

MDSW Workshop Universität Hamburg

Ralph Winzinger, Senacor

Agenda 09.01.2013

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

Agenda | 6.01.2013

- 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

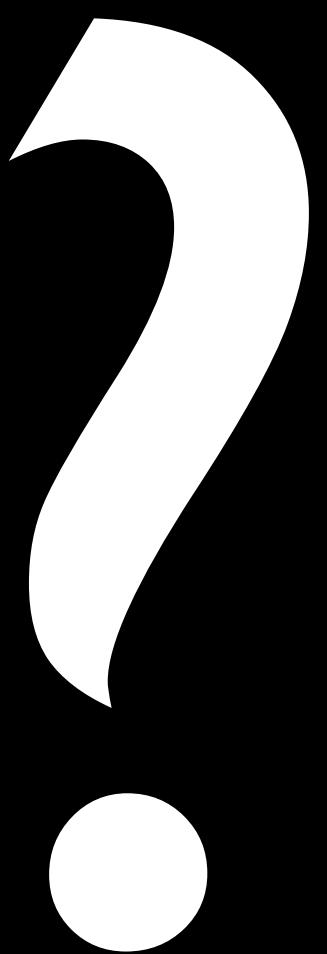
Branchenfokus

- Banking

Ausbildung und berufliche Erfahrung

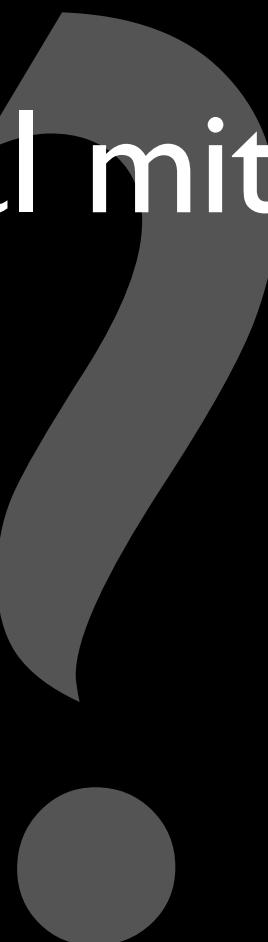
- Über 10 Jahre Erfahrung in der Anwendungsentwicklung, technischen Projektleitung und Architekturberatung im Bankenumfeld
- Architekt, Senacor Technologies AG (seit 2000)
- Freiberufliche Tätigkeit im Bereich Mobile Computing (1996 – 2002)
- Diplom in Informatik, Univ. Erlangen-Nürnberg

Warum DSLs?



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>

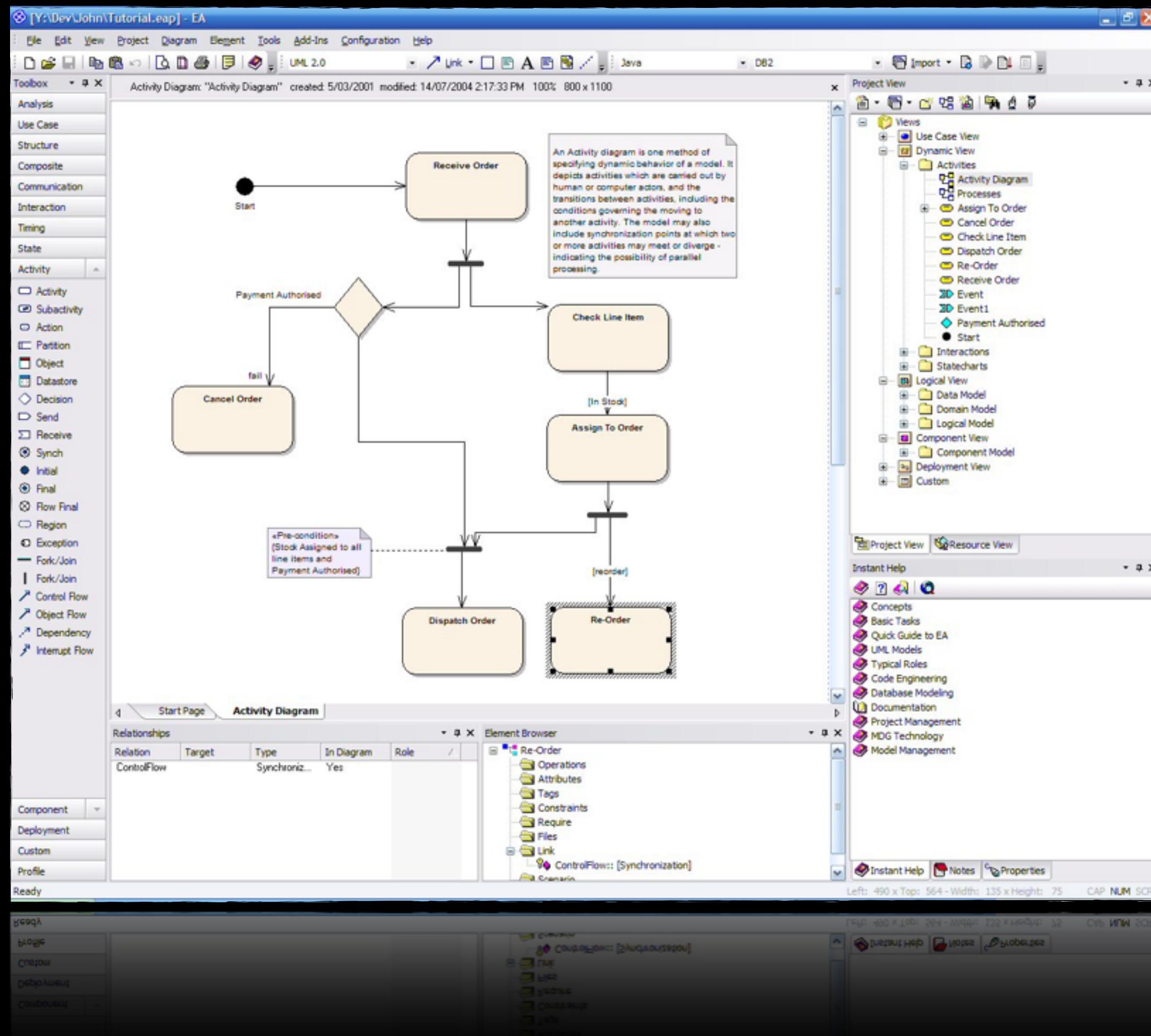
      <state name="uebergeben" id="3">
        <dependency module="PreScoring" state="vollstaendig"/>
        <transition name="bearbeiten" state="inBearbeitung"/>
      </state>

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

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

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

Warum DSLs?



- Dokumentation
- Definition von Sachverhalten mit komplexen Strukturen

Warum DSLs?

