BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
LEIPZIG
UNIVERSITY OF COOPERATIVE EDUCATION

Software Engineering (SE)

Prof. Dr.-Ing. Christian Heller
<christian.heller@ba-leipzig.de>

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

Diagram Definition (DD)

Related Specification Languages

Software Pattern

Pattern Catalogue

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

Diagram Definition (DD)

Related Specification Languages

Software Pattern

Pattern Catalogue

Learning Objectives

... to enable software engineering students to:

- analyse application domain
- design system using graphical notation
- document architecture using UML diagrammes
- work with CASE tool

Target Group

- students of computer science

Pre-Requisite

- advanced knowledge in computer programming
- familiarity with Object Oriented Programming

Software

- UML modelling tool

Curriculum

- 1st term: 28 hours
- 1st term: 50 hours
- 2nd term: 54 hours
- 3rd term: 48 hours
- 4th term: 42 hours
- 4th term: 35 hours
- 5th term: 4+60 hours
- 5th term: 36 hours
- 5th term: 55 hours
- 6th term: 56 hours

- Web Technologies
- Computer Programming
- Data Processing
- User Interaction
- Software Engineering
- Project Management
- Systems Implementation
- Programming C/C++
- CYBOP
- Server-side Technologies

Schedule


Revision: on blackboard (30 min)


Lecture: new matter (60 min)


Break (30 min)


Presentation: done by students (2 x 40 min)


Break (30 min)


Exercise: solving example tasks (90 min)


Self-Study (unlimited, > 50 % in a study ;-)

Literature - Requirements Analysis

- Suzanne Robertson: Mastering the Requirements Process. Addison-Wesley Professional, 2006
- Colin Hood: Requirements Management: Interface Between Requirements Development and All Other Engineering Processes. Springer, 2007
- Klaus Pohl: Requirements Engineering: Grundlagen, Prinzipien, Techniken. dpunkt.Verlag, 2008
- Chris Rupp & die SOPHISTen: Requirements Engineering und Management. Professionelle, iterative Anforderungsanalyse für die Praxis. Hanser, 2009
- Helmuth Partsch: Requirements-Engineering systematisch. Springer, 2010
- Christof Ebert: Systematisches Requirements Engineering und Management. dpunkt.Verlag, 2012

Literature - Analysis and Design

- Peter Coad: Objektorientierte Analyse. Prentice Hall, 1994
- Peter Coad: Objektorientiertes Design. Prentice Hall, 1994
- Ivar Jacobson: The unified software development process. UML. Addison-Wesley, 1999
- Heide Balzert: Lehrbuch der Objektmodellierung. Analyse und Entwurf. Spektrum, 1999
- Grady Booch: Object-Oriented Analysis and Design with Applications. Third Edition, Addison-Wesley, 2007

Literature - UML

- Bernd Oestereich: Objektorientierte Softwareentwicklung - Analyse und Design mit der UML. Oldenbourg Verlag, 1998
- Martin Fowler: UML konzentriert. Addison-Wesley, 2003
- Marc Born: Softwareentwicklung mit UML 2. ... UML 2, MOF 2, und MDA. Addison-Wesley, 2004
- Heide Balzert: UML 2 kompakt. Spektrum, 2005
- Dan Pilone: UML 2.0 in a Nutshell. O'Reilly, 2006
- Christoph Kecher: UML 2.0. Galileo Press, 2006
- Chris Rupp: UML 2 glasklar: Praxiswissen für die UML-Modellierung. Carl Hanser Verlag, 2012
- OOSE: UML Notation Overview. http://www.oose.de/
- Scott W. Ambler: Introduction to the Diagrams of UML 2.X. http://www.agilemodeling.com/essays/umlDiagrams.htm
- Kirill Fakhroutdinov: UML Diagrams Overview. http://www.uml-diagrams.org/

Literature - Pattern and Refactoring

- Joshua Kerievsky: Refactoring To Patterns. Addison-Wesley, 2004
- Martin Fowler: Refactoring. Wie Sie das Design vorhandener Software verbessern. Addison-Wesley, 2005
- Helmut Leitner: Mustertheorie - Einführung und Perspektiven auf den Spuren von Christopher Alexander. Nausner & Nausner, 2007
- Florian Siebler: Design Patterns mit Java: Eine Einführung in Entwurfsmuster. Hanser, 2014

Literature - Architectural Pattern

- Frank Buschmann: Pattern-orientierte Software-Architektur. Ein Pattern-System. Volume 1. Addison-Wesley-Longman, 1998
- Frank Buschmann: Pattern-orientierte Software-Architektur. Muster für nebenläufige und vernetzte Objekte. Volume 2. dpunkt.verlag, 2002
- Michael Kircher: Pattern-Oriented Software Architecture. Patterns for Resource Management. Volume 3. Wiley, 2004
- Frank Buschmann: Pattern-Oriented Software Architecture. A Pattern Language for Distributed Computing. Volume 4. Wiley, 2007
- Martin Fowler: Patterns für Enterprise Application-Architekturen. Mitp-Verlag, 2003
- Gregor Hohpe: Enterprise Integration Patterns: Designing, Building, and Deploying ... Addison-Wesley, 2003

Literature - Design Pattern

- Wolfgang Pree: Design Patterns for Object-Oriented Software Development. Addison Wesley, ACM Press, 1994
- Erich Gamma: Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software. Addison Wesley, 2004
- Helmut Herold: C++, UML und Design Patterns. Addison-Wesley, 2005
- Eric Freeman: Entwurfsmuster von Kopf bis Fuß. O'Reilly, 2006

Literature - Idiom (Pattern)

- Deepak Alur: Core J2EE Patterns: Best Practices and Design Strategies. Prentice Hall, 2003
- Adam Bien: Java EE 5 Architekturen. Java Patterns und Idiome. Entwickler.Press, 2007

Examination

- presentation on UML diagramme during semester 30 min
- examination at end of semester 180 min
  - 120 min SE computer
  - 60 min PM written
- create and transform UML diagrammes
- design business domain model
- document static and dynamic aspects
- apply CASE tool to generate software

Assessment

- 20 % presentation SE
- 80 % examination (SE : PM == 2 : 1)
  - 100 points SE 120 min
  - 50 points PM 60 min

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

Diagram Definition (DD)

Related Specification Languages

Software Pattern

Pattern Catalogue

Disciplines and Categorisation

Kernprozesse

1. Planung

- Anforderungserhebung
- Lastenheft (Anforderungsdefinition)
- Pflichtenheft (Mit technischen Ansätzen verfeinertes Lastenheft)
- Aufwandsschätzung (z. B. mittels Function-Point-Verfahren oder COCOMO)
- Vorgehensmodell

2. Analyse

- Auswertung
- Mock-up
- Prozessanalyse / Prozessmodell
- Systemanalyse
- Strukturierte Analyse (SA)
- Objektorientierte Analyse (OOA)

3. Entwurf

- Softwarearchitektur
- Strukturiertes Design (SD)
- Objektorientiertes Design (OOD)
- Fundamental Modeling Concepts (FMC)

4. Programmierung

- Normierte Programmierung
- Strukturierte Programmierung
- Objektorientierte Programmierung (OOP)
- Funktionale Programmierung

5. Validierung und Verifikation

- Modultests (Low-Level-Test)
- Integrationstests (Low-Level-Test)
- Systemtests (High-Level-Test)
- Akzeptanztests (High-Level-Test)

Unterstützungsprozesse

6. Anforderungsmanagement

7. Projektmanagement

- Risikomanagement
- Projektplanung
- Projektverfolgung und -steuerung
- Management von Lieferantenvereinbarungen

8. Qualitätsmanagement

- Capability Maturity Model
- Spice (Norm) (Software Process Improvement and Capability Determination)
- Incident Management
- Problem Management
- Softwaremetrik (Messung von Softwareeigenschaften)
- statische Analyse (Berechnung von Schwachstellen)
- Softwareergonomie

9. Konfigurationsmanagement

- Versionsverwaltung
- Änderungsmanagement / Veränderungsmanagement
- Releasemanagement
- Application-Management (ITIL)

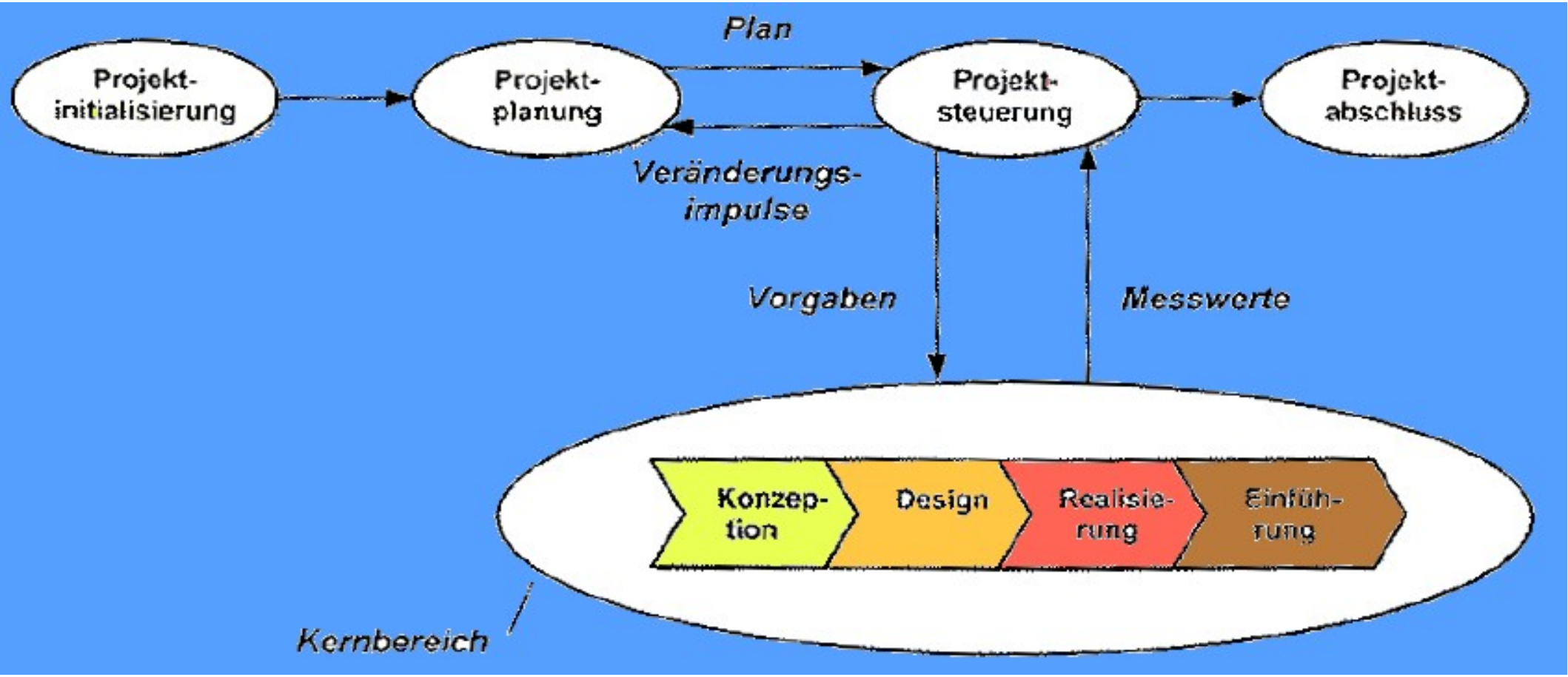
10. Softwareeinführung

11. Dokumentation

- Technische Dokumentation
- Softwaredokumentation
- Systemdokumentation (Weiterentwicklung und Fehlerbehebung)
- Betriebsdokumentation (Betreiber/Service)
- Bedienungsanleitung (Anwender)
- Geschäftsprozesse (Konzeption der Weiterentwicklung)
- Verfahrensdokumentation (Beschreibung rechtlich relevanter Softwareprozesse)

https://de.wikipedia.org/wiki/Softwaretechnik

Placement



Hans Brandt-Pook, Rainer Kollmeier. Softwareentwicklung kompakt und verständlich. Wie Softwaresysteme entstehen. Wiesbaden: Vieweg+Teubner, 2008

Requirements Engineering (Criterions)

Ermittlung und Analyse:

- vollständig: explizit, keine impliziten Annahmen
- eindeutig: abgegrenzt, präzise Definitionen
- verständlich beschrieben
- atomar: eine Anforderung pro Abschnitt oder Satz
- identifizierbar: z. B. über Kennung oder Nummer
- einheitlich: Anforderungen und ihre Quellen zusammen dokumentiert
- nachprüfbar: verknüpfen mit Abnahmekriterien, ableiten Testfälle
- verfolgbar: vorwärts (ob vollständig erfüllt), rückwärts (zu welcher Funktionalität welche Anforderung gehört)
- konsistent

Ergebnisliste --> Lastenheft

Strukturierung und Abstimmung:

- abhängig: ist Anforderung Voraussetzung für andere, gegenseitiges Bedingen
- zusammengehörig: gemeinsames Realisieren von fachlich-logisch zusammengehörenden Anforderungen
- rollenbezogen

Iterationen und Verfeinerung --> Unterscheidung funktionale und nichtfunktionale Anforderungen

Prüfung und Bewertung:

- korrekt: untereinander widerspruchsfrei
- machbar: realisierbar
- notwendig: nur was gefordert
- priorisiert: wichtigste markieren, schneller bereitstellen
- nutzbar: nützlich

Betrachtung Gesamtsystem --> Basis für Pflichtenheft

Requirement

functional

- defines function of system
- set of i/o and behaviour
- e.g. calculations, technical details, data manipulation
- WHAT to accomplish
- application architecture

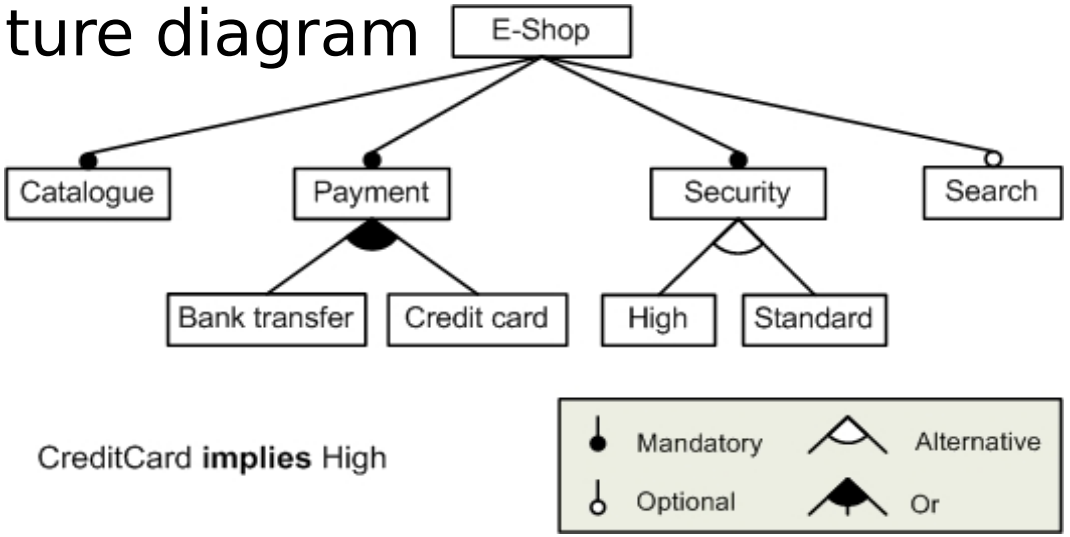
non-functional (quality)

- specify overall characteristics
- constraints on design
- e.g. reliability, performance, security, reliability, cost
- HOW to work
- technical architecture

https://de.wikipedia.org/wiki/Anforderung_%28Informatik%29

Feature Model (Structured Requirements)

- used in context of Software Product Line (SPL)
- compact representation of all products of an SPL
- input for analysis documents, architecture, pieces of code
- supports functional and architectural reuse
- visually represented by feature diagram
- result of domain analysis



https://en.wikipedia.org/wiki/Feature_model

Object Oriented Analysis and Design (OOAD)

- Notation: e.g. UML

- Programming Language: e.g. OOP in Java

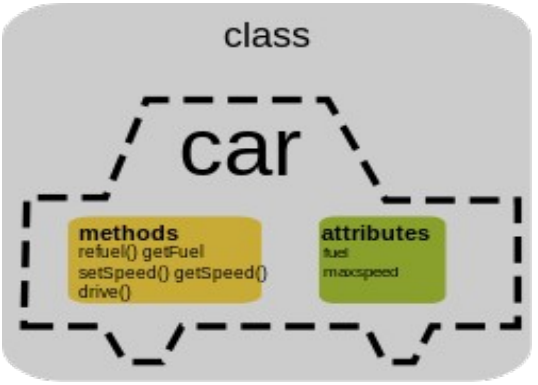
- Methodology: e.g. RUP, Scrum

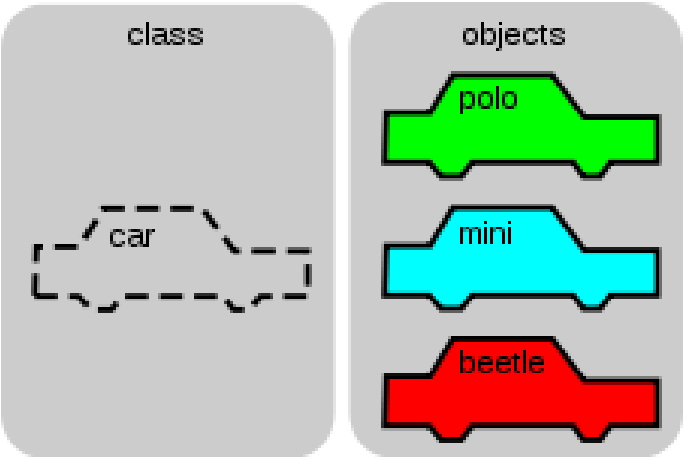
- Tool: e.g. BOUML, Modelio

= Assistive Means [Hilfsmittel]

https://de.wikipedia.org/wiki/Objektorientierte_Analyse_und_Design

OOAD Example "Car"





Summary

- Software engineering: technique, method, tool
- Focus: realisation phase of project management
- Requirements Engineering: analyse, structure, validate
- Requirement: functional and non-functional
- Feature Model
- OOAD

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

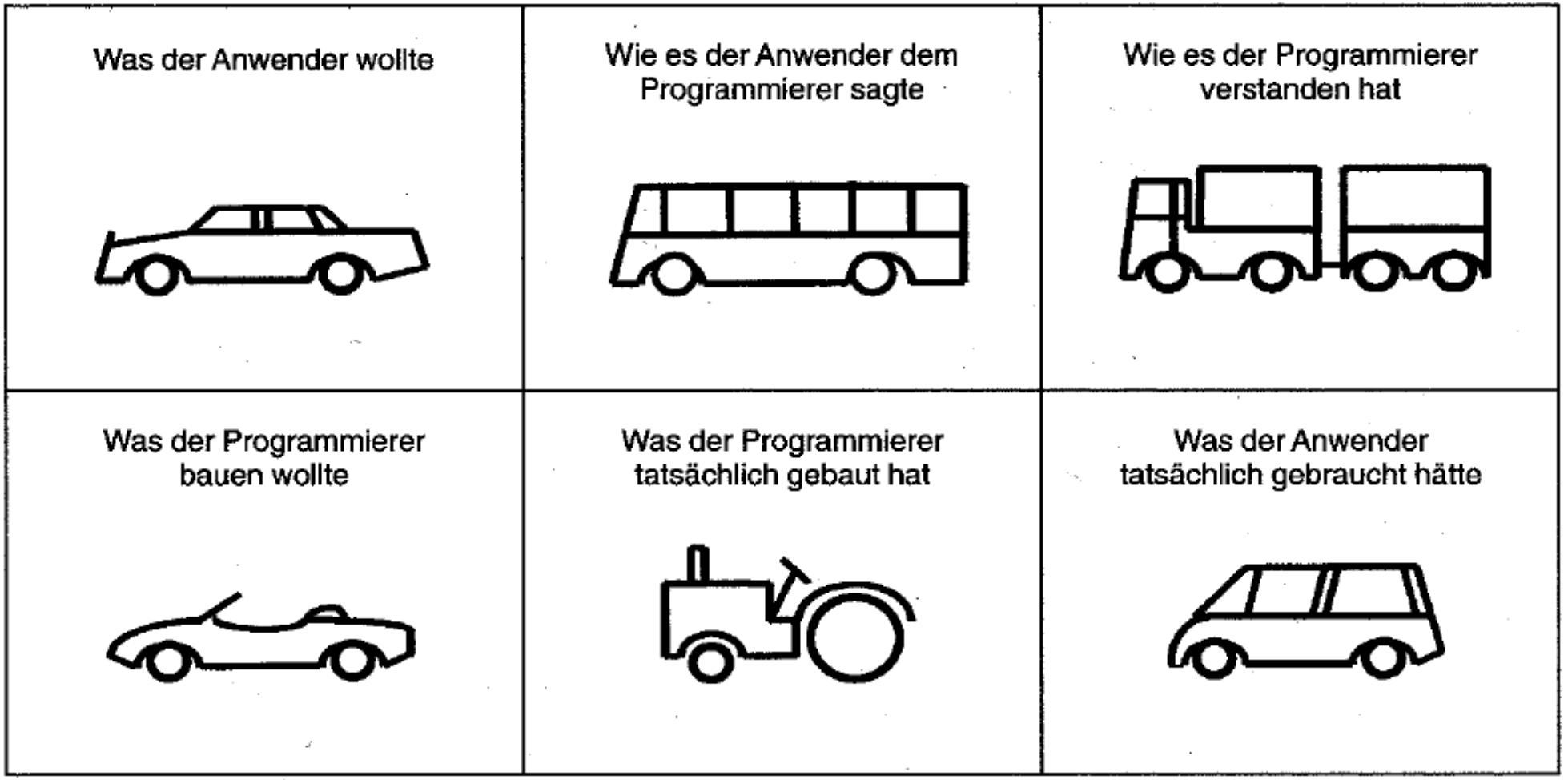
Diagram Definition (DD)

Related Specification Languages

Software Pattern

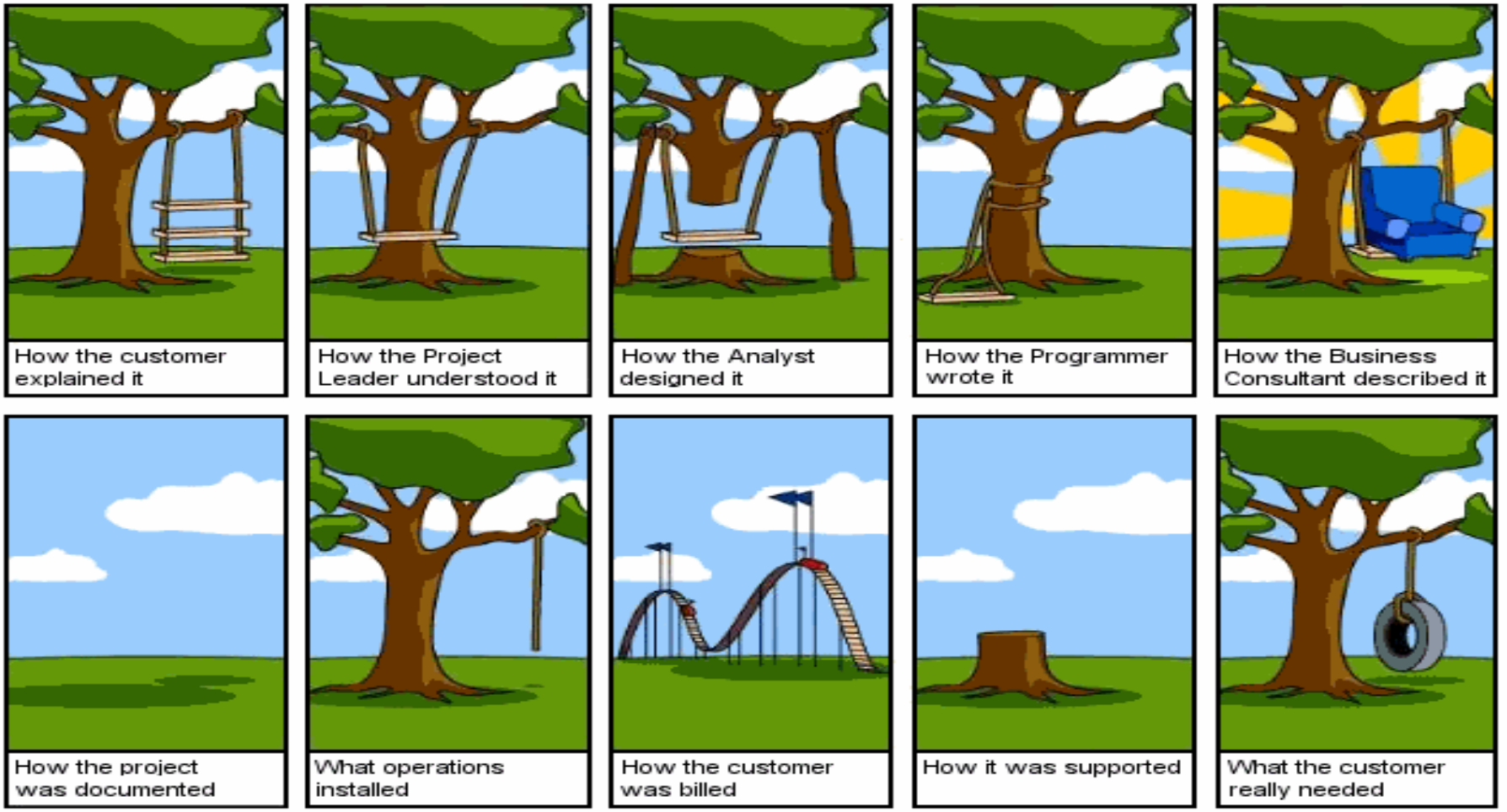
Pattern Catalogue

Motivation 1



Johannes Siedersleben. Softwaretechnik: Praxiswissen für Softwareingenieure. Hanser, 2002

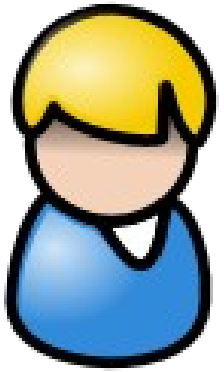
Motivation 2



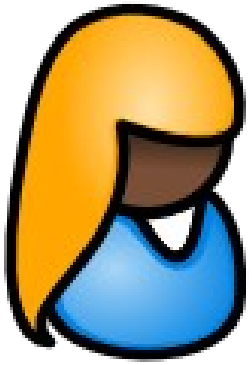
Alex Gorbatchev. Software Engineering Explained

Graphical Notation

client

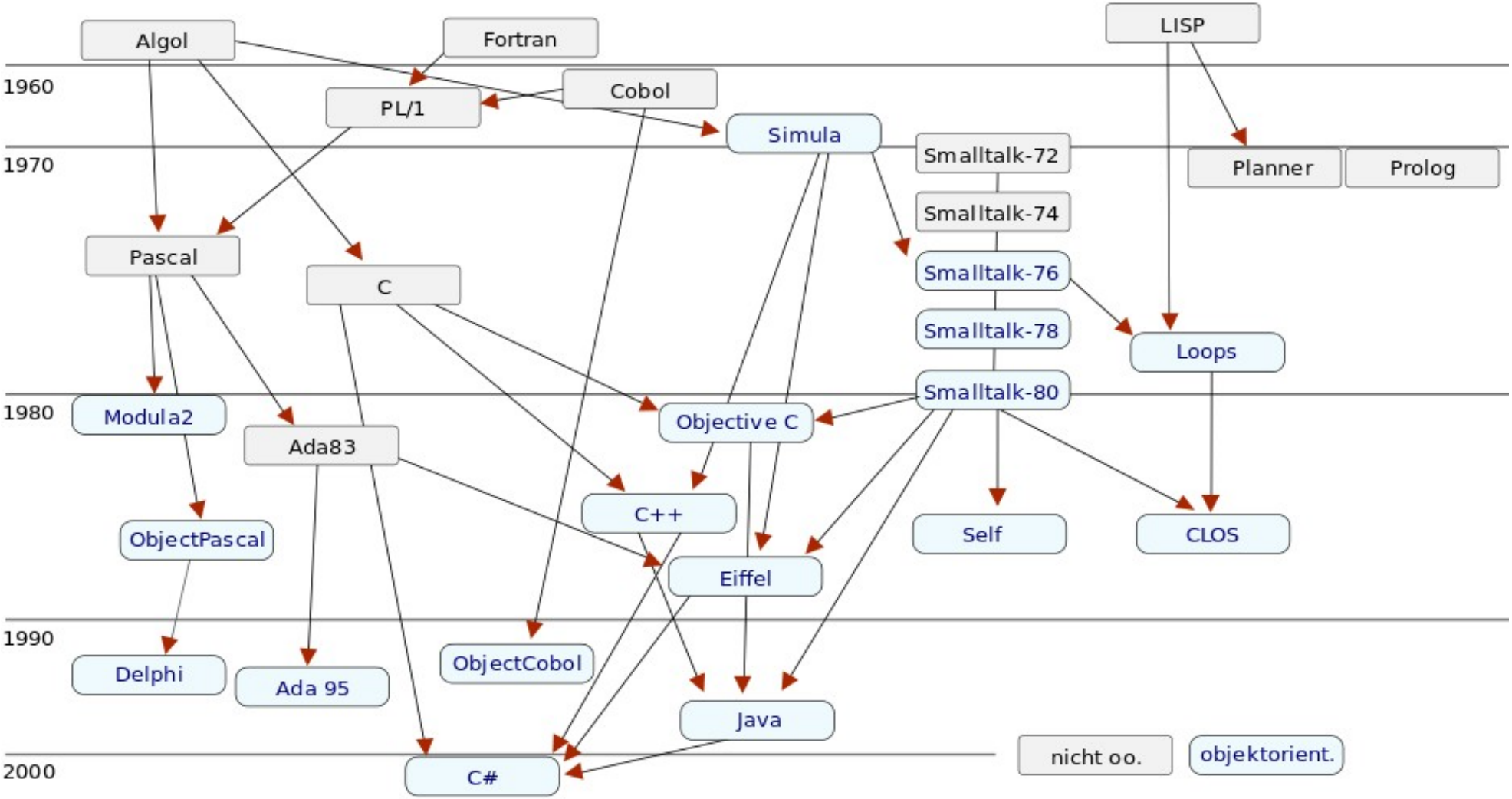


developer



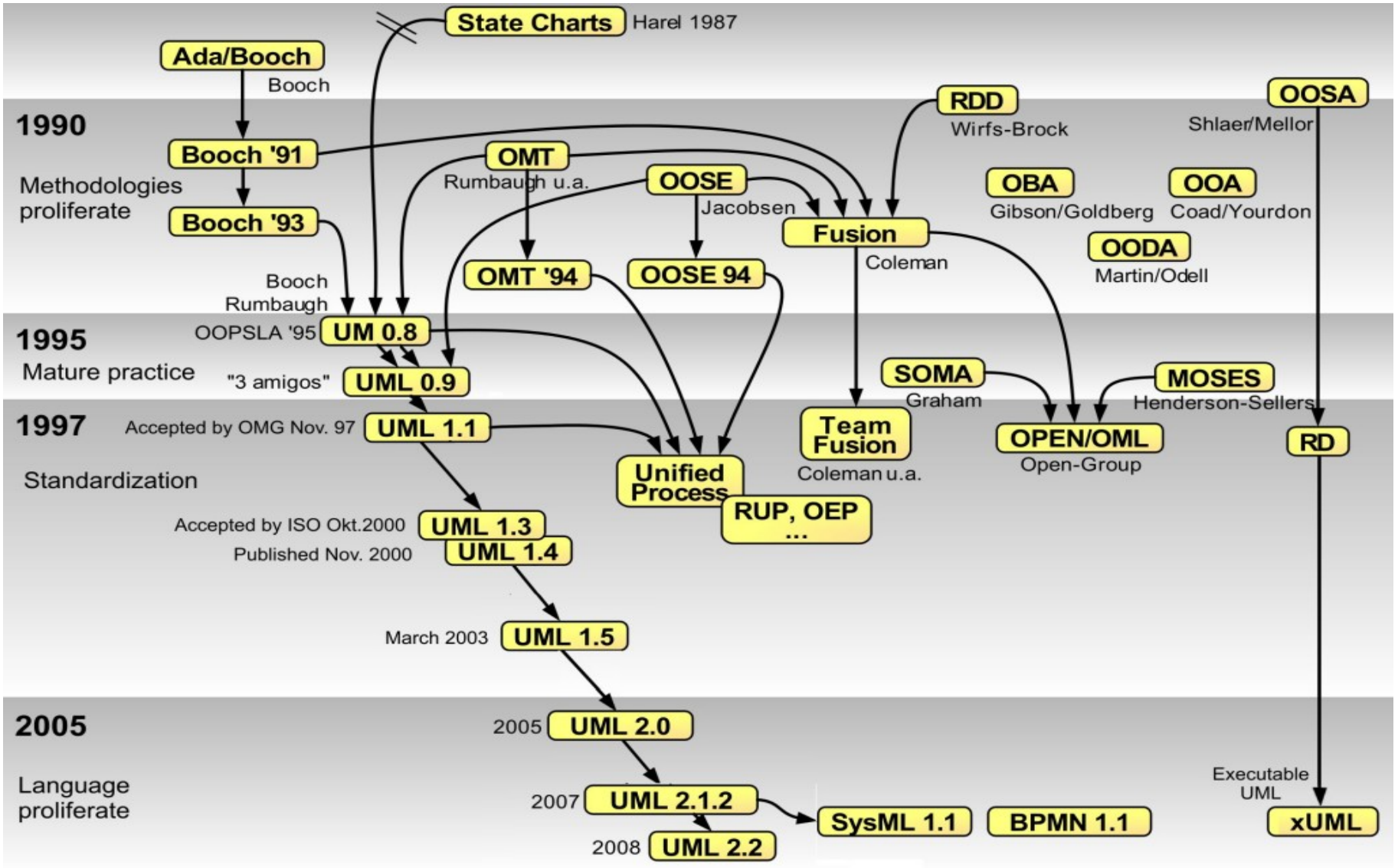
user

OOP Language History



https://de.wikipedia.org/wiki/Geschichte_der_Programmiersprachen

Notation History



https://en.wikipedia.org/wiki/Unified_Modeling_Language

UML and Related Specifications

- UML Superstructure + Infrastructure (merged since 2.5)
- Object Constraint Language (OCL)
- Diagram Definition (DD) replaces Diagram Interchange (DI) 1.0 from 2006-04-04

UML 2.5 specification is a single document (earlier versions separated infrastructure and superstructure)

Advantages and Deficiencies

- Standard
- Design
- Flexibility
- Abstraction
- Views
- Collaboration
- Process

- Oversize
- Complexity
- Synchronisation
- Inconsistency
- Overhead
- Interchange

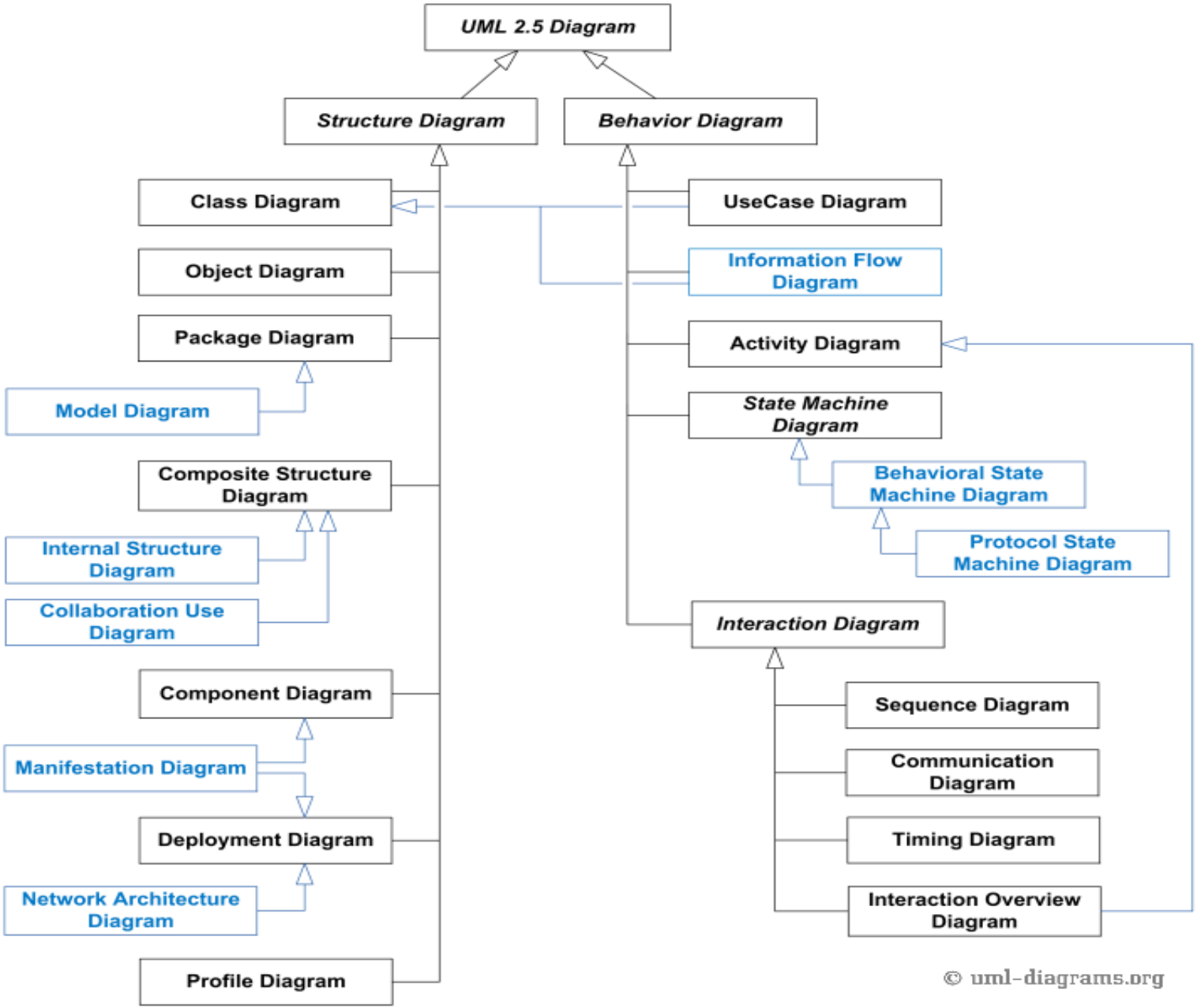

UML 2.5 Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Object Diagram
Package Diagram
Model Diagram
Composite Structure Diagram
Internal Structure Diagram
Collaboration Use Diagram
Component Diagram
Manifestation Diagram
Deployment Diagram
Network Architecture Diagram
Profile Diagram
UseCase Diagram
Information Flow Diagram
Activity Diagram
State Machine Diagram
Behavioral State Machine Diagram
Protocol State Machine Diagram
Interaction Diagram
Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram
© uml-diagrams.org

Diagram	Description	Priority
CD	static elements (classes, relationships)	high
OD	instance structure at a specific time	low
CMD	components and dependencies	medium
PD	system split-up into logical groupings	low
DD	system hardware, deployed components	medium
CSD	internal class structure, collaborations	low
PRD	at metamodel level, stereotype, profile	low

Structure Diagrams

Diagram	Description	Priority
UCD	functionality (actors, use cases, relation)	medium
AD	step-by-step workflows of components	high
SMD	dynamic behaviour of an entity	medium

Behaviour Diagrams

Diagram	Description	Priority
SD	objects and exchanged messages order	high
COD	interactions between objects	low
IOD	control flow within system, variant of AD	low
TD	object behaviour over period of time	low

Interaction Diagrams

Summary

UCD

AD

SD

SMD

Project Progress

CD

CMD

PD

DD

Subjective recommendation for diagramme usage (forget about the other diagrams)

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

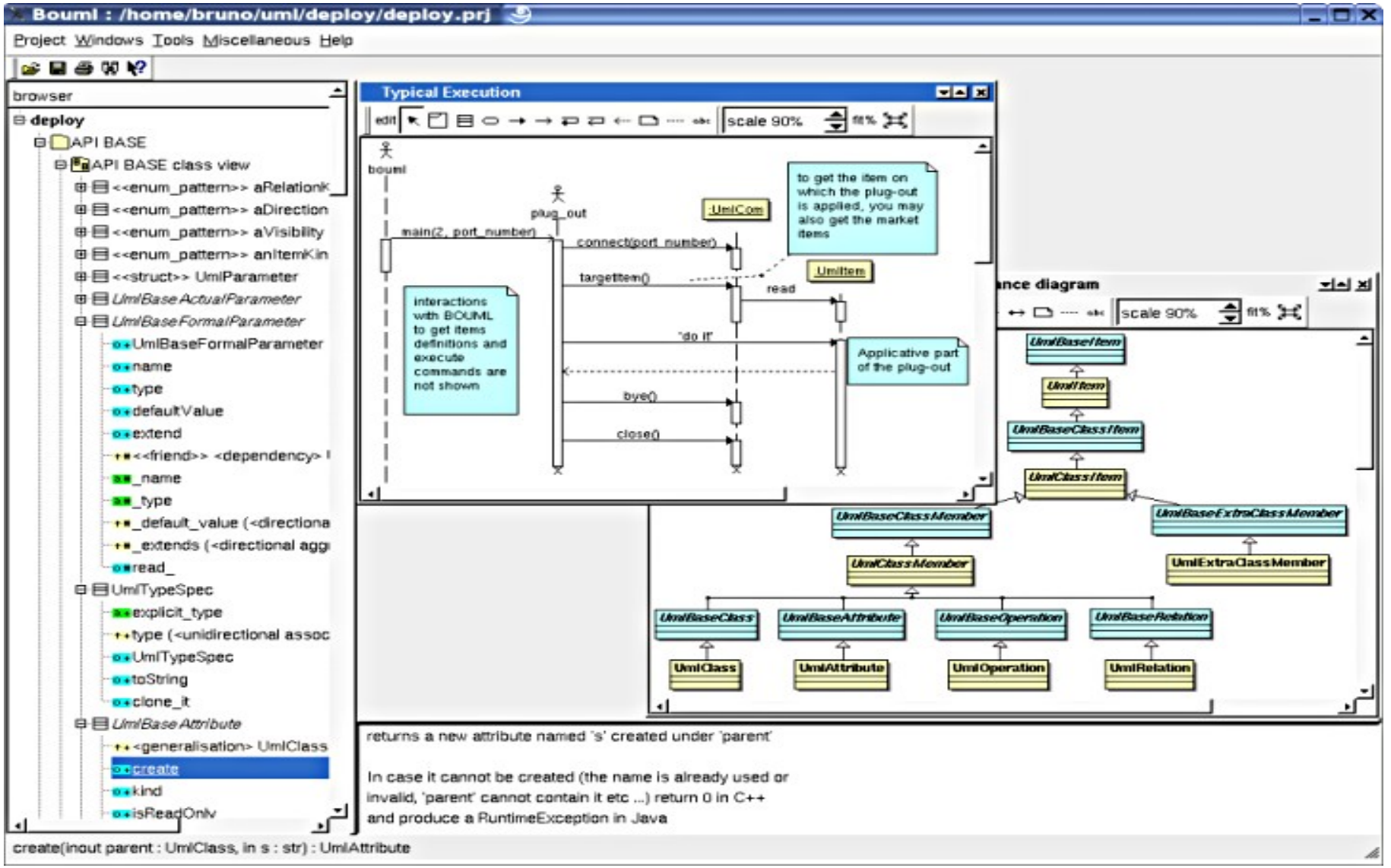
Diagram Definition (DD)

Related Specification Languages

Software Pattern

Pattern Catalogue

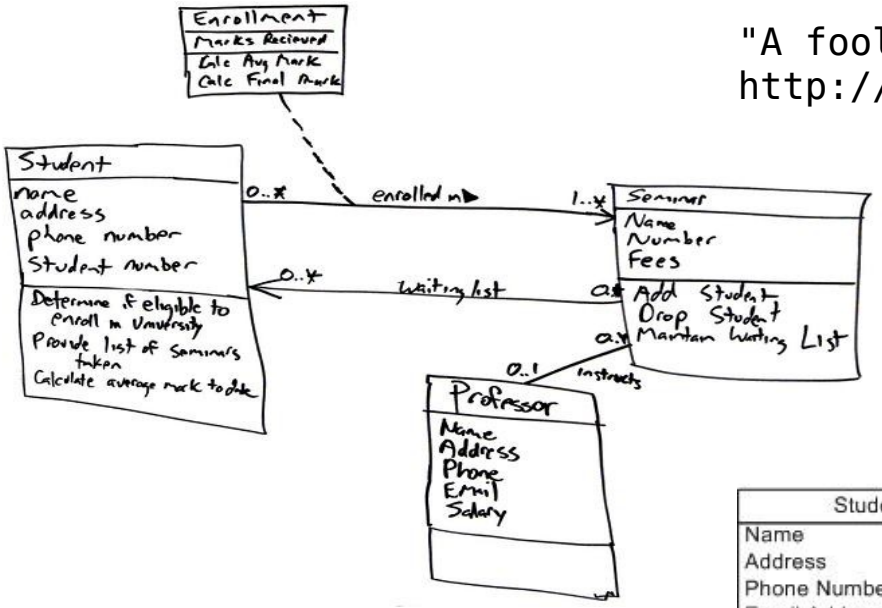
Tool Example

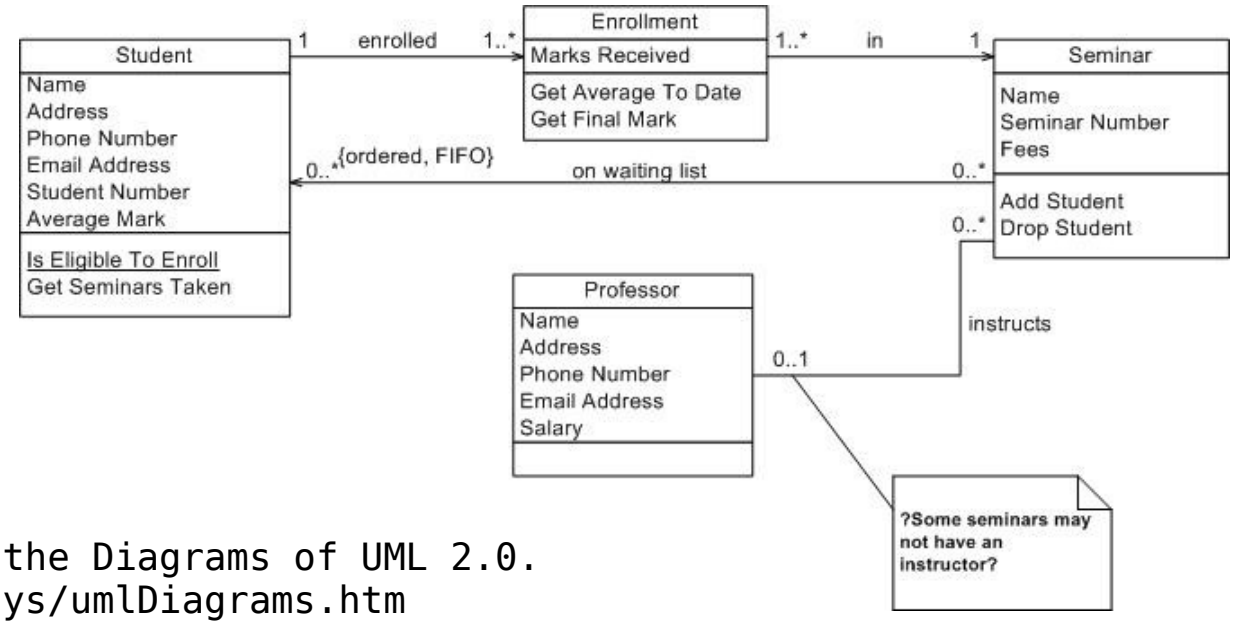


Manual versus CASE Tool-assisted Modelling

"Der Narr bleibt ein Narr, gäbst du ihm auch eine Pfarr"
http://www.wikiproverbs.com/

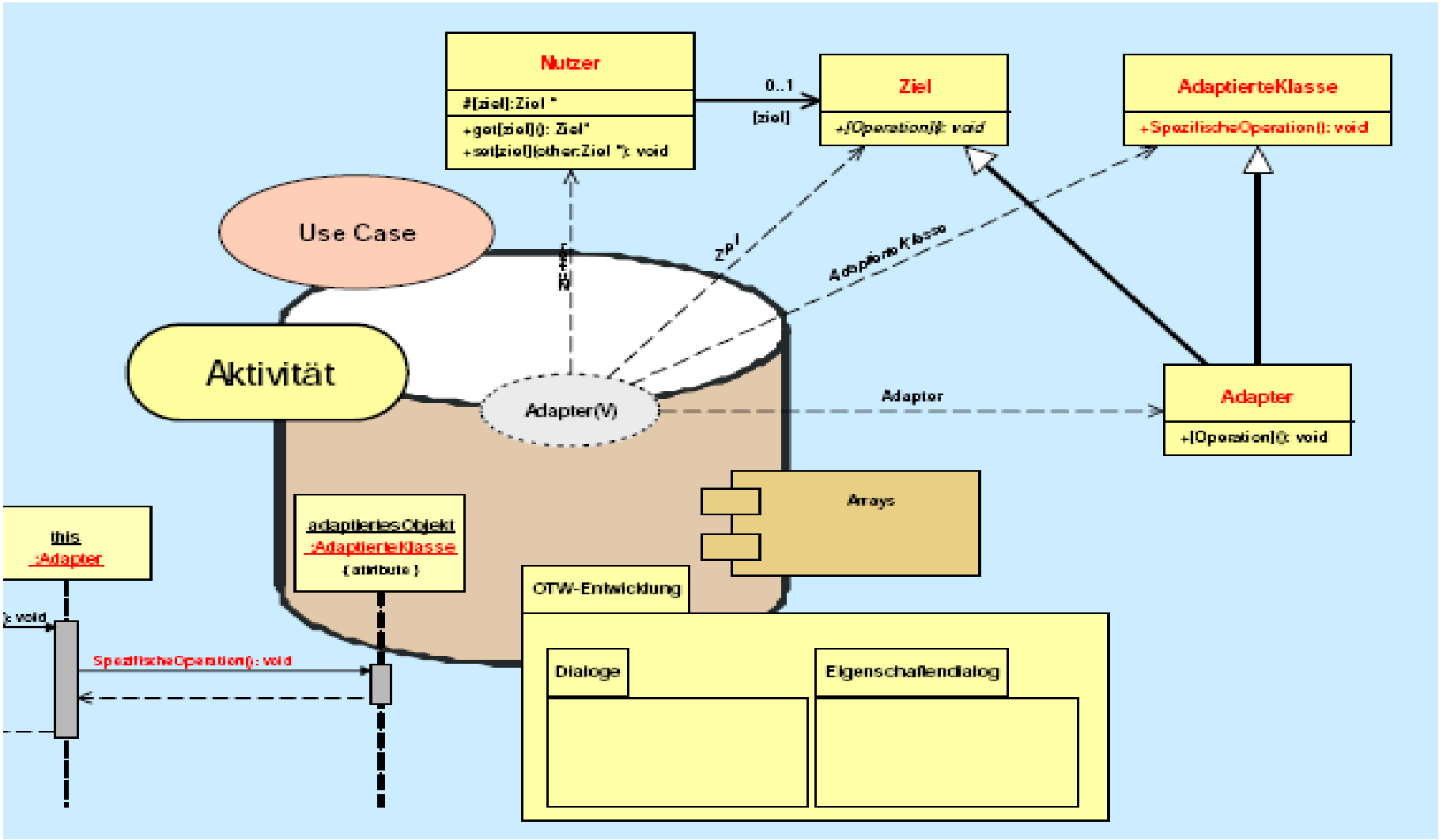
"A fool with a tool is still a fool" (Grady Booch)
http://dachkm.org/wiki/





Scott W. Ambler. Introduction to the Diagrams of UML 2.0.
http://www.agilemodeling.com/essays/umlDiagrams.htm

Model Repository and Diagramme Views



Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML. Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

Element Structuring

- Packages
- Diagrams
- Functional Views
- ...

