

Modellgetriebene Softwareentwicklung mit

Xtext



Wer ich bin



Christoph Knauf

» [xing.com/profile/Christoph_Knauf2](https://www.xing.com/profile/Christoph_Knauf2)

» [@C_Knauf](https://twitter.com/C_Knauf)

» christoph.knauf@itemis.de

Wer ist itemis?

Wer ist itemis?

Gegründet 2003

Wer ist itemis?

Gegründet 2003

Gründergeführt

Wer ist itemis?

200 Mitarbeiter

Gegründet 2003
Gründergeführt

Wer ist itemis?

200 Mitarbeiter

Gegründet 2003

Gründergeführt

Strategisches Mitglied der Eclipse Foundation

Wer ist itemis?

200 Mitarbeiter

Gegründet 2003

Gründergeführt

Strategisches Mitglied der Eclipse Foundation

Stark Involviert in Open-Source

Wer ist itemis?

200 Mitarbeiter

Gegründet 2003

Gründergeführt

Strategisches Mitglied der Eclipse Foundation

Stark Involviert in Open-Source

4+1

Wer ist itemis?

200 Mitarbeiter

Gegründet 2003

Gründergeführt

Strategisches Mitglied der Eclipse Foundation

Stark Involviert in Open-Source

4+1

9 Standorte national + Zürich und Paris



Agenda

- Grundlagen
 - Was ist Modellgetriebene Softwareentwicklung?
 - Was sind Domänenspezifische Sprachen?
- Xtext
 - Was ist Xtext?
 - Aufbau und Funktionsweise
 - Beispielprojekte

Modellgetriebene Softwareentwicklung (MDSD)

Modellgetriebene Softwareentwicklung (MDSD)

Definition

Das Ziel: Aus formalen Modellen automatisiert lauffähige Software generieren*

Modell: Repräsentation eines Teilaspekts eines Software-Systems

Formal: Modell ist **vollständig und eindeutig** beschrieben

- mittels mindestens einer **konkreten Syntax (KS)** und
- ist Instanz eines **Metamodells**.

Automatisierte Generierung: **Maschinelle Transformation**

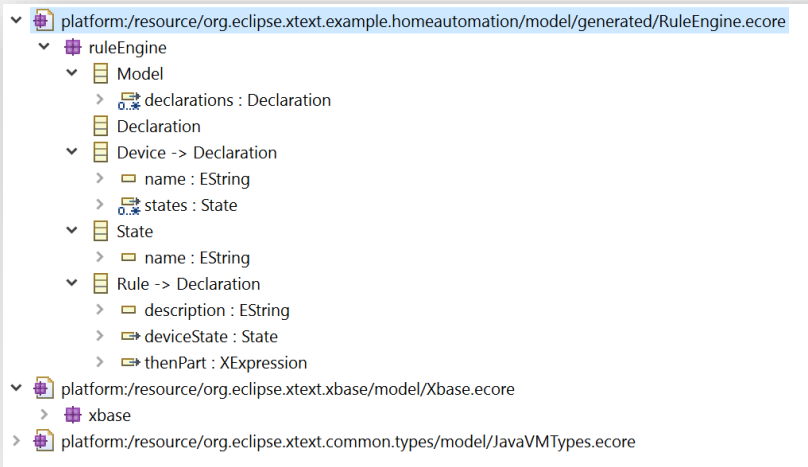
- Modell ist gleichbedeutend mit Programmcode

* Stahl, Völter, Effttinge, Haase. Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt.verlag, 2007

Modellgetriebene Softwareentwicklung (MDSD)

Konkrete Syntax

- Grafische oder Textuelle Repräsentation des Modells
- Ein Modell kann mehrere Repräsentationen haben
- Erlaubt das Editieren des Modells
- Muss benutzergerecht sein (Usability)



```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
name="ruleEngine" nsURI="http://www.eclipse.org/Xtext/example/RuleEngine"
nsPrefix="ruleEngine">
<eClassifiers xsi:type="ecore:EClass" name="Model">
<eStructuralFeatures xsi:type="ecore:EReference" name="declarations" upperBound="1"
eType="#//Declaration" containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Declaration"/>
<eClassifiers xsi:type="ecore:EClass" name="Device" eSuperTypes="#//Declaration">
<eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#//EString"/>
<eStructuralFeatures xsi:type="ecore:EReference" name="states" upperBound="-1"
eType="#//State" containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="State">
<eClassifiers xsi:type="ecore:EClass" name="Rule" eSuperTypes="#//Declaration">
<eStructuralFeatures xsi:type="ecore:EAttribute" name="description"
eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#//EString"/>
```

Modellgetriebene Softwareentwicklung (MDSD)

Metamodelle aka Abstrakte Syntax

- Jedes formale Modell ist Instanz eines Metamodells
- Beschreibt die Struktur und die Konstrukte des formalen Modells
- Interpreter und Generatoren arbeiten auf dem Metamodell

Beispiele:

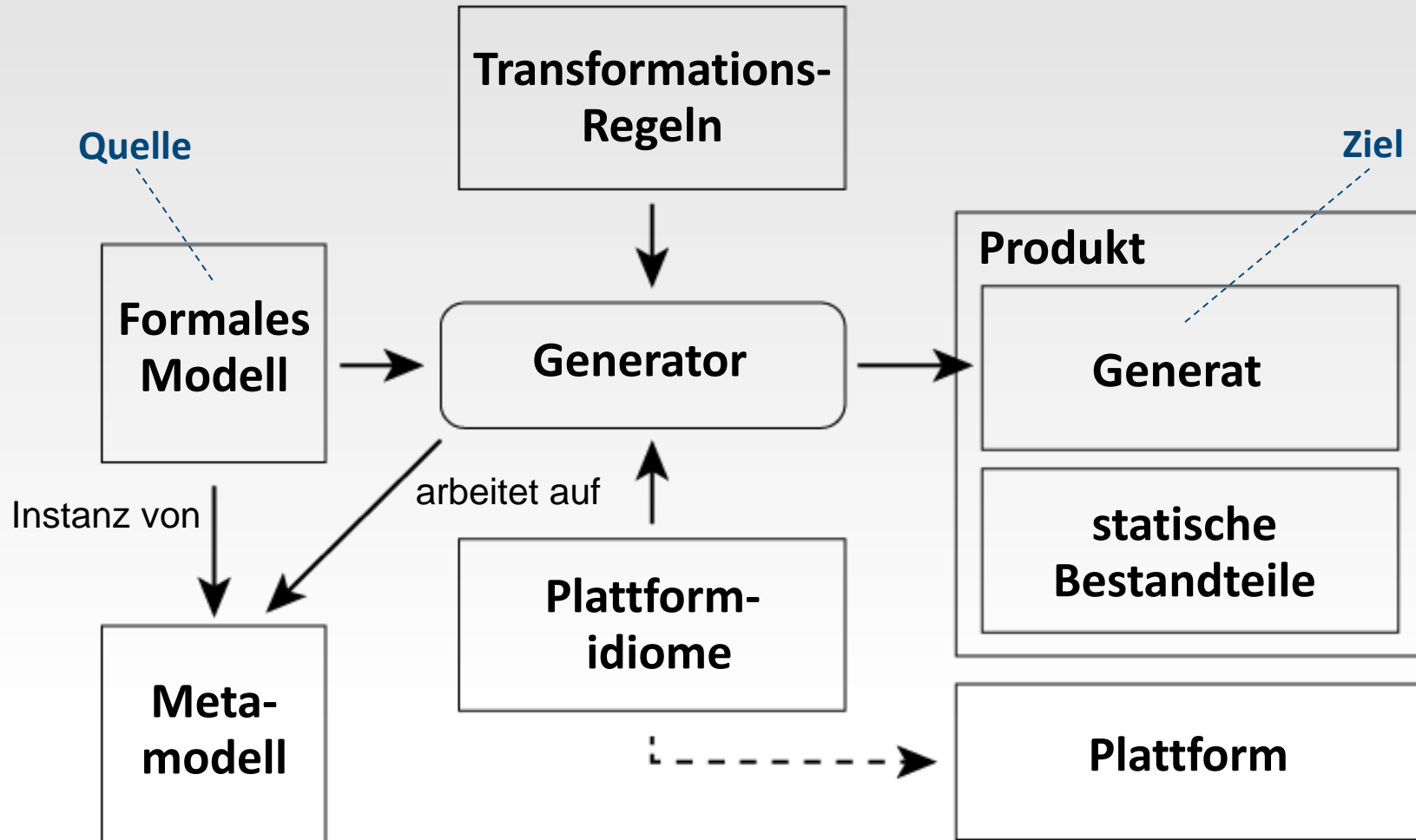
XML-Schema ←^{Instanz von} XML Dokument

Kontextfreie Grammatik (BNF) ←^{Instanz von} Programm der Sprache

UML Klassendiagramm ←^{Instanz von} UML Objektdiagramm

Modellgetriebene Softwareentwicklung (MDSD)

Generierung



Modellgetriebene Softwareentwicklung (MDSD)

Was kann alles Generiert werden?

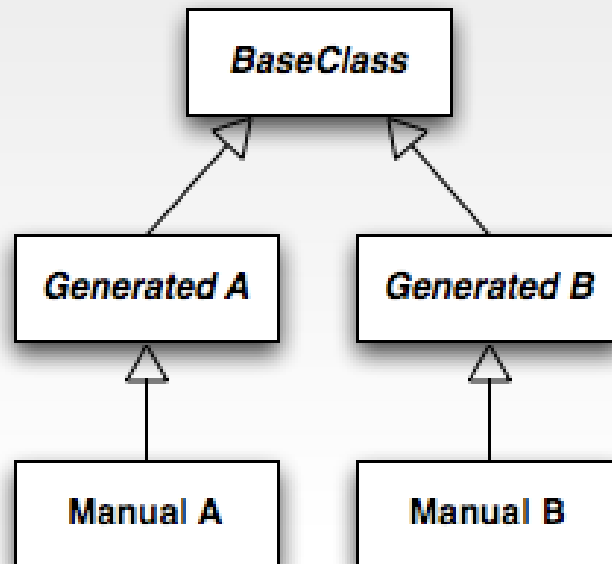
- Datenbankschemata
- Benutzerschnittstellen
- Teile der Anwendungslogik
- Datenobjekte
- Wrapper
- Dokumentation
- Konfigurationen (Spring, Hibernate, Maven, Gradle ...)
- Tests (Unit-Tests, Mocks, Lasttests, ...)
- Bibliotheken

...

Modellgetriebene Softwareentwicklung (MDSD)

Erweiterbarkeit generierten Codes

- Generation gap pattern



BaseClass
Belongs to common framework code and is manually implemented.

Generated A
Abstract class introduces behavior and methods specific to the model. It will be overwritten each time the generator produces artifacts.

Manual A
Concrete class allows manual extension of generated code.