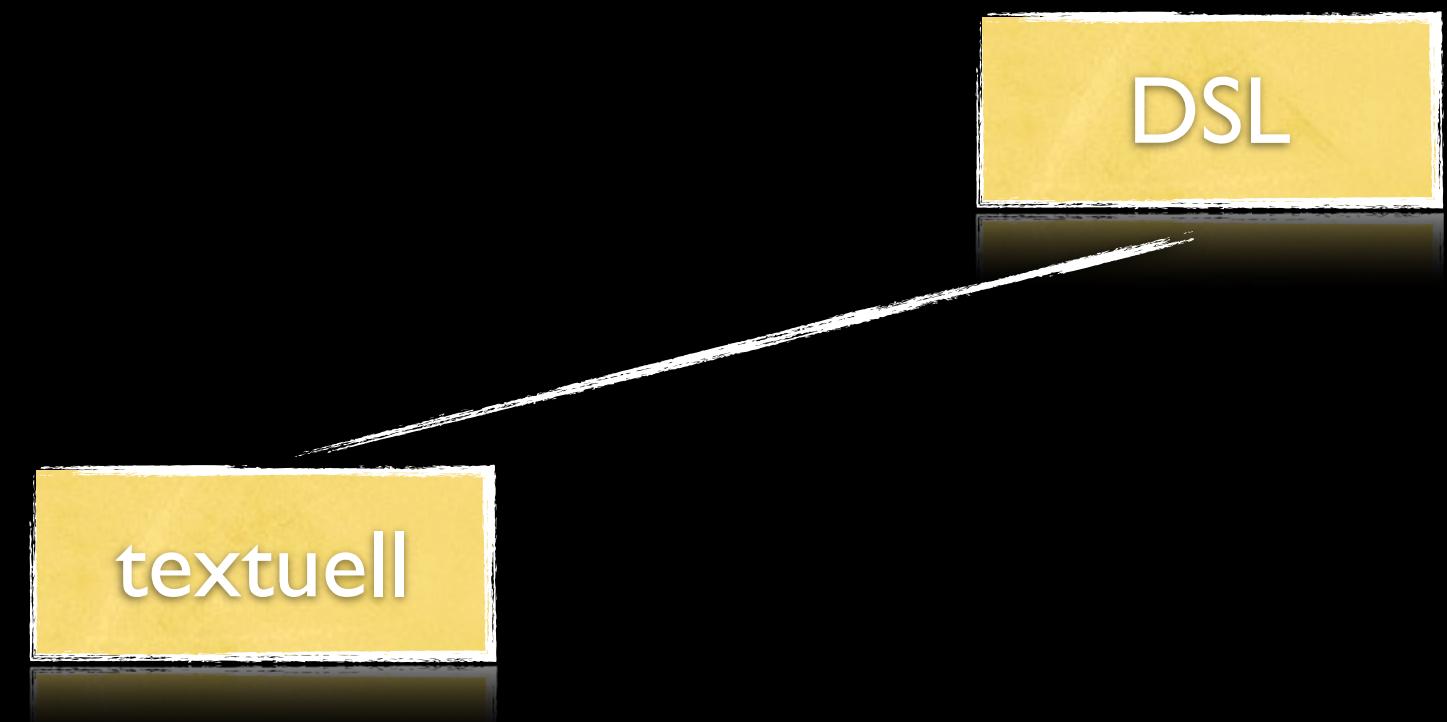


Dont
Repeat
Yourself

DSLs haben viele Gesichter

DSL

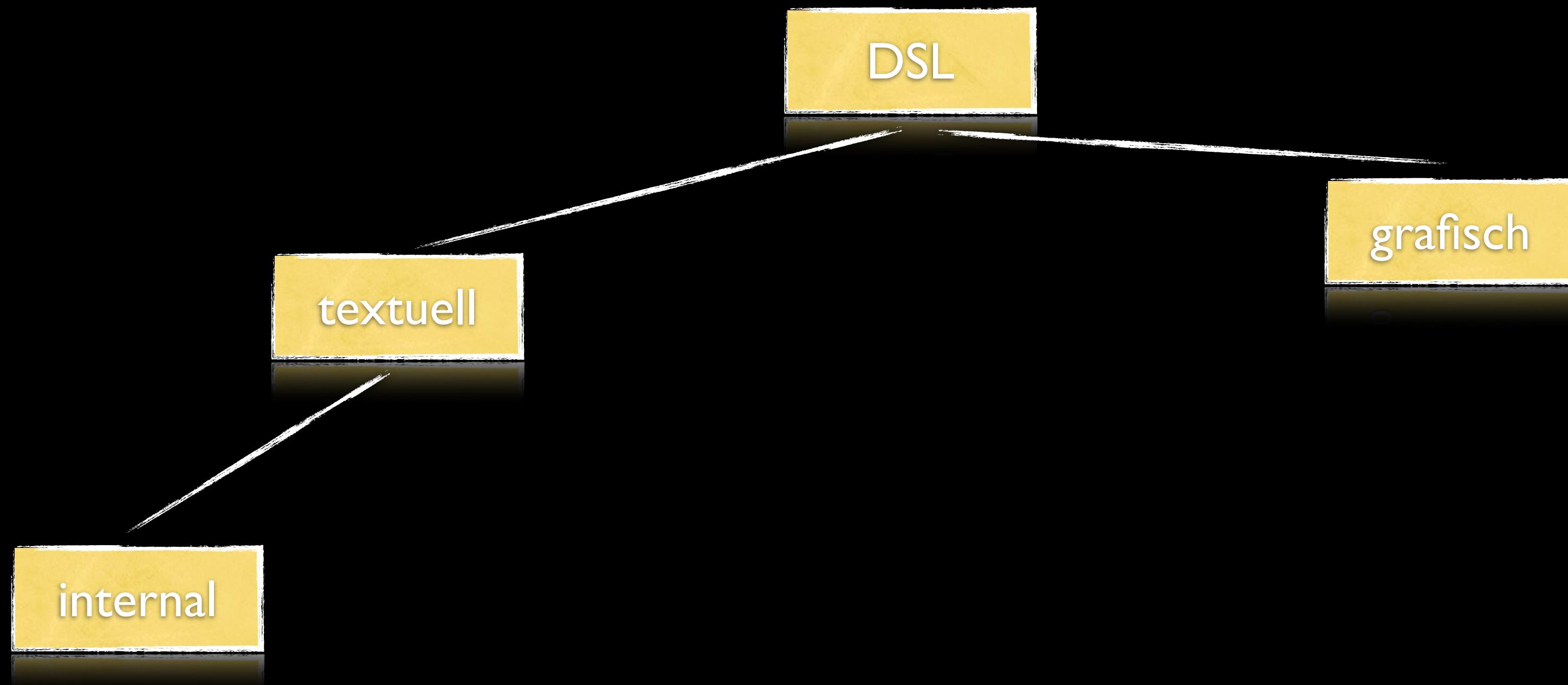
DSLs haben viele Gesichter



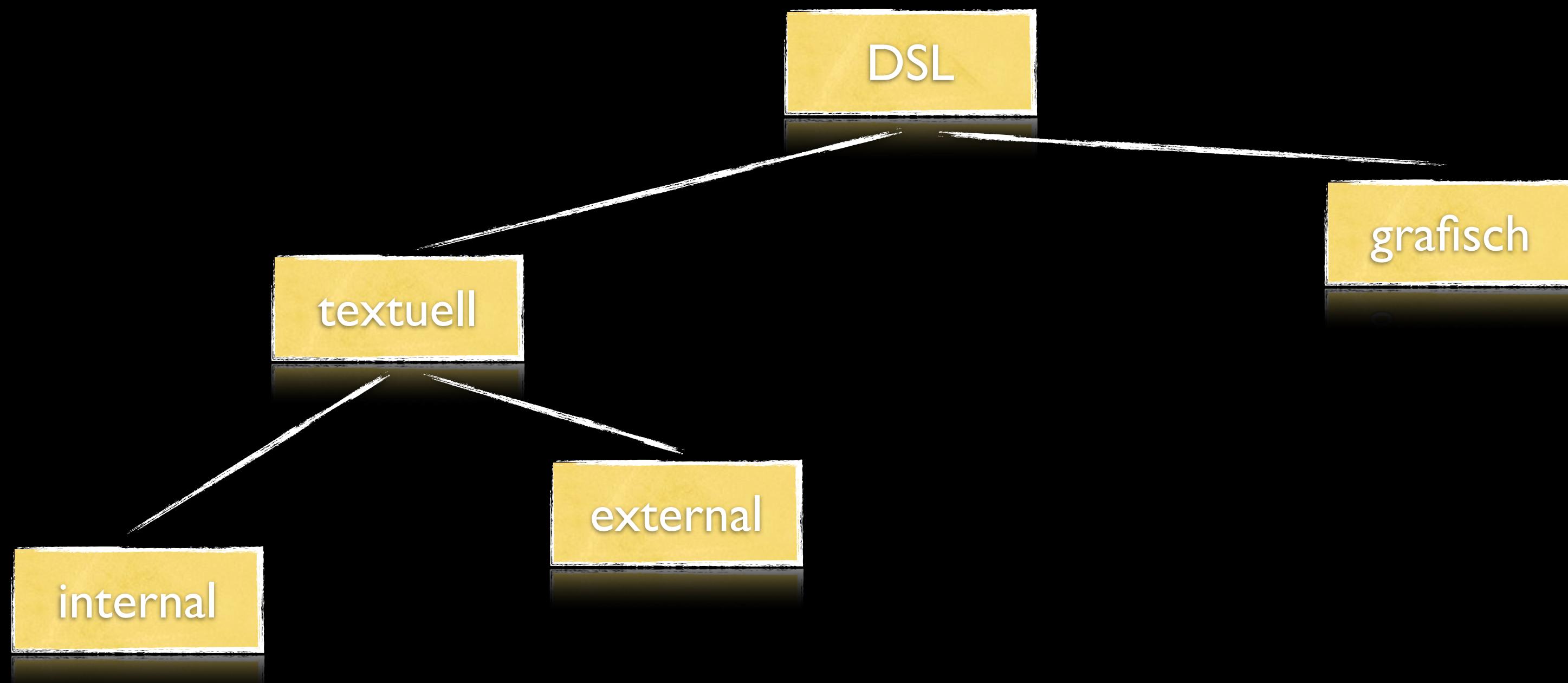
DSLs haben viele Gesichter



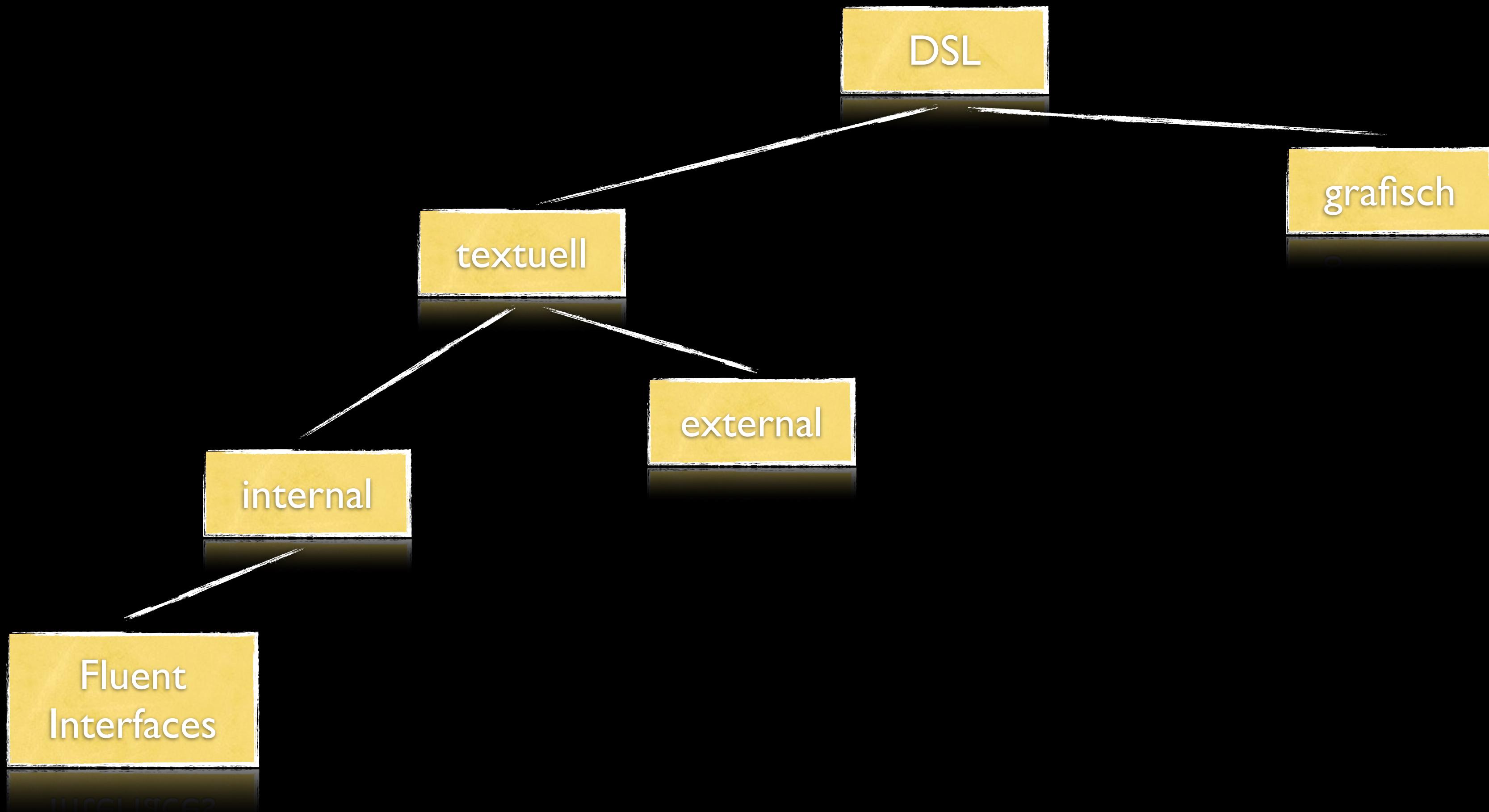
DSLs haben viele Gesichter



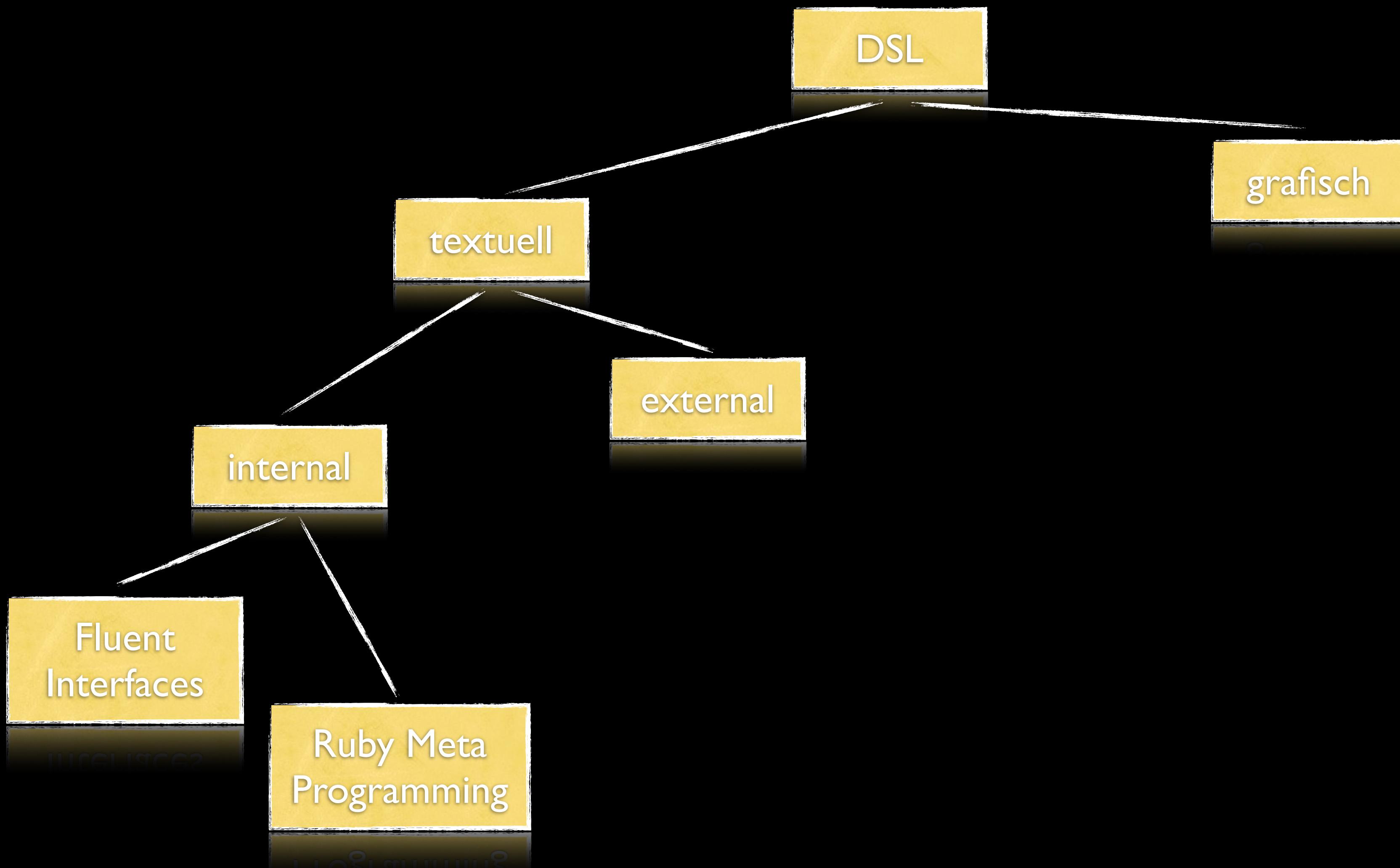
DSLs haben viele Gesichter



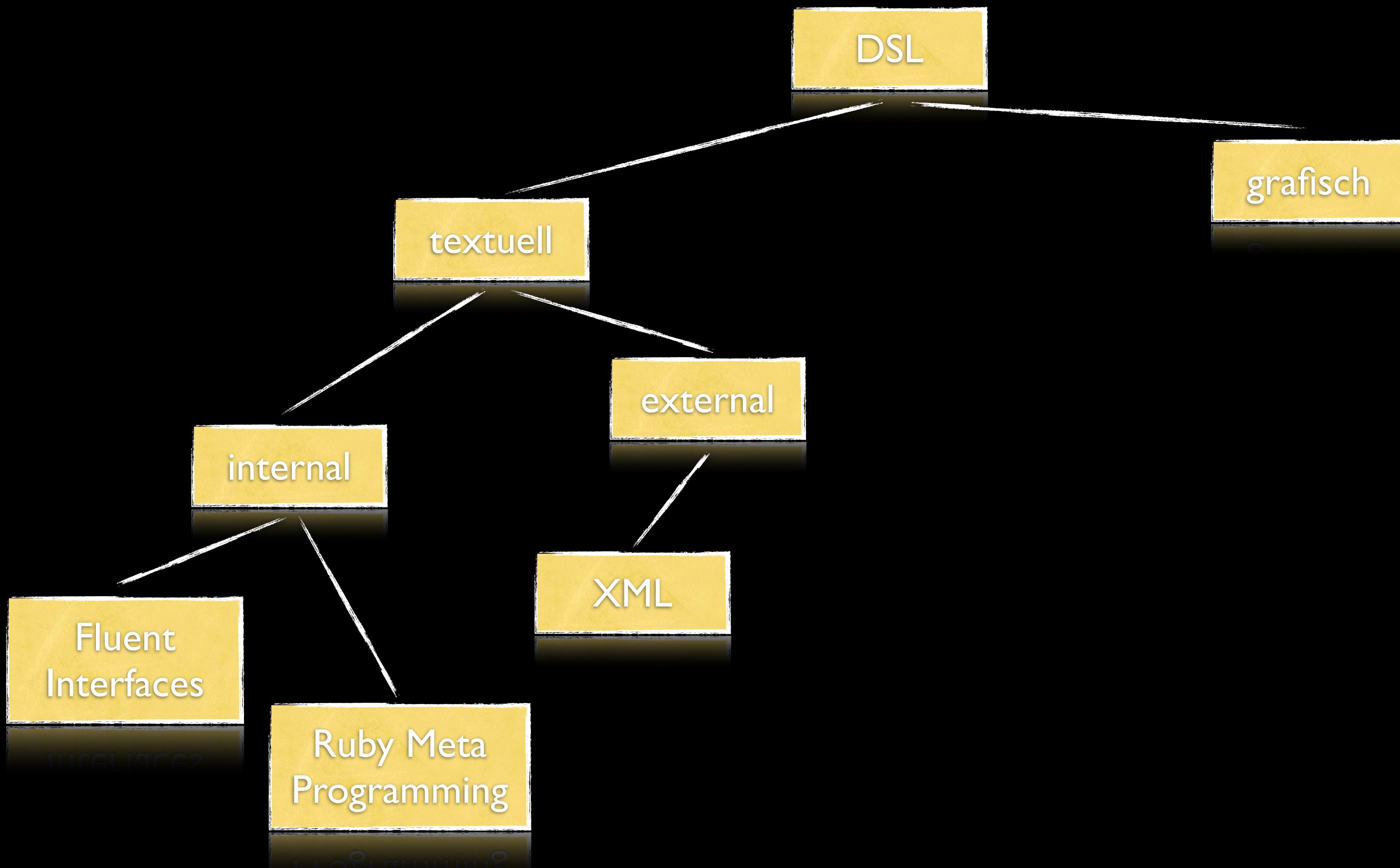
DSLs haben viele Gesichter



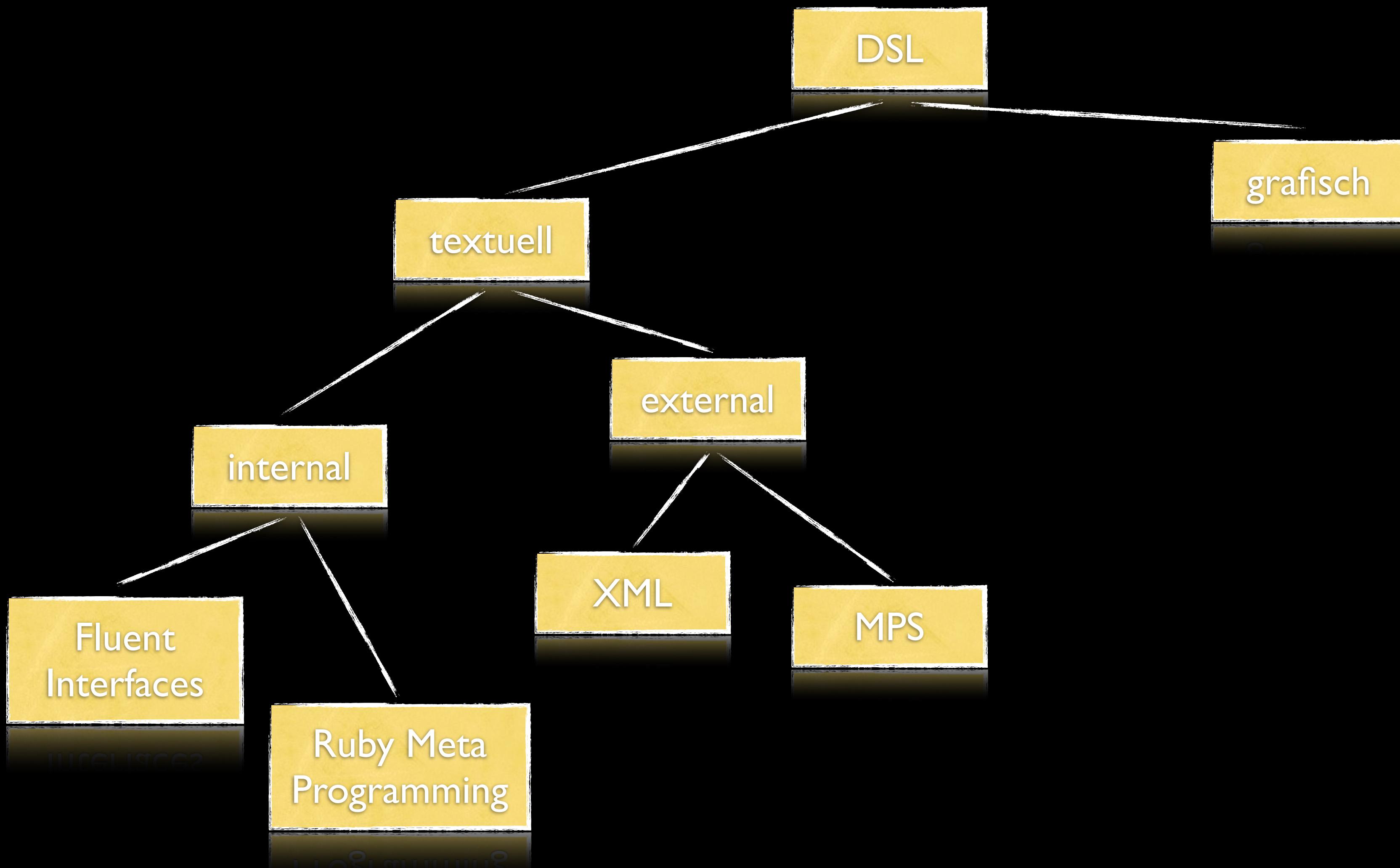
DSLs haben viele Gesichter