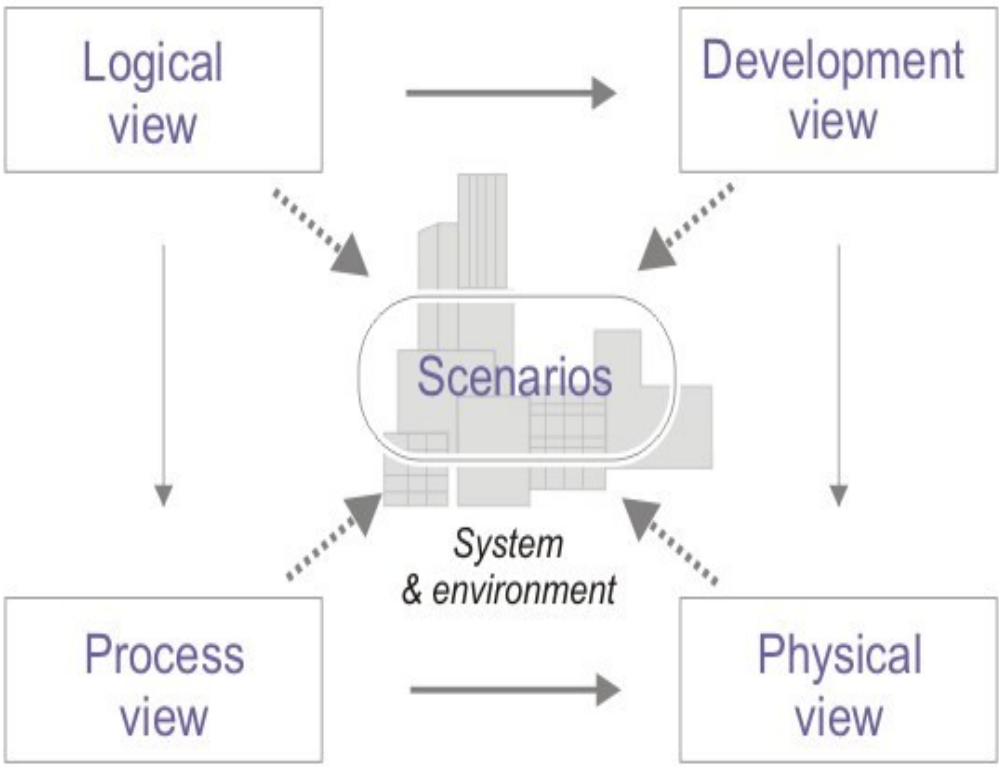
- Pakete
  - Pakete
  - Use Cases
  - Klassen
    - Basisklassen
    - Attribute
      - Datentyp
    - Methoden
      - Parameter, Rückgabewerte
  - Globale Objekte
    - Datentyp
  - Module
  - Knoten
  - Diagramme
- Diagramme
  - Use-Case-Diagramme
    - Use Cases
    - Akteure
  - Aktivitätsdiagramme
    - Aktivitäten
  - Sequenzdiagramme
    - Objekte
    - Botschaften
  - Kollaborationsdiagramme
    - ...
  - Klassendiagramme
    - ...
  - Komponentendiagramme
    - ...
  - ...

Some UML tools may use alternative structures ...

Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

Diagramme Views of a System Model

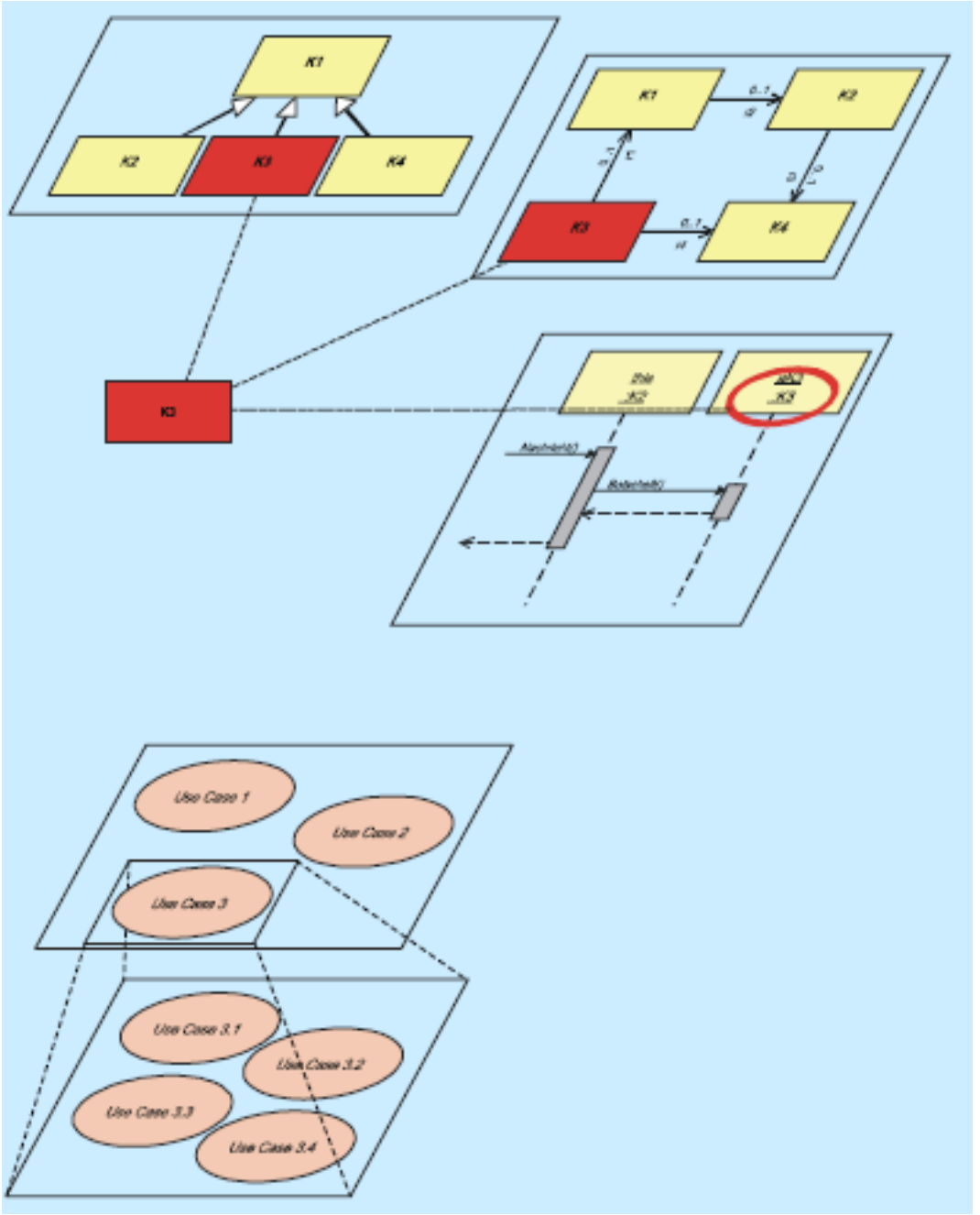
- Use case view
- Logical view
- Component view
- Process view
- Deployment view
- Entity relationship model



... in BOUML CASE tool

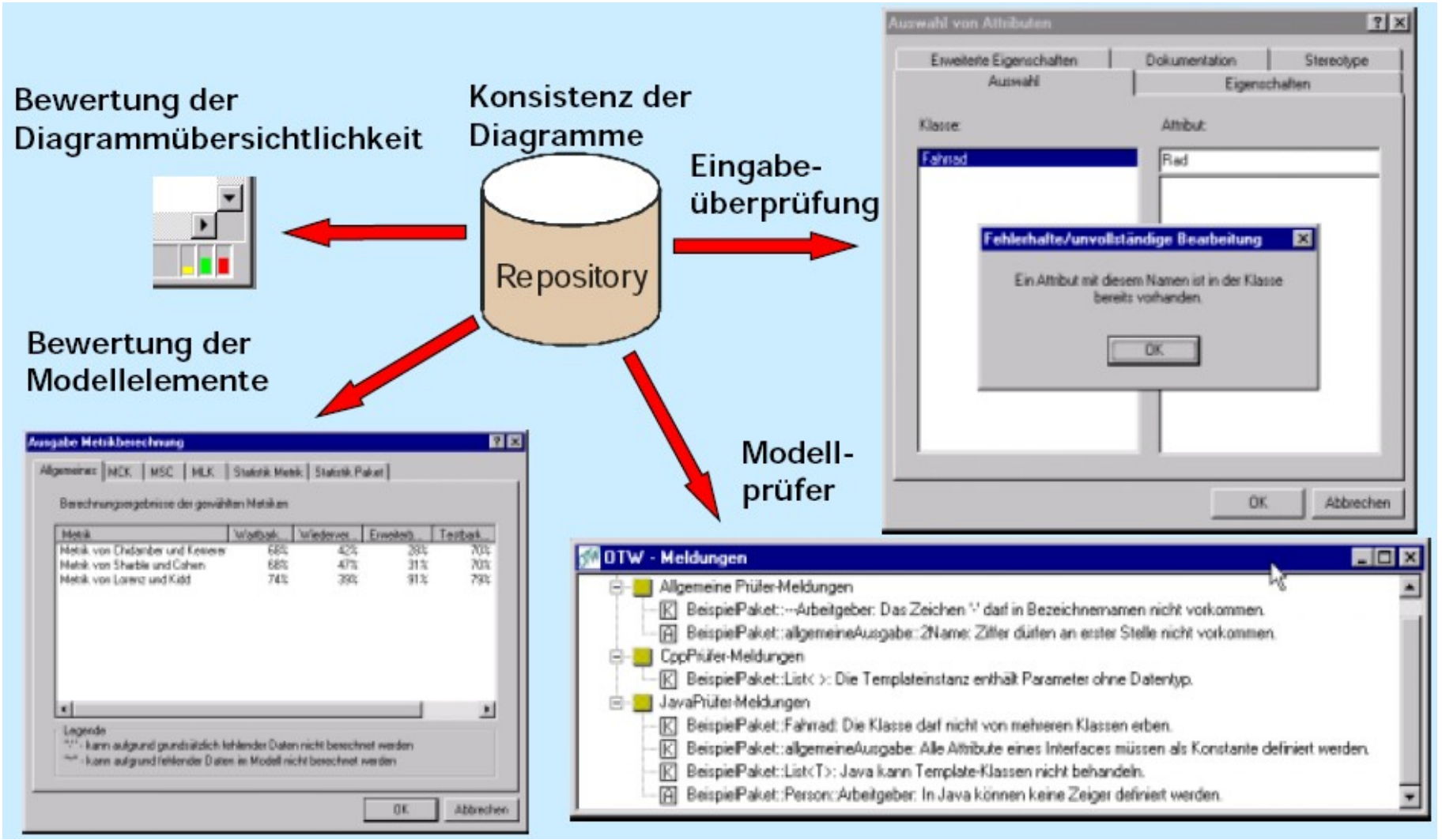
4+1 Architectural View Model [Kruchten]

Diagramme Refinement



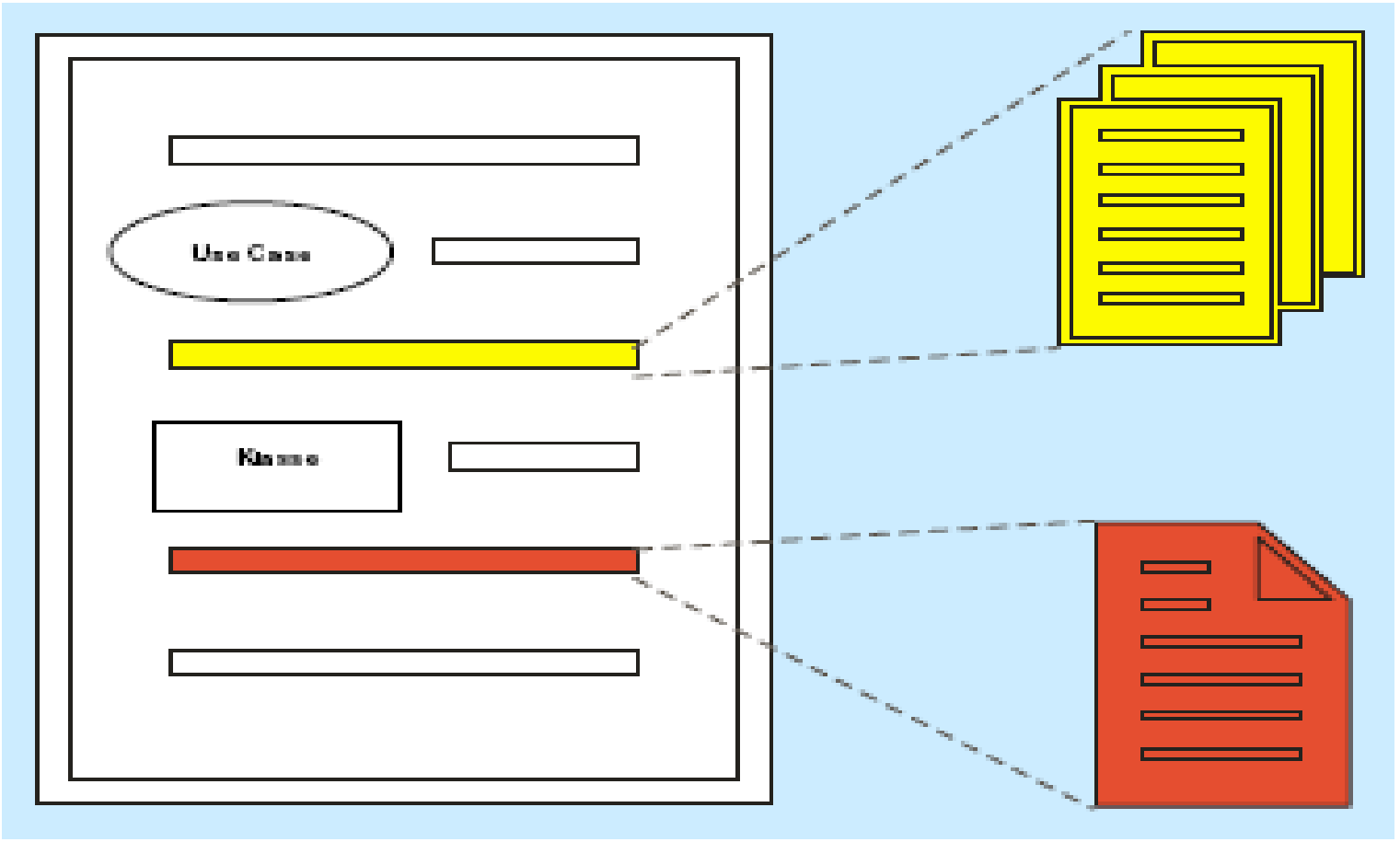
Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

Model Check



Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML. Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

Documentation Generation



Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

UML Compliance Level Matrix

Level 0-1-2-3

Information Flows, Templates, Packaging
Deployment, State Machines, Profiles
Use Cases, Interactions, Structures, Actions, Activities
Class-based Structures, Common OOP Elements, Primitive Types

Criteria:

- Abstract Syntax (Concepts)
- Concrete Syntax (Standard Notation)
- Diagram Interchange (DI)

Eliminated in UML 2.5, since not useful in practice
Tools have to support complete specification

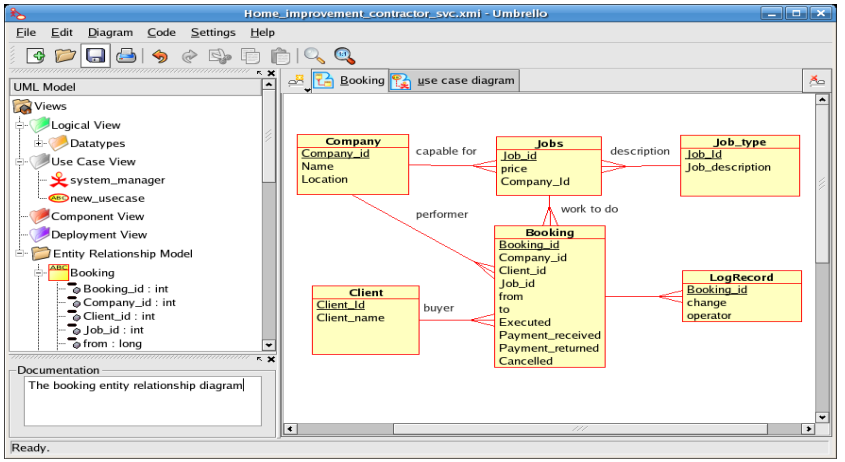
Proprietary UML Tools

- IBM Rational Software Architect (formerly Rose) (Java, C++)
- Borland Together
- Sparx Systems Enterprise Architect (C++)
- NoMagic MagicDraw (Java)
- MID Innovator, Nürnberg
- microTOOL ObjectiF (Java, C#, C++), Berlin
- Gentleware Poseidon (Java), based on ArgoUML, bei Freiburg
- MKLab StarUML (Delphi, Win32 only)
- Change Vision Astah (formerly JUDE, originally JOMT) (Java, C++, C#)
- Visual Paradigm (Java)
- ARIS Toolset, Darmstadt
- MS Visio (recommendation: stencils by Pavel Hruby NOT by Microsoft)

Open Source UML Tools

- BOUML (C++, Qt) http://bouml.free.fr/
- Modelio (Java) https://www.modelio.org/
- Umbrello (C++, Qt, KDE) http://uml.sf.net/
- ArgoUML (Java) http://argouml.tigris.org/
- Astade (C++, wxWidgets) http://astade.tigris.org/
- Fujaba (Java) http://www.fujaba.de/
- Papyrus (Java, Eclipse) http://www.topcased.org/
- UML Designer (Java, Sirius) http://www.umldesigner.org/
- WhiteStarUML (Object Pascal) http://whitestaruml.sourceforge.net/
- UMlet (Java) http://www.umlet.com/
- Dia (C, GTK+) https://wiki.gnome.org/Apps/Dia
- PlantUML (Java) http://plantuml.com/

```
https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools
http://www.oose.de/umltools.htm
```



Criterions

Obligatorische Anforderungen:

- FLOSS
- UML 2.x
- Repository und Logik (kein reines Zeichenwerkzeug)
- Generierung Quelltext und Dokumentation
- Stabilität
- Weiterentwicklung und neue Versionen
- Modellierbarkeit aller Altklausuren

Optionale Wünsche:

- Verfügbarkeit für GNU/Linux und Windows
- Bedienbarkeit gut, Reaktion schnell
- Musterkatalog und Musterinstantiierung

Summary

- model statics & dynamics
- embed and refine diagramme element
- check model consistency
- reuse models and patterns
- document and communicate
- generate and import source code

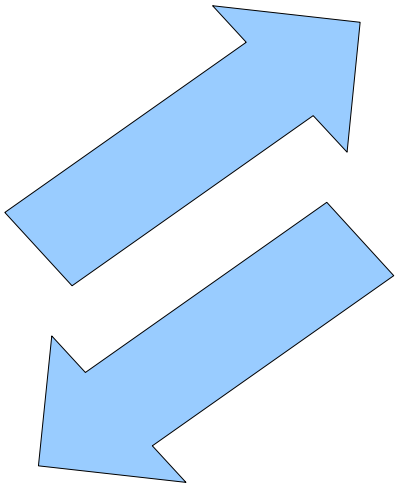
Contents

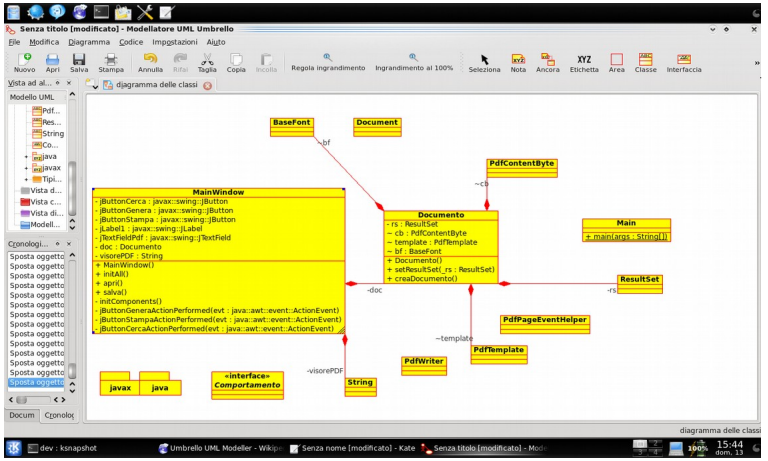
Introduction
Definition and Placement
Unified Modeling Language (UML)
Computer Aided Software Engineering (CASE)
Model Driven Architecture (MDA)
UML Meta Model
Meta Object Facility (MOF)
Profile and Stereotype
Object Constraint Language (OCL)
XML Metadata Interchange (XMI)
Diagram Definition (DD)
Related Specification Languages
Software Pattern
Pattern Catalogue

Round Trip Engineering (RTE)

Forward Engineering

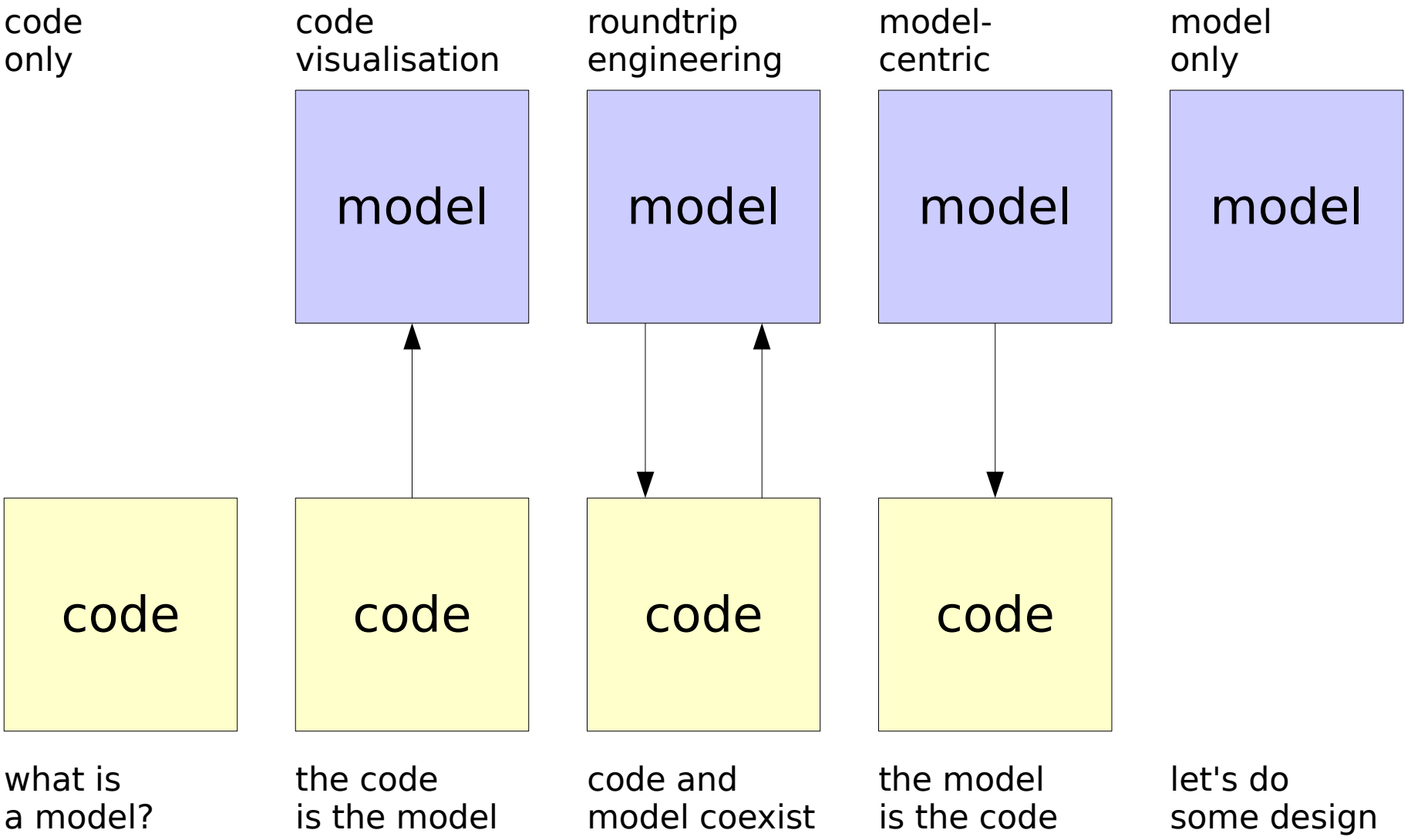
```java
// OddEven.java
import javax.swing.JOptionPane;

public class OddEven {
    // "input" is the number that the user gives to the computer
    private int input; // a whole number("int" means integer)

    /*
     * This is the constructor method. It gets called when an obj
     * is being created.
     */
    public OddEven() {
    /*
     * Code not shown for simplicity.  In most Java programs const
     * with default values, or create other objects that this obj
     * functions.  In some Java programs, the constructor may simp
     * needs to be initialized prior to the functioning of the ob
     * empty constructor would suffice, even if it is empty. A co
     * user doesn't put one in then the compiler will create an e
     */
    }

    // This is the main method. It gets called when this class is
    public static void main(String[] args) {
        /*
         * This line of code creates a new instance of this class
         * Object) and initializes it by calling the constructor.
         * the "showDialog()" method, which brings up a prompt to
         */
        OddEven number = new OddEven();
        number.showDialog();
    }

    public void showDialog() {
        /*
```



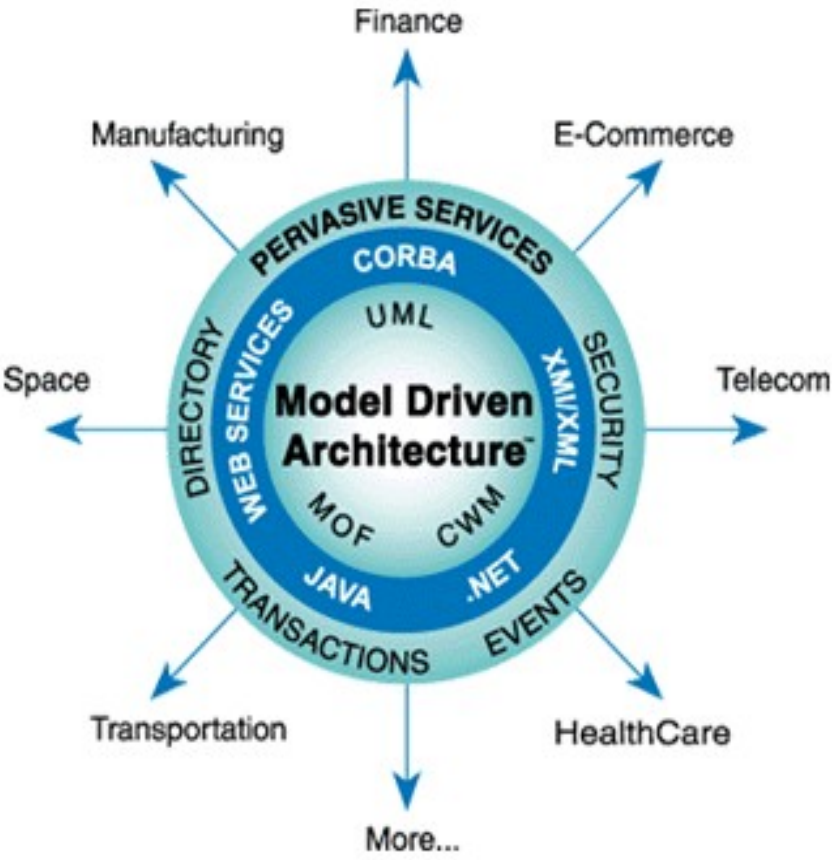


Reverse Engineering

Model-Code Synchronisation

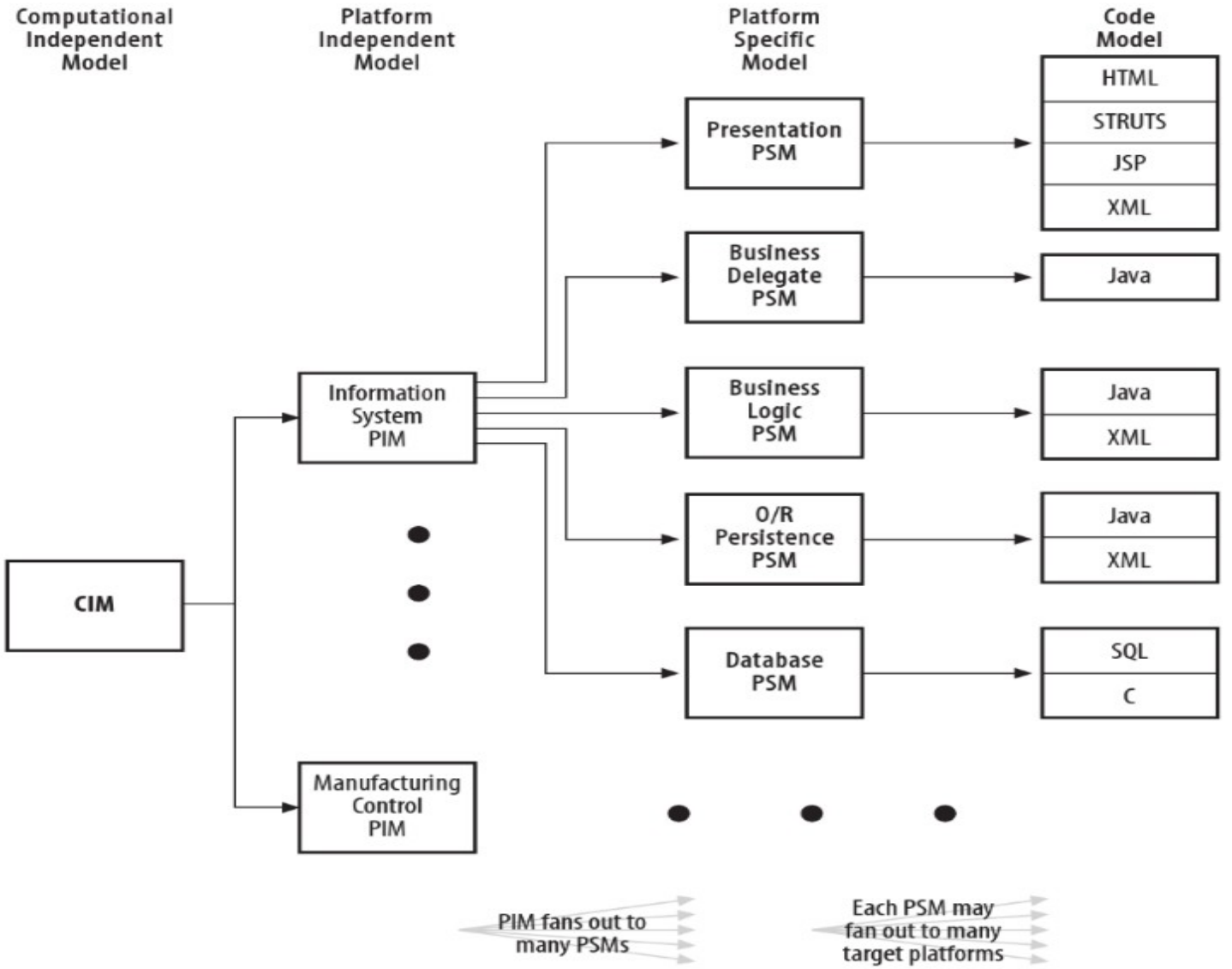


Model Driven Architecture (MDA)



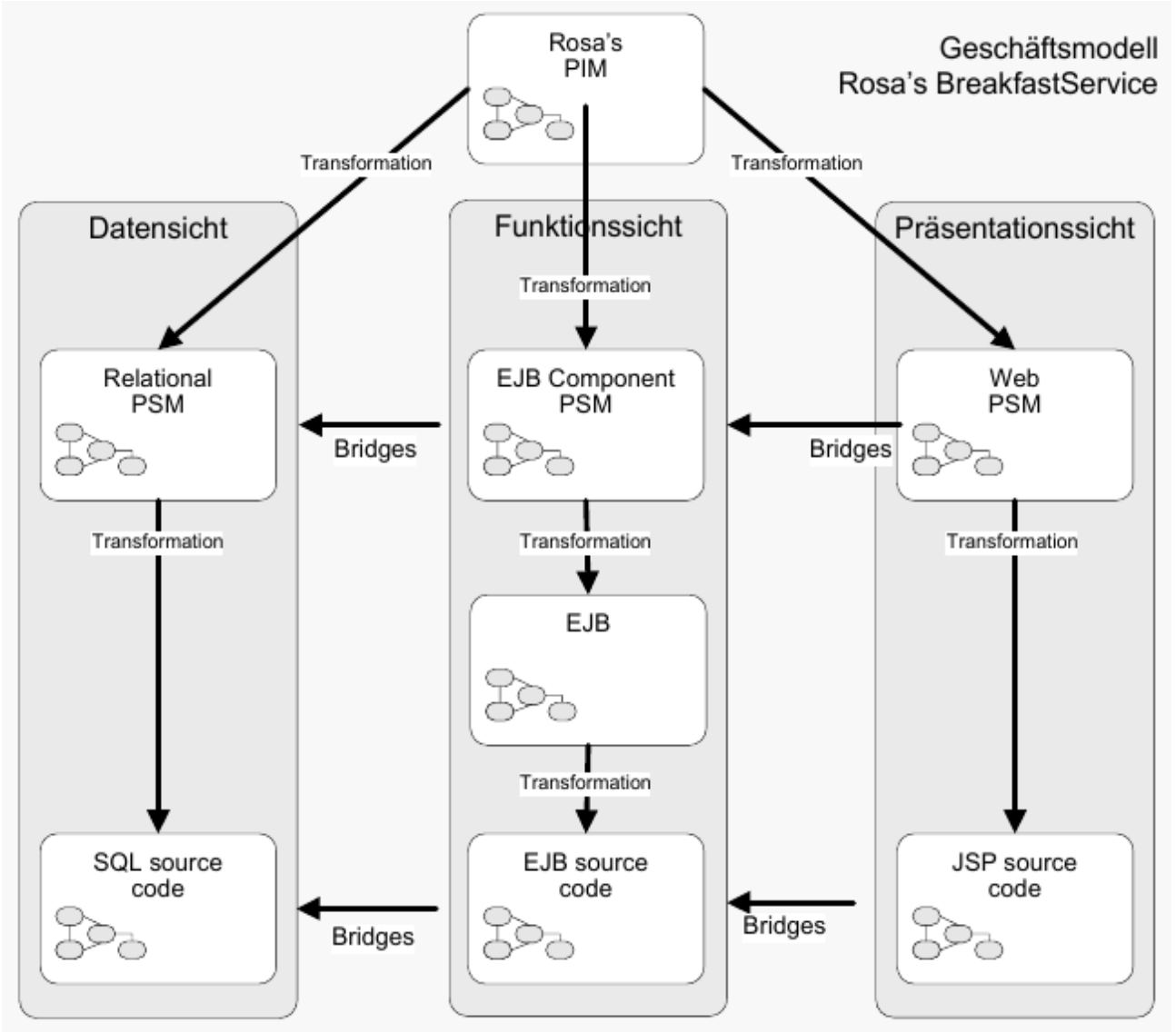
http://www.omg.org/mda/
https://www.gi.de/service/informatiklexikon/detailansicht/article/model-driven-architecture.html

MDA Models and Relationships



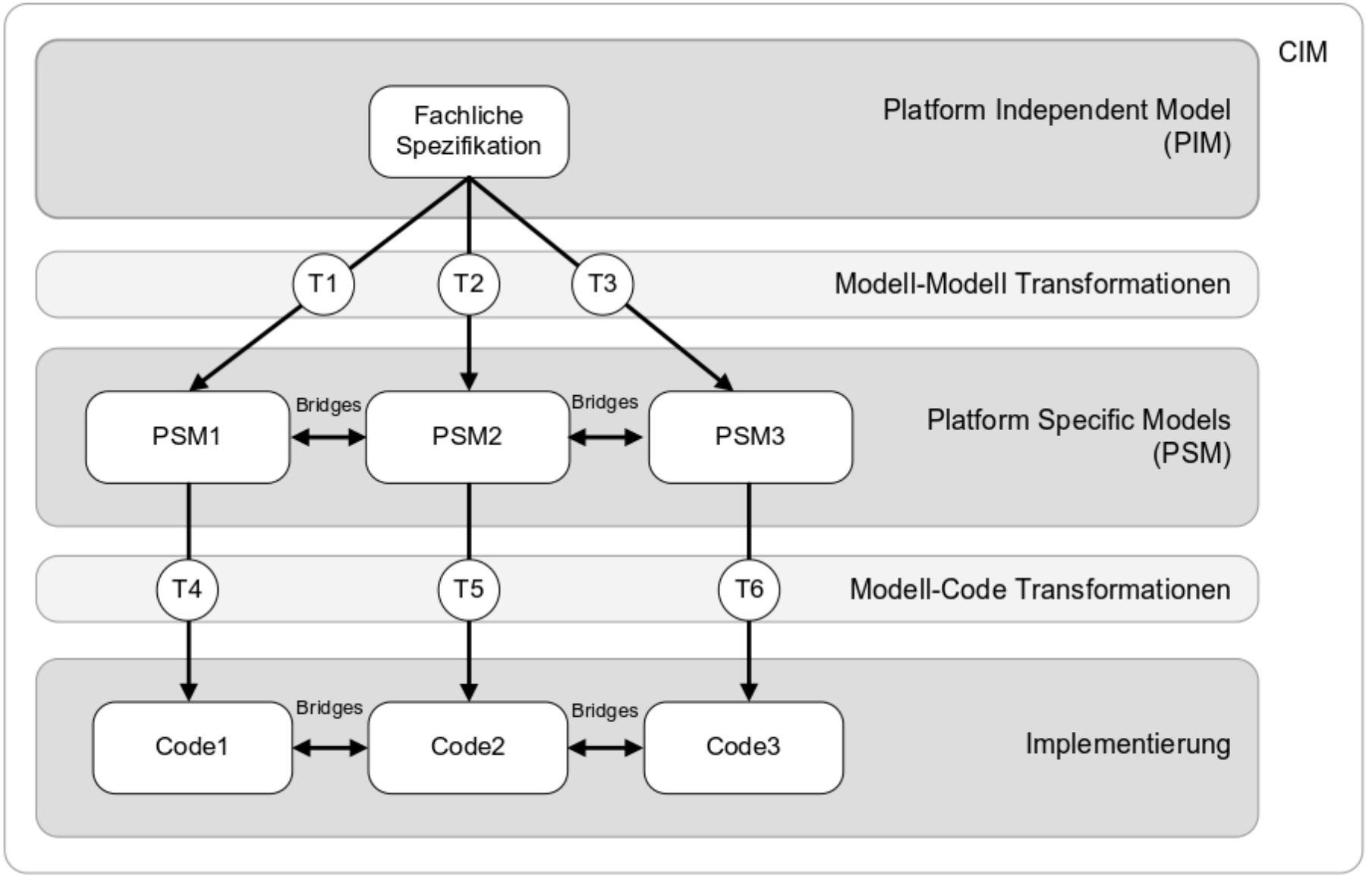
Dan Pilone, Neil Pitman: UML 2.0 in a Nutshell. O'Reilly Media, 2005

MDA Translation Example "Breakfast Service"



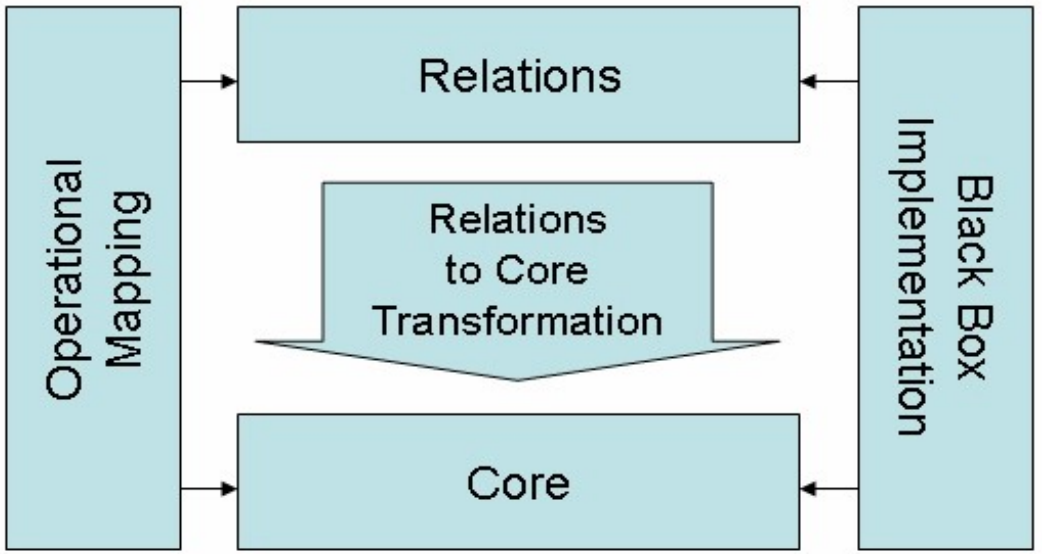
Markus Pepping http://is.uni-muenster.de/pi/lehre/ws0506/seminar/01_mda_folien.pdf

MDA Translation Process



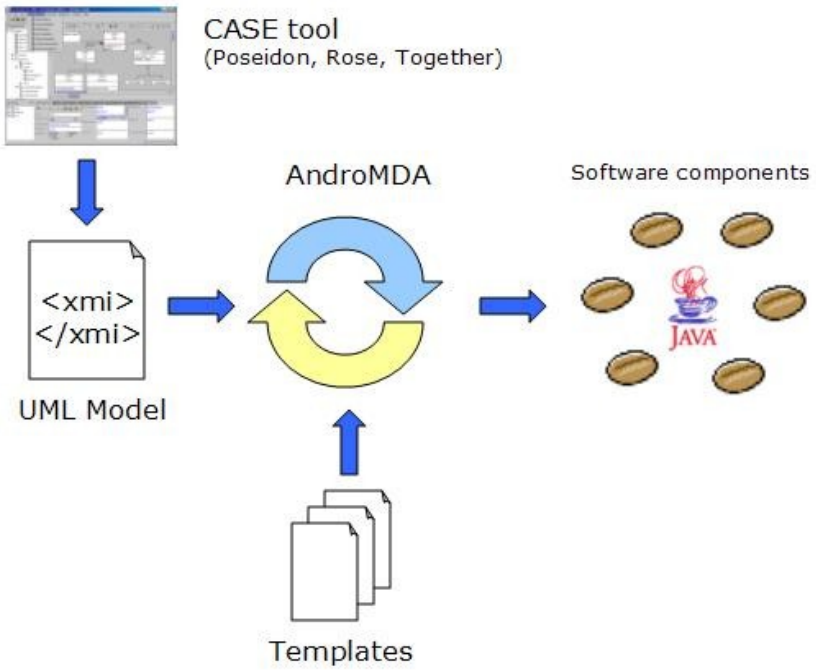
Markus Pepping http://is.uni-muenster.de/pi/lehre/ws0506/seminar/01_mda_folien.pdf

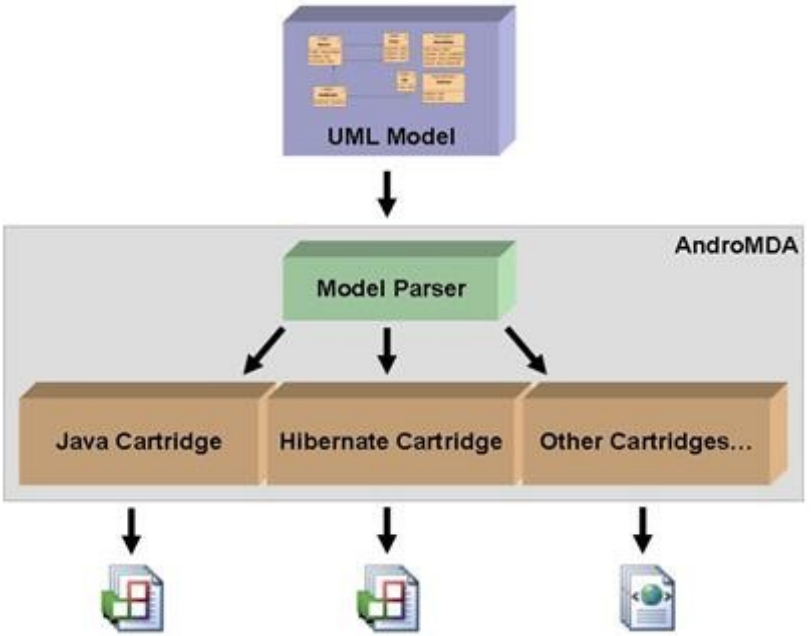
Query/View/Transformation (QVT) Languages



https://de.wikipedia.org/wiki/MOF_QVT

AndroMDA Framework Example





MTL

PIM → **PSM**

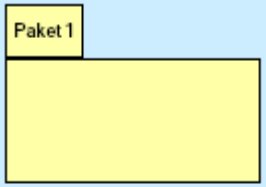
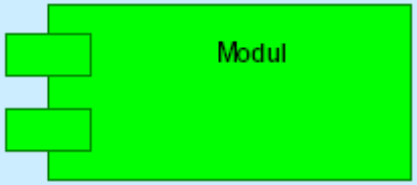
AndroMDA. Code Generator Framework. Apache Project. http://db.apache.org/
http://andromda.sourceforge.net/andromda-documentation/getting-started-java/index.html

Advantages and Deficiencies

- Creation of specification
- Abstraction of problems
- Automisation of tasks
- Reuse of components
- Consistency of models

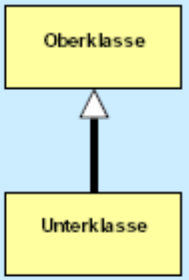
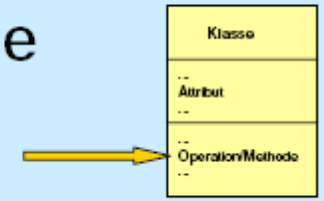
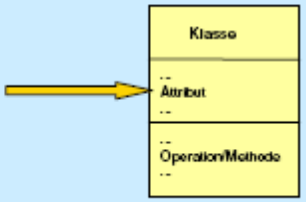
- Reverse engineering
- Transformation language
- Proprietary solutions
- Transformation of profiles
- Migration of data/application
- Cost and experience

UML Element - Source Code Interpretation 1

UML-Element	C++-Quelltext	Java-Quelltext
Paket 	Namensraum/ Verzeichnis	Package
Modul 	Deklarations- und Implementierungs-datei (.h- und .cpp)	Java-Datei (.java)
Klasse 	Klasse	Klasse

