INF5120 and INF9120 ”Modelbased System development”

Lecture 2: 23.01.2017

Arne-Jørgen Berre

arneb@ifi.uio.no and Arne.J.Berre@sintef.no

Course parts (16 lectures) - 2017

- January (1-3) (Introduction to Modeling, Business Architecture and the Smart Building project):
- 1-16/1: Introduction to INF5120
- 2-23/1: Modeling structure and behaviour (UML and UML 2.0 and metamodeling) - (establish Oblig groups)
- 3-30/1: WebRatio for Web Apps/Portals and Mobile Apps – and Entity/Class modeling – (Getting started with WebRatio)

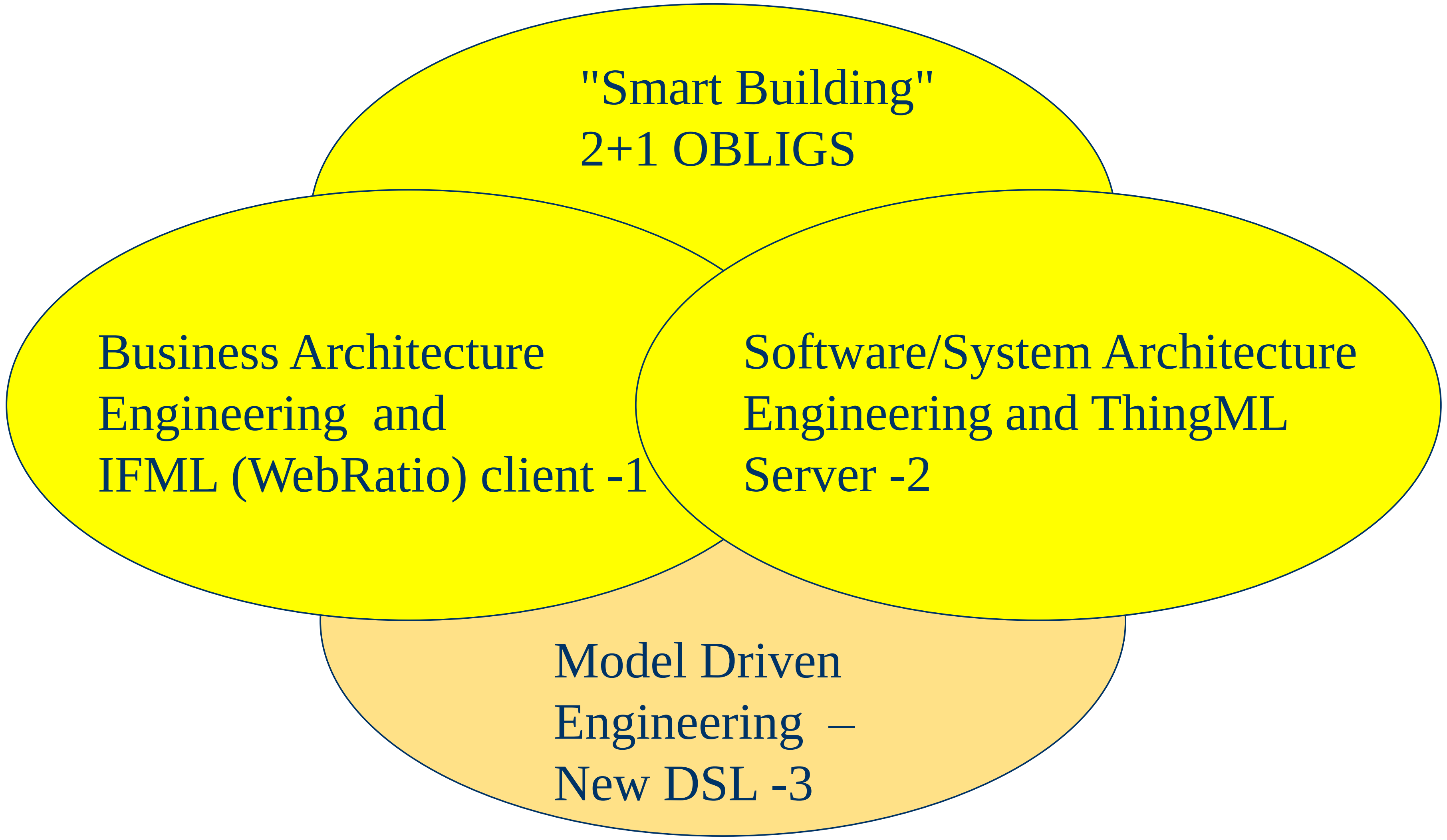
- February (4-7) (Modeling of User Interfaces, Flows and Data model diagrams, Apps/Web Portals - IFML/Client-Side):
- 4-6/2: Business Model Canvas, Value Proposition, Lean Canvas and Essence (Smart Building project) - User stories and Use case
- 5-13/2: IFML – Interaction Flow Modeling Language, WebRatio advanced – for Web and Apps
- 6-20/2: BPMN process, UML Activ.Diagrams, Workflow and Orchestration modelling value networks
- 7-27/2: Modeling principles – Quality in Models
- 27/2: Oblig 1: Smart Building – Business Architecture and App/Portal with IFML WebRatio UI for Smart Building

- March (8-11) (Modeling of IoT/CPS/Cloud, Services and Big Data – UML SM/SD/Collab, ThingML Server-Side):
- 8-6/3: DSL and ThingML, UML State Machines and Sequence Diagrams
- 9-13/3: UML Composite structures, State Machines and Sequence Diagrams II
- 10-20/3: Architectural models, Role modeling and UML Collaboration diagrams
- 11-27/3: UML Service Modeling, ServiceML,SoaML, REST, UML 2.0 Composition, MagicDraw
- 27/3: Oblig 2: Smart Building – Internet of Things control with ThingML – Raspberry Pi, Wireless sensors (temperature, humidity), actuators (power control)

- April/May (12-14) (MDE – Creating Your own Domain Specific Language):
- 12-3/4: Model driven engineering – Metamodels, DSL, UML Profiles, EMF, Sirius Editors
- EASTER – 10/4 og 17/4
- 13-24/4: MDE transformations, Non Functional requirements
- 1. Mai – Official holiday
- 14-8/5: Enterprise Architecture, TOGAF, UPDM, SysML – DSLs etc.
- 8/5: Oblig 3 - Your own Domain Specific Language

- May (15-17): (Bringing it together)
- 15-15/5: Summary of the course – Final demonstrations
- 16-22/5: Previous exams – group collaborations (No lecture)
- 17-29/5: Conclusions, Preparations for the Exam by old exams
- June (Exam)
- 13/6: Exam (4 hours), (June 13th, 0900)-1300

Course components


"Smart Building"
2+1 OBLIGS
Business Architecture Engineering and IFML (WebRatio) client -1
Software/System Architecture Engineering and ThingML Server -2
Model Driven Engineering – New DSL -3

This Lecture, January 23, 2017

Modeling structure and behaviour (UML and UML 2.0)

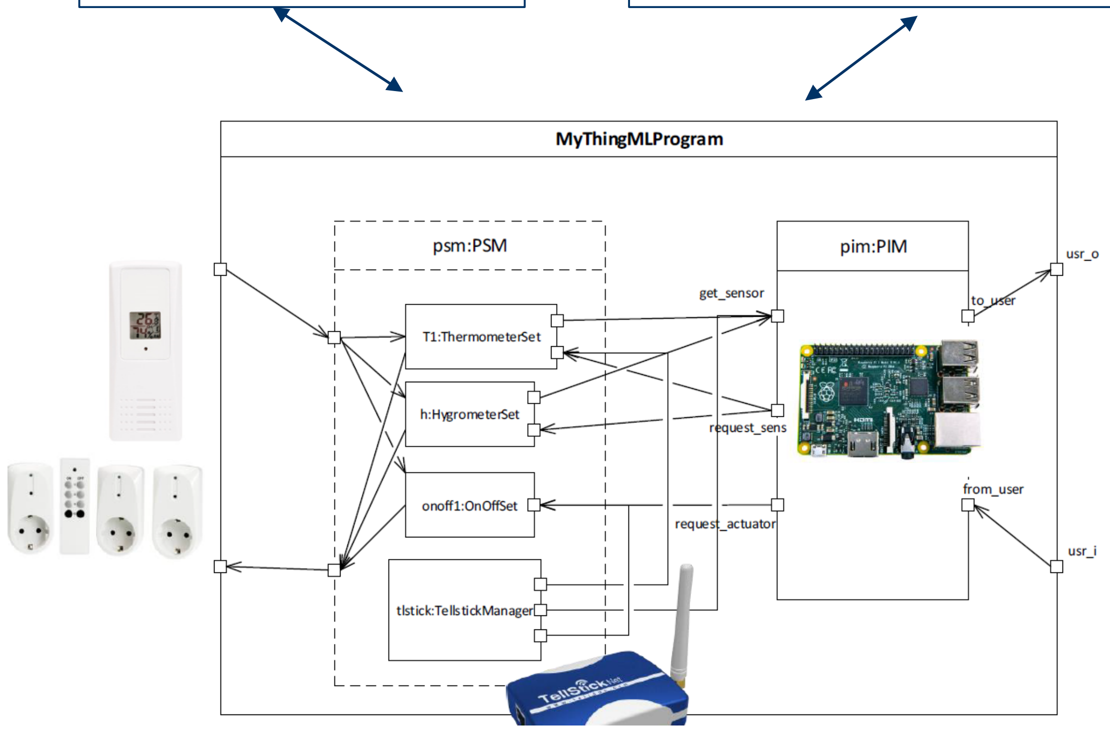
- Introduction to OBLIG, "Smart Building" with Web Portal and Mobile App development and control of Raspberry Pi with connected wireless sensors and actuators.