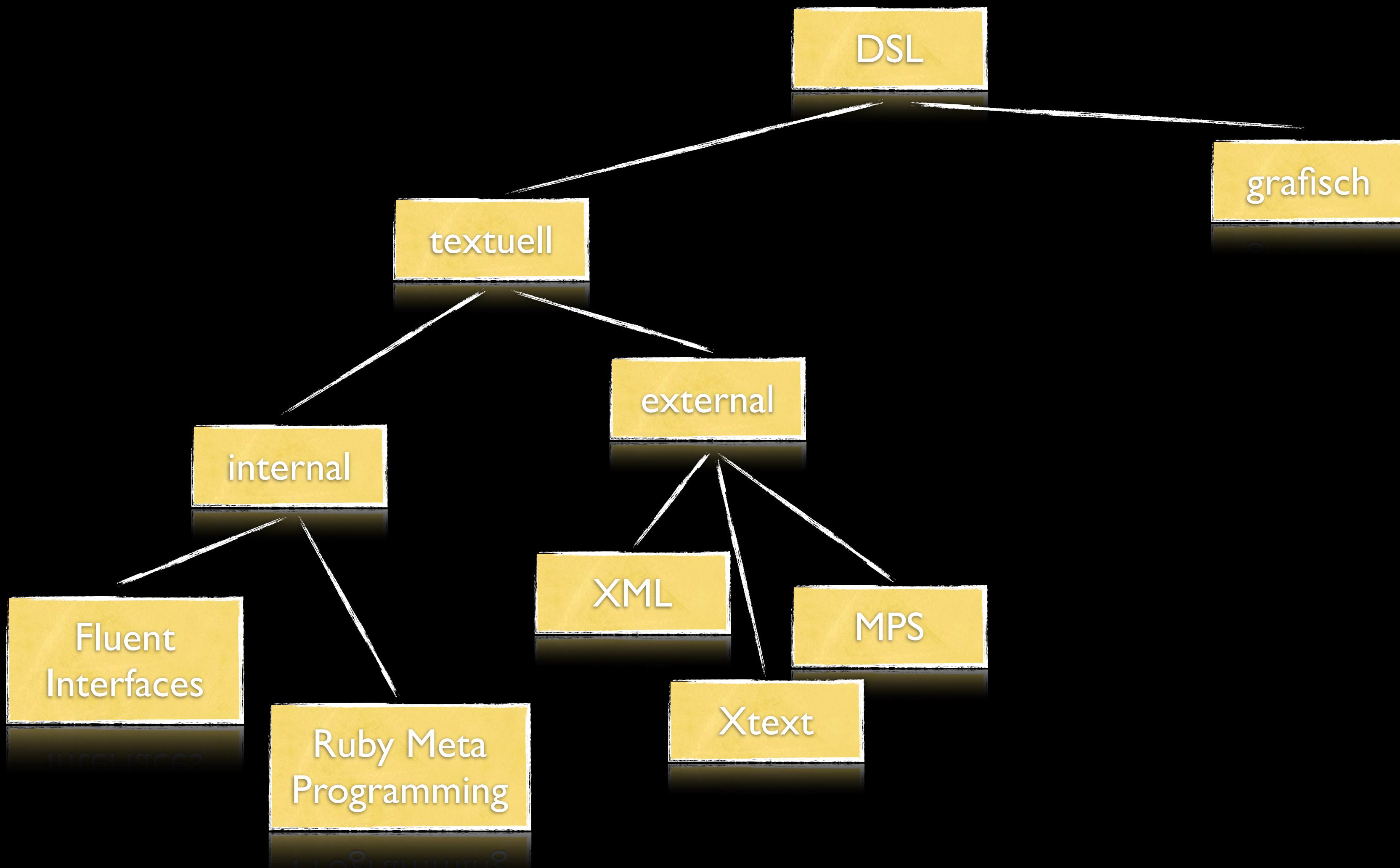
DSLs haben viele Gesichter



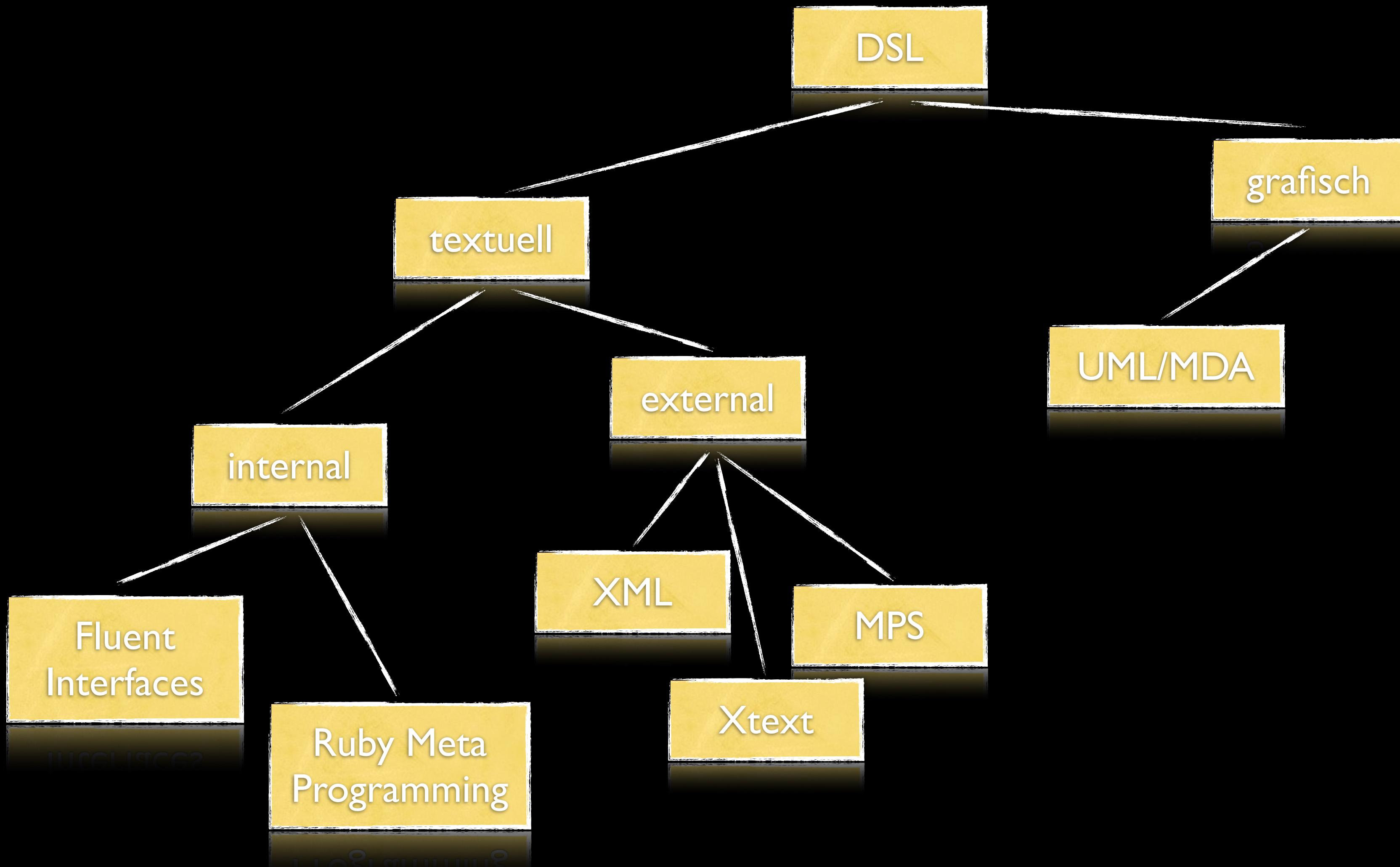
DSLs haben viele Gesichter



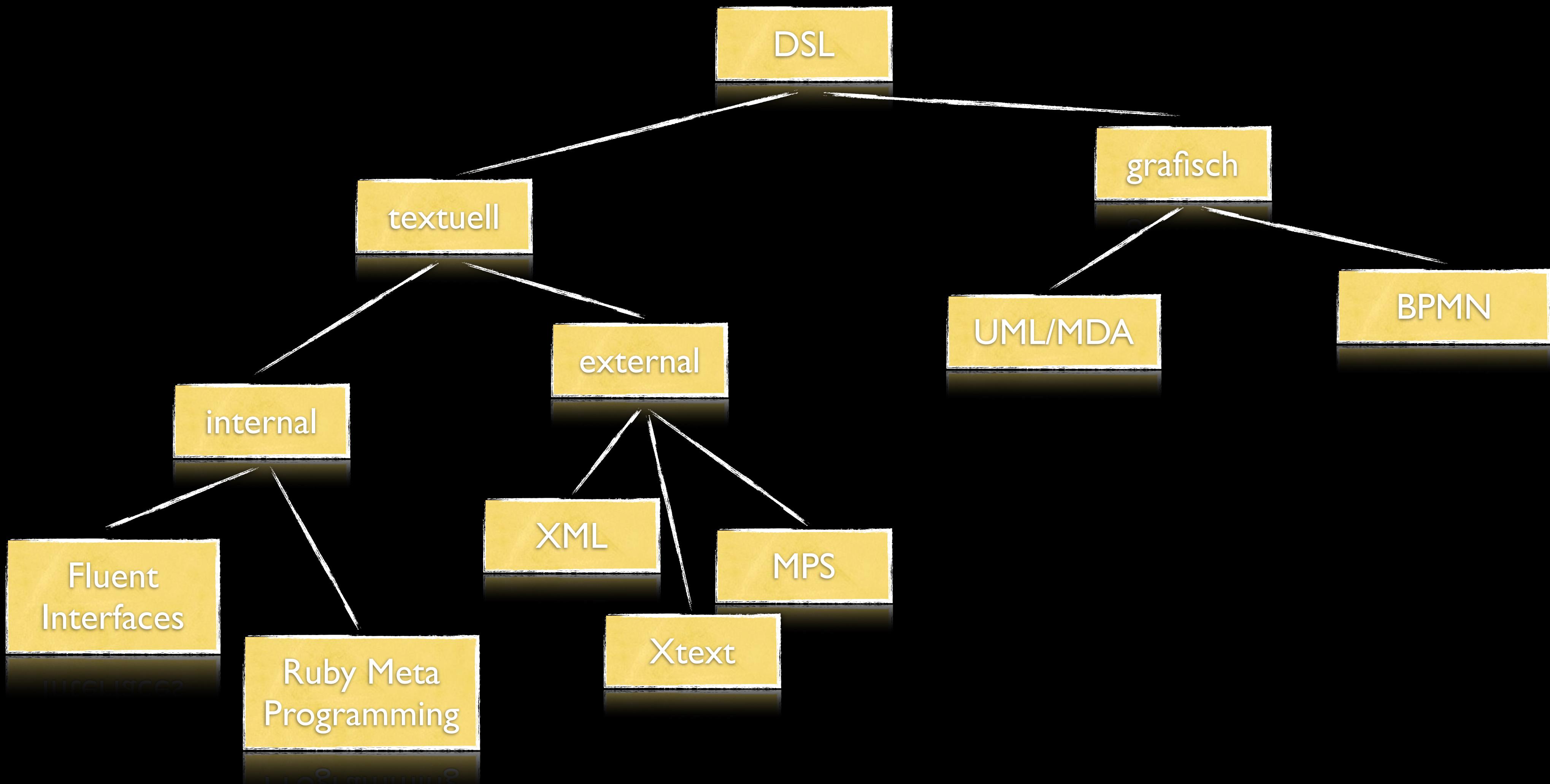
DSLs haben viele Gesichter



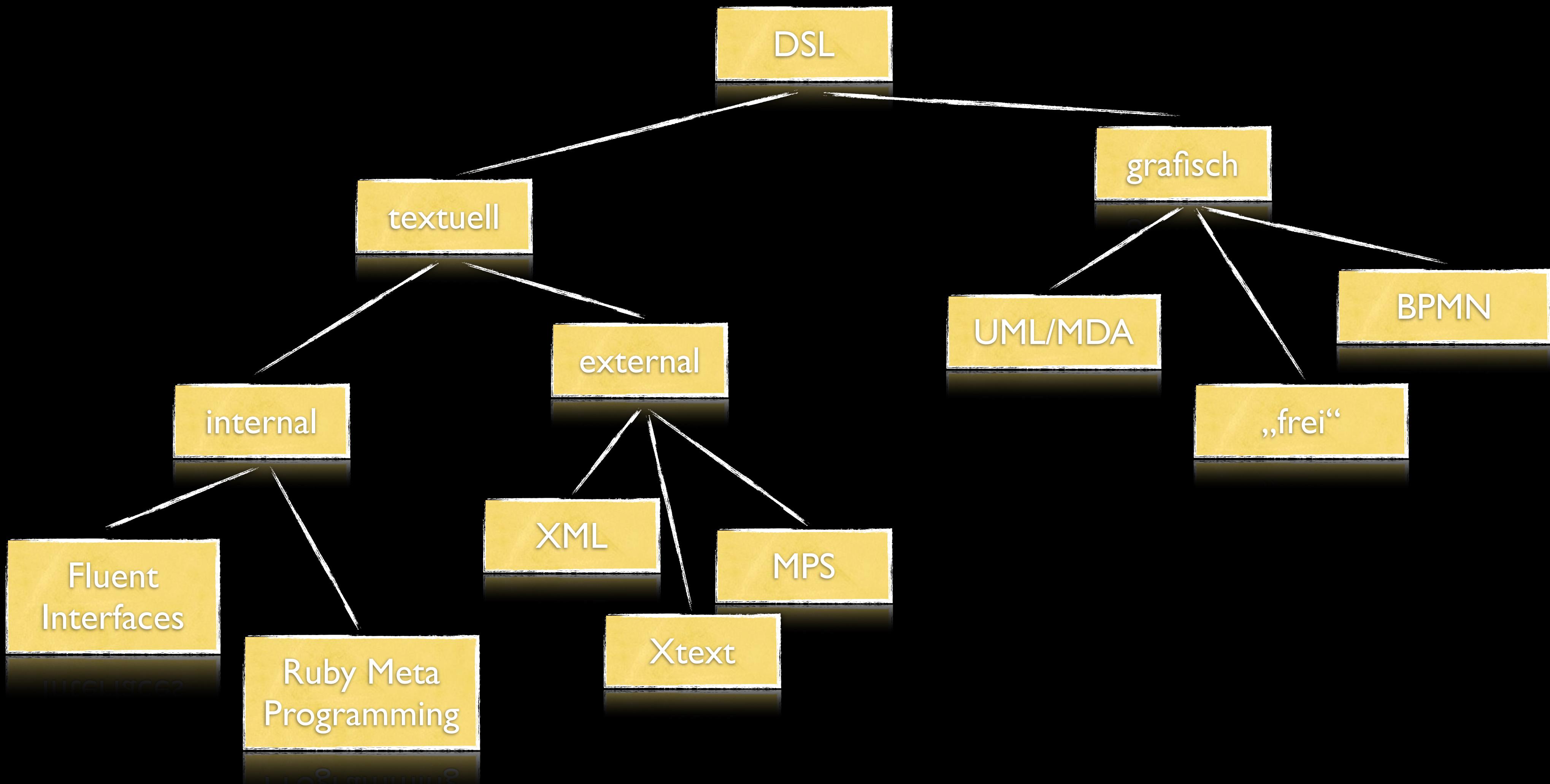
DSLs haben viele Gesichter



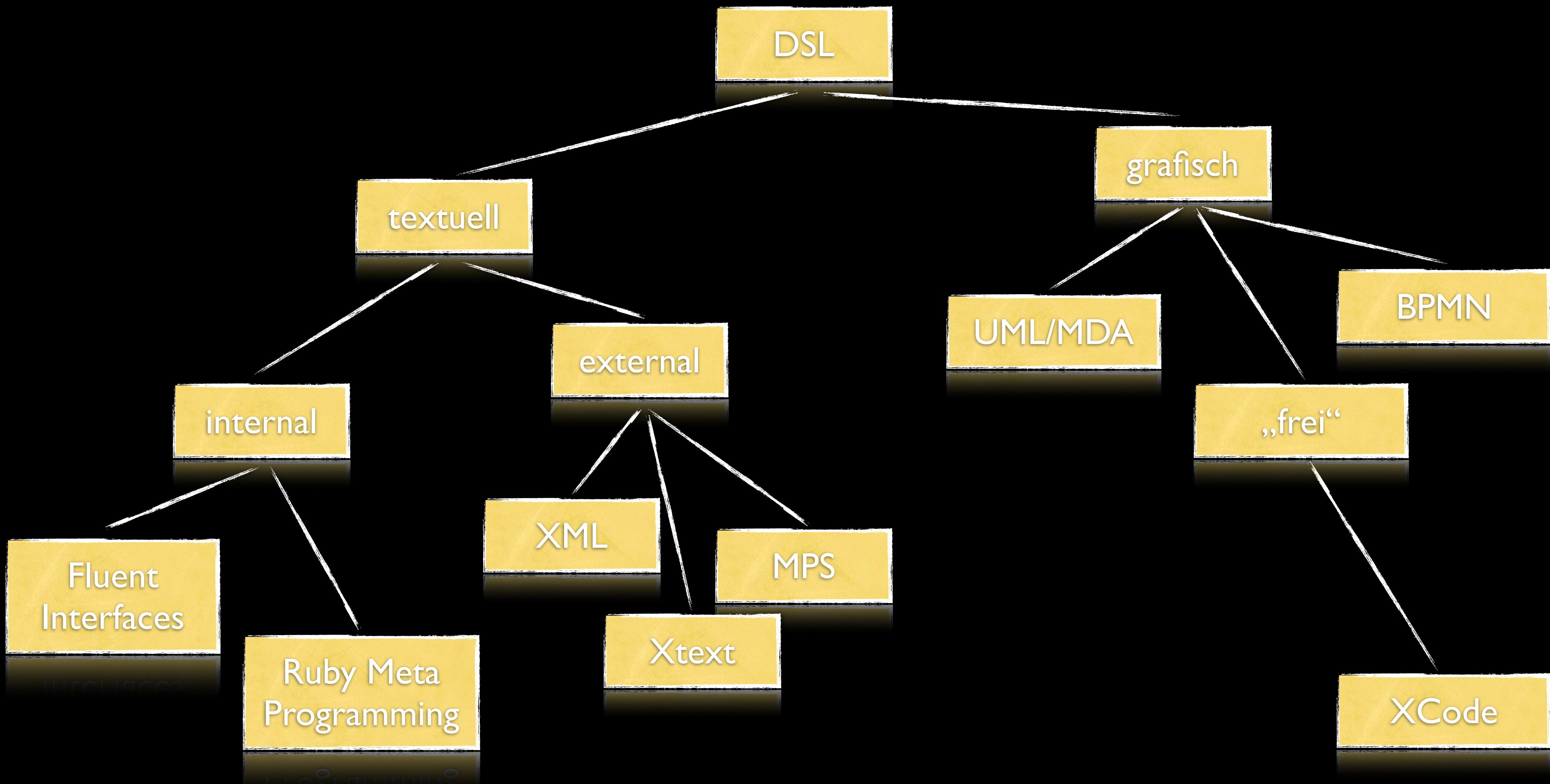
DSLs haben viele Gesichter



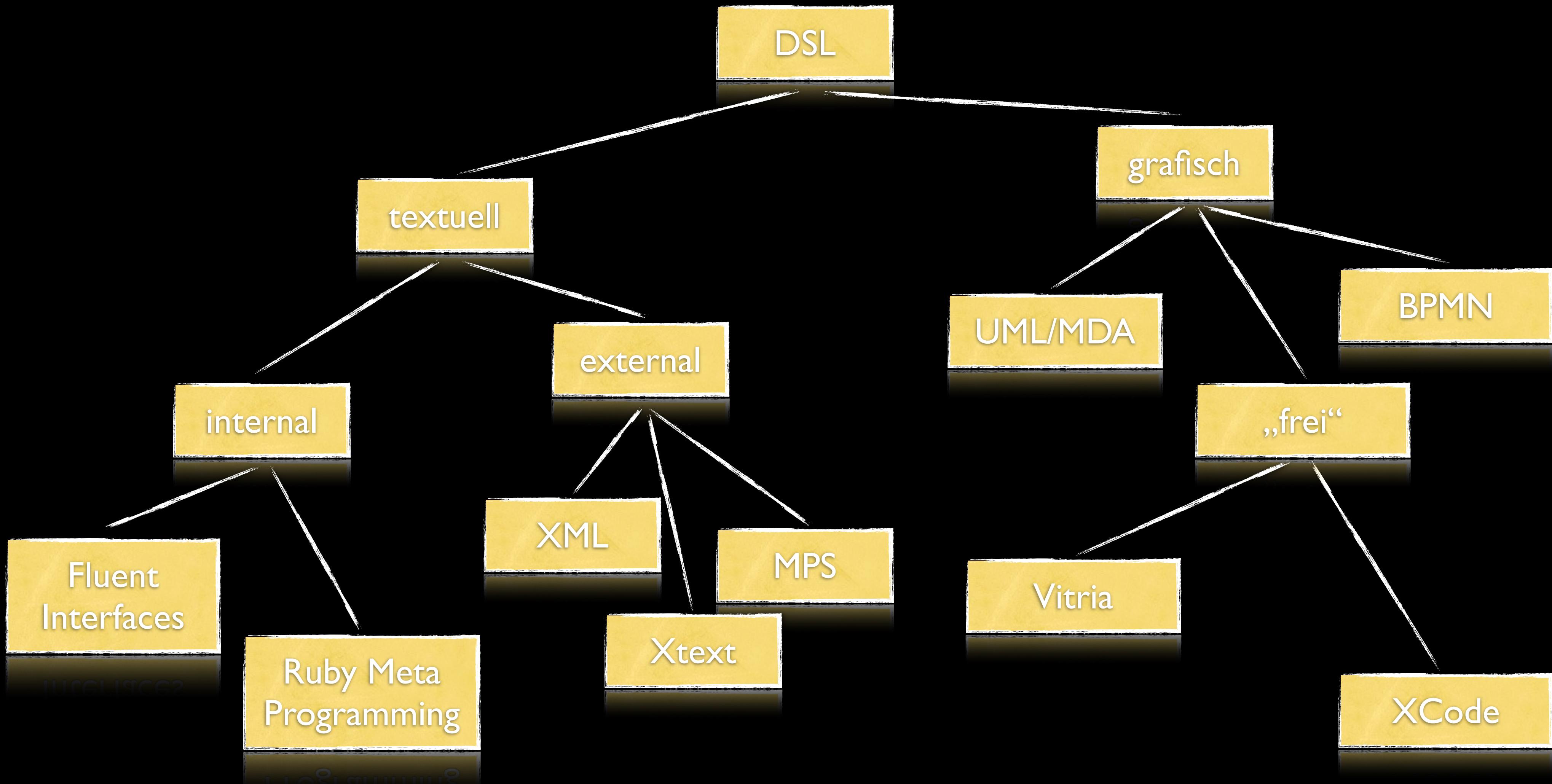
DSLs haben viele Gesichter



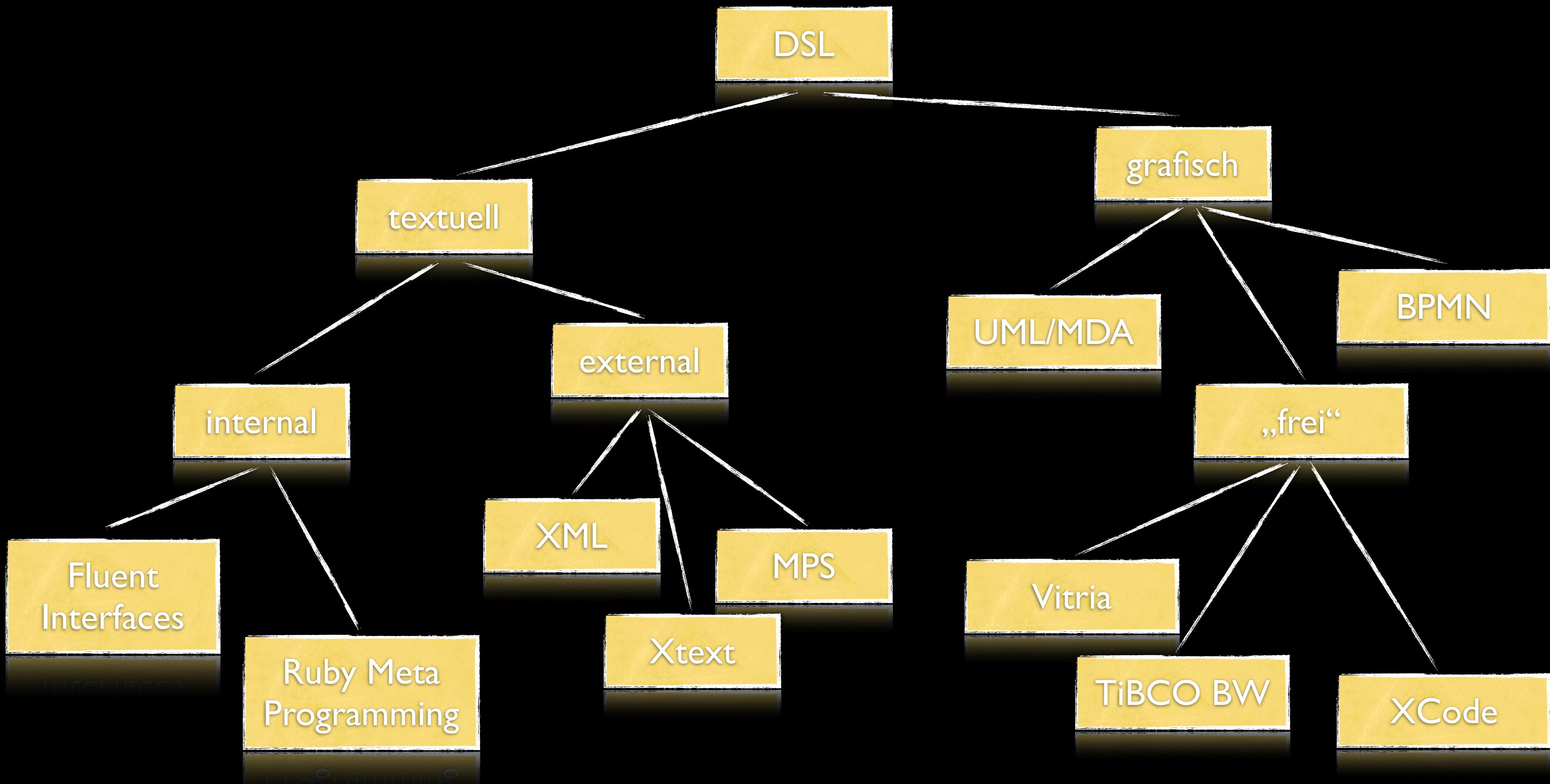
DSLs haben viele Gesichter