Quelle: <http://heikobehrens.net>

Modellgetriebene Softwareentwicklung (MDSD)

Vorteile

Erhöhter Abstraktionsgrad

- Gemeinsame Sprache für Entwickler und Fachexperten
- Trennung von fachlichen und technischen Aspekten
- Wiederverwendung
- Einfacherer Austausch von Technologien im Generat

Höhere Softwarequalität

- Standardisierung durch Generierung (keine Flüchtigkeitsfehler)
- Einheitliche Architektur durch Abbildungsregeln
- Tritt ein Fehler auf spiegelt er sich im Generat an allen Stellen wieder an denen die Transformationsregel greift

Modellgetriebene Softwareentwicklung (MDSD)

Vorteile

Erhöhte Entwicklungsgeschwindigkeit

- Generierung befreit von Boilerplate-Code
- Geringere Reaktionszeit bei Change-Requests (Anpassbarkeit)

Konsistenz der Anwendung

- Plattformidiome und Konventionen sind Teil der Generierung
 - Generierter Code ist konsistent bezüglich Aufrufkonventionen, Benamung, Parameterübergabe etc.
 - Dadurch potentiell leichtere Verständlichkeit im Vergleich zu handgeschriebenen Code

Domänenspezifische Sprachen (DSLs)

Domänenspezifische Sprachen (DSL)

Definition

A computer **programming language** of **limited expressiveness**
focused on a **particular domain***

- Möglichkeit zur **Umsetzung von MDSD**
- Sprachen zur Lösung von Problemen in einer bestimmten **Domäne**
- Differenzierbar in verschiedene Arten

Domänenspezifische Sprachen (DSL)

Arten von DSLs

Interne DSLs

- Implementiert in einer bestehenden Hostsprache
- Wiederverwendung bestehender Tools der Hostsprache
- Beispiele: UML2-Profile, JUnit

Externe DSLs

- Sprache mit unabhängiger Konkreter und Abstrakter-Syntax
- Werkzeuge müssen selbst bereitgestellt werden
- Spezifischere Werkzeuge → potentiell bessere User Experience
- Beispiele: SQL, RegEx

Grafische DSLs (Modellierungssprachen) vs. Textuelle DSLs

Domänenspezifische Sprachen (DSL)

Einsatzbereiche

Legacy Code

- Editorsupport für bestehende Sprachen
- Migration von Altsystem
 - Aspekte extrahieren und in DSL abbilden
 - Erhöht Wartbarkeit für die Zukunft

Standardisierung

- Wiederholung mit gleichem Muster
- Varianz im Generat muss gegeben sein
- Unterschiedliche Aspekte eines Systems (z.B. Daten+Queries)

Domänenspezifische Sprachen (DSL)

Einsatzbereiche

Artefakte für unterschiedl. Programmiersprachen und Plattformen

- Konsistente Funktionalität über Sprach/Plattformgrenzen
- z.B. Messaging/Eventing

In allen Projektphasen verwendbar

- Anforderungsanalyse
- Design und Entwicklung
- Testing

Domänenspezifische Sprachen (DSL)

Bestandteile

Domänenspezifische Sprachen (DSL)

Bestandteile

Lexer

Parser

Domänenspezifische Sprachen (DSL)

Bestandteile

Lexer

Parser

Abstract Syntax Tree (AST)

Domänenspezifische Sprachen (DSL)

Bestandteile

Parser

Lexer

Abstract Syntax Tree (AST)

Linker

Domänenspezifische Sprachen (DSL)

Bestandteile

Parser

Lexer

Abstract Syntax Tree (AST)

Linker

Validierung

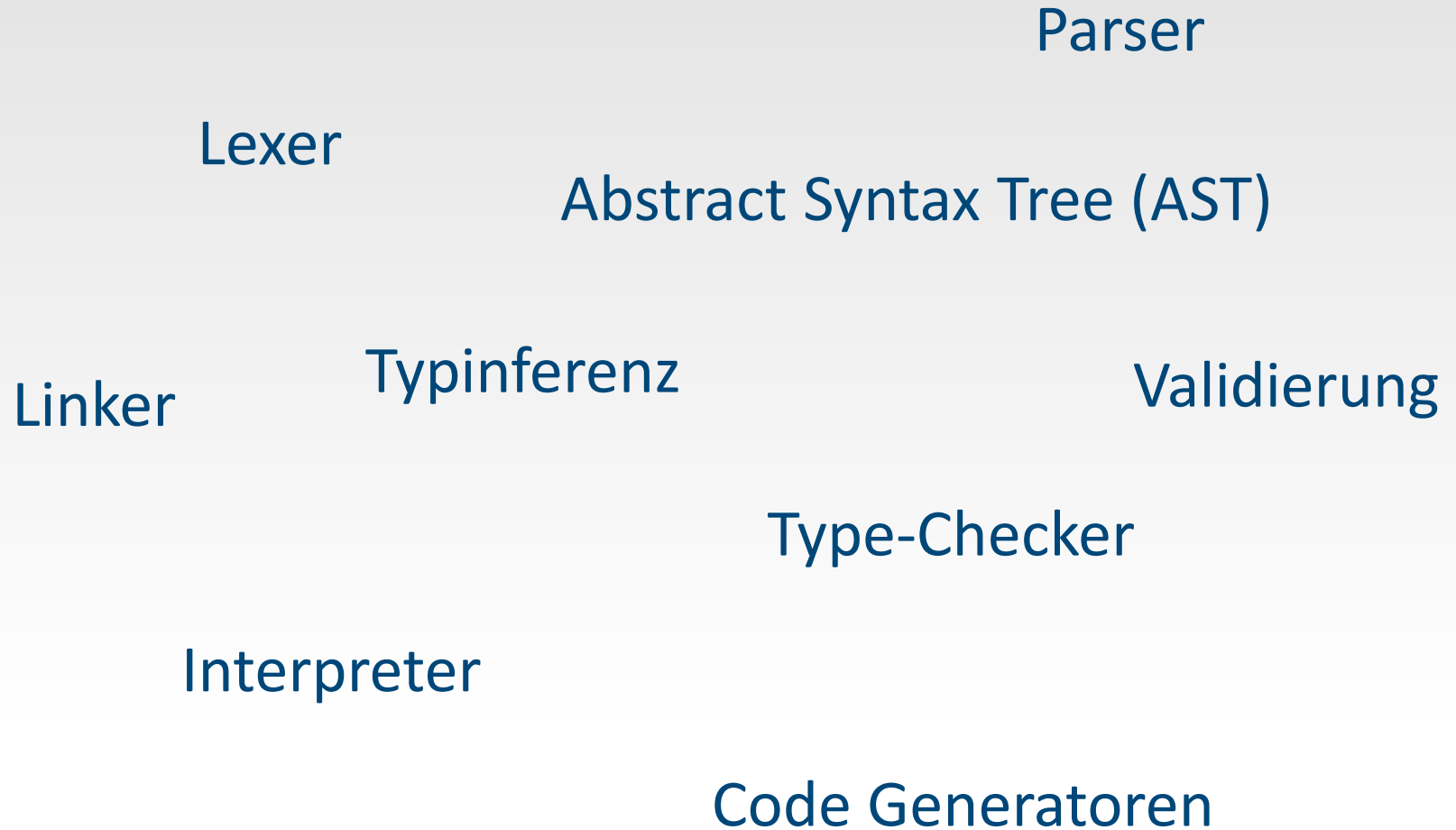
Domänenspezifische Sprachen (DSL)

Bestandteile



Domänenspezifische Sprachen (DSL)

Bestandteile



Domänenspezifische Sprachen (DSL)

Benutzerunterstützung

Refactoring

GoTo Deklarationen

Outline View

Code Folding

Syntax Coloring

Wizards

Hovers

Find References

Quick Fixes

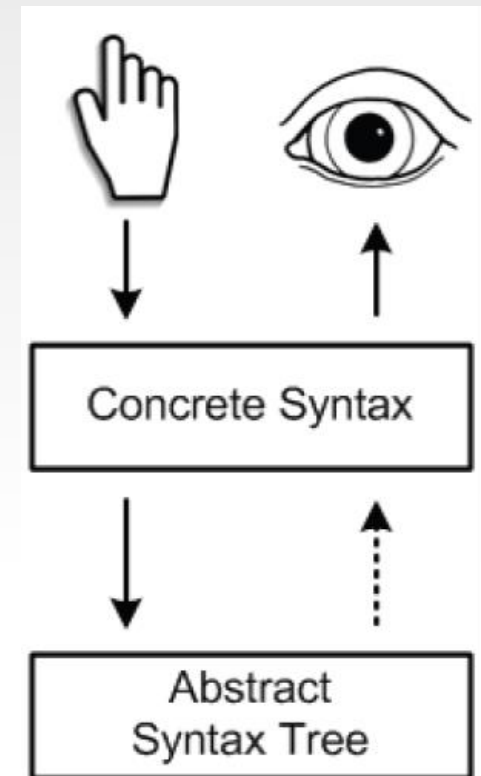
Content Assist

Templates & Proposals

Xtext

Xtext

- Seit 2006 von itemis entwickeltes Framework
- Basiert auf OSGi/Eclipse
 - Ist Teil des Eclipse Modeling Project
 - Ab 2008 Eclipse Project
 - Open-Source (EPL)
- Textuelle DSLs
- Parserbasiert
 - Grammatik ist zentraler Bestandteile



Xtext

Demo

Xtext - Aufbau und Runtime



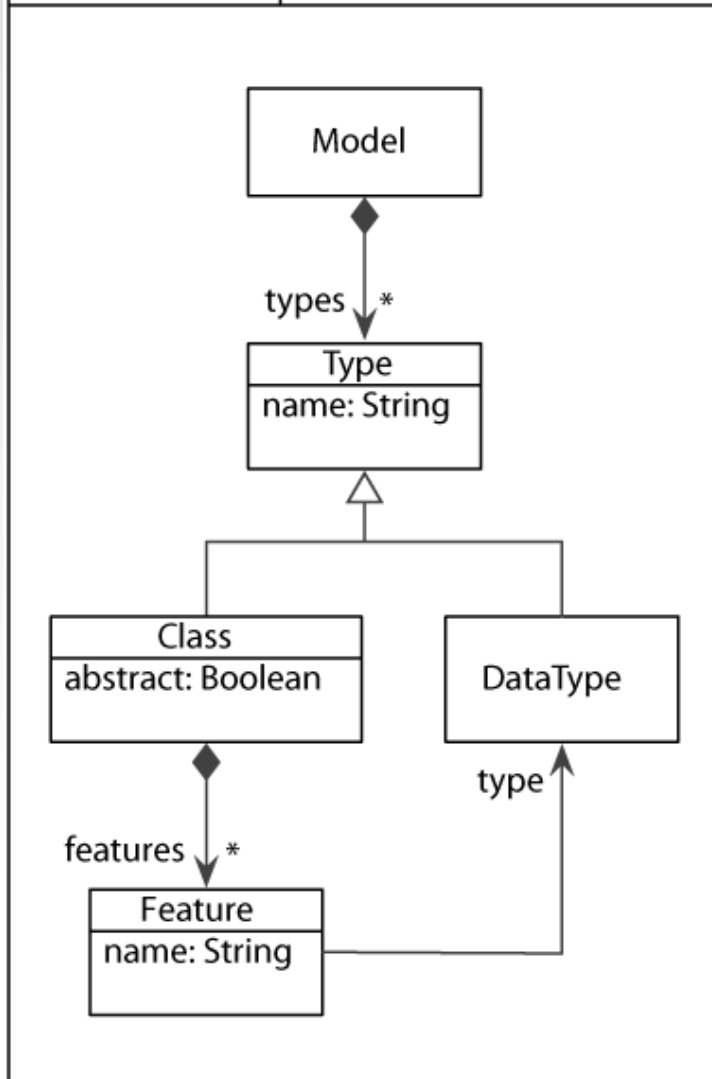
**Fachliches
Modell**

Xtext - Aufbau und Runtime



Grammatik

domainModel



```
grammar org.xtext.workshop.DomainModel
    with org.eclipse.xtext.common.Terminals
```

```
generate domainModel
    " http://www.xtext.org/workshop/DomainModel "
```

```
Model:
    (types+=Type)+;
```

```
Type:
    Class | DataType;
```