"Smart Building" - Project for spring 2017

Web app
User Interface
(Web portal)

SmartPhone
app
User Interface
(Android, iPhone)

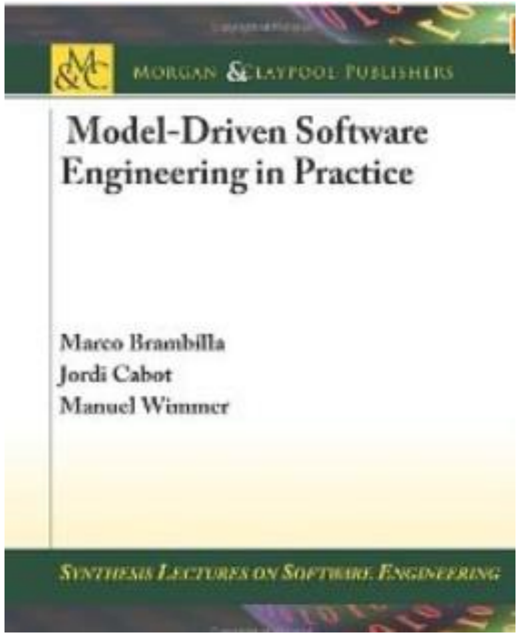
Using
IFML,
UML



Using
ThingML,
UML

Core Book on Model-Based system development

- Model-Driven Software Engineering in Practice
- ISBN 978-1-60845-882-0
- Morgan&Claypool Publishers, Synthesis lectures on Software Engineering
- 2012, 166 pages
- Marco Brambilla, Jordi Cabot and Manuel Wimmer


Morgan & Claypool Publishers

Model-Driven Software Engineering in Practice

Marco Brambilla
Jordi Cabot
Manuel Wimmer

Synthesis Lectures on Software Engineering

UML 2.0

- UML 2.0 and SysML Background and Reference material
- See www.uml-forum.com/specs.htm

- Also at OMG:
- http://www.omg.org/uml/ (UML)
- http://www.omg.org/mda/ (MDA)
- http://www.omg.org/cwm/ (MOF, XMI, CWM)

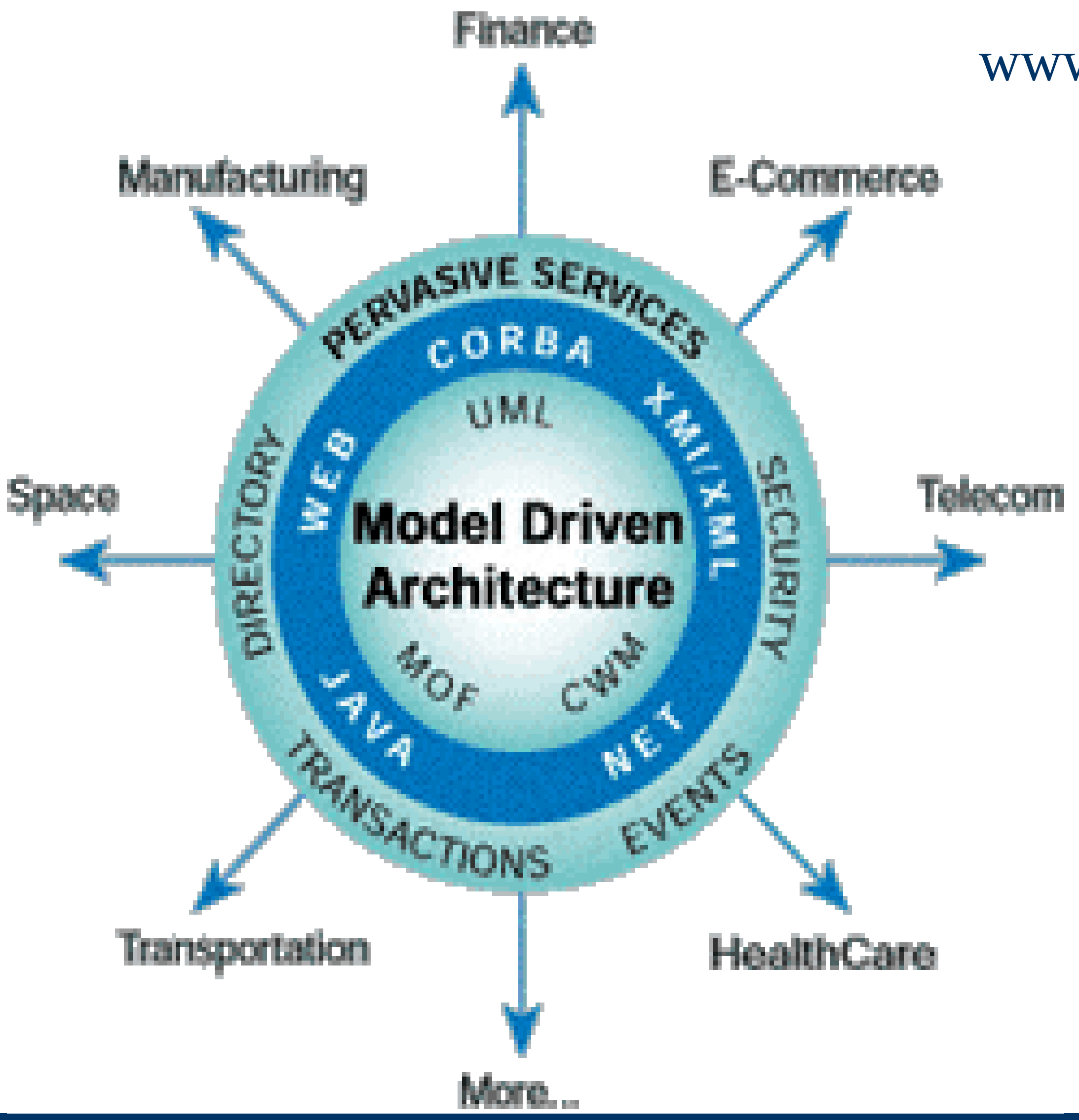
UML 2.0 recommended books:

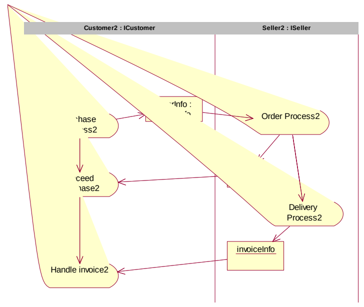
UML 2.0 in a Nutshell
by Dan Pilone (Author), Neil Pitman (Author)

The Unified Modeling Language User Guide
Second edition (ISBN 0-321-26797-4)
(G, Booch, J. Rumbaugh, Jacobsson)

OMG Model-Driven Architecture (MDA)

www.omg.org/mda



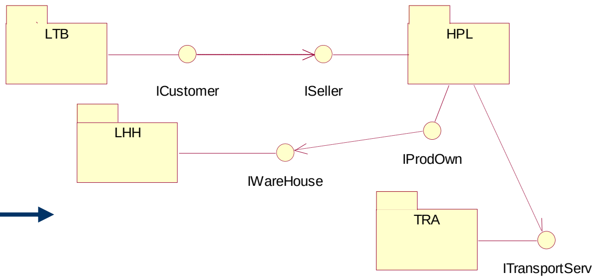


UML Activity model (or BPMN)