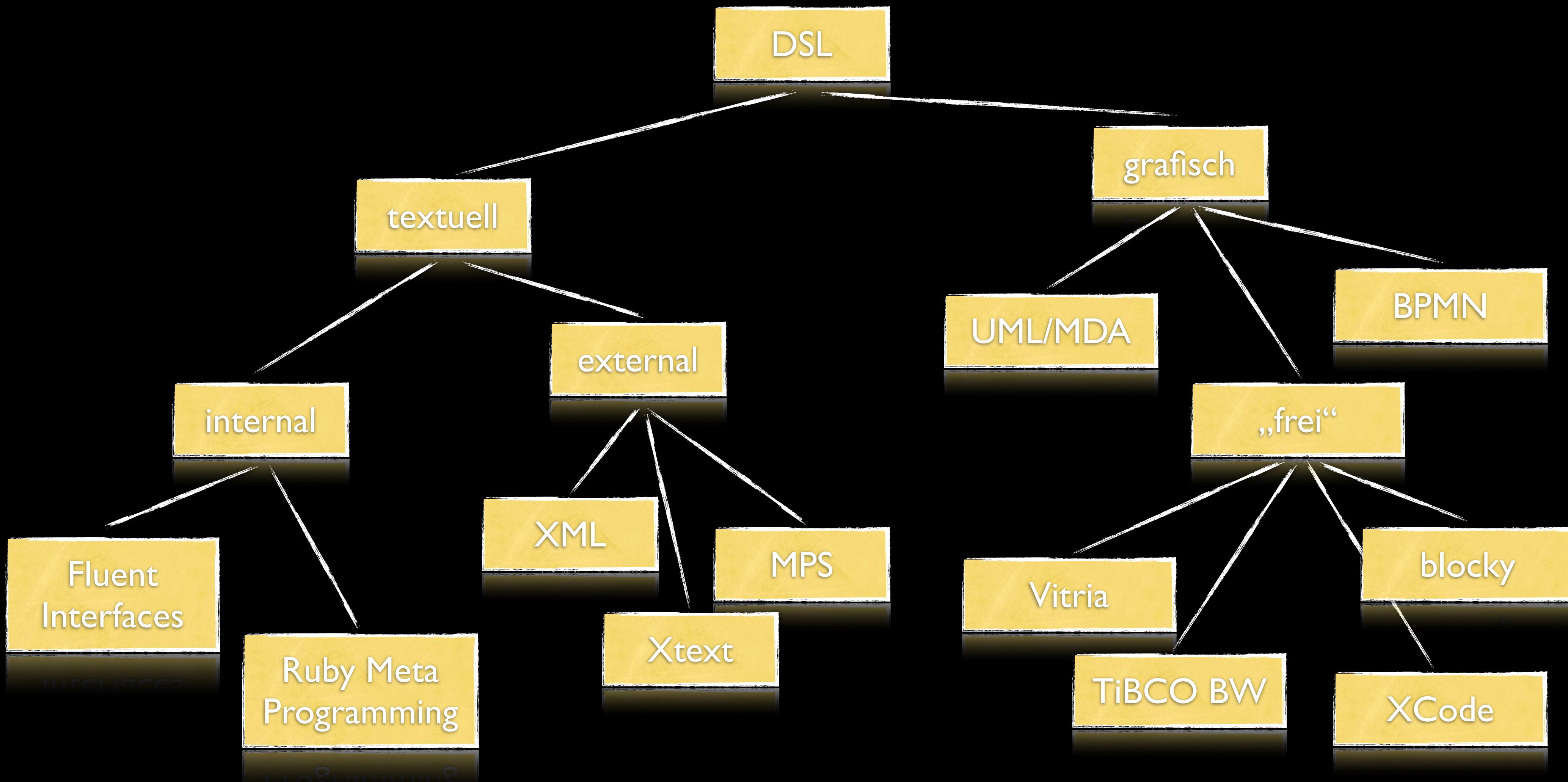
DSLs haben viele Gesichter



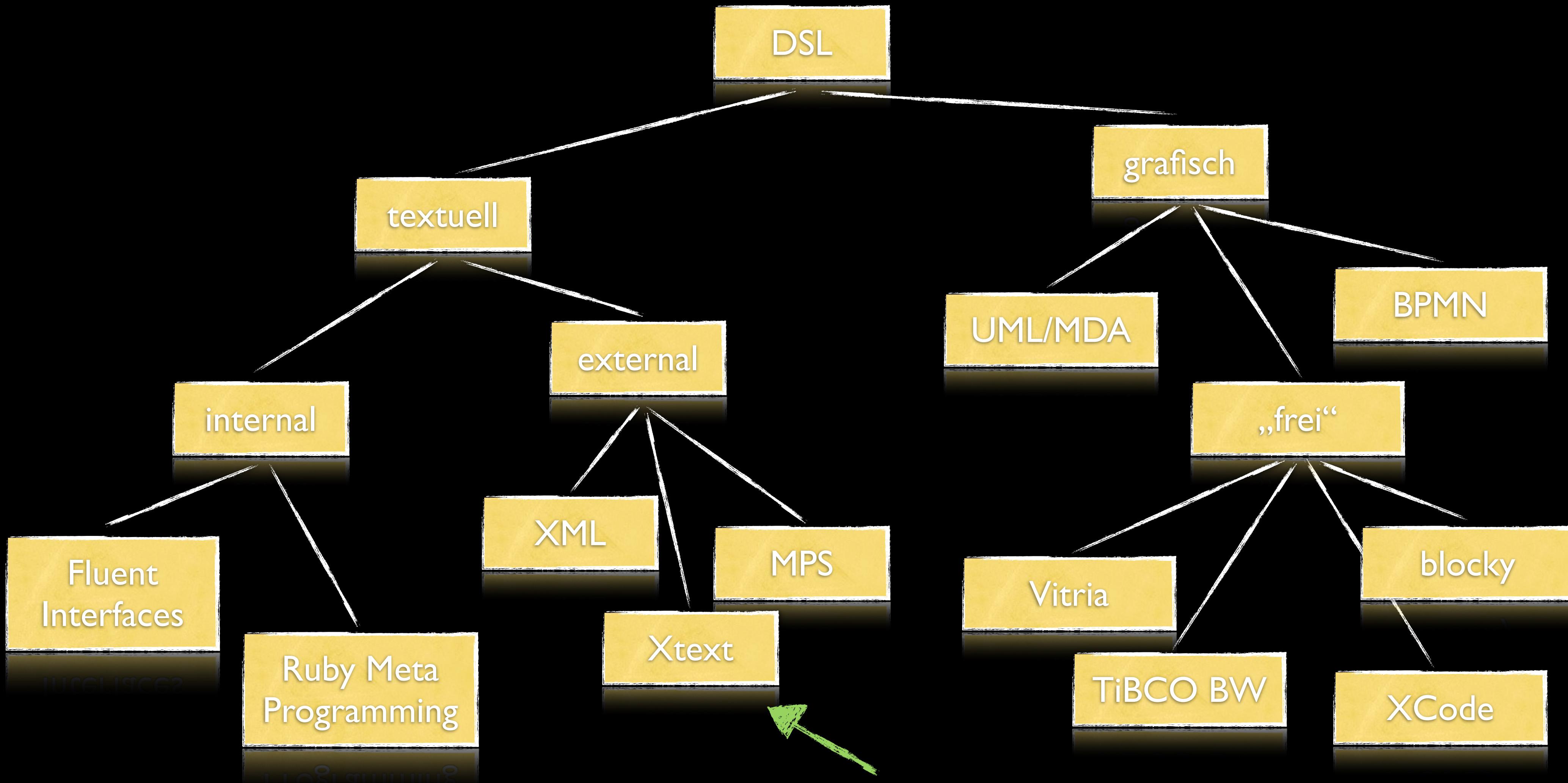
DSLs haben viele Gesichter



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?

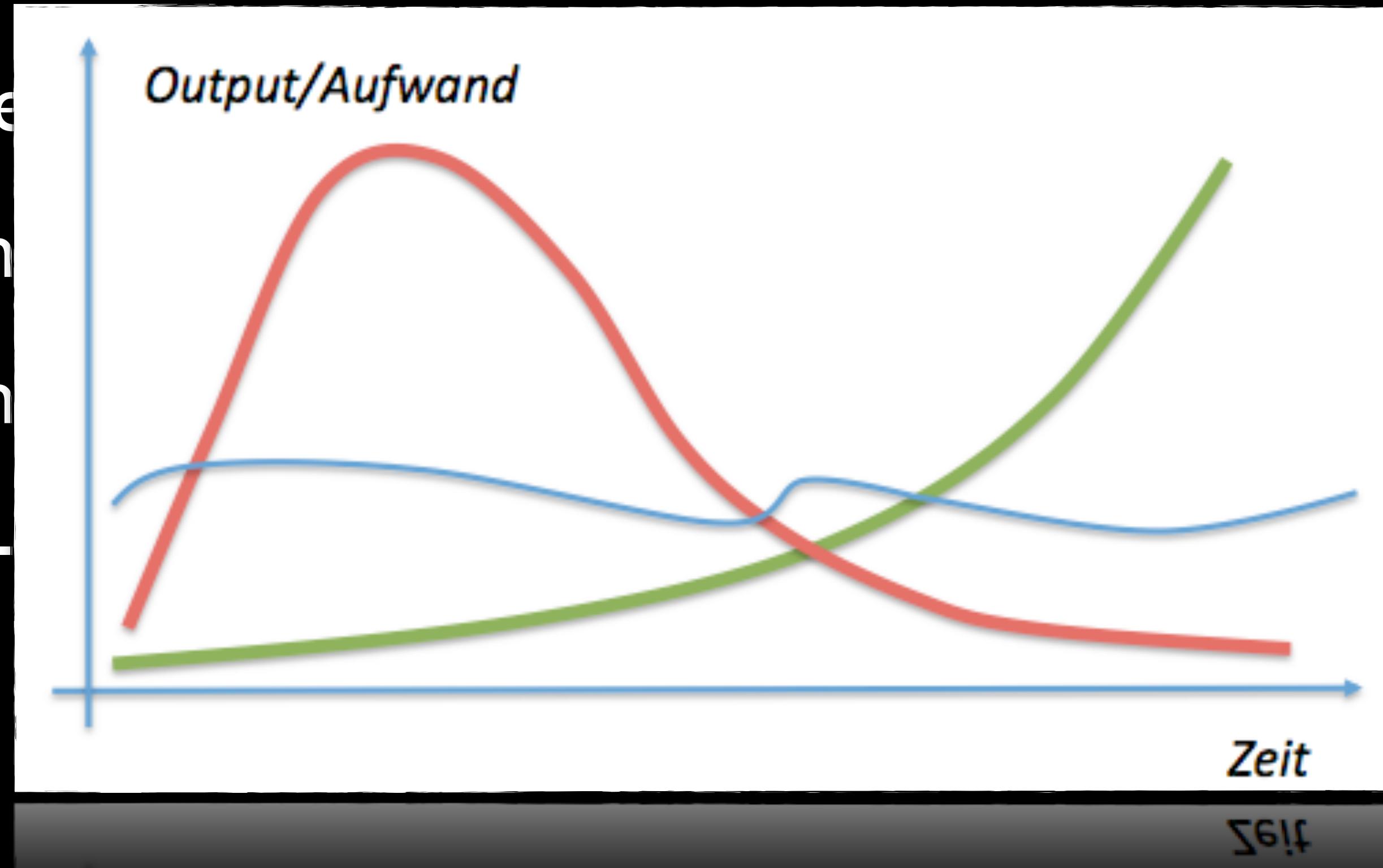
~~do obey „the law of the instrument“!~~

„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?“

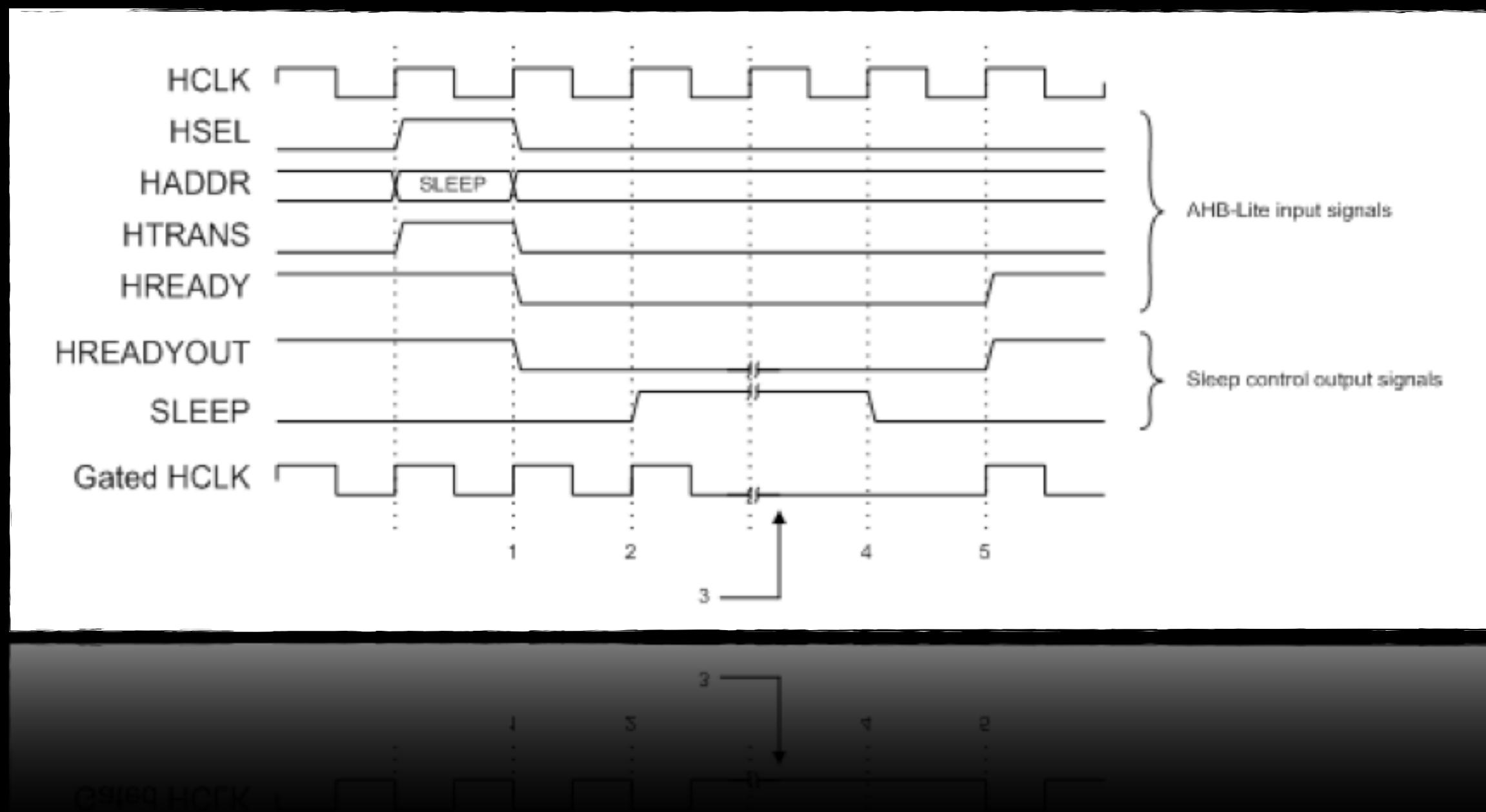
- Lern- bzw. Effizienzproblem?
- Problem von hinreichendem Wissen?
- Problem von hinreichendem Können?
- Spricht Domain-Spezialist mit dem DSL?



