Modellgetriebene Softwareentwicklung

mit

Wer ich bin

Christoph Knauf

xing.com/profile/Christoph_Knauf2

@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**

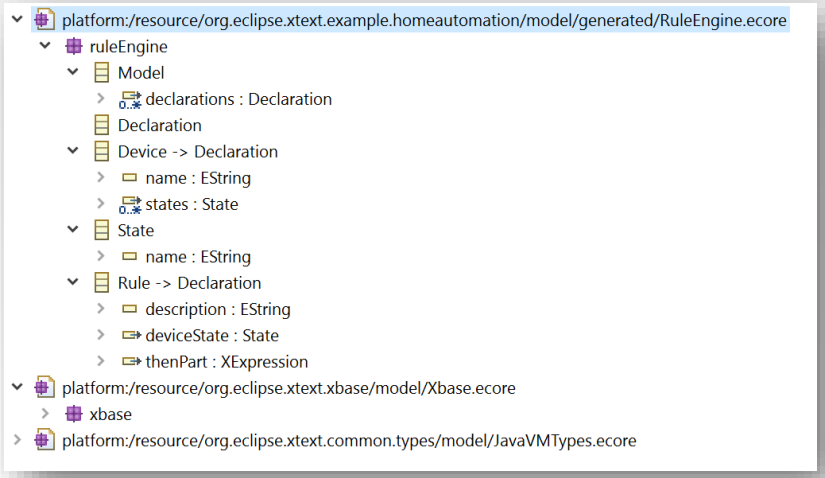
- Modell ist gleichbedeutend mit Programmcode

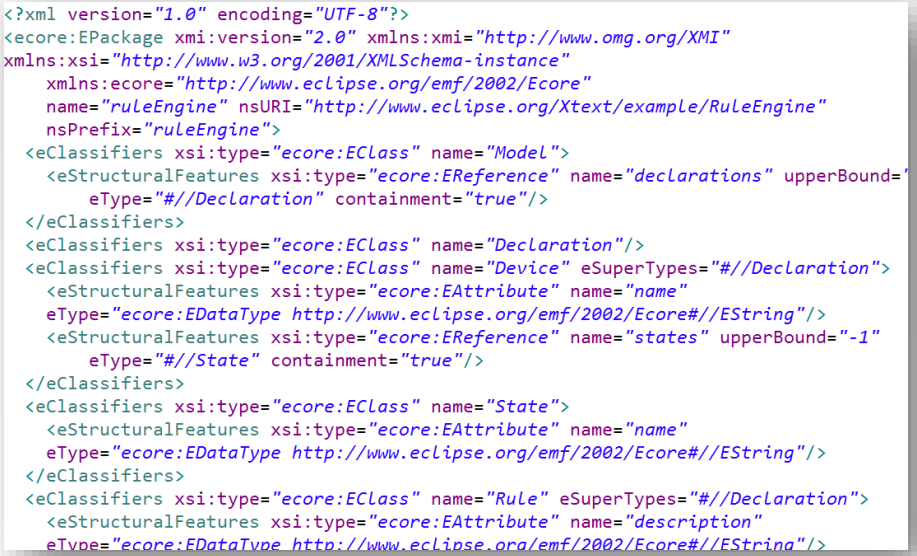
* Stahl, Völter, Efftinge, Haase. Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt.verlag, 2007

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)



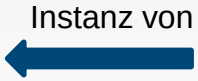


Modellgetriebene Softwareentwicklung (MDSD)

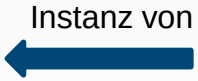
Metamodelle aka Abstrakte Syntax

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

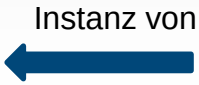
Beispiele:

XML-Schema 

XML Dokument

Kontextfreie Grammatik (BNF) 

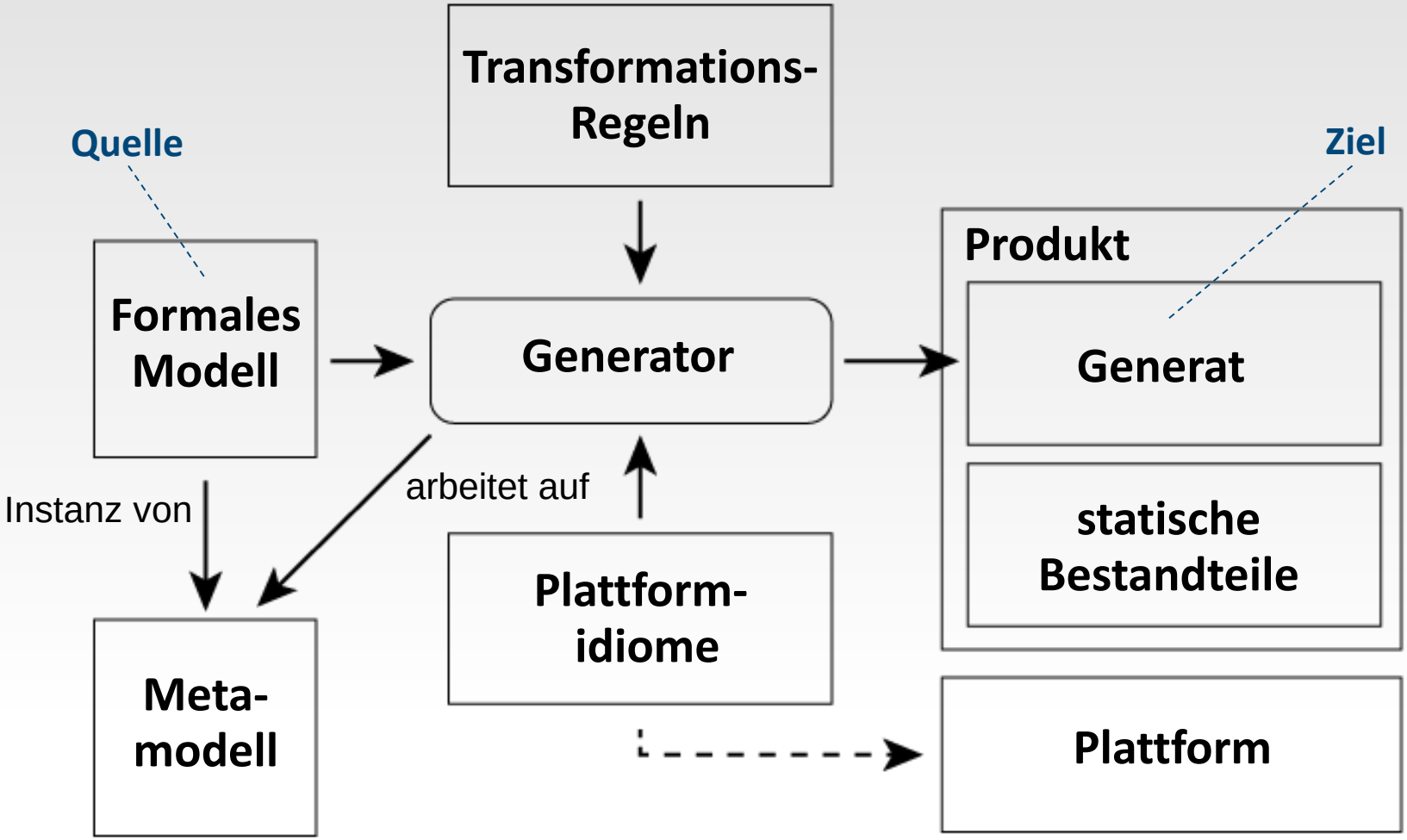
Programm der Sprache

UML Klassendiagramm 

UML Objektdiagramm

Modellgetriebene Softwareentwicklung (MDSD)

Generierung



Modellgetriebene Softwareentwicklung (MDSD)

Was kann alles Generiert werden?

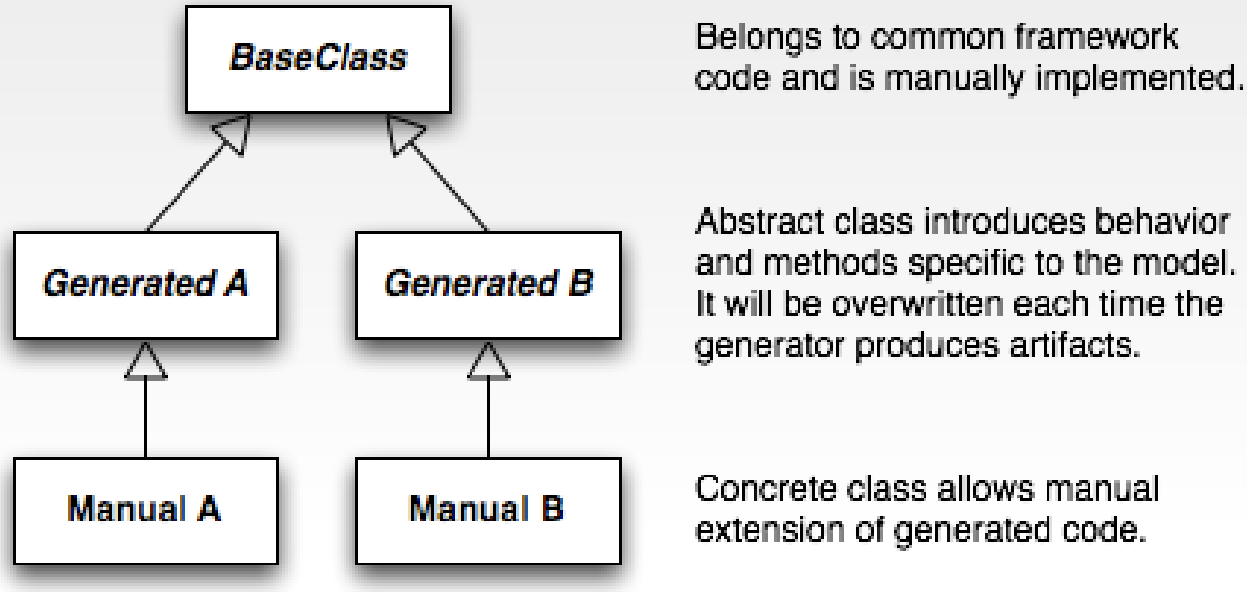
- Datenbankschemata
- Benutzerschnittstellen
- Teile der Anwendungslogik
- Datenobjekte
- Wrapper
- Dokumentation
- Konfigurationen (Spring, Hilbernate, Maven, Gradle …)
- Tests (Unit-Tests, Mocks, Lasttests, ...)
- Bibliotheken

...