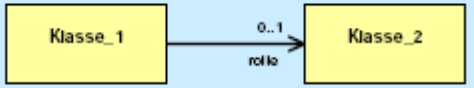
Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

UML Element - Source Code Interpretation 2

UML-Element	C++-Quelltext	Java-Quelltext
Vererbung 	Vererbung	Vererbung: «extends» oder «implements» (je nach Basisklasse)
Methode 	Methode einer Klasse	Methode einer Klasse
Attribut 	Attribut einer Klasse	Attribut einer Klasse

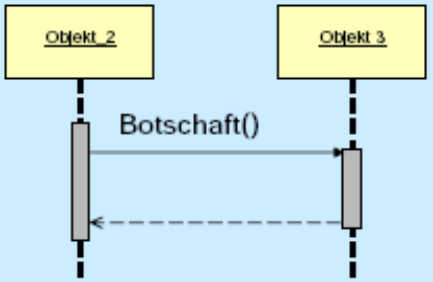
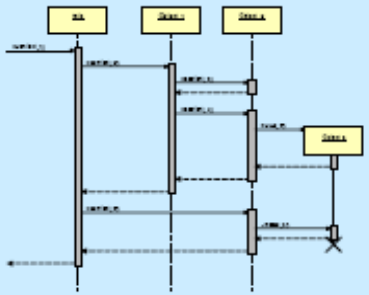
Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

UML Element - Source Code Interpretation 3

UML-Element	C++-Quelltext	Java-Quelltext
Rolle 	Attribut einer Klasse	Attribut einer Klasse
Kardinalität und Rollenklasse der Rolle 	Datentyp des Attributs	Datentyp des Attributs
Objekt 	Globales, lokales, Parameterobjekt	Globales, lokales, Parameterobjekt

Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

UML Element - Source Code Interpretation 4

UML-Element	C++-Quelltext	Java-Quelltext
Botschaft 	Aufruf einer Methode	Aufruf einer Methode
Sequenzdiagramm 	Implementierung einer Methode	Implementierung einer Methode

Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

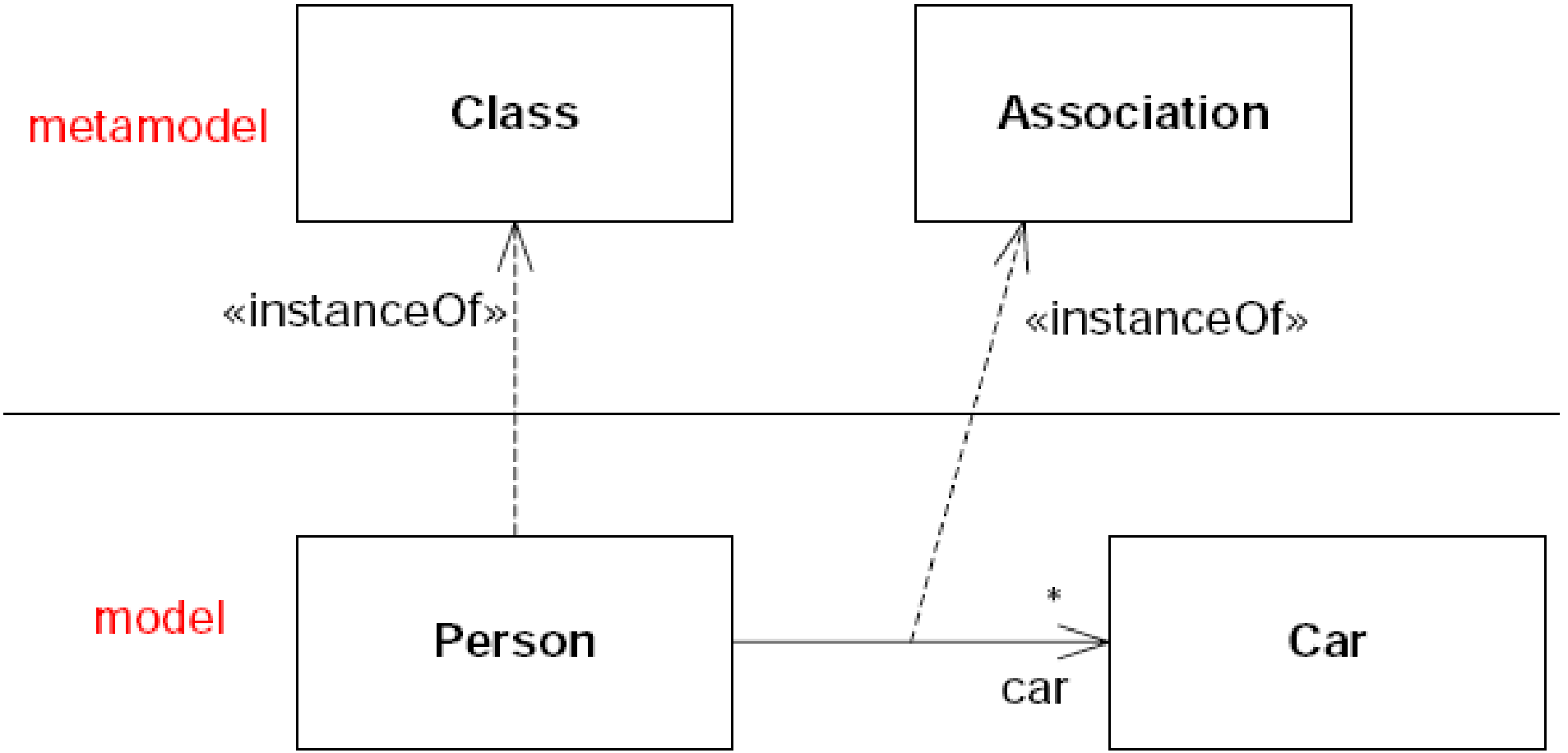
Summary

- Forward- & Reverse Engineering
- Model & Code
- MDA: PIM & PSM
- Translation Process
- Query/View/Transformation (QVT)

Contents

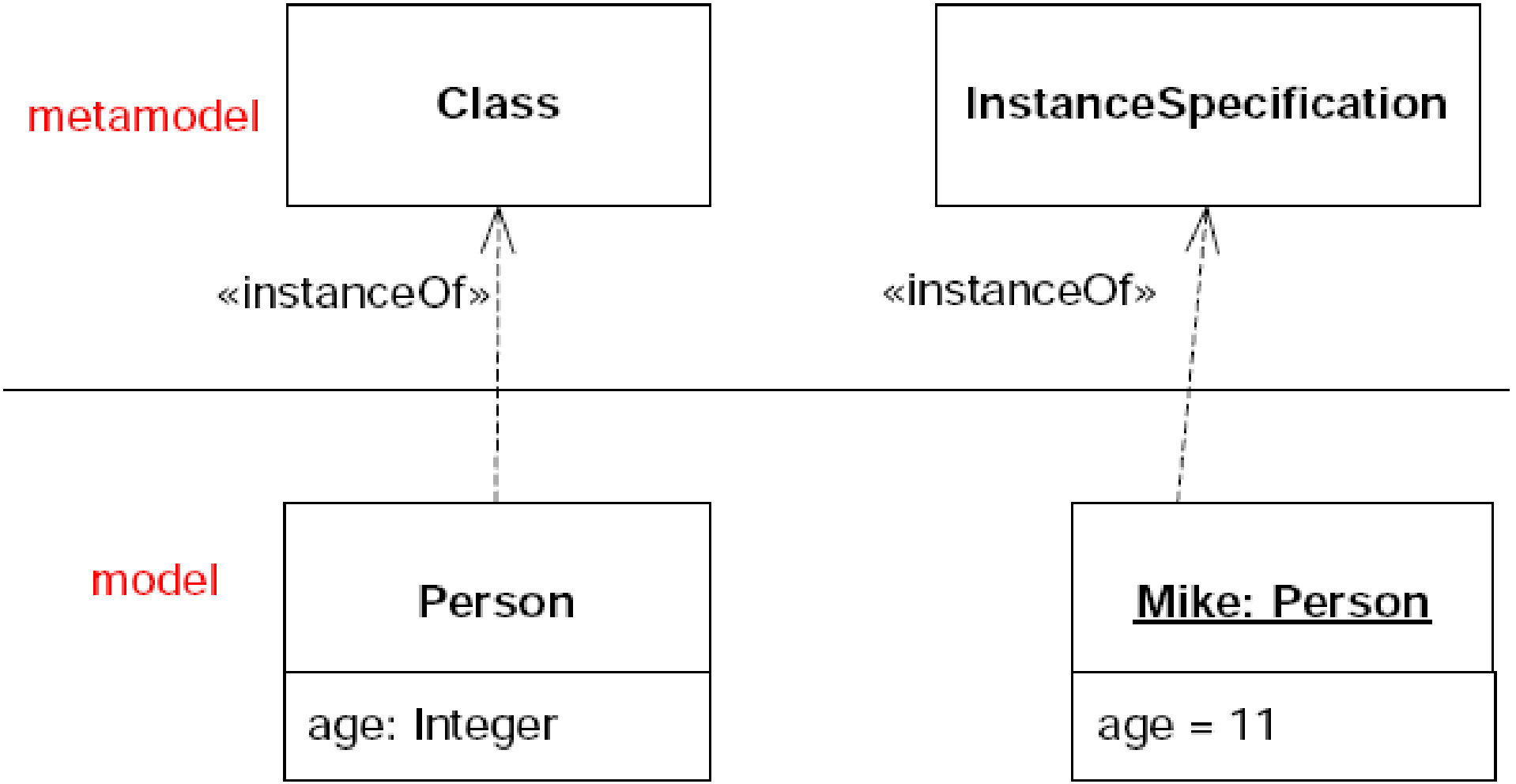
Introduction
Definition and Placement
Unified Modeling Language (UML)
Computer Aided Software Engineering (CASE)
Model Driven Architecture (MDA)
UML Meta Model
Meta Object Facility (MOF)
Profile and Stereotype
Object Constraint Language (OCL)
XML Metadata Interchange (XMI)
Diagram Definition (DD)
Related Specification Languages
Software Pattern
Pattern Catalogue

UML Meta Model Example: Class and Association



Object Management Group (OMG). http://www.omg.org/

UML Meta Model Example: Class and Instance



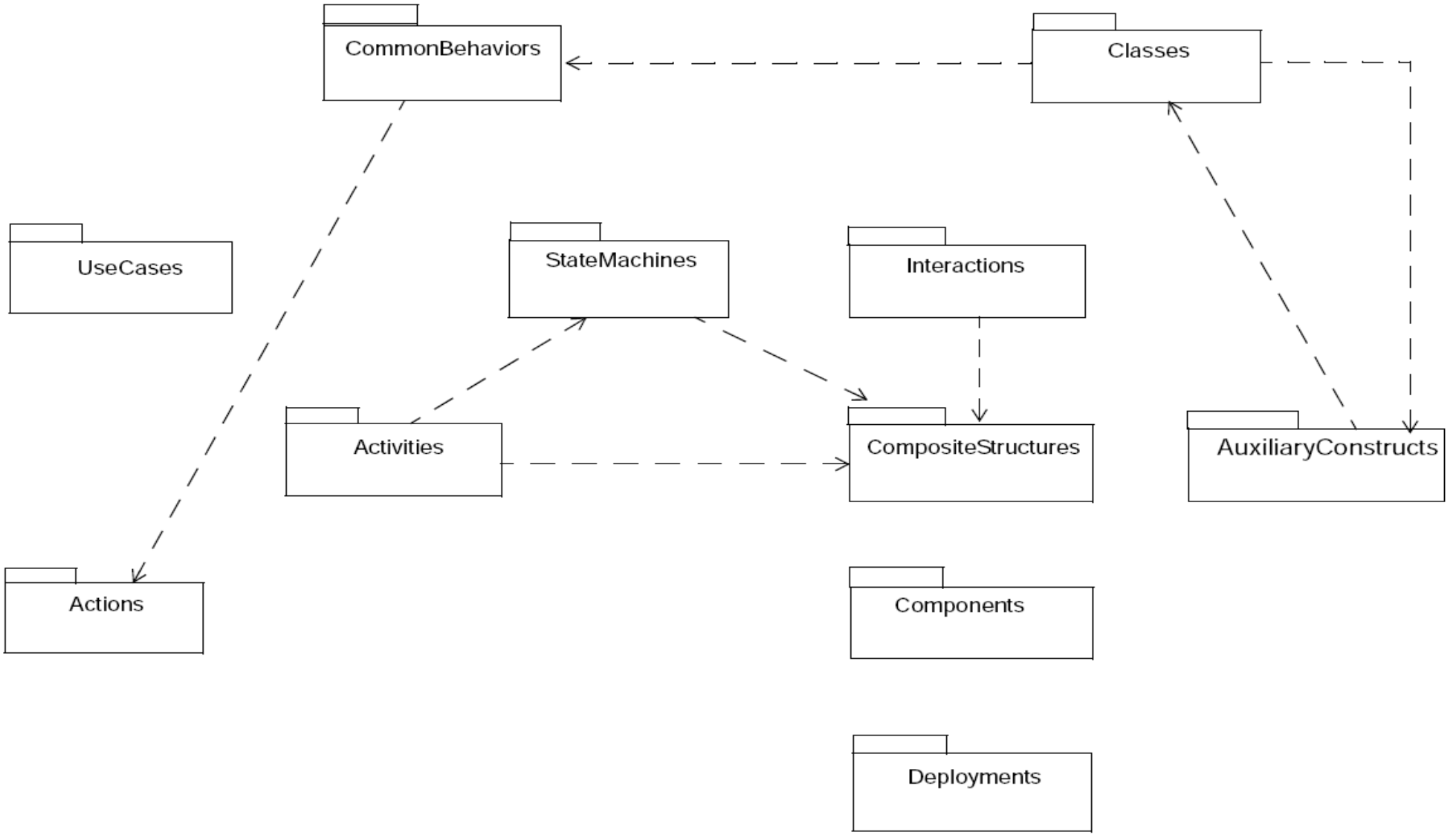
Object Management Group (OMG). http://www.omg.org/

UML Infrastructure Library Specifications

- Infrastructure + Superstructure (merged since 2.5)
- Object Constraint Language (OCL)

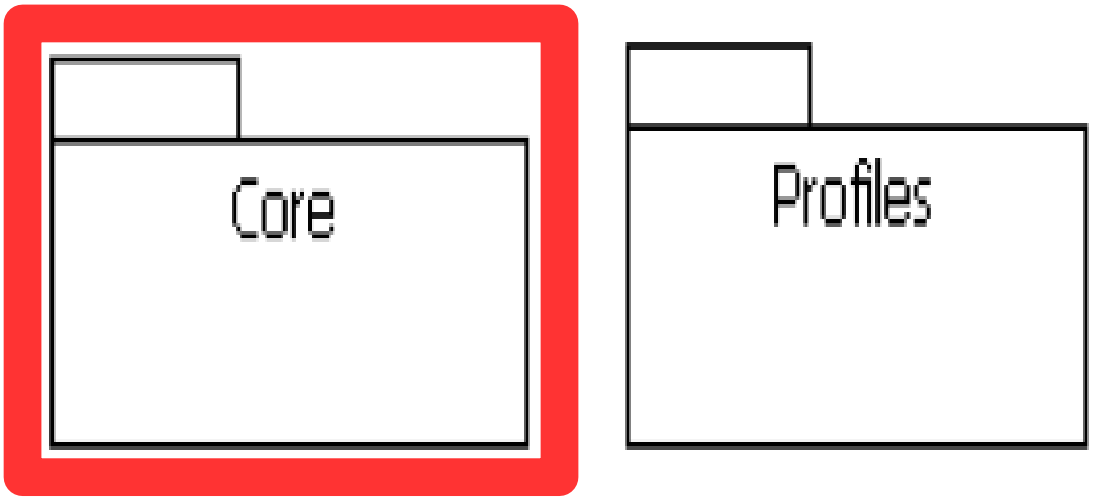
UML 2.5 specification is a single document (earlier versions separated infrastructure and superstructure)

Superstructure (Top Level Packages)



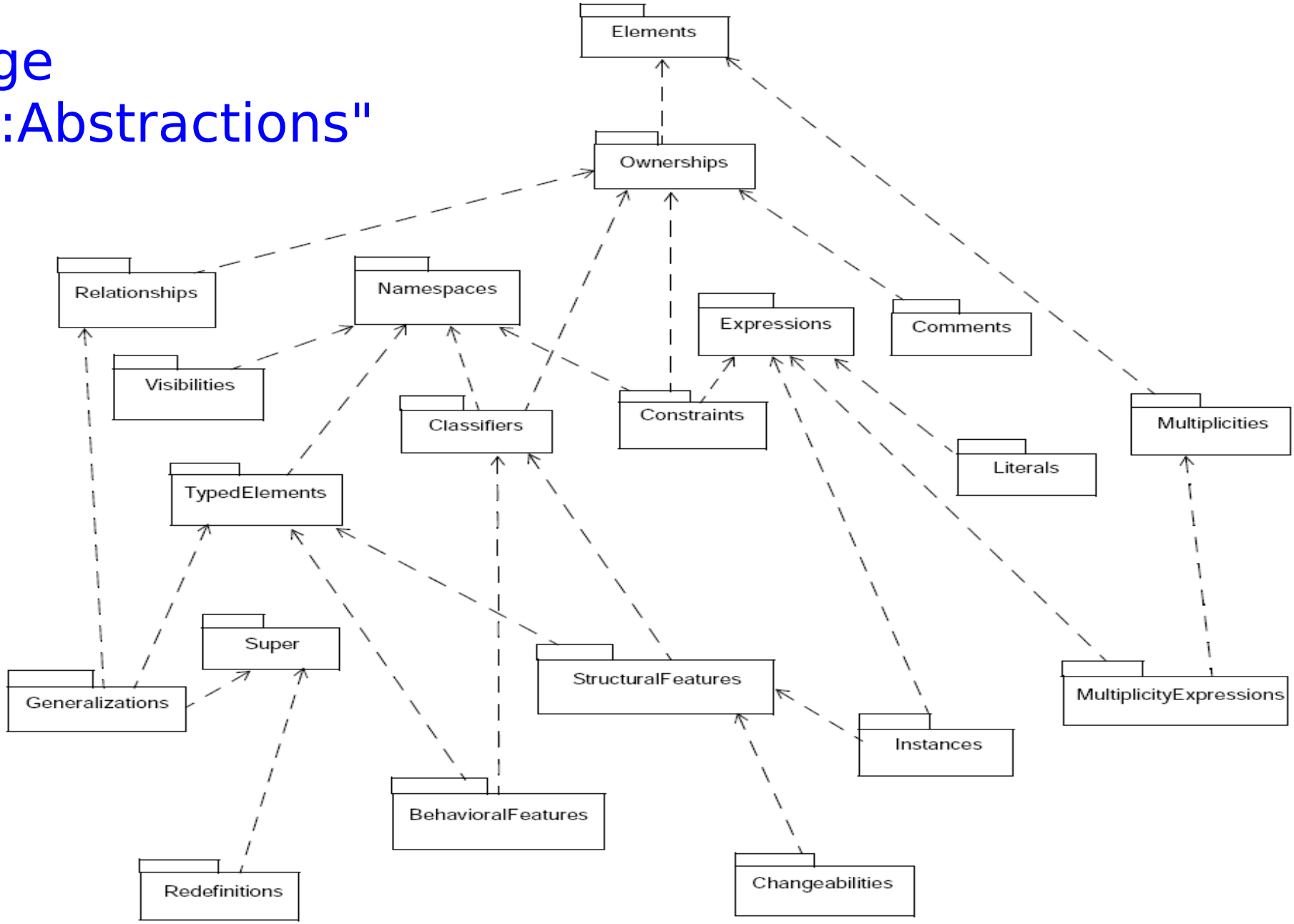
Object Management Group (OMG). http://www.omg.org/

Infrastructure Library



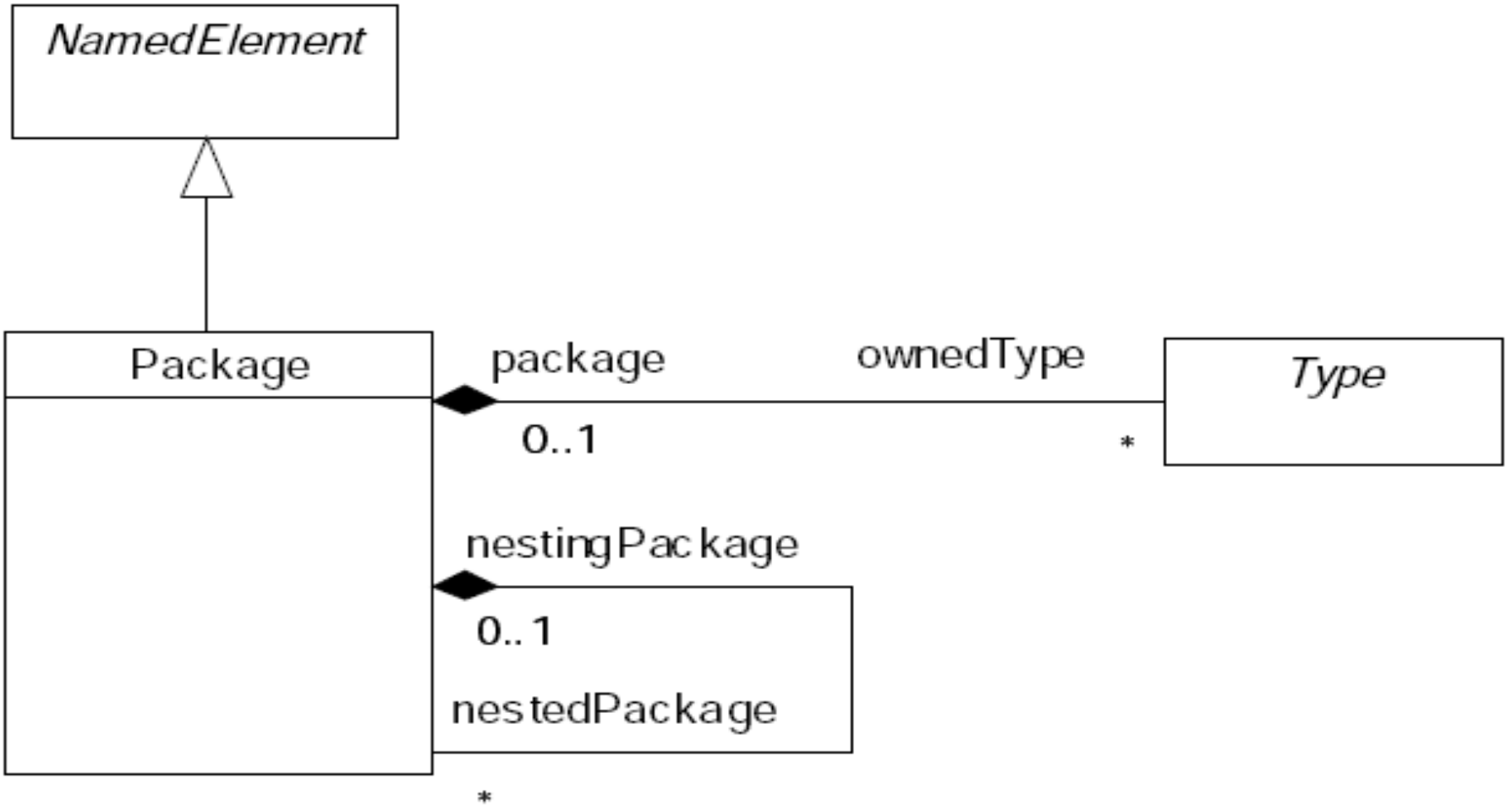
Object Management Group (OMG). http://www.omg.org/

Package "Core::Abstractions"



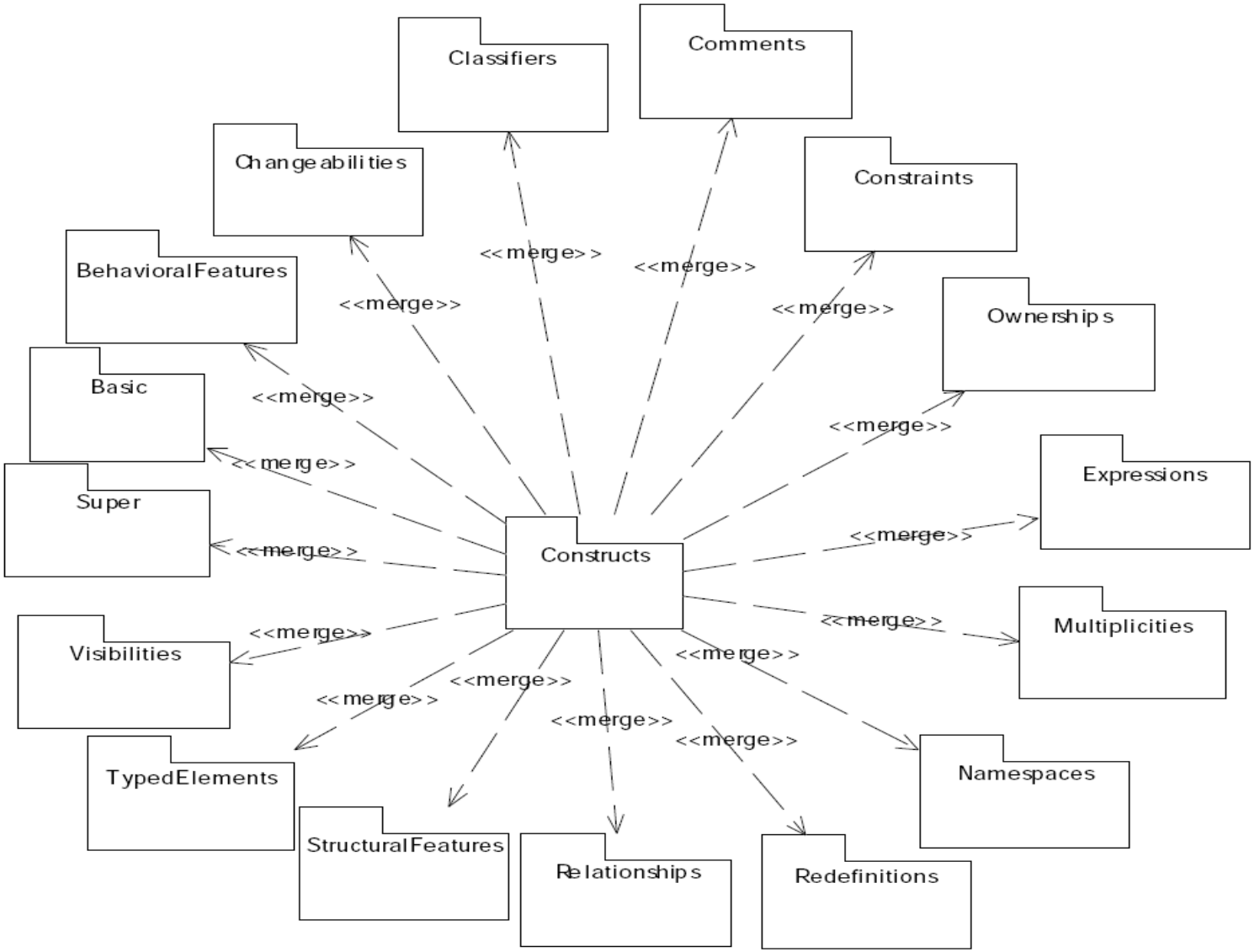
Object Management Group (OMG). http://www.omg.org/

Package "Core::Basic": Packages Diagram



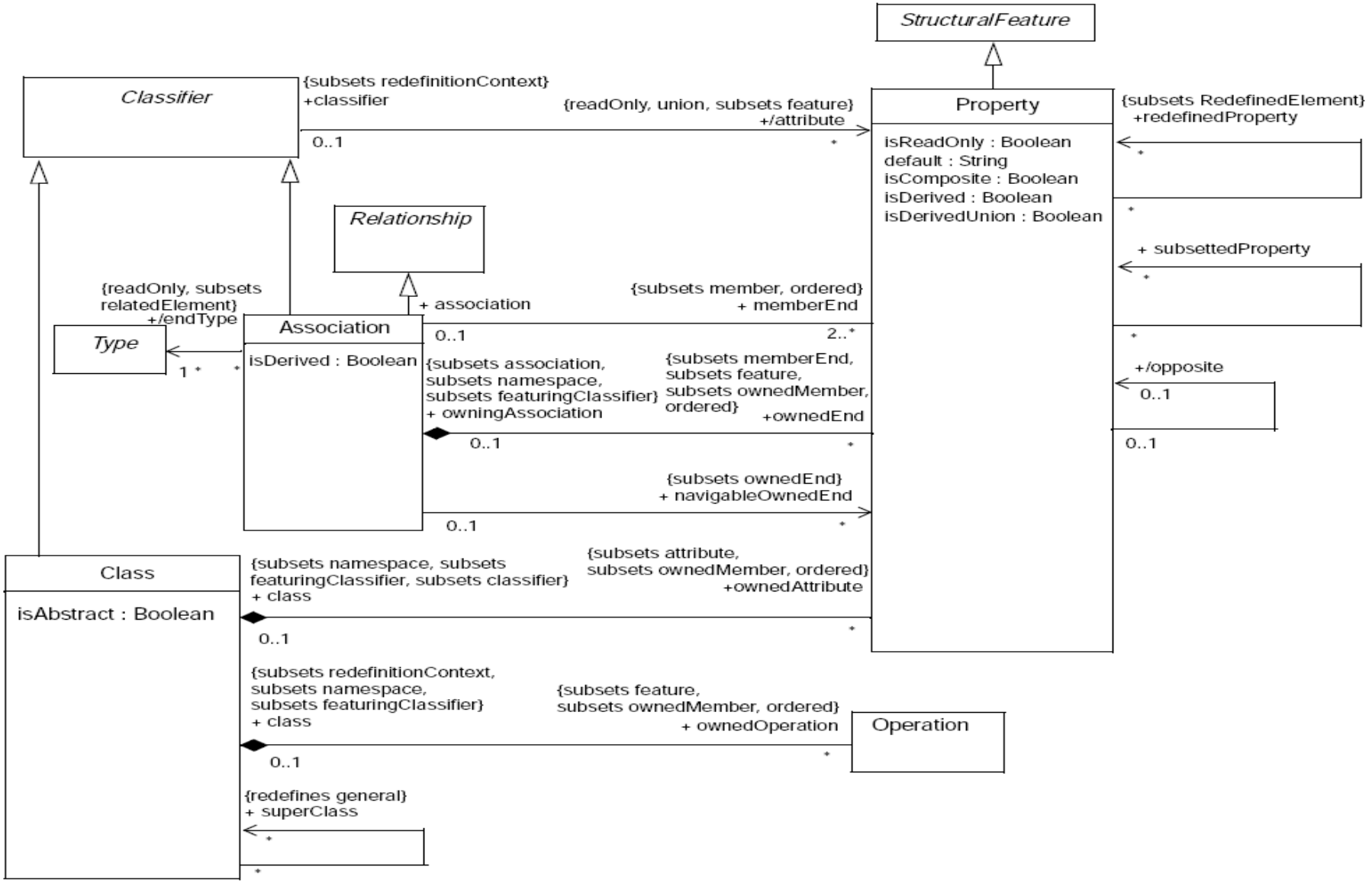
Object Management Group (OMG). http://www.omg.org/

Package "Core::Constructs": Sub Packages



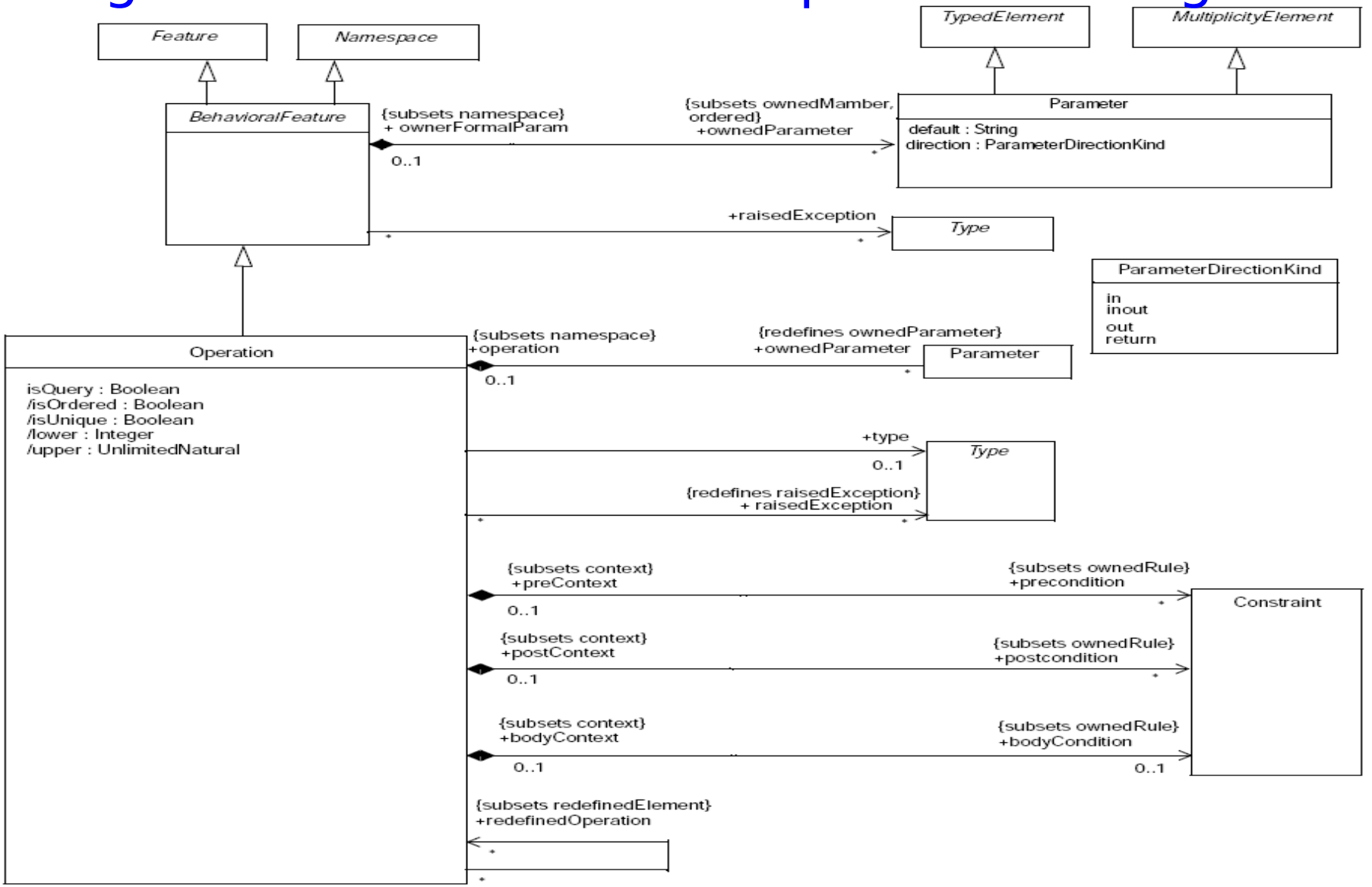
Object Management Group (OMG). http://www.omg.org/

Package "Core::Constructs": Classes Diagram



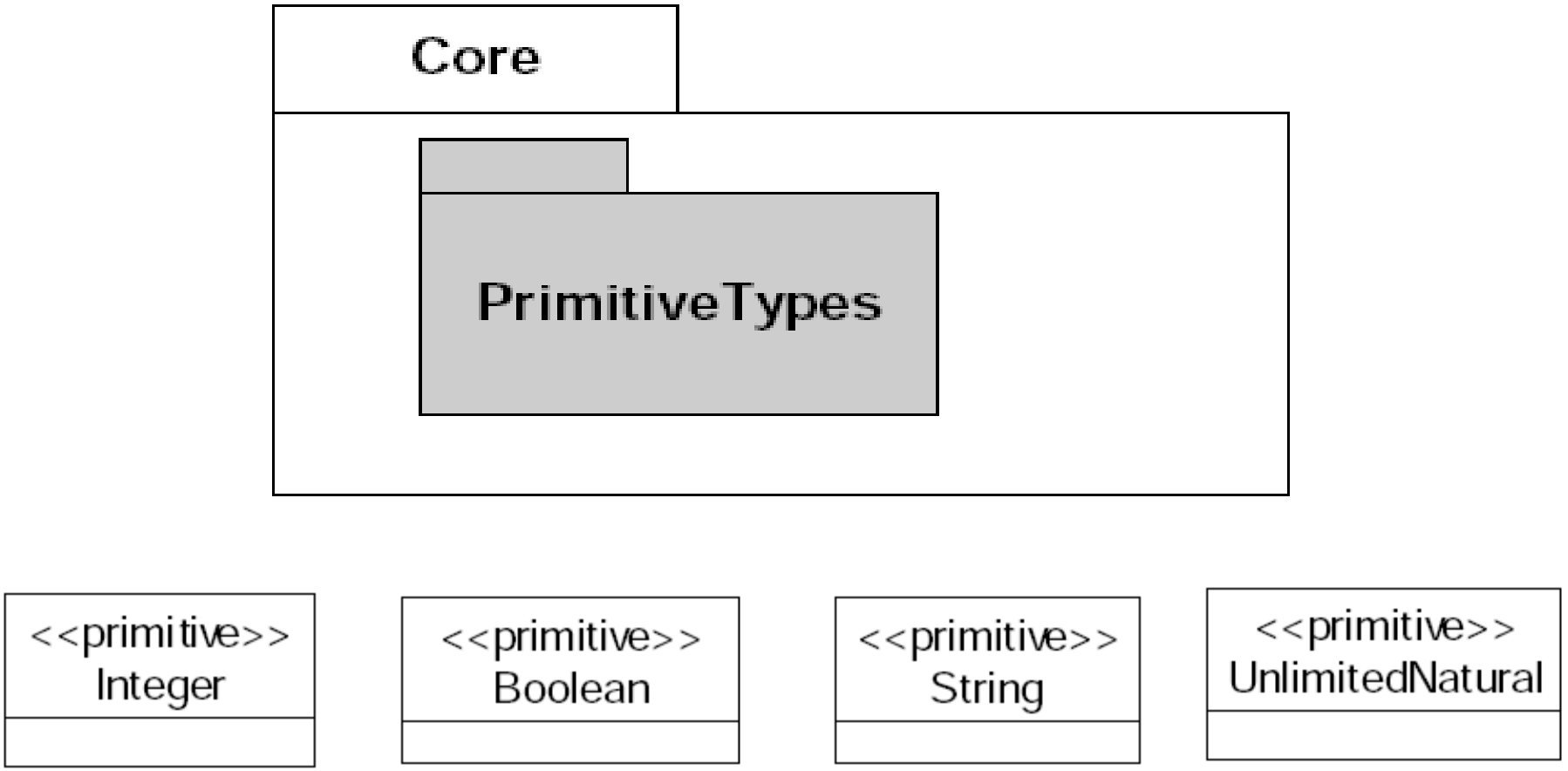
Object Management Group (OMG). http://www.omg.org/

Package "Core::Constructs": Operations Diagram



Object Management Group (OMG). http://www.omg.org/

Package "Core::Primitive Types" with Classes



Object Management Group (OMG). http://www.omg.org/

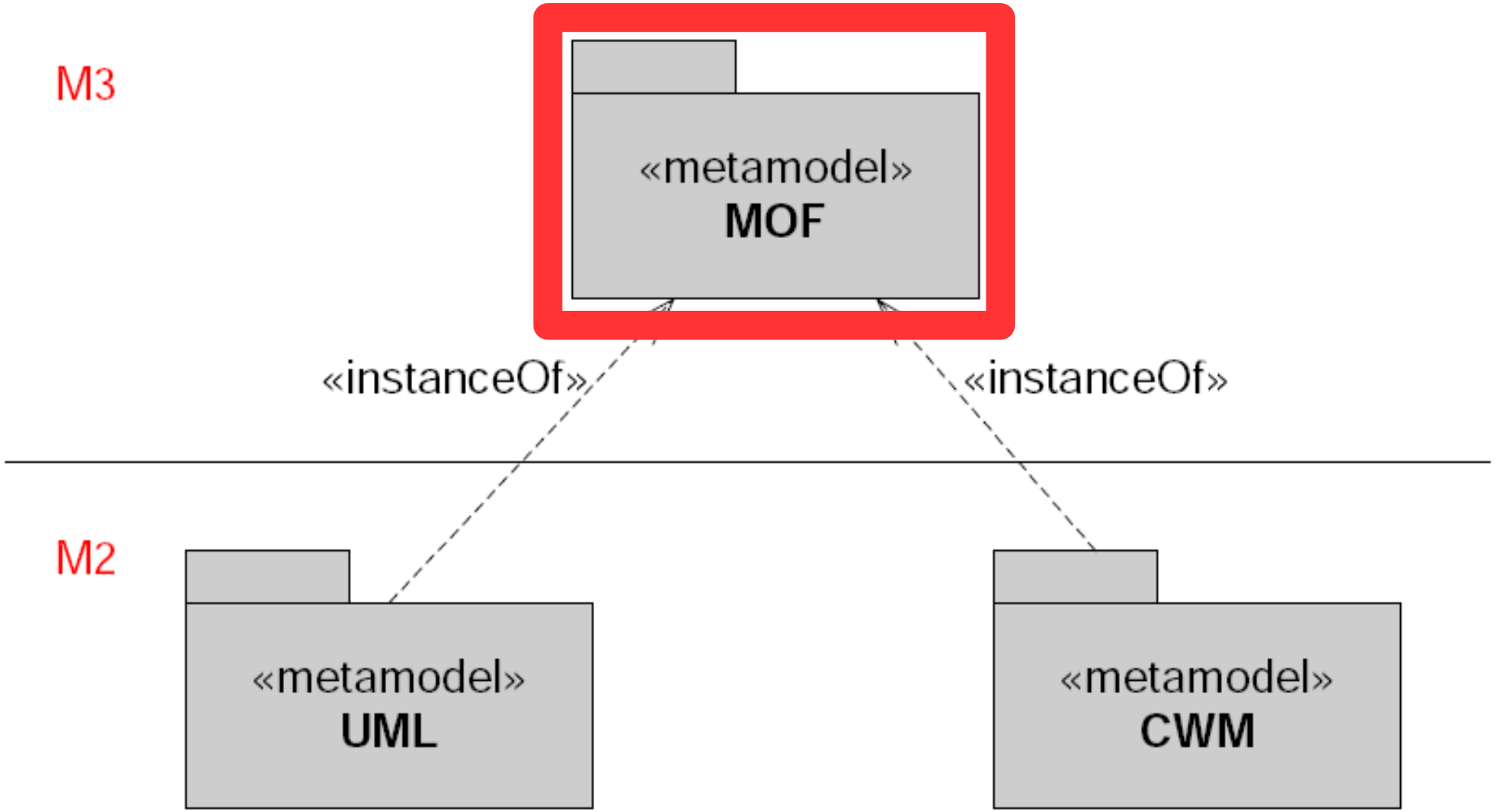
Summary

- Meta model specifies basic UML elements and relations
- Superstructure: class, component, state machine
- Infrastructure library core:
  - primitive types: e.g. integer, boolean
  - basic: e.g. package
  - abstractions: e.g. namespace, classifier, instance
  - constructs: e.g. super, multiplicity, visibility

Contents

Introduction
Definition and Placement
Unified Modeling Language (UML)
Computer Aided Software Engineering (CASE)
Model Driven Architecture (MDA)
UML Meta Model
Meta Object Facility (MOF)
Profile and Stereotype
Object Constraint Language (OCL)
XML Metadata Interchange (XMI)
Diagram Definition (DD)
Related Specification Languages
Software Pattern
Pattern Catalogue