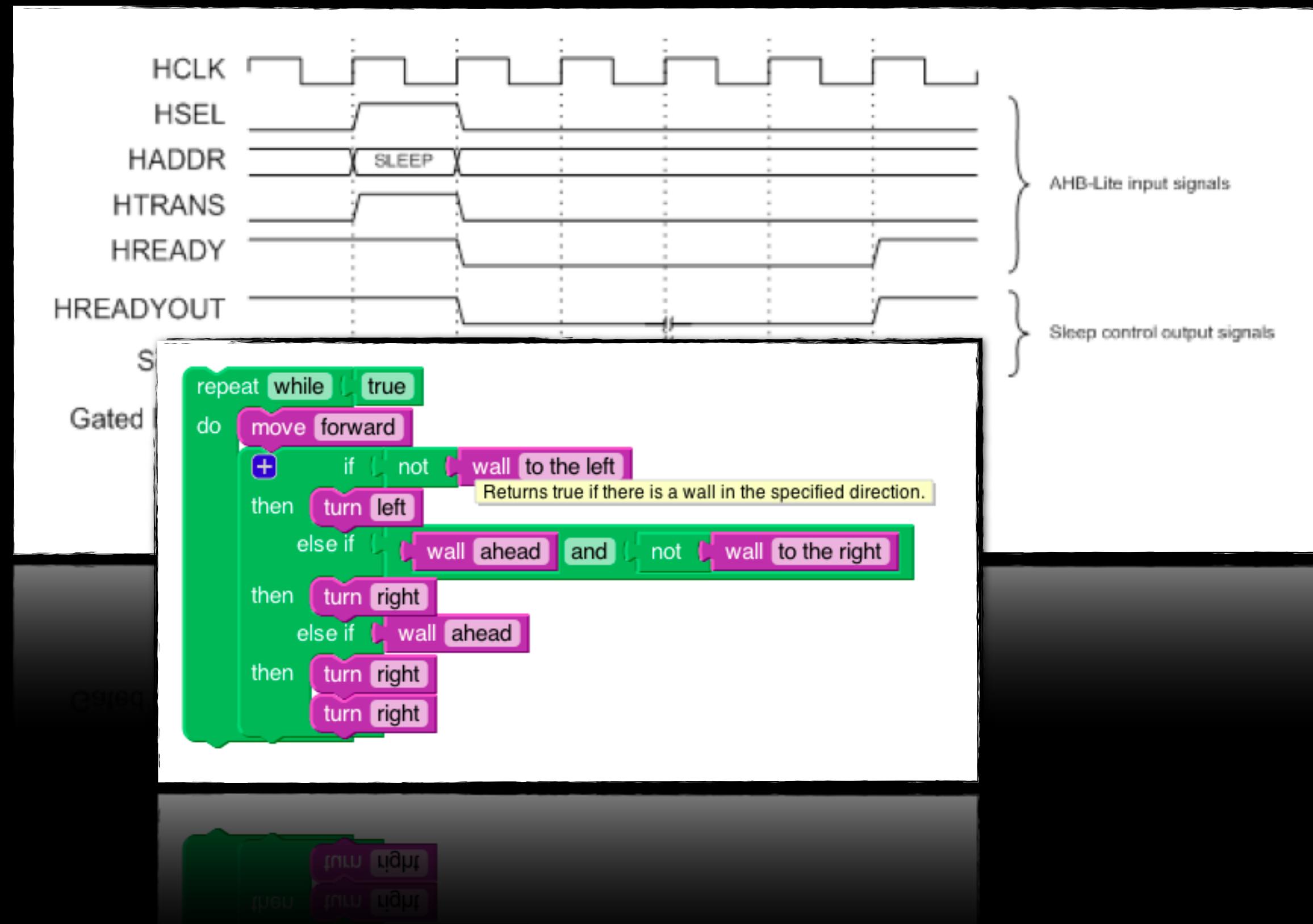
Grafisch oder Textuell?

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

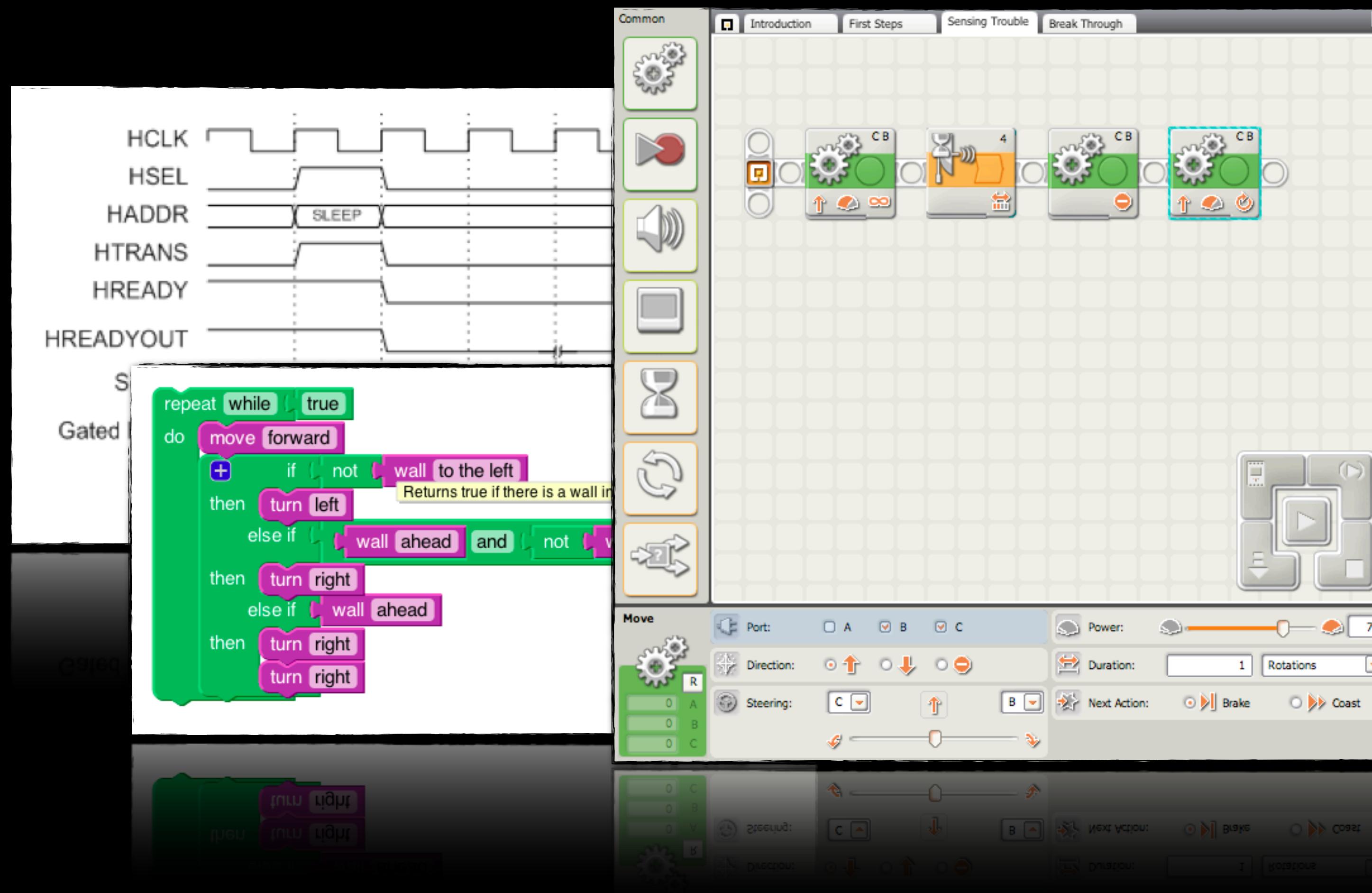
Grafisch oder Textuell?



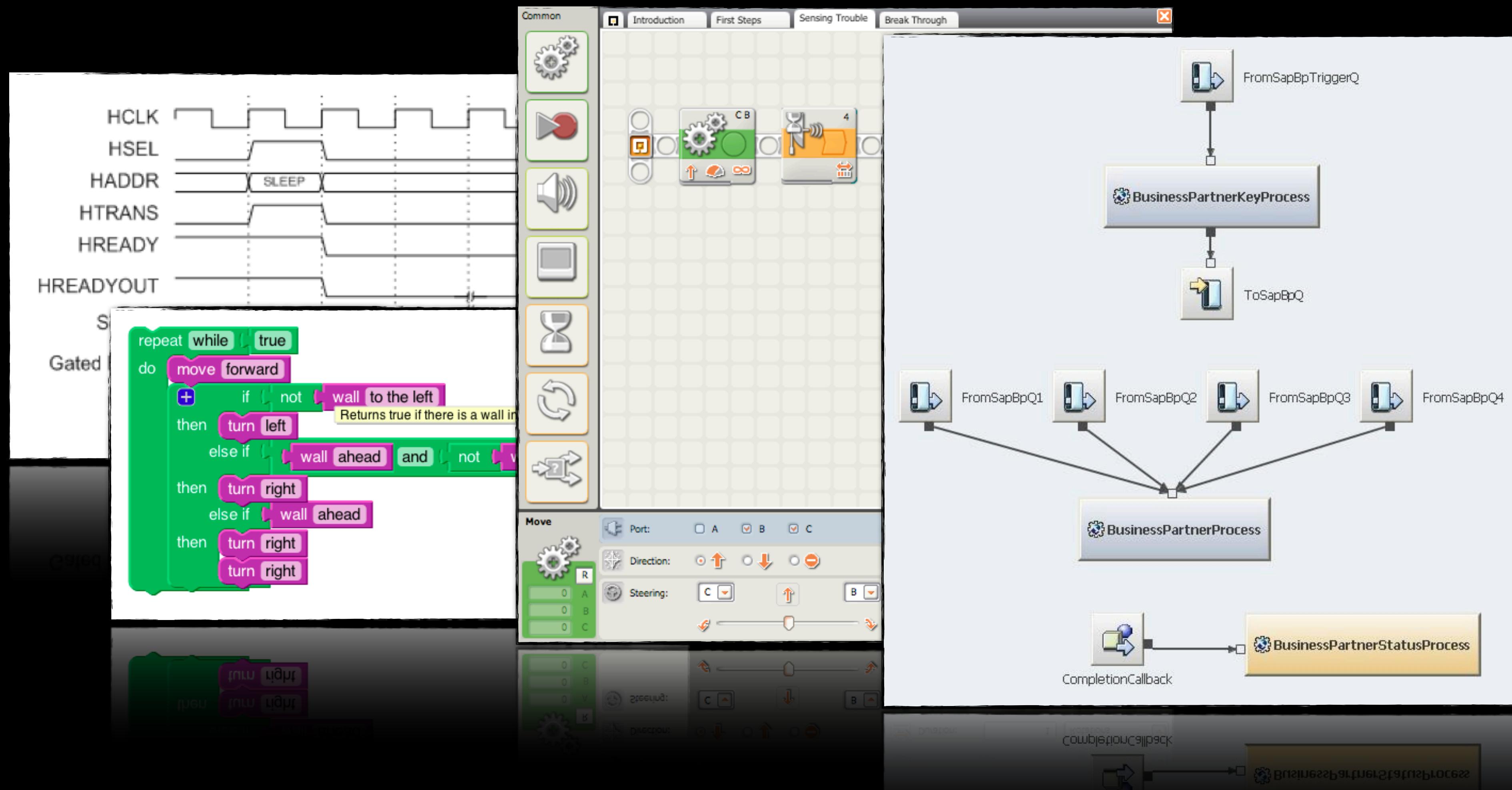
Grafisch oder Textuell?



Grafisch oder Textuell?



Grafisch oder Textuell?



Grafisch oder Textuell?

[Nichtamtliches Inhaltsverzeichnis](#)

§ 102 Ergänzende Leistungen

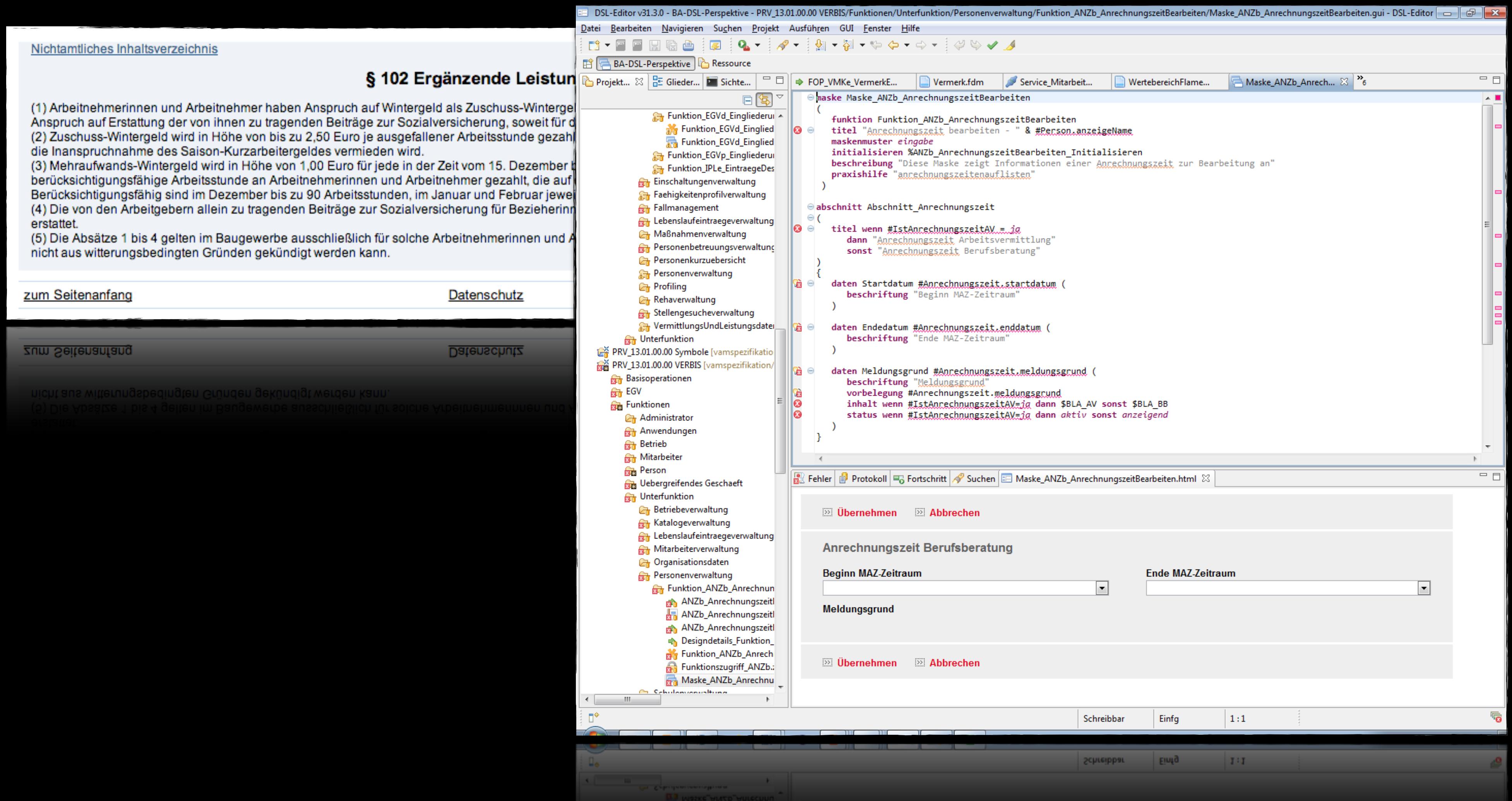
(1) Arbeitnehmerinnen und Arbeitnehmer haben Anspruch auf Wintergeld als Zuschuss-Wintergeld und Mehraufwands-Wintergeld und Arbeitgeber haben Anspruch auf Erstattung der von ihnen zu tragenden Beiträge zur Sozialversicherung, soweit für diese Zwecke Mittel durch eine Umlage aufgebracht werden.
(2) Zuschuss-Wintergeld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

[Zurück zur Seite mit dem letzten Artikel](#) [Datenschutz](#) [Seite ausdrucken](#)

Die Seite wurde zuletzt am 10.01.2013 um 10:45 Uhr aktualisiert. Letzte Änderungen wurden am 10.01.2013 um 10:45 Uhr vorgenommen. (2)

Grafisch oder Textuell?