Modellgetriebene Softwareentwicklung (MDSD)

Erweiterbarkeit generierten Codes

- Generation gap pattern



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

* Fowler. Domain Specific Languages. Addison-Wesley Professional, 2010

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

Parser

Lexer

Domänenspezifische Sprachen (DSL)

Bestandteile

Parser

Lexer

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)

Validierung

Linker

Domänenspezifische Sprachen (DSL)

Bestandteile

Parser

Lexer

Abstract Syntax Tree (AST)

Typinferenz

Linker

Validierung

Type-Checker

Domänenspezifische Sprachen (DSL)

Bestandteile

Parser

Lexer

Abstract Syntax Tree (AST)

Typinferenz

Linker

Validierung

Type-Checker

Interpreter

Code Generatoren

Domänenspezifische Sprachen (DSL)

Benutzerunterstützung

Refactoring

GoTo Deklarationen

Outline View

Syntax Coloring

Code Folding

Wizards

Hovers

Find References

Quick Fixes

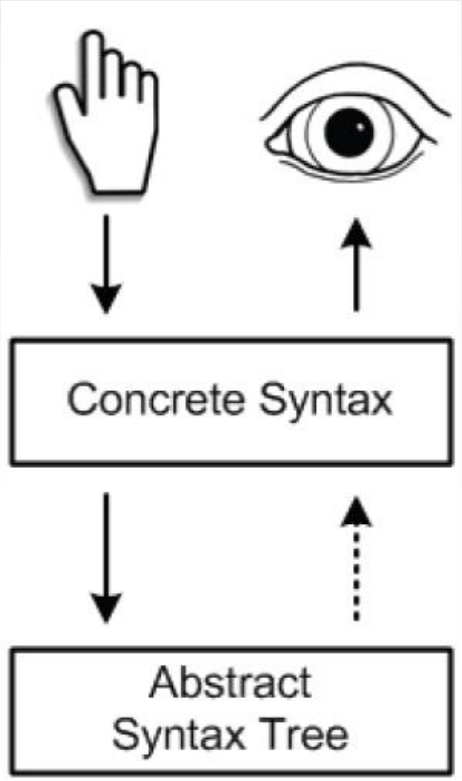
Content Assist

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

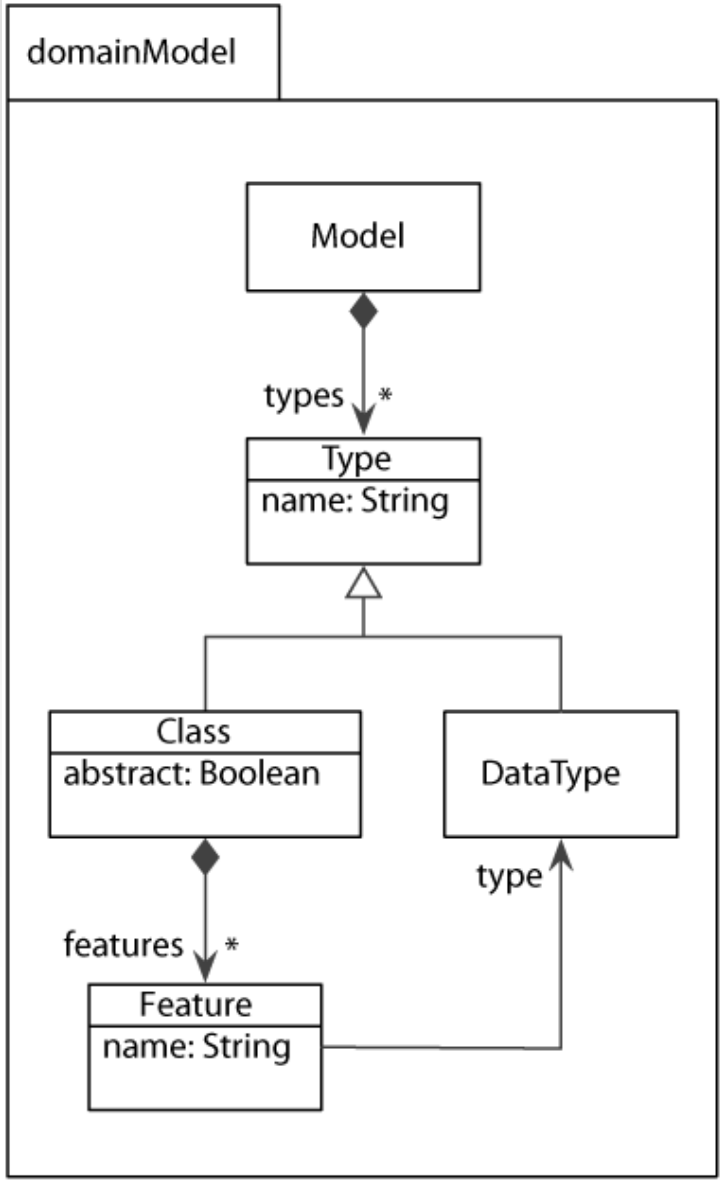


Demo

Xtext - Aufbau und Runtime

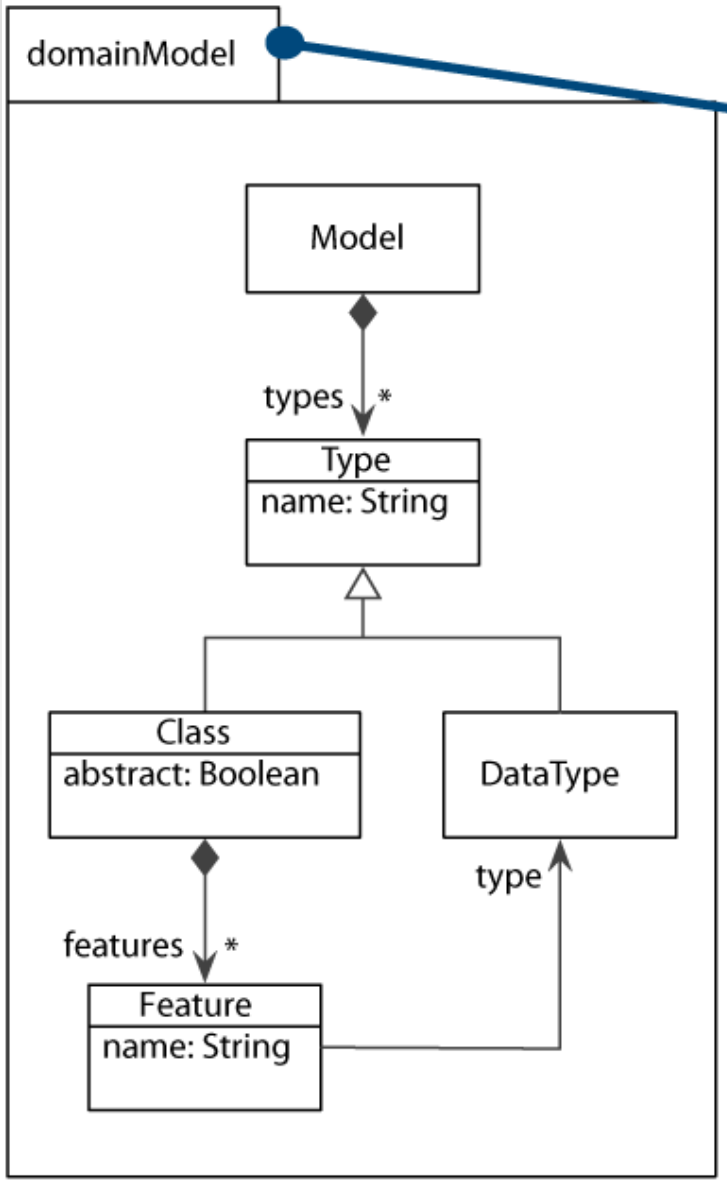
Xtext - Aufbau und Runtime

Grammatik



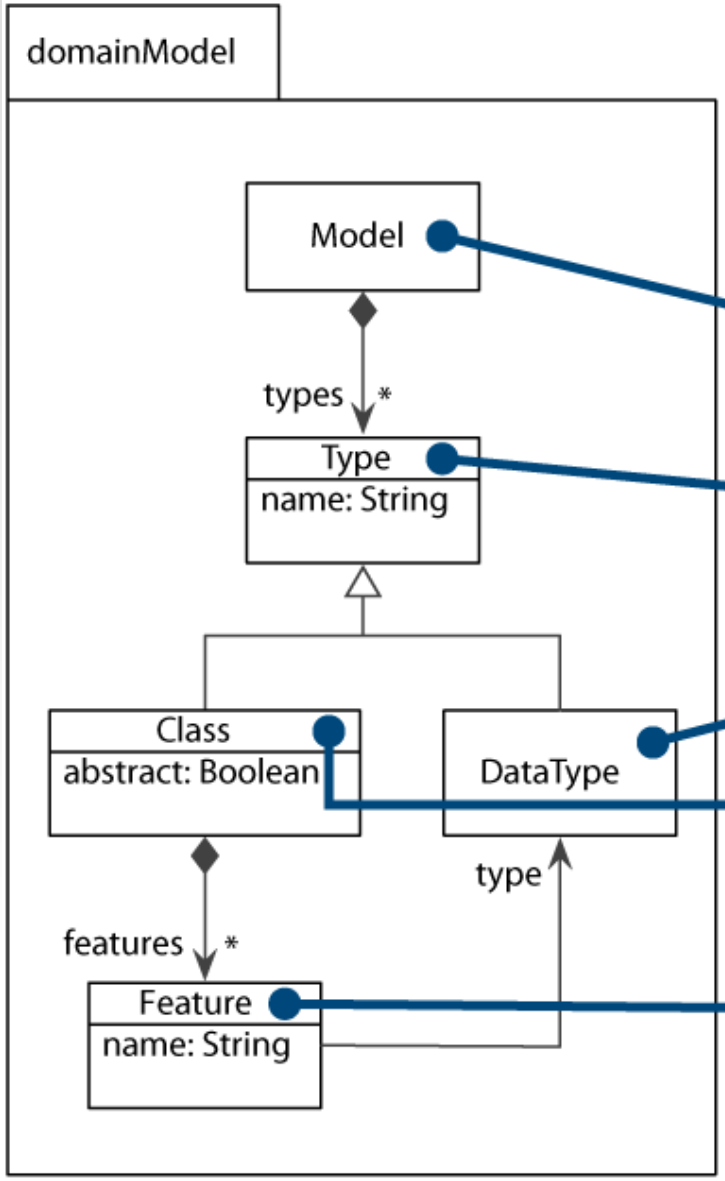

```
grammar  org.xtext.workshop.DomainModel
        with   org.eclipse.xtext.common.Terminals

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



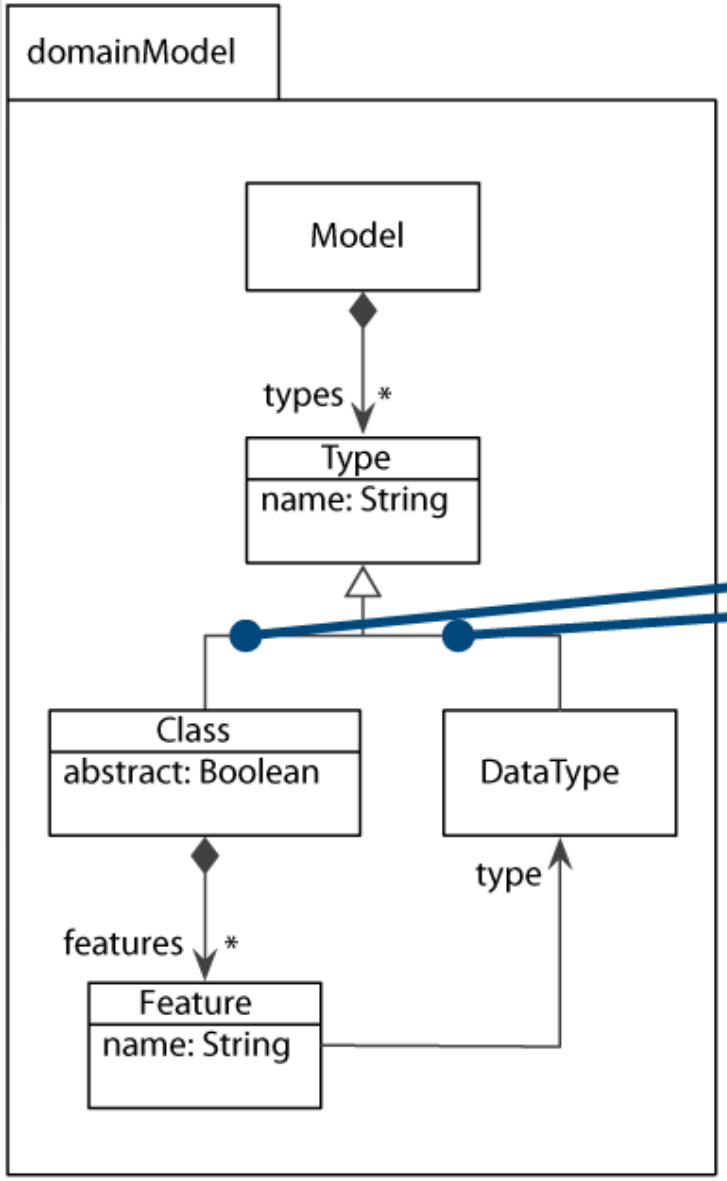