Activity model

Interaction model

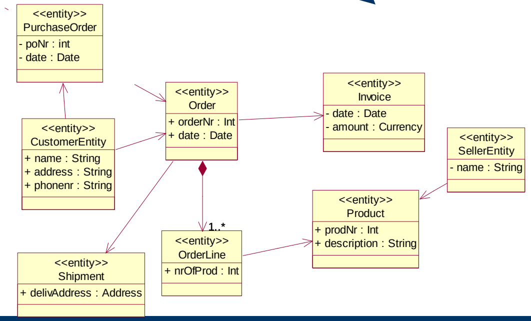
Information model



UML component diagram (enhanced in UML 2.0), SoaML

Information modeling

Semantic Models

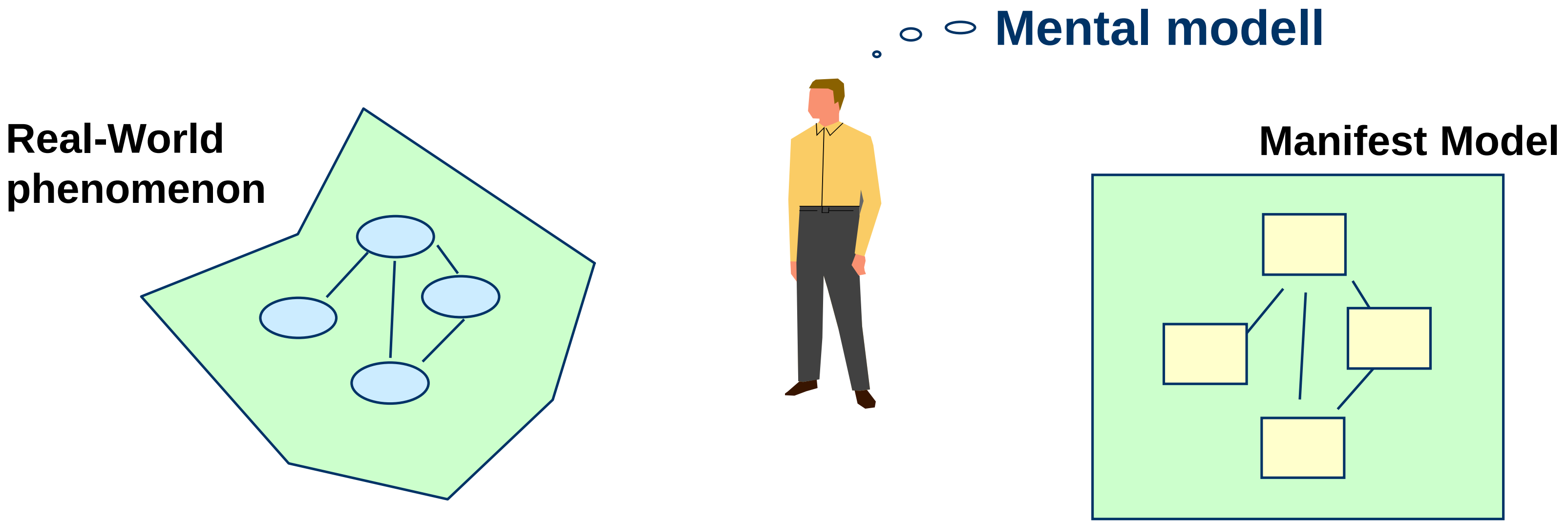


UML Class diagram

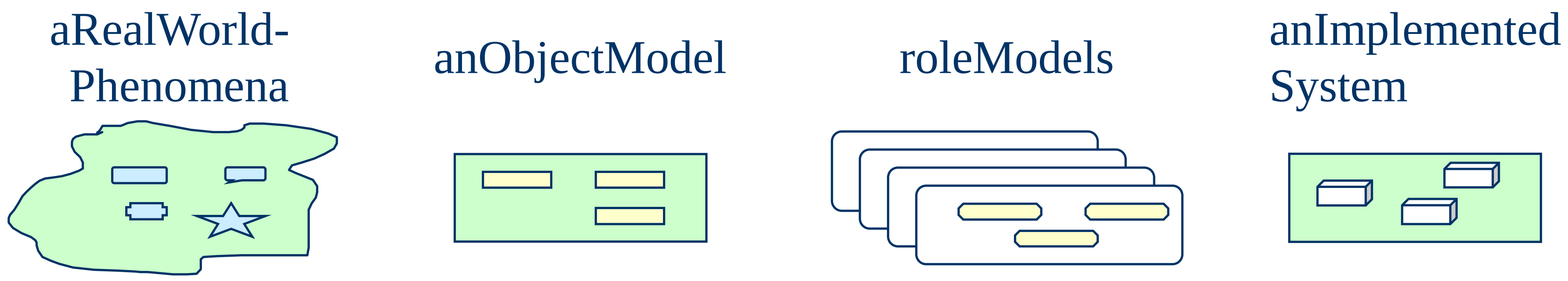
System and objects

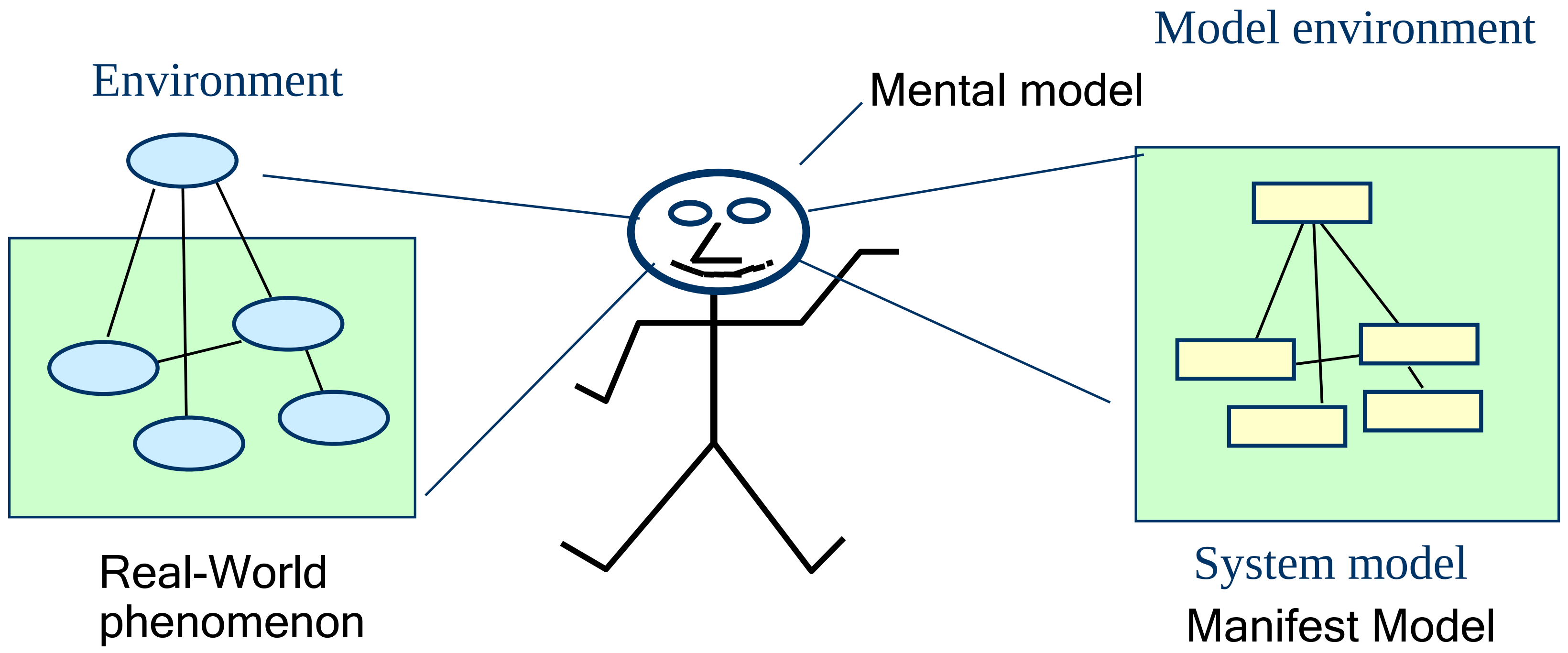
A *system* is a part of the real world which we choose to regard as a whole, separated from the rest of the world during some period of consideration.

A whole that we choose to consider as a collection of objects, each *object* being characterized by *attributes* and by *actions* which may involve itself and other objects.



Object oriented modeling





OO Programming Terminology

- Encapsulation
- Object
- Message
- Method
- Class
- Instance
- Inheritance
- Polymorphism
- Dynamic (Late) Binding

CRC Method, class, responsibilities, and collaborators

- Method to learn
  the most basic OO concepts plus OO “thinking”
  - “The most effective way of teaching the idiomatic way of thinking with objects is to immerse the learner in the "object-ness" of the material. To do this we must remove as much familiar material as possible, expecting that details such as syntax and programming environment operation will be picked up quickly enough once the fundamentals have been thoroughly understood.”
- Technique also very useful
  during informal and creative analysis and design
- Created by Kent Beck and Ward Cunningham,
  Textronix, 1989

The CRC-Card
an object of paper personalizing the object

Class (Name):	
Responsibility:	Collaborators:

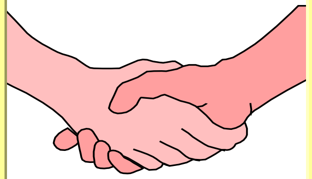
Class, responsibilities, and collaborators

- **Class**
  The class name of an object creates a vocabulary for discussing a design. Indeed, many people have remarked that object design has more in common with language design than with procedural program design. We urge learners (and spend considerable time ourselves while designing) to find just the right set of words to describe our objects, a set that is internally consistent and evocative in the context of the larger design environment.