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>

```
NSDictionary *fields = [[NSMutableDictionary alloc] init];
* savingsTargetID = [NSNumber numberWithInt:[results objectForKey:@"savingsTargetID"]];
* setObject:savingsTargetID forKey:@"savingsTargetID";
* categoryID = [NSNumber numberWithInt:[results objectForKey:@"categoryID"]];
* setObject:categoryID forKey:@"categoryID";
* parentGoalID = [NSNumber numberWithInt:[results objectForKey:@"parentGoalID"]];
* setObject:parentGoalID forKey:@"parentGoalID";
* name = [results objectForKey:@"name"];
* setObject:name forKey:@"name";
* color = [results objectForKey:@"color"];
* setObject:color forKey:@"color";
* saveAmount = [NSNumber numberWithInt:[results objectForKey:@"saveAmount"]];
* setObject:saveAmount forKey:@"saveAmount";
```

Umgebung & Technologie

```
1. grammar org.xtext.example.mydsl.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



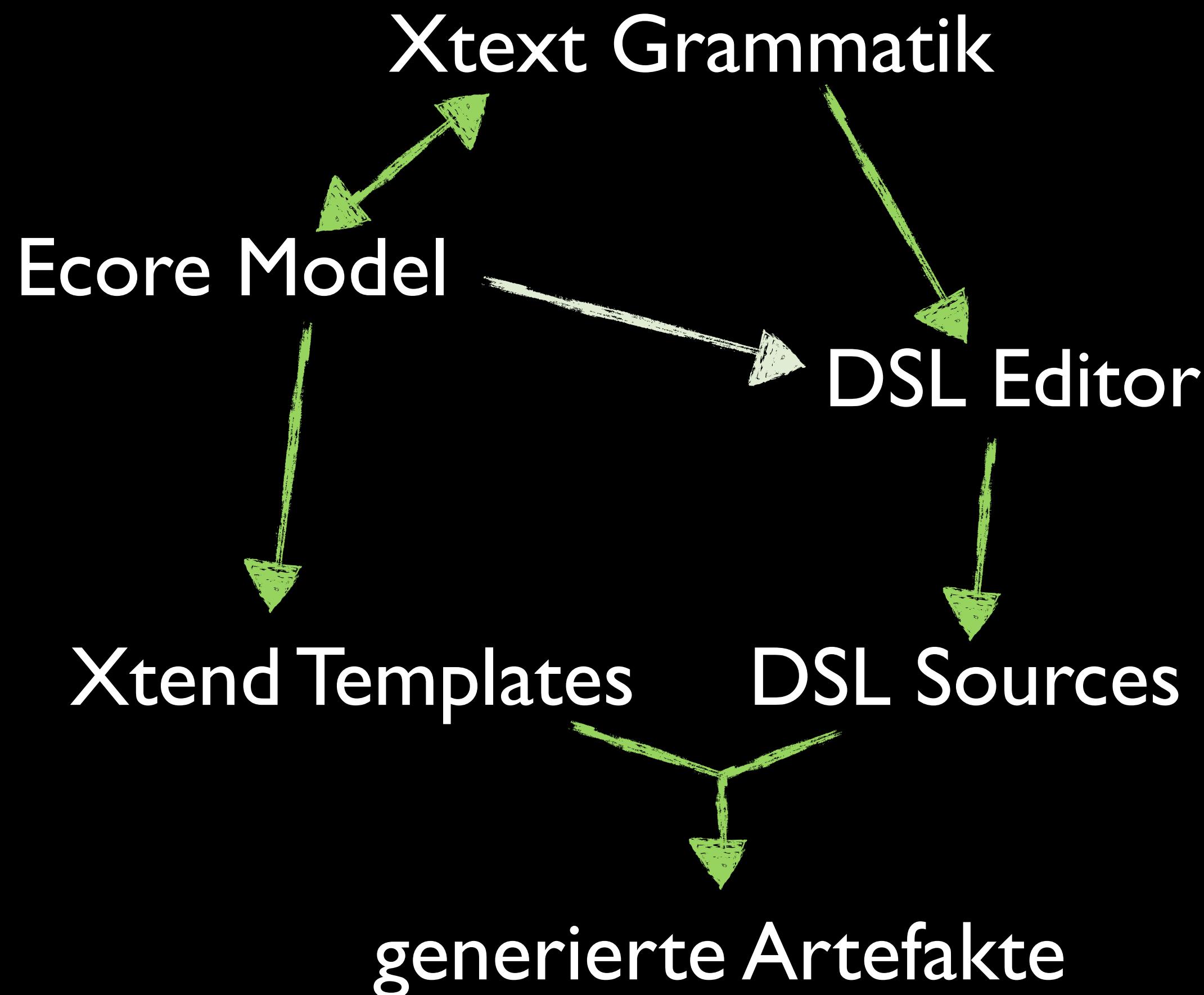
Umgebung & Technologie

- 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
- (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

<http://git.eclipse.org/c/tmf/org.eclipse.xtext.git/tree/plugins/org.eclipse.xtext/src/org/eclipse/xtext/Xtext.xtext>

Xtext Basics

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

Xtext Basics

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

Model:

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

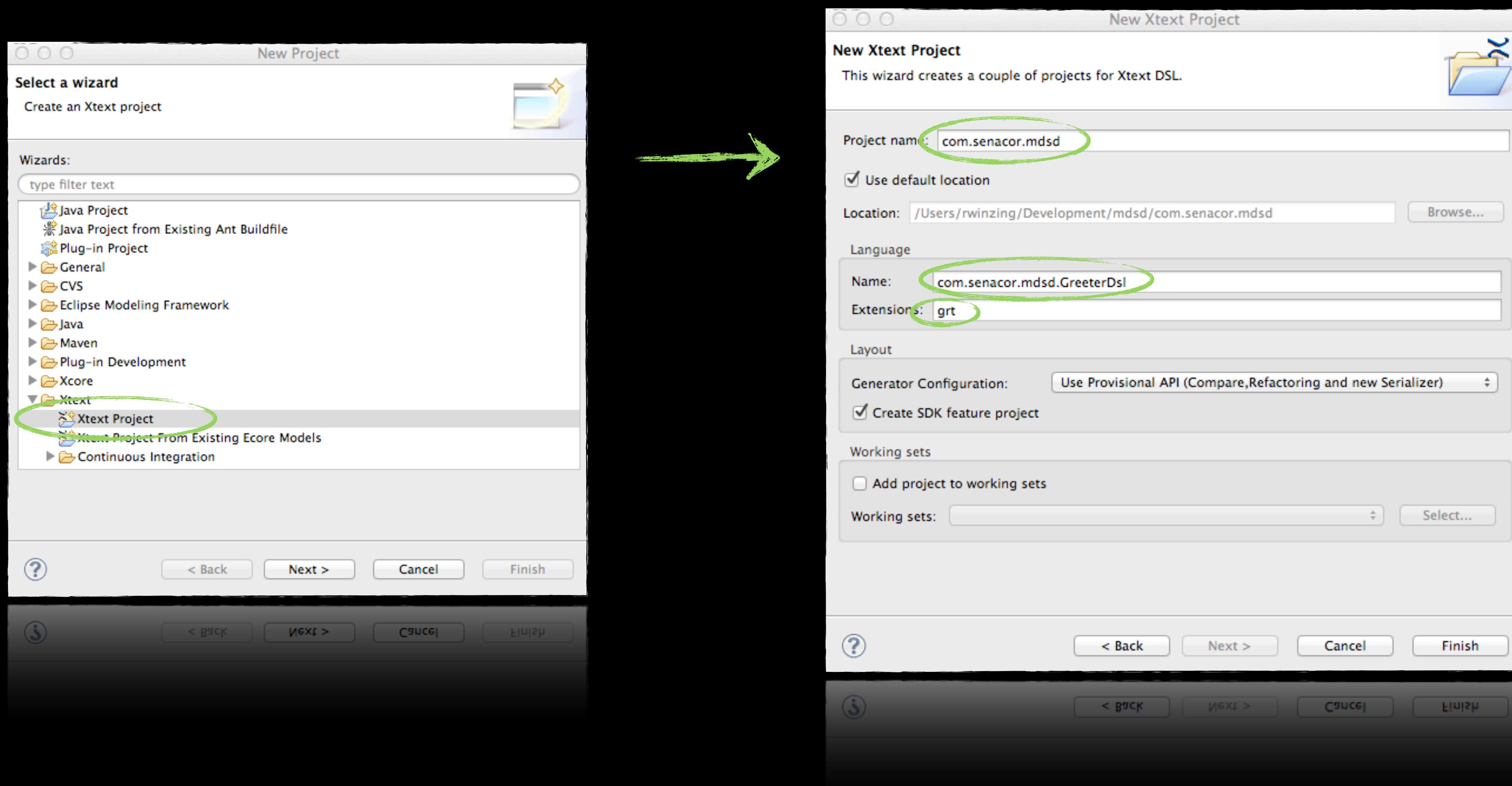
Greeting:

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

Hello,World! (I)

- Projekt erzeugen
- Editor testen
- generierte Artefakte ansehen

Ein erstes Xtext Projekt ...



Hello,World! (2)

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

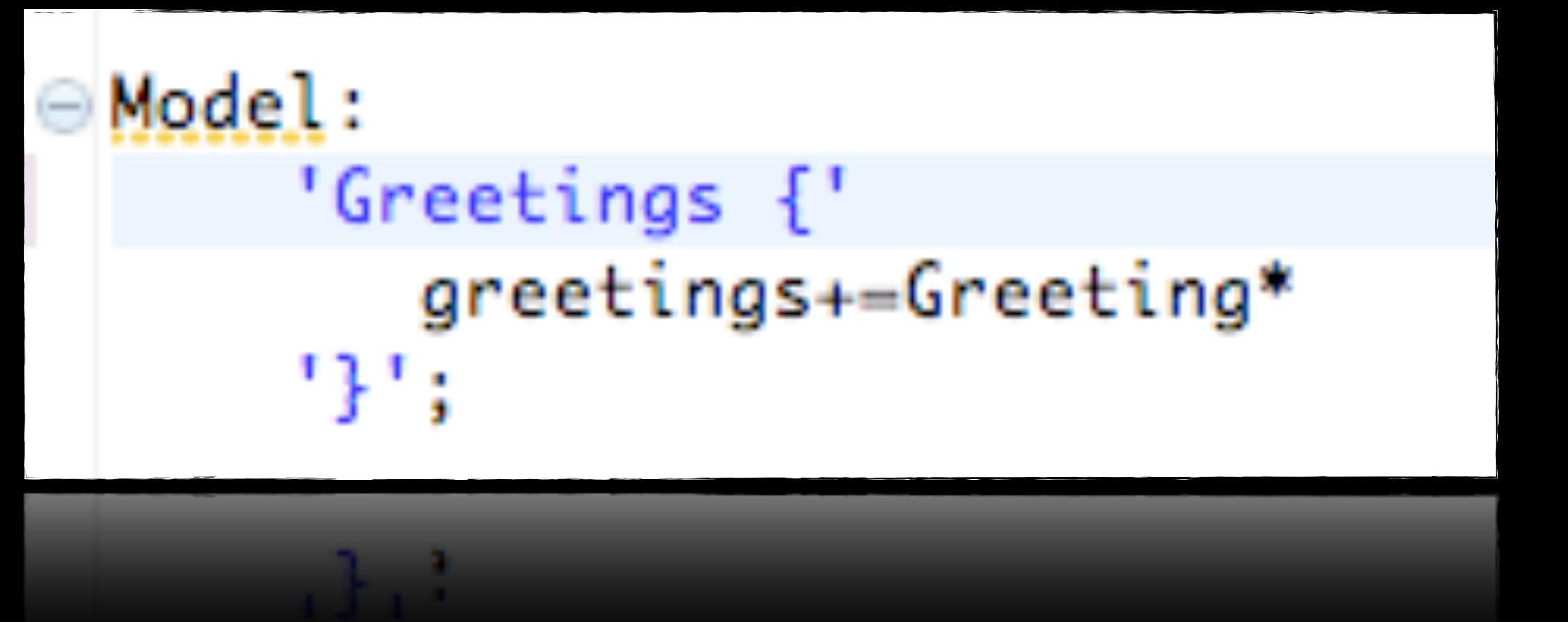


```
test.grt01 test.grt02
Greetings {
Hello "World"!
}
```

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

Hello,World! (2)

Wie wäre es mit



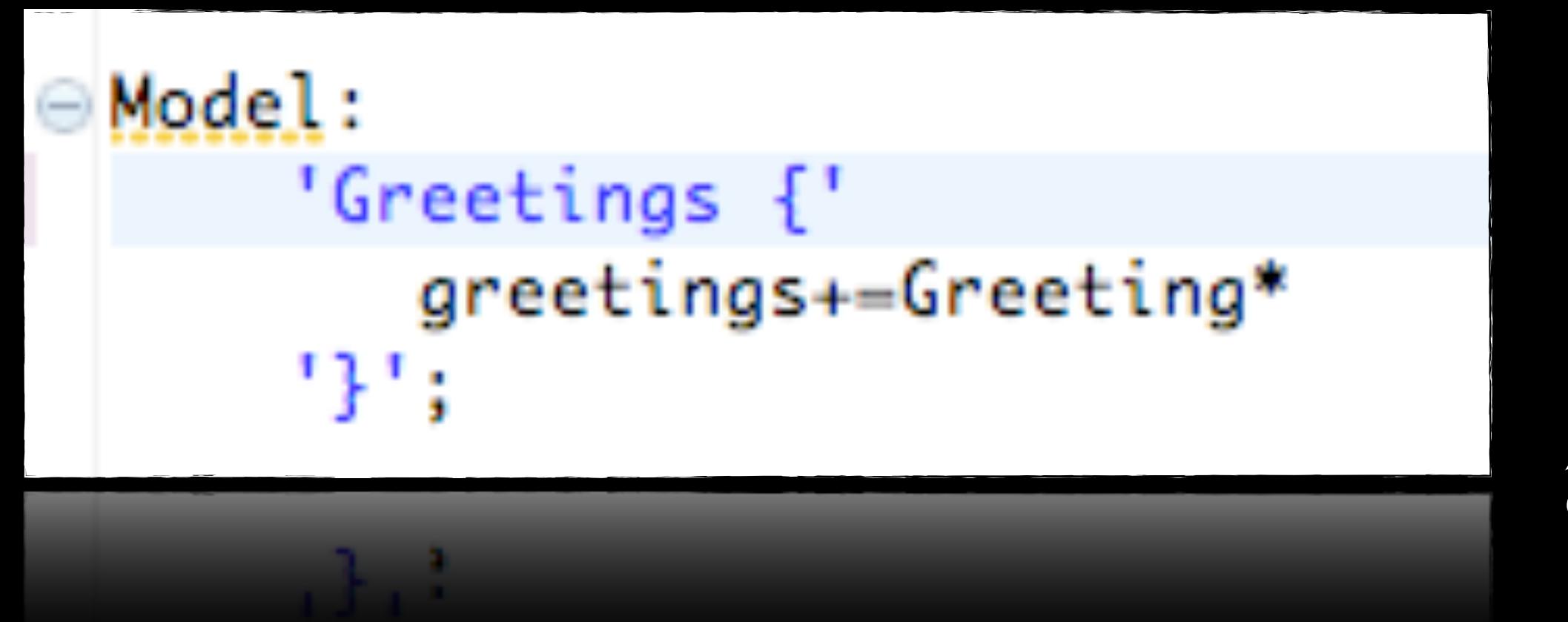
The screenshot shows a code editor window with Java code. The code is as follows:

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

A question mark icon (a small white question mark inside a black circle) is positioned to the right of the code editor window.

Hello,World! (2)

Wie wäre es mit



The screenshot shows a code editor window with a dark theme. A tooltip or code completion dropdown is open over some code. The visible code is:

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

The word "Model" is underlined with a blue squiggly line, indicating a potential error or warning.

SPACES ARE EVIL!

Hello,World! (3)

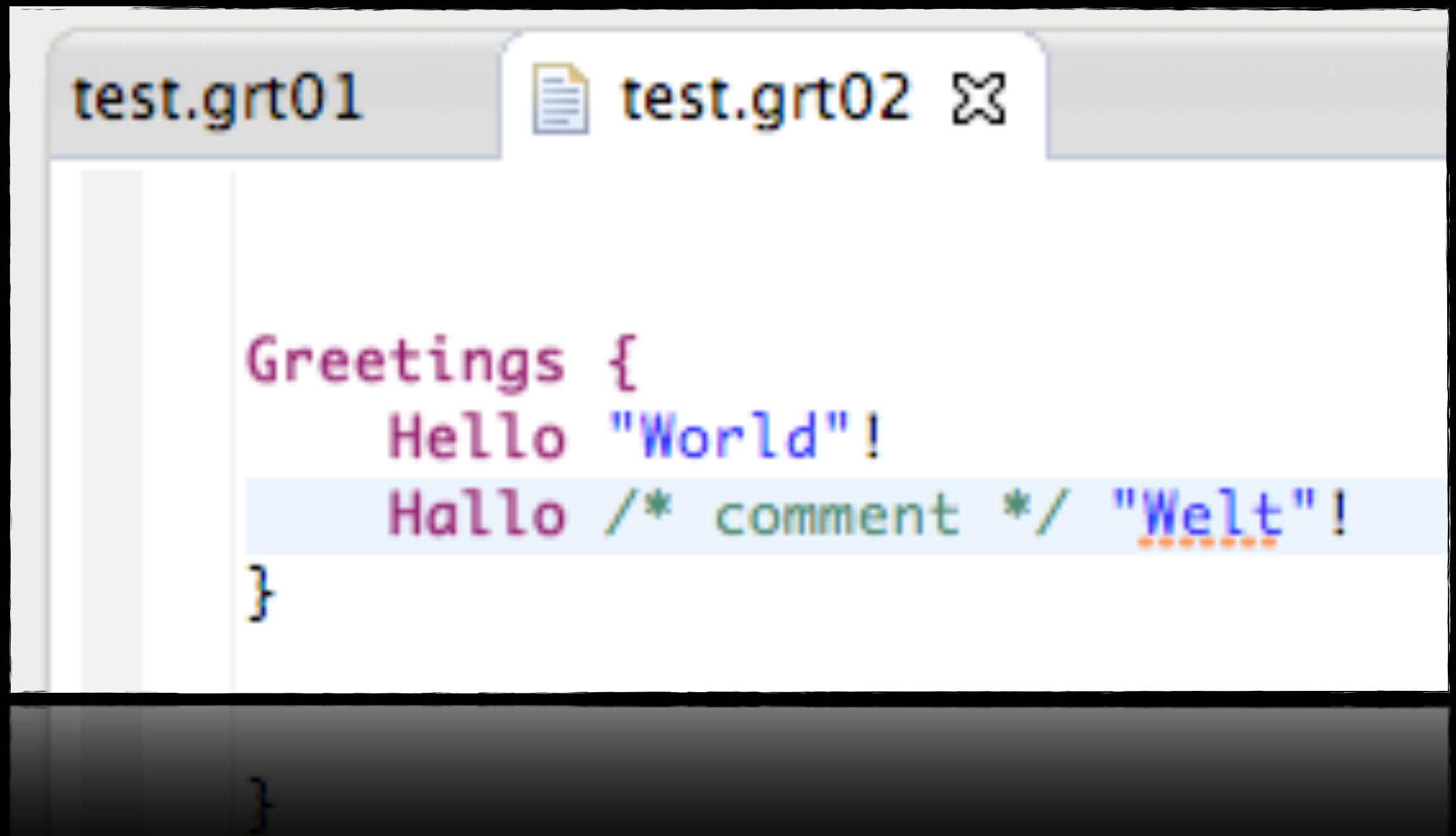
- Alternative Grußtexte
- Eine zusätzliche Rule



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

Hello,World! (4)

- Grußworte als Enum
- Kommentare



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

Agenda | 6.01.2013

- 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 , i,
```