```
grammar  org.xtext.workshop.DomainModel
      with   org.eclipse.xtext.common.Terminals

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



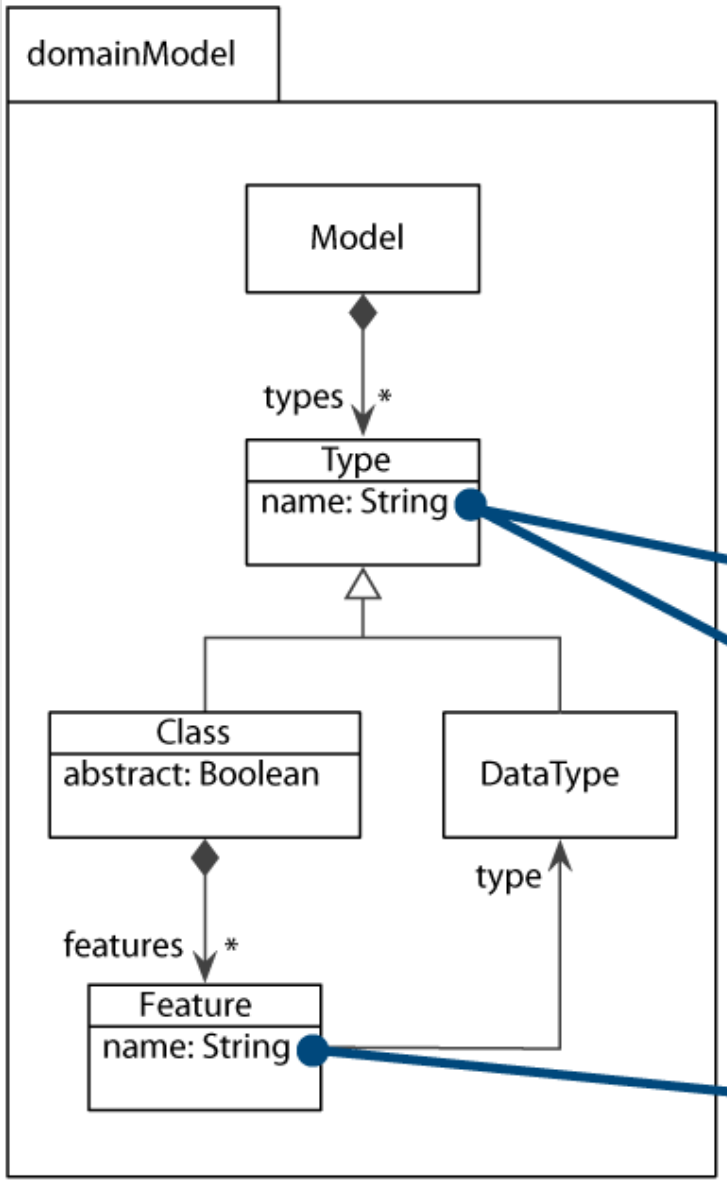

```
grammar  org.xtext.workshop.DomainModel
        with   org.eclipse.xtext.common.Terminals

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



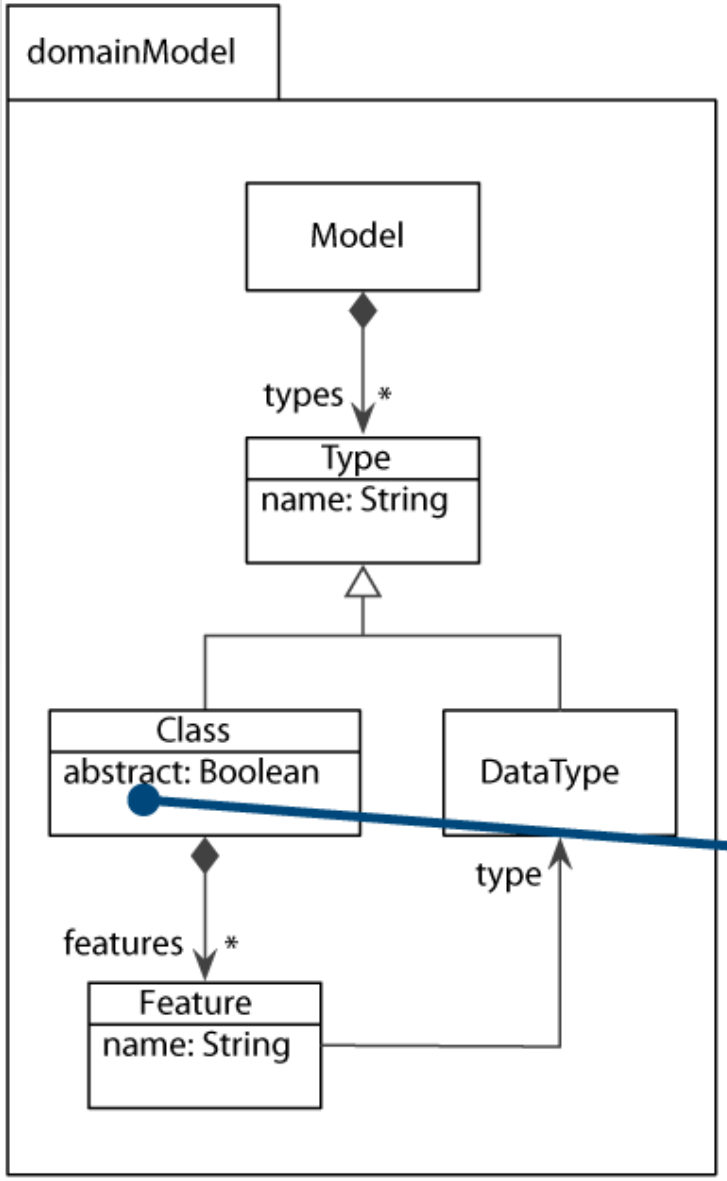

```
grammar  org.xtext.workshop.DomainModel
        with   org.eclipse.xtext.common.Terminals

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



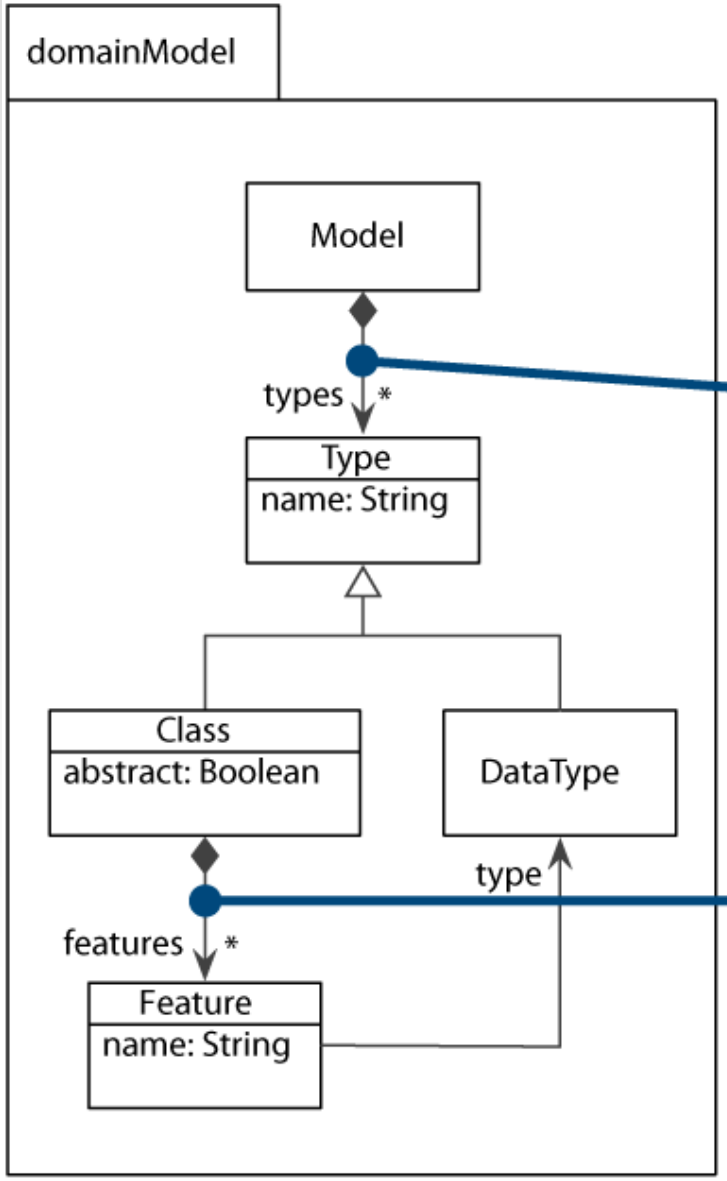

```
grammar  org.xtext.workshop.DomainModel
        with   org.eclipse.xtext.common.Terminals

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



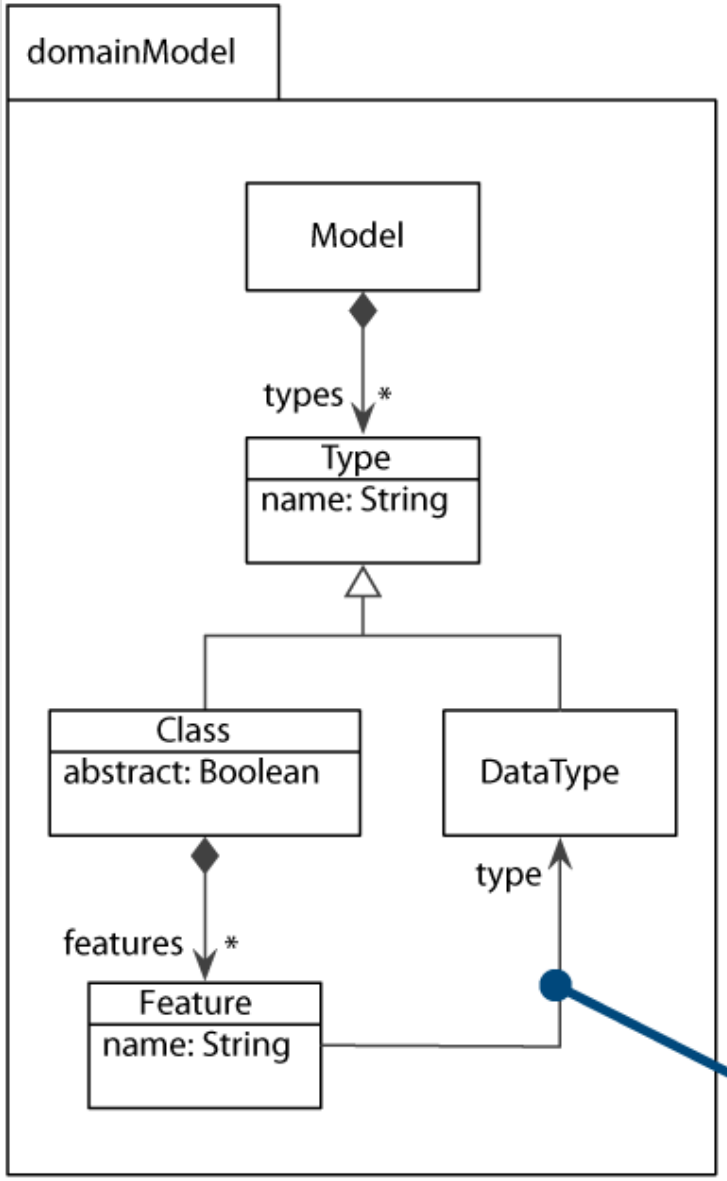