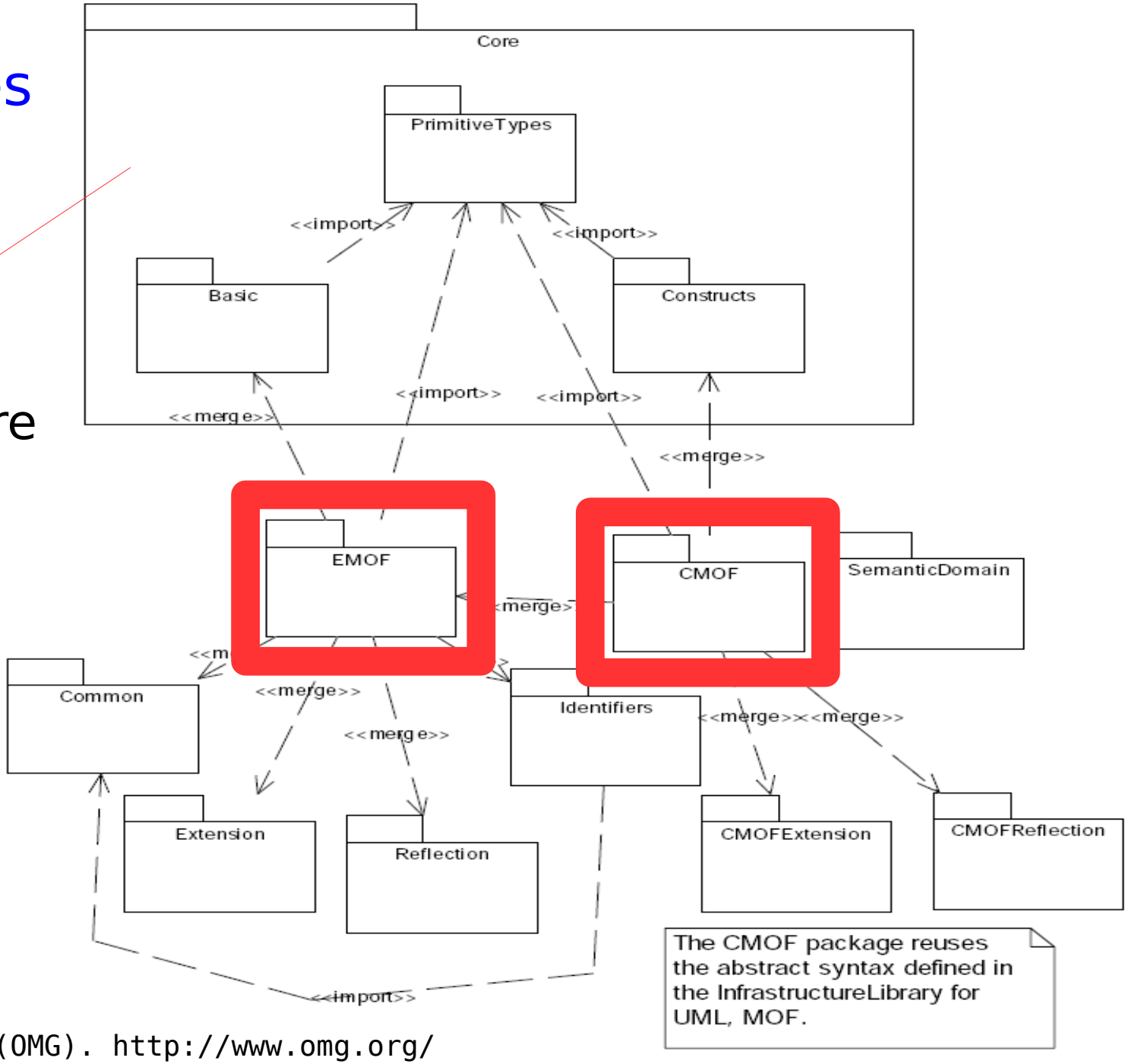
UML and MOF at different Meta Levels



Object Management Group (OMG). http://www.omg.org/

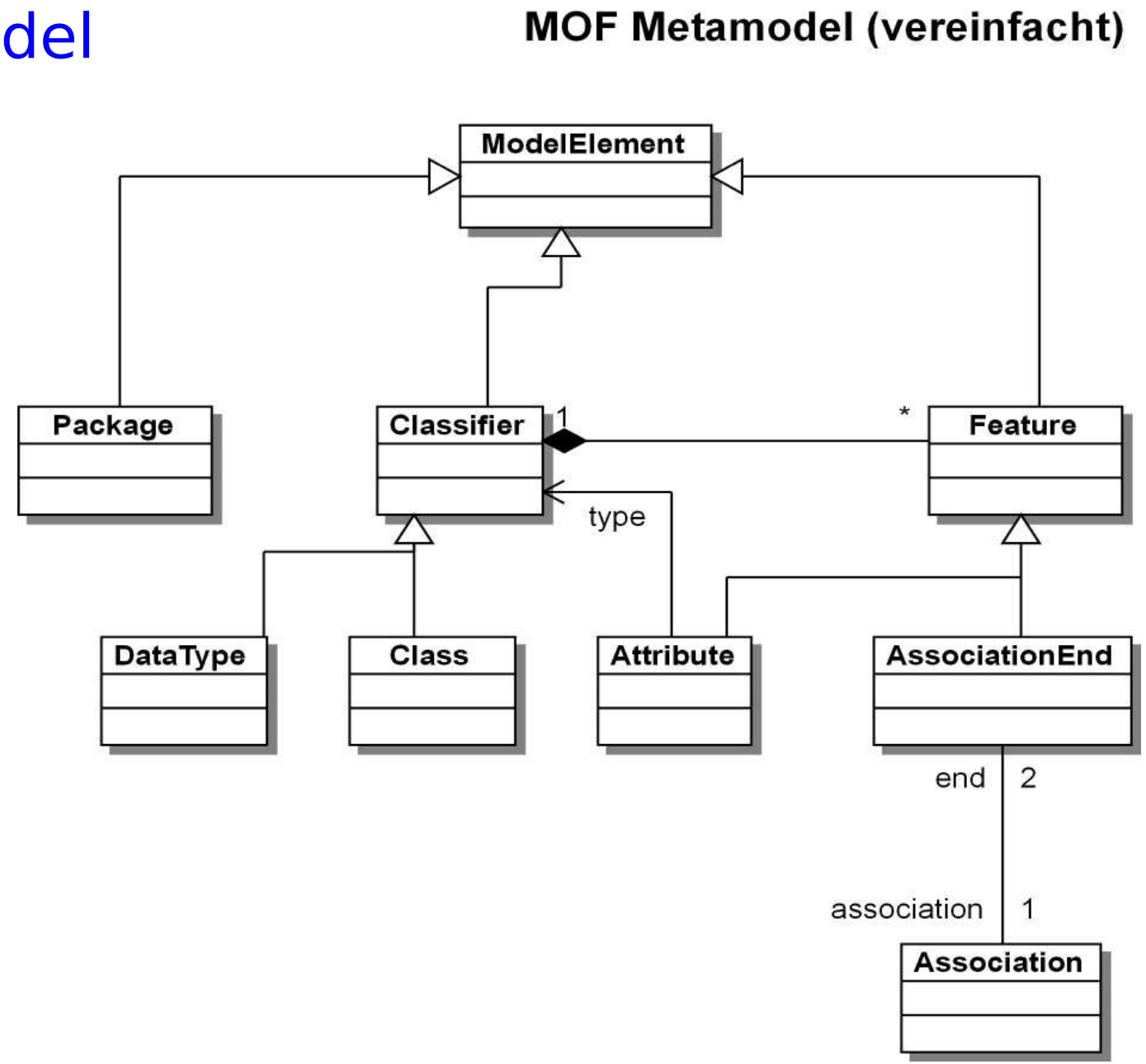
CMOF Packages

MOF reuses a subset of the UML Infrastructure Library packages



Object Management Group (OMG). http://www.omg.org/

MOF Meta Model



Object Management Group (OMG). http://www.omg.org/

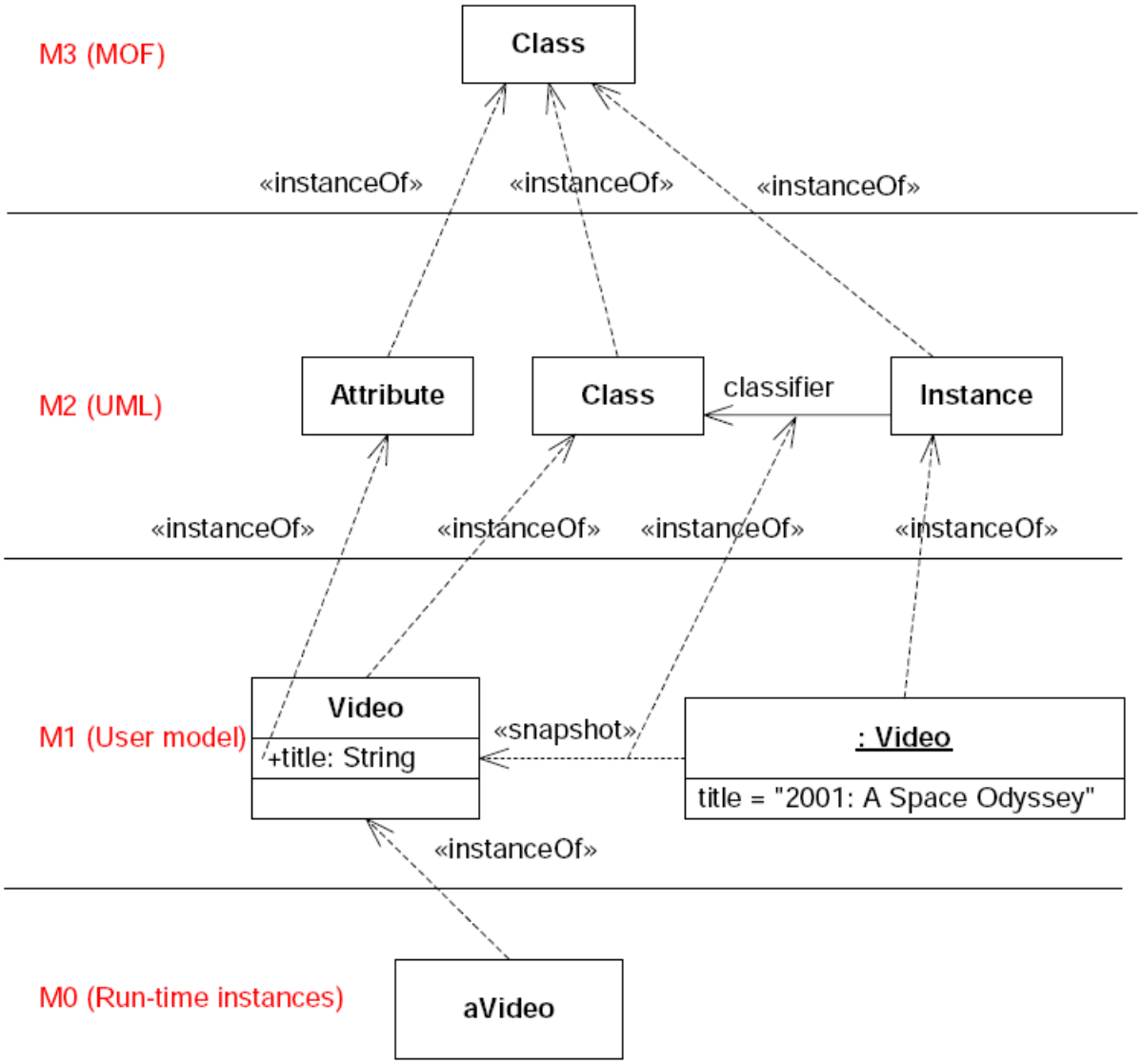
Four-Layer Meta Model Hierarchy

meta meta model

meta model/ language specification

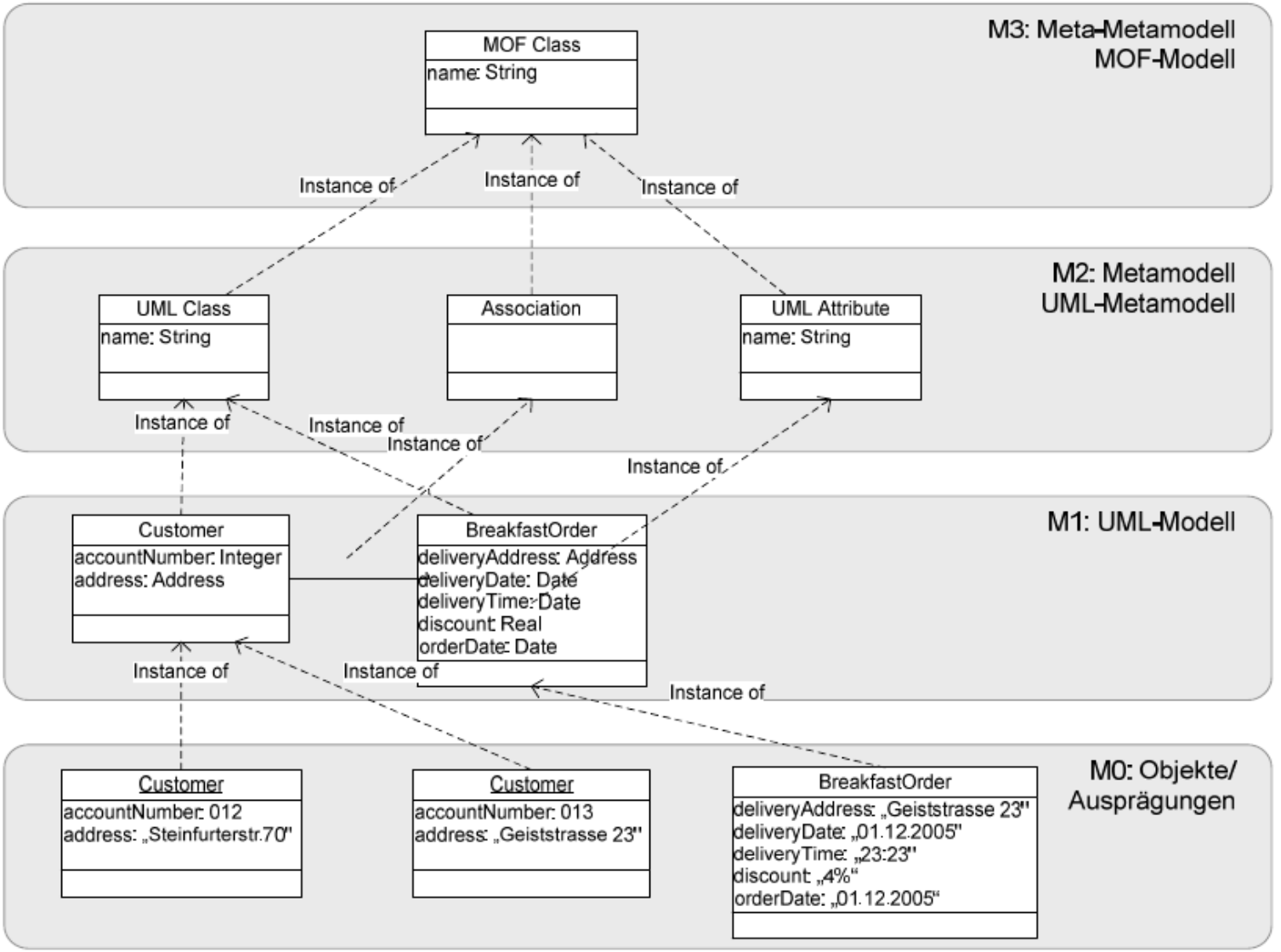
model/ user specification

instances/ objects of the model



Object Management Group (OMG). http://www.omg.org/

MOF Example "Breakfast Service"



Markus Pepping http://is.uni-muenster.de/pi/lehre/ws0506/seminar/01_mda_folien.pdf

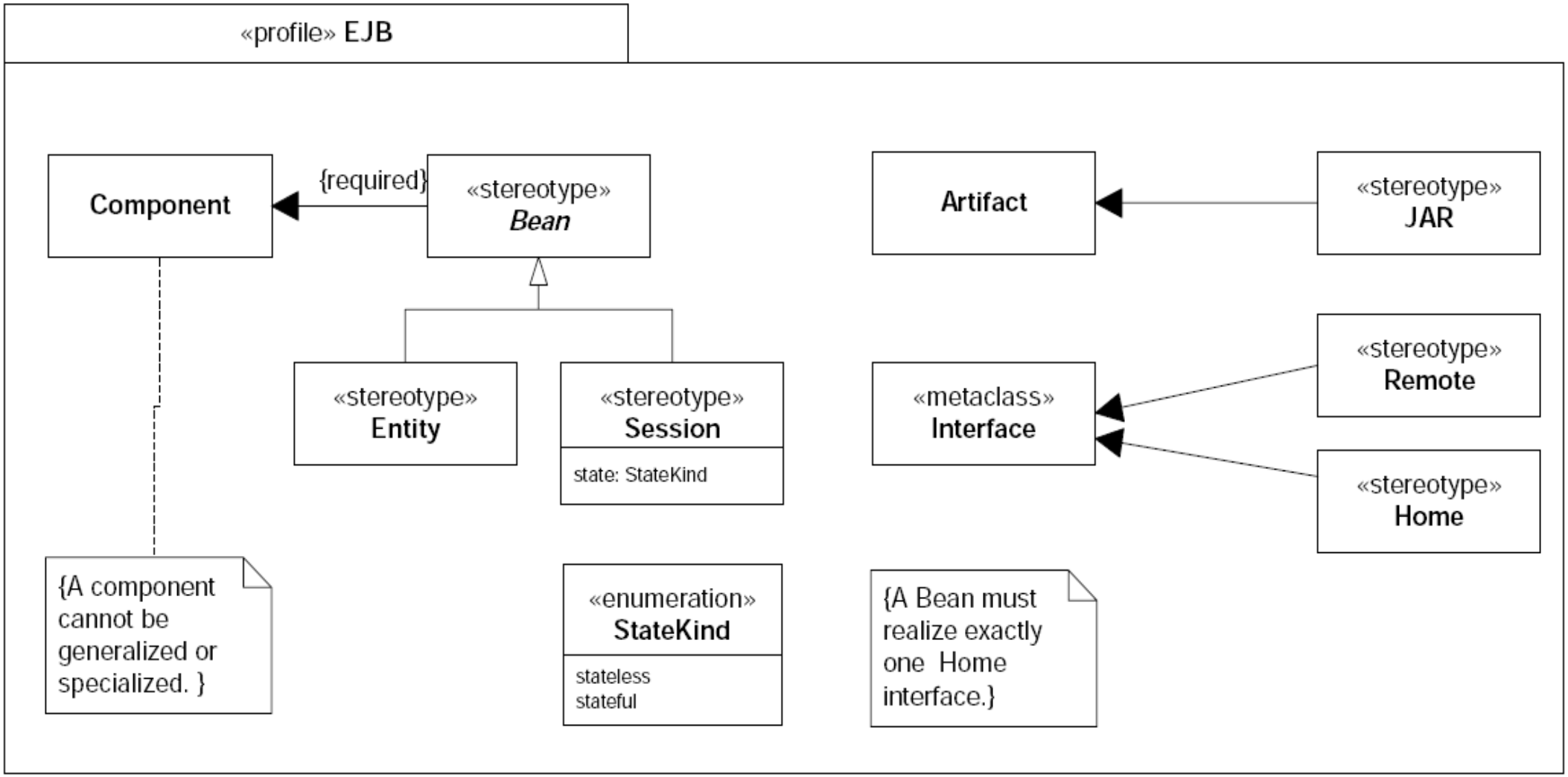
Summary

- MOF: defines structure of a language, e.g. UML or CWM
- EMOF: subset of MOF with simple concepts
- CMOF: extends EMOF with meta modelling capabilities
- Hierarchy: four-layer meta model

Contents

Introduction
Definition and Placement
Unified Modeling Language (UML)
Computer Aided Software Engineering (CASE)
Model Driven Architecture (MDA)
UML Meta Model
Meta Object Facility (MOF)
Profile and Stereotype
Object Constraint Language (OCL)
XML Metadata Interchange (XMI)
Diagram Definition (DD)
Related Specification Languages
Software Pattern
Pattern Catalogue

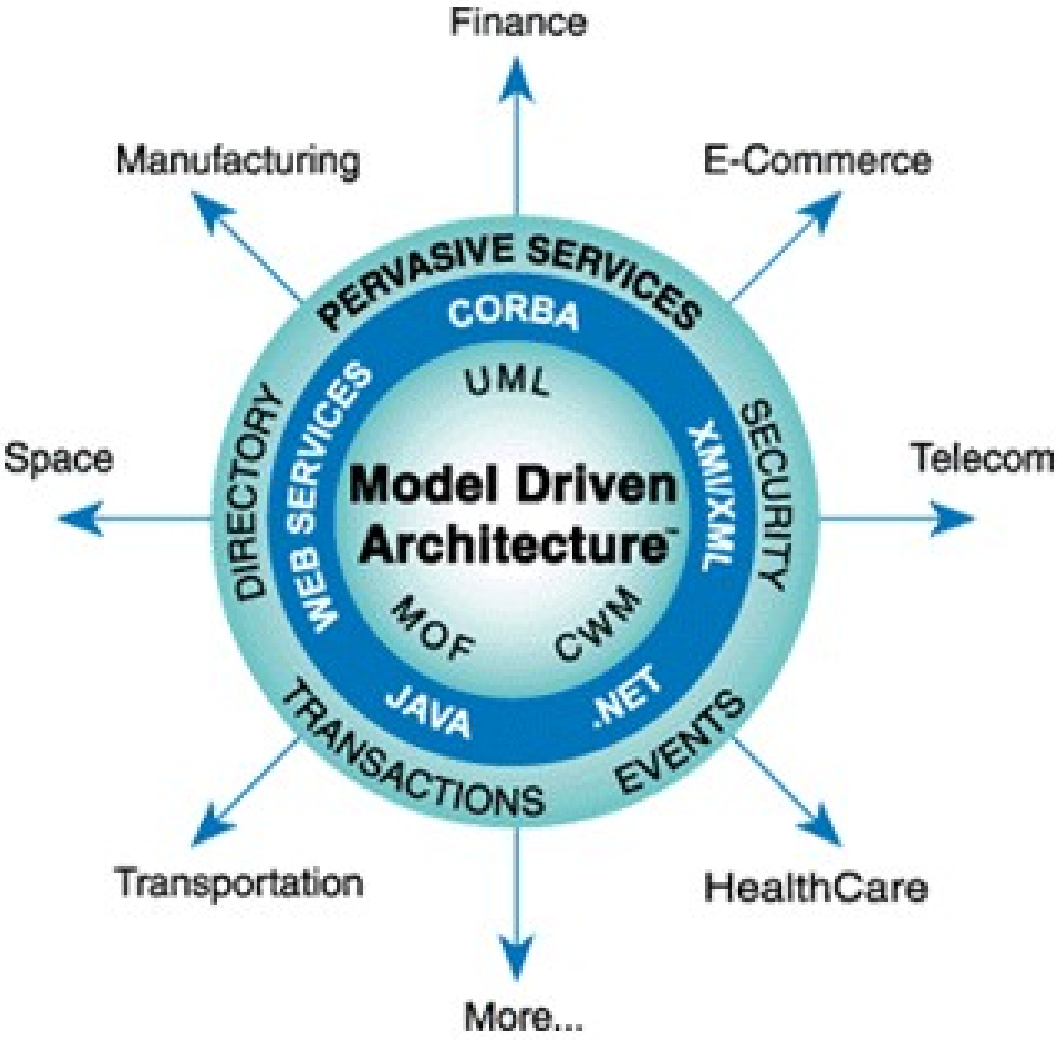
Profile Definition Example: EJB



Object Management Group (OMG). http://www.omg.org/

MDA Overview

various domains ...



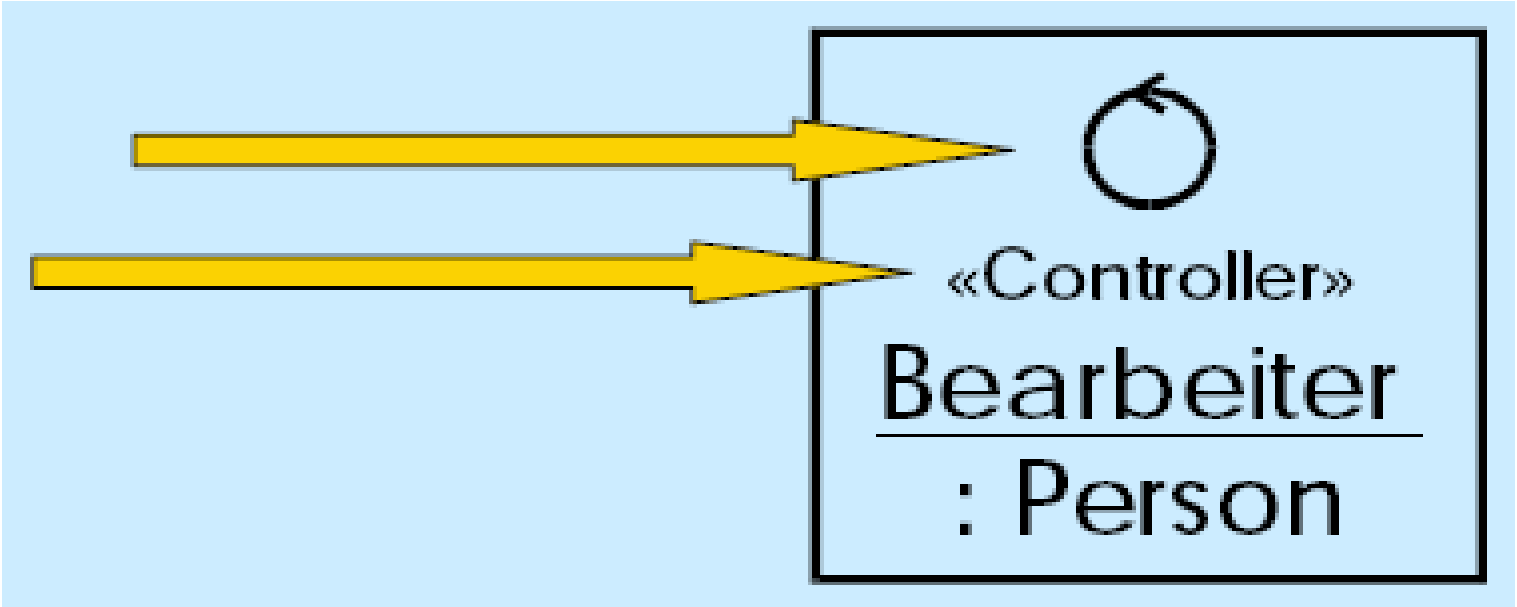
Object Management Group (OMG). http://www.omg.org/

OMG UML Profile Specifications

SPECIFICATION	acronym	topical area / domain	Document #
UML Profile for Advanced and Integrated Telecommunication Services	TelcoML	telecommunications	formal/2013-08-02
SES Management TelcoML Extension	TelcoML-SES	telecommunications	formal/2013-08-03
UML Profile for BPMN Processes	BPMNProfile	domain, modeling	formal/2014-07-01
UML Profile for CORBA	CORP	middleware	formal/2002-04-01
UML Profile for CCM	CCMP	middleware	formal/2005-07-06
UML Profile for CORBA and CCM	CCCMP	middleware	formal/2008-04-07
UML Profile for Enterprise Application Integration	EAI	domain, modeling	formal/2004-03-26
UML Profile for Enterprise Distributed Object Computing	EDOC	domain, modeling	formal/2004-02-01
UML Profile for MARTE: Modeling and Analysis of Real-time and Embedded Systems	MARTE	real-time, middleware	formal/2011-06-02
UML Profile for QoS and Fault Tolerance	QFTP	real-time, middleware, modeling	formal/2008-04-05
UML Profile for Schedulability, Performance and Time	SPTP	real-time, middleware, modeling	formal/2005-01-02
UML Profile for System on a Chip	SoCP	real-time, middleware, modeling	formal/2006-08-01
UML Profile for Software Radio (aka PIM & PSM for Software Radio Components)	SDRP	software-based communications	formal/2007-03-01
UML Profile for Voice	VOICP	telecommunications	formal/2008-04-06
UML Testing Profile	UTP	modeling	formal/2013-04-03

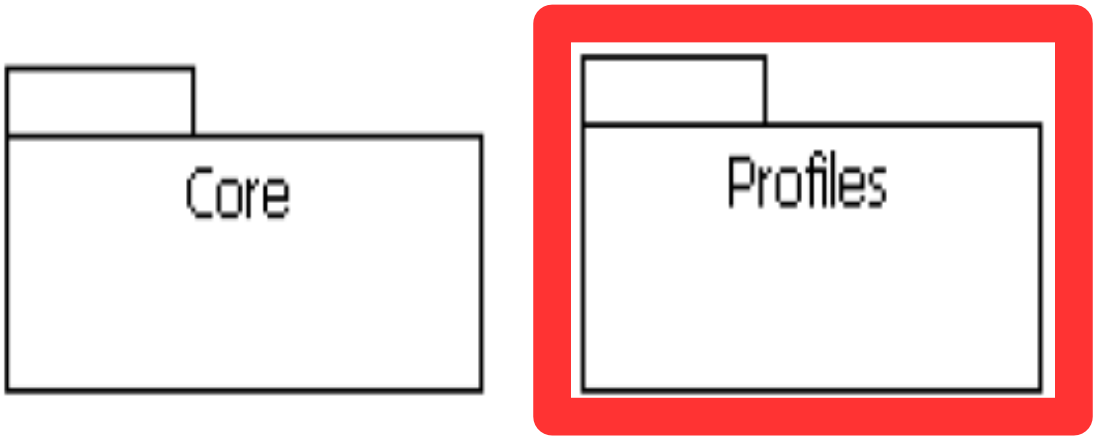
http://www.omg.org/spec/#Profile

UML Profile



Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

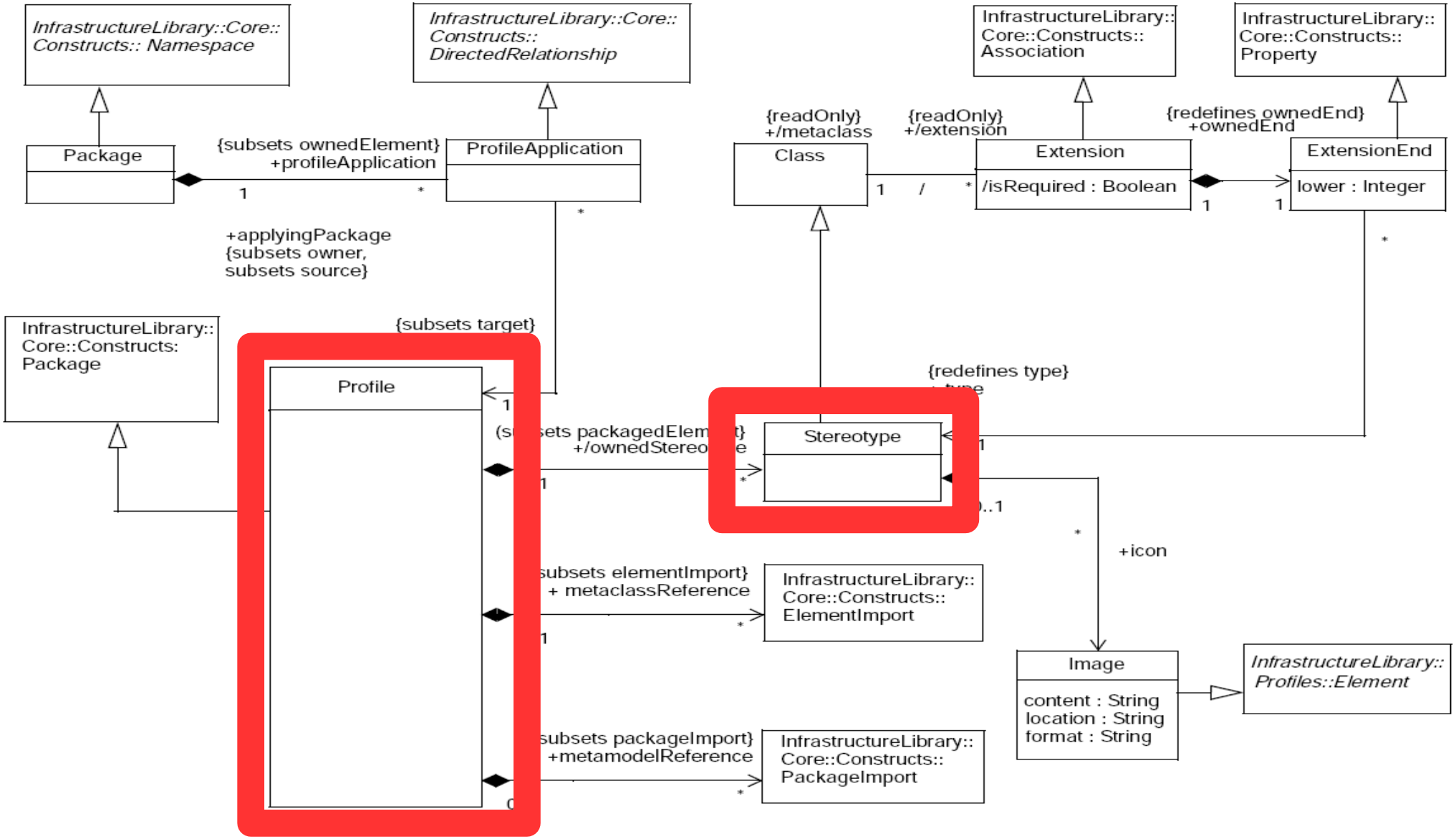
UML Infrastructure Library



special UML language unit

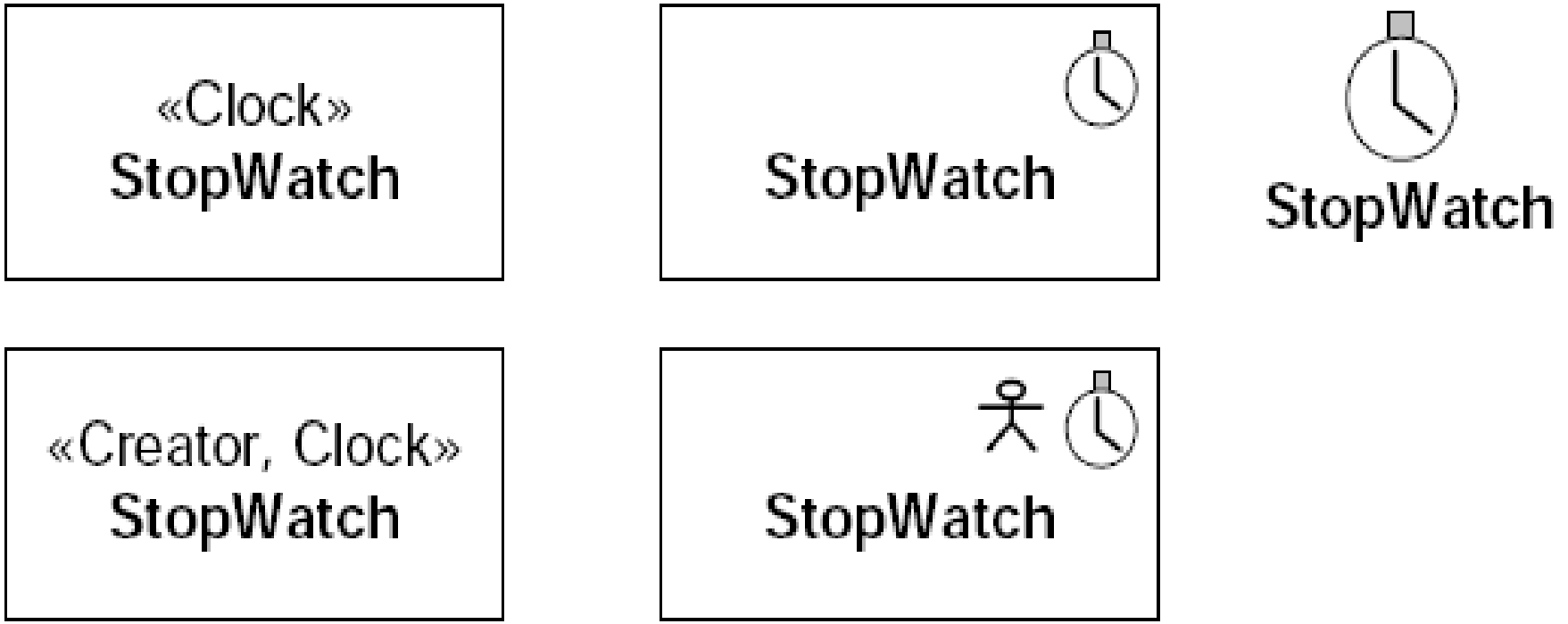
Object Management Group (OMG). http://www.omg.org/

Classes of Infrastructure Library Package "Profiles"

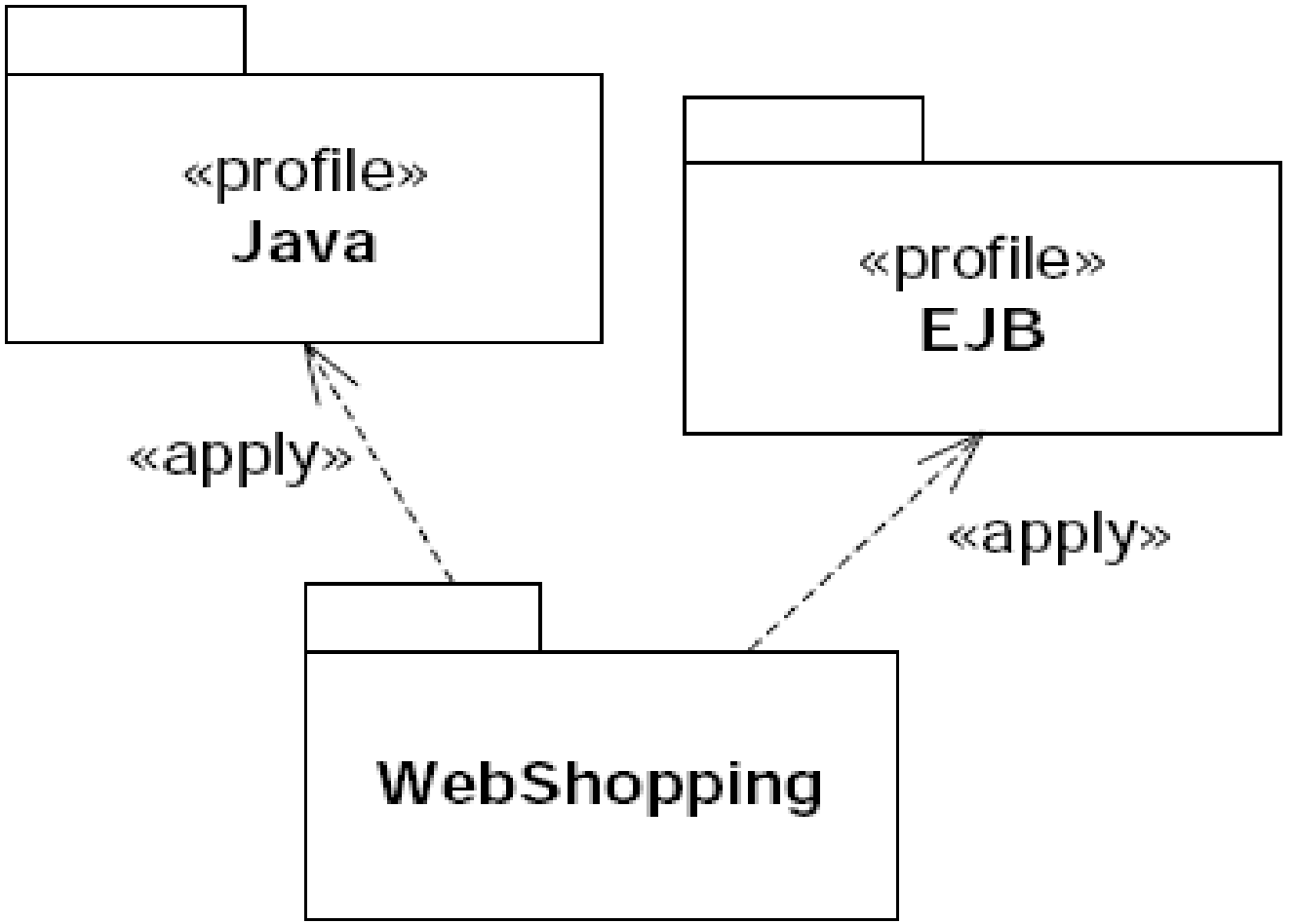


Object Management Group (OMG). http://www.omg.org/

Stereotype

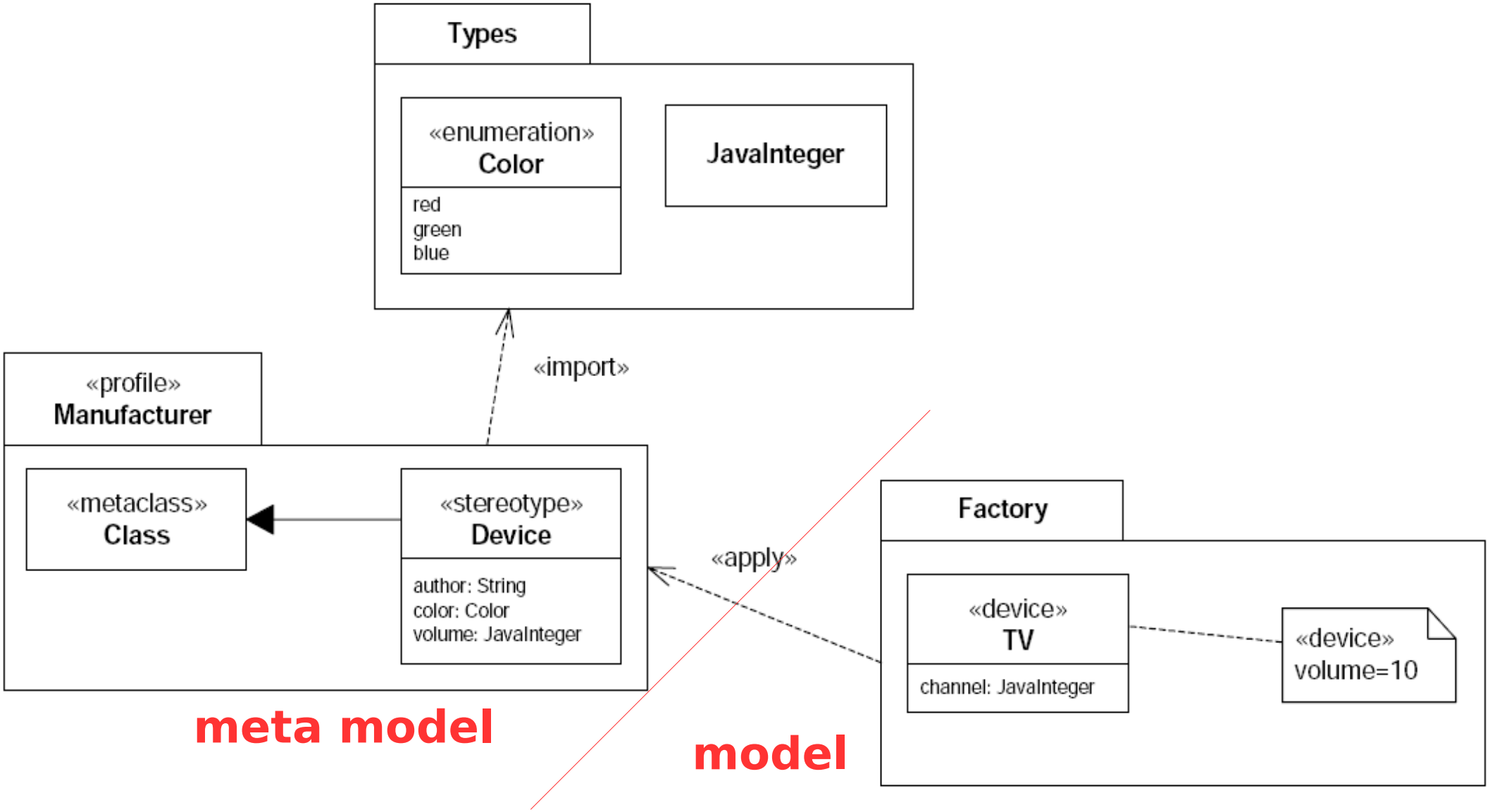

«Clock»
StopWatch
StopWatch
StopWatch
«Creator, Clock»
StopWatch
StopWatch

Profile Application Example: EJB



Kirill Fakhroutdinov. UML. 2011-08-22 (Zugriff). http://www.uml-diagrams.org/

Profile Application Example: Details



Kirill Fakhroutdinov. UML. 2011-08-22 (Zugriff). http://www.uml-diagrams.org/

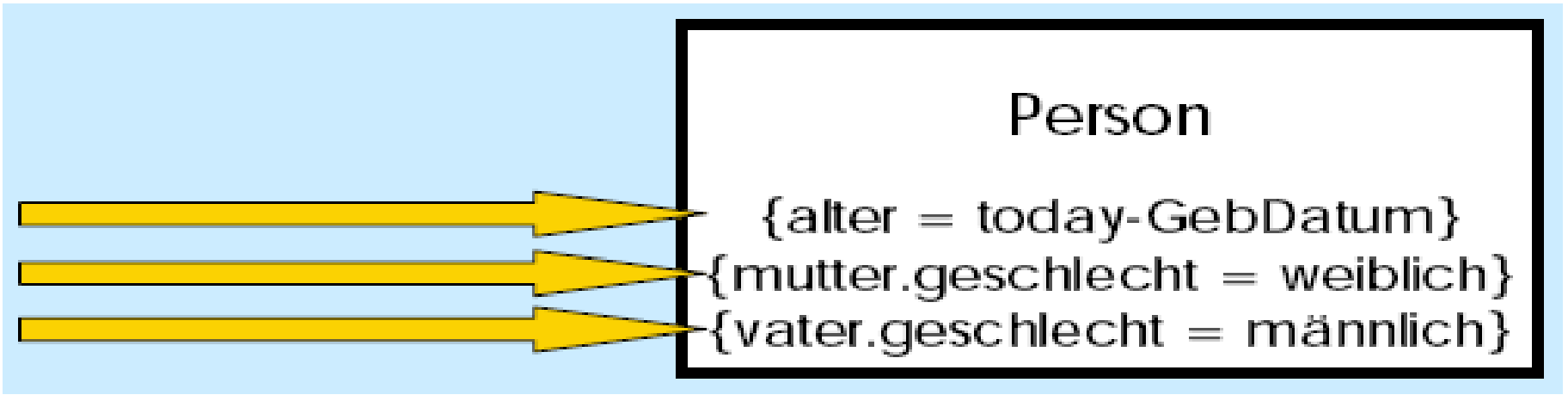
Summary

- Profile: generic extension mechanism to UML meta model
- Stereotype: additional property given as image or text

Contents

Introduction
Definition and Placement
Unified Modeling Language (UML)
Computer Aided Software Engineering (CASE)
Model Driven Architecture (MDA)
UML Meta Model
Meta Object Facility (MOF)
Profile and Stereotype
Object Constraint Language (OCL)
XML Metadata Interchange (XMI)
Diagram Definition (DD)
Related Specification Languages
Software Pattern
Pattern Catalogue

Object Constraint Language (OCL)

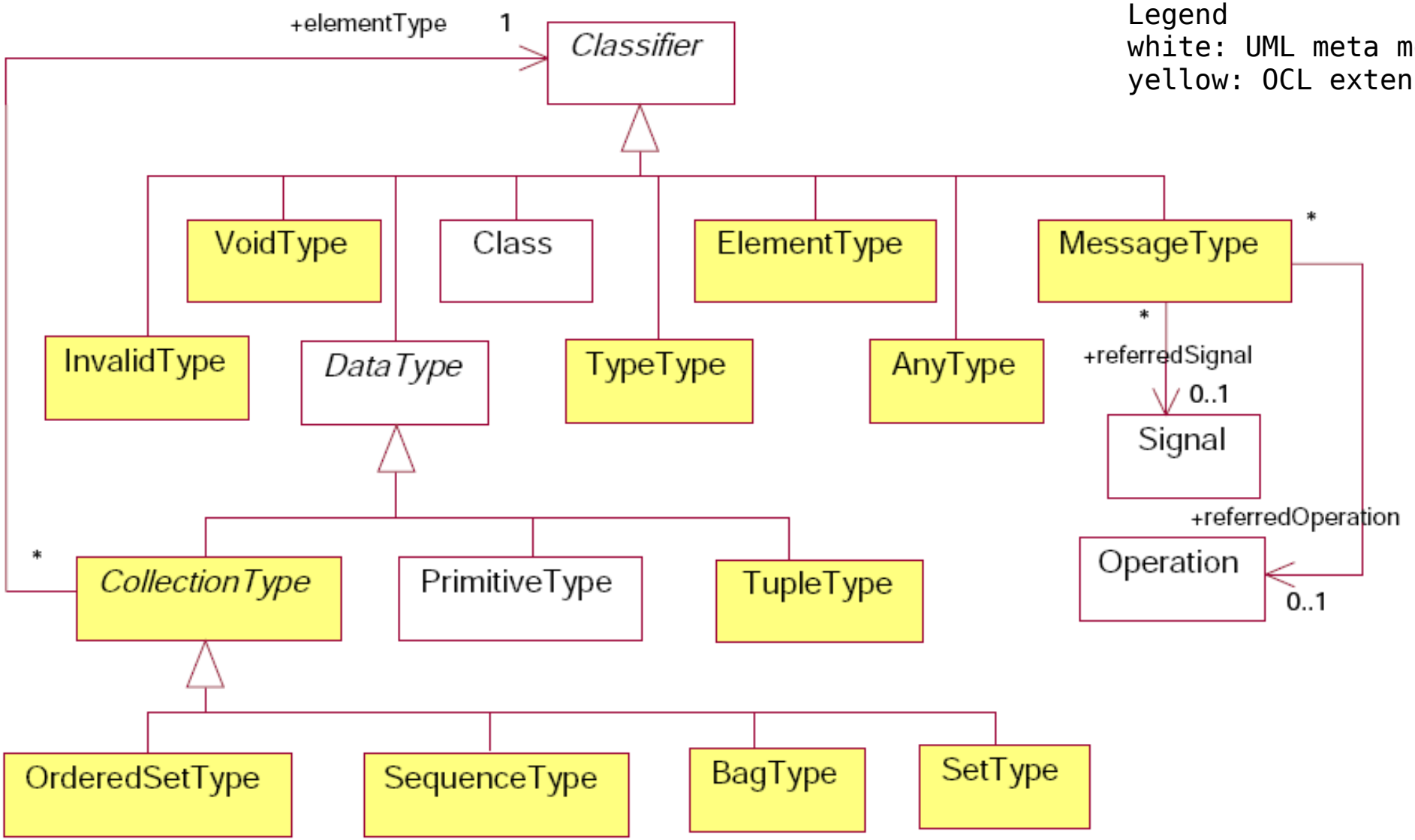


Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML.
Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

Application

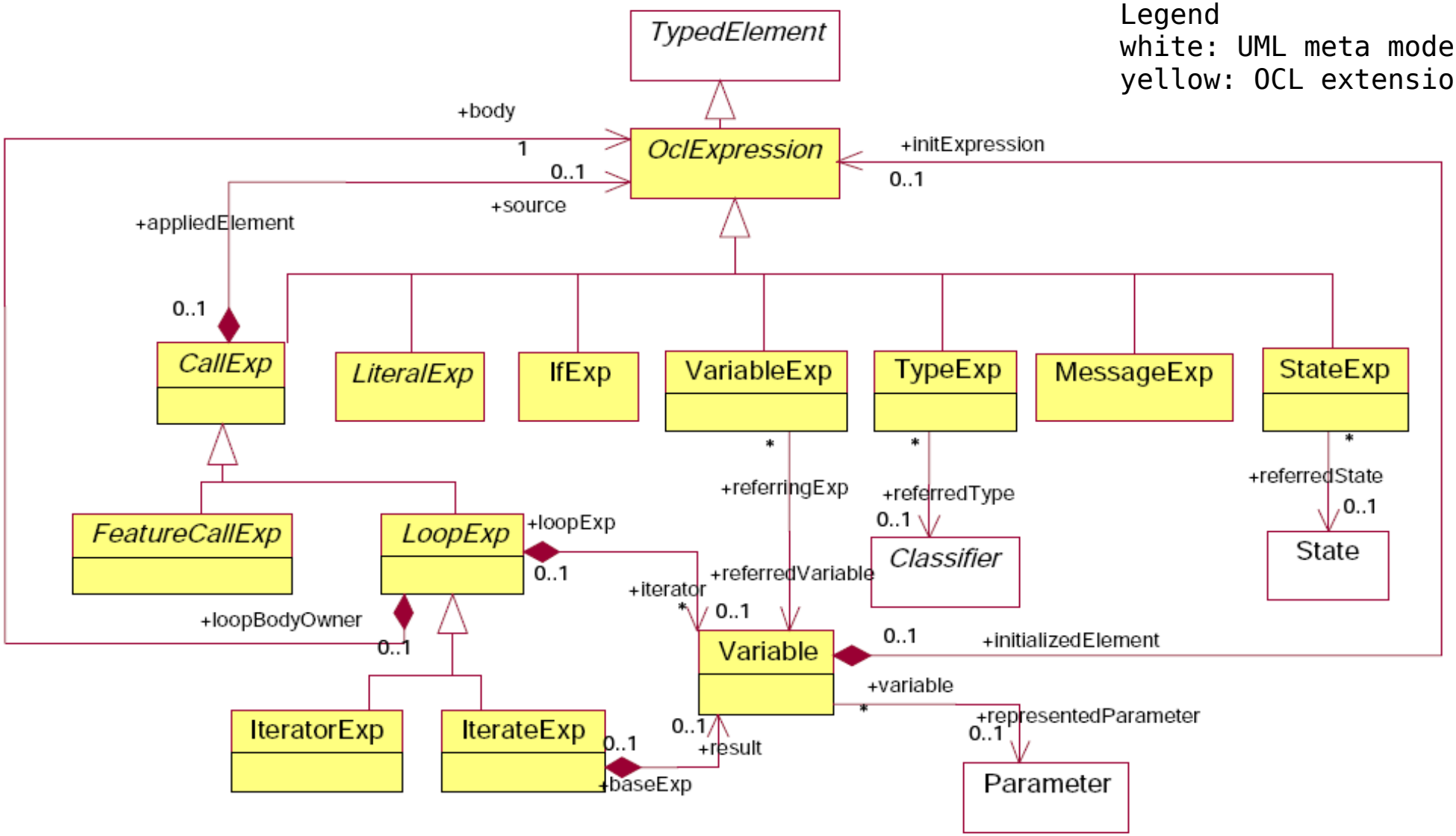
- invariants (conditions that have to be true) on classifiers
- pre- and post conditions on operations/ methods
- initial and derived values
- definition of attributes/operations that aren't in the model
- guards have to be valid on state changes

Meta Model for Types



Object Management Group (OMG). http://www.omg.org/

Meta Model for Expressions



Object Management Group (OMG). http://www.omg.org/

Keywords

- context: specifies about which elements we are talking
- self: current object
- result: return value

OCL can talk about collections (here: Sets)

- operations on collections: –>
- example operations: select, forAll, iterate
- iterate: can simulate all other operations on collections