Recap

Model:

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

Greeting:

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

```
:  
(,HELLO, | ,HALLO,) name=ID , i,
```

Recap

Model:

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

Greeting:
('Hello' | 'Hallo') name=**ID** '!'
;

Recap

Model:

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

Greeting:

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

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

Recap

Model:

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

Greeting:

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

2

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;
    }
}

}

}

LEGALU NAME;
```

Recap

Model:

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

Greeting:

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

2

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

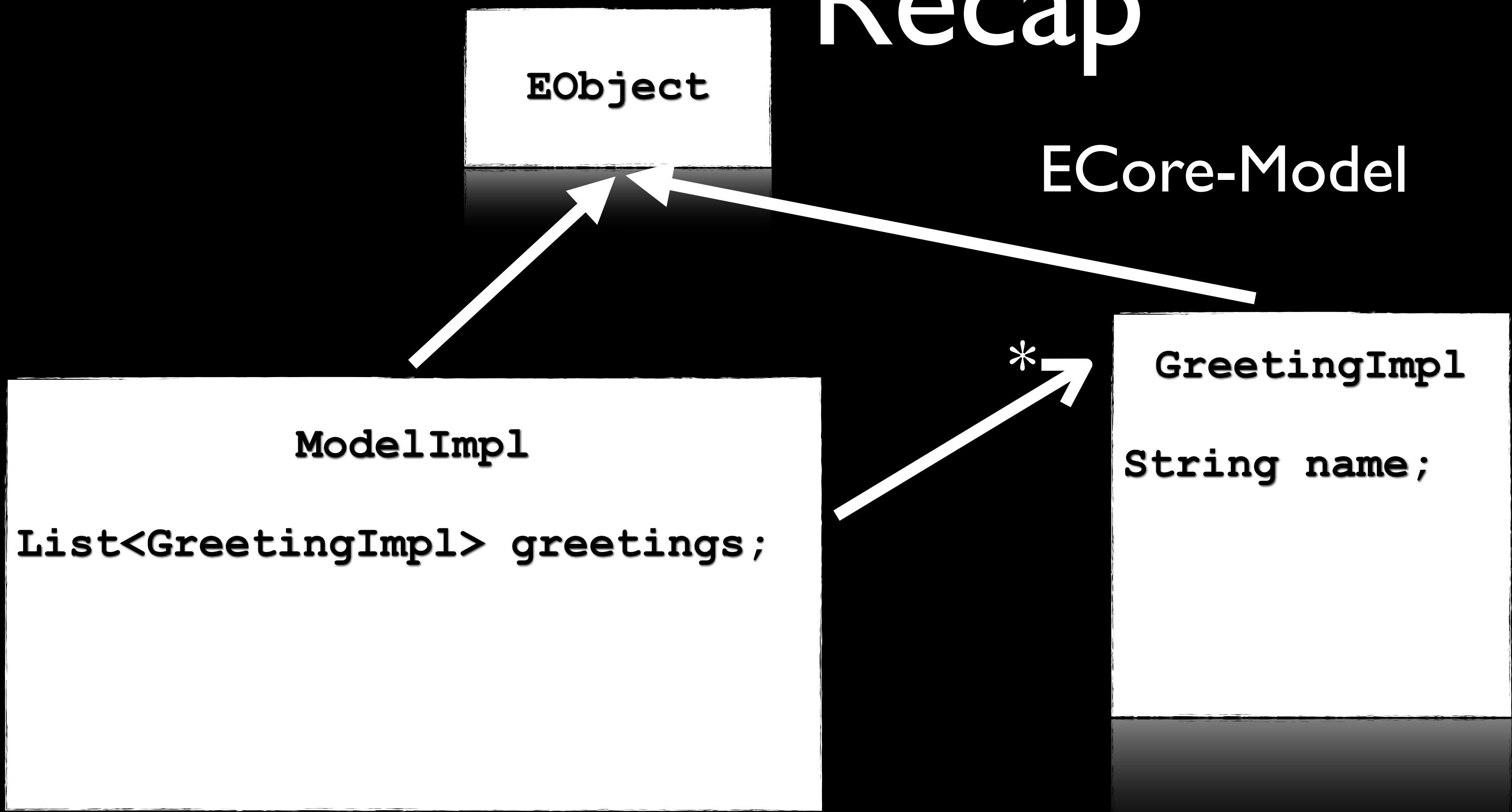
```
public class GreetingImpl extends EClass
{
    protected String name = NAME_EDEFAULT;

    protected GreetingImpl() {
        super();
    }

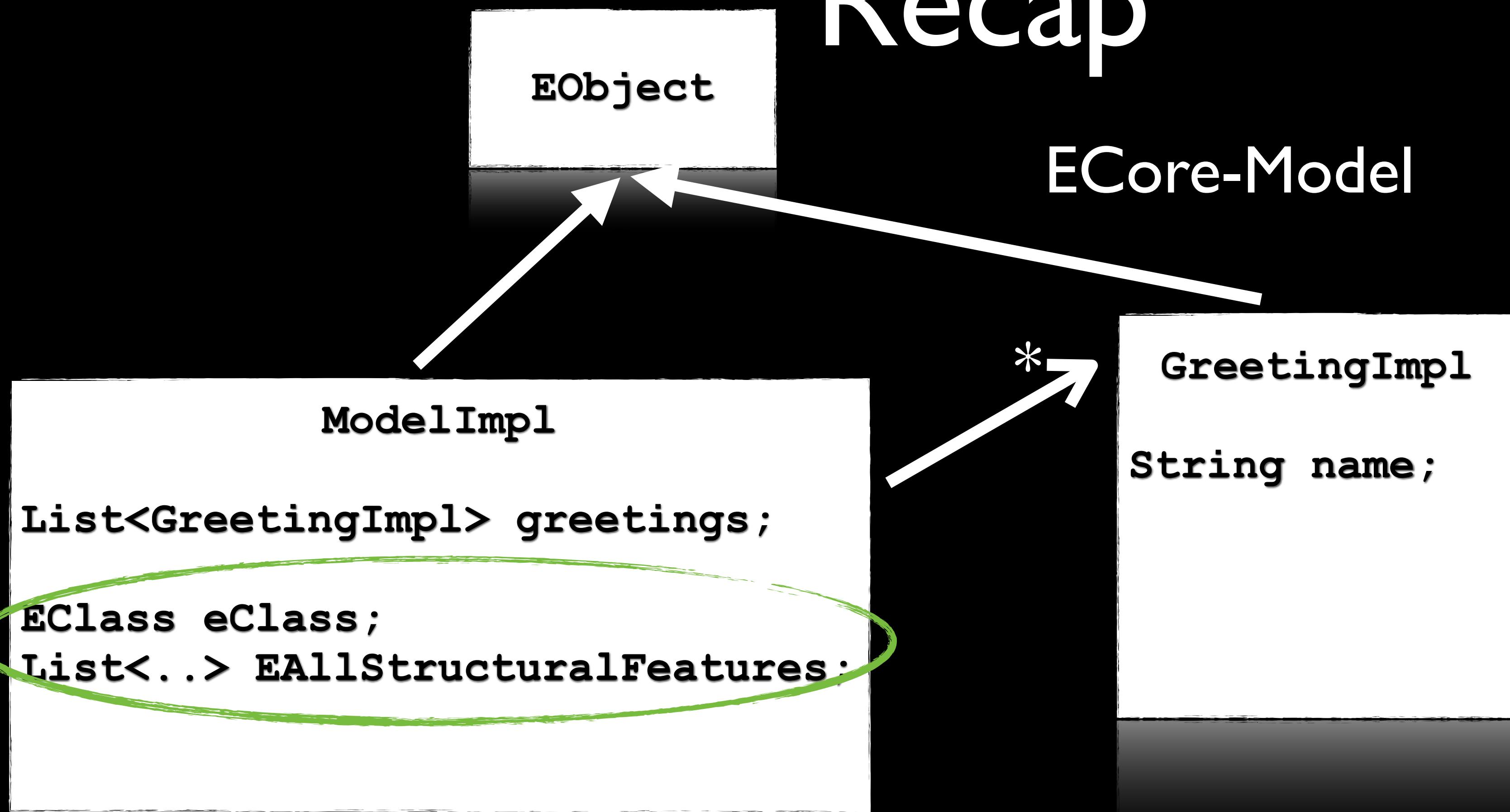
    protected EClass eStaticClass() {
        return GreeterDsl02Package.Literals.GREETING;
    }