```
grammar  org.xtext.workshop.DomainModel
        with   org.eclipse.xtext.common.Terminals

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




```
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];
```

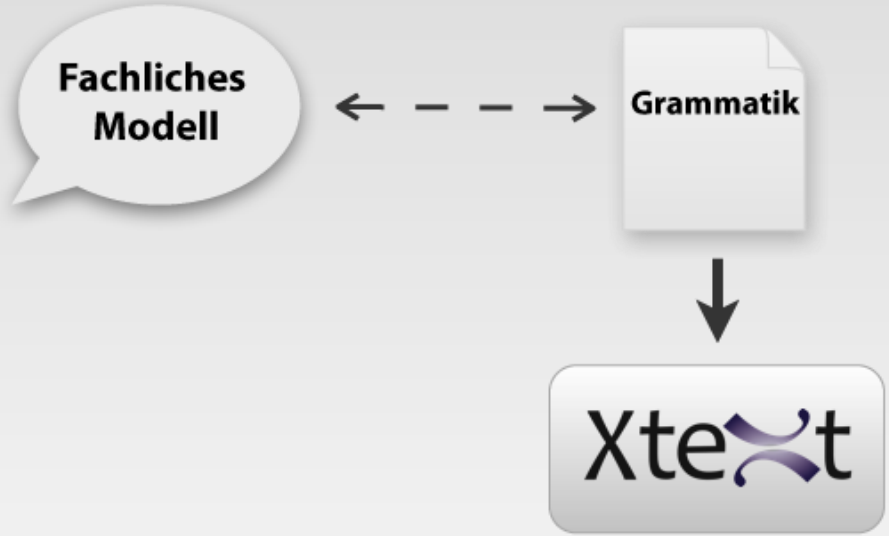
Grammatik



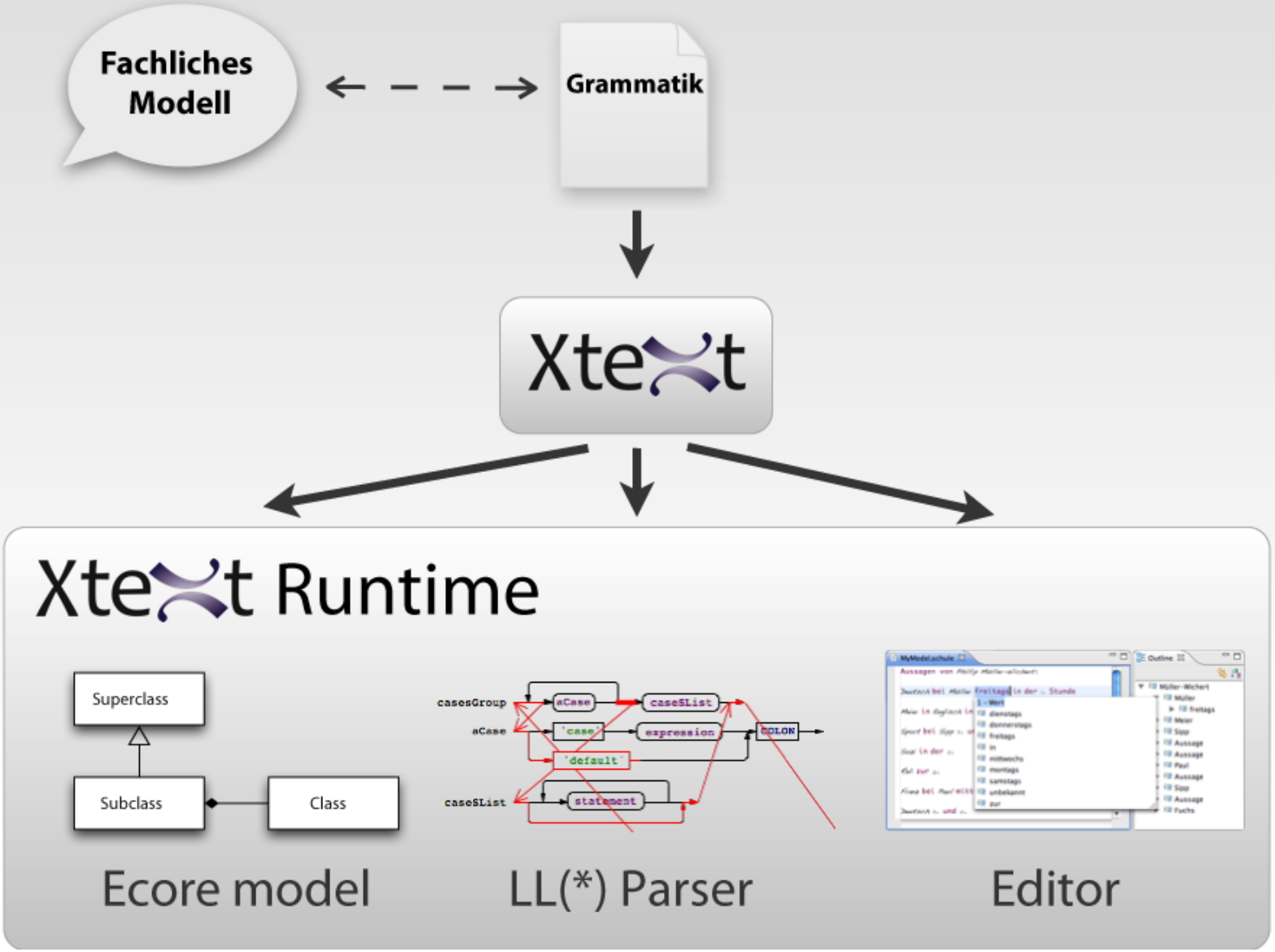

```
grammar  org.xtext.workshop.DomainModel
        with   org.eclipse.xtext.common.Terminals

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

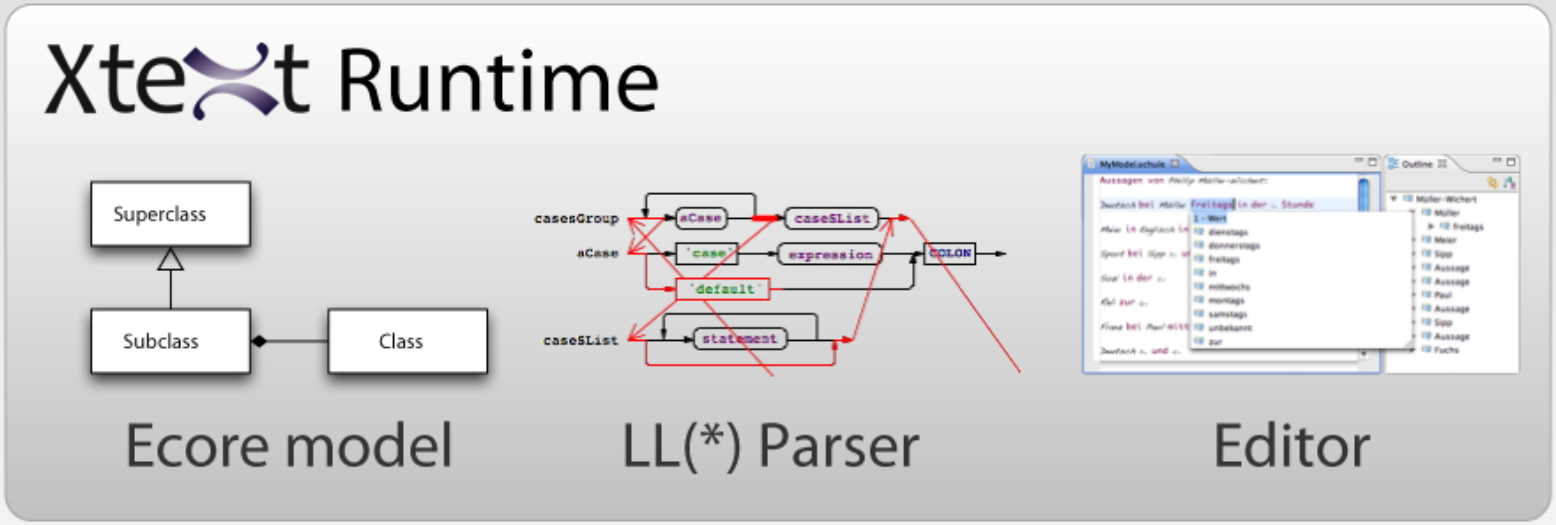
Xtext - Aufbau und Runtime