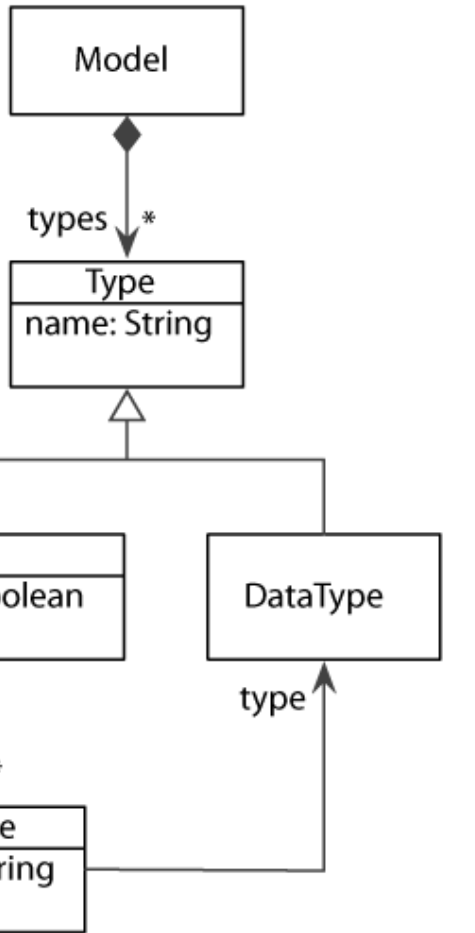
```
DataType:
    'datatype' name=ID;
```

```
Class:
    (abstract?= 'abstract')? 'class' name=ID '{'
    (features+=Attribute)*
    '}' ;
```

```
Attribute returns Feature:
    'attr' name=ID ':' type=[DataType];
```

Grammatik

domainModel



grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate domainModel
" <http://www.xtext.org/workshop/DomainModel> "

Model:
(types+=Type)+;

Type:
Class | DataType;

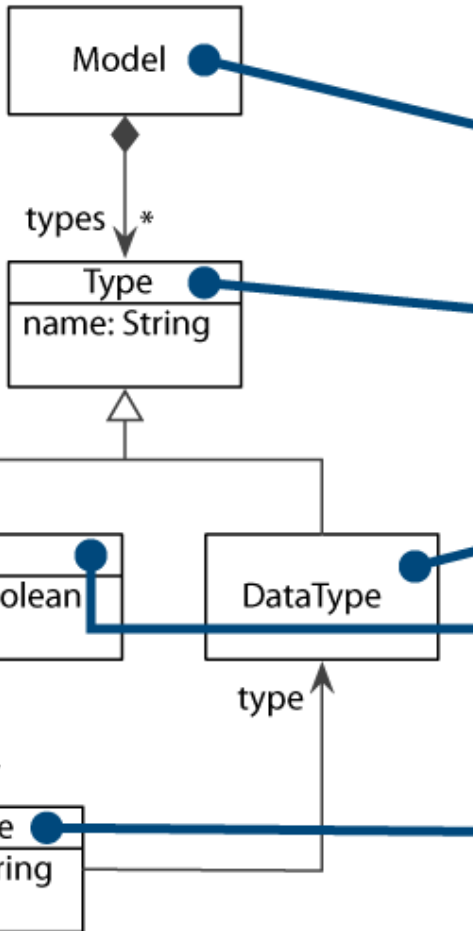
DataType:
'datatype' name=ID;

Class:
(abstract?='abstract')? 'class' name=ID '{'
(features+=Attribute)*
'}' ;

Attribute returns Feature:
'attr' name=ID ':' type=[DataType];

Grammatik

domainModel



grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate domainModel
" <http://www.xtext.org/workshop/DomainModel> "

Model:
(types+=Type)+;

Type:
Class | DataType;

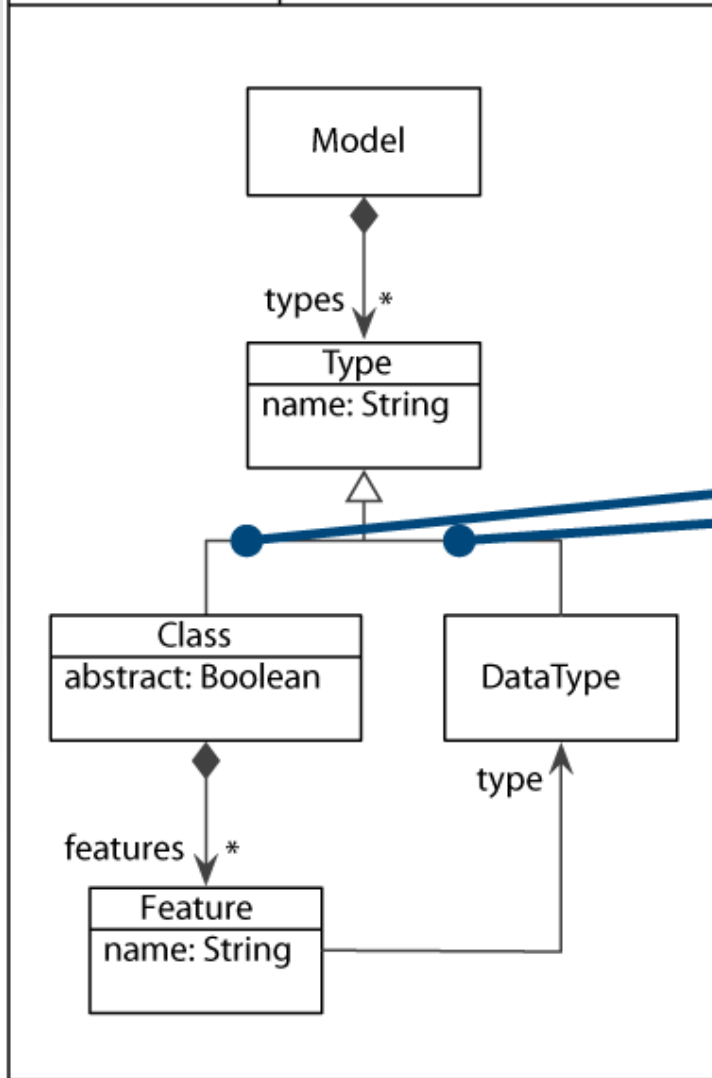
DataType:
'datatype' name=ID;

Class:
(abstract?='abstract')? 'class' name=ID '{'
(features+=Attribute)*
'}' ;

Attribute returns Feature:
'attr' name=ID ':' type=[DataType];

Grammatik

domainModel



```
grammar org.xtext.workshop.DomainModel
    with org.eclipse.xtext.common.Terminals
```

```
generate domainModel
    " http://www.xtext.org/workshop/DomainModel "
```

```
Model:
    (types+=Type)+;
```

```
Type:
    Class | DataType;
```

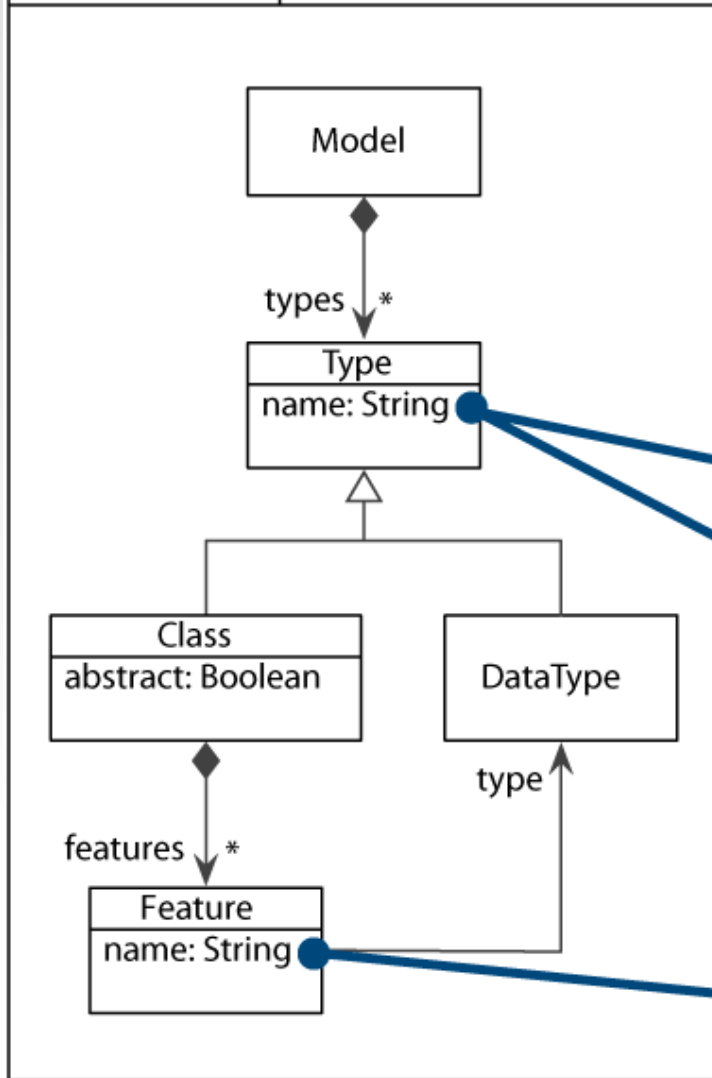
```
DataType:
    'datatype' name=ID;
```

```
Class:
    (abstract?='abstract')? 'class' name=ID '{'
    (features+=Attribute)*
    '}' ;
```

```
Attribute returns Feature:
    'attr' name=ID ':' type=[DataType];
```

Grammatik

domainModel



```
grammar org.xtext.workshop.DomainModel
    with org.eclipse.xtext.common.Terminals
```

```
generate domainModel
    " http://www.xtext.org/workshop/DomainModel "
```

```
Model:
    (types+=Type)+;
```

```
Type:
    Class | DataType;
```

