

**Problemseminar**  
**Model-Driven Software Development**

**Vergleich von Metametamodellen**

Marcel Hoyer  
Matrikelnummer: 9366338  
März 2006

# Inhalt

|     |  |    |
|-----|--|----|
| 1   | Einleitung .....                                     | 3  |
| 1.1 | Aufgabenstellung .....                               | 6  |
| 1.2 | Bedeutung von Metametamodellen.....                  | 6  |
| 2   | Begriffe.....  | 7  |
| 2.1 | Systeme und deren Modell.....                        | 7  |
| 2.2 | Metamodell.....                                      | 8  |
| 2.3 | Metametamodell.....                                  | 8  |
| 2.4 | Semantik und Syntax.....                             | 9  |
| 2.5 | Domain Specific Language .....                       | 9  |
| 2.6 | Technical Space.....                                 | 10 |
| 3   | Überblick über Metamodellierungssprachen .....       | 11 |
| 3.1 | Meta-Object Facility .....                           | 11 |
| 3.2 | Ecore.....   | 12 |
| 3.3 | MetaGME.....   | 13 |
| 3.4 | XML Schema .....                                     | 15 |
| 3.5 | Weitere Metametamodelle .....                        | 17 |
| 4   | Vergleich der vorgestellten Metametamodelle .....    | 20 |
| 4.1 | Komponenten .....                                    | 20 |
| 4.2 | Vergleich bezüglich Mächtigkeit und Praxisbezug..... | 21 |
| 5   | Fazit.....   | 22 |

## Abbildungsverzeichnis

|  |    |
|--|----|
| Abbildung 1 – System, Modell, Metamodell, Metametamodell (Quelle: [KUEHNE05]) .....  | 7  |
| Abbildung 2 – XML als Metametamodell .....   | 9  |
| Abbildung 3 – Technische und Modellierungsräume (Quelle: [DjGaDe05]) .....           | 10 |
| Abbildung 4 – MOF Metametamodell (Quelle: [OMGMOF02]) .....                          | 11 |
| Abbildung 5 – Ecore Metametamodell (Quelle: [EMFOV]) .....                           | 12 |
| Abbildung 6 – MetaGME (Quelle: [LeMaBa01]) .....                                     | 13 |
| Abbildung 7 – Komponentenmodell von XML Schema (Quelle: [W3C]) .....                 | 15 |
| Abbildung 8 – Vereinfachtes Microsoft DSL Metametamodell (Quelle: [BEZI05DSL]) ..... | 17 |
| Abbildung 9 – EMOF Metametamodell (Quelle: [FLEUREY]) .....                          | 18 |
| Abbildung 10 – KM3 Metametamodell (Quelle: [BEZI05DSL]) .....                        | 19 |

## **Tabellenverzeichnis**

|   |    |
|---|----|
| Tabelle 1 – MMM Komponenten für die Objektabbildung .....                           | 20 |
| Tabelle 2 – MMM Komponenten für Relationen.....                                     | 20 |
| Tabelle 3 – MMM Komponenten für eine Strukturierung des Metamodells.....            | 20 |
| Tabelle 4 – MMM Komponenten für Datentypen .....                                    | 20 |
| Tabelle 5 – MMM Komponenten für Einschränkungen, Sichten und Aspekte.....           | 21 |
| Tabelle 6 – Spezielle MMM Komponenten.....  | 21 |
| Tabelle 7 – Vergleich der MMM bezüglich Mächtigkeit und Werkzeugunterstützung ..... | 21 |

## Abkürzungsverzeichnis

|               |  |
|---------------|--|
| BPEL4WS ..... | Business Process Execution Language for Web Services |
| CWM.....      | Common Warehouse Metamodel                           |
| DSL .....     | Domain Specific Language                             |
| EMF.....      | Eclipse Modeling Frameworks                          |
| EMOF .....    | Essential MOF  |
| FCO .....     | First Class Objects                                  |
| GME .....     | Generic Modeling Environment                         |
| KM3 .....     | Kernel Metametamodell                                |
| IDL .....     | Interface Definition Language                        |
| MMM .....     | Metametamodell                                       |
| MOF .....     | Meta-Object Facility                                 |
| OCL.....      | Object Constraint Language                           |
| OMG.....      | Object Management Group                              |
| UML .....     | Unified Modeling Language                            |
| W3C .....     | World Wide Web Consortium                            |
| XMI .....     | XML Metadata Interchange                             |
| XML .....     | Extensible Markup Language                           |
| XSD.....      | XML Schema Document                                  |

# 1 Einleitung

## 1.1 Aufgabenstellung

Ziel der Untersuchung ist die Erstellung einer Übersicht über Metamodellierungssprachen und deren Modellierungskonzepte. Dazu sollten zunächst die für die Beschreibung notwendigen Begriffe erläutert werden. Darauf basierend kann eine konkrete Untersuchung an existierenden Metamodellierungssprachen und deren Vergleich stattfinden.

## 1.2 Bedeutung von Metametamodellen

Grundsätzlich bilden Metametamodelle eine weitere Abstraktionsschicht für Metamodelle. Diese Abstraktion stellt Möglichkeiten zur Verfügung zwischen unterschiedlichen Metamodellen Beziehungen aufzubauen und eine gemeinsame Grundlage für die Metamodellgenerierung zu schaffen. Mit Hilfe dieser Generalisierung erlauben Metametamodelle das Überbrücken (*Bridging*) von Technical und Modeling Spaces. Zum Beispiel kann unter Verwendung von XMI eine mögliche Transformation zwischen Metamodellen über XML als Technical Space vollzogen.

Im Bereich der modellgetriebenen Softwareentwicklung können somit Transformationen zwischen verschiedenen Modellen und Metamodellen erreicht werden. Vor allem in Hinsicht auf die Überbrückung von technischen Räumen werden vorteilhafte Voraussetzungen zum Beispiel für das Reverse Engineering geschaffen.

## 2 Begriffe

Für die Bearbeitung der Aufgabe ist es zunächst notwendig auf einige Begriffe, welche sich mit dem Thema Metametamodellierung befassen, einzugehen. Diese werden im Folgenden definiert, um die in der Literatur verwendeten, teilweise verschiedenen Deutungen für die Ausarbeitung zu vereinheitlichen.

### 2.1 Systeme und deren Modell

Auf dem Gebiet der Informatik und in anderen Fachbereichen ist es notwendig Systeme schematisch abzubilden, um sie besser verstehen und analysieren zu können. Als *Systeme* kann man in diesem Zusammenhang real existierende Strukturen, wie zum Beispiel Prozessabläufe, Wissen oder Landschaften verstehen. Da sich ein komplettes System nur schwer in seiner vollen Gesamtheit erfassen lässt wird es auf bestimmte, zu untersuchende Bereiche eingeschränkt. Diese Teilauszüge werden mit dafür optimierten *Modellen* dargestellt. Ein Modell kann also als Mittel verstanden werden, mit dessen Hilfe es möglich ist ein reales System bezüglich bestimmter Gesichtspunkte zu beschreiben. Abbildung 1 zeigt im unteren Bereich die Einordnung und den Zusammenhang von System und

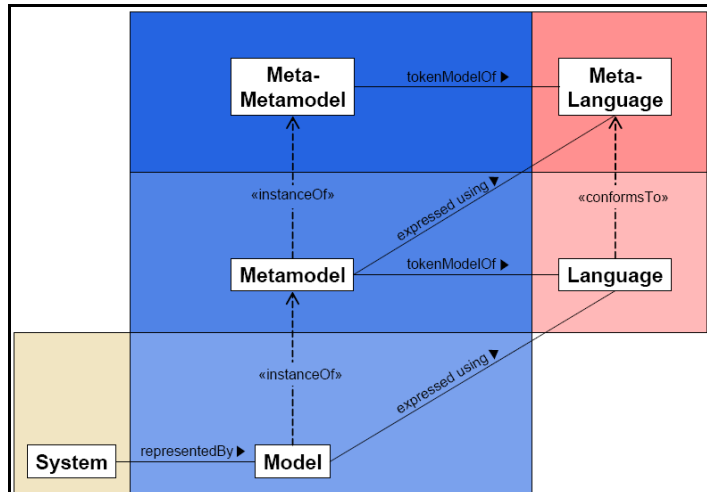


Abbildung 1 – System, Modell, Metamodell, Metametamodell  
(Quelle: [KUEHNE05])

Modell.