Constraints Exercise

1. context Person inv: self.Alter >=0
2. context Person inv: self.Eltern->forAll(e|e.Alter>self.Alter)
3. context Person::hatGeburtstag() post: self.Alter=self.Alter@pre+1
4. context Person inv: self.Eltern->size()=2
5. context Person::bekommtKind() post: self.Kinder->notEmpty() and
   self.Kinder->size() > self.Kinder@PRE->size()
6. context Person inv: self.Alter<18 implies autos->size()=0
7. context Auto inv: self.Erstzulassung>=self.Baujahr
8. context Person inv: self.autos->size()>0 implies
   self.autos->exists(a | Calendar.YEAR - a.baujahr < self.alter))
9. context Person inv: not self.eltern->includes(self)
10. context Person inv: Person.allInstances->exists(p | p.autos->size() > 0)

Class Diagram Example

```
-- This is a comment
-- Example: Constraining a classifier

-- The current classifier is referenced with "self"
context Kurs
self.dozent.honorar > 0.00
-- Use arrow instead of dot for sets/containers
context Schedule
self.Kurs->notEmpty()
-- Assure that all courses have a room
context Schedule
self.Kurs->forAll(raumZugewiesen <> 'Kein Raum!')
-- The context may be given explicitly
context Student
inv: self.Notendurchschnitt > 2.0
-- The classifier may be given a name
context s : Student
inv: s.Notendurchschnitt > 2.0
-- A label may be given for an expression
-- It has no function in OCL
context s : Student
inv mindestNotendurchschnittRegel: s.Notendurchschnitt > 2.0
```

Constraining an Operation

```
-- Assure that each student having subscribed has payed his fees.
context Kurs::studentenAnmelden(s : Student) : boolean
pre: s.studiengebuehrGezahlt = true

-- The keyword "result" may be used for postconditions.
context Kurs::studentenAnmelden(s : Student) : boolean
pre: s.studiengebuehrGezahlt = true
post: result = true

-- A label may be assigned to pre- and postconditions.
context Kurs::studentenAnmelden(s : Student) : boolean
pre hatStudiengebuehrGezahlt: s.studiengebuehrGezahlt = true
post studentWurdeAngemeldet: result = true

-- The "@pre" keyword may be used in postconditions to access the value of
-- an element before an operation is executed.
context Kurs::studentenAnmelden(s : Student) : boolean
pre: s.studiengebuehrGezahlt = true
post: result = true and self.studenten = self.studenten@pre + 1

-- The keyword "body" may be used to access the results of a query.
-- The example selects all top students whose marks are greater than 2.5.
context Kurs::getTopStudenten() : Student
body: self.studenten->select(Notendurchschnitt > 2.5)
```

Constraining an Attribute

```
-- The initial and following values of attributes may be given.
context Schule::studiengebuehr : float
init: 2500.00

-- The keyword "derive" may be used to access an attribute value following the initial.
context Schule::studiengebuehr : float
derive: tuition * 10%
```

Advanced Expressions

```
-- The following example uses a condition.
context Student inv:
if studiengebuehrGezahlt = true then
    examensjahr = 2005
else
    examensjahr = 0000
endif

-- The following example shows variable declaration.
context Kurs inv:
let honorar : float = self.dozent.honorar in
if self.Kursniveau > 4000 then
    honorar > 80000.00
else
    honorar < 80000.00
endif

-- The following example tests if Dozent is a type or subtype of Lehrer.
context Kurs
inv: self.dozent.oclIsKindOf(Lehrer)

-- The following example finds top students within the collection "studenten".
context Kurs::getTopStudenten(): Student
body: self.studenten->select(Notendurchschnitt > 2.5)
```

Summary

- formal specification language describing rules
- makes UML diagrams more precise
- expressions use vocabulary of UML class diagram
- invariants of objects; pre-/ post conditions of operations
- assignment in curly braces, to basic UML elements

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

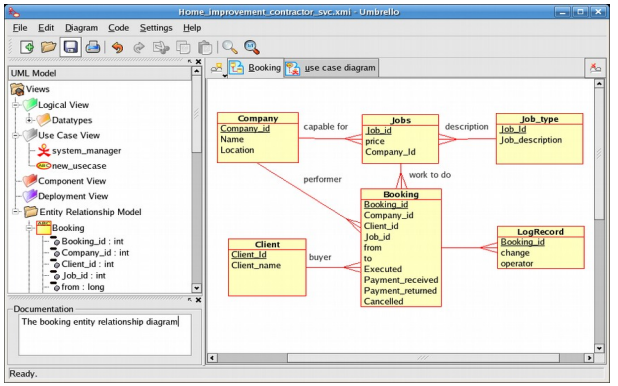
Diagram Definition (DD)

Related Specification Languages

Software Pattern

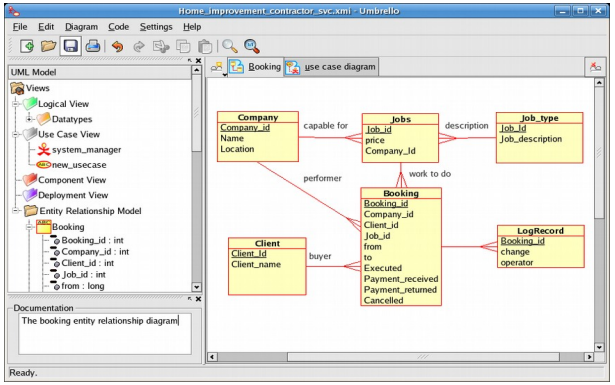
Pattern Catalogue

XMI Example



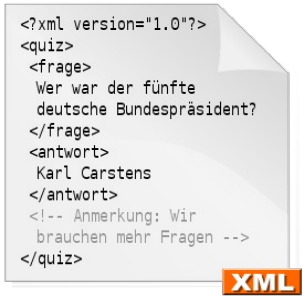
CASE tool 2

XMI file



CASE tool 1

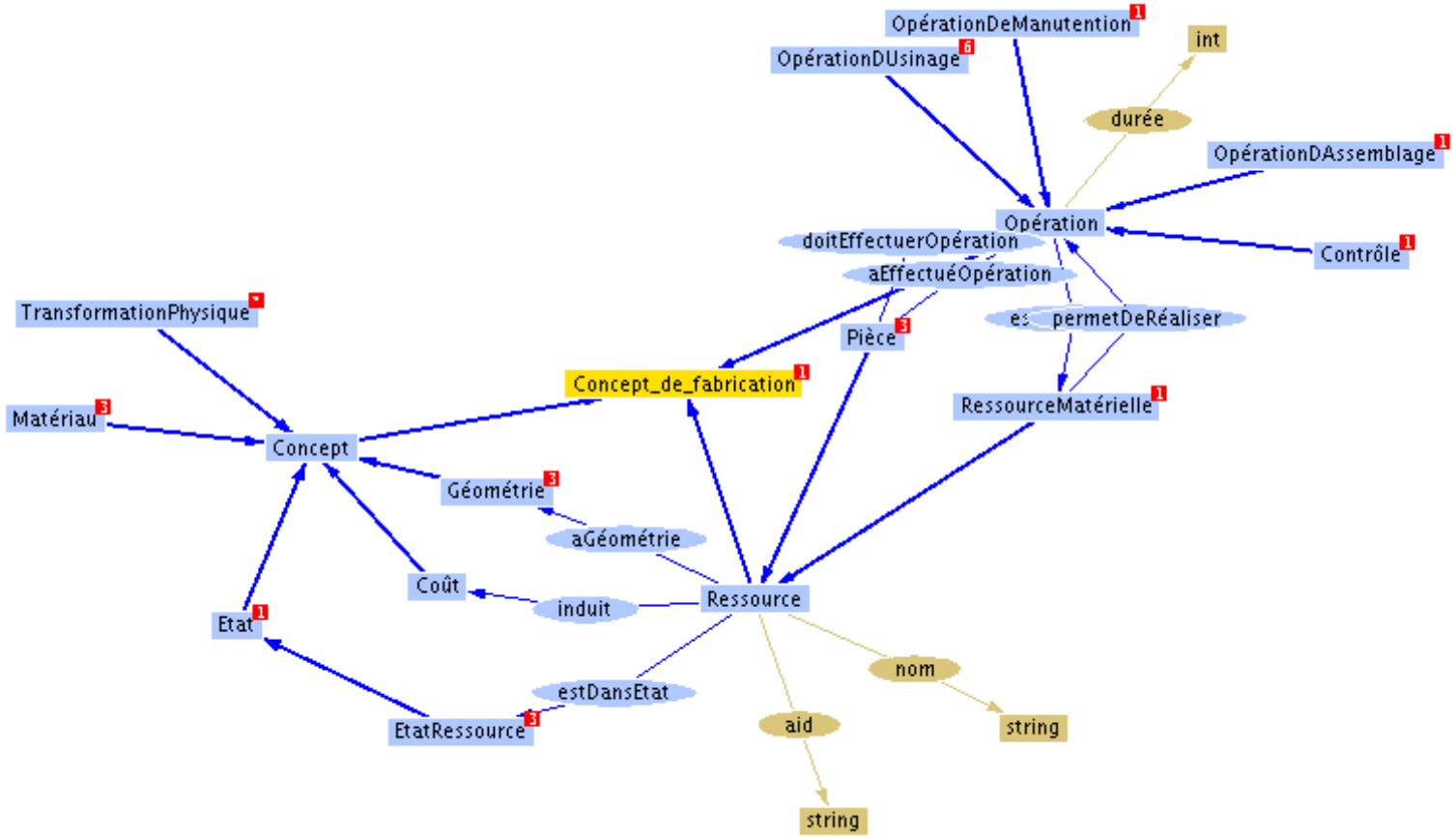
Standards for Representing Metadata



- Document Type Definition (DTD)
- XML Schema Definition (XSD)
- Resource Description Framework (RDF)
- Web Ontology Language (OWL)

- Human-Usable Textual Notation (HUTN)
- XML Metadata Interchange (XMI), also known as MOF to XML Mapping

Meta Model and Ontology



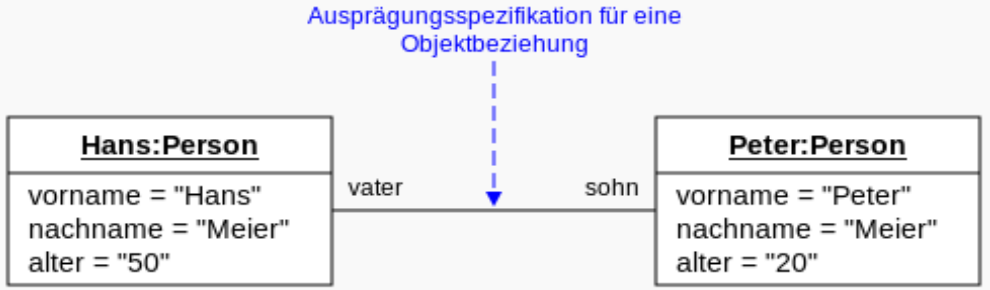
https://en.wikipedia.org/wiki/Metamodeling

```xml
<?xml version = "1.0"?>
<XMI>
 <XMI.header>
  <XMI.model xmi.name = "PersonenPackage" xmi.version = "1.1"/>
 </XMI.header>
 <XMI.content>
  <PersonenPackage xmi.id="xmi-id-001">
   <PersonenPackage.Person xmi.id="Hans">
    <PersonenPackage.Person.Vorname>
     Hans
    </PersonenPackage.Person.Vorname>
    <PersonenPackage.Person.Name>
     Meier
    </PersonenPackage.Person.Name>
    <PersonenPackage.Person.Alter>
     50
    </PersonenPackage.Person.Alter>
    <PersonenPackage.Person.Sohn>
     <PersonenPackage.Person xmi.idref="Peter">
    </PersonenPackage.Person.Sohn>
   </PersonenPackage.Person>
   <PersonenPackage.Person xmi.id="Peter">
    <PersonenPackage.Person.Vorname>
     Peter
    </PersonenPackage.Person.Vorname>
    <PersonenPackage.Person.Name>
     Meier
    </PersonenPackage.Person.Name>
    <PersonenPackage.Person.Alter>
     20
    </PersonenPackage.Person.Alter>
    <PersonenPackage.Person.Vater>
     <PersonenPackage.Person xmi.idref="Hans">
    </PersonenPackage.Person.Vater>
   </PersonenPackage.Person>
  </PersonenPackage>
 </XMI.content>
</XMI>
```

HUTN

```
PersonenPackage “id-001” {
 Person “Hans” {
  Vorname: “Hans”
  Nachname: “Meier”
  Alter: 50
  Sohn: Person “Peter”
 }
 Person “Peter” {
  Vorname: “Peter”
  Nachname: “Meier”
  Alter: 20
  Vater: Person “Hans”
 }
}
```

UML



https://de.wikipedia.org/wiki/Human-Usable_Textual_Notation

Summary

- Metamodel: definition and placement
- XMI: interchange of UML models between CASE tools
- Technology: serialised data in XML format
- Problem: missing graphical information for diagrams
- HUTN: human-readable text-based meta model description

Contents

Introduction
Definition and Placement
Unified Modeling Language (UML)
Computer Aided Software Engineering (CASE)
Model Driven Architecture (MDA)
UML Meta Model
Meta Object Facility (MOF)
Profile and Stereotype
Object Constraint Language (OCL)
XML Metadata Interchange (XMI)
Diagram Definition (DD)
Related Specification Languages
Software Pattern
Pattern Catalogue

SVG UML Class Example

```
<!--Class Position-->
<g style="font-size:9;font-family:dialog;" onmousemove="removePopup()"
    onclick="class_click(evt,'Class Position')"
    <g style="fill:none;stroke:black;stroke-width:1">
        <rect style="fill:white" width="88" height="61"/>
        <line style="fill:none;stroke:black;stroke-width:1" x2="88" y2="20" y1="20"/>
        <line style="fill:none;stroke:black;stroke-width:1" x2="88" y2="41" y1="41"/>
    </g>
    <text style="text-anchor:middle" dx="88" dy="9">Position</text>
    <!-- attribute compartment -->
    <g transform="translate(0,20)">
        <text dx="2" dy="9">#<tspan x="12">posnum</tspan> : int</text>
    </g>
    <!-- operation compartment -->
    <g transform="translate(0,41)">
        <text class="standardfont" dx="2" dy="9">
            +
            <tspan x="12">getPosnum</tspan>
            ()
        </text>
    </g>
</g>
```

Only graphical information gets stored (e.g. two black lines); no semantics (e.g. name, type)

XSLT UML Class Example

```
    <rect height="{$height}" width="{$width}" style="fill:white"/>
    <xsl:if test="$draw_attributes">
        <line y1="{$Attrib_PartStart}"y2="{$Attrib_PartStart}" x2="{$width}"
            style="fill:none;stroke:black;strokewidth:1"/>
    </xsl:if>

        <line y1="{$Operation_PartStart}" y2="{$Operation_PartStart}" x2="{$width}"
            style="fill:none;stroke:black;stroke-width:1"/>
    </xsl:if>
</g>
```

Placeholders get replaced, e.g.:
y1="{$Attrib_PartStart}

Compliance Levels

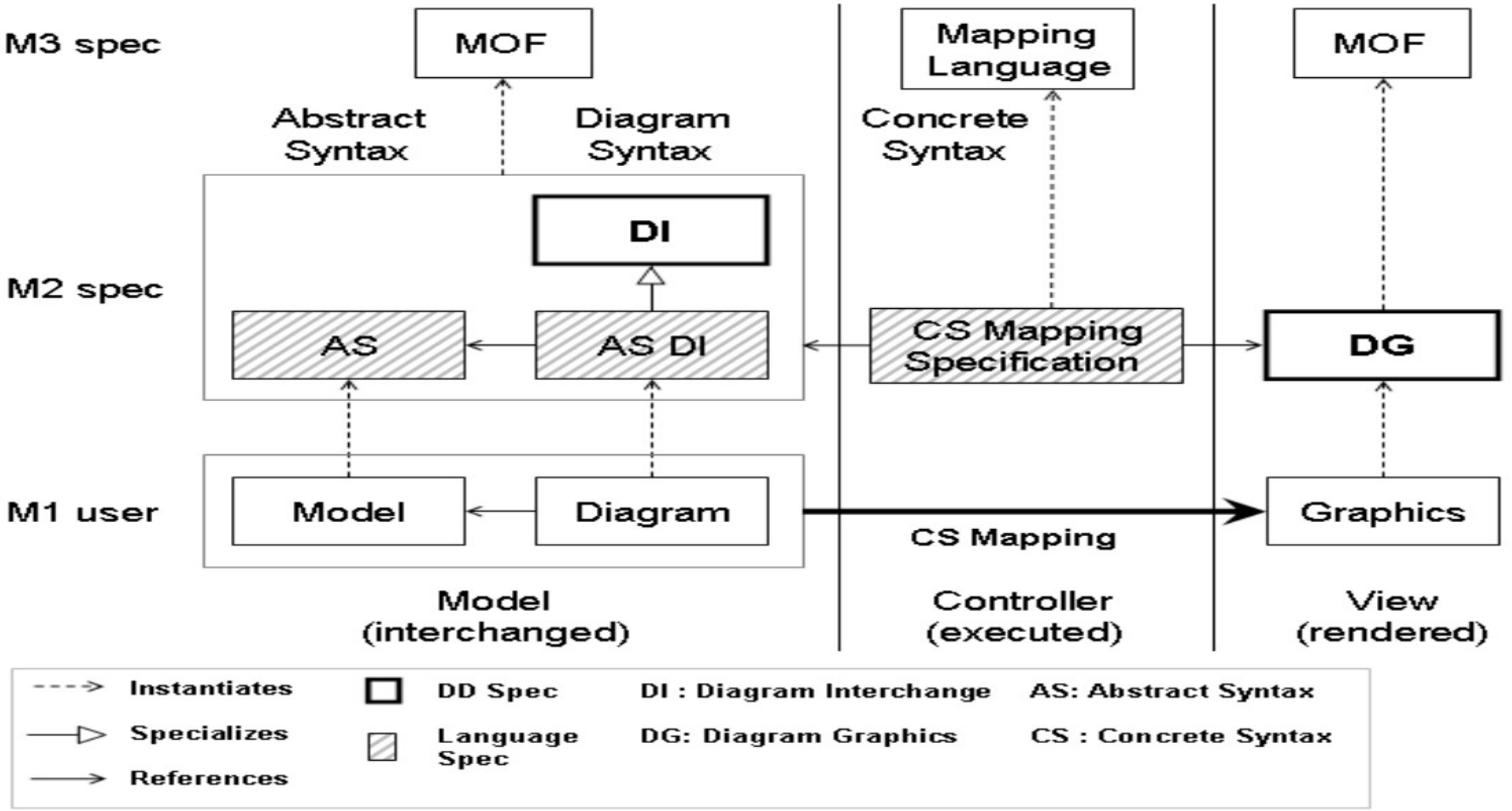
- XMI[UML]
- XMI[DI]
- XMI[UML+DI]

Two Kinds of Graphical Information

- Diagram Interchange (DI)
- Diagram Graphics (DG)

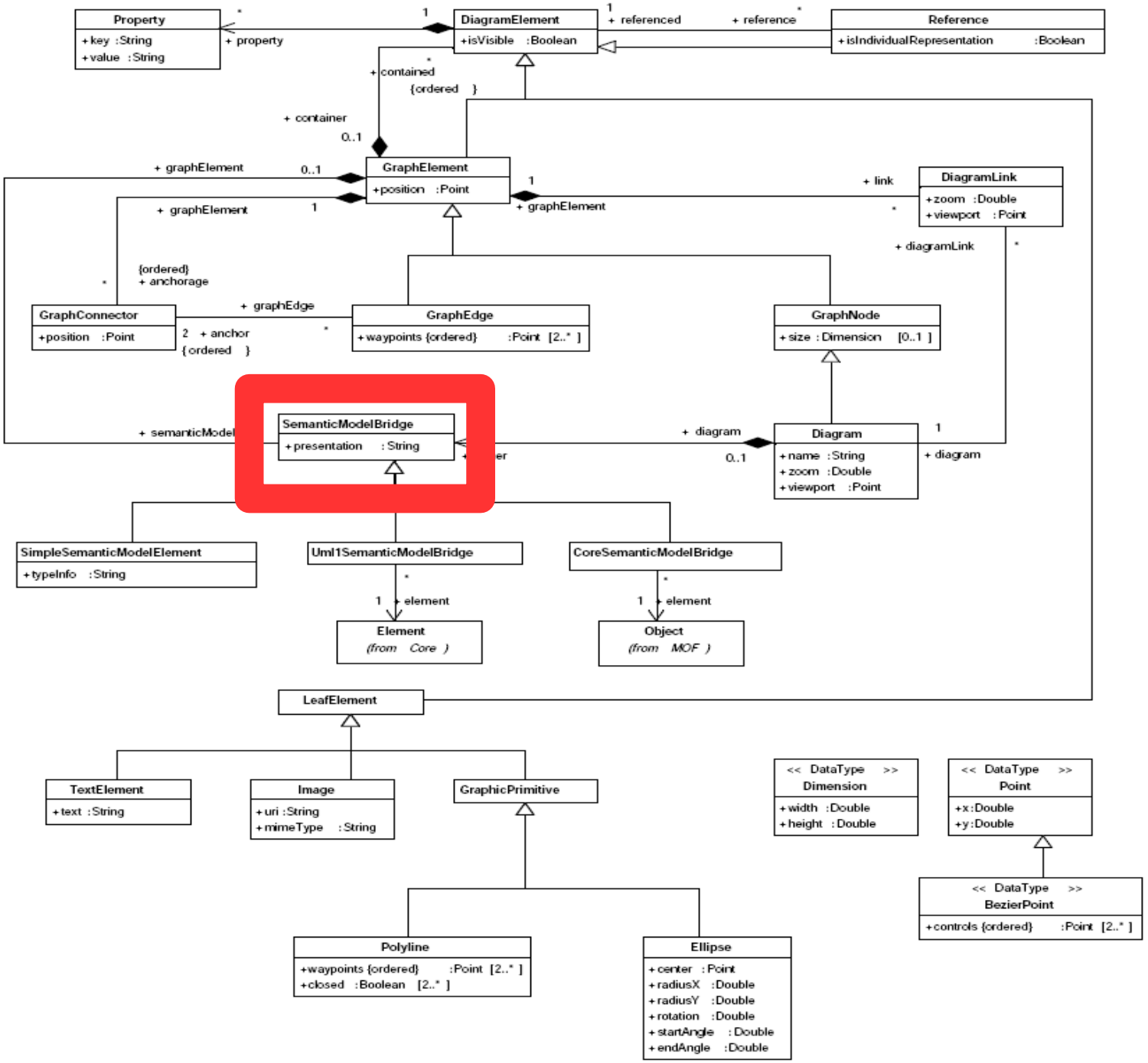
Diagram Definition (DD) 1.1
replaced Diagram Interchange (DI) 1.0

Diagram Definition Architecture



OMG: Diagram Definition (DD) Specification. Version 1.1. 2015-06-01 http://www.omg.org/

Meta Model

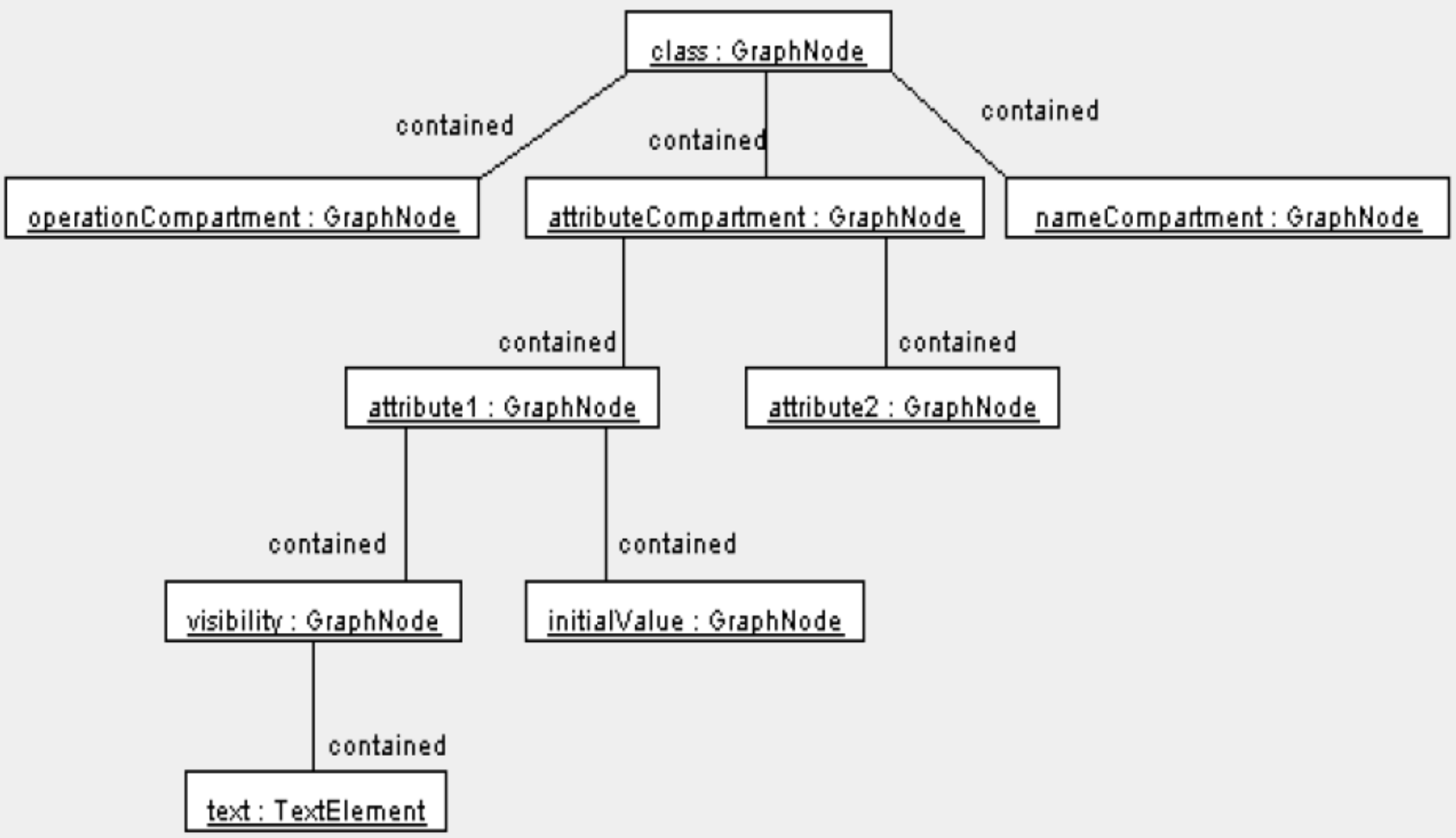


http://www.omg.org/

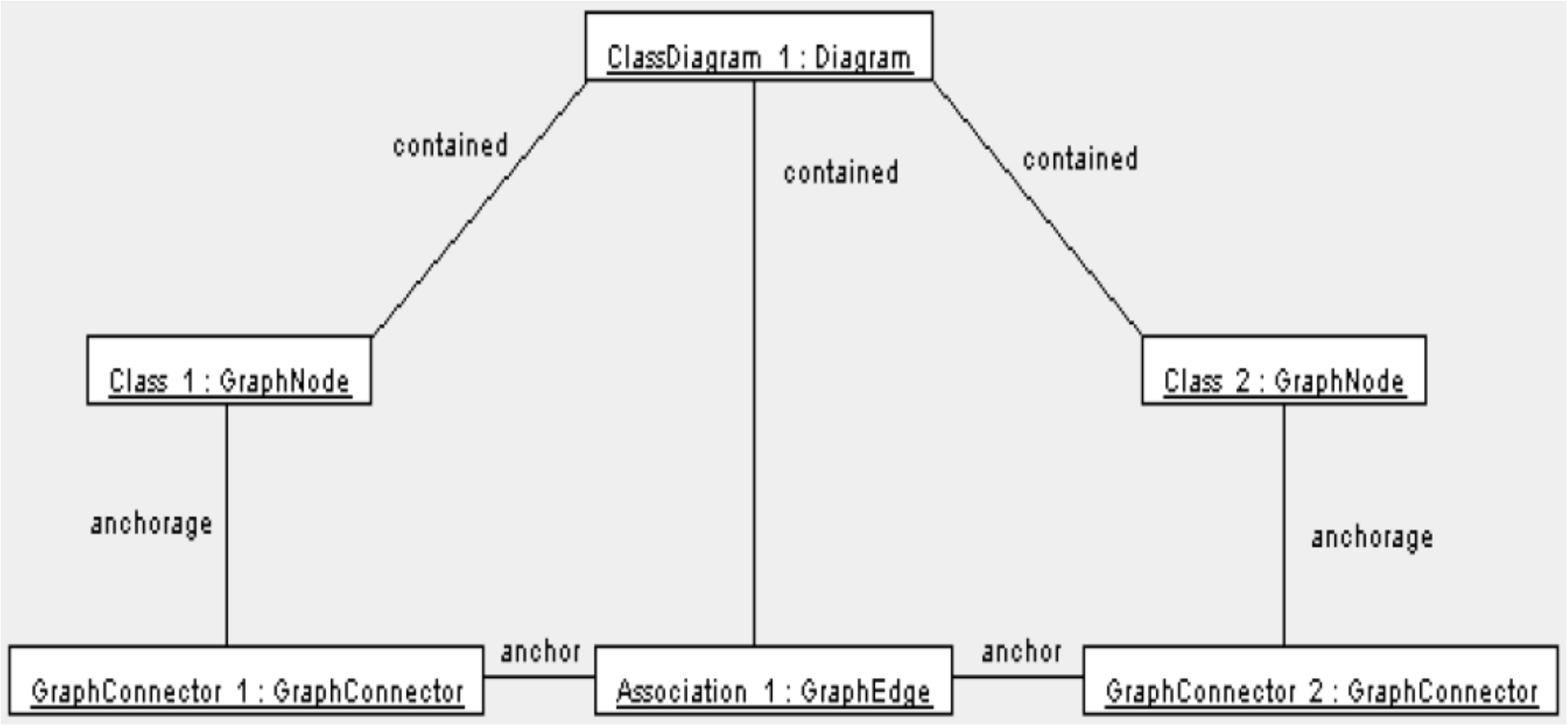
Graphical Representation Alternatives

- Special classes for each and every kind of shape that UML diagrams consist of
- Graph schema known from graph theory

OD representing a Class as Nested Graph Elements



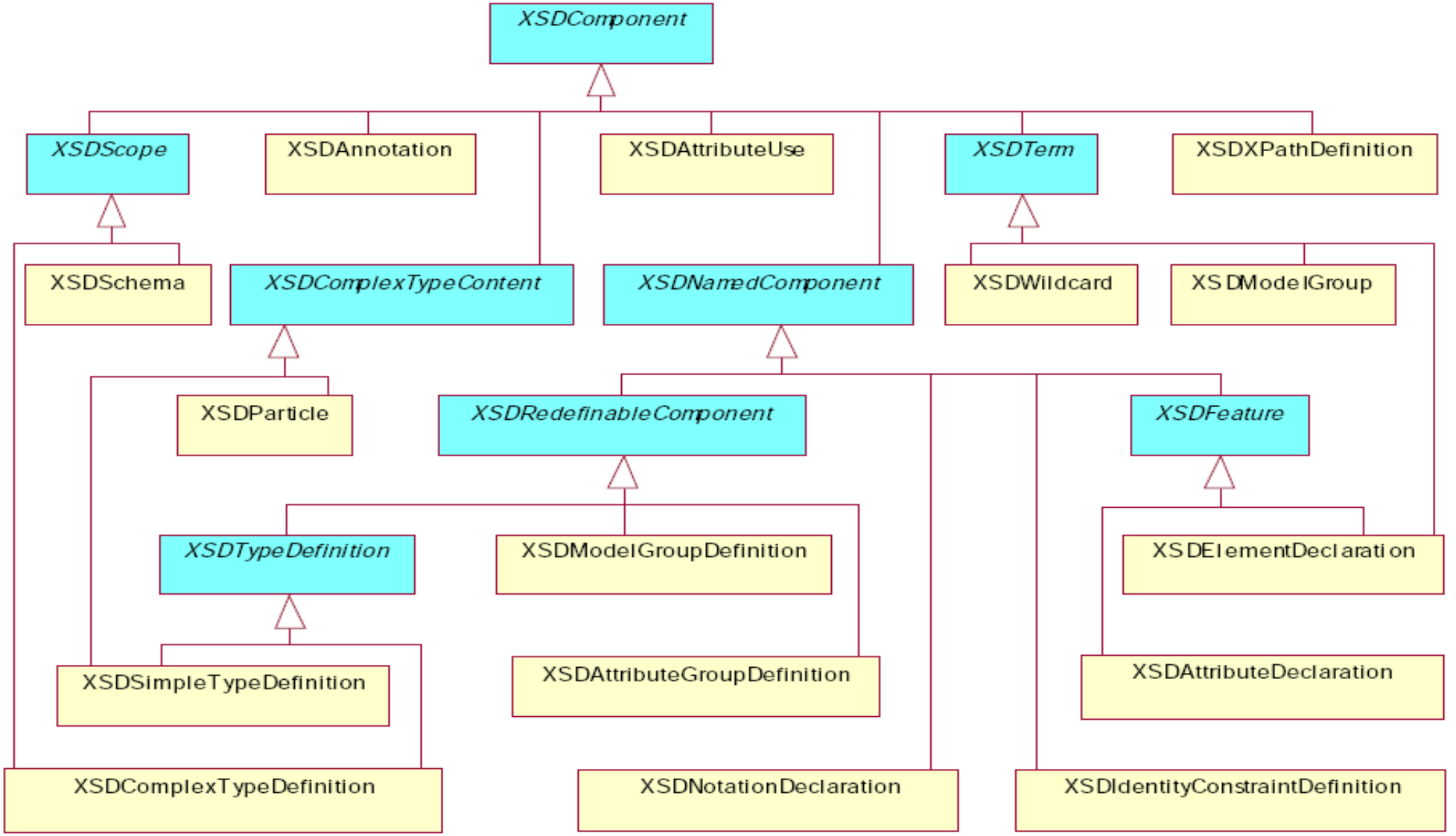
OD representing a Class Diagram with Nodes/ Edge



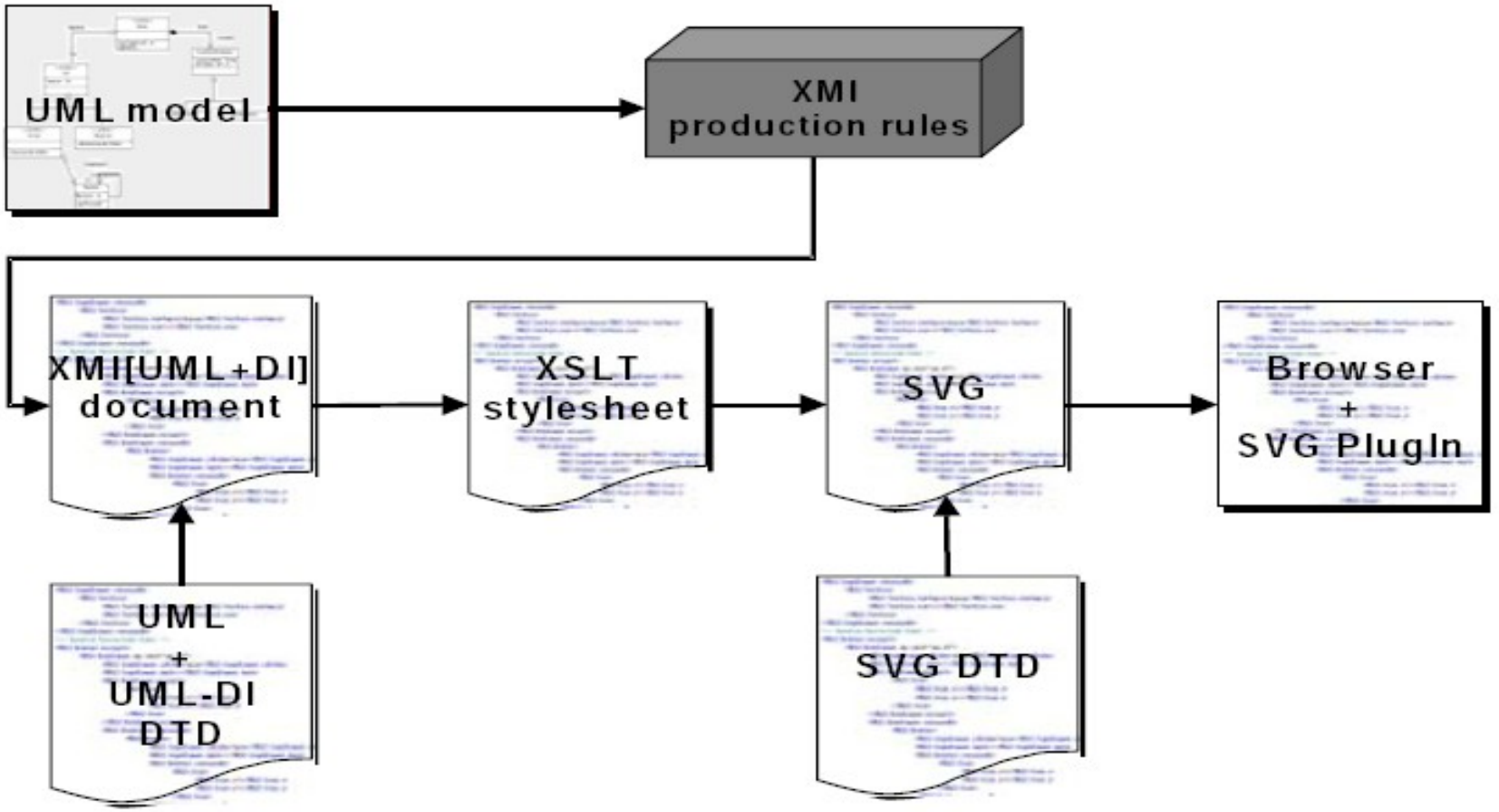
XMI Production Rules exist for ...

- MOF-based metamodel → XML schema
- Model based on MOF 2.0 Core → XML document
- XML input sources → object definitions in MOF
  - DTD → MOF
  - XML → MOF
  - XML Schema → MOF

XML Schema Infoset Model



Scalable Vector Graphics (SVG) Creation Example



Tools supporting XMI

- ARIS Toolset
- Astah
- MagicDraw
- Rational System Architect
- Sparx Enterprise Architect
- MID Innovator
- Visual Paradigm
- ArgoUML
- BOUML
- Modelio
- Umbrello
- StarUML

Summary

- DD: extension of XMI providing graphical diagram data
- XMI[UML+DD]: combination of classical XMI and DD
- Technology: schema as known from graph theory

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

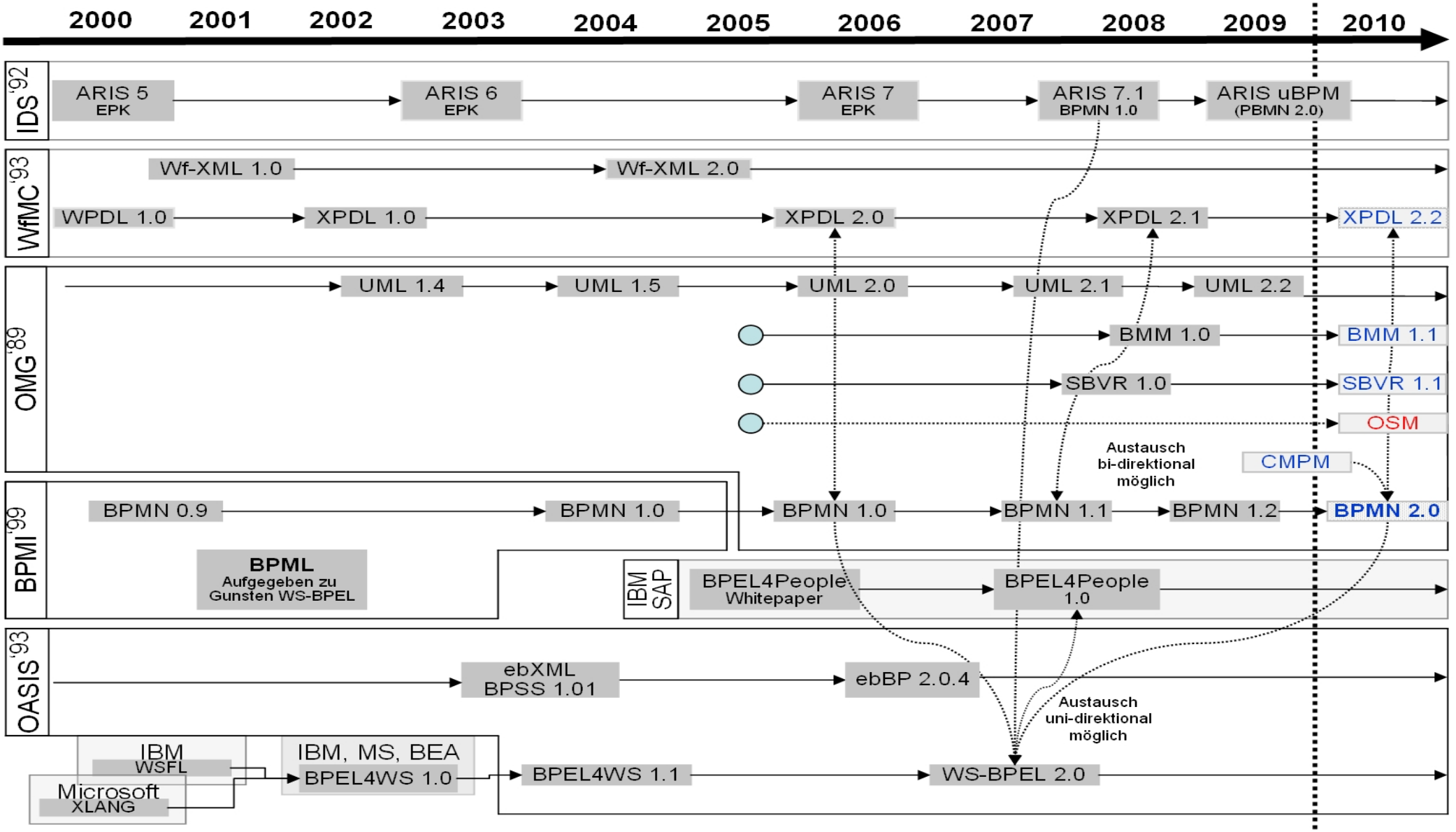
Diagram Definition (DD)

Related Specification Languages

Software Pattern

Pattern Catalogue

Specification Languages and Notations



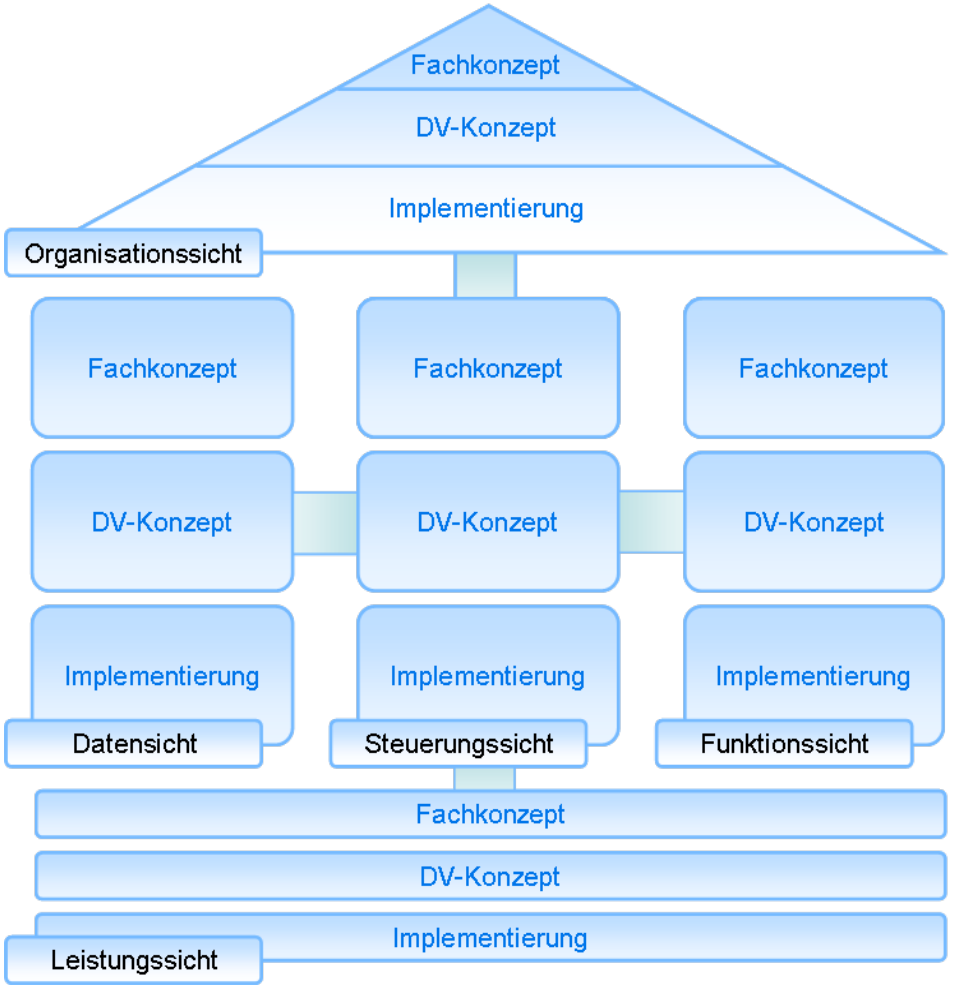
https://de.wikipedia.org/wiki/Business_Process_Model_and_Notation

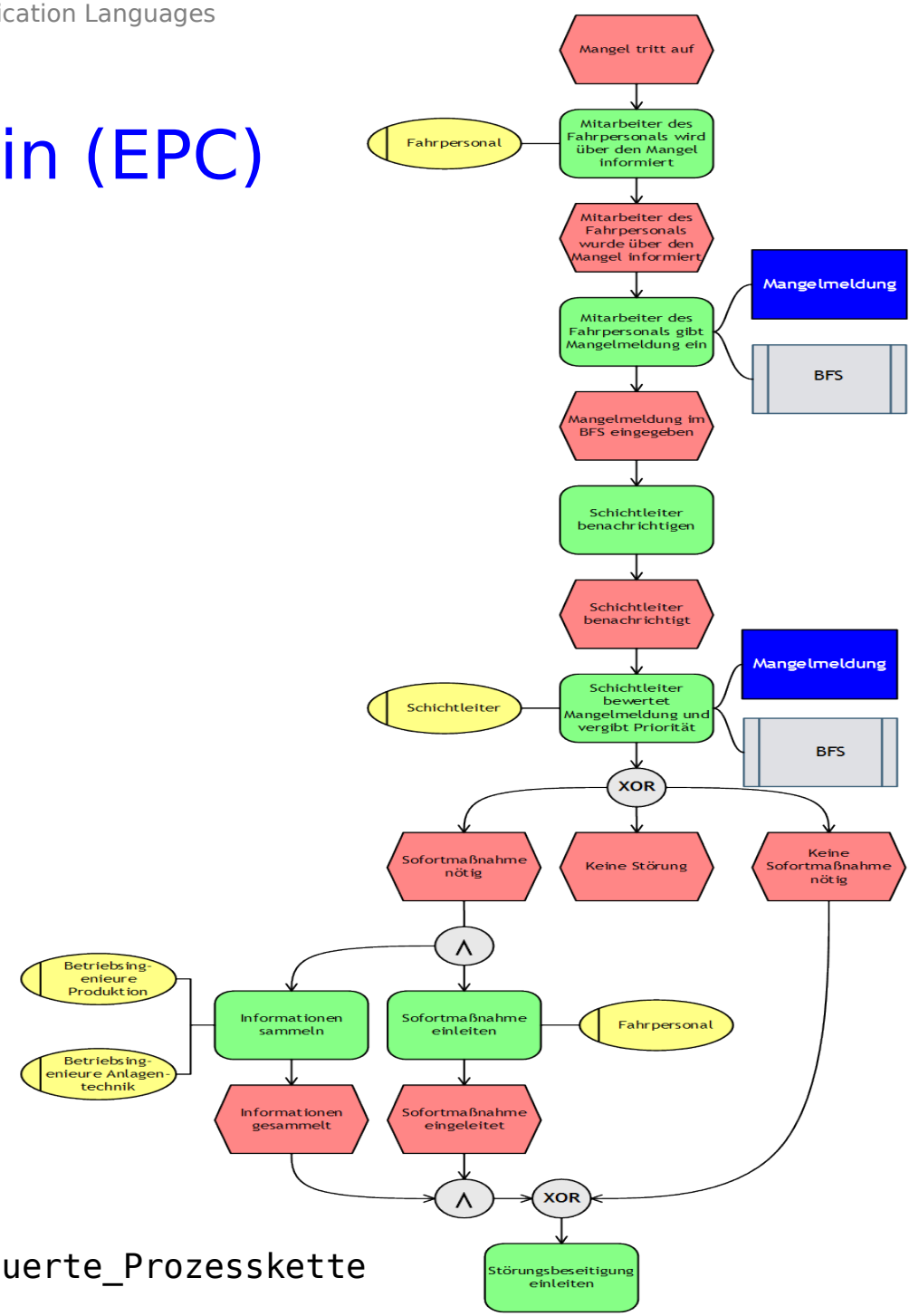
Dr. Martin Bartonitz, Nov. 2009

Notations for Structured Programming

- Programme Flow Chart
- Structure Chart/Nassi-Shneiderman Diagram (NSD)
- Jackson Structured Programming (JSP) Diagram
- Warnier/Orr Diagram