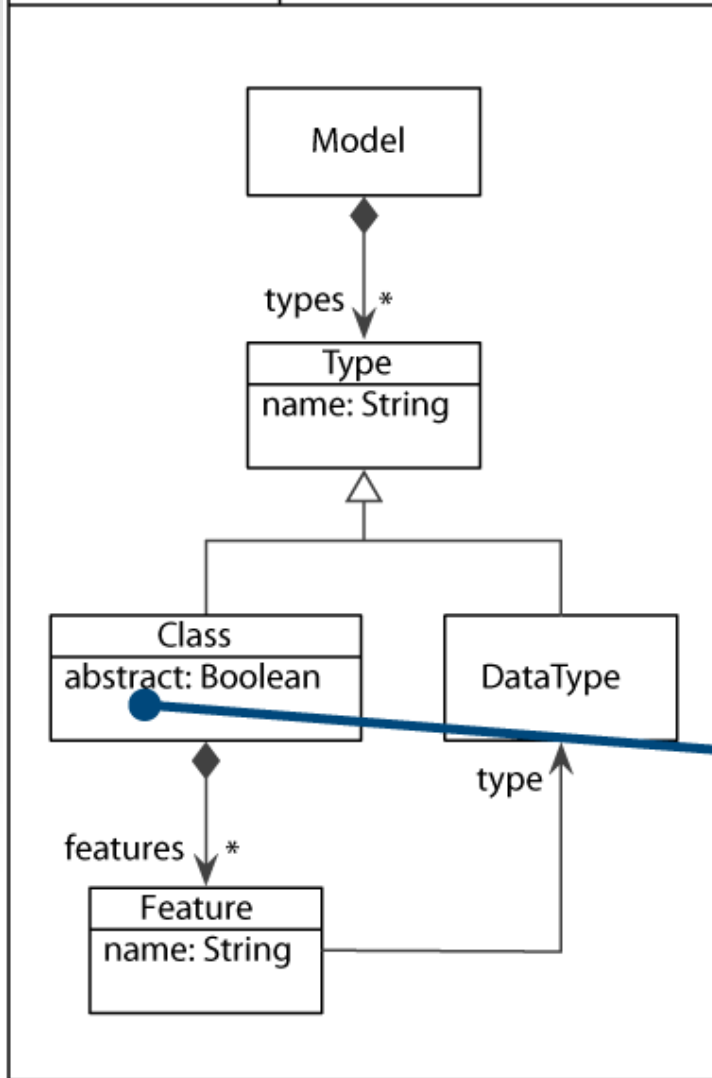
```
DataType:
    'datatype' name=ID;
```

```
Class:
    (abstract?='abstract')? 'class' name=ID '{'
    (features+=Attribute)*
    '}' ;
```

```
Attribute returns Feature:
    'attr' name=ID ':' type=[DataType];
```

Grammatik

domainModel



```
grammar org.xtext.workshop.DomainModel
    with org.eclipse.xtext.common.Terminals
```

```
generate domainModel
    " http://www.xtext.org/workshop/DomainModel "
```

```
Model:
    (types+=Type)+;
```

```
Type:
    Class | DataType;
```

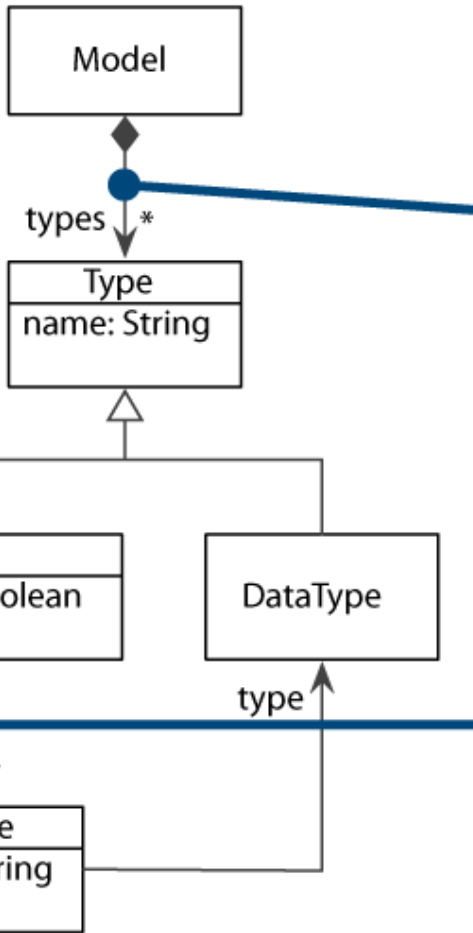
```
DataType:
    'datatype' name=ID;
```

```
Class:
    (abstract?='abstract')? 'class' name=ID '{'
    (features+=Attribute)*
    '}' ;
```

```
Attribute returns Feature:
    'attr' name=ID ':' type=[DataType];
```

Grammatik

domainModel



grammar org.xtext.workshop.DomainModel
with org.eclipse.xtext.common.Terminals

generate domainModel
" <http://www.xtext.org/workshop/DomainModel> "

Model:
● (types+=Type)+;

Type:
Class | DataType;

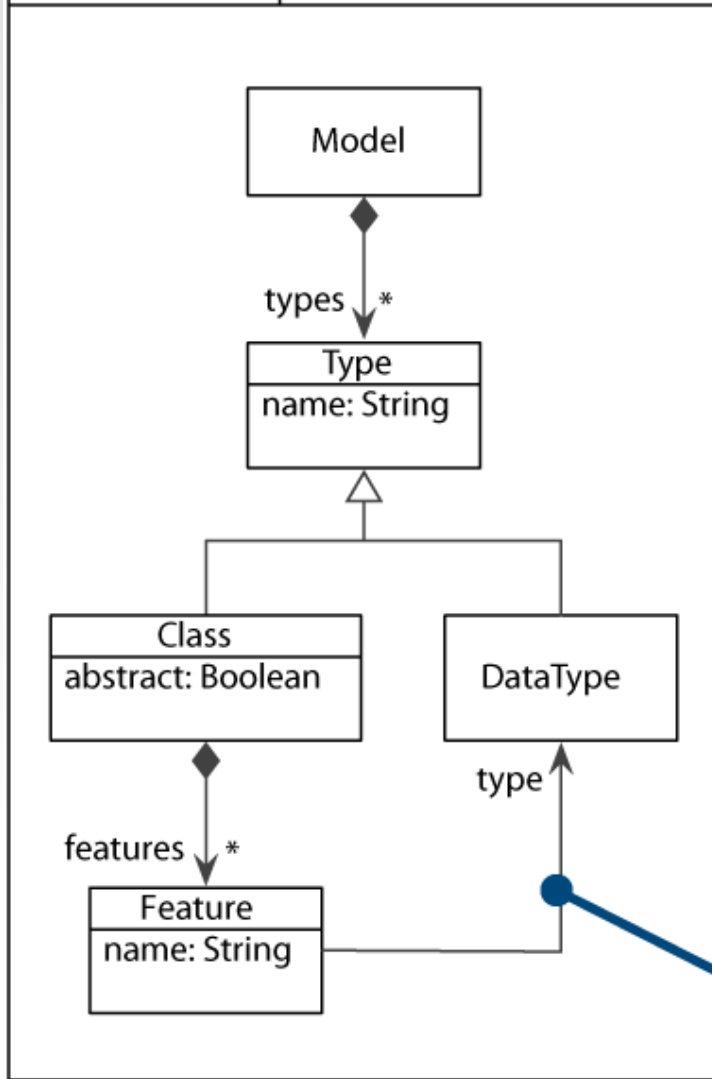
DataType:
'datatype' name=ID;

Class:
(abstract?='abstract')? 'class' name=ID '{'
● (features+=Attribute)*
'}' ;

Attribute returns Feature:
'attr' name=ID ':' type=[DataType];

Grammatik

domainModel



```
grammar org.xtext.workshop.DomainModel
    with org.eclipse.xtext.common.Terminals
```

```
generate domainModel
    " http://www.xtext.org/workshop/DomainModel "
```

```
Model:
    (types+=Type)+;
```

```
Type:
    Class | DataType;
```

```
DataType:
    'datatype' name=ID;
```

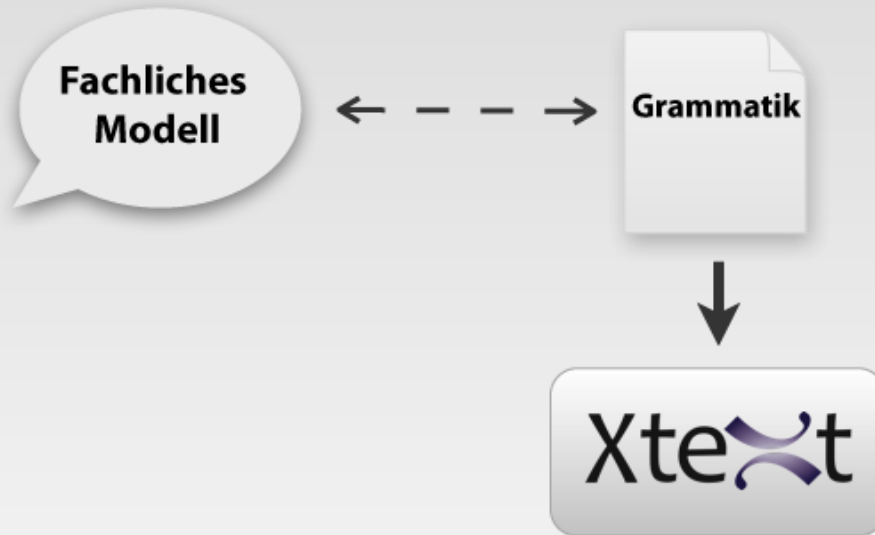
```
Class:
    (abstract?='abstract')? 'class' name=ID '{'
    (features+=Attribute)*
    '}' ;
```

```
Attribute returns Feature:
    'attr' name=ID ':' type=[DataType];
```