Da ein Modell bei der Abbildung nur bestimmte *Aspekte* betrachtet, kann es verschiedene Modelle zu ein und demselben System geben. Diese können durch einen mehr oder weniger ausgeprägten Detaillierungsgrad unterschiedliche Abstraktionsstufen darstellen.

## 2.2 Metamodell

Nach [JECKLE00] ist ein *Metamodell* „ein Modell, das ein Modell beschreibt“. Prinzipiell kann ein Metamodell aus einem konkret existierenden Modell erstellt werden. Dieses Metaisierungsprinzip drückt eine gewählte Abstraktionsebene über der tatsächlichen Modellebene aus.

Die Elemente (Wörter) zum Erstellen eines Modells werden durch diese *Metamodelle* festgelegt. Der Modellierer benötigt für den Abbildungsvorgang des Systems eine bestimmte Menge zuvor definierter Komponenten. Die Art und Weise der Zusammensetzung dieser bildet das eigentliche Modell und repräsentiert letztendlich das nachgebildete System unter Betrachtung eines gewissen Gesichtspunktes. Ein Metamodell bietet also die syntaktischen Grundlagen für eine *Sprache*, die verwendet wird, um Modelle zu beschreiben. Wie bei dem zu erstellenden Modell, ist auch dessen benutzte Sprache auf eine bestimmte Betrachtungsweise ausgerichtet. Der Wörterumfang des Metamodells drückt somit die im Modell verwendeten Möglichkeiten zur Abbildung des Systems aus.

Wie in [BEZI05] soll zum besseren Verständnis das Beispiel verschiedenartiger geographischer Karten angebracht werden. Durch eine gegebene Legende ist es möglich, Karten, welche eine real existierende Landschaft abbilden zu erstellen und zu lesen. Dabei stellen Landkarten jeglicher Art (zum Beispiel wirtschaftliche oder topografische) eine Abbildung – also ein Modell – der real existierenden Umgebung dar. Das Metamodell wird durch die Elemente der Legende geprägt. Die Landschaft kann hierbei als System verstanden werden.

## 2.3 Metametamodell

Verfolgt man das in Kapitel 2.2 gegebene Landkartenbeispiel stellt sich die Frage, in welcher Form die Legenden der Karten beschrieben werden können. Auch die Form der Legendendarstellung kann in diesem konkreten Beispiel definiert werden. Da diese Beschreibung in sich erneut eine Sprache darstellt, wird diese als *Metasprache* bezeichnet – also als Sprache, mit der eine Sprache beschrieben werden kann. In Bezug auf Metamodelle bilden so genannte *Metametamodelle* die modellhafte Abbildung einer solchen Metasprache.

Theoretisch lässt sich dieses Metaisierungsprinzip weiter fortsetzen, so dass durch mehr und mehr Abstraktionsschichten weitere Metametamodelle erstellt werden können. Außerdem können Metasprachen in der Lage sein sich selbst erneut zu definieren. Der Zusammenhang zwischen Modell, Metamodell und Metametamodell wird nun mit Hilfe des Beispiels XML besser verdeutlicht.

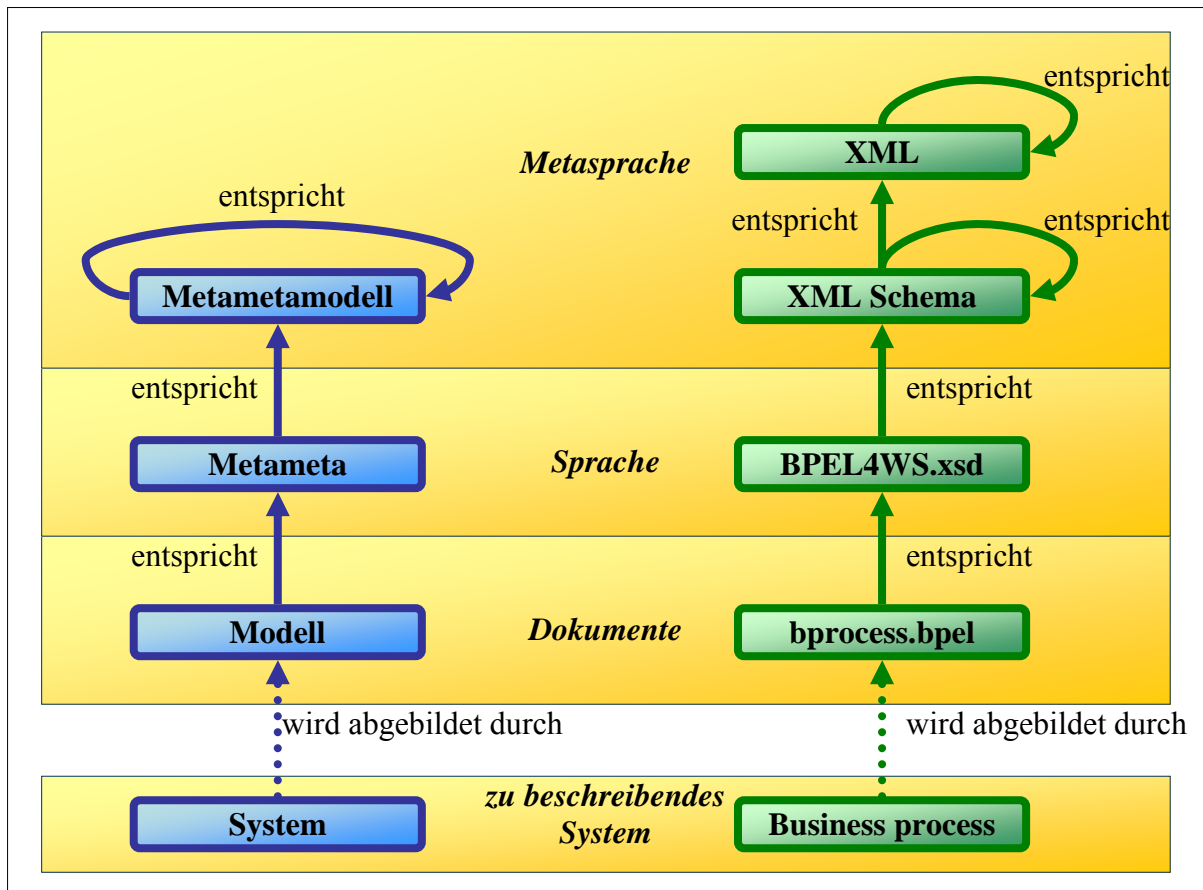


Abbildung 2 – XML als Metametamodell

Zu Beginn soll XML als oberstes Metametamodell gelten. Dabei stellt XML die „Metasprache“ dar, mit deren Hilfe XML Dokumente der Syntax folgend erzeugt werden können. In Abbildung 1 ist zu erkennen, dass XML Schema als Metamodell entstehen kann. Die vom W3C verabschiedete Spezifikation für diese Modellierungssprache kann nun in sich wiederum als Metasprache betrachtet werden, mit deren Hilfe erneut Metamodelle als Instanzen des XML Schema generiert werden können. Dieses Beispiel verdeutlicht diesen Vorgang mit der bereits existierenden Spezifikation von BPEL4WS. Die dort beschriebenen Elemente und Strukturierungsmöglichkeiten bilden die Voraussetzung, um valide BPEL4WS Dokumente als Modell eines real existierenden Geschäftsprozesses anzufertigen.

