

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

```
def dumpToConsole(String s) {  
    System.out.println(s);  
}
```

```
dumpToConsole(„hello“);
```

```
„hello“.dumpToConsole();
```

# Weshalb eigentlich „Xtend“?

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

```
def dumpToConsole(String s) {  
    System.out.println(s);  
}
```

```
dumpToConsole(„hello“);
```

```
„he  
    for (gw: resource.allContents.toIterable.filter(typeof(GW))) {  
        generateFile(gw, fsa);  
    }  
}
```

# Templates

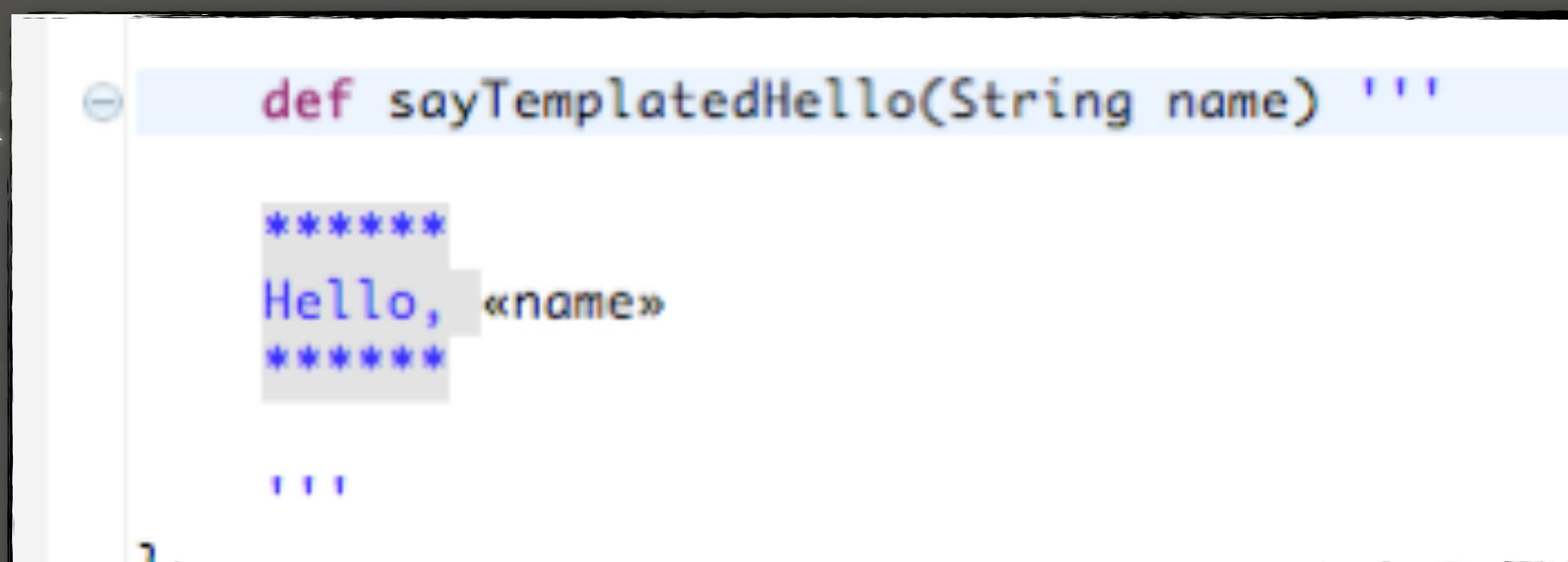
- Triple-Quotes leiten ein Template ein

- String-Ausdruck

- IF / ENDIF

- FOR / ENDFOR

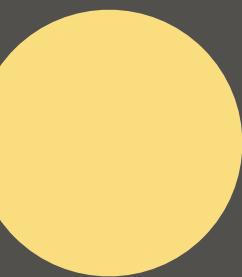
- BEFORE / SEPARATOR / AFTER



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

```
def sayTemplatedHello(String name) ***  
    ****  
    Hello, «name»  
    ****  
    ...  
1
```

The string "Hello, «name»" is highlighted in blue, indicating it is a placeholder or variable in the template.



# editor tweaks

- Validatoren

- Labels, Icons

# putting stuff together ...

- xtext und xtend verbinden
- und ein wenig generieren

# Microservices & DDD

domain

bounded  
context

aggregat

# Microservices & DDD

domain

bounded  
context

aggregat

# Generation Gap

- Generierter und manuell implementierter Code müssen coexistieren können
- „Protected Regions“ reserviert Bereiche im generierten Code - leider fehleranfällig
- „Generation Gap“ verwendet OO-Mechanismen. Manueller Code wird in Subklassen verlagert, generiert werden nur (abstrakte) Oberklassen

<CODE>

```
next() {
    NSMutableDictionary *fields = [NSMutableDictionary dictionary];
    * savingsTargetID = [NSNumber numberWithInt:[row[0] intValue]];
    setObject:savingsTargetID forKey:@"savingsTargetID";
    * categoryID = [NSNumber numberWithInt:[row[1] intValue]];
    setObject:categoryID forKey:@"categoryID";
    * parentGoalID = [NSNumber numberWithInt:[row[2] intValue]];
    setObject:parentGoalID forKey:@"parentGoalID";
    * name = [results objectForKey:@"name"];
    setObject:name forKey:@"name";
    * color = [results objectForKey:@"color"];
    setObject:color forKey:@"color";
    * saveAmount = [NSNumber numberWithInt:[row[5] intValue]];
    setObject:saveAmount forKey:@"saveAmount";
}
```

# Best Practices

- Generation Gap (statt Protected Region)
- Vollautomatisierung über Maven (CI!)
- sehr (wirklich sehr) viel Zeit auf Design verwenden
- Integration mit Methodik planen
  - keinen unveränderlichen Monolithen bauen
- lieber unabhängige Teile zu 100% als alles zu 70%
- nicht zuviel Logik in xtend
- Tests!