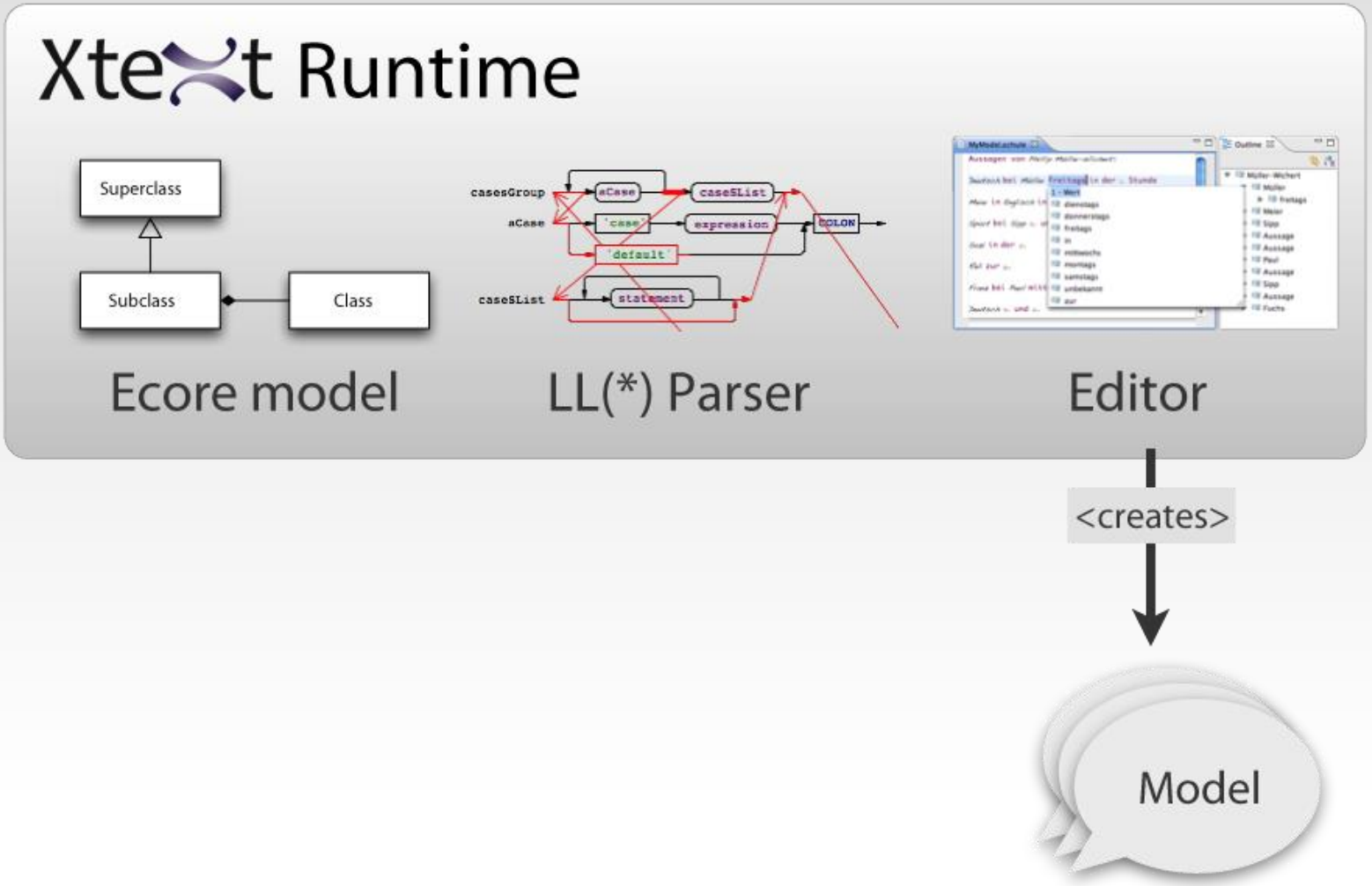
Xtext - Aufbau und Runtime



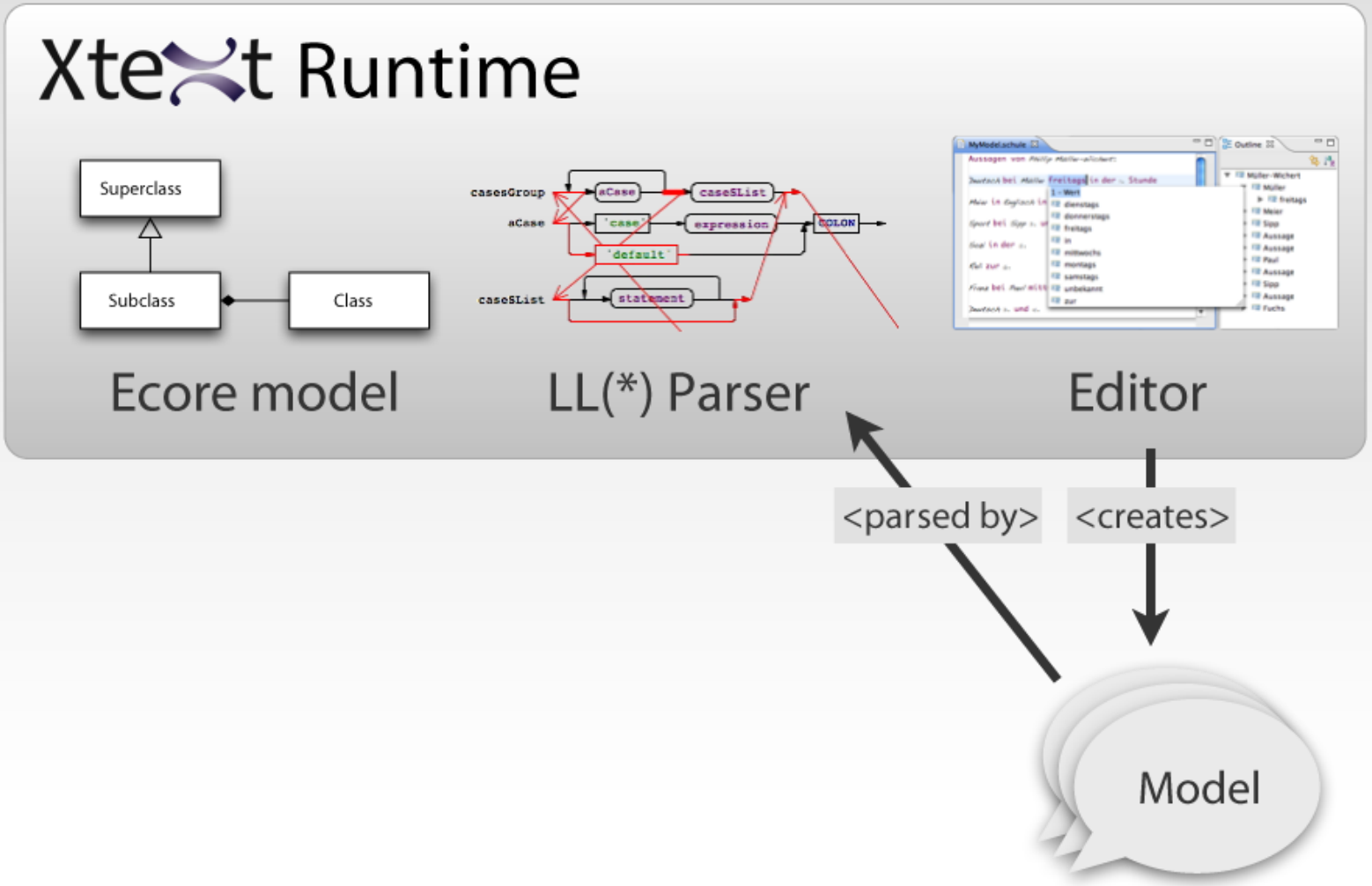
Xtext - Aufbau und Runtime



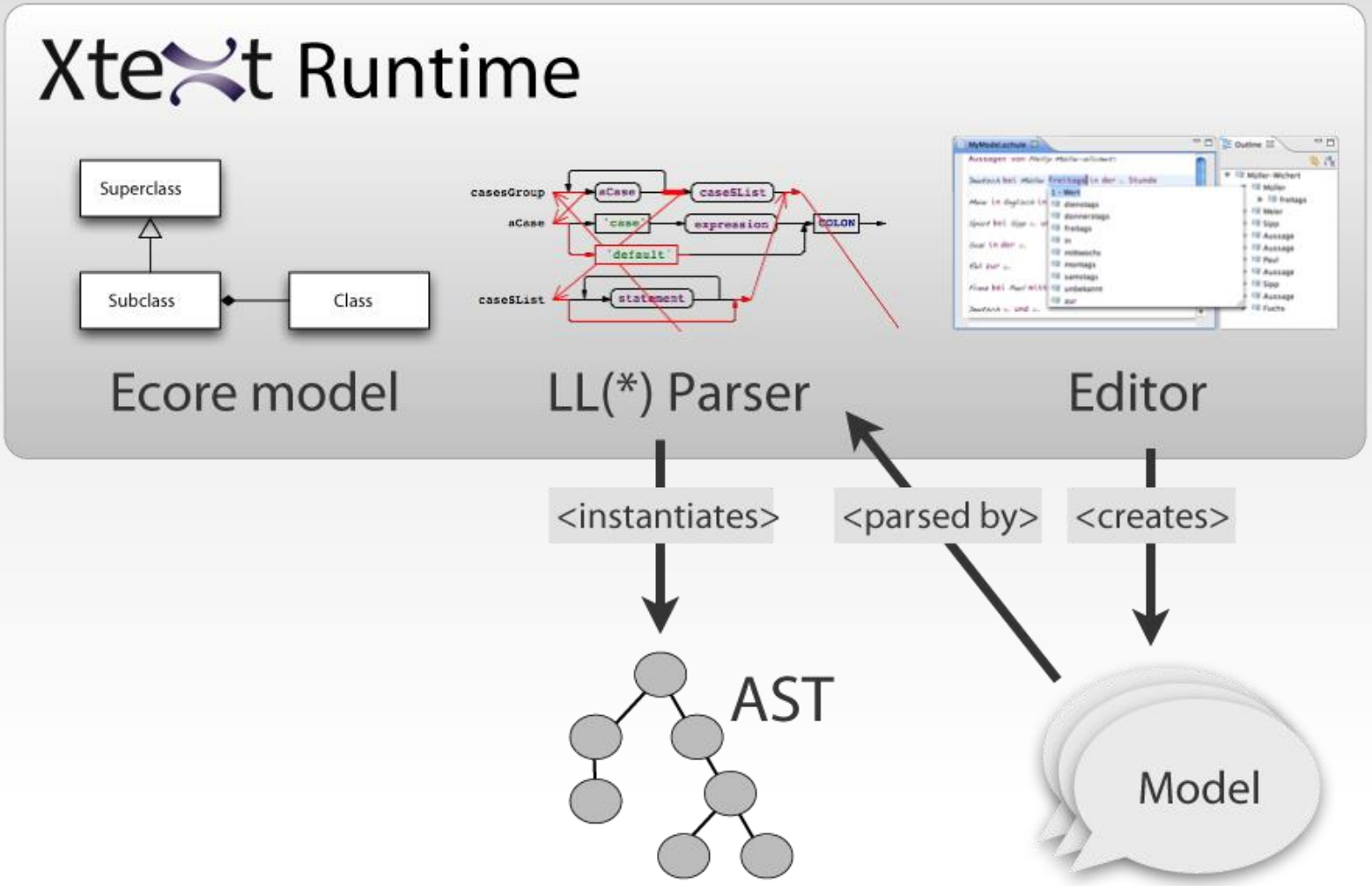
Xtext - Aufbau und Runtime



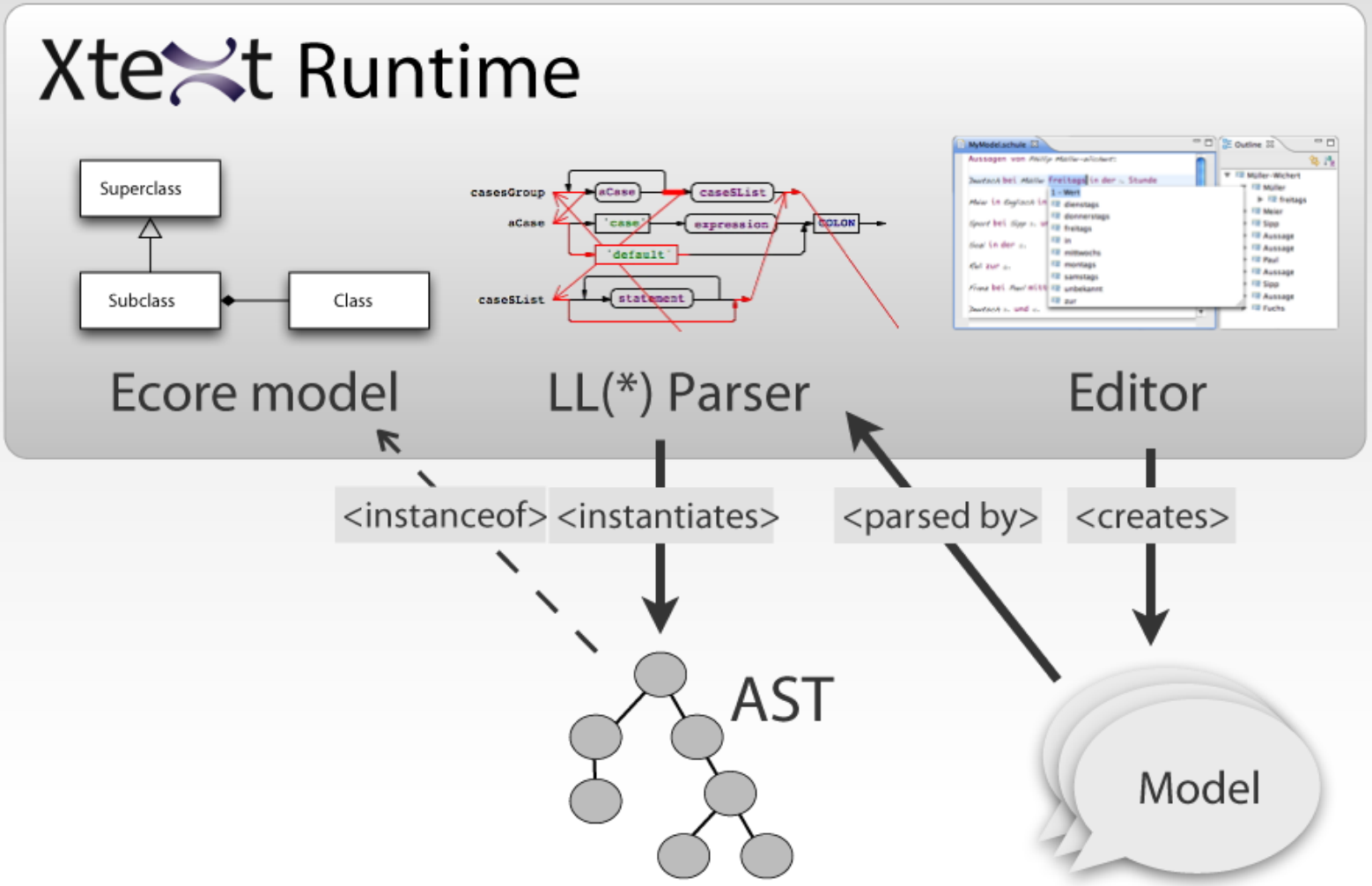
Xtext - Aufbau und Runtime



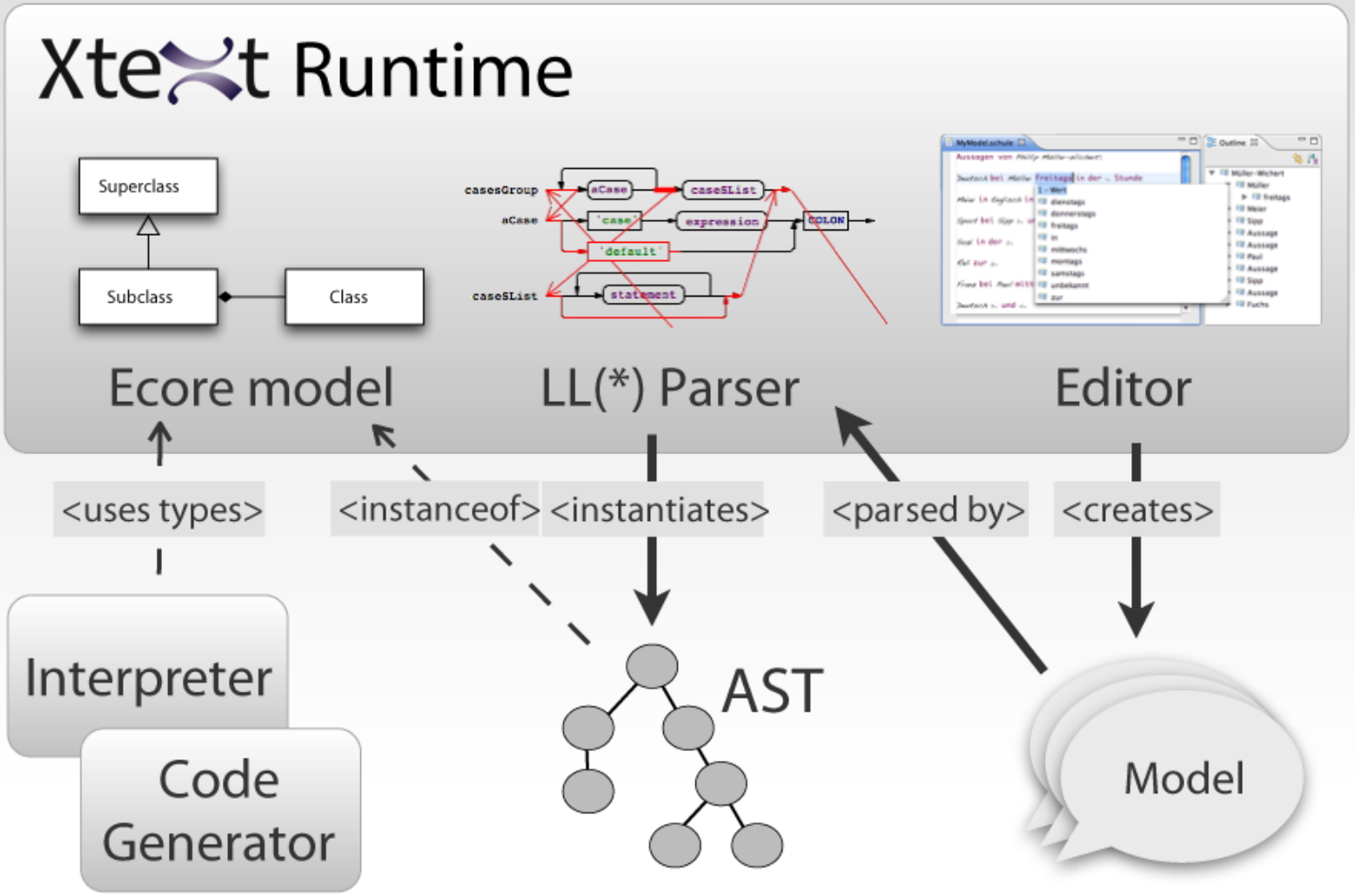
Xtext - Aufbau und Runtime



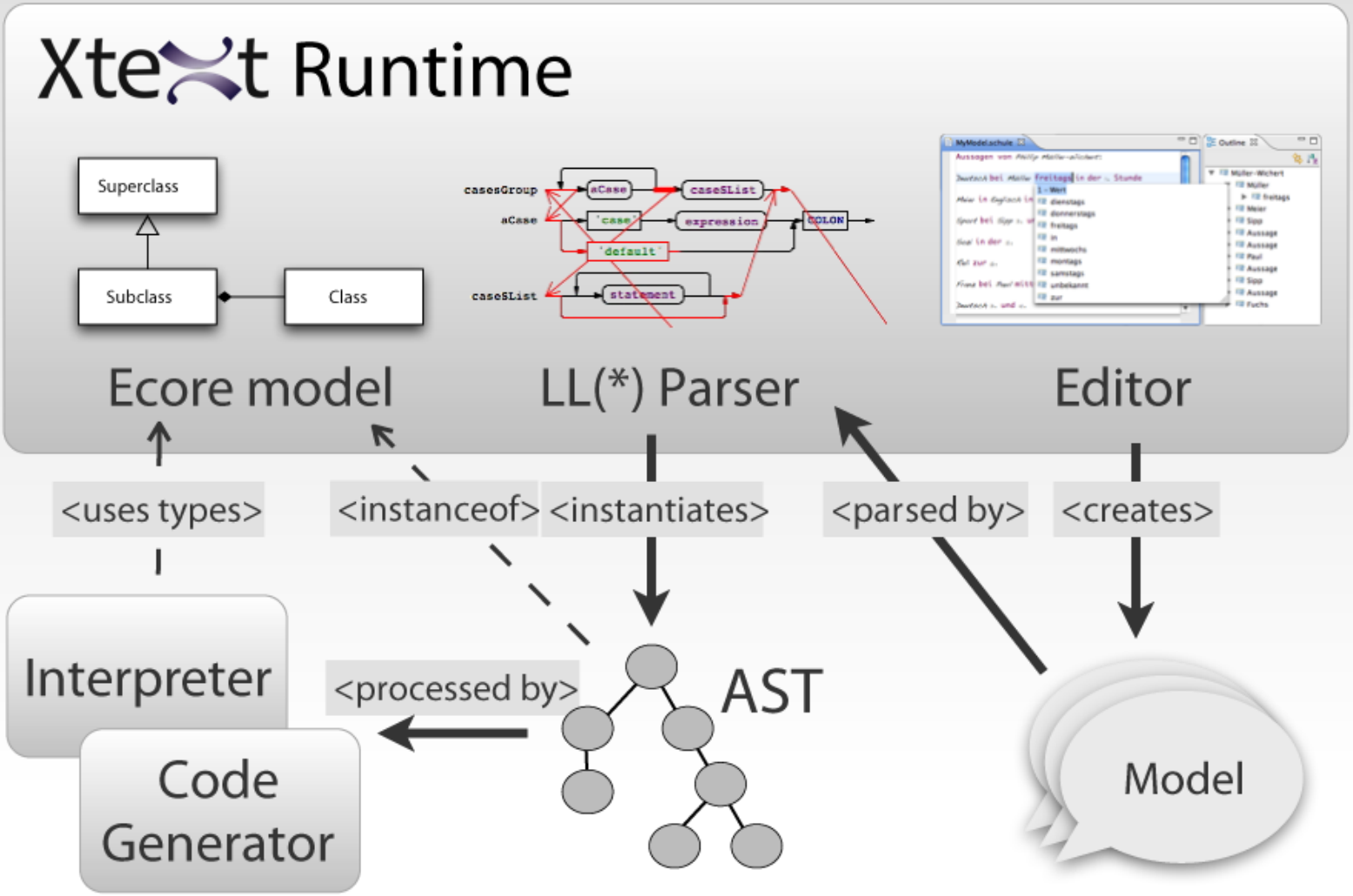
Xtext - Aufbau und Runtime



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

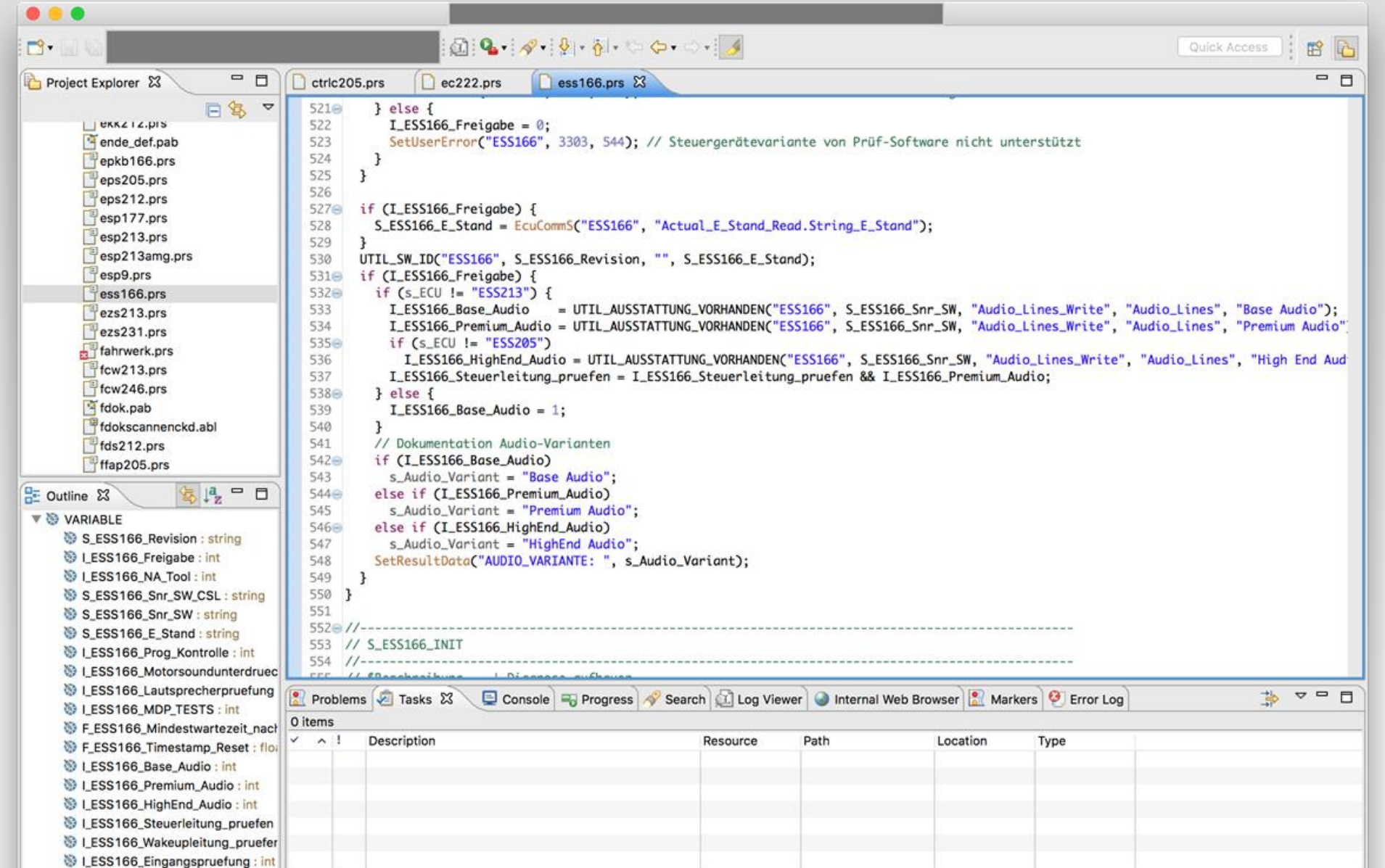