## 2.4 Semantik und Syntax

Als Bestandteile formaler Modellierungssprachen spielen Semantik und Syntax auch im Rahmen der Metametamodelle eine wichtige Rolle. Dabei versteht man unter *Semantik* die definierte Bedeutung der Sprachkomponenten. Die *Syntax* liefert die dazugehörige Grammatik und kann weiterhin in abstrakte und konkrete Syntax unterteilt werden. Als *abstrakte Syntax* bezeichnet man die Grammatik – also die Sprachkonzepte und die Art und Weise, wie diese strukturiert und kombiniert werden können. Die *konkrete Syntax* liefert die dazu gehörigen Notationsvorschriften und gibt somit an, in welcher Form die Modellierung zu erfolgen hat. Man kann hierbei zwischen grafischer und textueller Syntax unterscheiden.

## 2.5 Domain Specific Language

Als *Domäne* versteht man bei *Domain Specific Languages (DSL)* den Bezug einer Sprache zu einem bestimmten Problembereich. Dabei stehen spezifische Fachbereiche im Vordergrund

und bilden die Grundlage der DSLs. Die Sprache ist also in ihrer Syntax und Semantik auf ein bestimmtes Anwendungsgebiet eingeschränkt und daraufhin optimiert.

Im Landkartenbeispiel bilden die verschiedenen Legenden jeweils eine DSL. So gehört die Wanderkartenlegende einer anderen Domäne an als Legenden von Straßen- oder Flusskarten.

## 2.6 Technical Space

*Technische Räume* können als Umgebung verstanden werden, in der Metametamodelle, Metamodelle und Modelle beschrieben sind, die auf der gleichen technischen Ebene einer konkreten Syntax basieren. Technische Ebene bedeutet in diesem Zusammenhang die technologische Grundlage in der die jeweiligen Modelle beschrieben sind. Solche technical Spaces können zum Beispiel XML Dokumente, Sprachkonstrukte von Java oder die modellgetriebene Architektur des OMG sein.

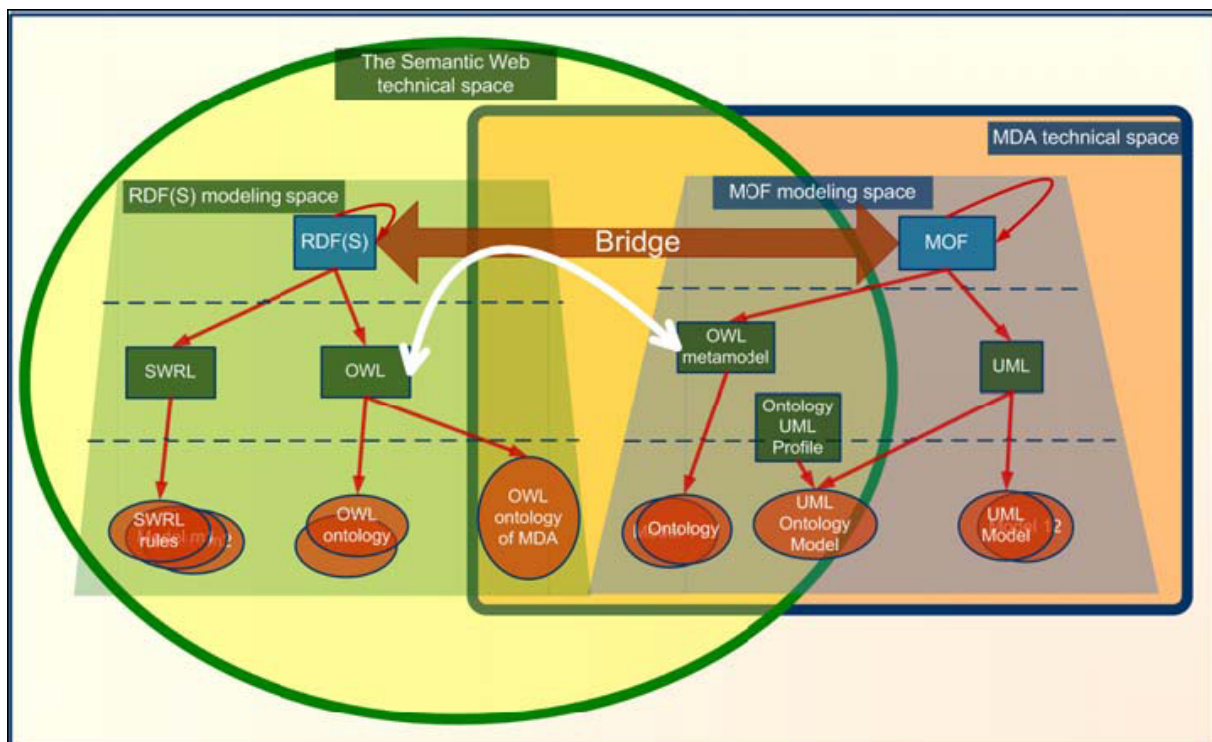


Abbildung 3 – Technische und Modellierungsräume (Quelle: [DjGaDe05])

Neben Technischen Räumen können auch Modellierungsräume (*Modeling Spaces*) als Abgrenzung definiert werden. Diese kreisen die zusammenhängenden Modelle, Metamodelle und Metametamodelle ein. In Abbildung 3 wird dieser Zusammenhang an einem Beispiel dargestellt.

### 3 Überblick über Metametamodelle

Da Metametamodelle möglichst generisch angelegt sein sollten ist deren Komponentenmenge begrenzt und wird sich bei den folgenden Vertretern, abgesehen von unterschiedlichen Bezeichnungen, oft überschneiden. Diese Spezifikationen, erfahren bereits starken praktischen Nutzen durch darauf aufbauende Metamodelle und Sprachumsetzungen. Vier aktuelle Metametamodelle sollen in diesem Kapitel vorgestellt werden, um sie dann in Kapitel 4 miteinander zu vergleichen.

#### 3.1 Meta-Object Facility

Die MOF Spezifikation ist eine Entwicklung der OMG (Object Management Group) und wurde seit Ende der 90er Jahre entwickelt. Zurzeit wird an der zweiten Version gearbeitet. Die Spezifikation beinhaltet zum einen die Beschreibung des MOF Metametamodells zur Spezifizierung, Erstellung und Verwaltung von technologie-neutralen Metamodellen. Zum anderen wird eine IDL Umsetzung als Framework zur Verfügung gestellt, mit deren Hilfe es möglich ist, MOF-konformen Metamodellen in unterschiedlichen Sprachen zu erstellen.

Die MOF Spezifikation hat einen bedeutenden Wert in der heutigen modellgetriebenen Softwareentwicklung. Das wohl wichtigste Metamodell liegt der UML Sprache zugrunde.