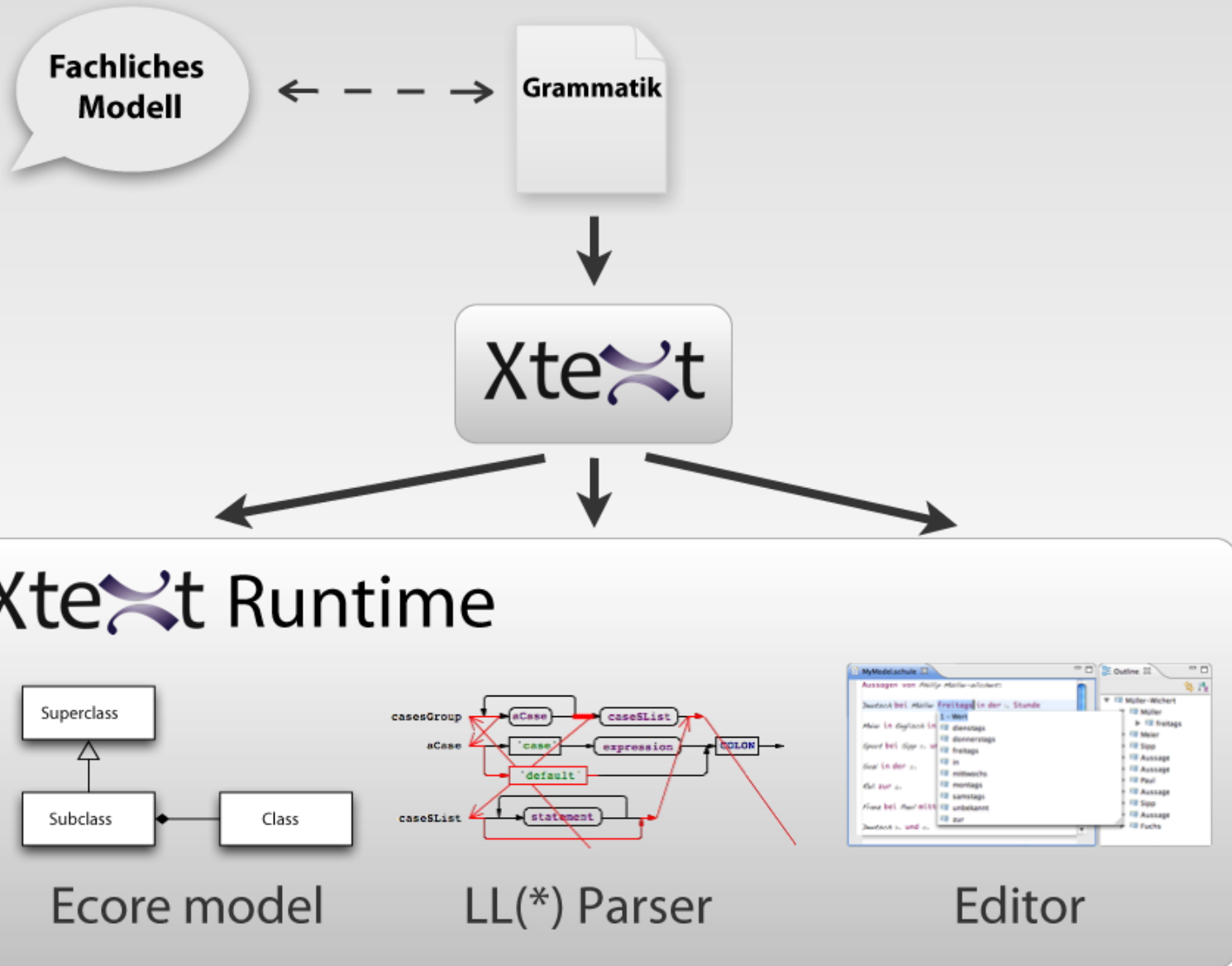
Xtext - Aufbau und Runtime



Xtext - Aufbau und Runtime

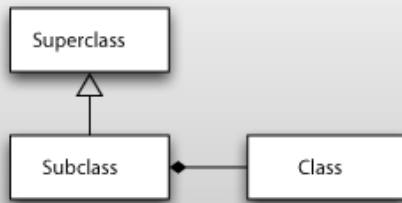


Xtext - Aufbau und Runtime

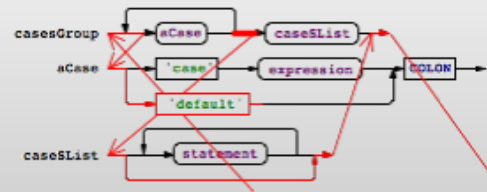


Xtext - Aufbau und Runtime

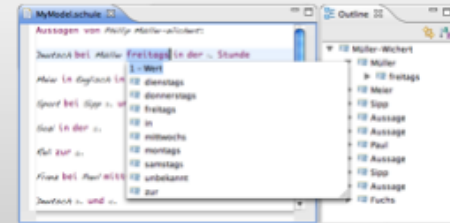
Xtext Runtime



Ecore model



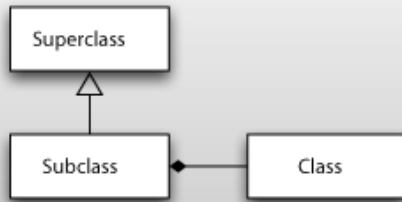
LL(*) Parser



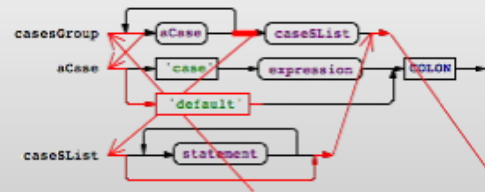
Editor

Xtext - Aufbau und Runtime

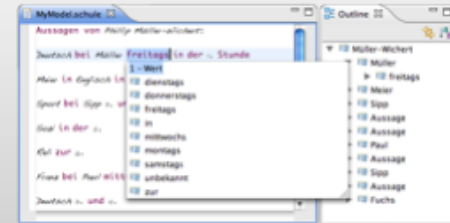
Xtext Runtime



Ecore model



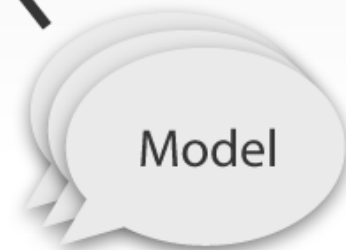
LL(*) Parser



Editor

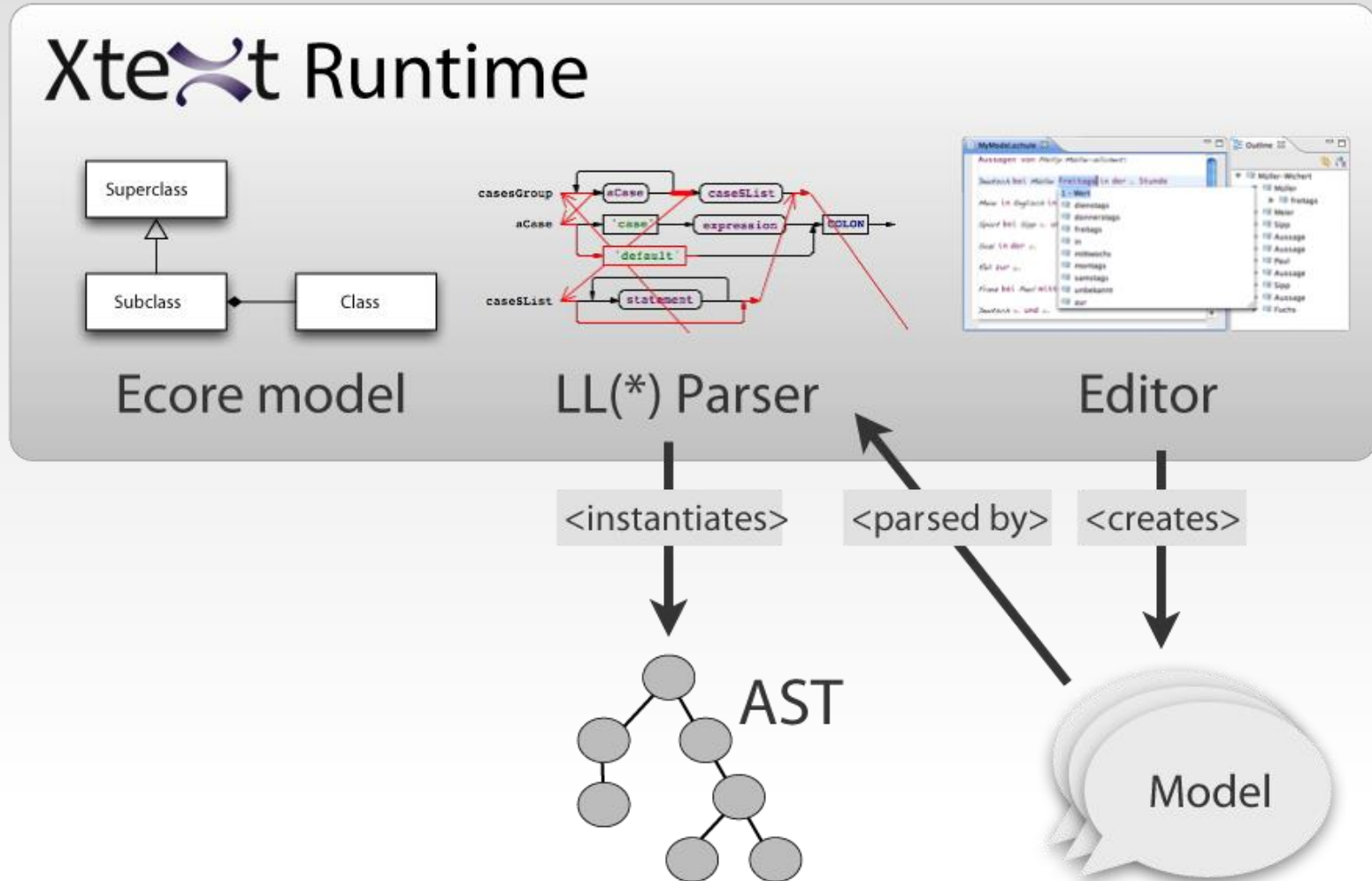
<parsed by>

<creates>

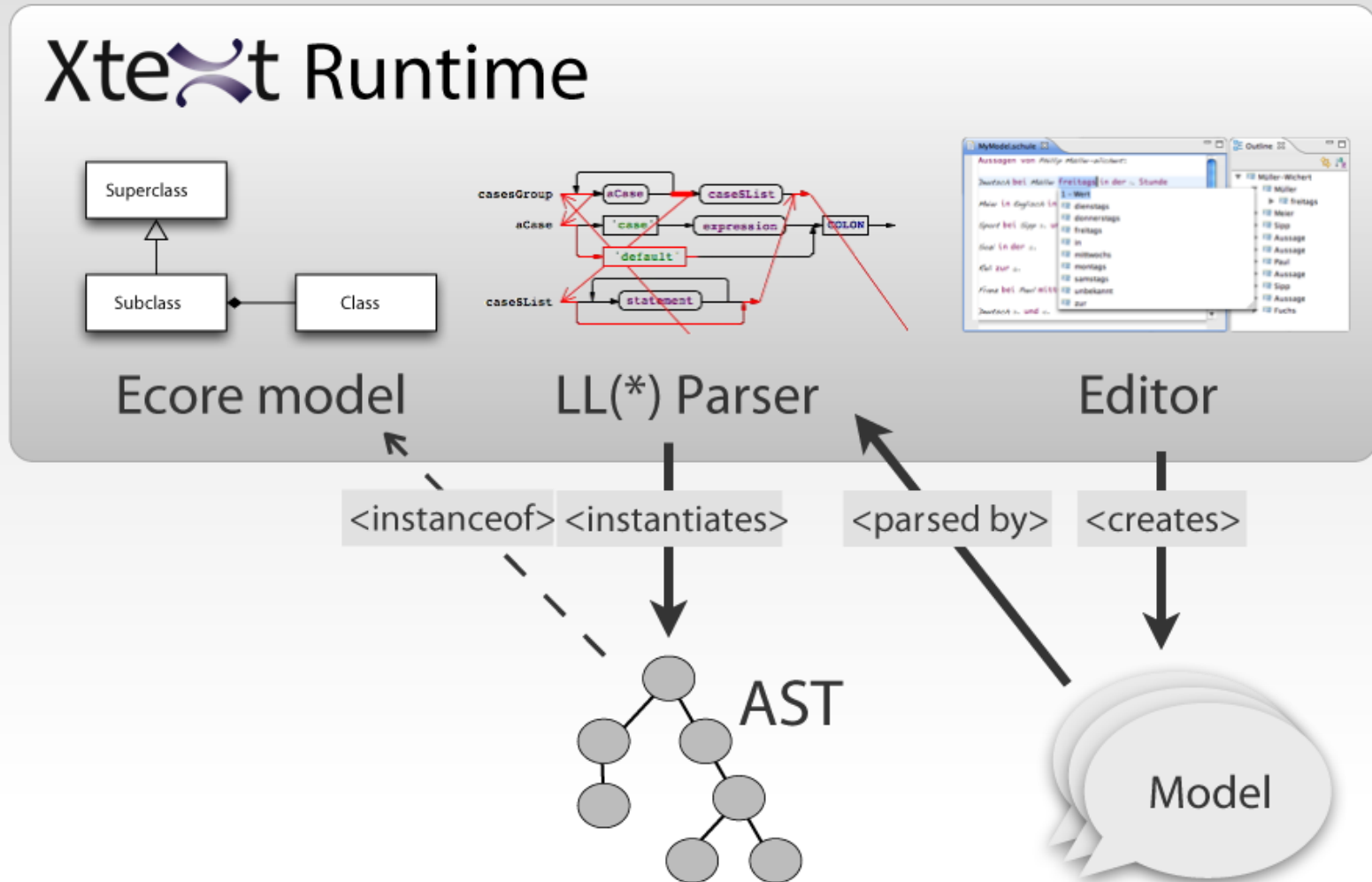


Model

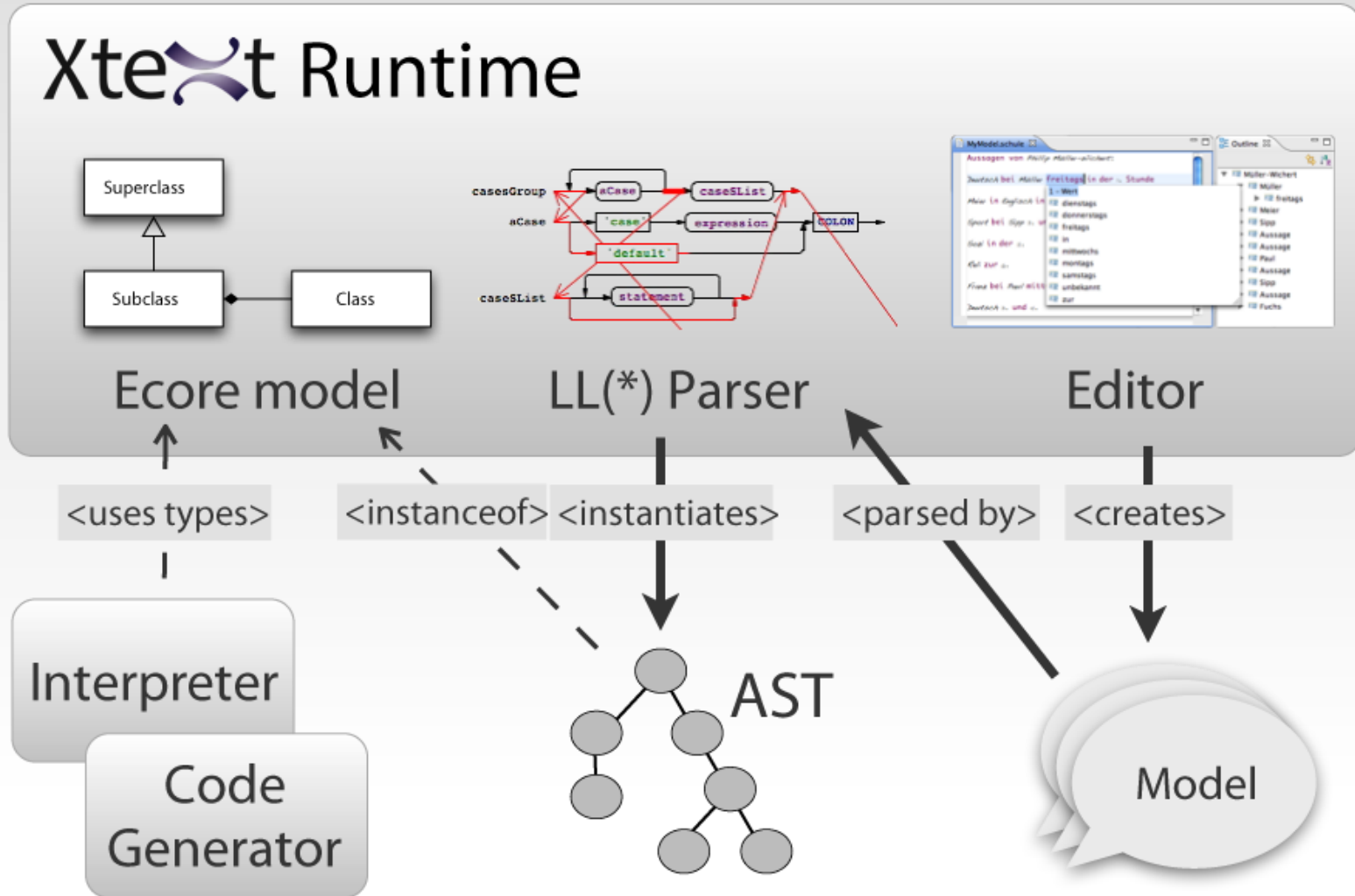
Xtext - Aufbau und Runtime



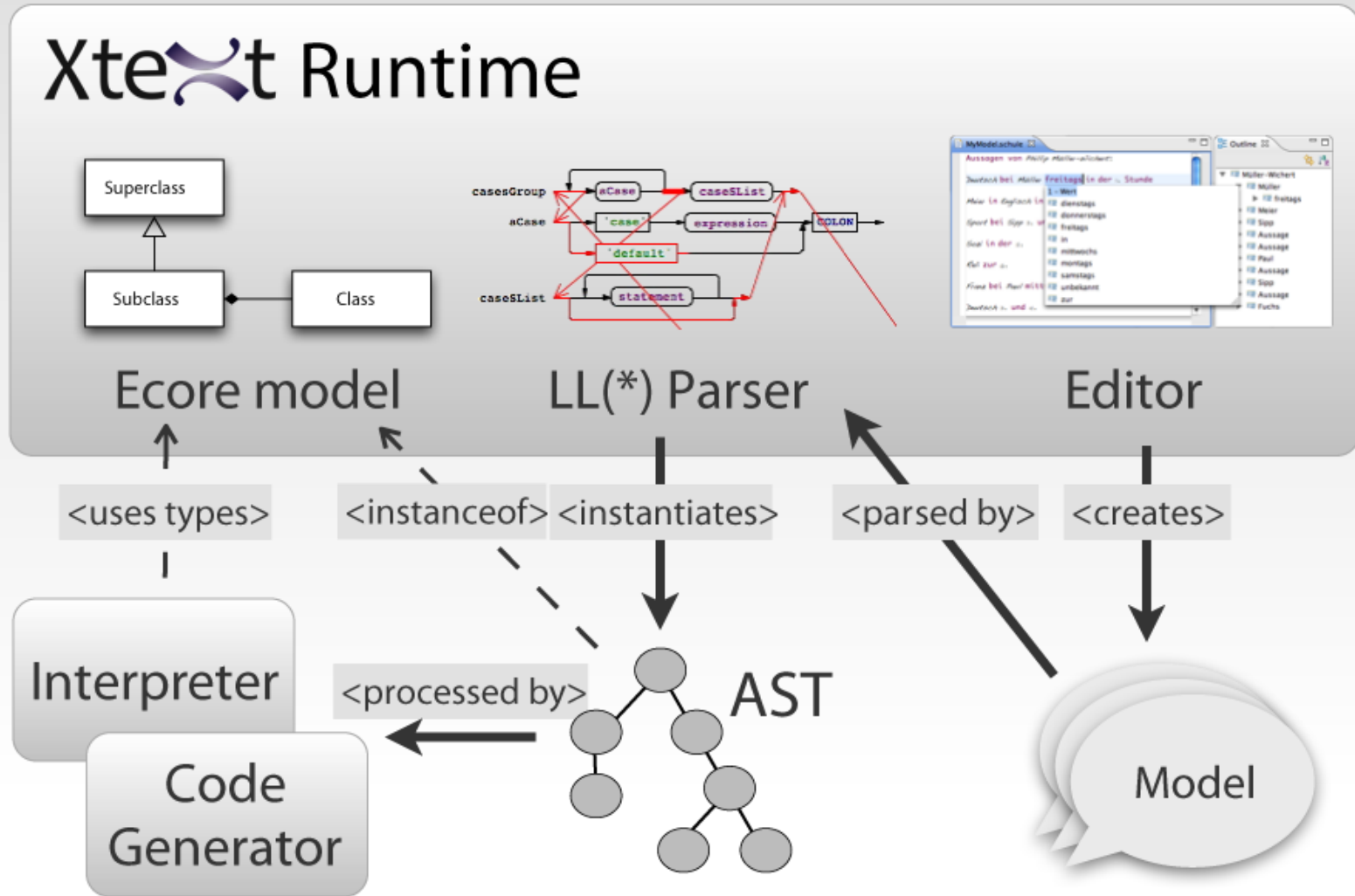
Xtext - Aufbau und Runtime



Xtext - Aufbau und Runtime



Xtext - Aufbau und Runtime



Xtext @ itemis

- Ca. 40 Mitarbeiter an verschiedenen Standorten
- DSLs und MDSD von Beginn an eines der Kernthemen
- Breites Spektrum an Projekten und Kunden



Xtext @ itemis

Kunden



SIEMENS



Mercedes-Benz



ZURICH



thyssenkrupp



BOSCH



COMMERZBANK



DEUTSCHE BÖRSE
GROUP

NORD / LB

Beispielprojekte

Programmablaufsprache (PAS)

- Seit 2016 in Entwicklung für Daimler (closed Source)
- Ziel: Testabläufe in Fahrzeugen beschreiben
- Bestehende prozedurale Sprache ohne Editorsupport
- Genutzt von Testingenieuren
- Bis zu 20.000 Zeilen in einer Datei
 - Performance und Skalierung

Programmablaufsprache (PAS)