- **Responsibilities**
  Responsibilities identify problems to be solved. The solutions will exist in many versions and refinements. A responsibility serves as a handle for discussing potential solutions. The responsibilities of an object are expressed by a handful of short verb phrases, each containing an active verb. The more that can be expressed by these phrases, the more powerful and concise the design. Again, searching for just the right words is a valuable use of time while designing.
- **Collaborators**
  Objects which will send or be sent messages in the course of satisfying responsibilities. Collaboration is not necessarily a symmetric relation. For example in Smalltalk, View and Controller operate as near equals while OrderedCollection offers a service with little regard or even awareness of its client.

UML og (R)UP

Two parts of a Harmonized Whole

Unified Modeling Language



Process

- Convergence Today
- Unification
  leads to “standards”

- Convergence
  in the future
- Process frameworks
  through consensus
- Essence standard

UML Structural Modeling

- Class Diagram
- Object Diagram
- Component Diagram **(new in UML 2.0)**
- Package Diagram
- Deployment diagram

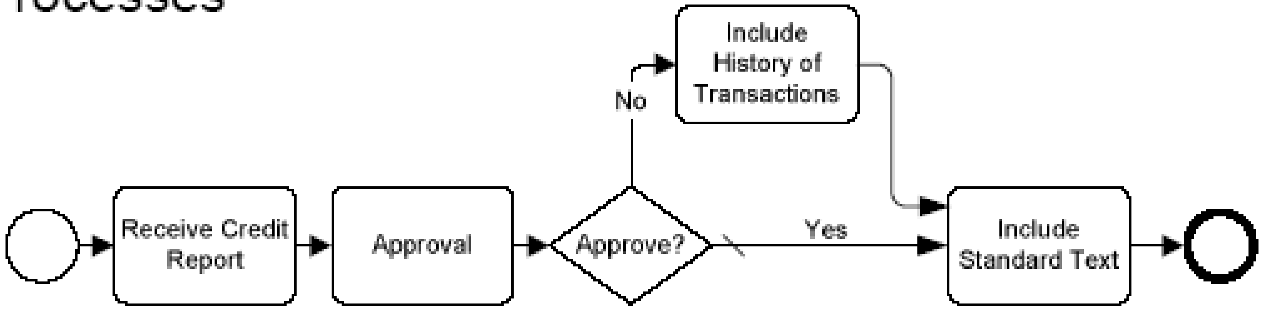
UML Behavioral Modelling

- **Use Case Diagrams**
- Interactions
  - Sequence diagrams **(enhanced in UML 2.0)**
  - Timing diagrams **(new in UML 2.0)**
  - Interaction overview diagrams **(new in UML 2.0)**
  - Communication diagrams (i.e. collaboration diagram)
- State machine diagrams **(enhanced in UML 2.0)**
- Activity Diagrams **(enhanced in UML 2.0)**

- BPMN 2.0

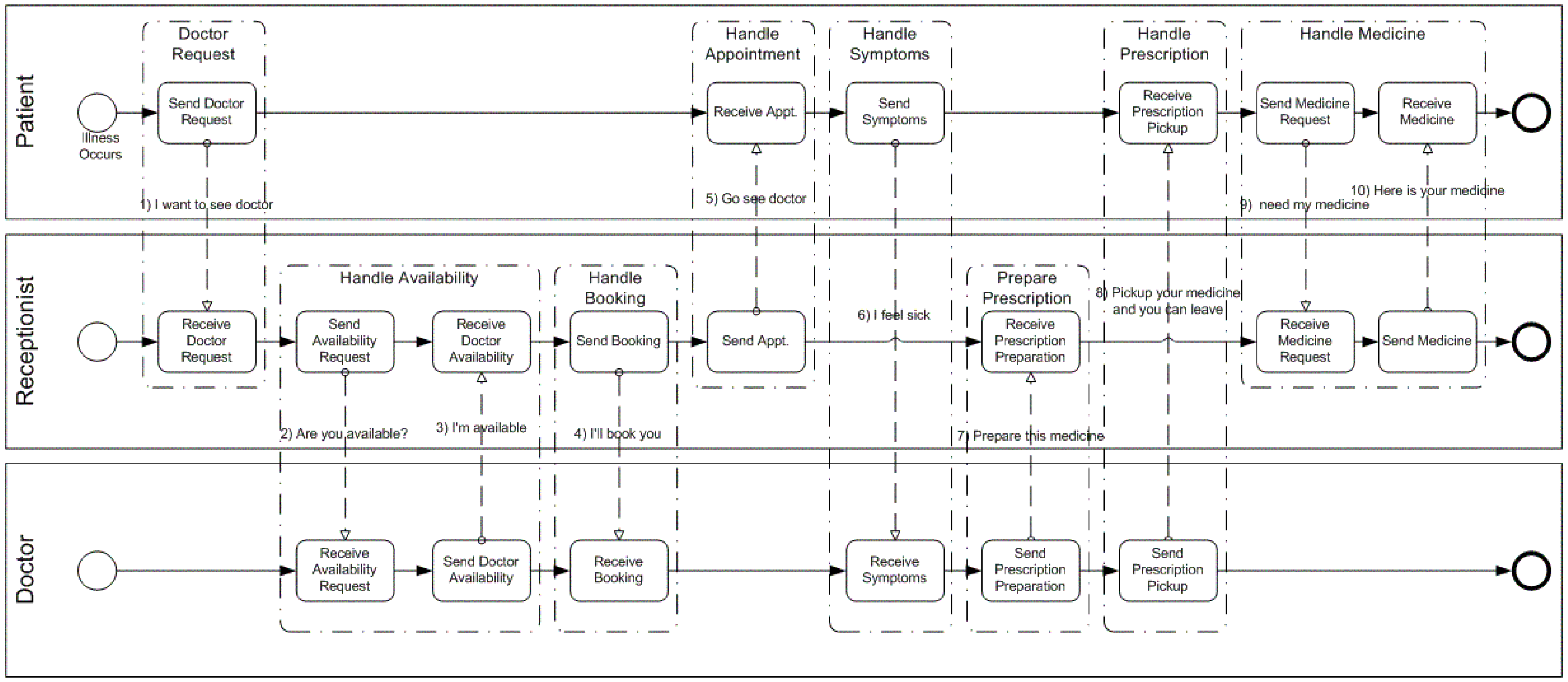
What is BPMN (Business Process Modeling Notation) ?