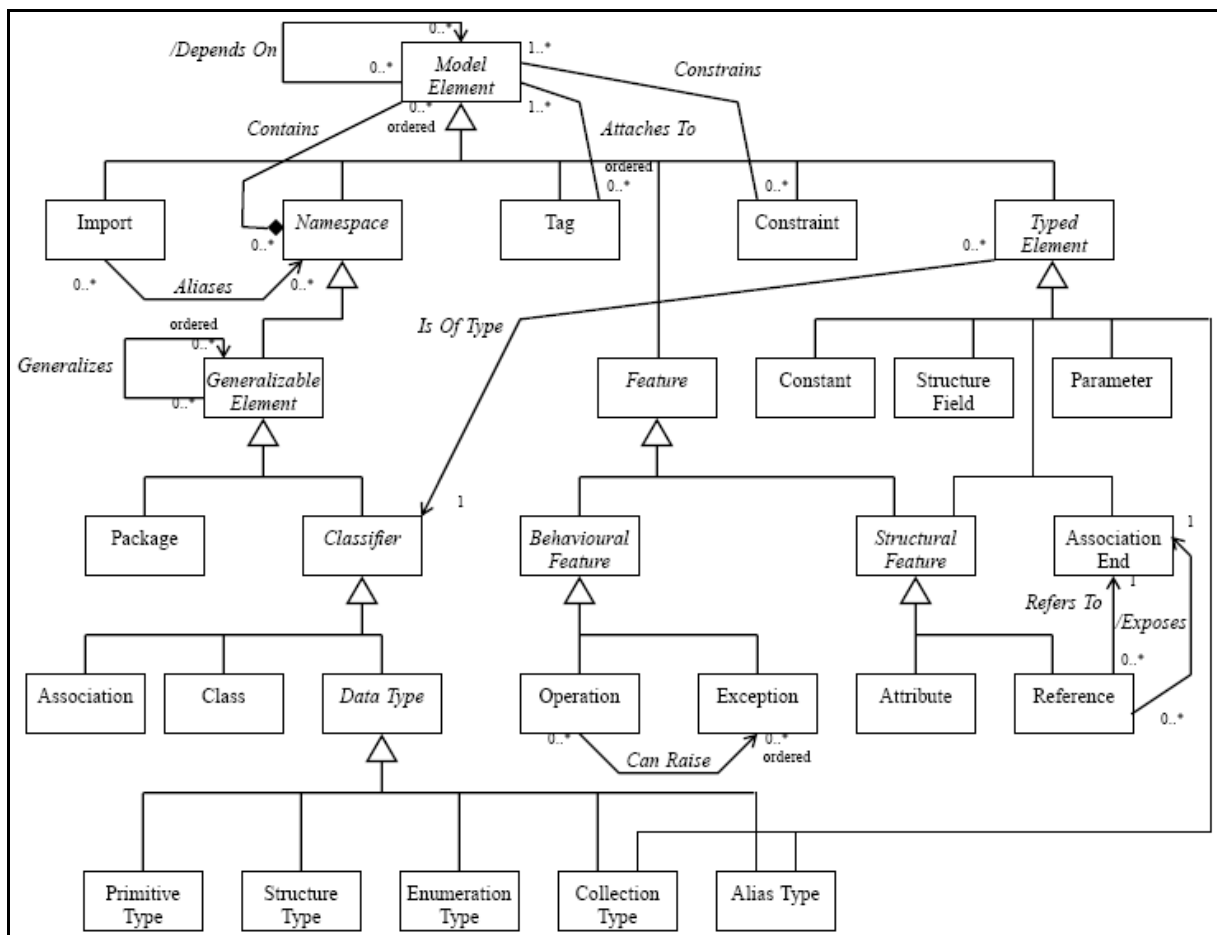


Abbildung 4 – MOF Metametamodell (Quelle: [OMGMOF02])

Neben den generischen Elementen zur Bildung von Klassenhierarchien (Class, Association) bietet das Metametamodell zusätzliche Komponenten zur Gruppierung und Klassifizierung (Package, Namespace, Tag, Classifier), sowie zur verfeinerten Beschreibung der zu

erstellenden Metamodellobjekte (Operation, Exception, Attribute, Reference). Beziehungen zwischen Klassen werden durch Referenzen und Assoziationen abgebildet. Außerdem können Einschränkungen (Constraints) mit Hilfe von OCL (Object Constraint Language) sowie strukturierte Datentypen definiert werden. Abbildung 4 zeigt den konzeptionellen Zusammenhang des MOF.

### 3.2 Ecore

Im Rahmen des Eclipse Modeling Frameworks (EMF) bildet *Ecore* das zugrunde liegende Metametamodell dieser Werkzeugsammlung. Ecore ist jünger und besitzt eine geringere Komponentenmenge als das in Kapitel 3.1 vorgestellte MOF. Allerdings wurden bei der Entwicklung von Ecore auch Konzepte des MOF verwendet.

Zu bemerken ist, dass Relationen (EReference) nur unidirektional erstellt werden können. Für eine bidirektionale Verwendung müssen zwei Referenzen erstellt werden. Außerdem lassen sich keine Einschränkungen bezüglich der zu modellierenden Komponenten erstellen.

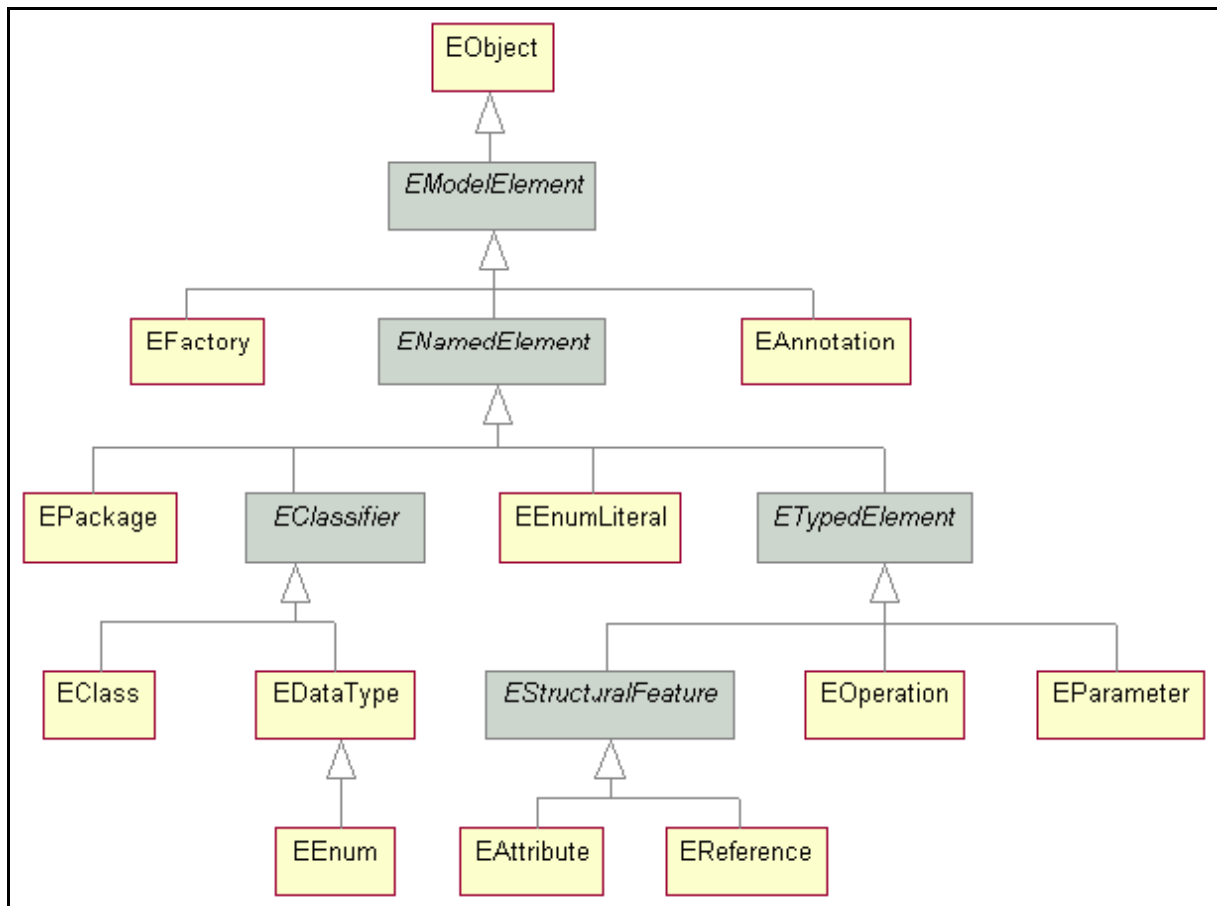


Abbildung 5 – Ecore Metametamodell (Quelle: [EMFOV])

Ecore kommt in den Editoren des EMF zum Einsatz, welche als Plugin für die Eclipse Plattform zur Verfügung stehen. Dabei kann mit Hilfe von UML oder XMI auf Ecorebasis modelliert werden.