Kunden

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)



Programmablaufsprache (PAS)

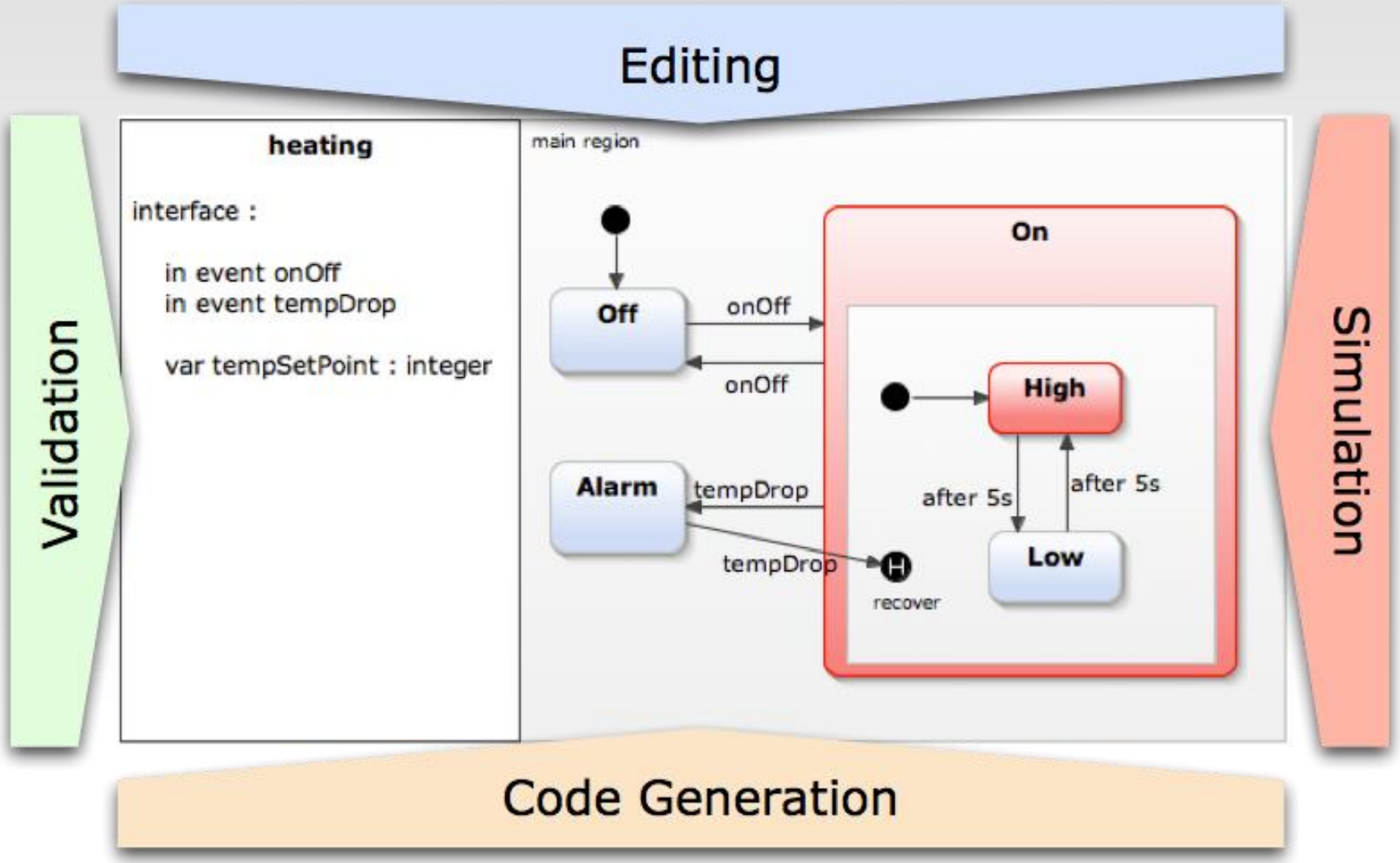
```
  if (I_ESS166_Freigabe) {
    if (s_ECU != "ESS213") {
      I_ESS166_Base_Audio    = UTIL_AUSSTATTUNG_VORHANDEN("ESS166", S_ESS166_Snr_SW, "Audio_Lines
      I_ESS166_Premium_Audio = UTIL_AUSSTATTUNG_VORHANDEN("ESS166", S_ESS166_Snr_SW, "Audio_Lines
      if (s_ECU != "ESS205")
        I_ESS166_HighEnd_Audio = 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