- BPMN is flow-chart based notation for defining Business Processes



- BPMN is an agreement between multiple modeling tools vendors, who had their own notations, to use a single notation for the benefit of end-user understand and training
- BPMN provides a mechanism to generate an executable Business Process (BPEL) from the business level notation
  - A Business Process developed by a business analyst can be directly applied to a BPM engine instead of going through *human* interpretations and translations into other languages

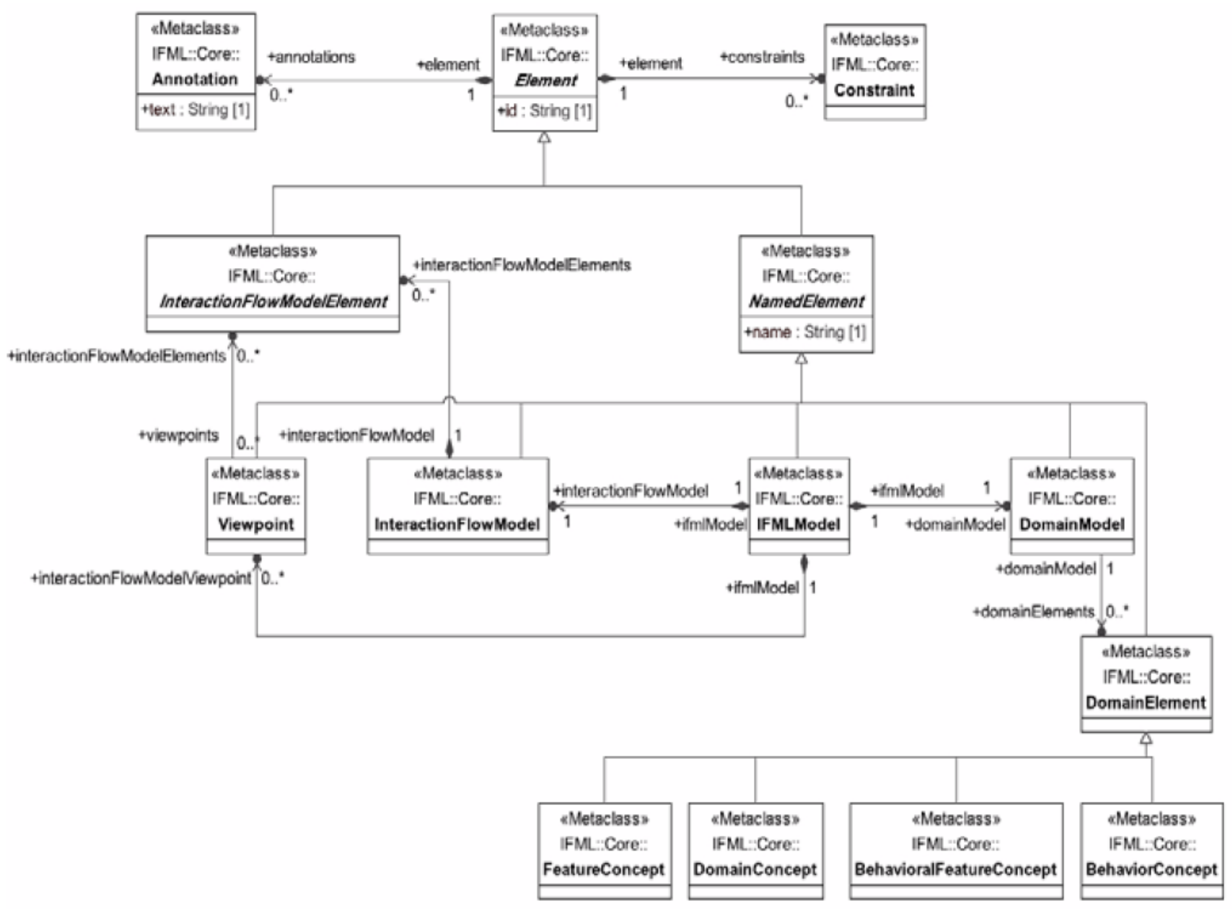
BPMN example



Different kind of models

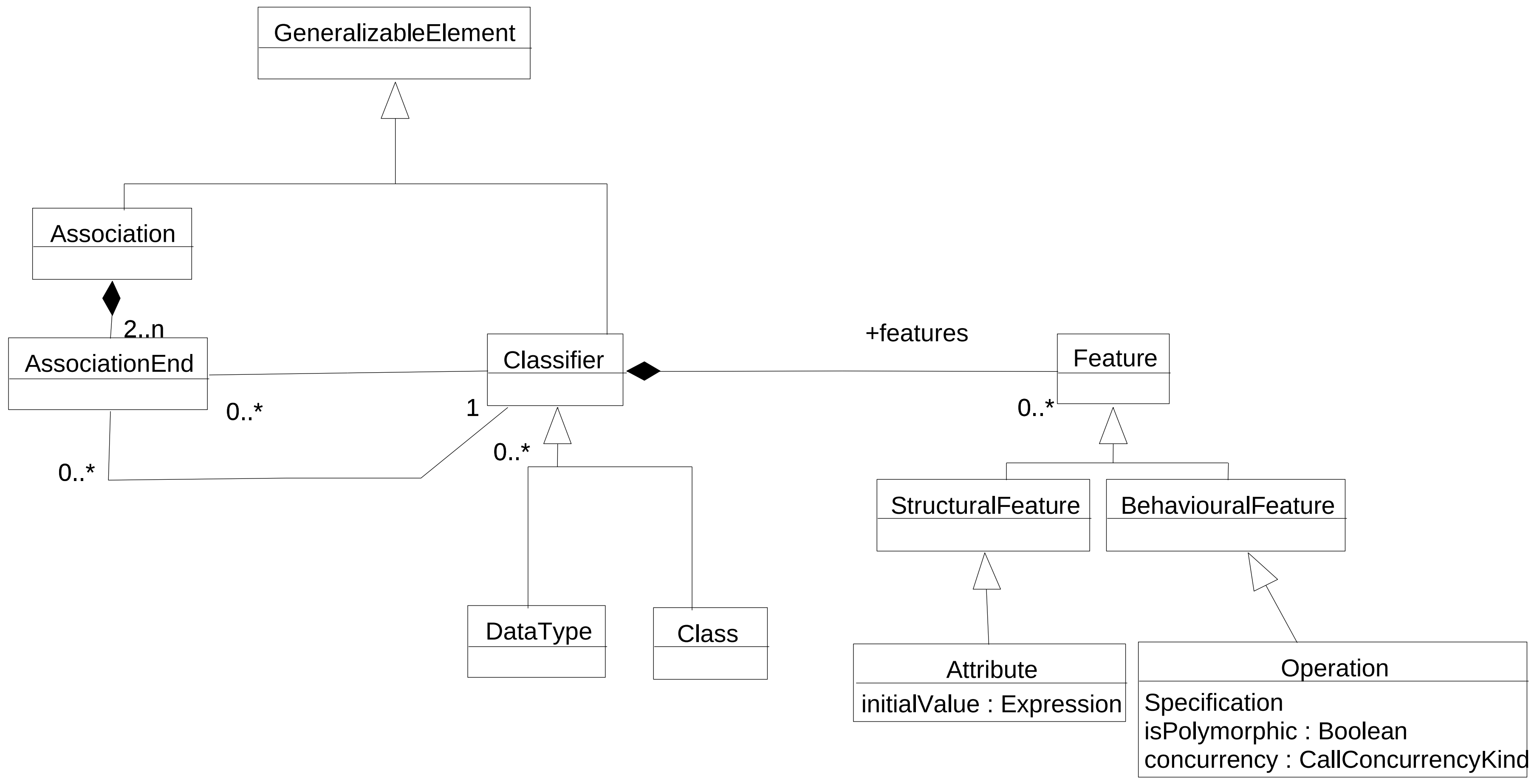
- Conceptual models
- Specification models
- Implementation models

Models and MetaModels


«Metaclass» IFML::Core:: Annotation
+text : String [1]
+annotations
0..*
+element
1
«Metaclass» IFML::Core:: Element
+id : String [1]
+element
1
+constraints
0..*
«Metaclass» IFML::Core:: Constraint
«Metaclass» IFML::Core:: InteractionFlowModelElement
+interactionFlowModelElements
0..*
«Metaclass» IFML::Core:: NamedElement
+name : String [1]
+interactionFlowModelElements
0..*
+viewpoints
0..*
+interactionFlowModel
1
«Metaclass» IFML::Core:: Viewpoint
«Metaclass» IFML::Core:: InteractionFlowModel
+interactionFlowModel
1
+ifmlModel
1
«Metaclass» IFML::Core:: IFMLModel
+ifmlModel
1
+domainModel
1
«Metaclass» IFML::Core:: DomainModel
+interactionFlowModelViewpoint
0..*
+ifmlModel
1
+domainModel
1
+domainElements
0..*
«Metaclass» IFML::Core:: DomainElement
«Metaclass» IFML::Core:: FeatureConcept
«Metaclass» IFML::Core:: DomainConcept
«Metaclass» IFML::Core:: BehavioralFeatureConcept
«Metaclass» IFML::Core:: BehaviorConcept

Parts of IFML Metamodel

Parts of UML Metamodel



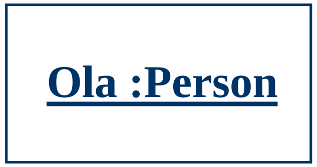
UML Information Modeling

- Ref also ISO 19103 Standard for Conceptual Modeling
- The following material is for reference …..