### 3.3 MetaGME

Das Generic Modeling Environment (GME) wurde am Institute for Software Integrated Systems der Vanderbilt Universität entwickelt und stellt ein Werkzeugset zur Verfügung, mit dessen Hilfe es möglich ist domänenspezifisch Modelle zu entwerfen. Die Konfiguration dieser Werkzeuge erfolgt durch Verwendung von Metamodellen, die ebenfalls mit dem Framework modelliert werden können. Dabei greift die Entwicklungsumgebung auf das in dem Framework integrierte Metametamodell *MetaGME* zurück. Die Metamodelle werden mit Editoren des GME implementiert und können nach dem so genannten Modellinterpretationsprozess in der GME Entwicklungsumgebung zum Modellieren der domänenspezifischen Modelle verwendet werden.

GME stellt somit – ähnlich dem EMF – ein komplettes Toolkit zur Verfügung, um MetaGME basierte Metamodelle zu entwerfen, welche in der gleichen Umgebung als Basis für den eigentlichen Modellierungsprozess benutzt werden können.

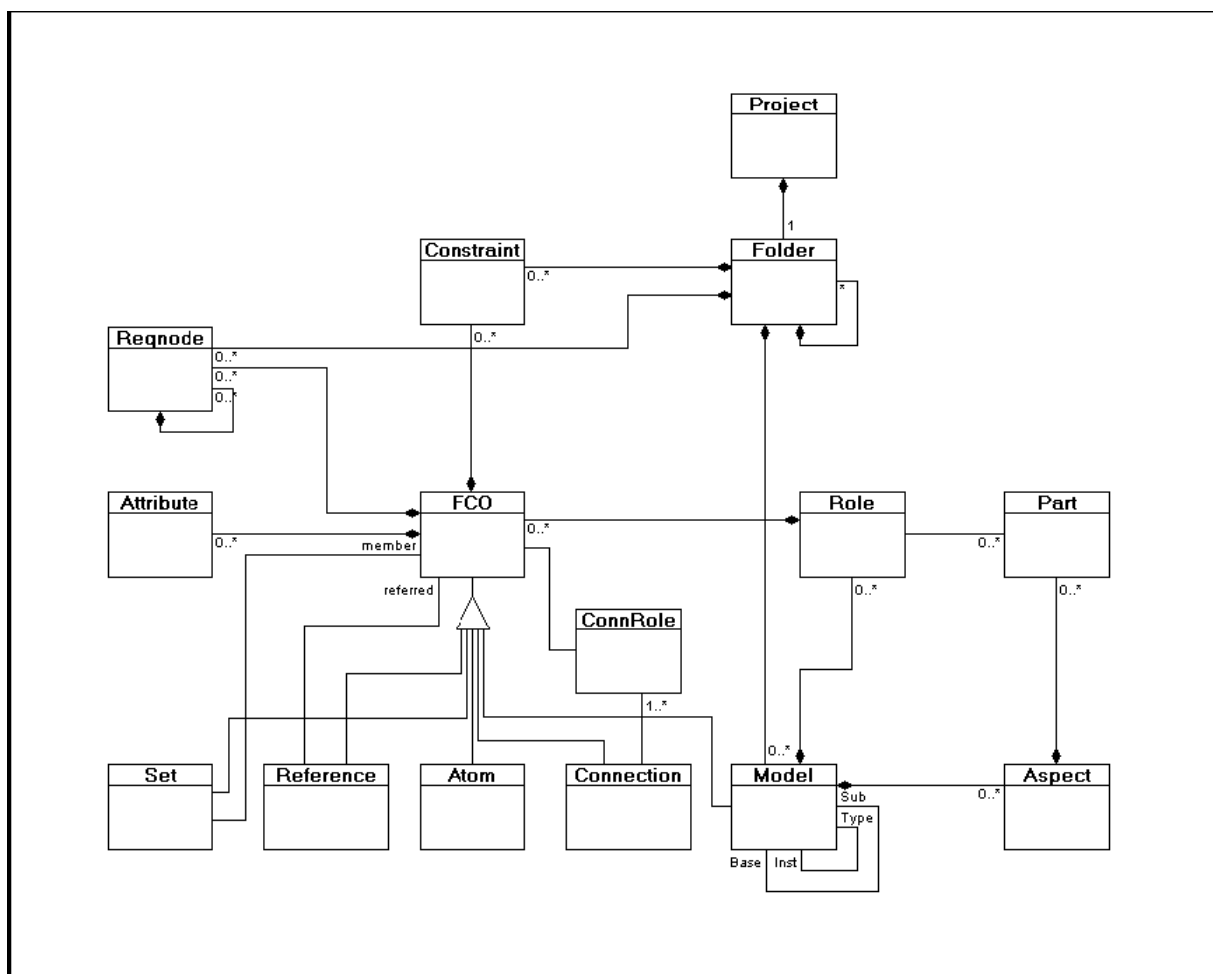


Abbildung 6 – MetaGME (Quelle: [LeMaBa01])

Betrachtet man das Metametamodell (vgl. Abbildung 6) von GME, so ist erkennbar, dass im Vergleich zu Ecore und MOF eine nicht so starke Hierarchie der Elemente vorhanden ist. Im Mittelpunkt des MetaGME stehen die First Class Objects (FCO). Dabei ist zu bemerken, dass das Element Model mit der Komponente Class (MOF) oder EClass (Ecore) gleichgesetzt werden kann. Das Atom bildet die kleinste Einheit im Metametamodell und kann – abgesehen von Attributen und Constraints – keine weiteren Elemente in sich aufnehmen. Zur Klassifikation von zu generierenden Metamodellteilen können die Elemente Project und

Folder benutzt werden. Hervorzuheben ist bei diesem Ansatz, dass sich mit Hilfe von Aspects, Roles und Parts verschiedene Sichten auf ein komplettes Metamodell beschreiben lassen.

Binäre Verbindungen (Connections) können verwendet werden, um gerichtete und ungerichtete Beziehungen zwischen Elementen im gleichen oder darunter liegenden Metamodell auszudrücken. Für Verweise auf Elemente anderer Metamodelle stehen References zur Verfügung. Mit Hilfe von Sets können Objekte gruppiert und als Ganzes als Referenzendpunkt verwendet werden. Zusätzlich bieten Constraints einen OCL basierten Ansatz, um Einschränkungen für alle FCOs zu implementieren.

Im Gegensatz zu MOF, Ecore und XML Schema werden bei MetaGME keine Datentypen vom Metametamodell angeboten.

### 3.4 XML Schema

Mit *XML Schema* steht ein generisches Metametamodell zur Verfügung, welches nicht mit der Hauptabsicht der Metamodellierung entwickelt wurde. Dennoch bietet es – gerade durch den abstrakten Ansatz – ideale Voraussetzungen für die Implementierung von Sprachen. In Kapitel 2.3 wurde bereits an Hand des BPEL4WS Beispiels gezeigt das bereits eine Vielzahl an Metamodellen existiert.