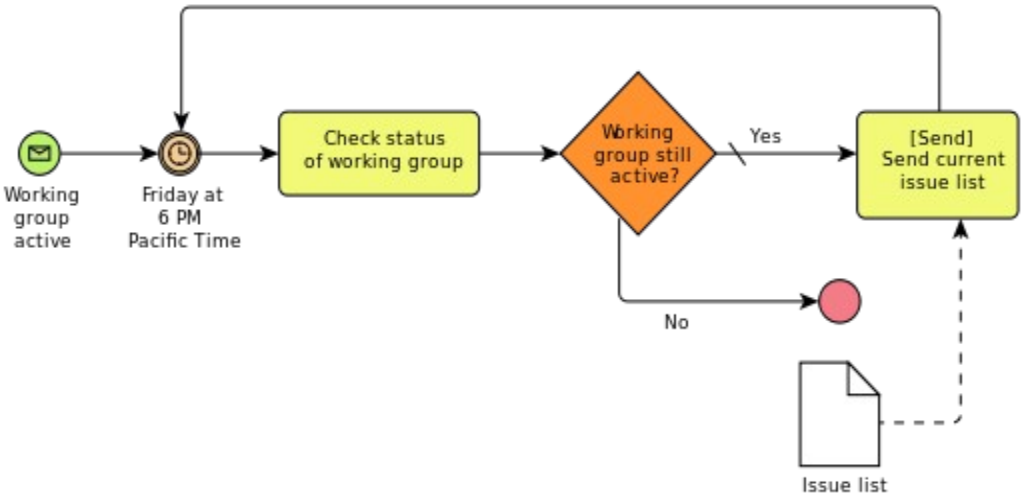
Event-Driven Process Chain (EPC)

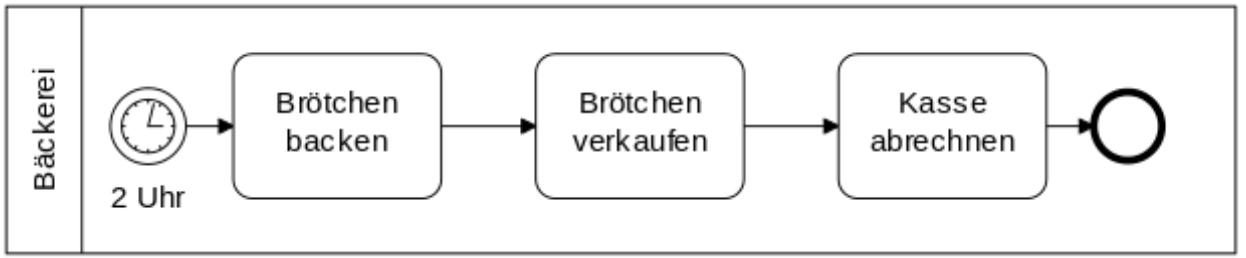




https://de.wikipedia.org/wiki/Ereignisgesteuerte_Prozesskette
https://de.wikipedia.org/wiki/ARIS

Business Process Model and Notation (BPMN)





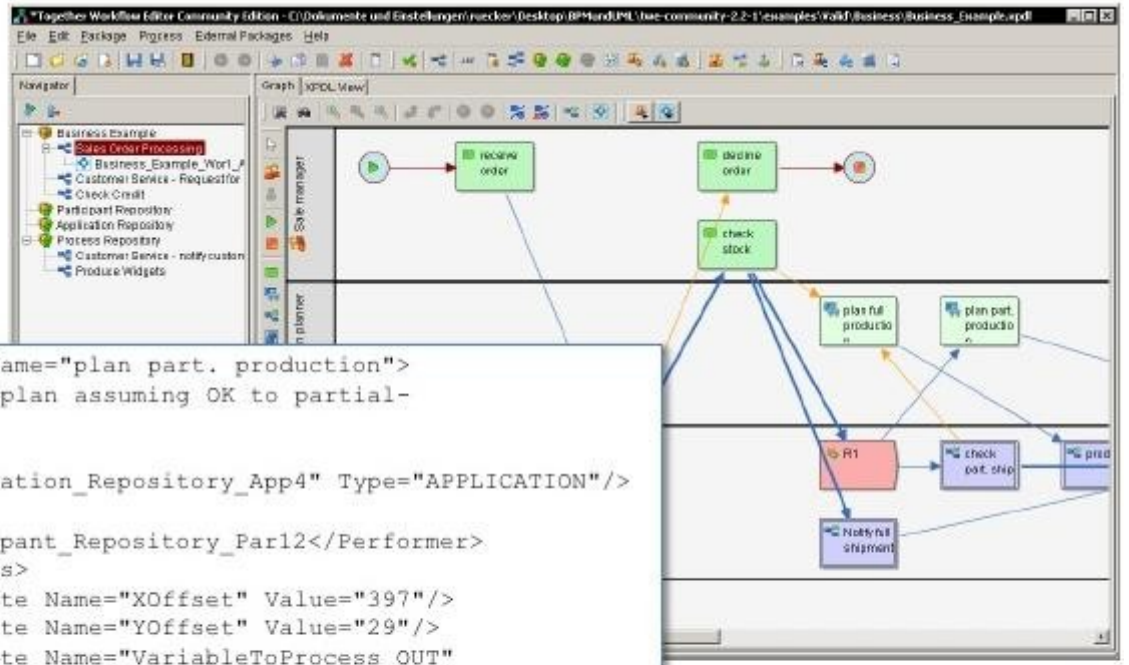
https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation
https://de.wikipedia.org/wiki/Business_Process_Model_and_Notation

OMG Modeling and Metadata Specifications

SPECIFICATION	acronym	topical area / domain	Document #
Action Language for Foundational UML	ALF	modeling	formal/2013-09-01
Common Terminology Services 2	CTS2™	modeling	formal/2015-04-01
Common Warehouse Metamodel	CWM™	data warehousing, modeling	formal/2003-03-02
CWM Metadata Interchange Patterns	MIPS	data warehousing, modeling	formal/2004-03-25
Diagram Definition	DD	modeling	formal/2015-06-01
Distributed Ontology, Modeling, and Specification Language	DOL	modeling	ptc/2016-02-37
Essence - Kernel and Language for Software Engineering Methods	Essence	modeling	formal/2015-12-02
Interaction Flow Modeling Language	IFML™	modeling	formal/2015-02-05
Languages, Countries, and Codes	LCC	modeling	ad/2015-11-04
Meta Object Facility Core	MOF™	modeling	formal/2015-06-05
Model Driven Message Interoperability	MDMI™	modeling	formal/2010-03-01
MOF Model to Text Transformation Language	MOFM2T	modeling	formal/2008-01-16
MOF Query / View / Transformation	QVT™	modeling	ptc/2015-10-02
MOF Support for Semantic Structures	SMOF	modeling	formal/2013-04-02
MOF 2 Facility and Object Lifecycle	MOFFOL	modeling	formal/2010-03-04
MOF 2 Versioning and Development Lifecycle	MOFVD	modeling	formal/2007-05-01
Object Constraint Language	OCL	modeling	formal/2014-02-03
OMG Systems Modeling Language	SysML®	modeling	formal/2012-06-01
Ontology Definition Metamodel	ODM	modeling	formal/2014-09-02
Precise Semantics of UML Composite Structures	PSCS™	modeling	formal/2015-10-02
Reusable Asset Specification	RAS	modeling	formal/2005-11-02
Semantics of a Foundational Subset for Executable UML Models	FUML™	modeling	formal/2016-01-05
Service oriented architecture Modeling Language	SoaML®	modeling	formal/2012-05-01
Software Process Engineering Metamodel	SPEM	modeling	formal/2008-04-01
Unified Modeling Language®	UML®	modeling	formal/2015-03-01
UML 2.0 Diagram Interchange (see Diagram Definition above for current version)	UMLDI	modeling	formal/2006-04-04
UML Human-Usable Textual Notation	HUTN	modeling	formal/2004-08-01
XML Metadata Interchange (aka MOF to XMI Mapping)	XMI®	modeling	formal/2015-06-07

http://www.omg.org/spec/#M&M

XML Process Definition Language (XPDL)



```
<Activity Id="xxx" Name="plan part. production">
  <Description>Make plan assuming OK to partial-
ship</Description>
  <Implementation>
    <Tool Id="Application_Repository_App4" Type="APPLICATION"/>
  </Implementation>
  <Performer>Participant_Repository_Par12</Performer>
  <ExtendedAttributes>
    <ExtendedAttribute Name="XOffset" Value="397"/>
    <ExtendedAttribute Name="YOffset" Value="29"/>
    <ExtendedAttribute Name="VariableToProcess_OUT"
                       Value="No_Of_Items_To_Produce"/>
    <ExtendedAttribute Name="VariableToProcess_IN"
                       Value="Order_Quantity"/>
    <ExtendedAttribute Name="VariableToProcess_IN"
                       Value="No_Of_Stocked_Items"/>
  </ExtendedAttributes>
</Activity>
```

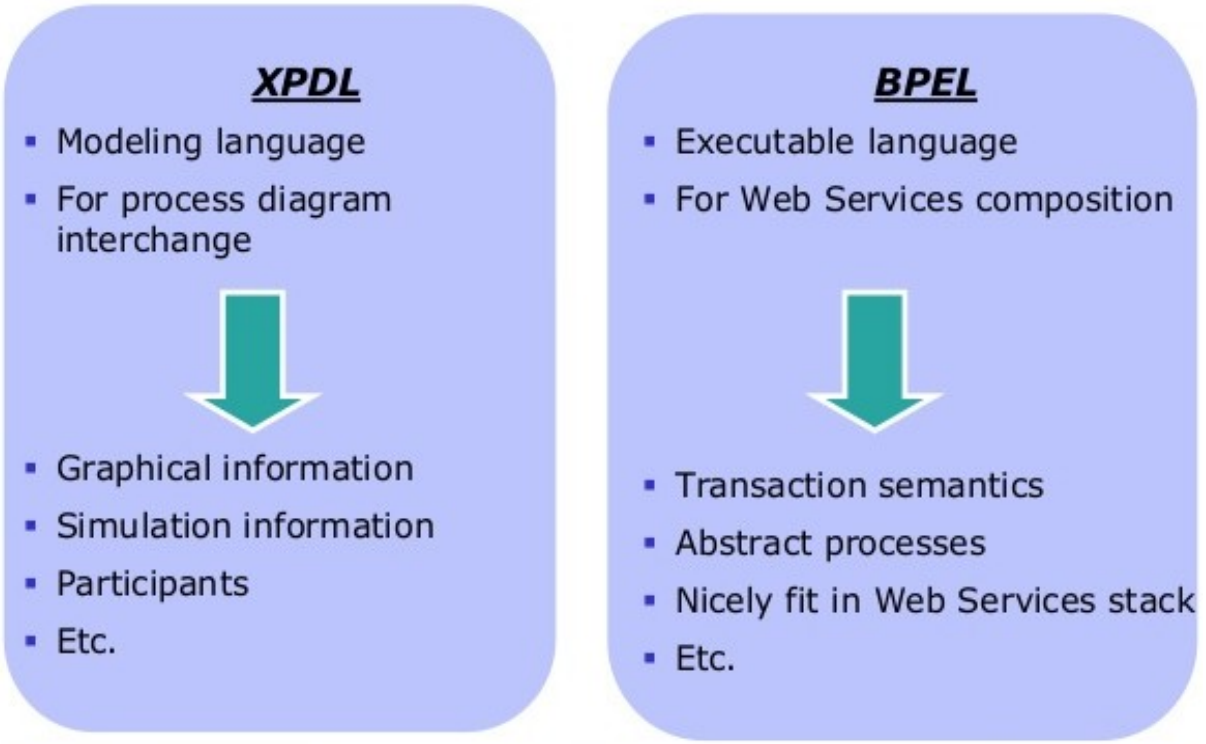
camunda services GmbH
http://de.slideshare.net/camunda/bpm-soa-prozesse-sind-keine-workflows-presentation
https://de.wikipedia.org/wiki/XML_Process_Definition_Language

WS-Business Process Execution Language (BPEL)

```
<process name="NCName" targetNamespace="anyURI" xmlns="http://docs.oasis-open.org/...">
    <extensions>? ... </extensions>
    <import namespace="anyURI"? .../>*
    <partnerLinks>?
        <!-- Note: At least one role must be specified. -->
        ...
    </partnerLinks>
    <messageExchanges>?
        <messageExchange name="NCName" />+
    </messageExchanges>
    <variables>? ... </variables>
    <correlationSets>? ... </correlationSets>
    <faultHandlers>?
        <!-- Note: There must be at least one faultHandler -->
        ...
    </faultHandlers>
    <eventHandlers>?
        <!-- Note: There must be at least one onEvent or onAlarm. -->
        <onEvent partnerLink="NCName" ...?>* ... </onEvent>
        <onAlarm>* ... </onAlarm>
    </eventHandlers>
    activity
</process>
```

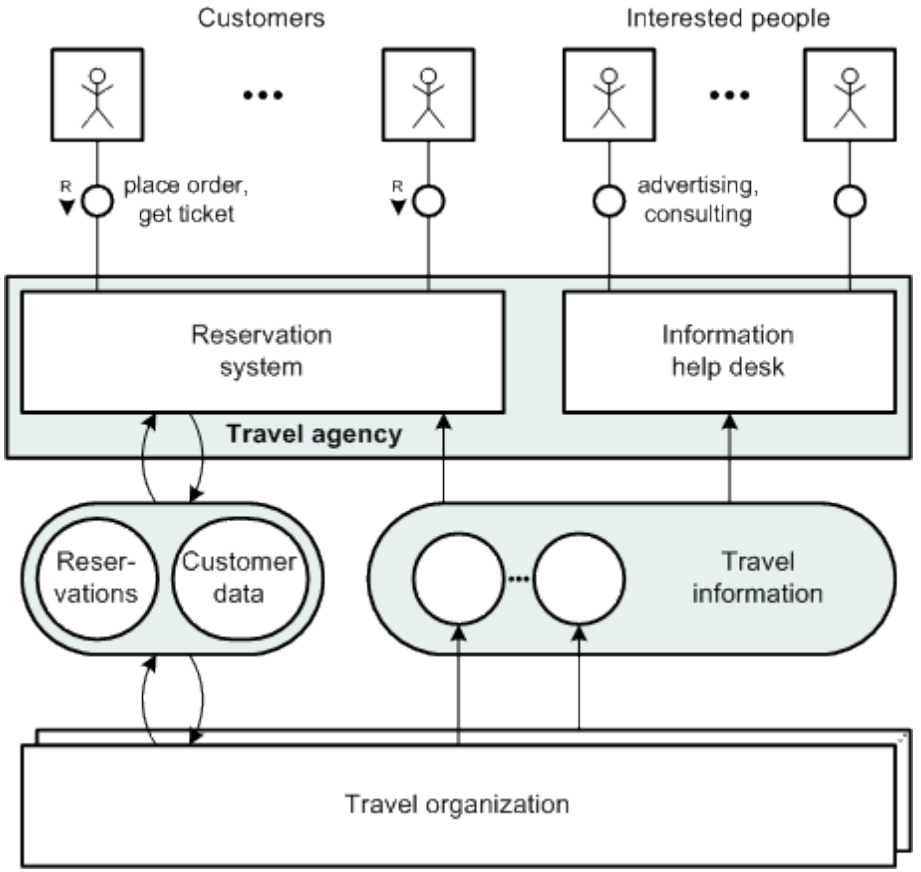
http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
https://de.wikipedia.org/wiki/WS-Business_Process_Execution_Language

XPDL-BPEL Comparison



Mike Marin, Software Architect at IBM Software Group
http://www.slideshare.net/MikeMarin1/2007-1109-mm-costa-rica-incae-cit-omg-modeling-with-bpmn-and-xpdl

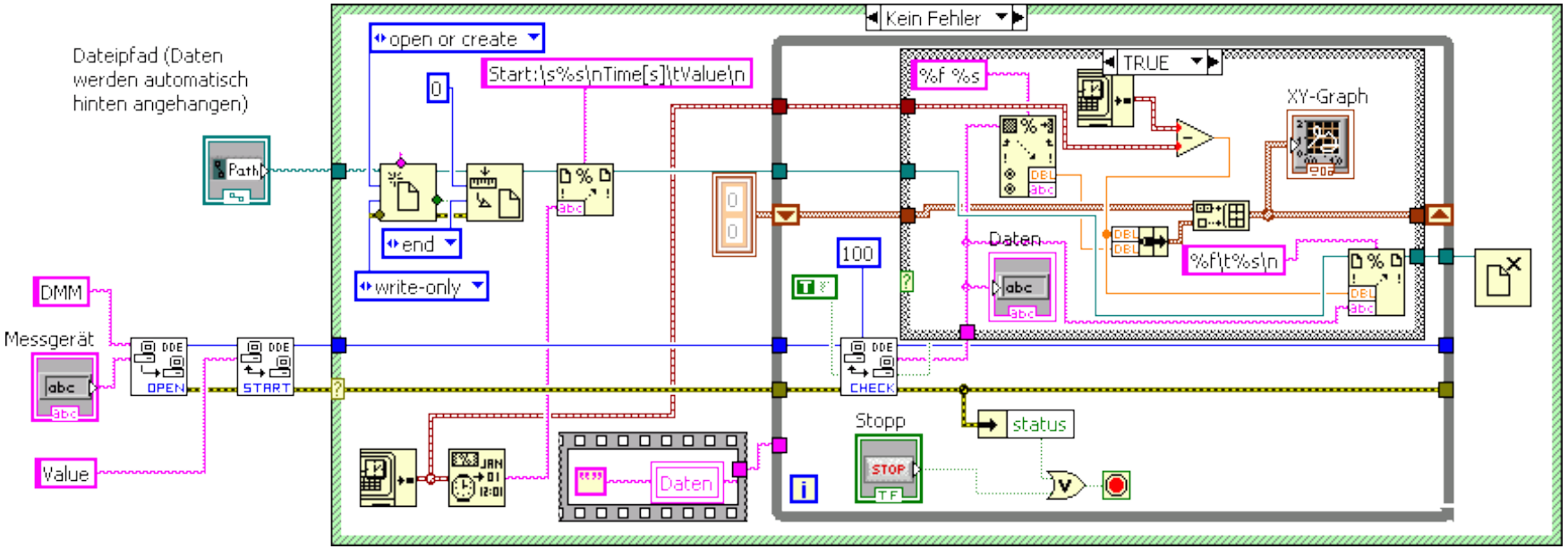
Fundamental Modeling Concepts (FMC)



http://www.fmc-modeling.org/quick-intro
https://de.wikipedia.org/wiki/Fundamental_Modeling_Concepts

Visual Programming Language (VPL)

MIT Scratch an Tafel
https://de.wikipedia.org/wiki/Visuelle_Programmiersprache



National Instruments (NI) Laboratory Virtual Instrument Engineering Workbench (LabVIEW)
Henrik Haftmann https://www-user.tu-chemnitz.de/~heha/hs/UNI-T/

Summary

- Notations for Structured Programming (Flowchart etc.)
- Modelling Language (EPC, BPMN, XPDL, FMC)
- Execution Language (BPEL)
- Visual Programming Language

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

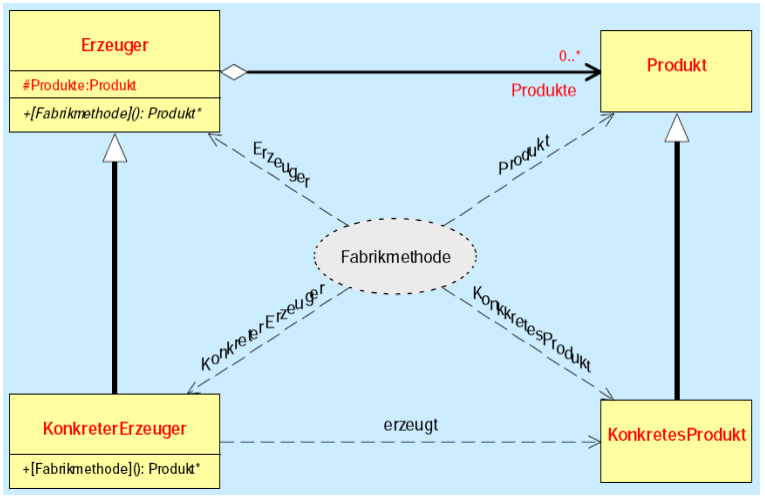
Diagram Definition (DD)

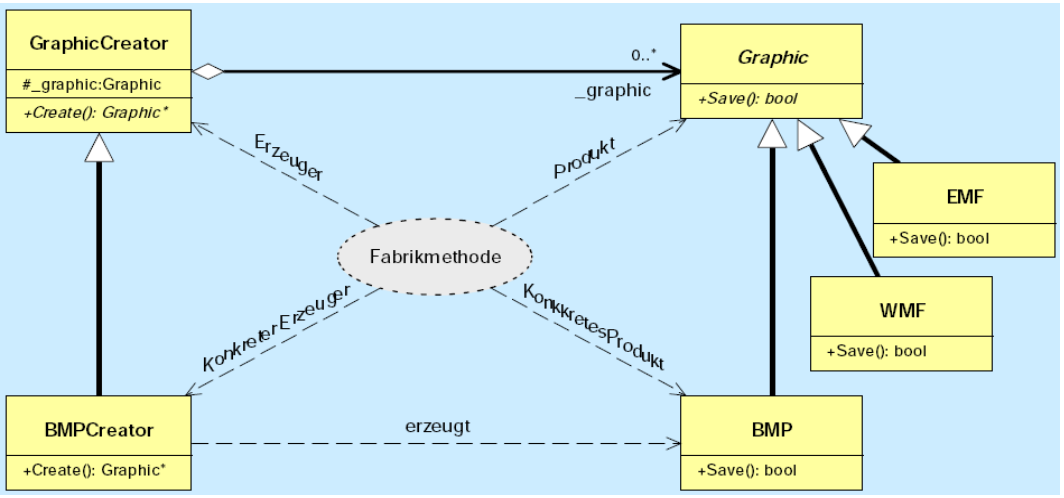
Related Specification Languages

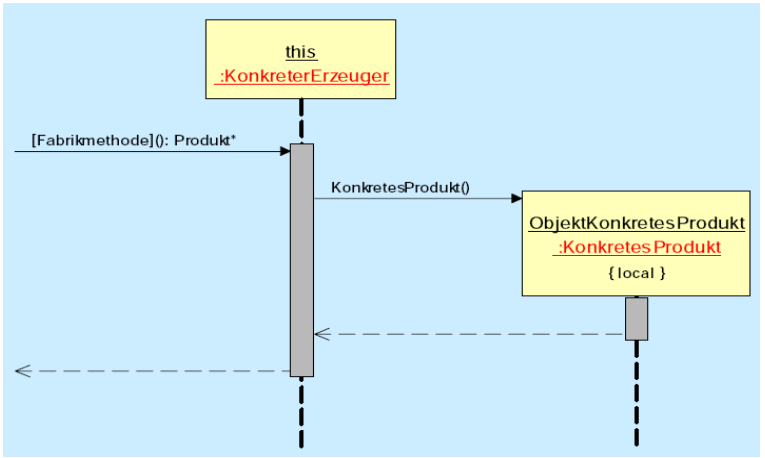
Software Pattern

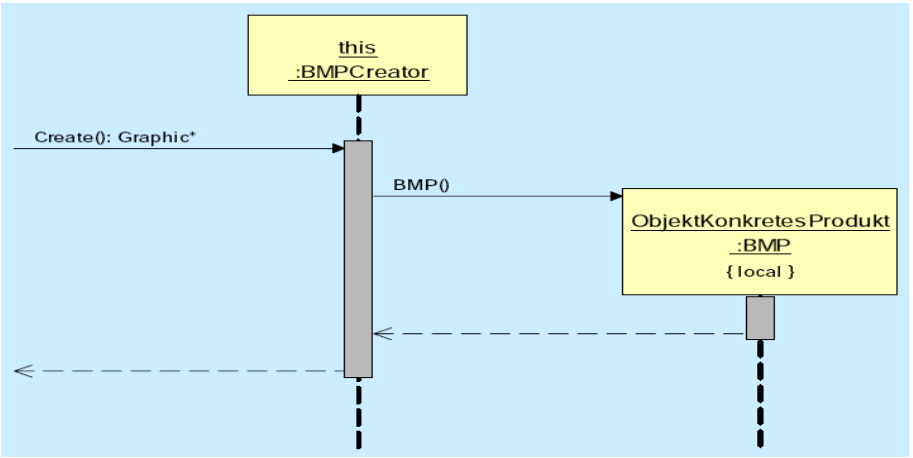
Pattern Catalogue

Model Reuse Example (Pattern Instantiation)



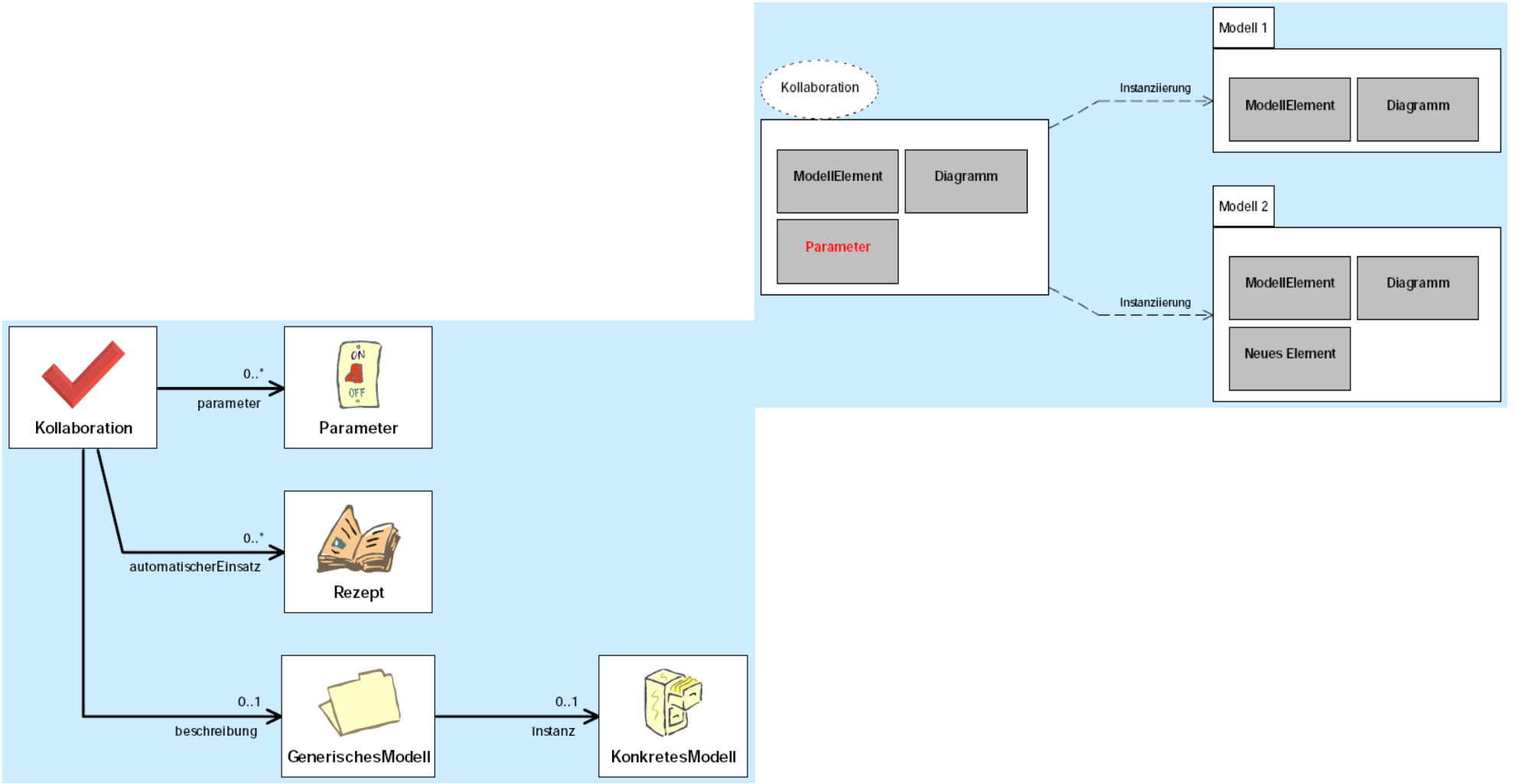






Objektorientierte und Wissensbasierte Systeme (OWiS)
Object Technology Workbench (OTW) UML Tool

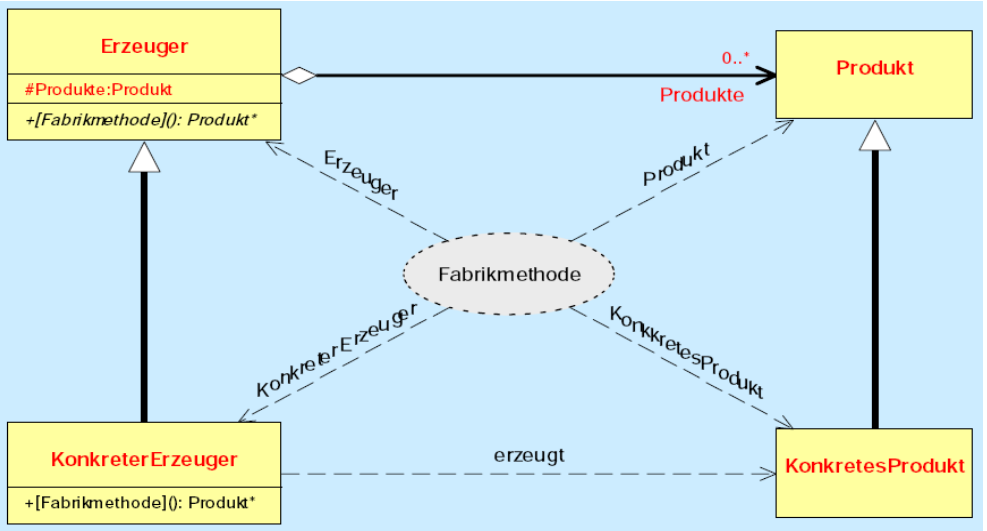
CASE Tool Support



Martin Wolf. Objektorientierte Prozeßmodellierung mit der UML. Vorlesungsunterlagen Wintersemester 2001/2002. Technische Universität Ilmenau, 2002

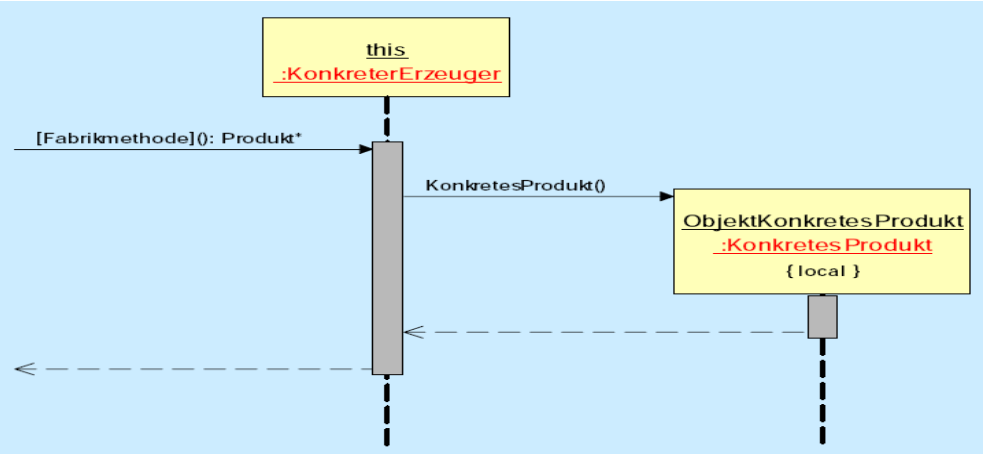
Pattern

- general reusable solution to commonly occurring problem
- not a finished design that can be transformed into code
- description or template for how to solve a problem
- can be used in many different situations



Not a pattern:

- algorithm: solves computational (not design) problems
- paradigm: describes a programming style



Documentation

- **Pattern Name and Classification:** A descriptive and unique name that helps in identifying and referring to the pattern.
- **Intent:** A description of the goal behind the pattern and the reason for using it.
- **Also Known As:** Other names for the pattern.
- **Motivation (Forces):** A scenario consisting of a problem and a context in which this pattern can be used.
- **Applicability:** Situations in which this pattern is usable; the context for the pattern.
- **Structure:** A graphical representation of the pattern. Class diagrams and Interaction diagrams may be used for this purpose.
- **Participants:** A listing of the classes and objects used in the pattern and their roles in the design.
- **Collaboration:** A description of how classes and objects used in the pattern interact with each other.
- **Consequences:** A description of the results, side effects, and trade offs caused by using the pattern.
- **Implementation:** A description of an implementation of the pattern; the solution part of the pattern.
- **Sample Code:** An illustration of how the pattern can be used in a programming language.
- **Known Uses:** Examples of real usages of the pattern.
- **Related Patterns:** Other patterns that have some relationship with the pattern; discussion of the differences between the pattern and similar patterns.

Terms and Definitions

- Programme
- Programme Library
- Software Pattern
- Framework

History

Kent Beck

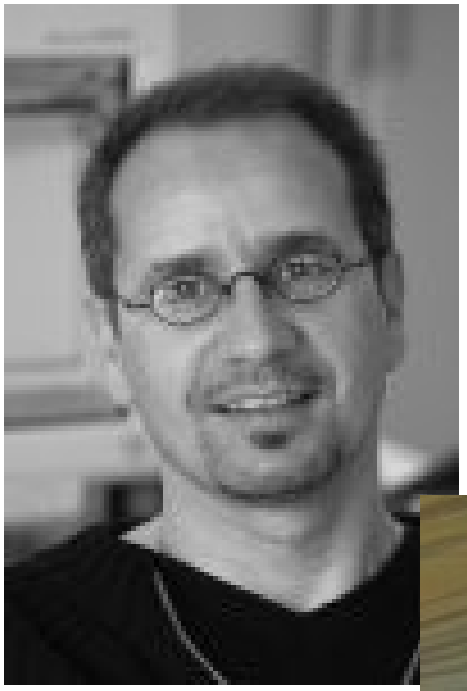
Christopher Alexander

Ward Cunningham

http://www.wikipedia.org/

Authors

Martin Fowler



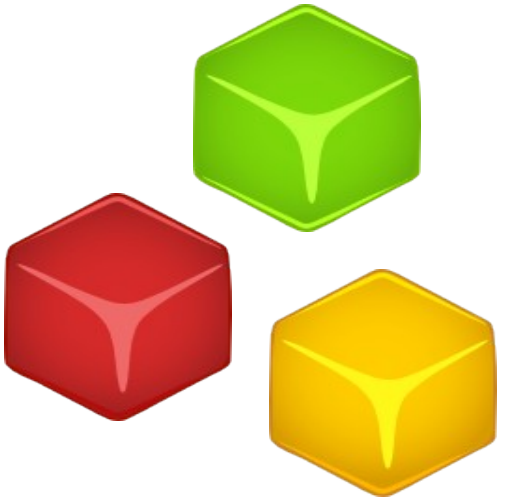
Erich Gamma

Gregor Hohpe

Douglas C. Schmidt

http://www.wikipedia.org/

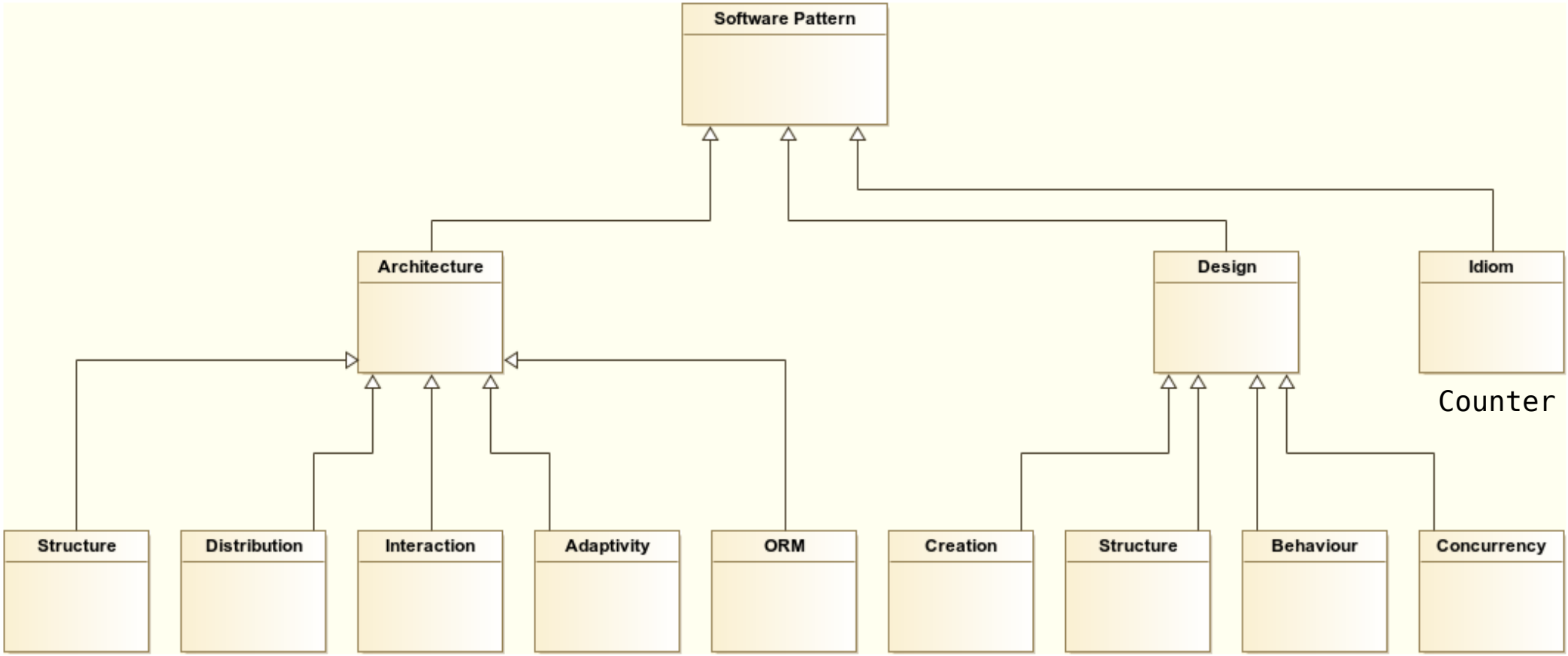
Kinds of Patterns (Examples)



- Analysis
- Communication
- Organisation
- Anti
- Architecture
- Design
- Idiom

https://de.wikipedia.org/wiki/Entwurfsmuster
https://en.wikipedia.org/wiki/Software_design_pattern

Classification



Alternative Classification (CYBOP)

Grouping (Categorisation):
Layers, Layer Supertype, Domain Model, Data Mapper, Model View Controller (MVC), Hierarchical MVC (HMVC), Pipes and Filters, Broker

Itemisation (Discrimination):
Data Transfer Object (DTO), Command / Action / Transaction, State, Memento, Envelope-Letter, Prototype

1:1 Association (Composition):
Delegator, Wrapper / Adapter, Object Adapter, Proxy / Surrogat / Client-Server Stub, Handle-Body / Pointer / Wrapper Object, Bridge

1:n Association (Composition):
Whole-Part, View Handler, Broker / Mediator, Master-Slave, Command Processor, Counted Pointer

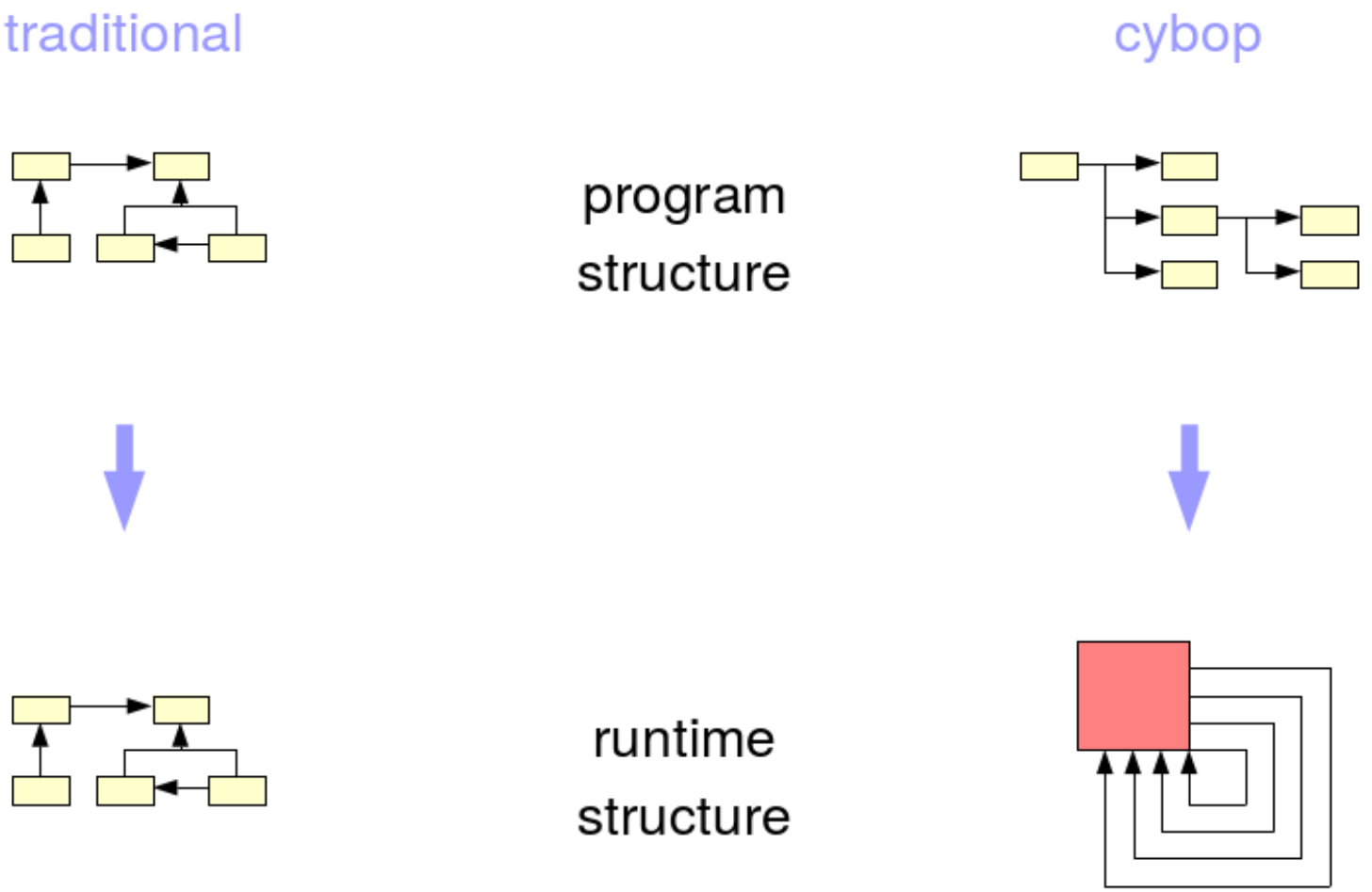
Recursion (Composition):
Composite, Chain of Responsibility / Event Handler / Bureaucrat / Responder, Decorator, Interpreter, Linked Wrapper

Polymorphism (Categorisation):
Template Method / Hook Method, Factory Method, Abstract Factory / Kit, Visitor, Builder, Class Adapter, Strategy / Validator / Policy, Iterator / Cursor, Interface, Inversion of Control (IoC) / Dependency Injection (DI)

Bidirectionalism (-):
Observer / Publisher-Subscriber / Callback, Reflexion / Meta-Level Architecture / Open Implementation, Forwarder-Receiver

Global Access (-):
Singleton, Registry, Manager, Flyweight

Universal Schema - Container Unification



Summary

- Model reuse: pattern instantiation
- Tool support: pattern catalogue
- Pattern: definition and documentation
- Classification: kinds of patterns

Contents

Introduction

Definition and Placement

Unified Modeling Language (UML)

Computer Aided Software Engineering (CASE)

Model Driven Architecture (MDA)

UML Meta Model

Meta Object Facility (MOF)

Profile and Stereotype

Object Constraint Language (OCL)

XML Metadata Interchange (XMI)

Diagram Definition (DD)

Related Specification Languages

Software Pattern

Pattern Catalogue

Classic Programmer Paintings

Enterprise Architect
(H. Rigaud circa 1701)

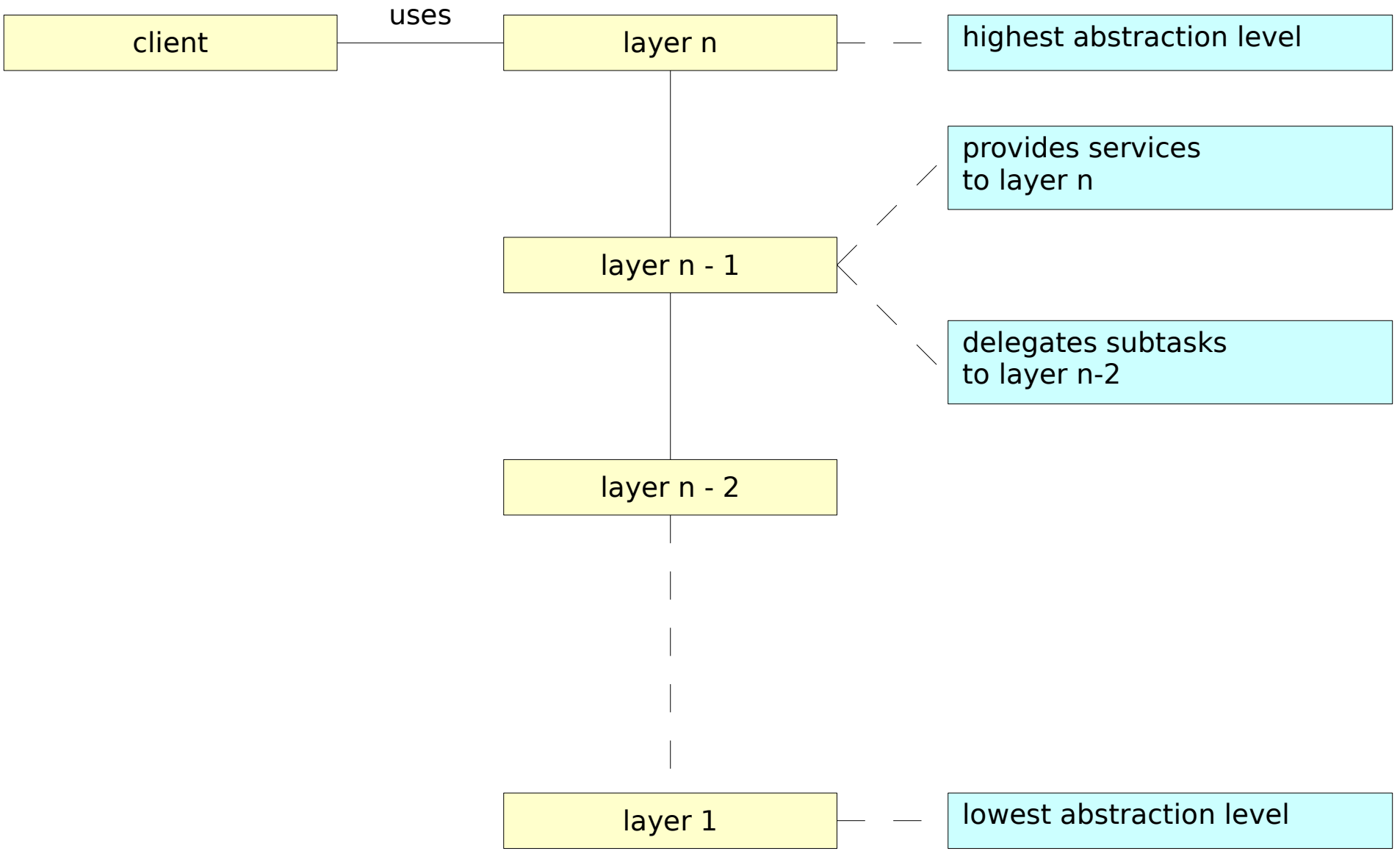
http://classicprogrammerpaintings.com/

Programmer finds 1395 conflicts after git merge, three days before deadline
(Gustav Courbet 1844–1845)