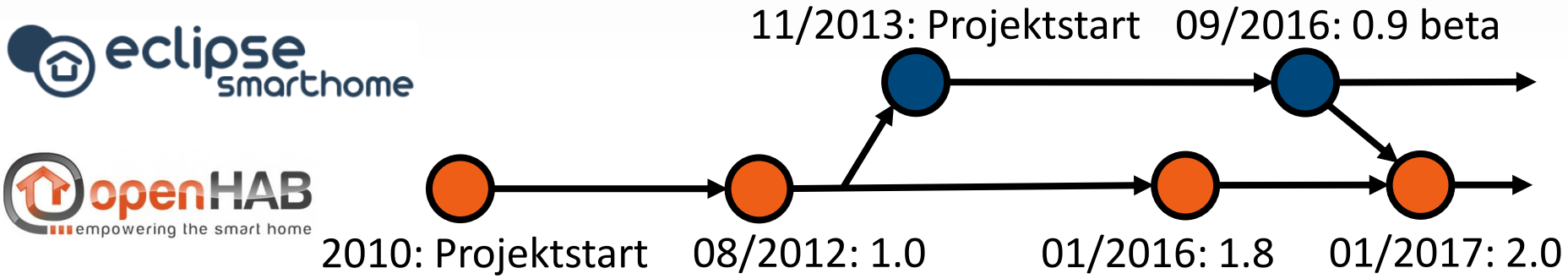


YAKINDU

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 Scenarios mittels Regeln



Szenario

Morgenroutine

Wenn

- es 7:00 Uhr ist

dann

- mach das Licht an,
- starte die Kaffeemachine,
- heize das Badezimmer,
- öffne die Rolläden,
- mach das Radio an,
- …

aber nur

- während der Woche.