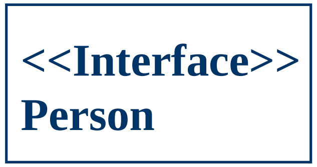
Objects

Can represents

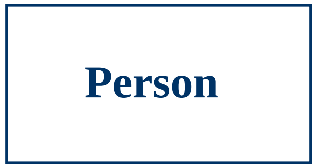
- One instance



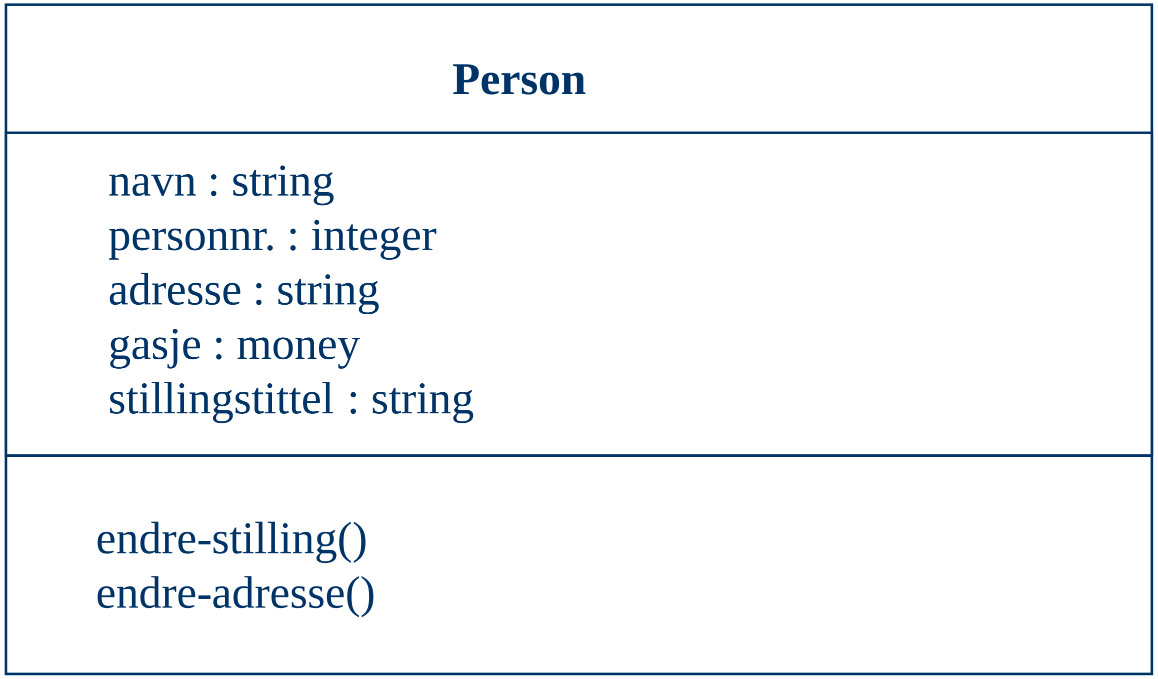
- One type, interface



- One class

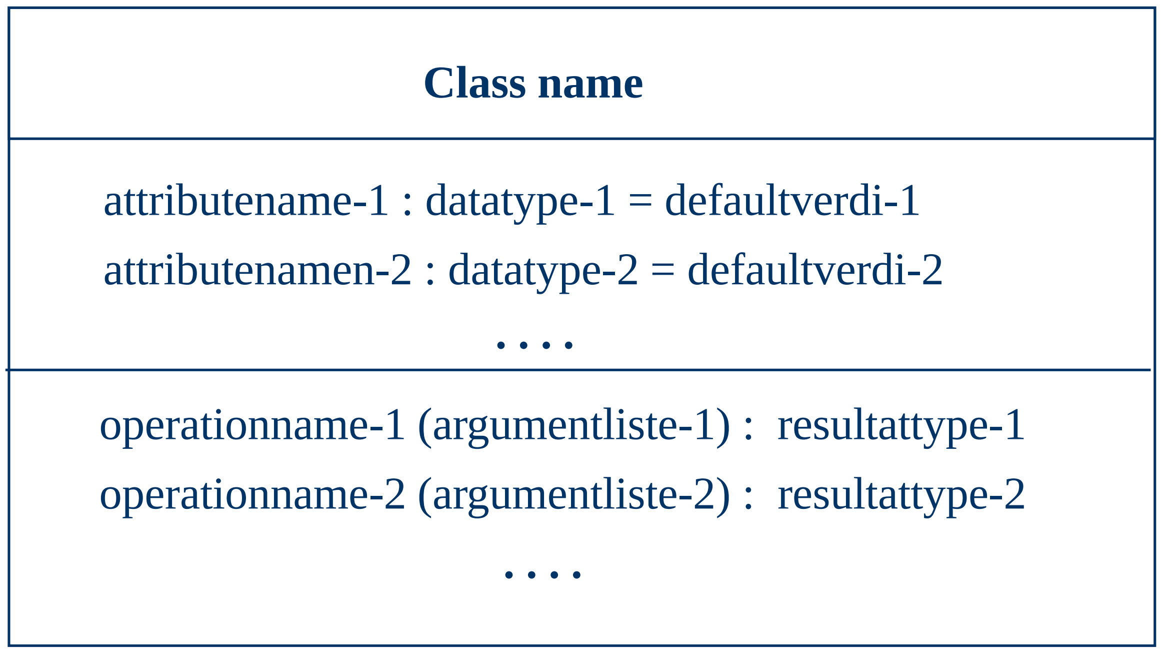


Object and classes - notation

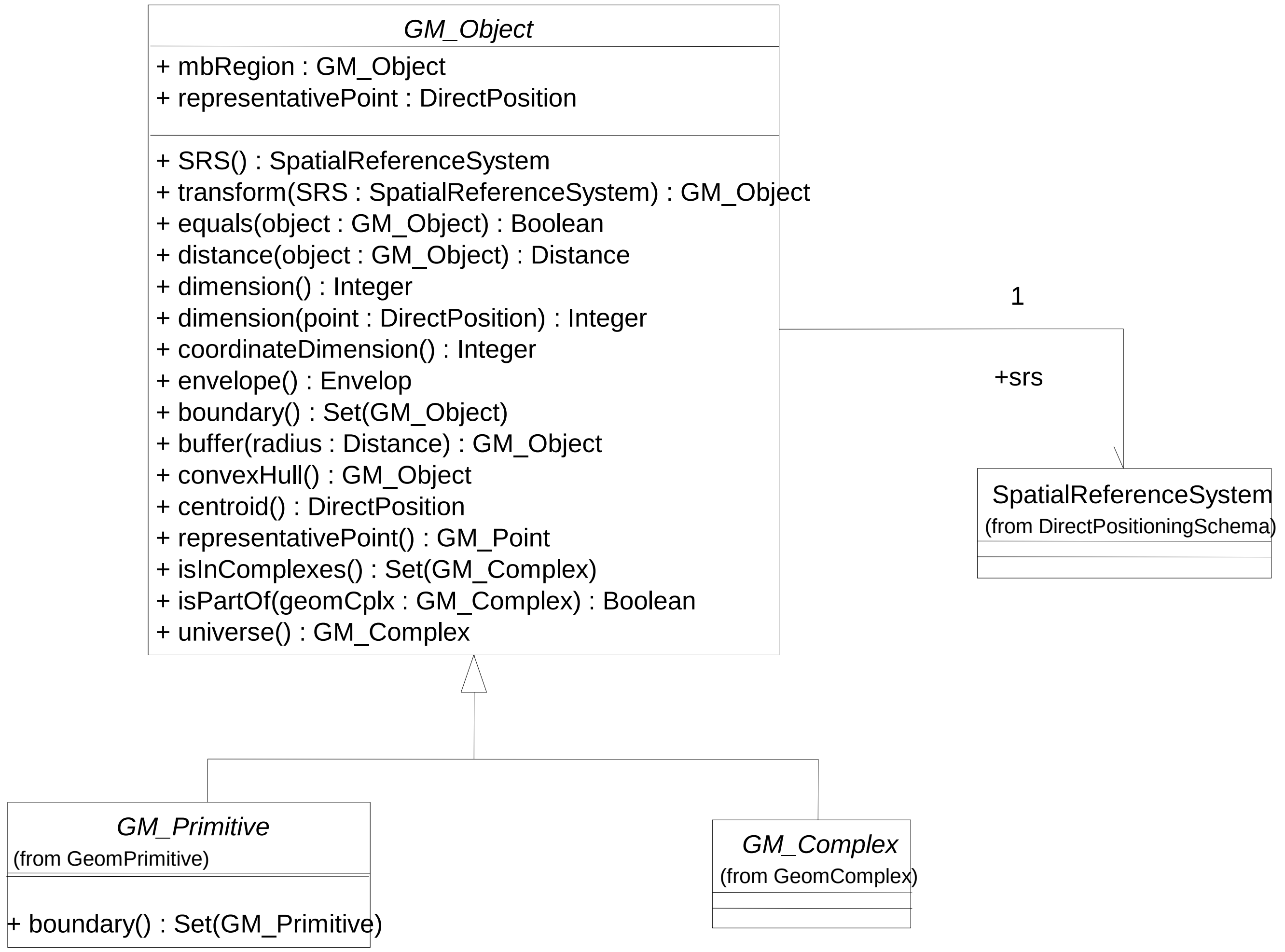


Example - object class

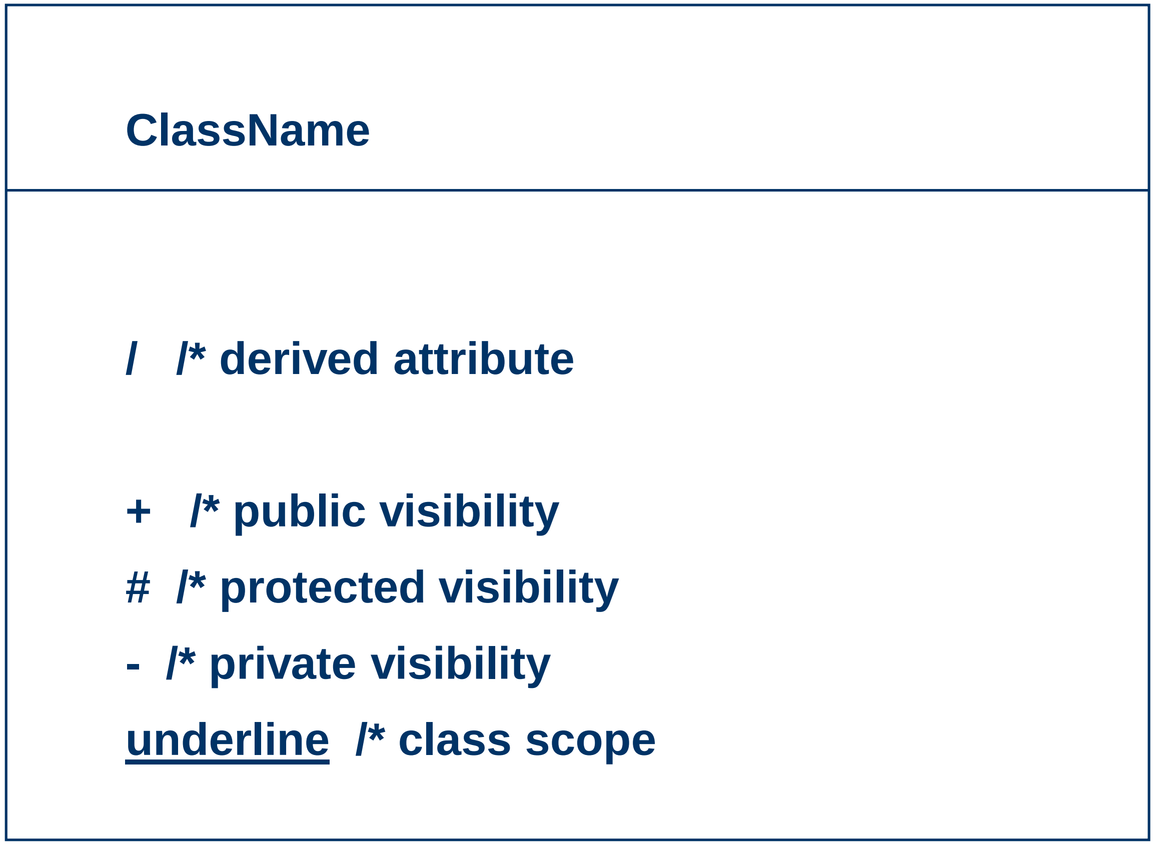
Object classes - notation


Class name
attributename-1 : datatype-1 = defaultverdi-1
attributenamen-2 : datatype-2 = defaultverdi-2
. . . .
operationname-1 (argumentliste-1) : resultattype-1
operationname-2 (argumentliste-2) : resultattype-2
. . . .

Class diagram


GM_Object
+ mbRegion : GM_Object
+ representativePoint : DirectPosition
+ SRS() : SpatialReferenceSystem
+ transform(SRS : SpatialReferenceSystem) : GM_Object
+ equals(object : GM_Object) : Boolean
+ distance(object : GM_Object) : Distance
+ dimension() : Integer
+ dimension(point : DirectPosition) : Integer
+ coordinateDimension() : Integer
+ envelope() : Envelop
+ boundary() : Set(GM_Object)
+ buffer(radius : Distance) : GM_Object
+ convexHull() : GM_Object
+ centroid() : DirectPosition
+ representativePoint() : GM_Point
+ isInComplexes() : Set(GM_Complex)
+ isPartOf(geomCplx : GM_Complex) : Boolean
+ universe() : GM_Complex
1
+srs
SpatialReferenceSystem
(from DirectPositioningSchema)
GM_Primitive
(from GeomPrimitive)
+ boundary() : Set(GM_Primitive)
GM_Complex
(from GeomComplex)

Class attributes



ClassName

/ /* derived attribute

+ /* public visibility

# /* protected visibility

- /* private visibility

underline /* class scope

Class attributes

[visibility] name [multiplicity] [:type] [= initial value] [{property-string}]

+ origin [0..1] : Point = (0,0) {frozen}

defined properties: changeable, addOnly, frozen (const)

Attributes and data types

- A data type specifies a legal value domain and the operations on values of that domain
- Four categories
  - a) Basic data types *(integer, real, string, …)*
  - b) Collection data types (from OCL)
  - c) Enumerated data types (user-definable finite sets)
  - d) Model types

Operations

An *operation* is a specification of

- a transformation, or
- a query

A *method* is a procedure that implements an operation.

Operations

[visibility] name [(parameter-list)] [:return-type] [{property-string}]

[(parameter-list element)] ::=

  [direction] name : type [= default-value]

 [direction] ::= in | out | inout

+ set (in name : Name, in place : String = ‘Oslo’)

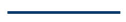
: Boolean {concurrency=sequential}

defined properties: isQuery, concurrency, ...

Optional/Conditional

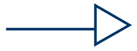
- In UML all attributes and operations are per default mandatory.
- Optional data values for attributes can be shown through multiplicity [0..1].
- Conditional should relate to a note with constraint expressed in text/OCL (ISO 19103)

Relationships



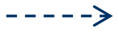
Association (the “glue”)

A semantic relationship between elements (involves connections among their instances)



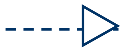
Generalization (inheritance)

A relationship between an element and the sub-elements that may be substituted for it