The screenshot displays a development environment with the following components:

- Project Explorer:** Shows a project structure with files like `ende_def.pab`, `epkb166.prs`, `ess166.prs`, and others.
- Outline:** Lists variables such as `S_ESS166_Revision`, `I_ESS166_Freigabe`, and `I_ESS166_Base_Audio`.
- Code Editor:** Contains Pascal code for the `ess166.prs` file, including logic for setting audio variants based on ECU type and component status.
- Bottom Panel:** Includes tabs for Problems, Tasks, Console, Progress, Search, Log Viewer, Internal Web Browser, Markers, and Error Log. Below these is a table with the following structure:

0 items					
	Description	Resource	Path	Location	Type
✓					

Programmablaufsprache (PAS)

```
if (I_ESS166_Freigabe) {
  if (s_ECU != "ESS213") {
    I_ESS166_Base_Audio = UTIL_AUSSTATTUNG_VORHANDEN("ESS166", S_ESS166_Snr_SW, "Audio_Lines
    I_ESS166_Premium_Audio = UTIL_AUSSTATTUNG_VORHANDEN("ESS166", S_ESS166_Snr_SW, "Audio_Lines
    if (s_ECU != "ESS205")
      I_ESS166_HighEnd_Audio = UTIL_AUSSTATTUNG_VORHANDEN("ESS166", S_ESS166_Snr_SW, "Audio_Lin
    I_ESS166_Steuerleitung_pruefen = I_ESS166_Steuerleitung_pruefen && I_ESS166_Premium_Audio;
  } else {
    I_ESS166_Base_Audio = 1;
  }
  // Dokumentation Audio-Varianten
  if (I_ESS166_Base_Audio)
    s_Audio_Variant = "Base Audio";
  else if (I_ESS166_Premium_Audio)
    s_Audio_Variant = "Premium Audio";
  else if (I_ESS166_HighEnd_Audio)
    s_Audio_Variant = "HighEnd Audio";
  SetResultData("AUDIO_VARIANTE: ", s_Audio_Variant);
}
}
```