    public String getName() {
        return name;
    }
}
```

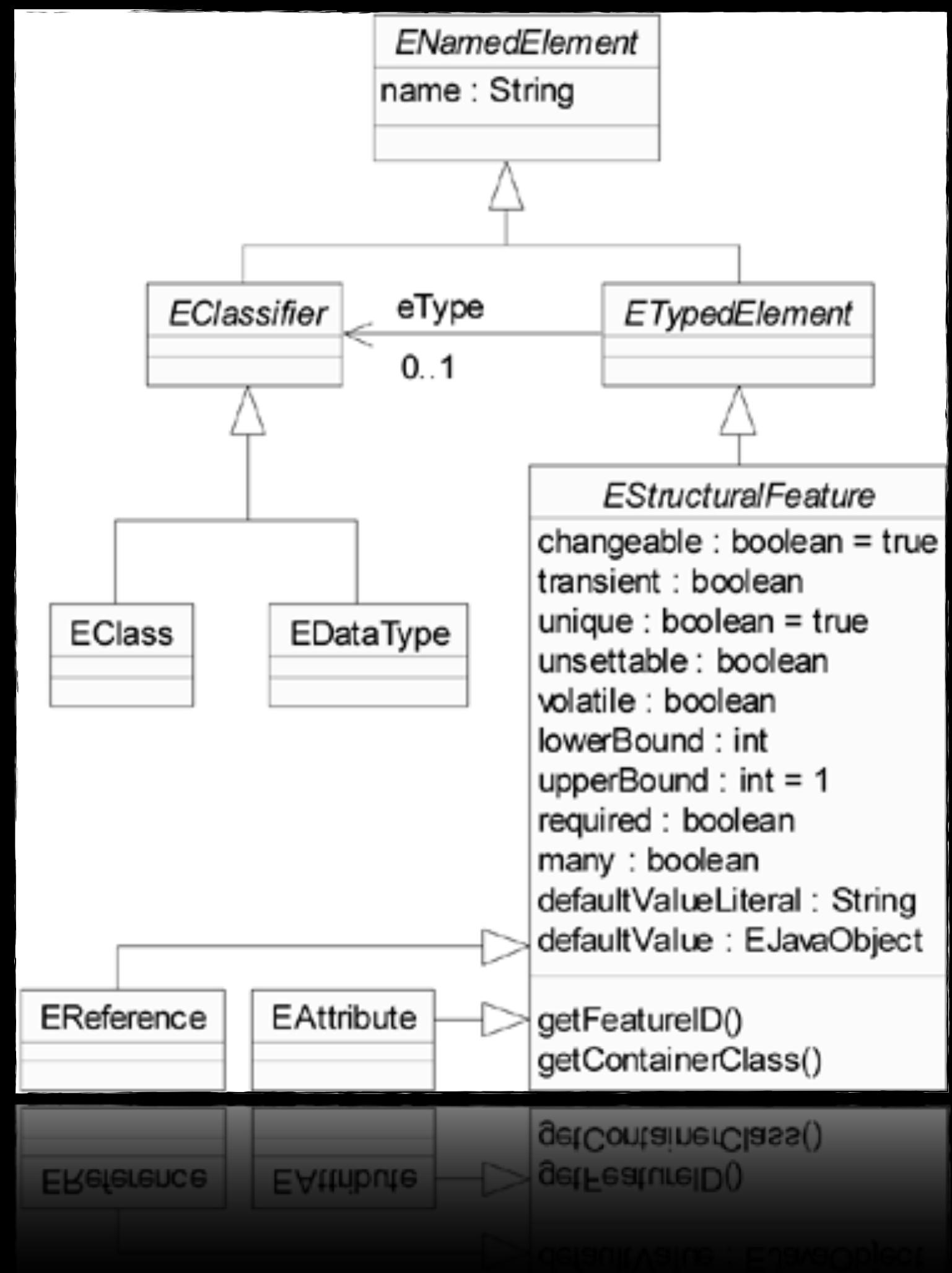
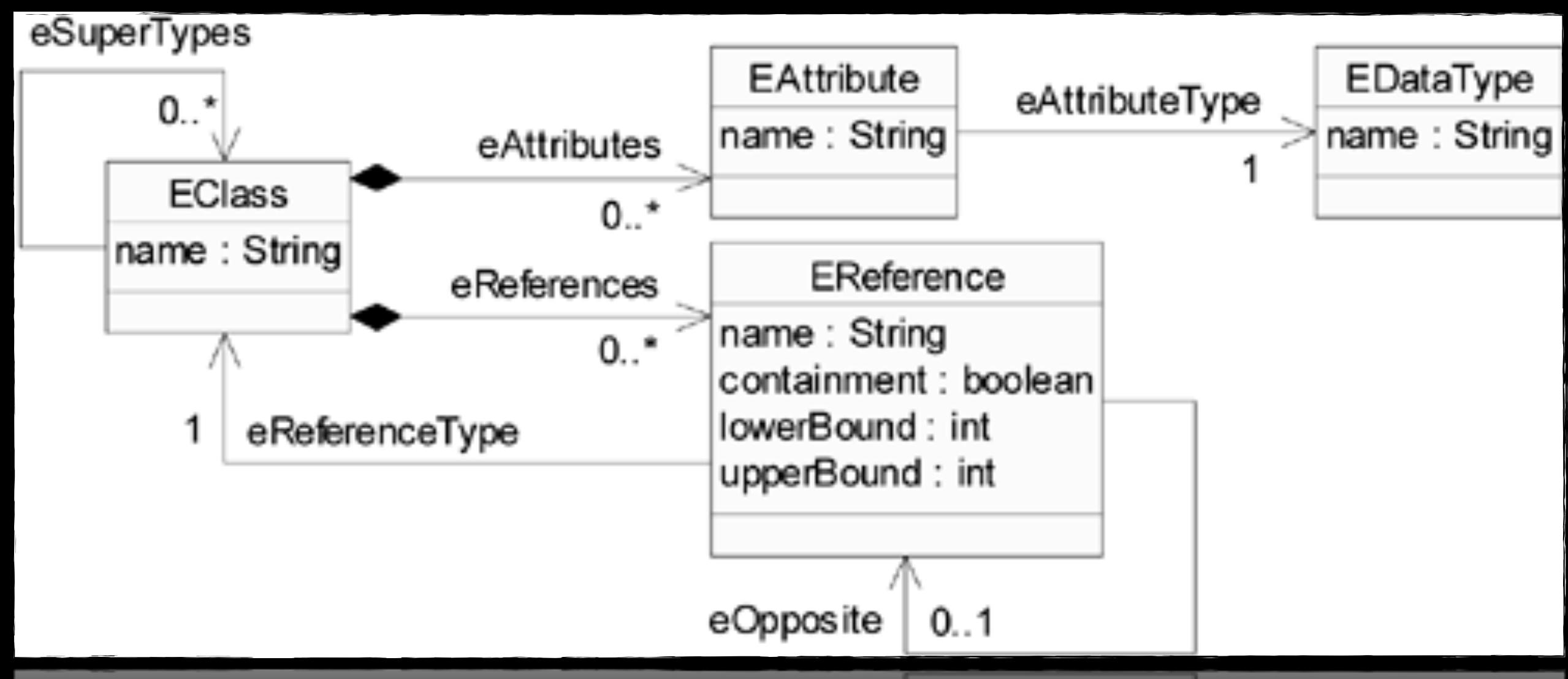
Recap



Recap



ecore Model



Hello,World! (5)

- Personen & Grüße
- Cross-References



The screenshot shows a code editor with two tabs: "test.grt01" and "*test.grt02". The file "test.grt01" contains the following Groovy-like code:

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

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:

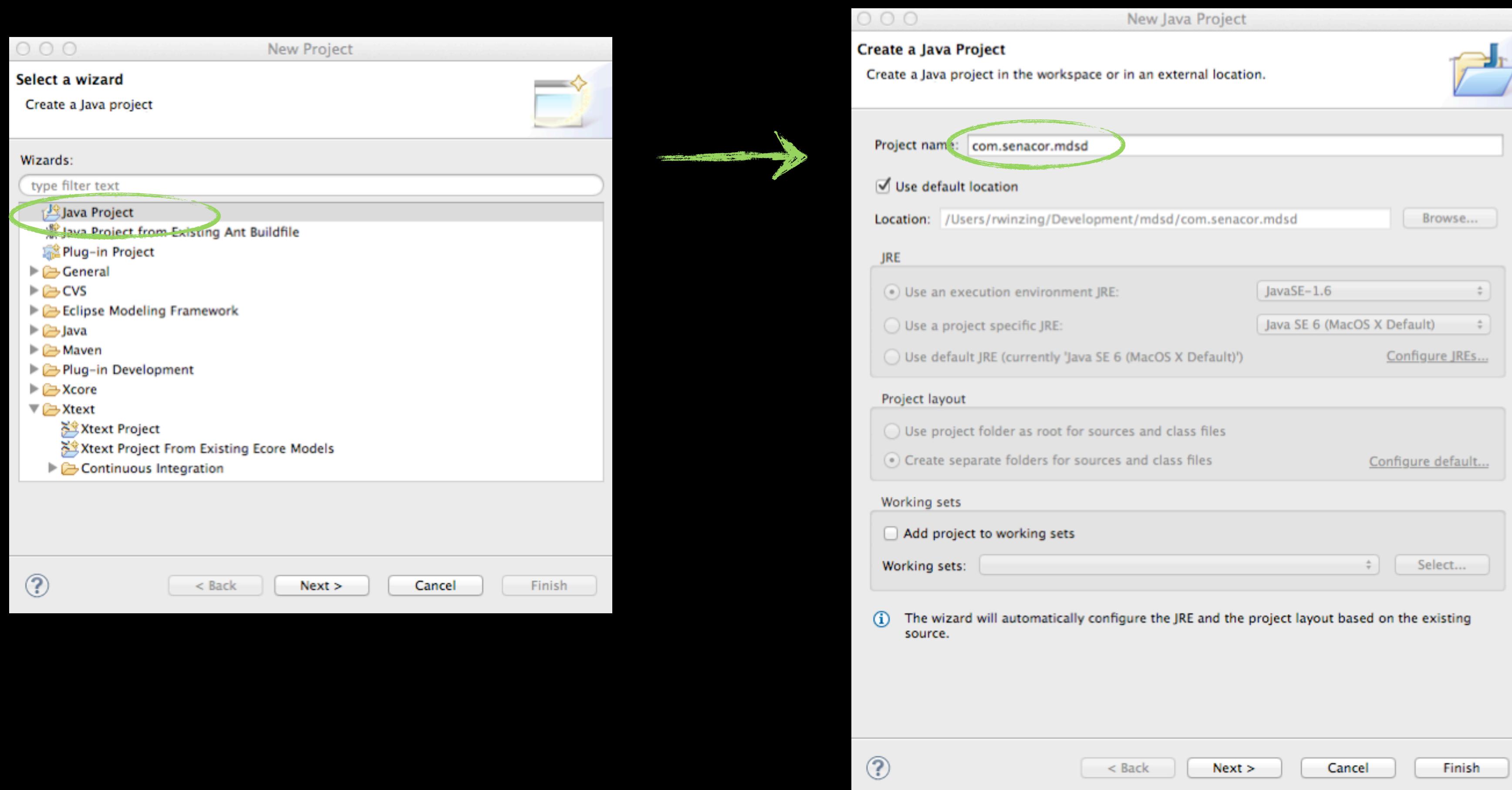
Xtend Basics

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

Hello,World! (wieder mal)

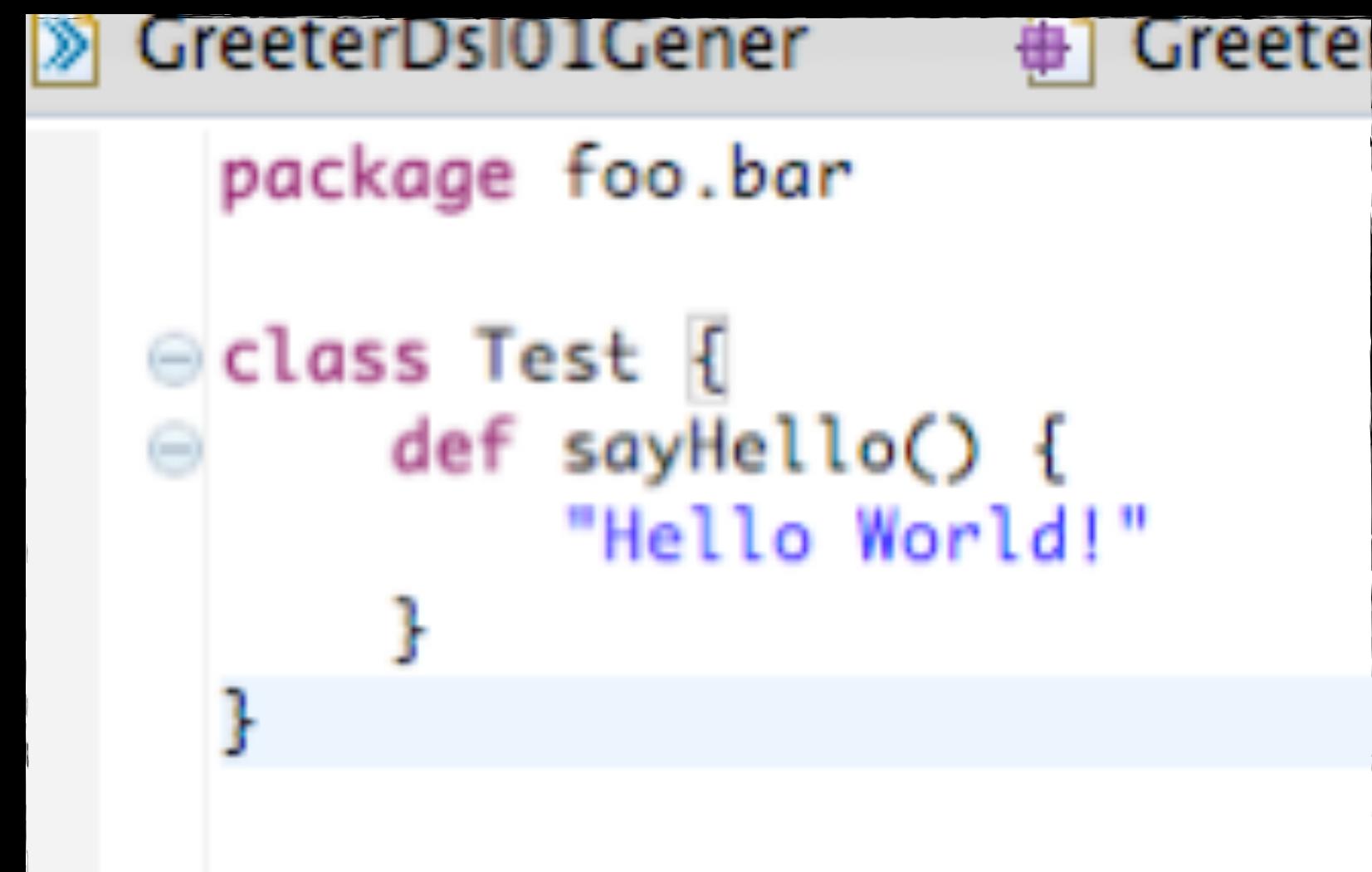
- Projekt erzeugen
- Xtend Klasse anlegen & speichern
- generierte Artefakte ansehen

Ein erstes Xtend Projekt ...



def, var & val

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

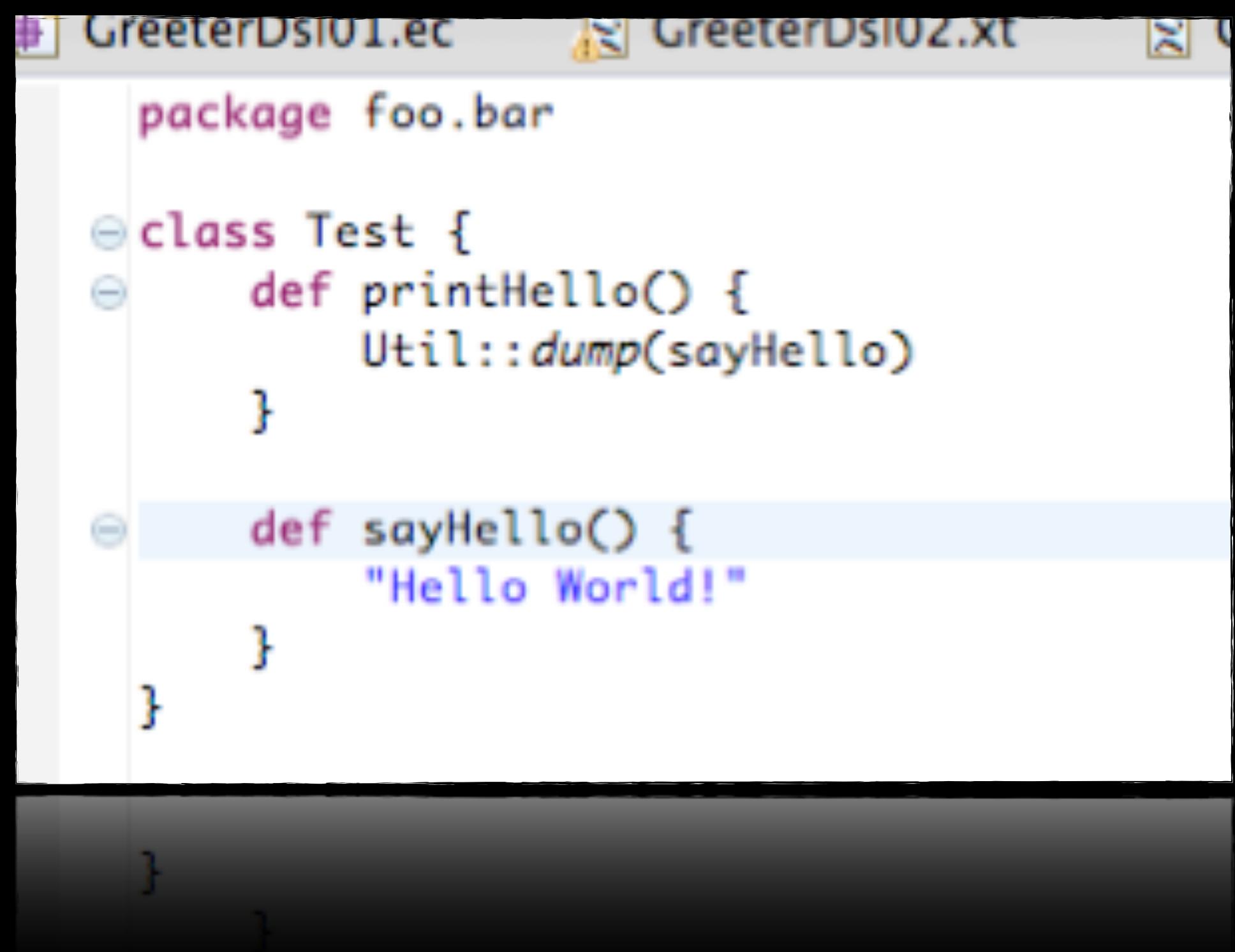


```
GreeterDSL01Gener GreeterDSL01Gener
package foo.bar

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

Transparenter Zugriff zwischen Java & Xtend

- „::“ für Zugriff auf statische Elemente



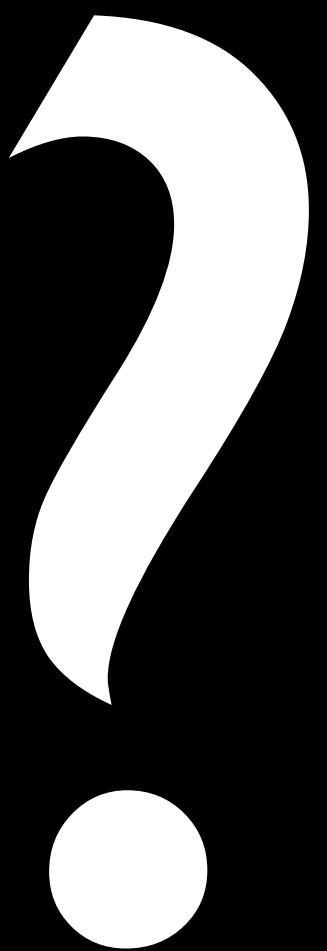
The screenshot shows an IDE interface with two tabs at the top: "GreeterDSL01.ec" and "GreeterDSL02.xt". The "GreeterDSL02.xt" tab is active, displaying Xtend code. The code defines a class "Test" with a static method "printHello" that calls a static method "Util::dump" on the "sayHello" static field. Below it, another static method "sayHello" is defined, returning the string "Hello World!". The code is as follows:

```
package foo.bar

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

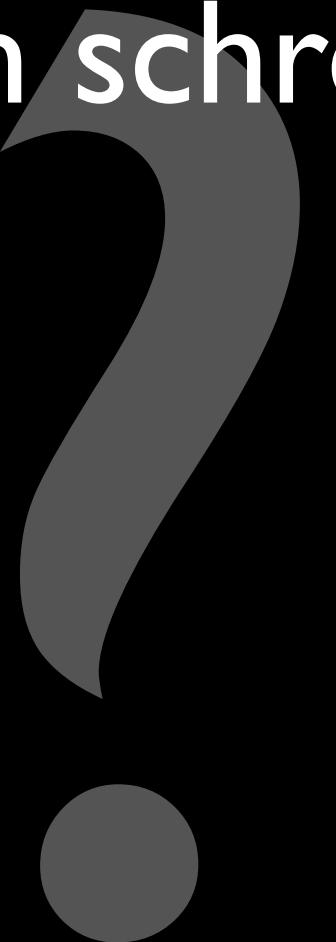
    def sayHello() {
        "Hello World!"
    }
}
```

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, 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!“);  
„hello,  
world!“.dumpToConsole();
```

Weshalb eigentlich „Xtend“?

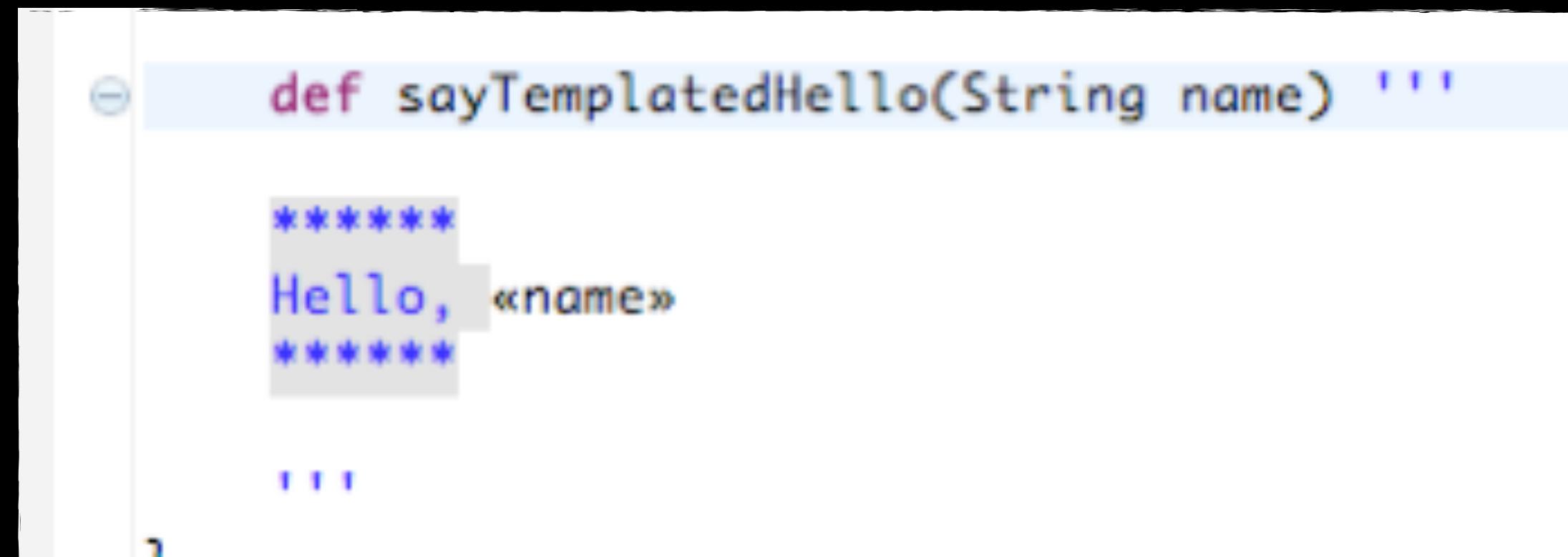
foo(a, b) kann man schreiben als a.foo(b)

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

```
dumpToConsole(„hello, world!“);  
„hel  
worl  
} }  
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»  
    ****  
    ...  
]
```

The word 'Hello' is highlighted with a blue selection bar, indicating it is selected or being edited.

putting stuff together ...

- **xtext und xtend verbinden**
- **und ein wenig generieren**

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>

```
NSDictionary *fields = [[NSMutableDictionary alloc] init];
* savingsTargetID = [NSNumber numberWithInt:[row objectForKey:@"savingsTargetID"]];
* setObject:savingsTargetID forKey:@"savingsTargetID";
* categoryID = [NSNumber numberWithInt:[row objectForKey:@"categoryID"]];
* setObject:categoryID forKey:@"categoryID";
* parentGoalID = [NSNumber numberWithInt:[row objectForKey:@"parentGoalID"]];
* setObject:parentGoalID forKey:@"parentGoalID";
* name = [results stringForColumn:@"name" row:[row index]];
* setObject:name forKey:@"name";
* color = [results stringForColumn:@"color" row:[row index]];
* setObject:color forKey:@"color";
* saveAmount = [NSNumber numberWithInt:[row objectForKey:@"saveAmount"]];
* setObject:saveAmount forKey:@"saveAmount";
```