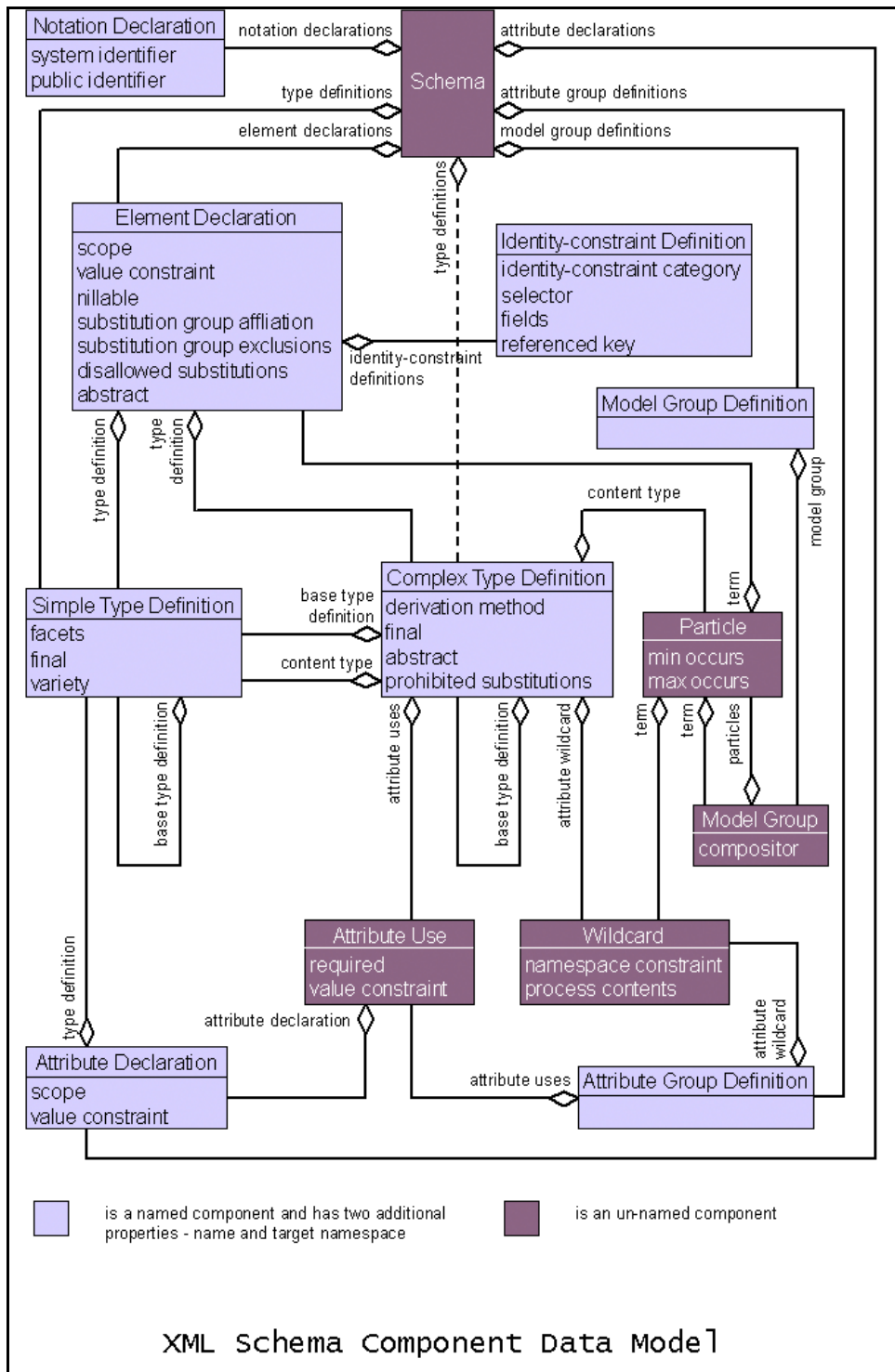


Abbildung 7 – Komponentenmodell von XML Schema (Quelle: [W3C])

In Abbildung 7 wird ersichtlich das Elemente und Attribute die Grundlage für die Erstellung von Metamodellen bilden. Die in XML Schema spezifizierten Komponenten idref, id, key und keyref schaffen die Möglichkeit der Implementierung von Relationen. Diese Elemente können gleichzeitig zur Modellierung von Einschränkungen verwendet werden. Dafür sind außerdem restriction, unique, field, selector und union hilfreich.

### 3.5 Weitere Metametamodelle

Neben den vier vorgestellten Metametamodellen existieren noch weitere, welche an dieser Stelle nur genannt werden sollen.

Die von Microsoft entwickelten *DSL Tools* bilden die Grundlage für die angestrebte Technologie der Software Factories. Mit Hilfe des in die Entwicklungsumgebung Visual Studio 2005 integrierten Werkzeugs bietet Microsoft die Möglichkeit domänenspezifische Sprachen zu entwickeln. Darauf basierend können Fachkompetenzen Modelle ihrer Systeme erstellen, welche mit einfachen Mitteln zur Codegenerierung und letztendlich zur Softwareentwicklung verwendet werden können. Abbildung 8 zeigt eine vereinfachte Version des Metametamodells.<sup>1</sup>

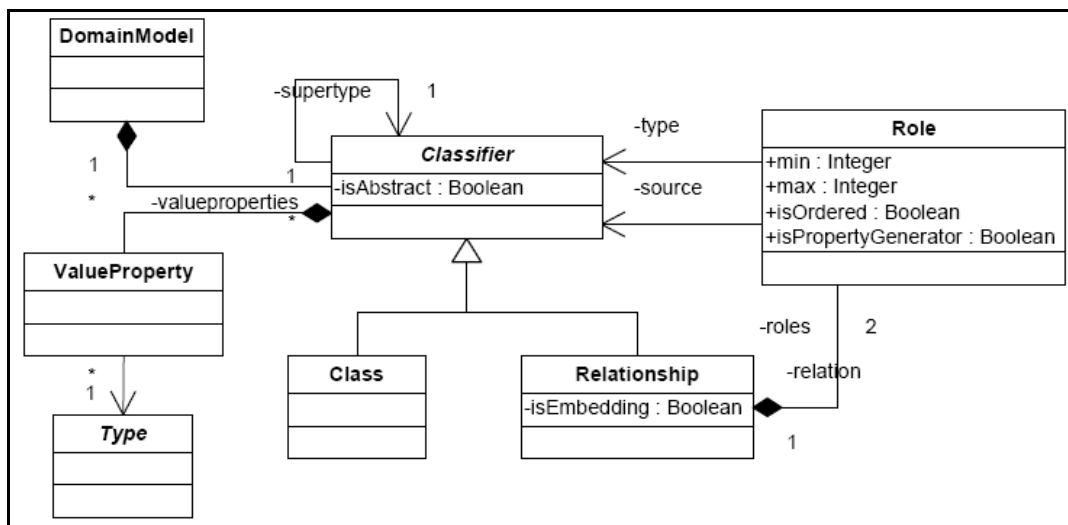


Abbildung 8 – Vereinfachtes Microsoft DSL Metametamodell (Quelle: [BEZI05DSL])

<sup>1</sup> Außer in [BEZI05DSL] konnten keine weiteren Metametamodellinformationen zu Microsoft DSL recherchiert werden.

Bei der aktuellen Version der MOF 2.0 Spezifikation wird das separate Metametamodell *EMOF* (Essential MOF) mitgeliefert. Die Elemente dieser Metasprache gleichen abgesehen von der Namensgebung annähernd denen des Ecore Metametamodells und sind damit komplett kompatibel. Durch EMOF soll eine Kopplung von MOF und Ecore erreicht werden. Abbildung 9 zeigt die Elemente des EMOF Metametamodells.