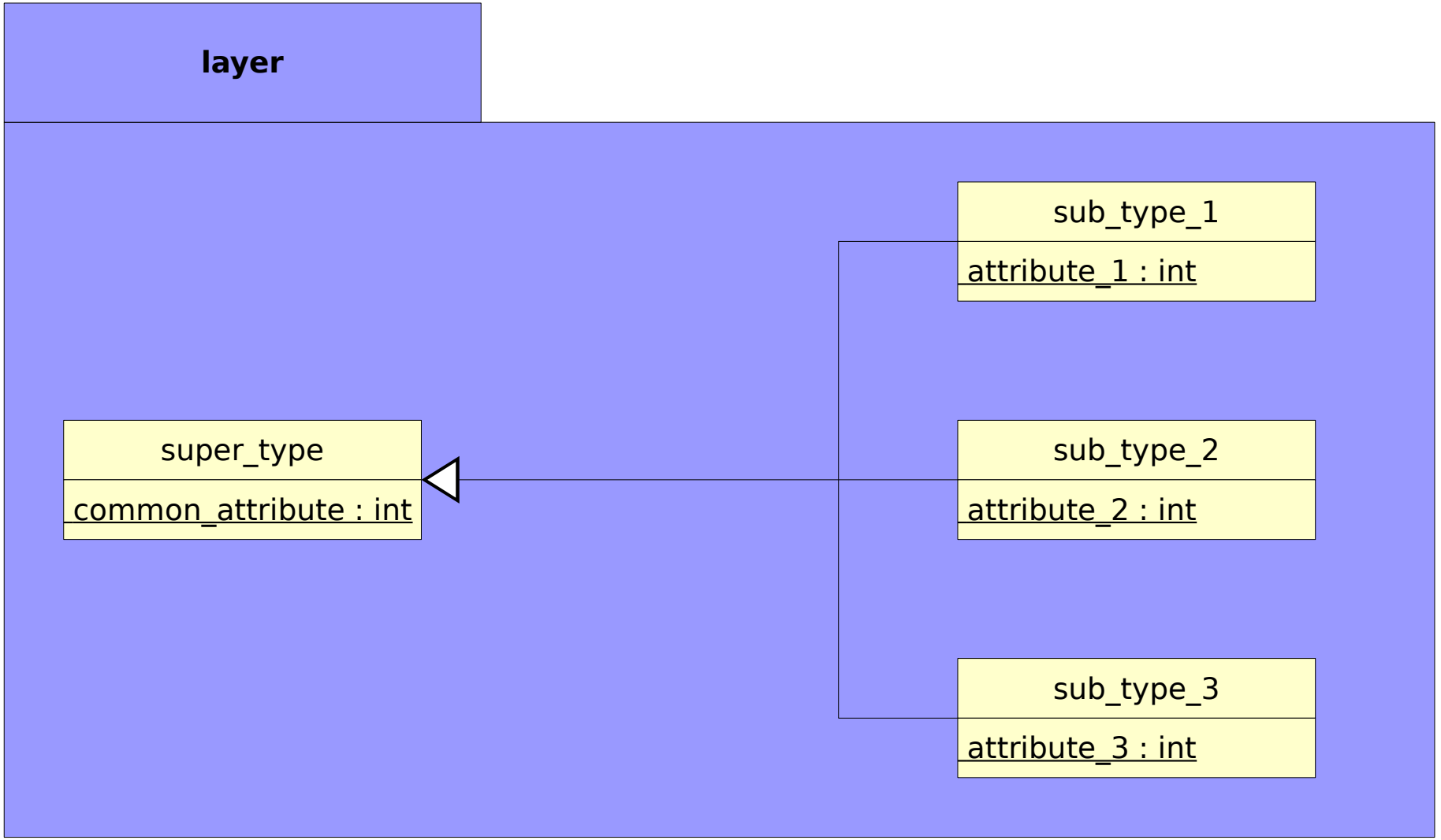
Multiple inheritance in C++
(Jacopo Ligozzi circa 1590)

Grouping

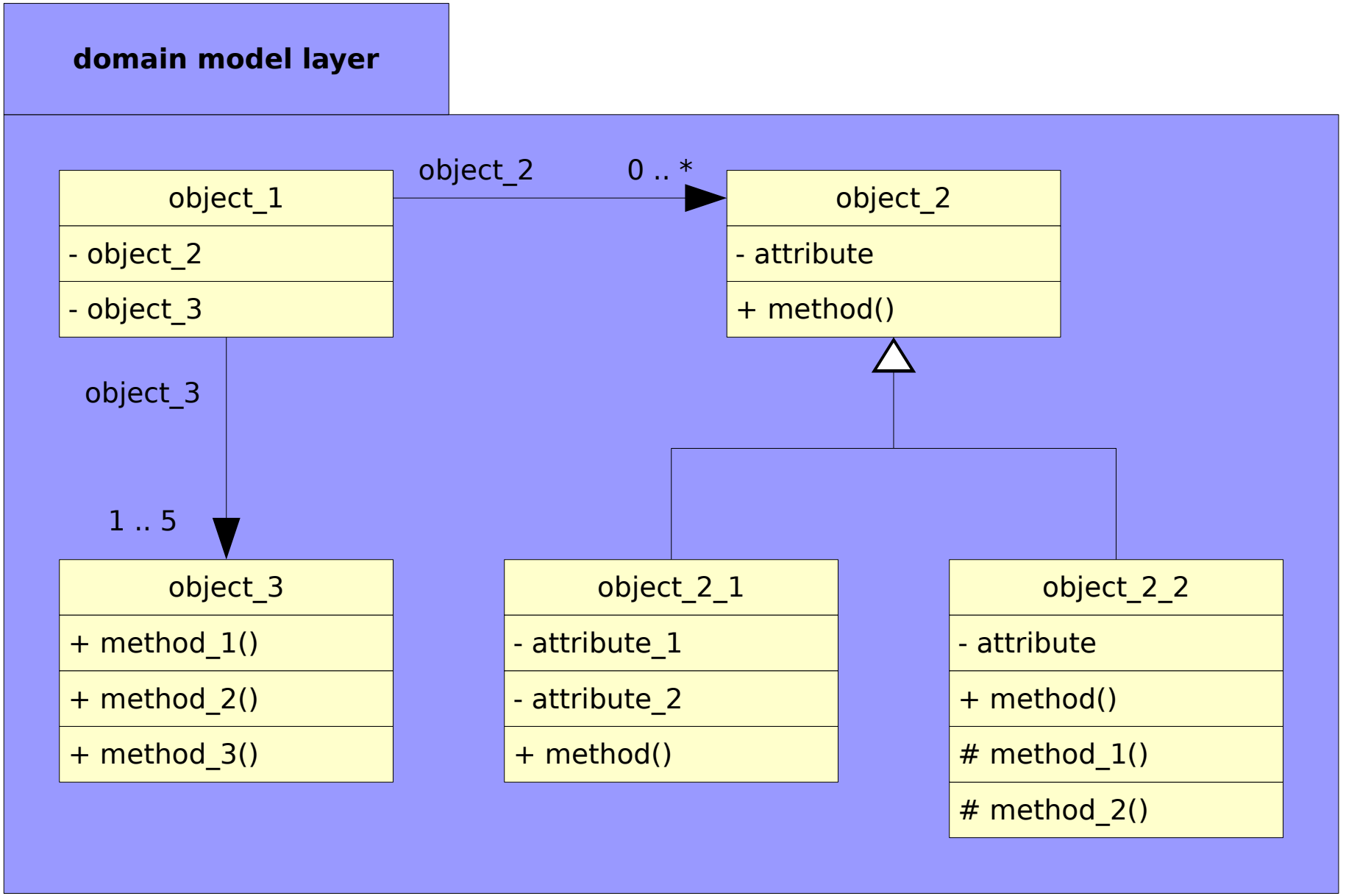
Layers (Buschmann)



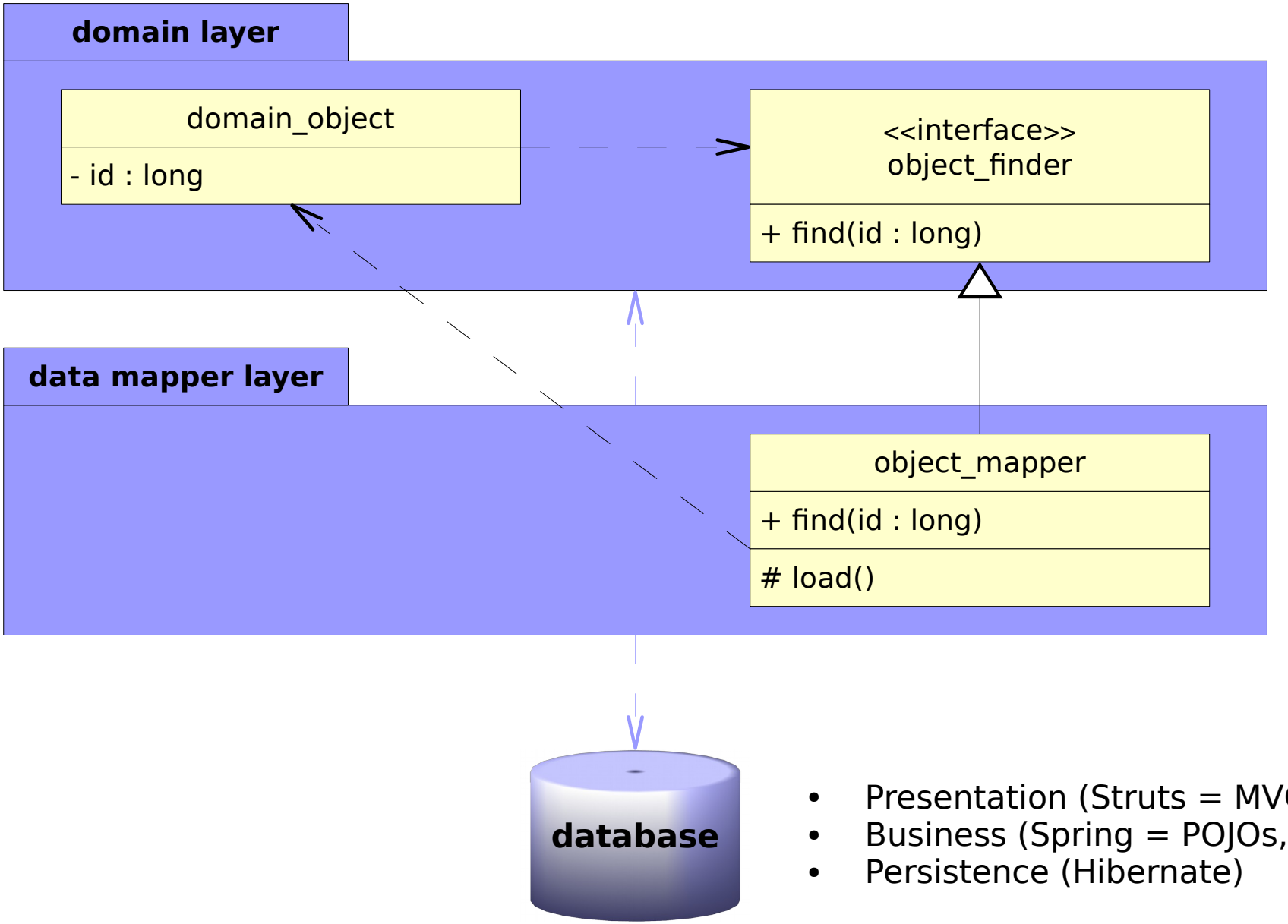
Supertype


layer
sub_type_1
attribute_1 : int
super_type
common_attribute : int
sub_type_2
attribute_2 : int
sub_type_3
attribute_3 : int

Domain Model

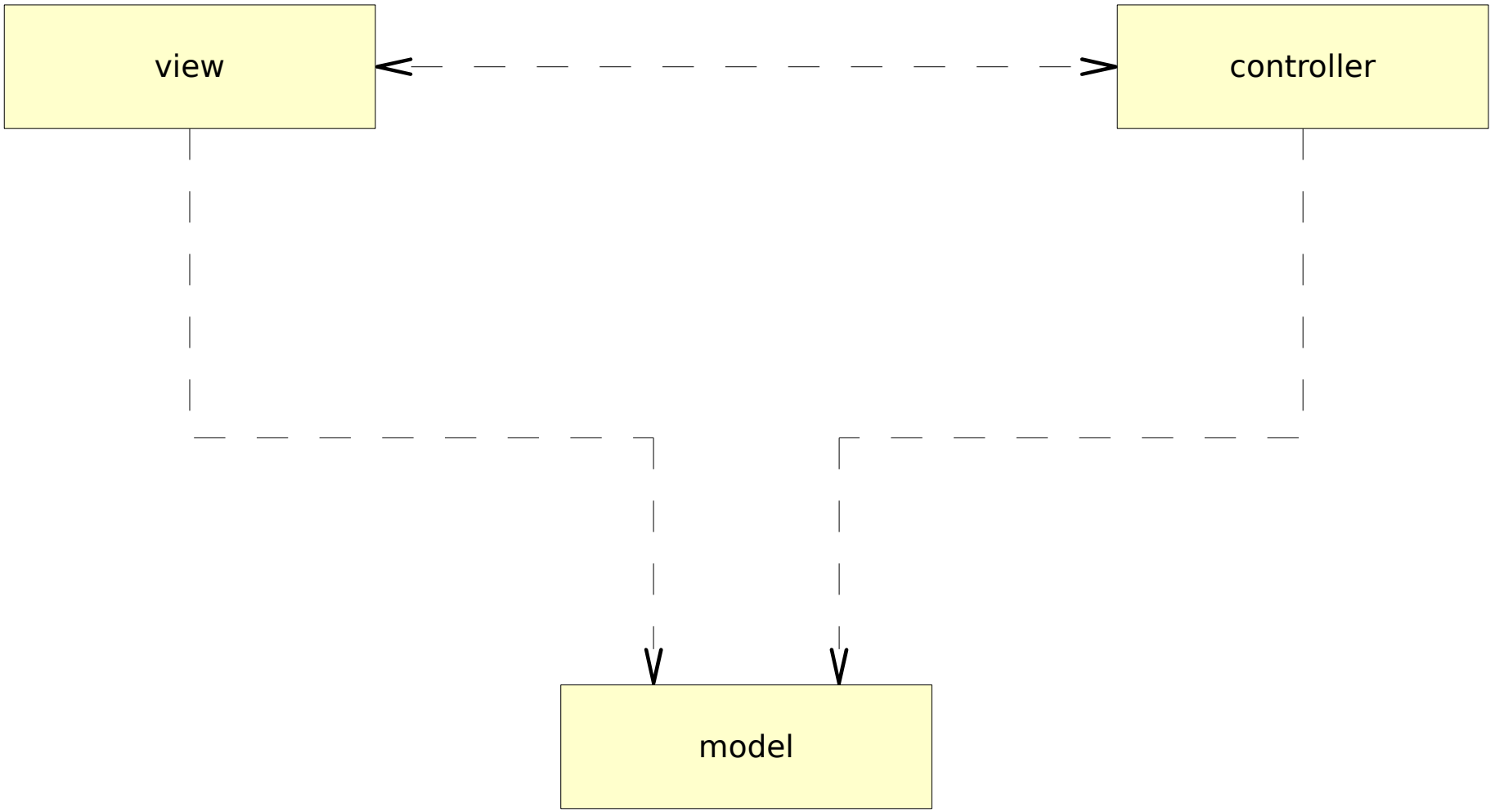


Data Mapper (Fowler)

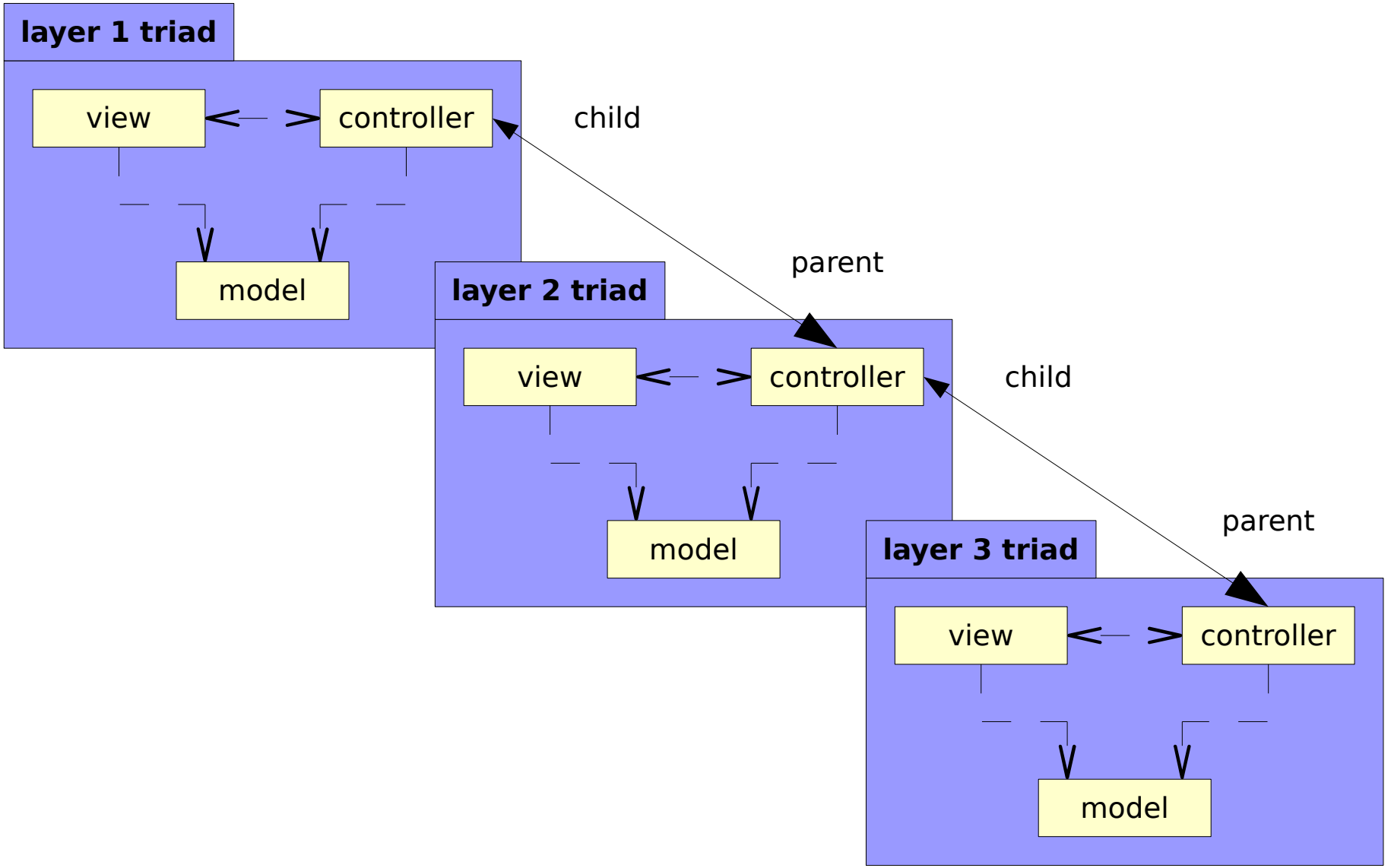


- Presentation (Struts = MVC Web)
- Business (Spring = POJOs, IoC)
- Persistence (Hibernate)

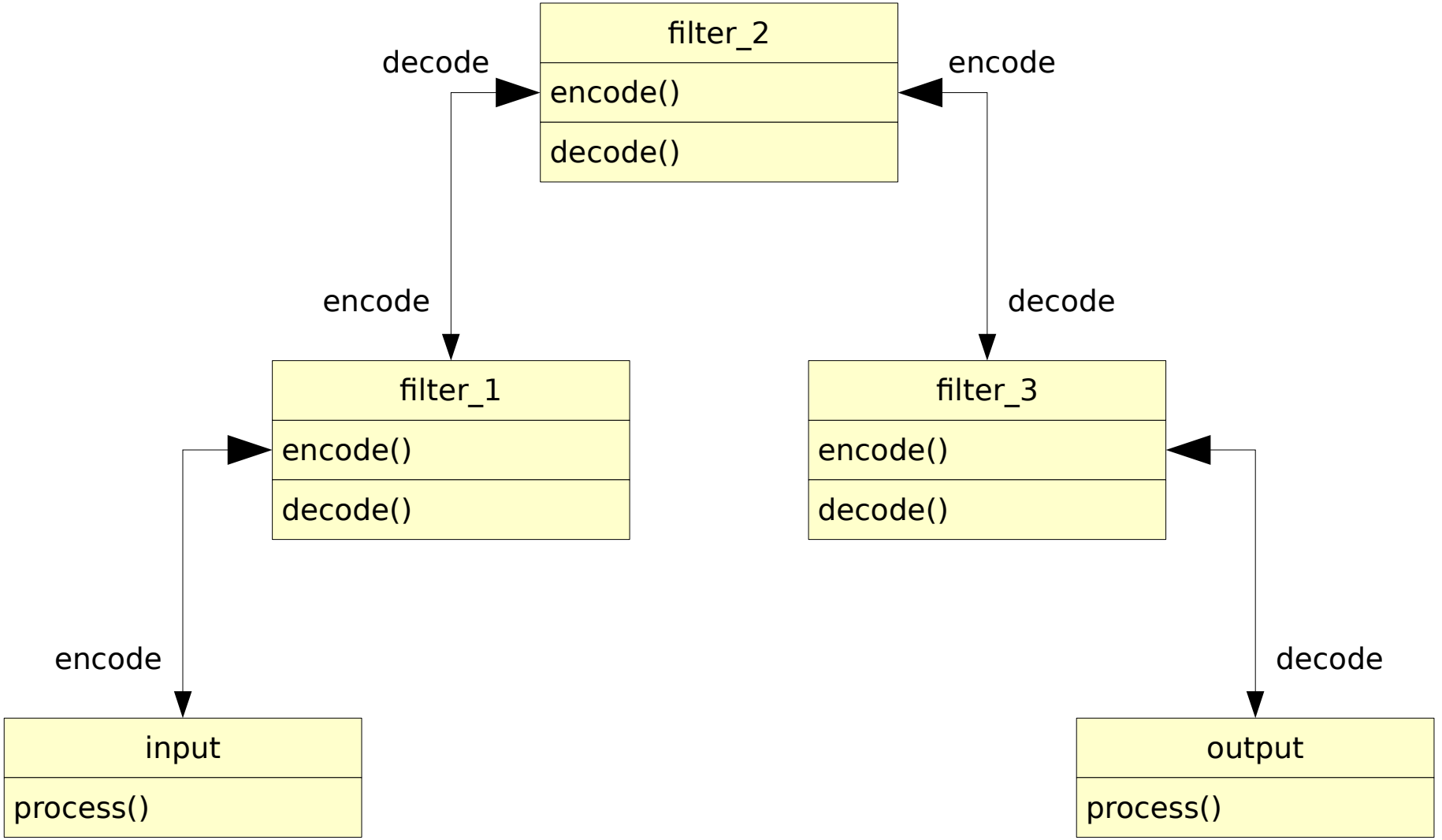
Model View Controller (MVC)



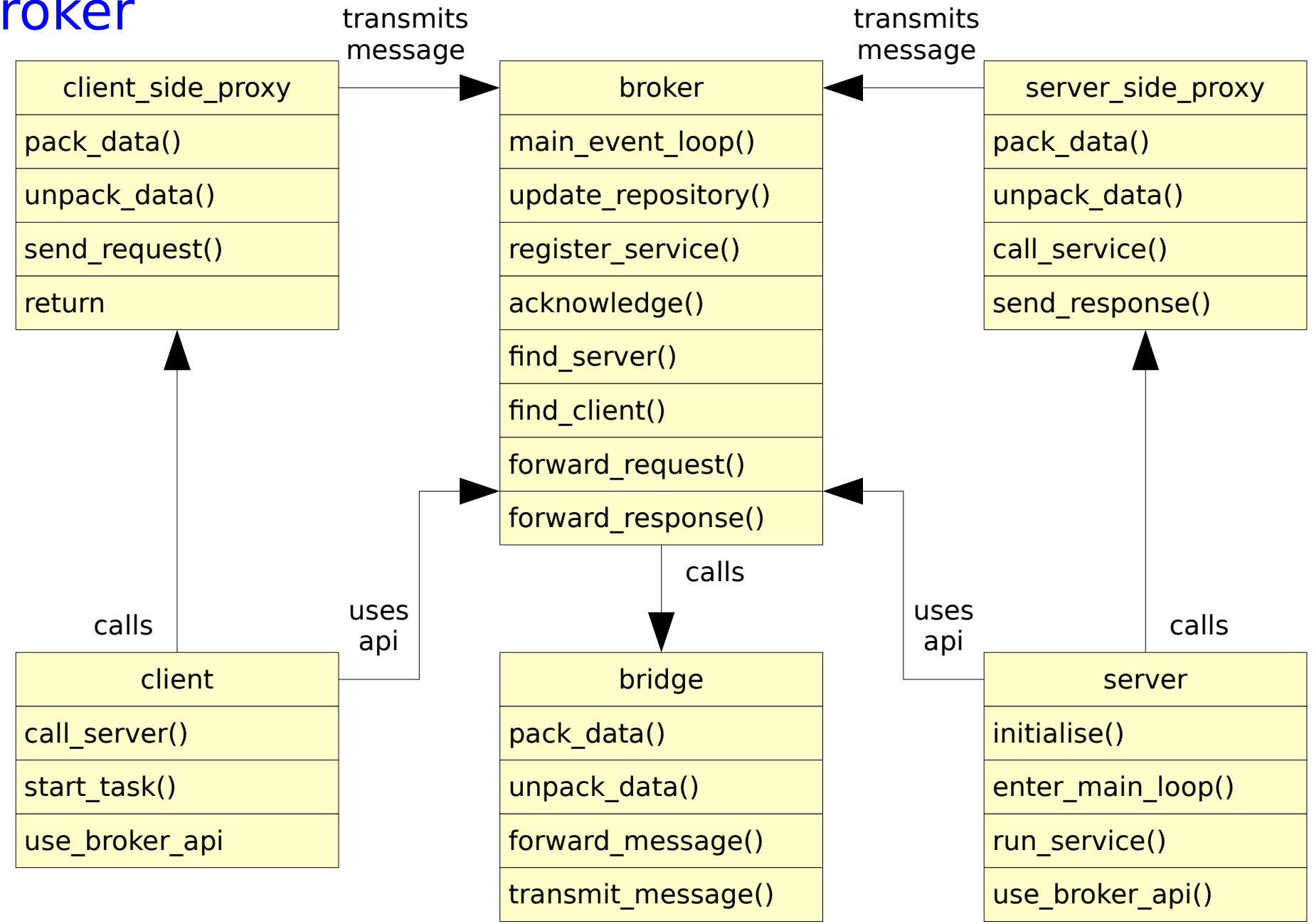
Hierarchical MVC (HMVC)



Pipes and Filters

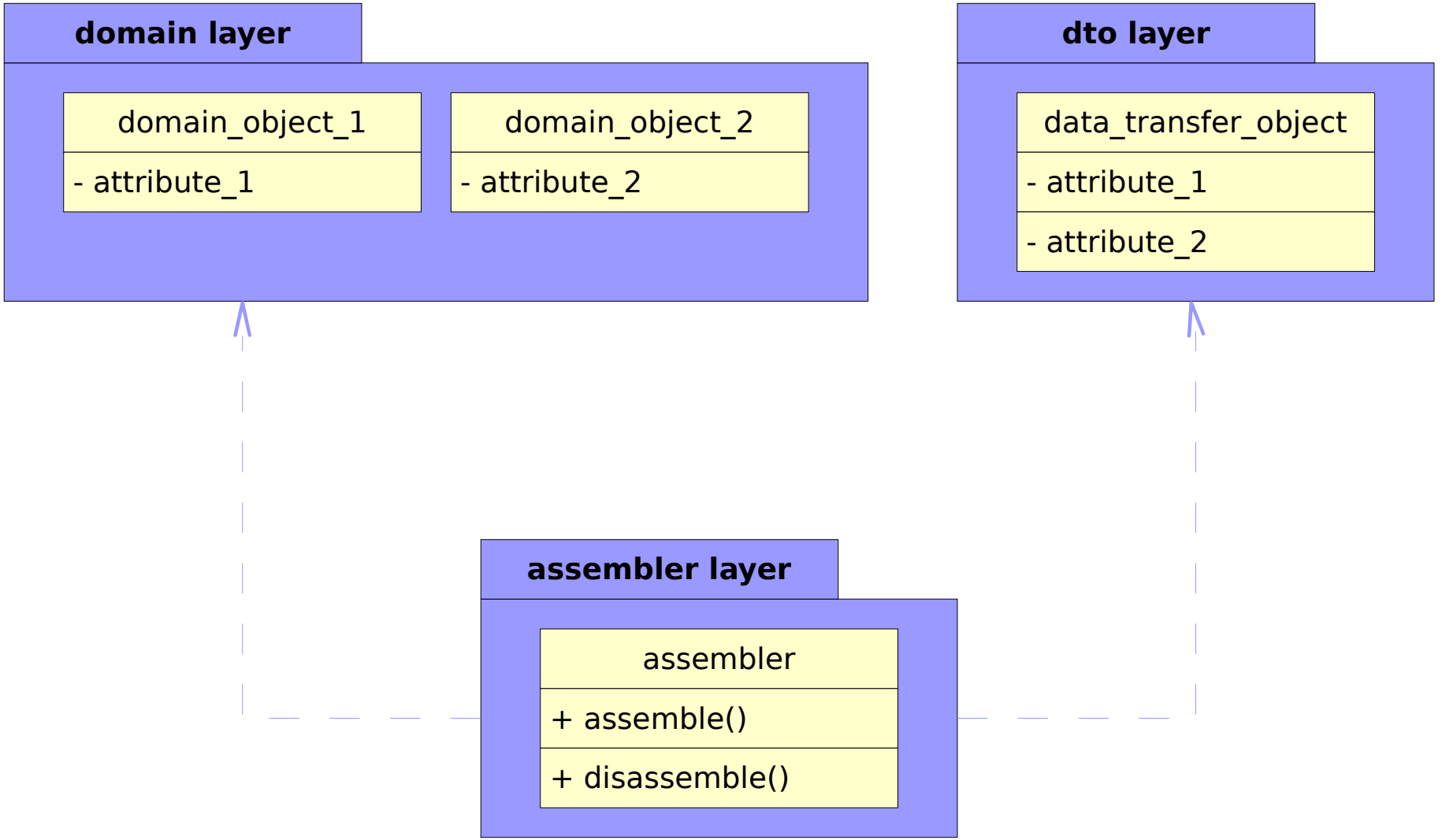


Broker

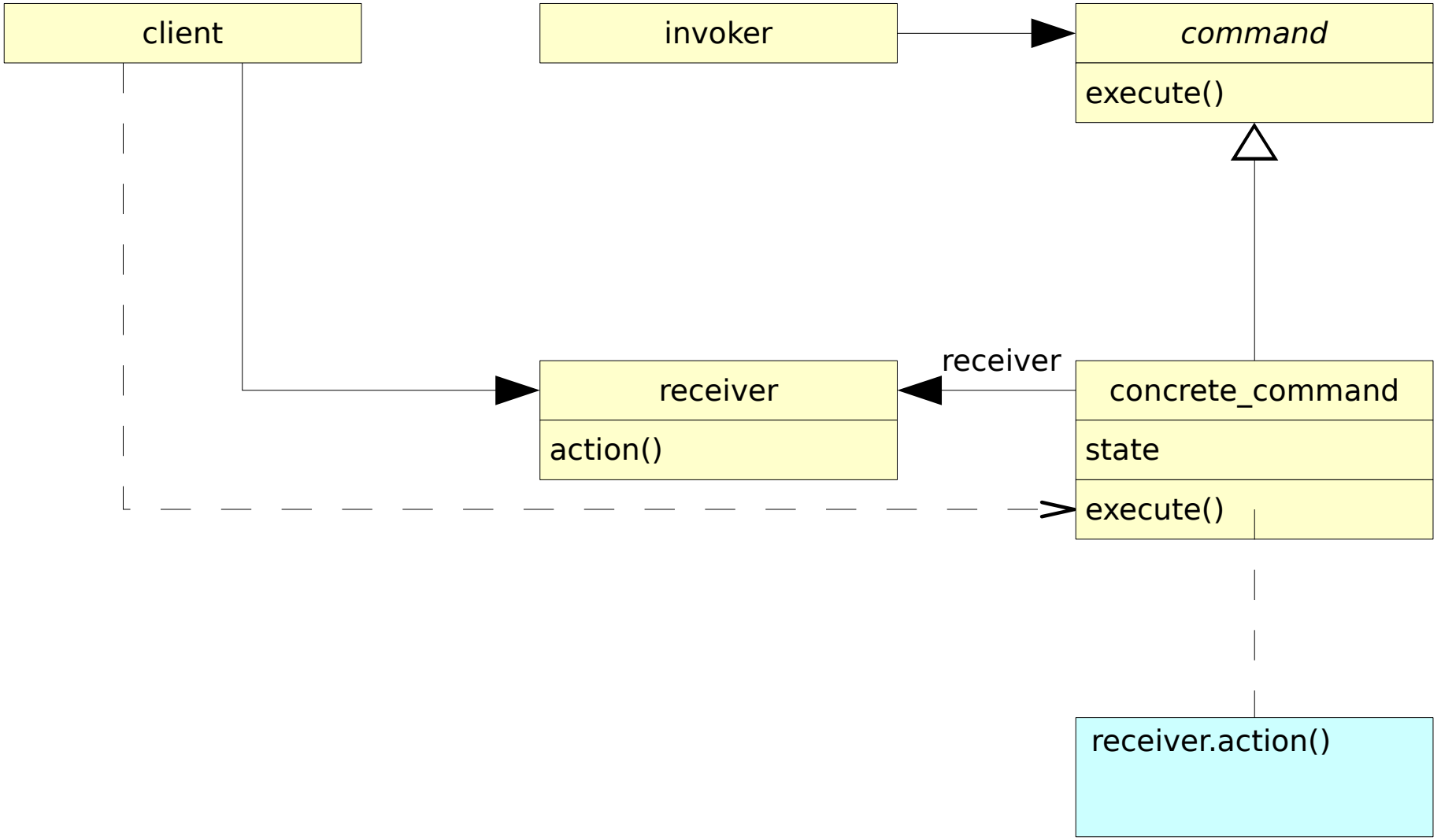

transmits message
transmits message
client_side_proxy
pack_data()
unpack_data()
send_request()
return
broker
main_event_loop()
update_repository()
register_service()
acknowledge()
find_server()
find_client()
forward_request()
forward_response()
server_side_proxy
pack_data()
unpack_data()
call_service()
send_response()
calls
uses api
calls
uses api
calls
client
call_server()
start_task()
use_broker_api
bridge
pack_data()
unpack_data()
forward_message()
transmit_message()
server
initialise()
enter_main_loop()
run_service()
use_broker_api()

Itemisation

Data Transfer Object (Fowler)



Command / Action / Transaction

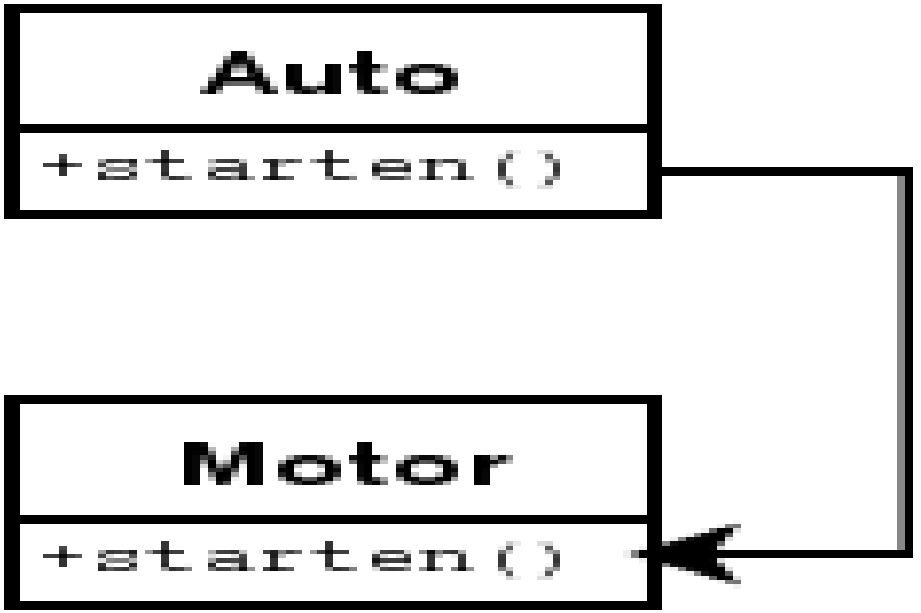


Immutable

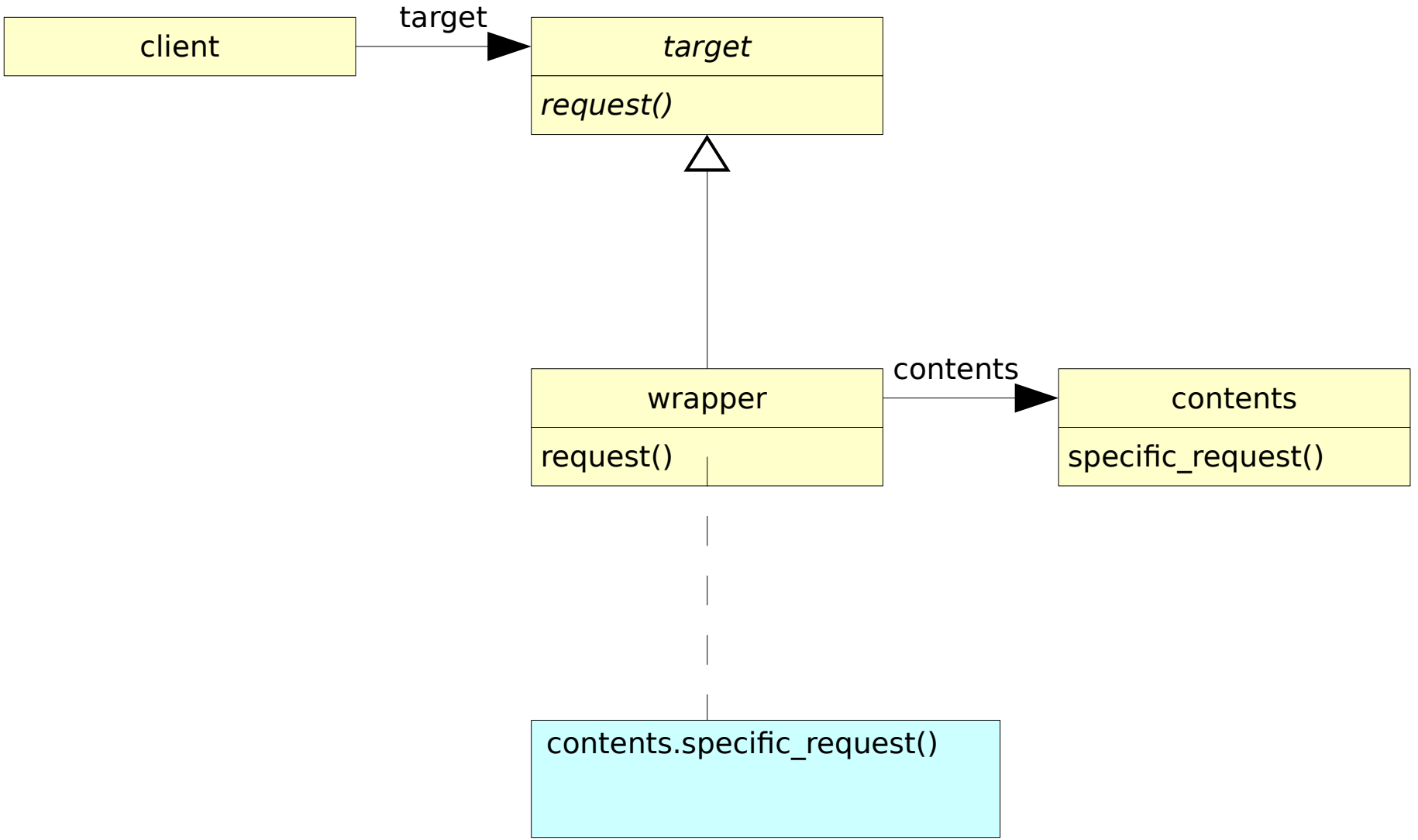
- object which is not changeable after its instantiation
- examples: java.lang.String, Wrapper classes (Integer, ...)
- all member variables are private
- member variables are set in constructor only
- only get, but no set methods
- read access is forbidden, if attribute is a container
- containers handed over to constructor have to be cloned
- final keyword prevents inheritance

1:1 Association

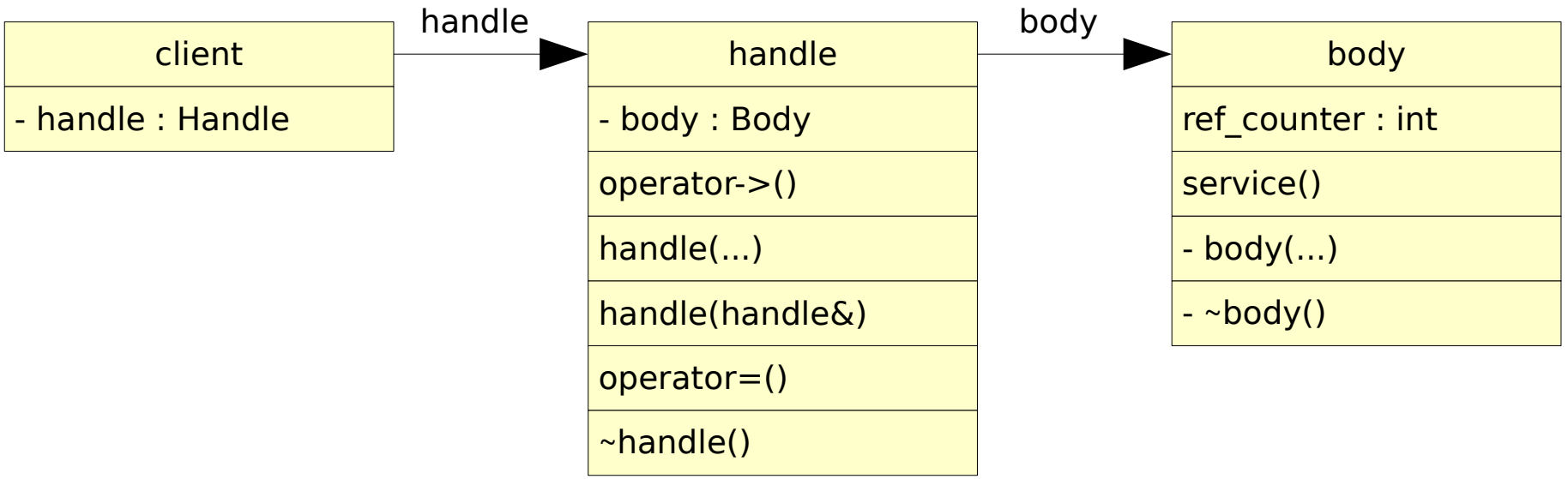
Delegation



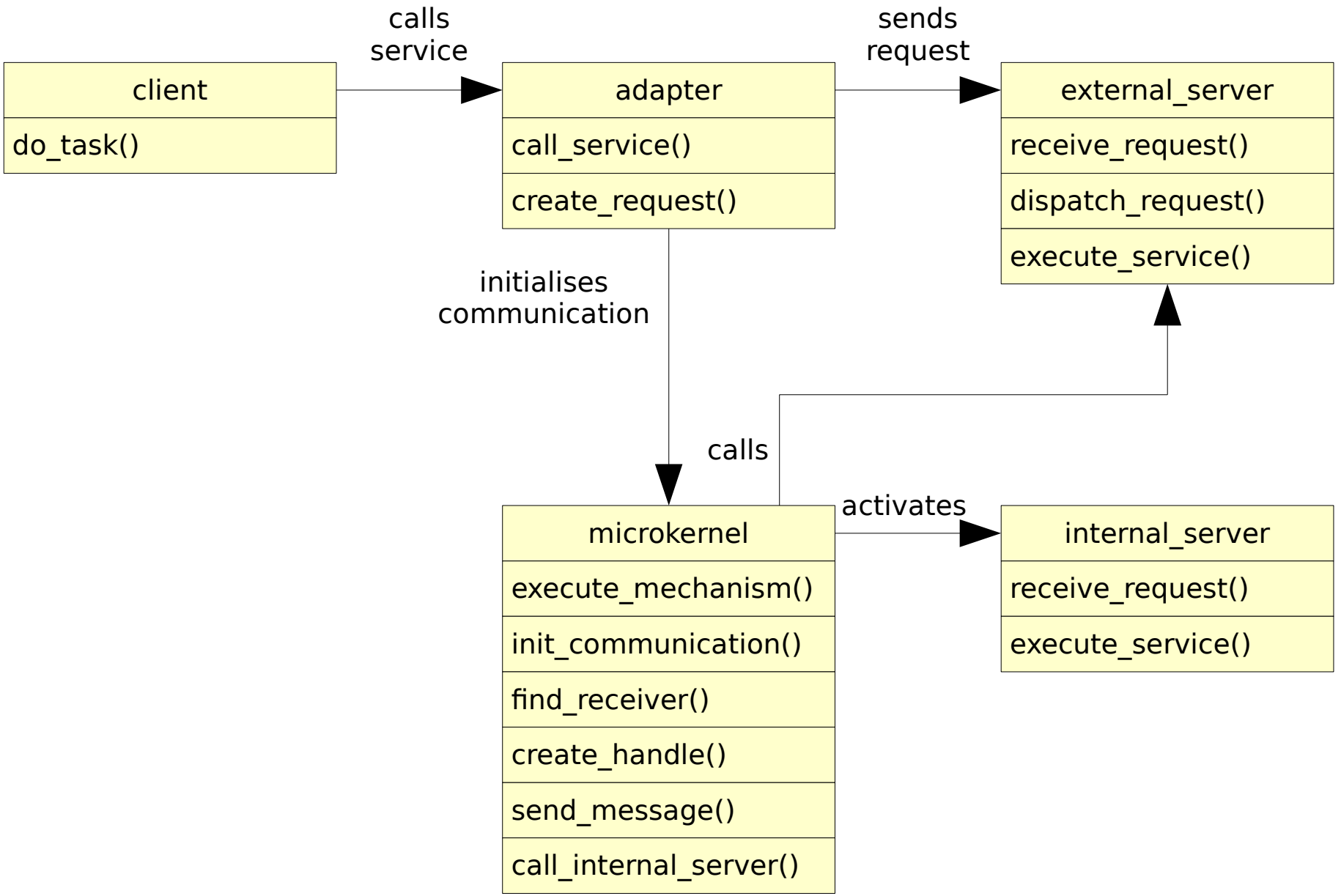
Wrapper [Umhüller] / Adapter (Gamma)



Bridge / Pointer / Handle-Body / Wrapper Object

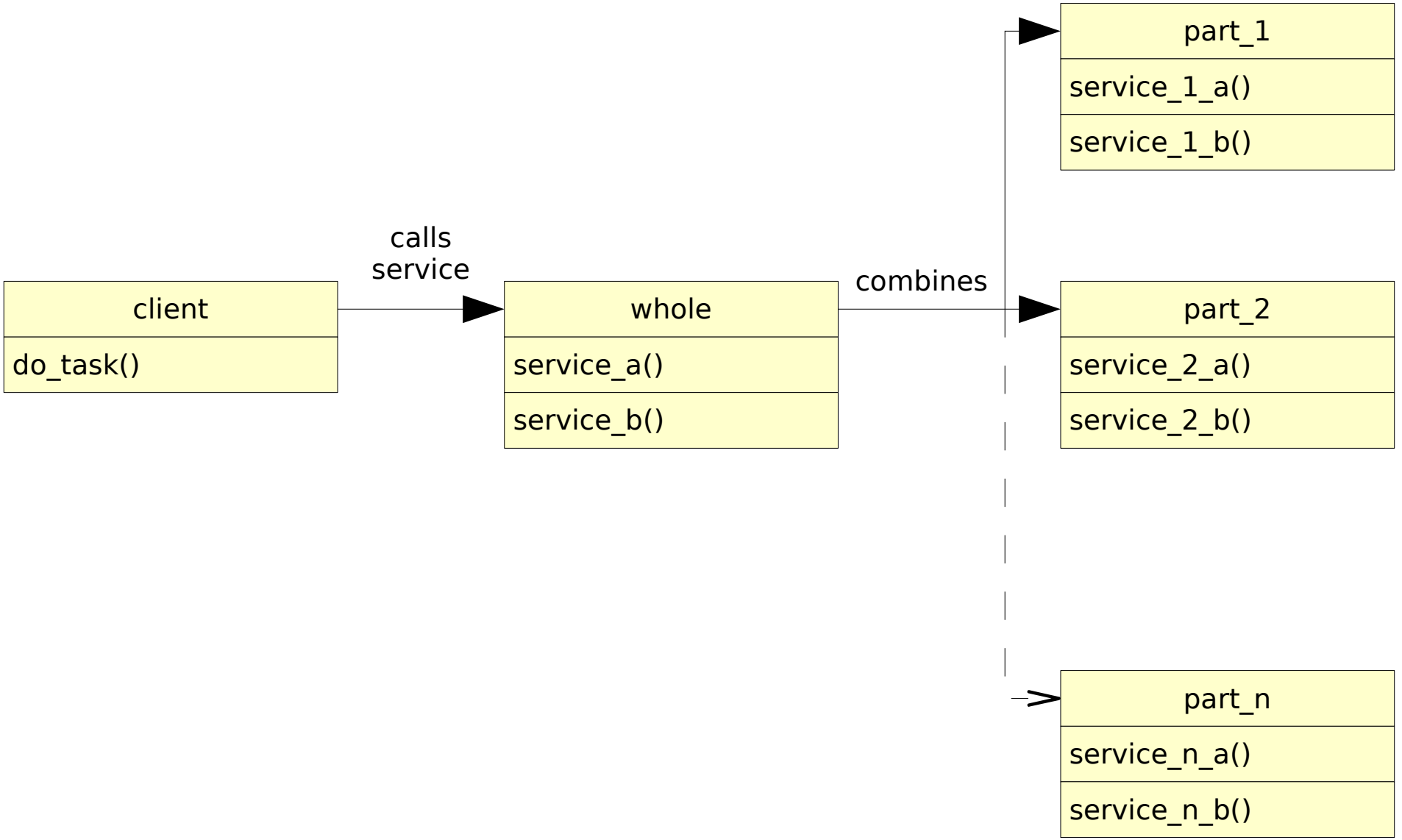


Microkernel

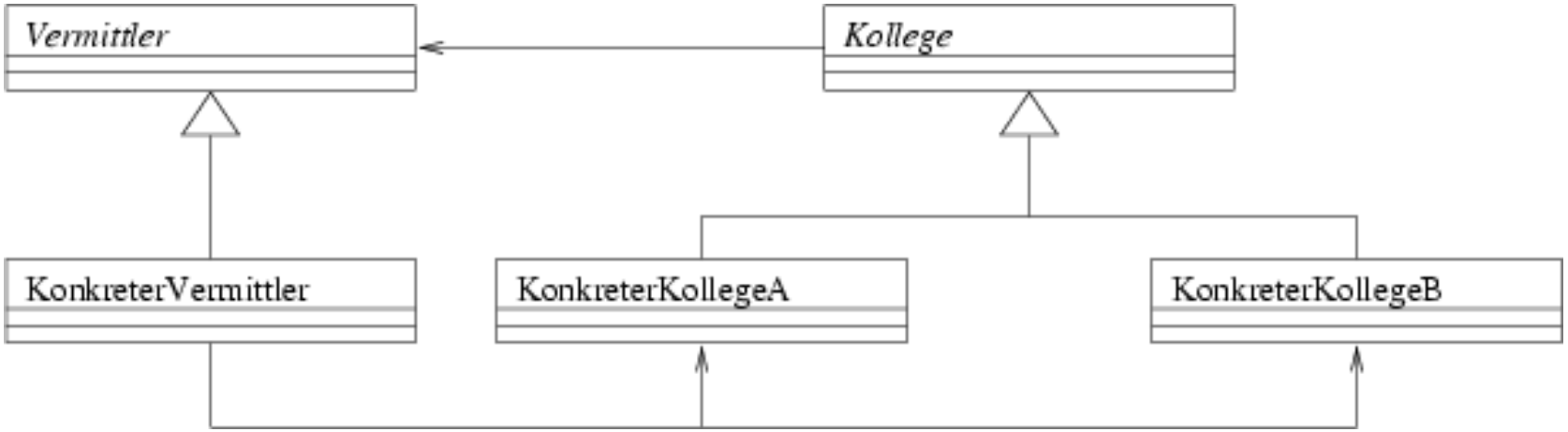


1:n Association

Whole Part


part_1
service_1_a()
service_1_b()
calls
service
client
do_task()
whole
service_a()
service_b()
combines
part_2
service_2_a()
service_2_b()
part_n
service_n_a()
service_n_b()