Yakindu Statechart Tools

- Ziel: Modellierung von Zustandsautomaten
- Open Source (EPL)
- Seit über 10 Jahren in Entwicklung
- 600-800 Downloads pro Monat
- Hauptsächlich im embedded Bereich eingesetzt
- Kombiniert grafische und textuelle DSLs
- Genutzt von Universitäten rund um die Welt



Yakindu Statechart Tools



Yakindu Statechart Tools

Editing

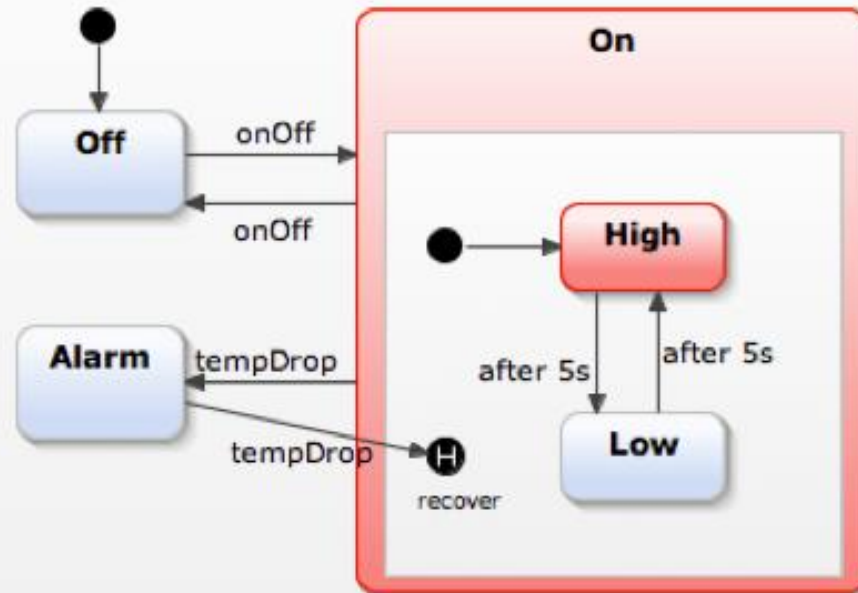
heating

interface :

in event onOff
in event tempDrop

var tempSetPoint : integer

main region



Validation

Simulation

Code Generation

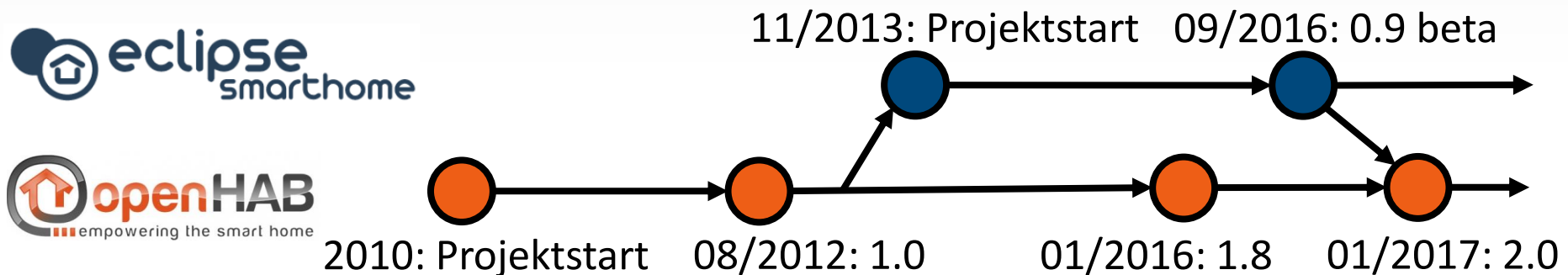


YAKINDU

Demo

openHAB

- Open Source Smart Home Plattform
- Hersteller- und technologieneutral
- Basierend auf OSGi und Java (embedded)
- Seit Version 2.0 basierend auf Eclipse Smarthome
- Das Ziel:
 - Geräte versch. Hersteller und Protokollen ansteuern
 - Verbinden der Geräte in Szenarios mittels Regeln



Szenario

Morgenroutine

Wenn

- es 7:00 Uhr ist

dann

- mach das Licht an,
- starte die Kaffeemaschine,
- heize das Badezimmer,
- öffne die Rolläden,
- mach das Radio an,
- ...

aber nur

- während der Woche.



Morgenroutine

Wenn

- es 7:00 Uhr

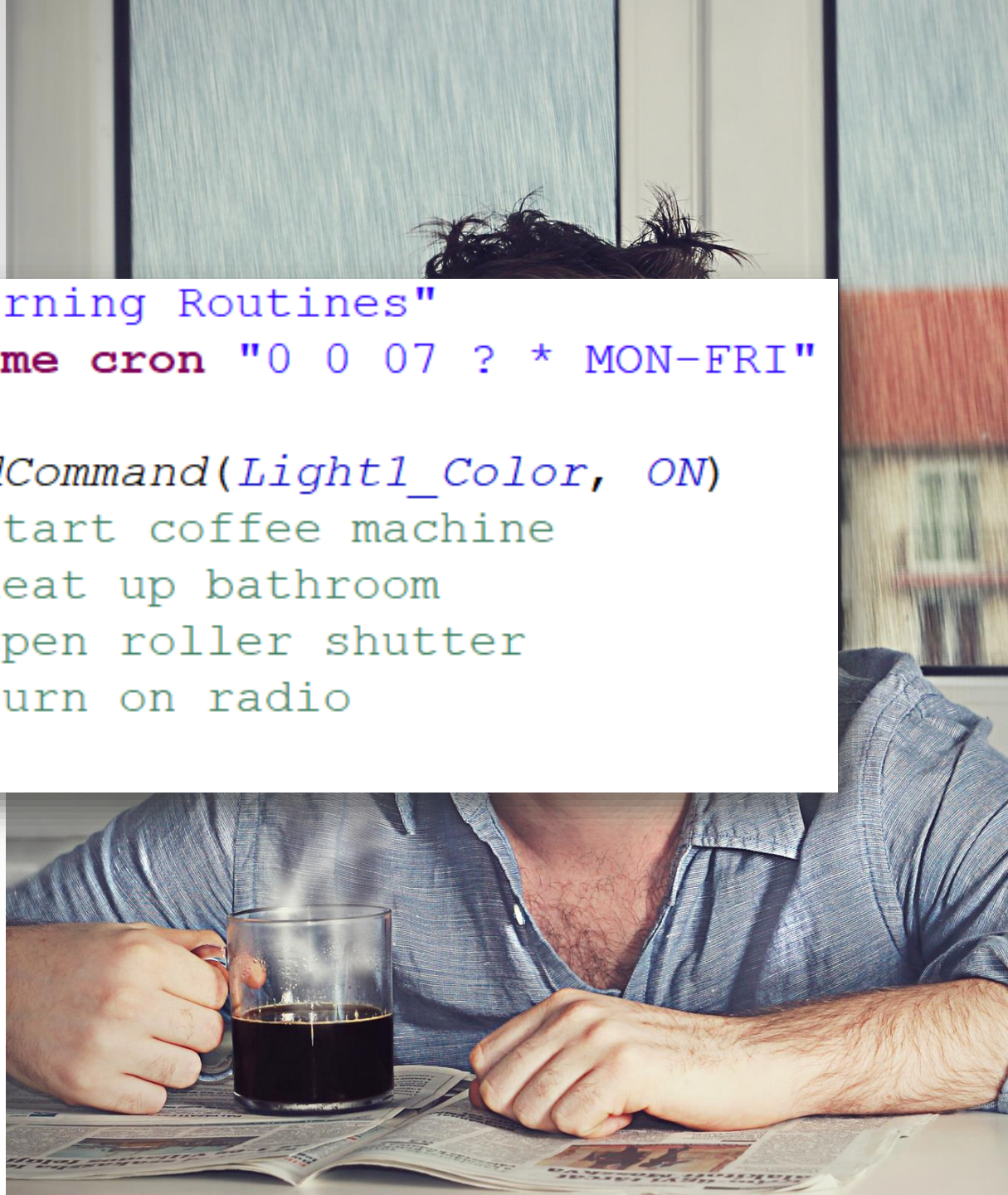
dann

- mach das Licht an
- starte die Kaffeemache
- heize das Bad
- öffne die Rollläden
- mach das Radio an,
- ...

aber nur

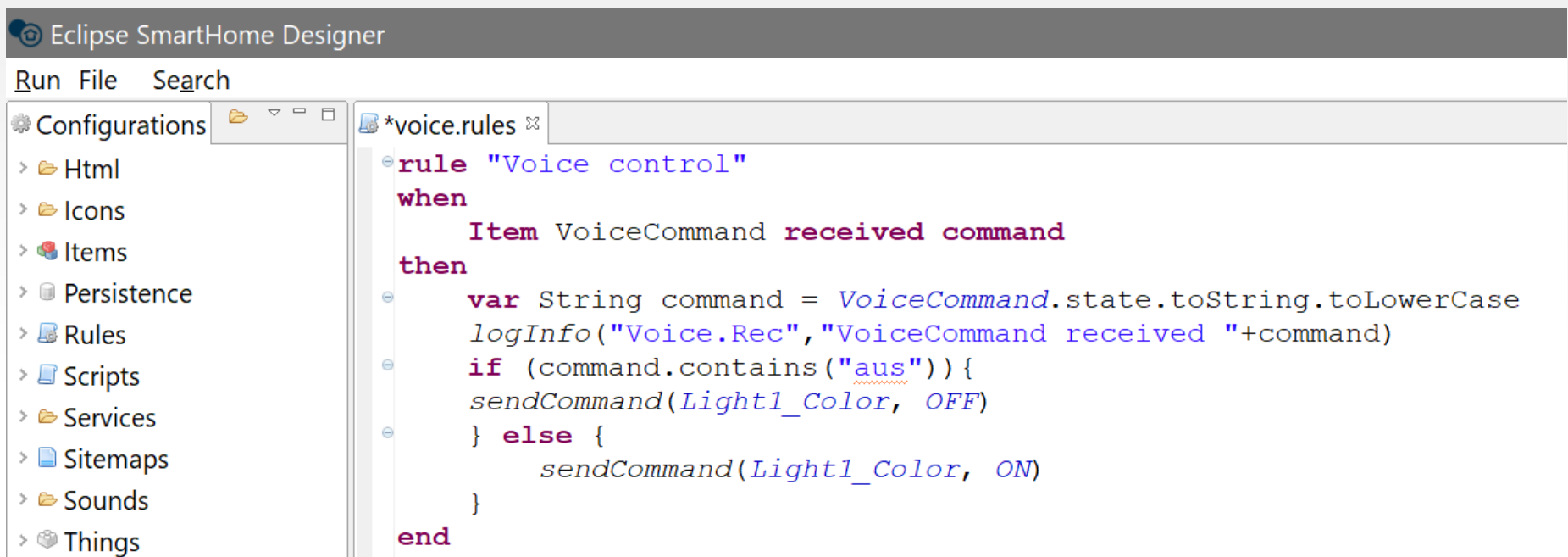
- während der Woche.

```
rule "Morning Routines"  
when Time cron "0 0 07 ? * MON-FRI"  
then  
    sendCommand(Light1_Color, ON)  
    // start coffee machine  
    // heat up bathroom  
    // open roller shutter  
    // turn on radio  
end
```



openHAB & Xtext

- Xtext-basierte Konfiguration von
 - Regeln, Skripten, Items, ...
- Mittels Samba Freigegeben
- Updates während der Laufzeit