Dependency

The use of one element by another

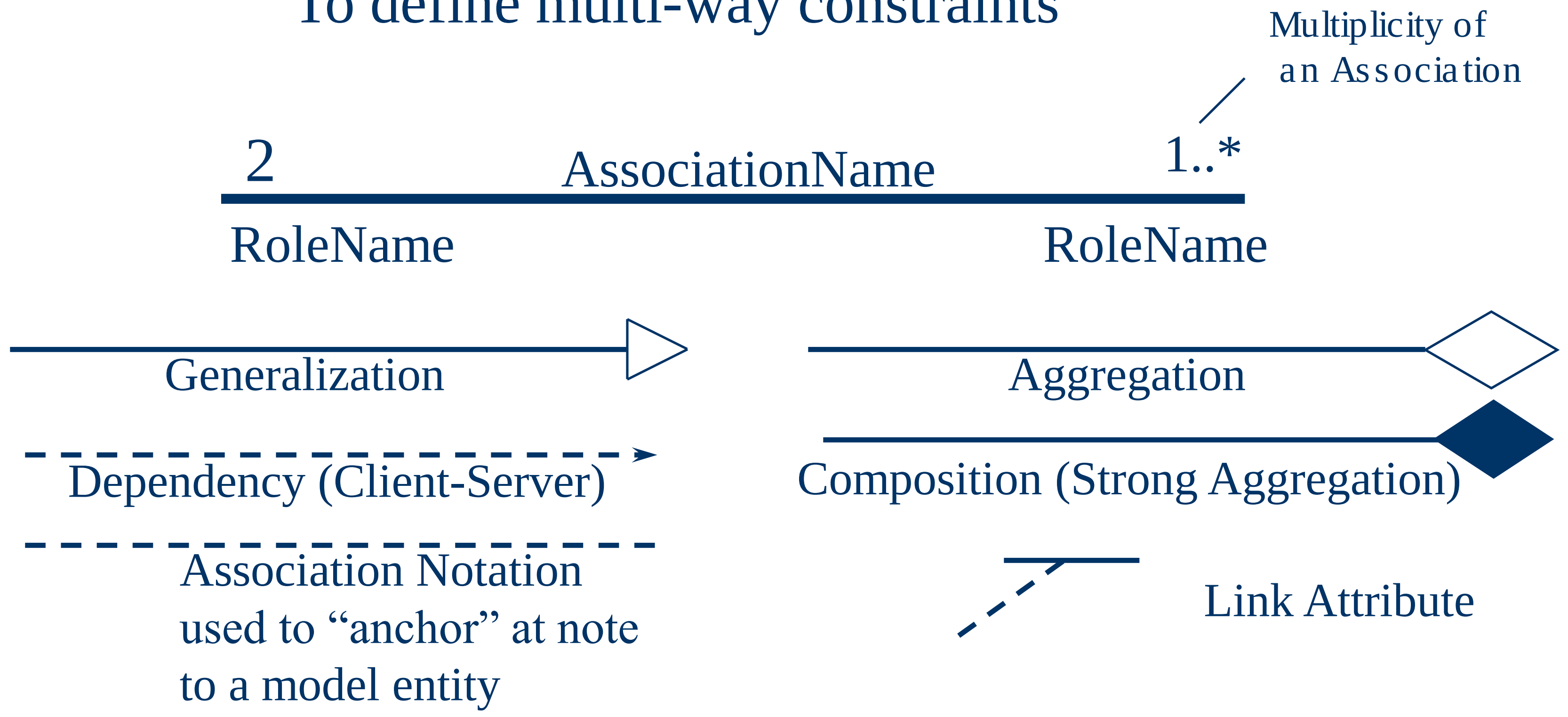


Refinement/realisation

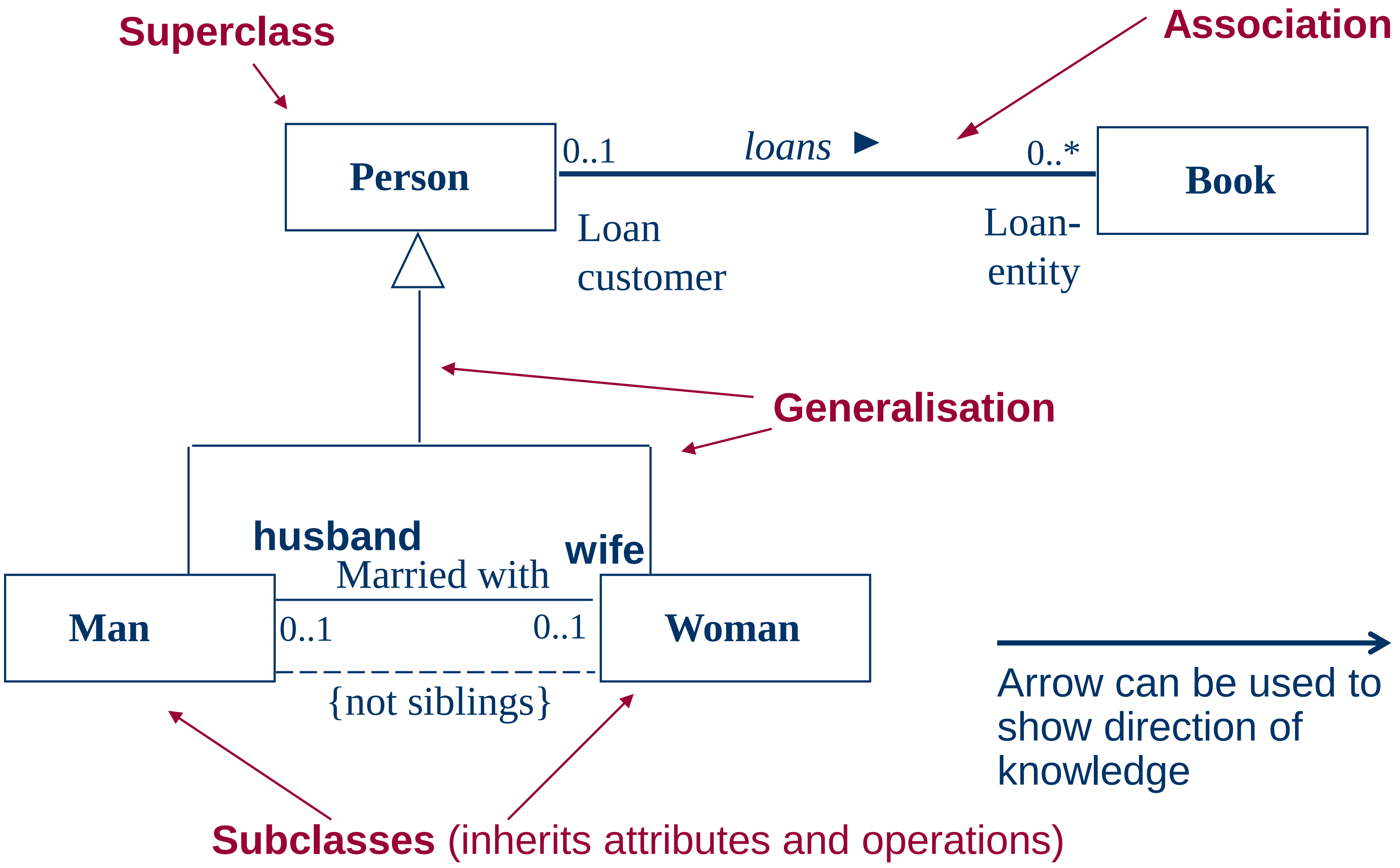
A shift in levels of abstraction

Different relationships in UML Diagrams

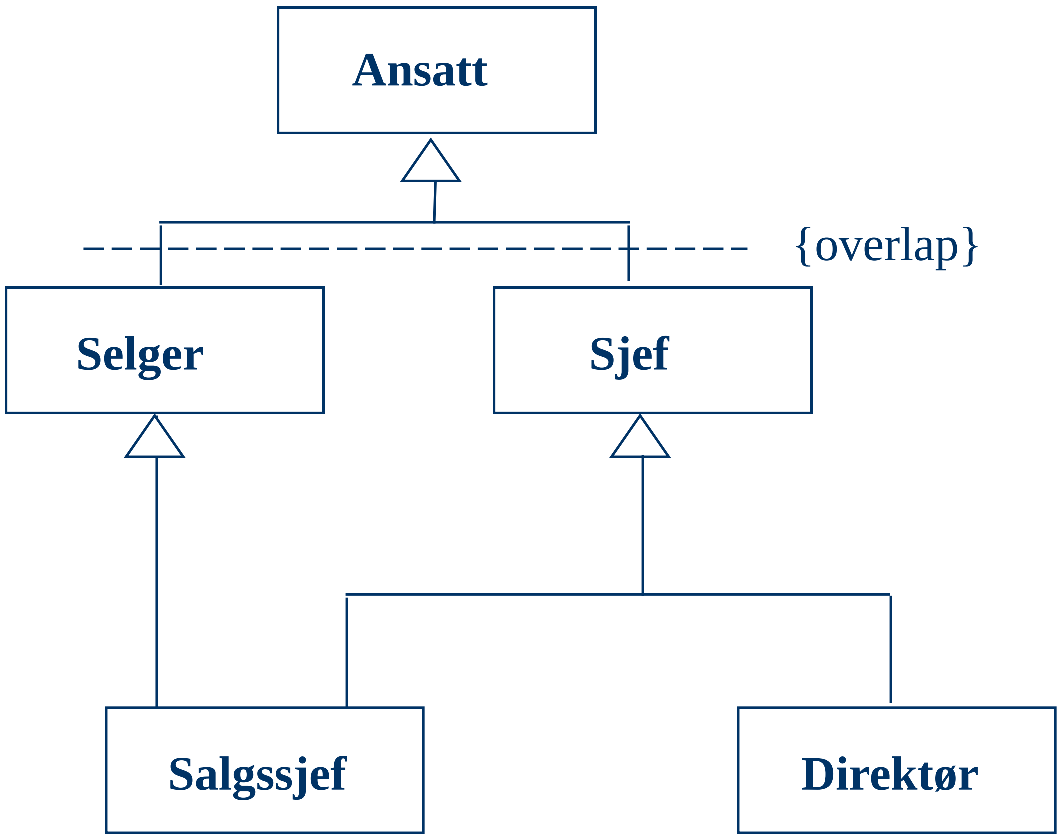
Purpose: To show relationships between model entities
To define multi-way constraints


Multiplicity of an Association
2
AssociationName
1..*
RoleName
RoleName
Generalization
Aggregation
Dependency (Client-Server)
Composition (Strong Aggregation)
Association Notation used to “anchor” at note to a model entity
Link Attribute

Generalisation and Association

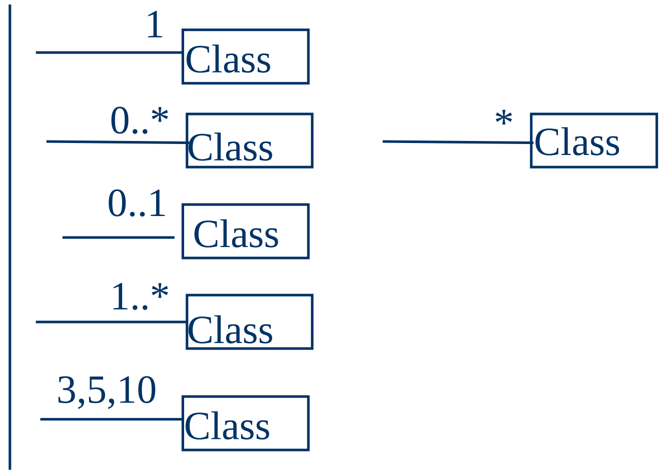


Multiple inheritance - example



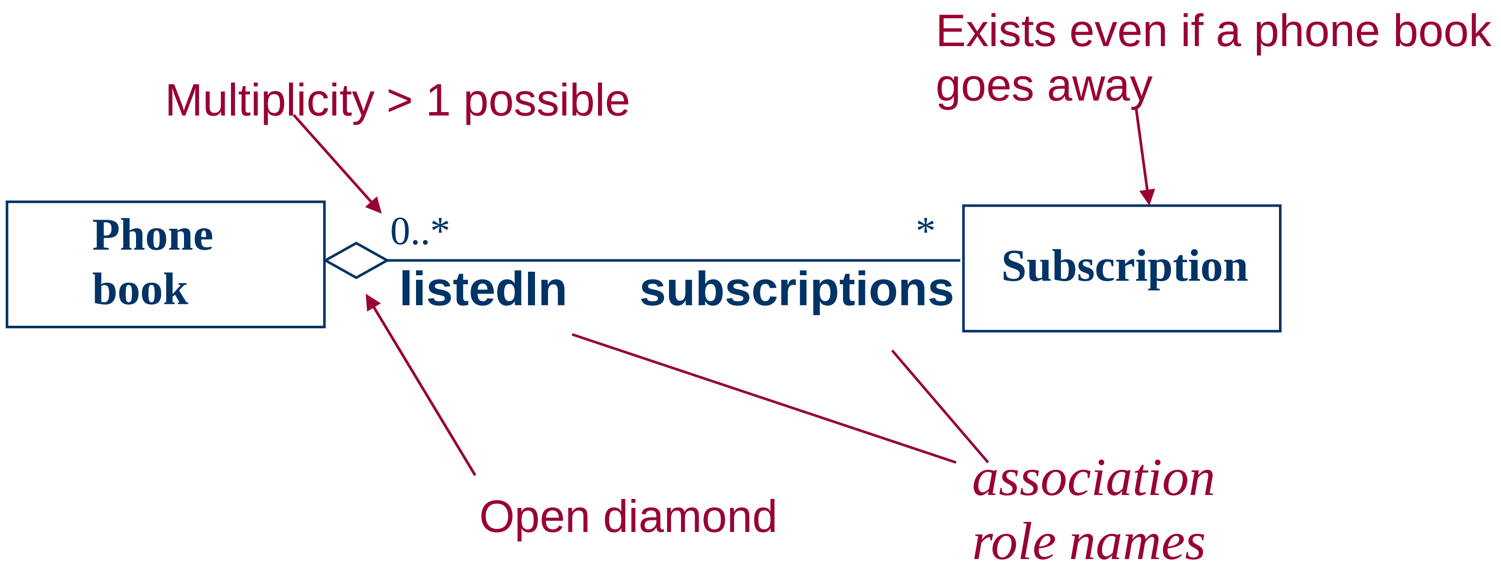
Multiplicity constraints

Multiplicity shows how many instances of a class that can be related to one instance of the class at the other end of the association