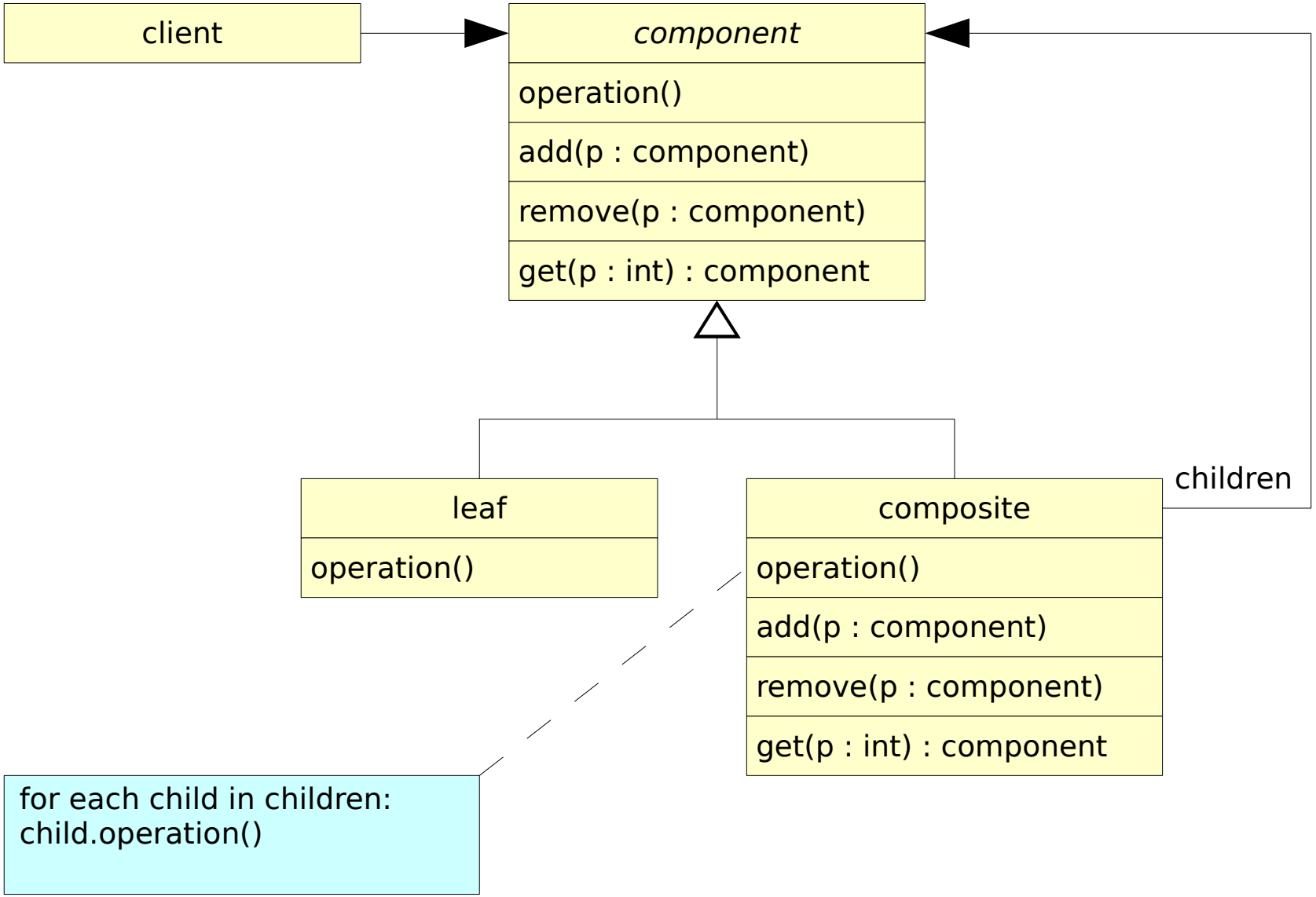
Mediator [Vermittler] (Gamma) / Broker



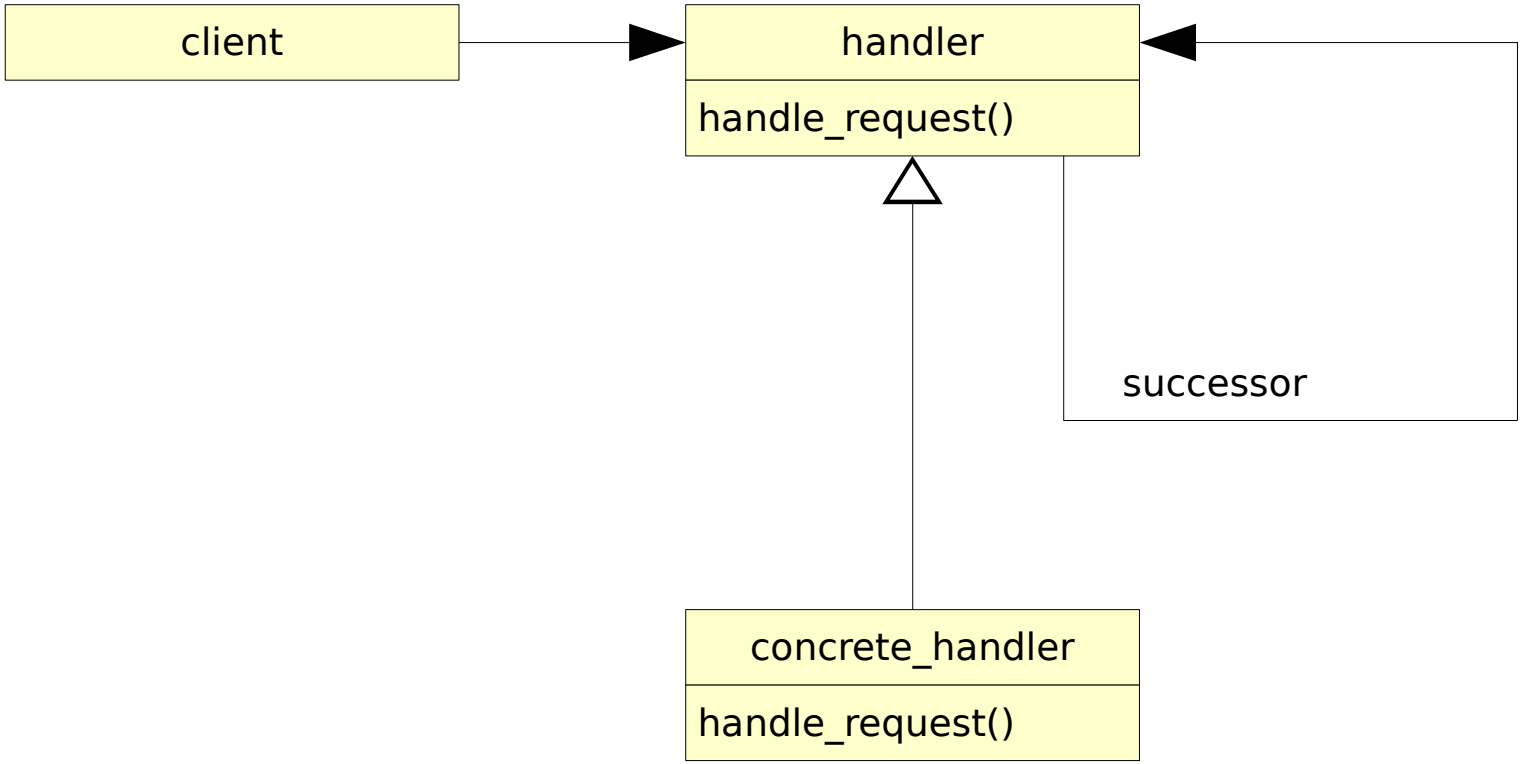
Wikipedia. http://www.wikipedia.org/

Recursion

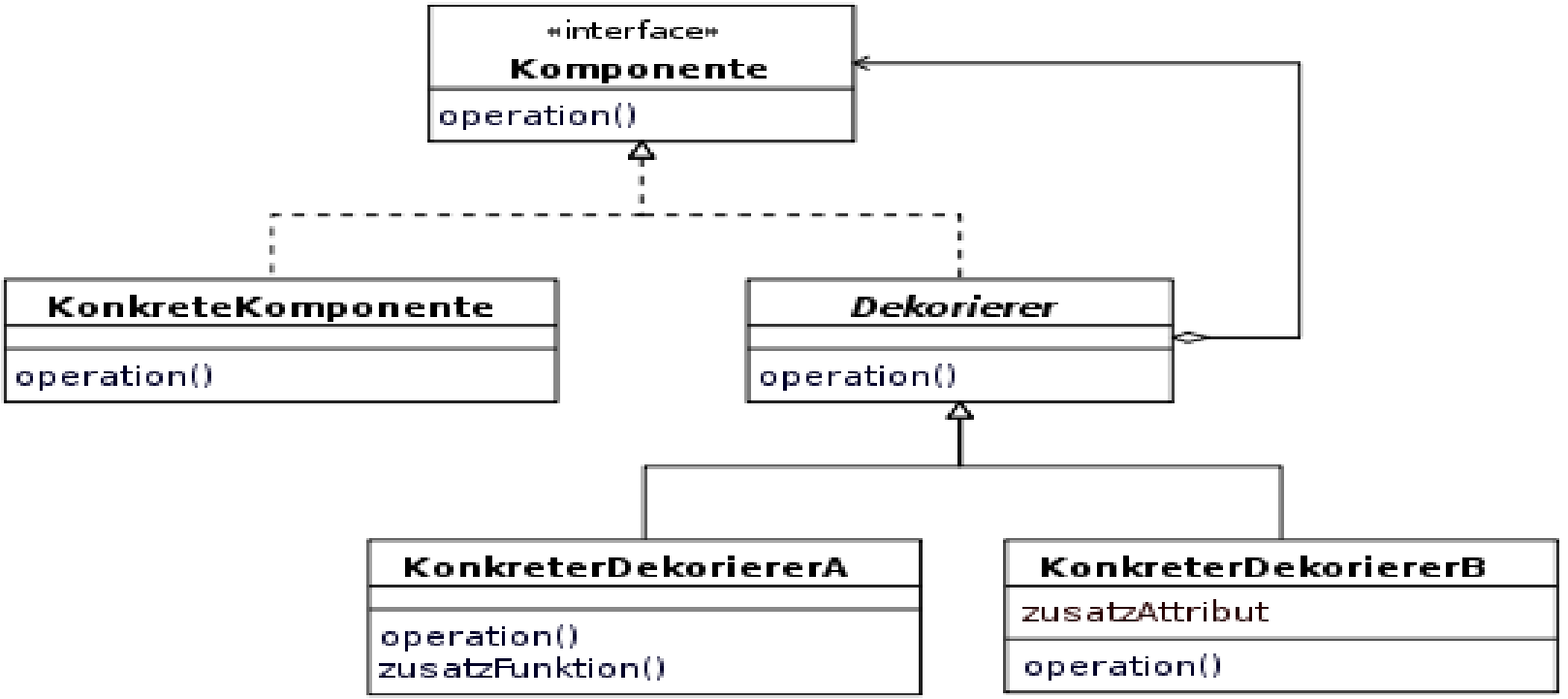
Composite (Gamma)



Chain of Responsibility / Event Handler / Bureaucrat / Responder



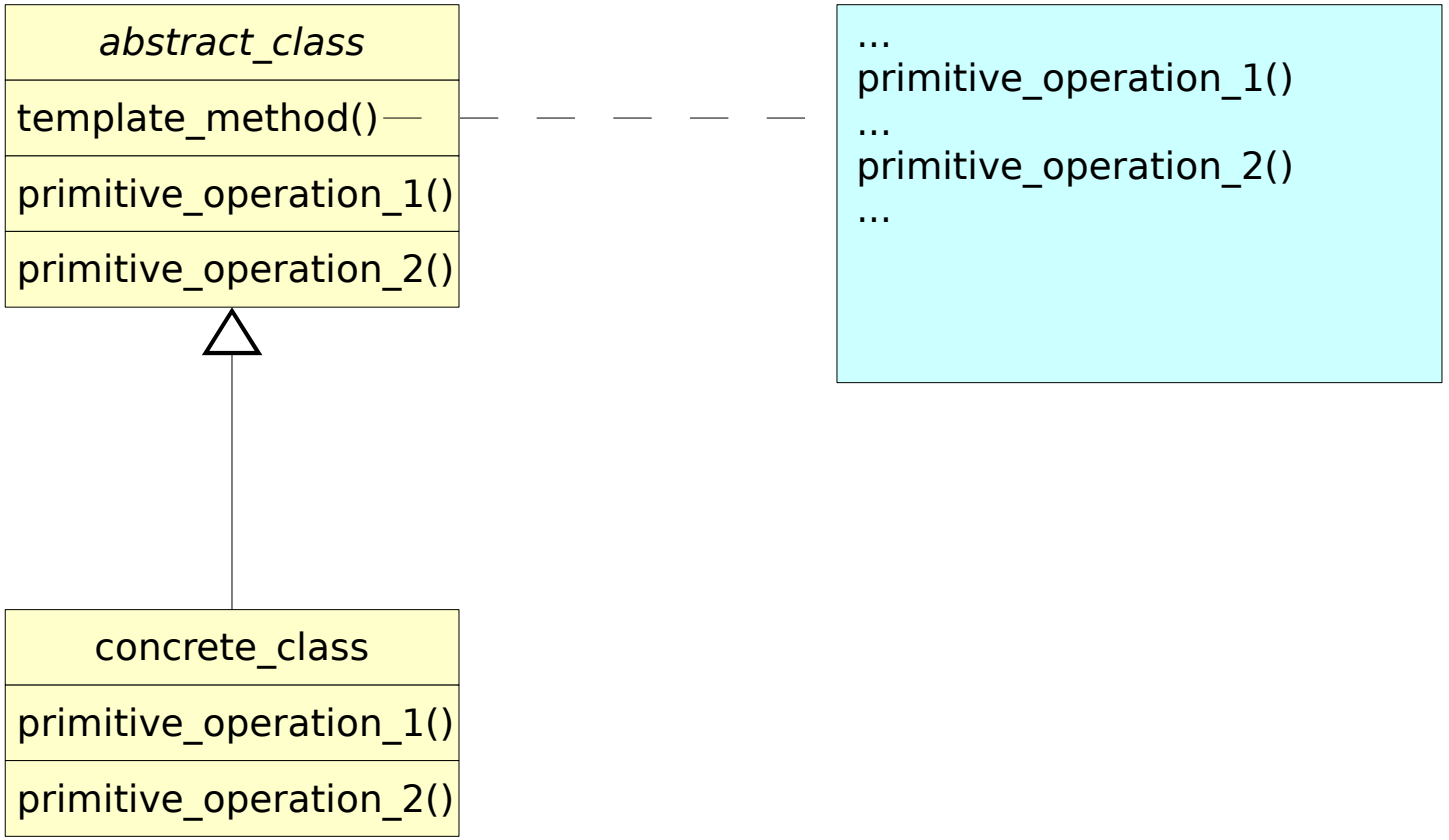
Decorator [Dekorierer] (Gamma)



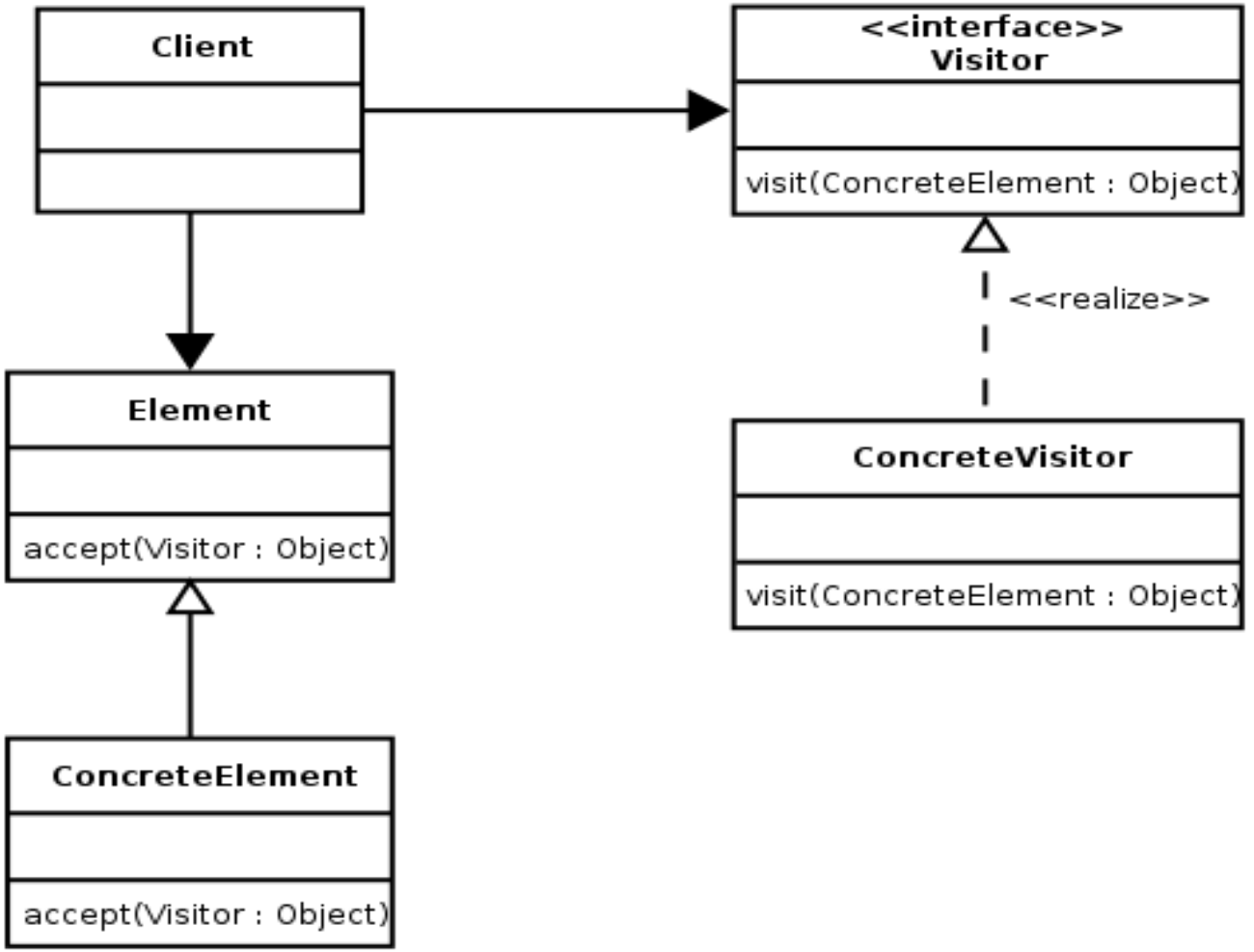
Wikipedia. http://www.wikipedia.org/

Polymorphism

Template Method / Hook Method

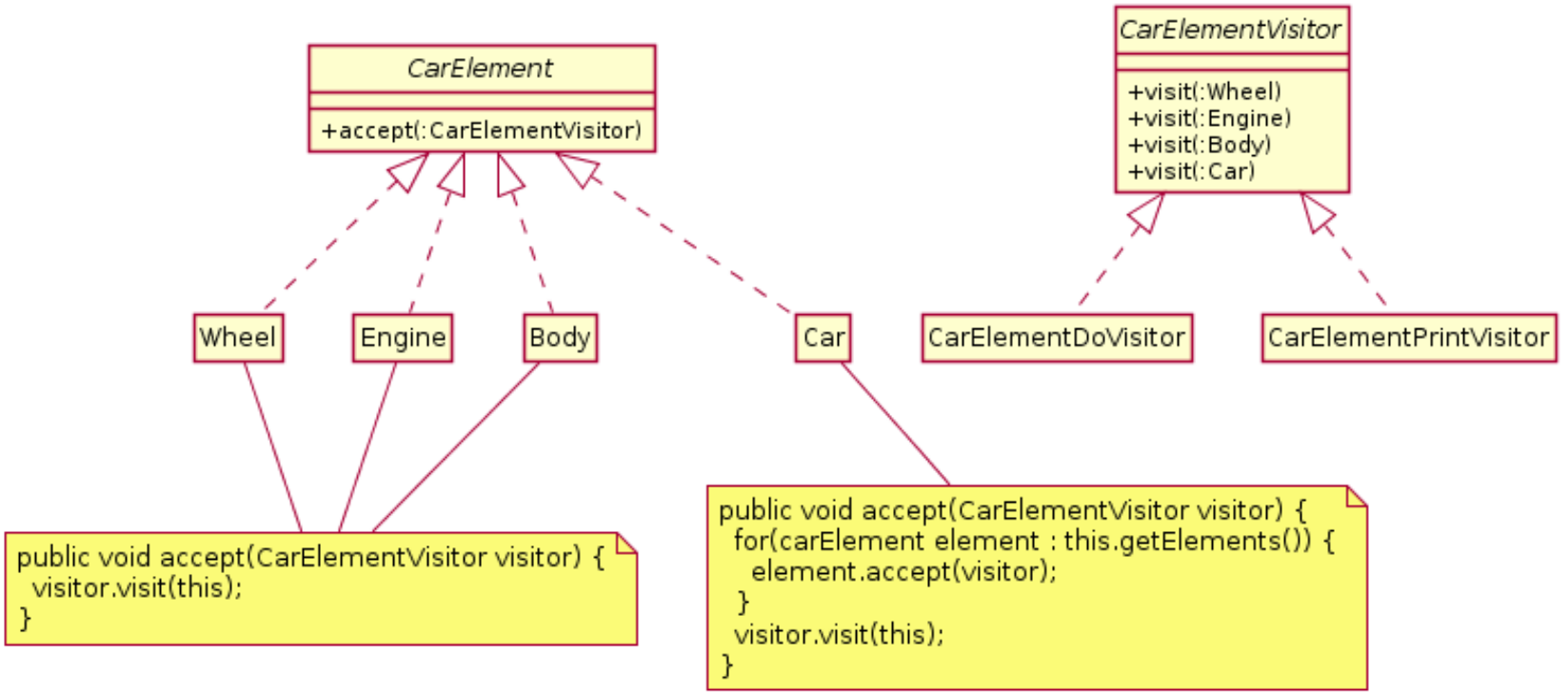


Visitor [Besucher] (Gamma)



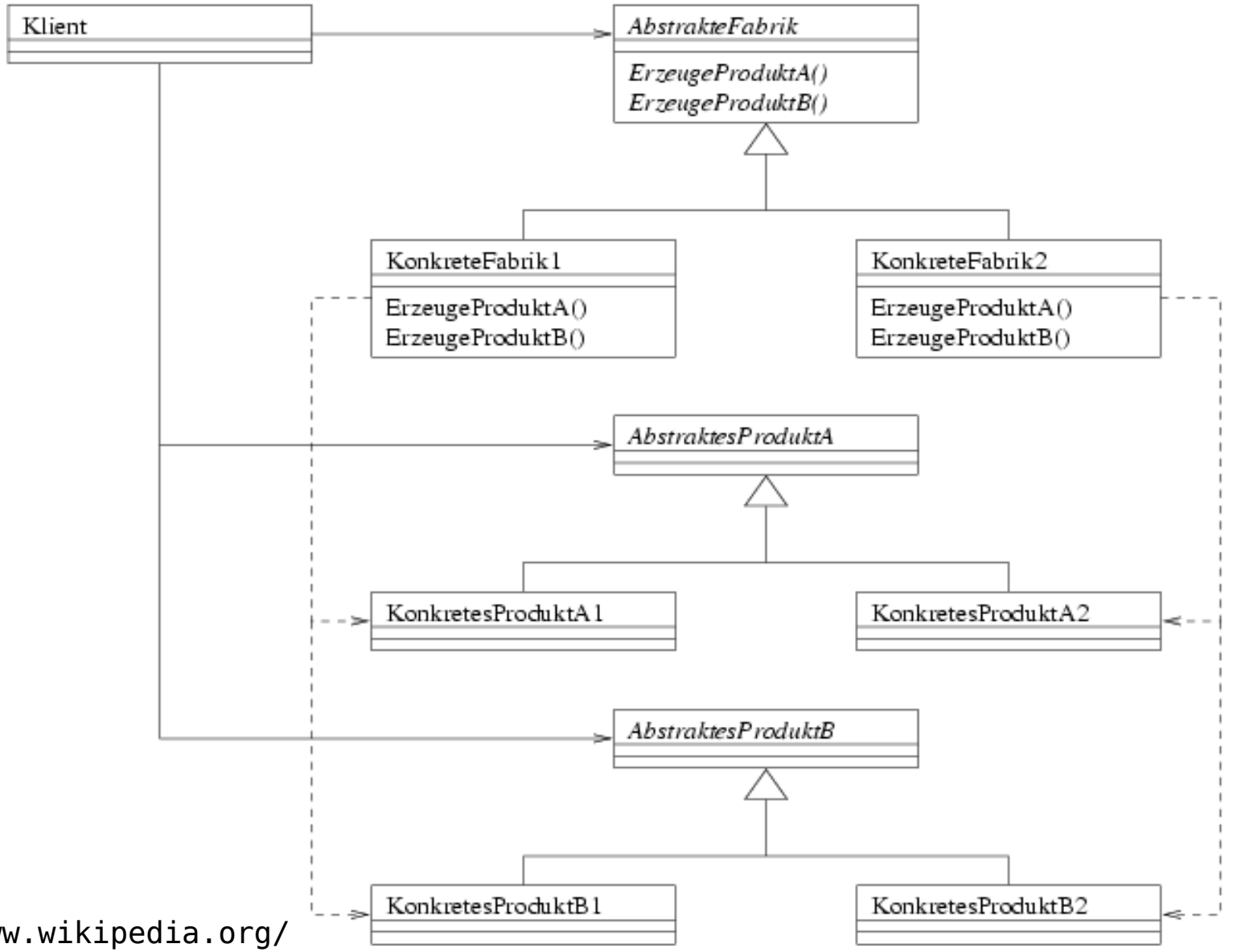
Wikipedia. http://www.wikipedia.org/

Visitor [Besucher] Example



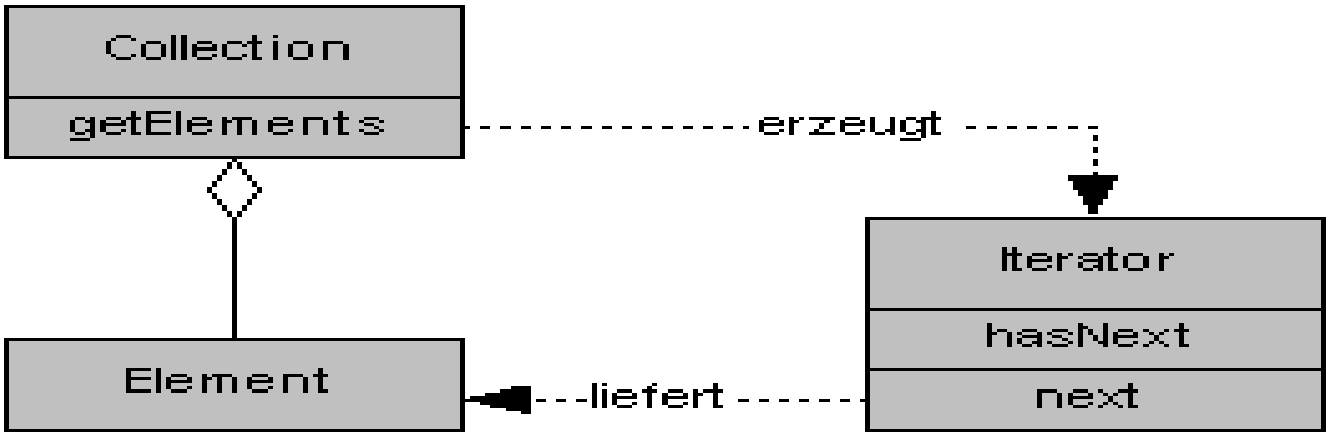
Wikipedia. http://www.wikipedia.org/

Abstract Factory (Gamma)



Wikipedia. http://www.wikipedia.org/

Iterator


Collection
getElements
erzeugt
Iterator
hasNext
next
Element
liefert

Interface

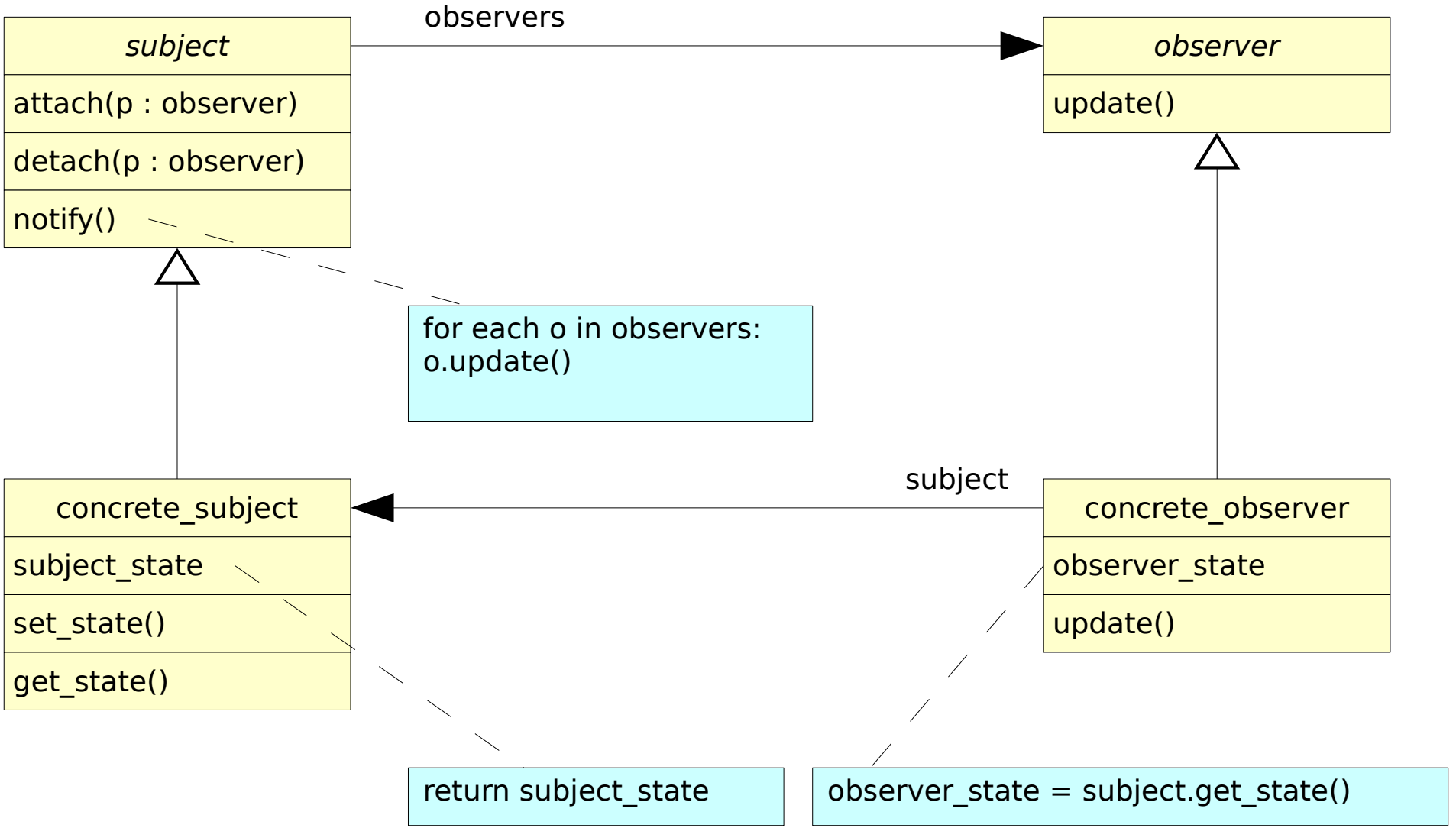
- refers to an abstraction that an entity provides of itself to the outside
- separates methods of external communication from internal operation
- allows methods to be internally modified without affecting the way outside entities interact with it
- provides multiple abstractions of itself

Inversion of Control (IoC)/Dependency Injection (DI)

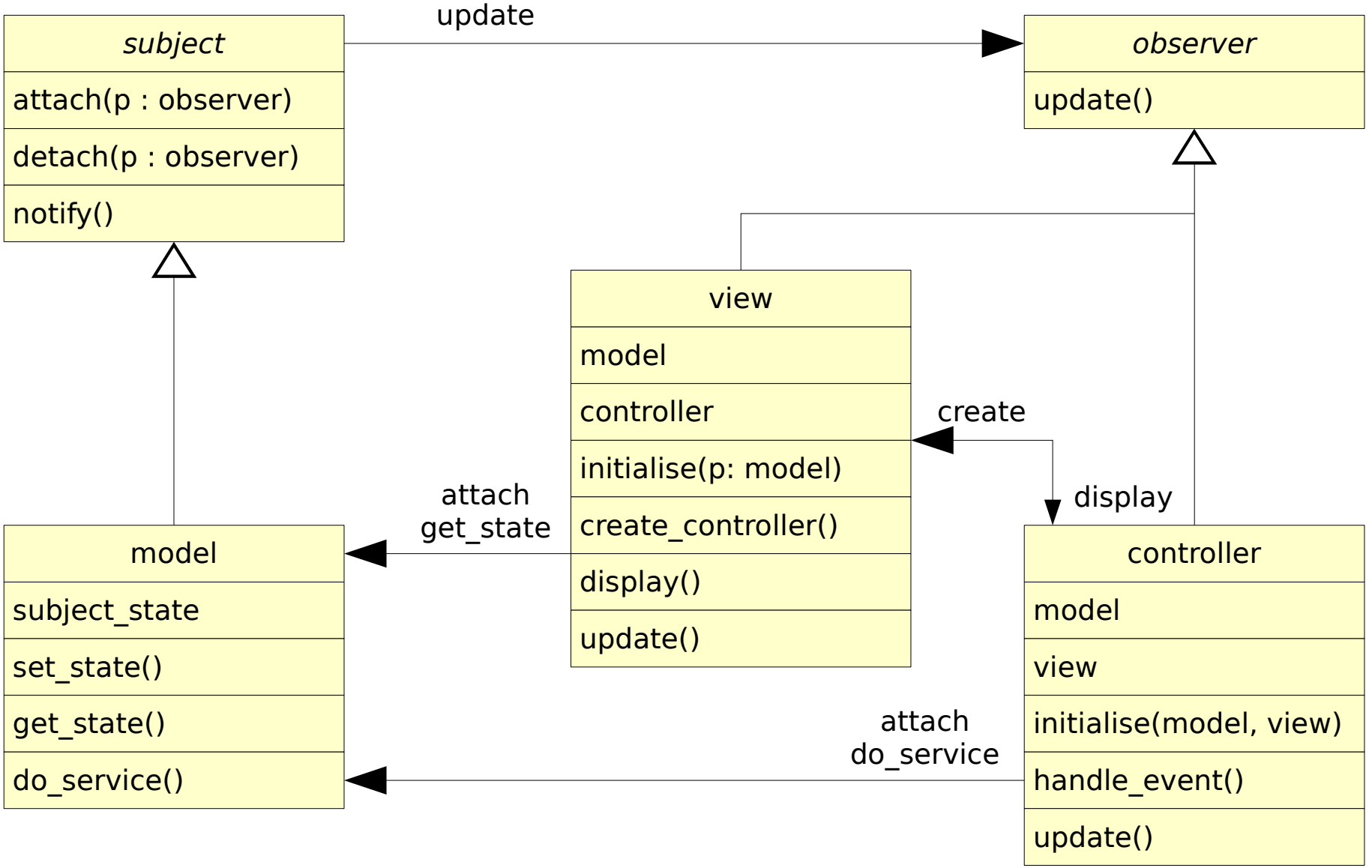
TODO

Bidirectionalism

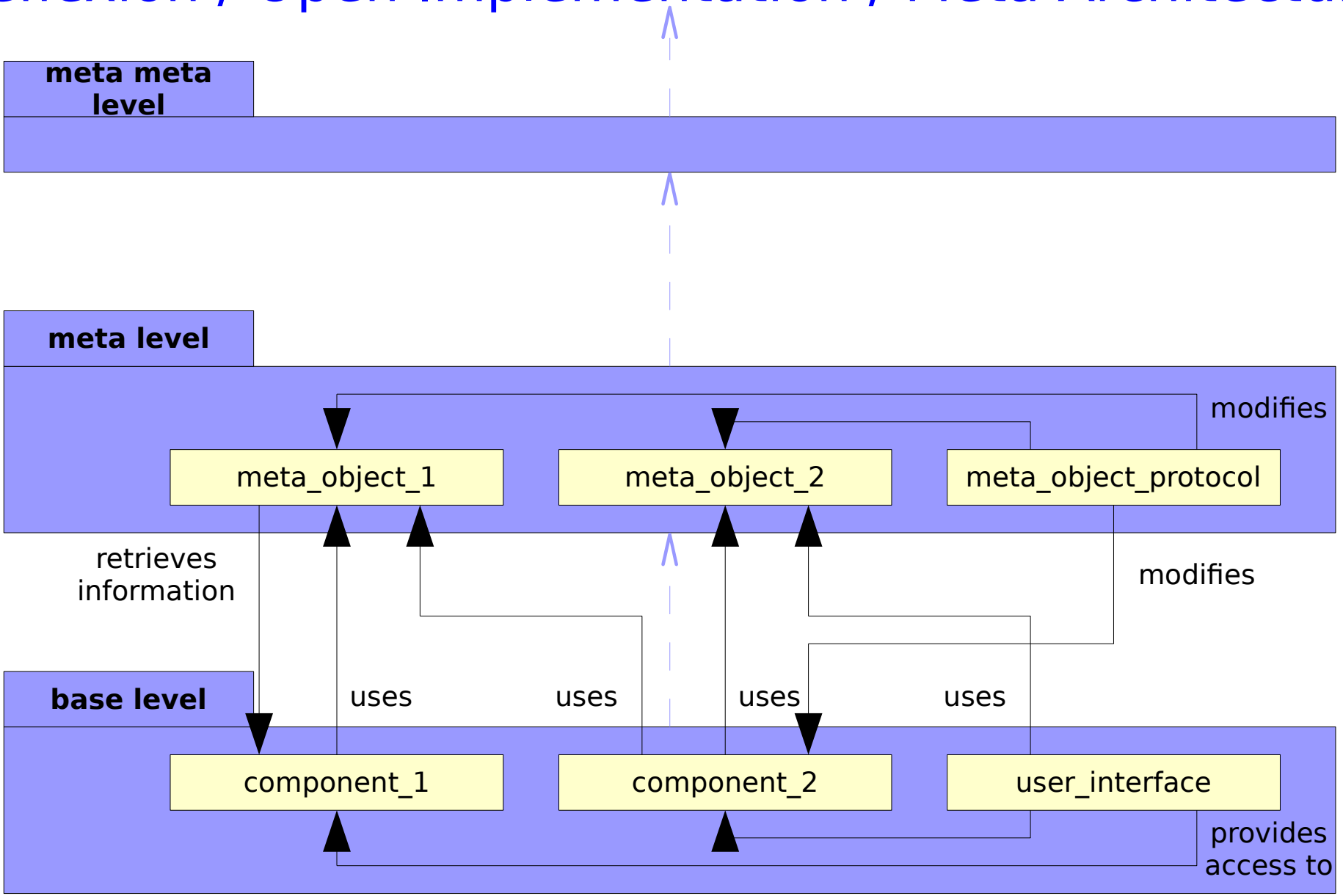
Observer (Gamma) / Publisher-Subscriber / Callback



MVC using Observer (Gamma)



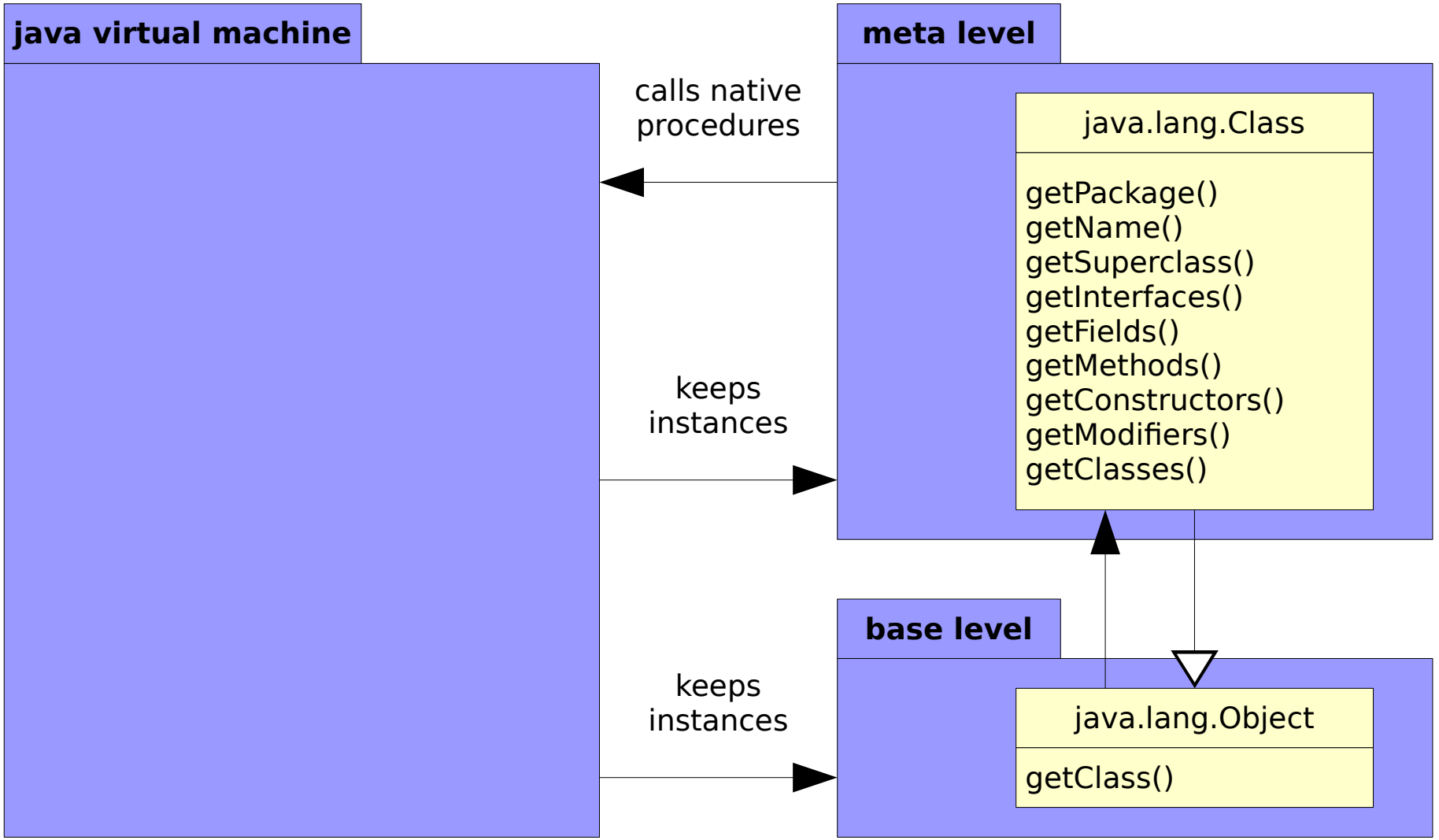
Reflexion / Open Implementation / Meta Architecture



Java Type System

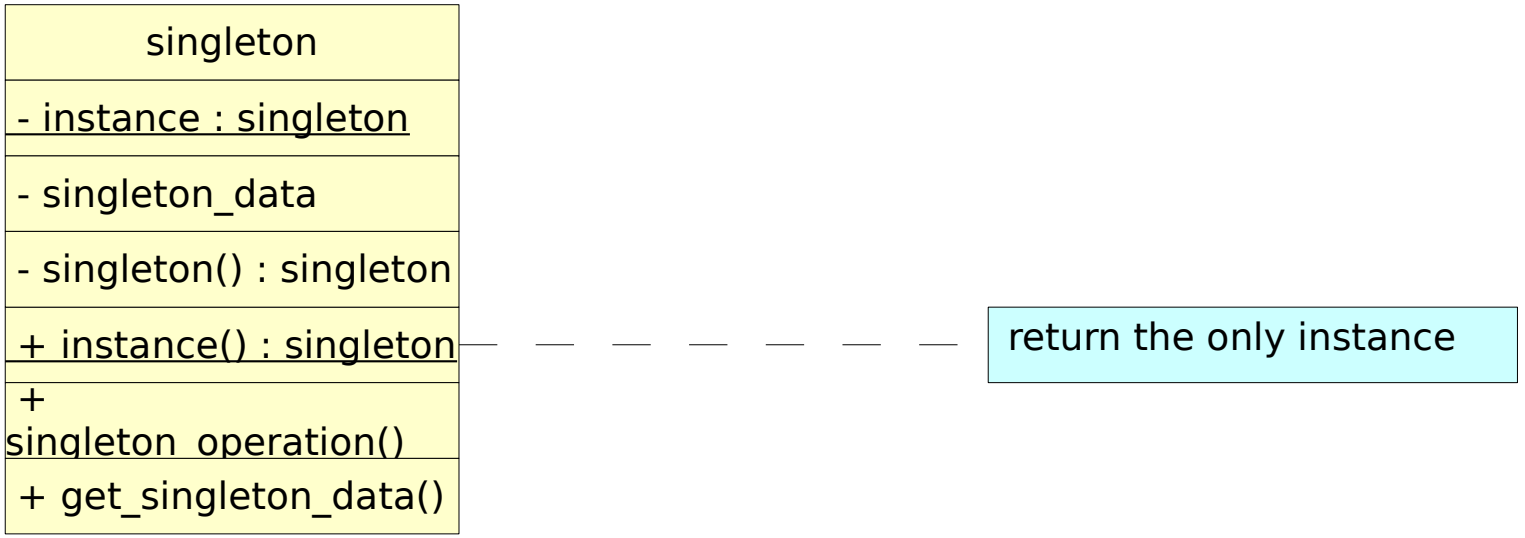
runtime dynamics

design time statics



Global Access

Singleton


singleton
- instance : singleton
- singleton_data
- singleton() : singleton
+ instance() : singleton
+ singleton_operation()
+ get_singleton_data()
return the only instance

Special Solution

Closure

```
// Outer function
var f1 = function() {
    // Block spans a namespace
    var wert = 22;
    // Inner function
    var f2 = function () {
        // Return value of variable belonging to outer namespace
        return wert;
    }
    // Return f2 durch f1 zurückgeben, womit f2 zum closure wird.
    return f2;
}

// a ist die von f1() zurückgegebene closure-Funktion
var a = f1();

console.log(f1()); // in other words: function() { return wert; }
console.log(typeof wert); // undefined
console.log(a()); // returns 22
console.log(f1()()); // returns 22, but f2() is not accessible here
```

https://de.wikipedia.org/wiki/Closure_%28Funktion%29#JavaScript