The screenshot shows the Eclipse SmartHome Designer interface. The left sidebar contains a tree view of project resources: Configurations, Html, Icons, Items, Persistence, Rules, Scripts, Services, Sitemaps, Sounds, and Things. The main editor window displays the content of a file named `*voice.rules`. The code is written in Xtext and defines a rule for voice control.

```
rule "Voice control"
when
    Item VoiceCommand received command
then
    var String command = VoiceCommand.state.toString.toLowerCase
    logInfo("Voice.Rec", "VoiceCommand received "+command)
    if (command.contains("aus")) {
        sendCommand(Light1_Color, OFF)
    } else {
        sendCommand(Light1_Color, ON)
    }
end
```

openHAB & Xtext

```
val initial = Light1_Color.state as HSBType
val initialBrightness = initial.getBrightness.intValue
val stepWith = 30
val fixedSaturation = 100

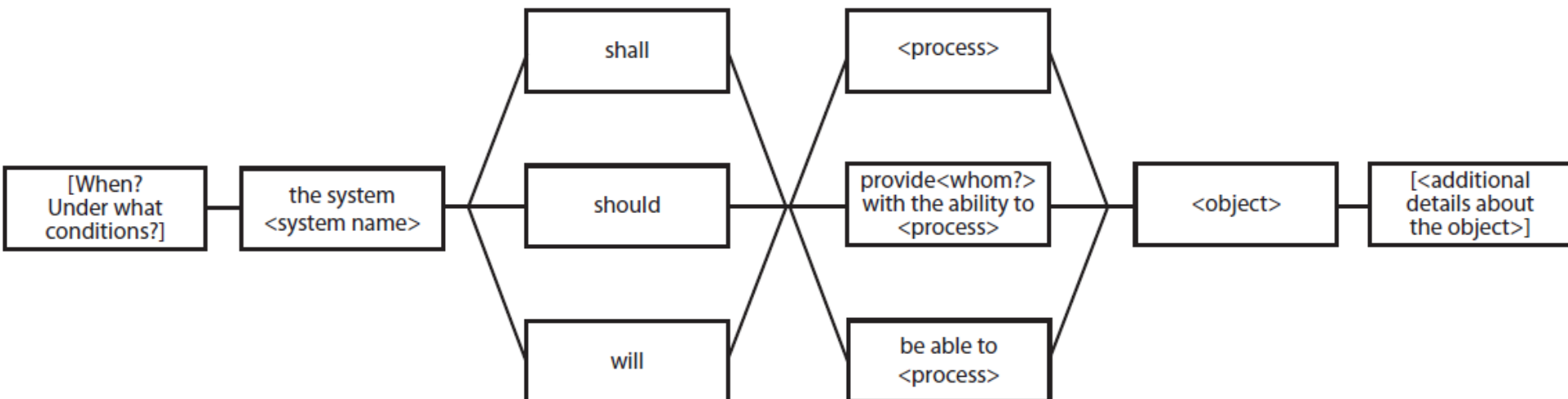
var reversed = false
for (step : (0..10)) {
    var current = Light1_Color.state as HSBType
    var currentColor = current.getHue.intValue
    // 360 -> 0
    if (currentColor + stepWith >= 360 ){
        reversed = true
        // 0 -> 360
    } else if (currentColor - stepWith <= 0){
        reversed = false
    }
    var int newColor
    if (reversed){
        newColor = currentColor - stepWith
    } else {
        newColor = currentColor + stepWith
    }
    var command = new HSBType(newColor+", "+fixedSaturation+", "+initialBrightness)
    current = Light1_Color.state as HSBType
    logInfo("Rainbow", "Rainbow mode: current color:"+current+", new color:"+command+", reversed "+reversed)
    sendCommand(Light1_Color, command.toString)
    Thread.sleep(1000)
}
```

Sefo-Rel

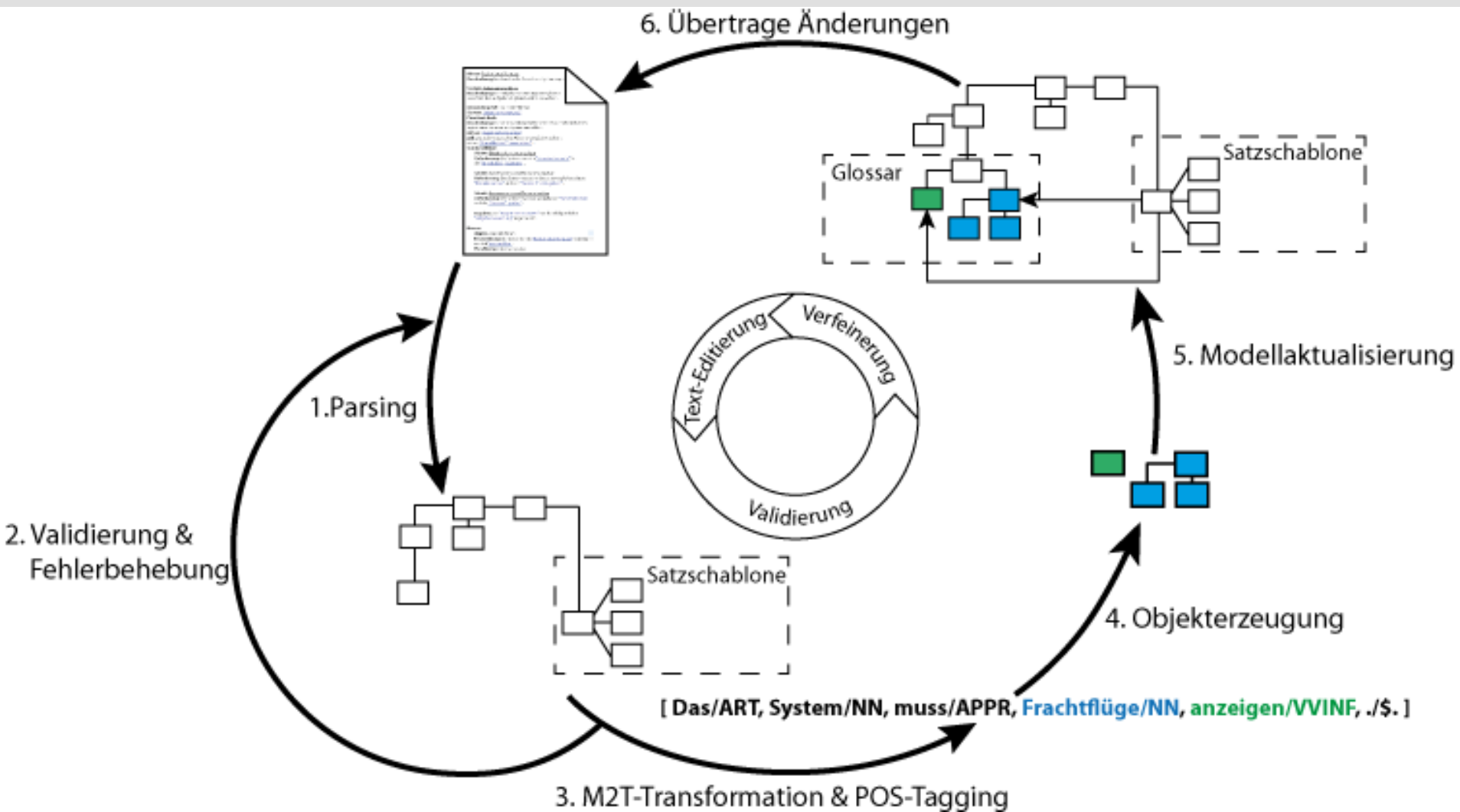
Semi Formal Requirements
elicitation language

Sefo-rel

- Forschungsprojekt
- Ziel: Semiformale Anforderungsanalyse
 - Validierung und Einschränkung natürlicher Sprache
- Basis: Satz- und Strukturschablonen
- Xtext + Natürliche Sprachverarbeitung für Freitexte



Sefo-rel



Anwendungsfall: Login durchführen

System: "Aufgabenverwaltung"


Prioritaet: Hoch

Beschreibung: Dieser Anwendungsfall beschreibt wie sich ein bereits registrierter Benutzer am System anmeldet.

Akteur: "Registrierter Benutzer"

Ziel: *Das System muss dem Akteur ermöglichen sich mit seinen Anmelde*daten "anzumelden".

Glossar:

 Zum Geschäftsobjekt 'Anmeldedaten' liegt kein Glossareintrag vor

1 quick fix available:

 [Erstelle einen neuen Glossareintrag für das Objekt 'Anmeldedaten'](#)

Anwendungsfall: Login durchführen

System: "Aufgabenverwaltung"

Prioritaet: Hoch

Beschreibung: Dieser Anwendungsfall beschreibt wie sich ein bereits registrierter Benutzer am System anmeldet .

Akteur: "Registrierter Benutzer"

Ziel: *Das System muss dem Akteur ermöglichen sich mit seinen "Anmeldedaten" "anzumelden".*

Glossar:

Objekt: Anmeldedatum

Pluralformen: Anmeldedaten

Singularformen: Anmeldedatums

Anwendungsfall: Login durchführen

System: "Aufgabenverwaltung"

Prioritaet: Hoch

Beschreibung: Dieser Anwendungsfall beschreibt wie sich ein bereits registrierter Benutzer am System anmeldet .

Akteur: "Registrierter Benutzer"

Ziel: Das System muss dem Akteur ermöglichen sich mit seinen "Anmeldedaten" "anzumelden" .

Standardablauf:

Schritt: Eingabeelemente anzeigen

Anforderung: Das System muss die "Eingabeelemente" für die "Anmeldung" "anzeigen" .

Schritt: Benutzername und Passwort eingeben

Anforderung: Das System muss dem Akteur ermöglichen seinen "Benutzernamen" und sein "Passwort" "einzugeben" .

Schritt: Benutzername und Passwort prüfen

Anforderung: Das System muss den eingegebenen "Benutzernamen" und das "Passwort" "prüfen" .

Ergebnis: Der "Registrierter Benutzer" wurde erfolgreich der "Aufgabenverwaltung" angemeldet.

Glossar:

Objekt: Anmeldedatum

Beschreibung: Die Daten die ein "Registrierter Benutzer" benötigt um sich "anzumelden".

Pluralformen: Anmeldedaten

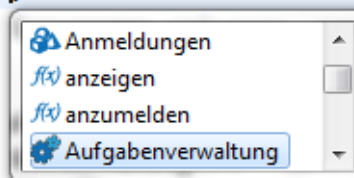
Singularformen: Anmeldedatum

Eigenschaften: "Passwort" : EINS, "Benutzername" : EINS

Funktion: anmelden

Beschreibung: Ein "Registrierter Benutzer" meldet sich an der

l.



System: Aufgabenverwaltung
Beschreibung: Die Aufgabenverwaltung ermöglicht es Benutzern ihre Aufgaben zu planen und zu verwalten

Danke für die Aufmerksamkeit!

Weitere Informationen:

- www.blogs.itemis.com
- www.eclipse.org/Xtext/
- www.statecharts.org
- www.eclipse.org/smarthome/
- www.openhab.org

Twitter:

@itemis

@xtext