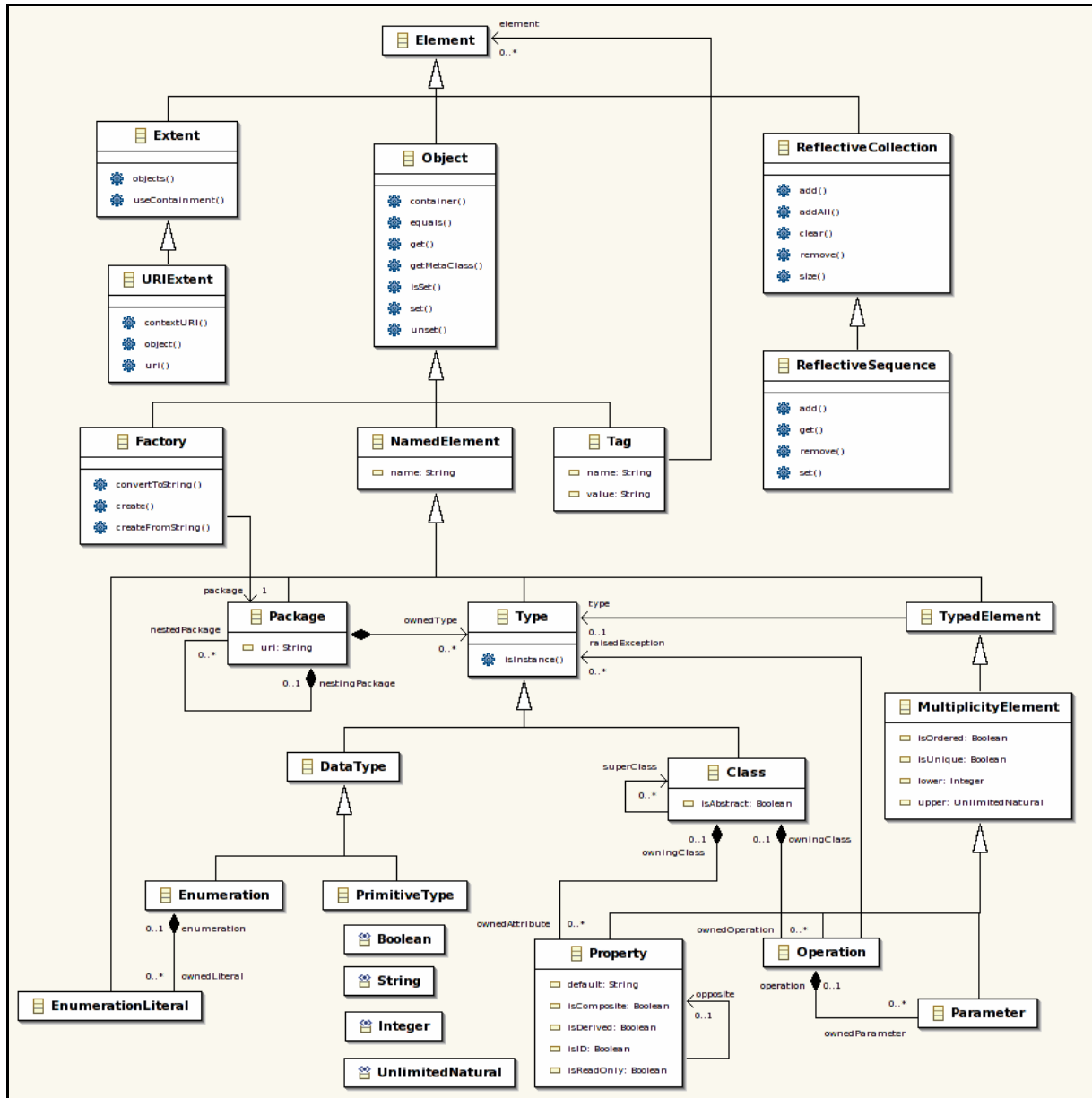


Abbildung 9 – EMOF Metametamodell (Quelle: [FLEUREY])

Wie zum Beispiel in [BBJC05] oder [BEZI05DSL] wird von Jean Bézivin für das Bridging zwischen verschiedenen Metamodellen häufig mit einem dazwischen liegenden Übergangsmetamodell gearbeitet. Dafür wurde in den genannten Dokumenten das *Kernel Metametamodell* (KM3) verwendet. Es ist stark an Ecore und EMOF angelehnt, bietet aber zusätzlich eine textuelle konkrete Syntax an. Abbildung 10 bietet einen Überblick über die beinhalteten Komponenten.

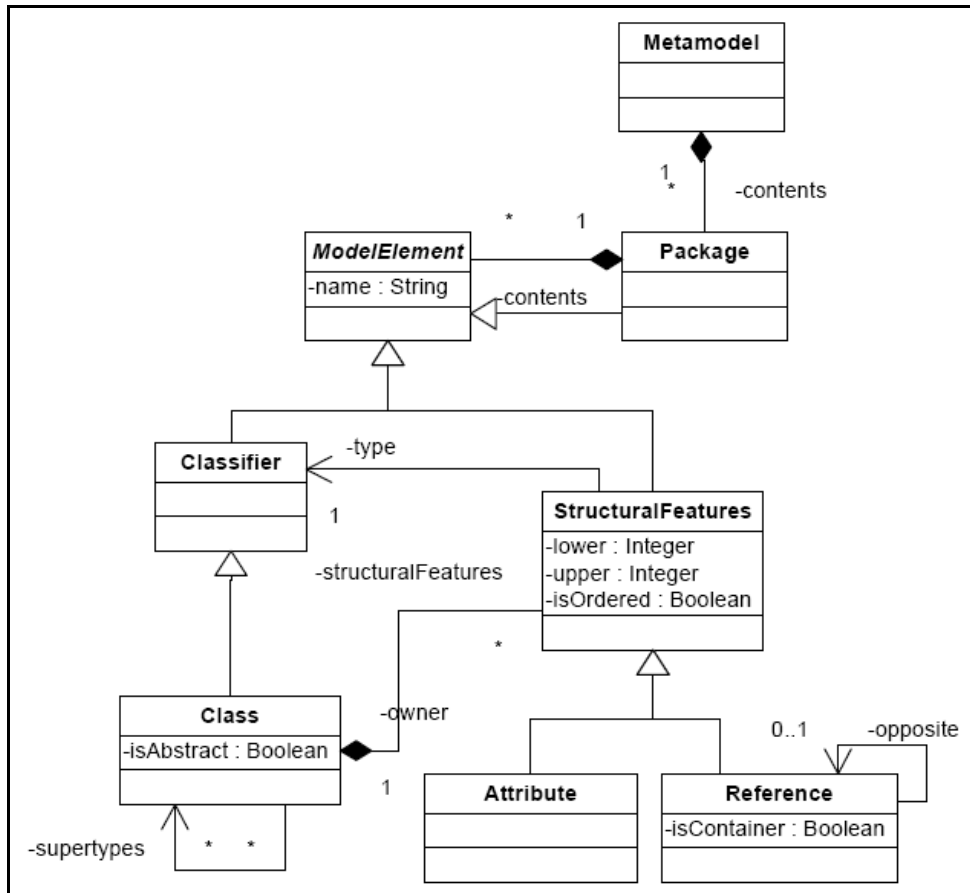


Abbildung 10 – KM3 Metametamodell (Quelle: [BEZI05DSL])

## 4 Vergleich der vorgestellten Metametamodelle

### 4.1 Komponenten

Zunächst sollen die Komponenten der vorgestellten Metametamodelle einander gegenüber gestellt werden. Dabei werden die Elemente ihren Verwendungszwecken nach gruppiert.

#### 4.1.1 Objektabbildung

Für die Erstellung von Elementen in der zu modellierenden Sprache bieten die Metasprachen mehr oder weniger ausgeprägte Konzepte an. MOF und Ecore stellen neben Klassen und Attributen zusätzlich Operation und Parameter (und Exceptions) zur Implementierung von Metamodellen bereit. Dies weist auf den objekt-orientierten Hintergrund der OMG, der Einfluss auf die Konzeption des MOF und Ecore hatten.

| MOF       | Ecore      | MetaGME   | XML Schema              |
|-----------|------------|-----------|-------------------------|
| Class     | EClass     | Model     | element, complexContent |
| Attribute | EAttribute | Attribute | attribute               |
| Operation | EOperation | Atom      |                         |
| Parameter | EParameter |           |                         |
| Exception |            |           |                         |

Tabelle 1 – MMM Komponenten für die Objektabbildung

#### 4.1.2 Relationen

| MOF                         | Ecore      | MetaGME    | XML Schema  |
|-----------------------------|------------|------------|-------------|
| Reference                   | EReference | Reference  | @id, @idref |
| Association, AssociationEnd |            | Connection | key, keyref |
|                             |            |            | extension   |

Tabelle 2 – MMM Komponenten für Relationen

#### 4.1.3 Strukturierung

| MOF              | Ecore    | MetaGME | XML Schema                   |
|------------------|----------|---------|------------------------------|
| Namespace/Import | EFactory | Project | Namespaces (import, include) |
| Package          | EPackage | Folder  | schema                       |
|                  |          | Set     | all, group, choice, sequence |
| Tag              | -        | Aspect  | attributeGroup               |