Szenario

Morgenroutine

Wenn

- es 7:00 Uhr

dann

- mach das Li
- starte die Kaffeemach
- heize das Ba
- öffne die Ro
- mach das Radio an,
- …

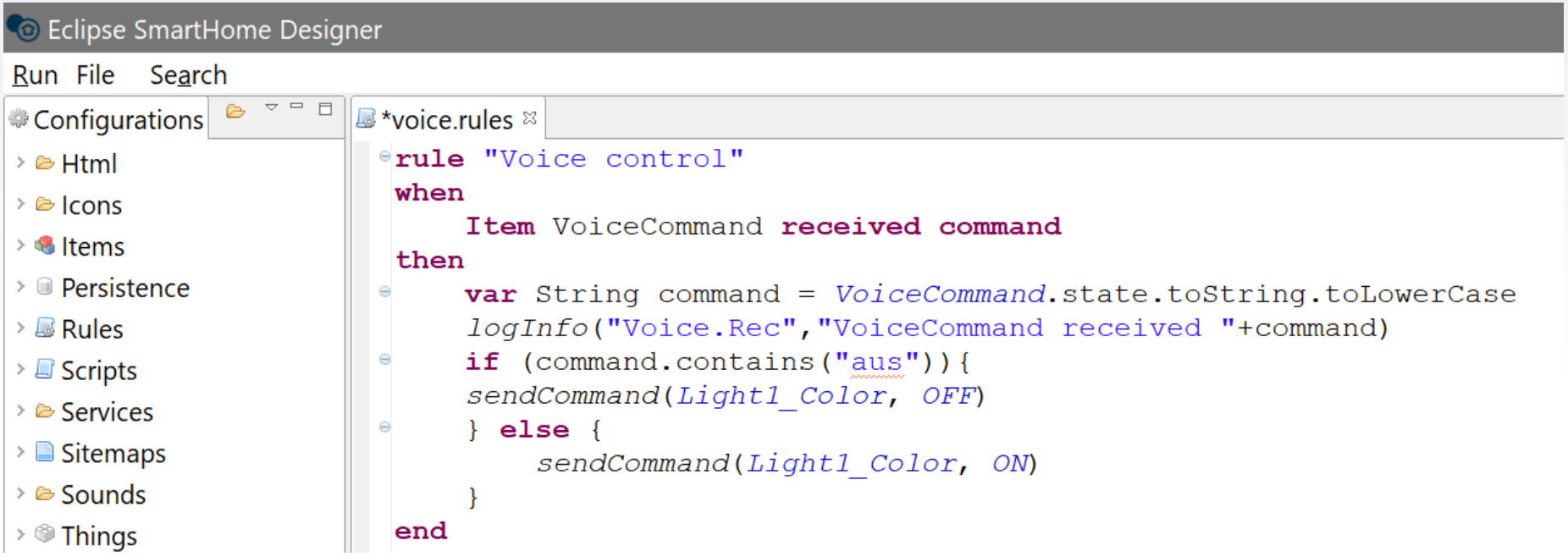
aber nur

- 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



openHAB & Xtext

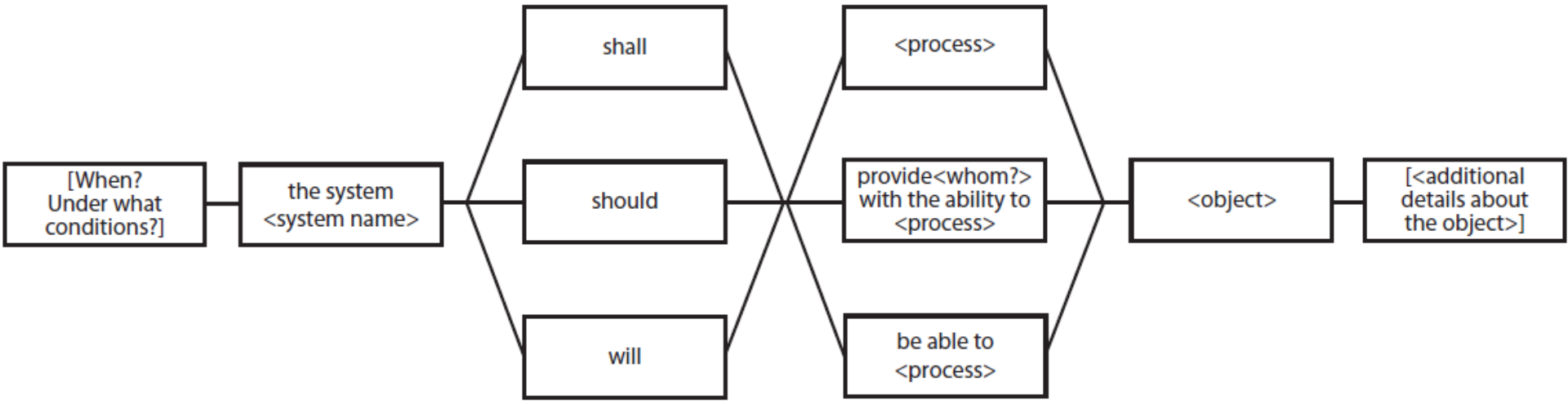
```
val initial = Light1_Color.state as HSBType
val initialBrightness = initial.getBrightness.intValue
val stepWith = 30
val fixedSaturation = 100

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
    }
    val 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

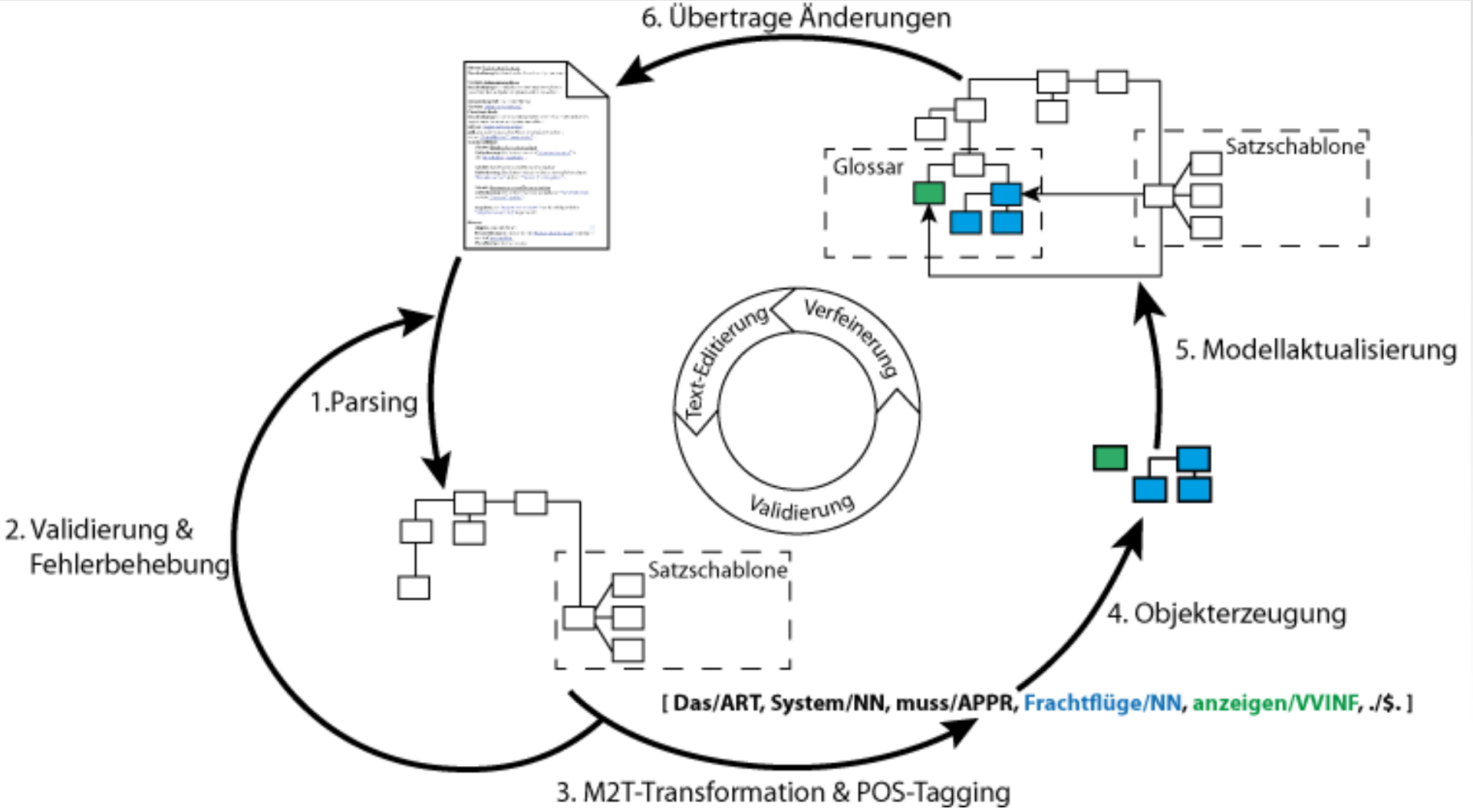
Semi Formal Requirements elecitation language

Sefo-rel

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

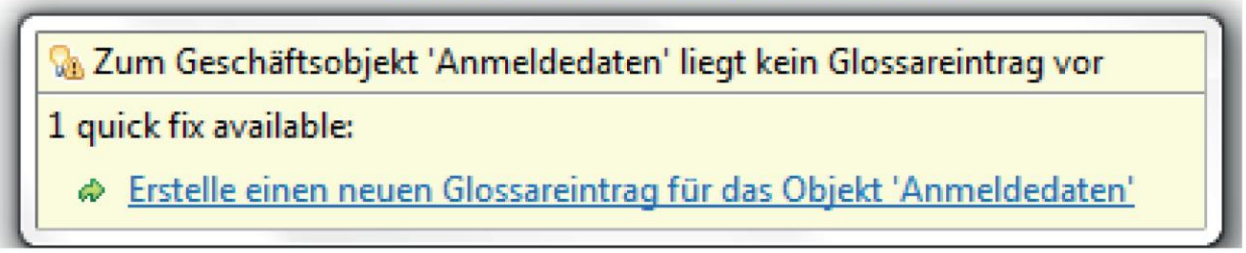


Sefo-rel


6. Übertrage Änderungen
Glossar
Satzschablone
5. Modellaktualisierung
Text-Editierung
Verfeinerung
Validierung
1.Parsing
2. Validierung & Fehlerbehebung
Satzschablone
4. Objekterzeugung
[Das/ART, System/NN, muss/APPR, Frachtflüge/NN, anzeigen/VVINF, ./$.]
3. M2T-Transformation & POS-Tagging

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 ermoeglichen* sich mit seinen Anmeldedaten "anzumelden".

Glossar:



Anwendungsfall: Login durchführen
System: "Aufgabenverwaltung"
Prioritaet: Hoch
Beschreibung: Dieser Anwe ndungsfall beschreibt wie sich ein bereits registrierter Benutzer am System anmeldet .
Akteur: "Registrierter Benutzer"
Ziel: *Das System muss dem Akteur ermoeglichen* 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 ermoeglichen* 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 ermoeglichen* seinen "Benutzernamen" und sein "Passwort" "einzugeben" .

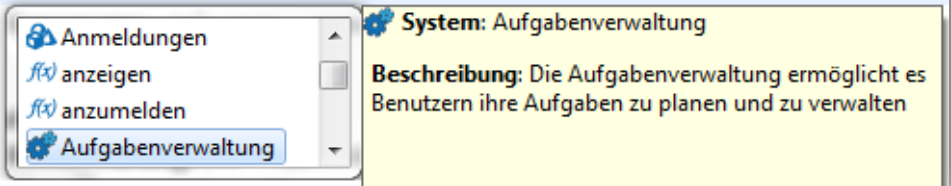
Schritt: Benutzername und Passwort prüfen
Anforderung: *Das System muss* den eingebenen "Benutzernamen" und das "Passwort" "prüfen" .

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: Anmeldedatums
Eigenschaften: "Passwort" : **EINS**, "Benutzername": **EINS**

Funktion: anmelden
Beschreibung: Ein "Registrierter Benutzer" meldet sich an der



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