Tabelle 3 – MMM Komponenten für eine Strukturierung des Metamodells

#### 4.1.4 Datentypen

Wie bereits in Kapitel 3.3 festgestellt bietet das Metametamodell des GME Frameworks keinerlei Möglichkeiten an Datentypen oder Enumerationen zu modellieren.

| MOF              | Ecore        | MetaGME | XML Schema  |
|------------------|--------------|---------|-------------|
| Primitive Type   | EDataType    |         | simpleType  |
| Structure Type   | -            |         | complexType |
| Enumeration Type | EEnum        |         | list        |
| Collection Type  | EEnumLiteral |         |             |
| Alias Type       | -            |         |             |
| Constant         | -            |         |             |
| Structured Field | -            |         |             |

Tabelle 4 – MMM Komponenten für Datentypen

#### 4.1.5 Einschränkungen, Sichten, Aspekte

Außer Ecore bieten alle Metametamodellkonzepte die Möglichkeiten Einschränkungen bezüglich der zu modellierenden Metamodelle zu definieren.

| MOF               | Ecore | MetaGME           | XML Schema          |
|-------------------|-------|-------------------|---------------------|
| Constraints (OCL) |       | Constraints (OCL) | field, selector     |
|                   |       | Aspect            | restriction, unique |
|                   |       | Part              | union               |
|                   |       | Role, ConnRole    |                     |

Tabelle 5 – MMM Komponenten für Einschränkungen, Sichten und Aspekte

#### 4.1.6 Spezielle Komponenten

Ecore und XML Schema bieten zusätzliche Elemente, die den Prozess der Metamodellierung unterstützen.

| Ecore       | XML Schema    |
|-------------|---------------|
| EAnnotation | notation      |
|             | redefine      |
|             | documentation |
|             | appInfo       |
|             | any           |
|             | anyAttribute  |
|             | simpleContent |

Tabelle 6 – Spezielle MMM Komponenten

### 4.2 Vergleich bezüglich Mächtigkeit und Praxisbezug

Leider konnten bei der Recherche keine einheitlichen Kriterien für einen Vergleich bezüglich der Mächtigkeit gefunden werden. Daher wird dieser auf Grund eigener Einschätzungen und den vorangegangenen Untersuchungen erfolgen.

|                 | MOF   | Ecore  | MetaGME  | XML Schema   |
|-----------------|---|--|--|--|
| Relationen      | ●●  | ●○   | ●●   | ●●   |
| Einschränkungen | ●○  | ○○   | ●○   | ●●   |
| Pro/Kontra      | + sehr mächtig<br>+ sprachunabhängig (IDL, XMI)<br>+ viele bekannte Metasprachen (UML, CWM)<br>– sehr komplex | + einfacher Aufbau<br>– komplexe Metamodelle erfordern mehr Aufwand<br>– fehlende Constraints<br>– stark Java orientiert | + Aspekte bereits im MMM integriert<br>– MMM nur bei GME verwendet | + sehr mächtig<br>+ sprachunabhängig (XML)<br>+ großes Spektrum existierender Metasprachen |
| Komplexität     | ●●  | ●○   | ●○   | ●●   |
| Mächtigkeit     | ●●  | ●○   | ●●   | ●●   |
| Werkzeuge       | UML Editoren  | EMF  | GME  | XML/XSD Editoren   |

Tabelle 7 – Vergleich der MMM bezüglich Mächtigkeit und Werkzeugunterstützung

## 5 Fazit

Die vorgestellten Metametamodelle bieten alle eine ausreichende Grundlage um domänenspezifische Sprachen zu entwickeln. Während mit EMF und GME komplette Frameworks für die Metamodellimplementierung und -verwendung zur Verfügung stehen, besitzen XML Schema und MOF ein größeres Potential bezüglich ihrer Modellierungsfähigkeiten.

Aktuelle Aufgaben zum Thema Metamodellierung befassen sich mit den Möglichkeiten der Transformation, um Metametamodelle übergreifend erstellte Modelle in andere Domänen zu überführen. Ein weiterer wichtiger und weiter zu verfolgender Aspekt dieser Thematik ist das Konzept der Software Factories, mit deren Hilfe es möglich sein wird auf einfache Art und Weise Software Systeme aus verschiedenen bereits bestehenden Komponenten zu kreieren. Metametamodelle bilden hierfür eine entscheidende Grundlage für die Umsetzung.

## Quellenangaben

- [BBJC05]..... Jean Bézivin, Christian Brunette, Régis Chevrel; „**Bridging the Generic Modeling Environment (GME) and the Eclipse Modeling Framework (EMF)**“; Paper von [http://www.sciences.univ-nantes.fr/lina/atl/publications/pubpres2005/linkpage\\_view2/](http://www.sciences.univ-nantes.fr/lina/atl/publications/pubpres2005/linkpage_view2/); 2005
- [BEZI05] ..... Jean Bézivin; „**On the Unification Power of Models**“; Paper von [http://www.sciences.univ-nantes.fr/lina/atl/publications/pubpres2005/linkpage\\_view2/](http://www.sciences.univ-nantes.fr/lina/atl/publications/pubpres2005/linkpage_view2/); 2005
- [BEZI05DSL].... Jean Bézivin, Guillaume Hillairet, Frédéric Jouault, Ivan Kurtev, William Piers; „**Bridging the MS/DSL Tools and the Eclipse Modeling Framework**“ ; Paper von [http://www.sciences.univ-nantes.fr/lina/atl/publications/pubpres2005/linkpage\\_view2/](http://www.sciences.univ-nantes.fr/lina/atl/publications/pubpres2005/linkpage_view2/); 2005
- [DjGaDe05]..... Dragan Djurić, Dragan Gašević and Vladan Devedžić; „**Adventures in Modeling Spaces: Close Encounters of the Semantic Web and MDA Kinds**“; Paper von [http://www.mel.nist.gov/msid/conferences/SWESE/accepted\\_papers.html](http://www.mel.nist.gov/msid/conferences/SWESE/accepted_papers.html); 2005
- [EMFOV] ..... „**The Eclipse Modeling Framework (EMF) Overview**“; Webseite: <http://www.eclipse.org/emf/docs.php?doc=references/overview/EMF.html>
- [FLEUREY] ..... „**EMOF model and diagram**“;  
Webseite: <http://www.fleurey.com/weblog/pivot/entry.php?id=28>
- [GeRa03] ..... Anna Gerber, Kerry Raymond; „**MOF to EMF: There and Back Again**“; Paper von <http://www.dstc.edu.au/Research/Projects/Pegamento/publications/>; 2003
- [JECKLE00]..... Mario Jeckle; „**Konzepte der Metamodellierung – Zum Begriff Metamodell**“; Paper von <http://www.jeckle.de/publikat.htm>; Publiziert in „Softwaretechnik Trends“, Bd. 20, Heft 2; 2000
- [KUEHNE05].... Thomas Kühne; „**What is a Model?**“; Paper von <http://drops.dagstuhl.de/opus/volltexte/2005/23/>; 2005
- [LeMaBa01] ..... Akos Ledeczi, Miklos Maroti, Arpad Bakay, Gabor Karsai; „**The Generic Modeling Environment – High Level Whitepaper**“; Paper von <http://www.isis.vanderbilt.edu/>; 2001
- [LeMaVö03]..... Ákos Lédeczi, Miklós Maróti and Péter Völgyesi; „**The Generic Modeling Environment – Technical Report**“; Paper von <http://www.isis.vanderbilt.edu/>; 2003
- [OMGMOF02] .. OMG; „**Meta Object Facility Specification**“, Version 1.4; Spezifikation von <http://www.omg.org/technology/documents/formal/mof.htm>; 2002
- [W3C]..... W3C; „**XML Schema Part 1: Structures Second Edition**“; Spezifikation von <http://www.w3.org/TR/xmlschema-1/> und <http://w3c.org/TR/2005/WD-xmlschema11-1-20050224/>; 2005