Aggregation

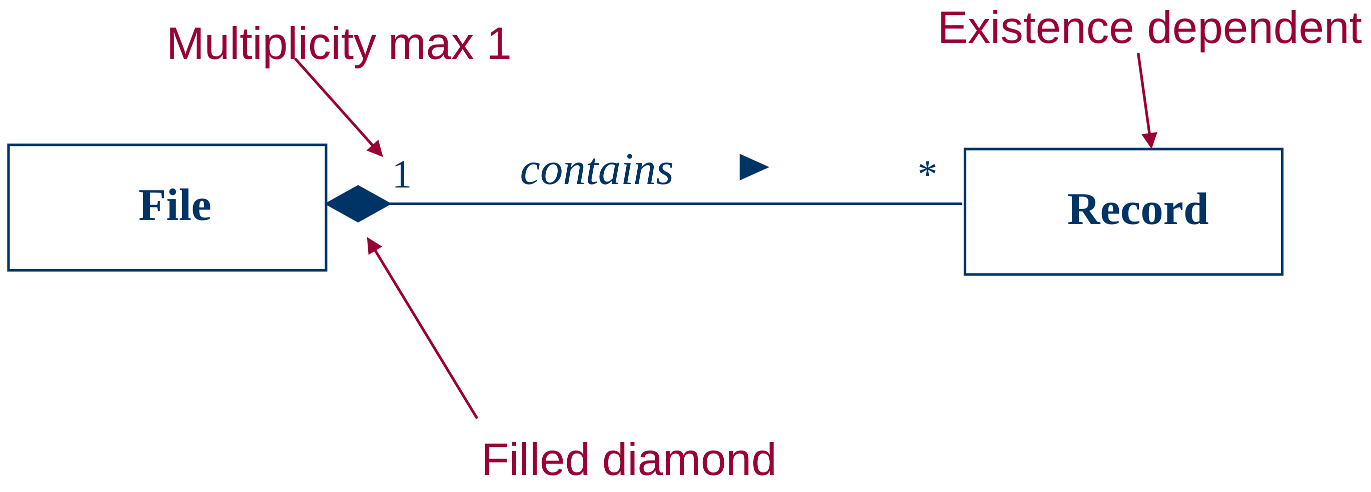
Loose (weak) part-of relationship



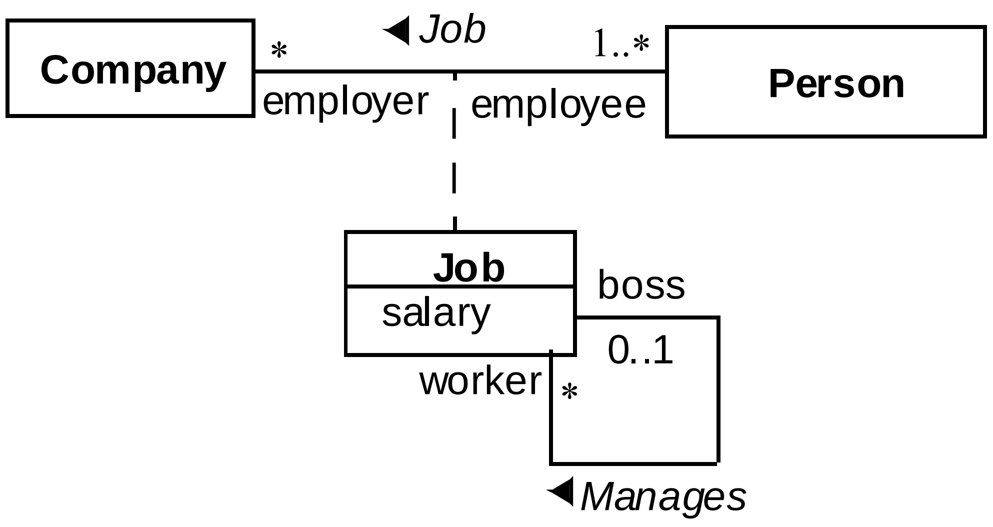
Composition

("strong aggregation")

Strong part-of relationship



Associations



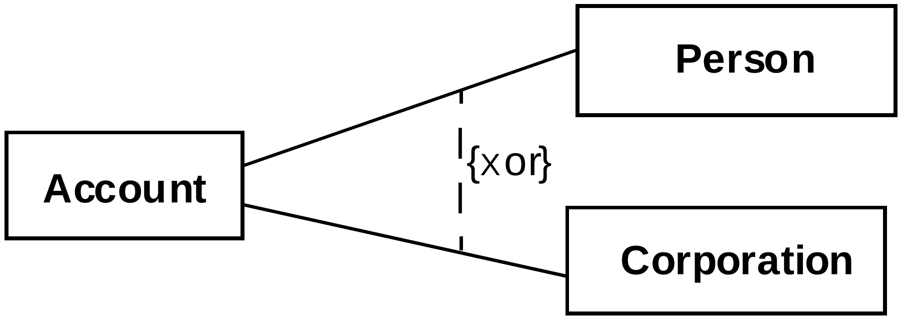


Fig. 3-31, *UML Notation Guide*

Association Ends

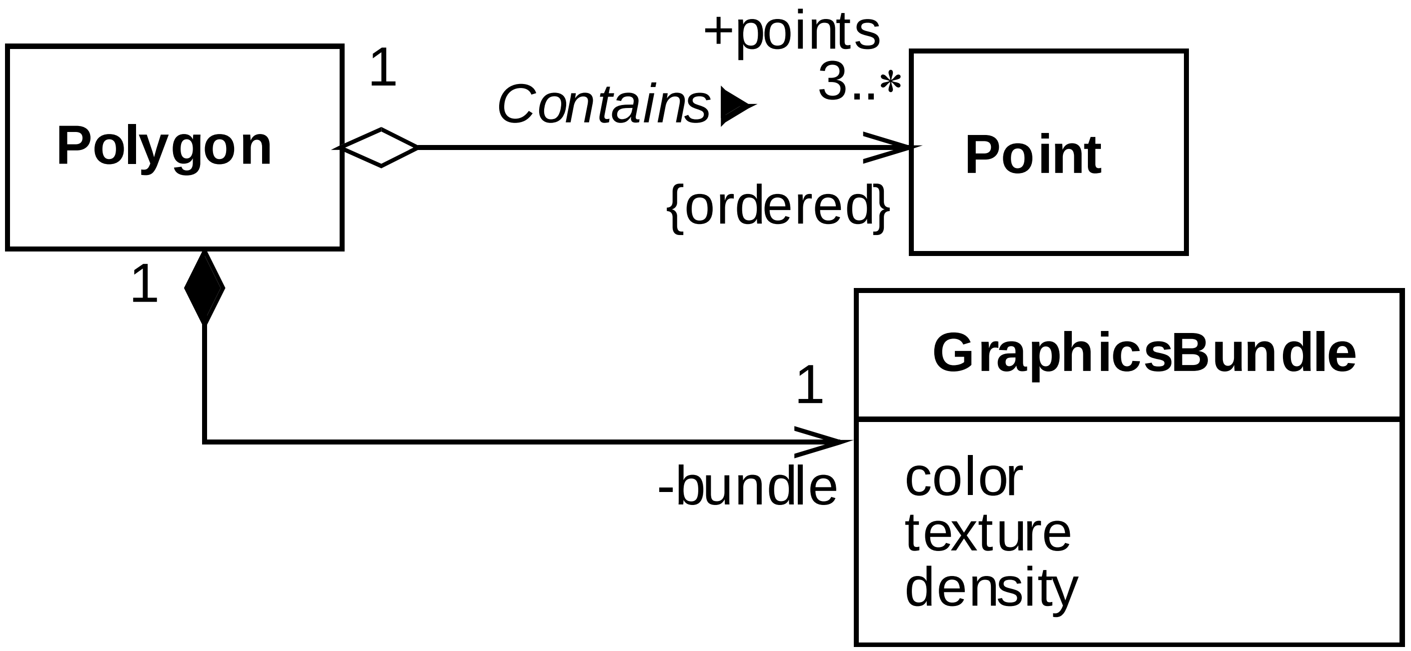
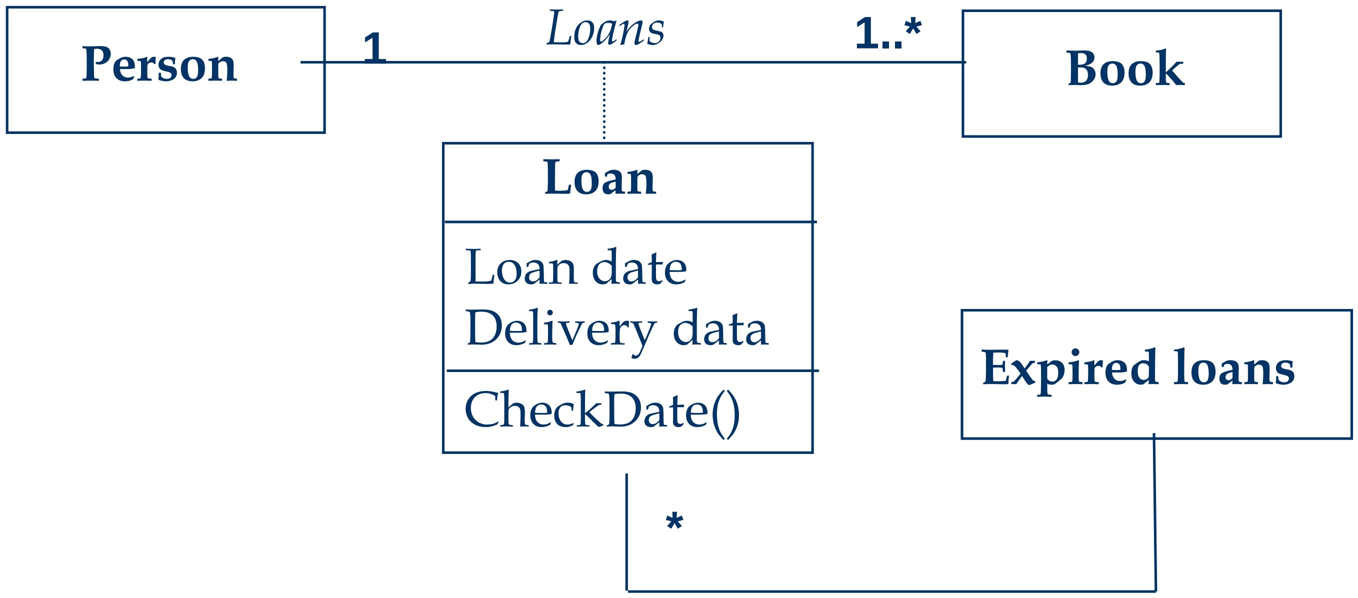


Fig. 3-32, *UML Notation Guide*

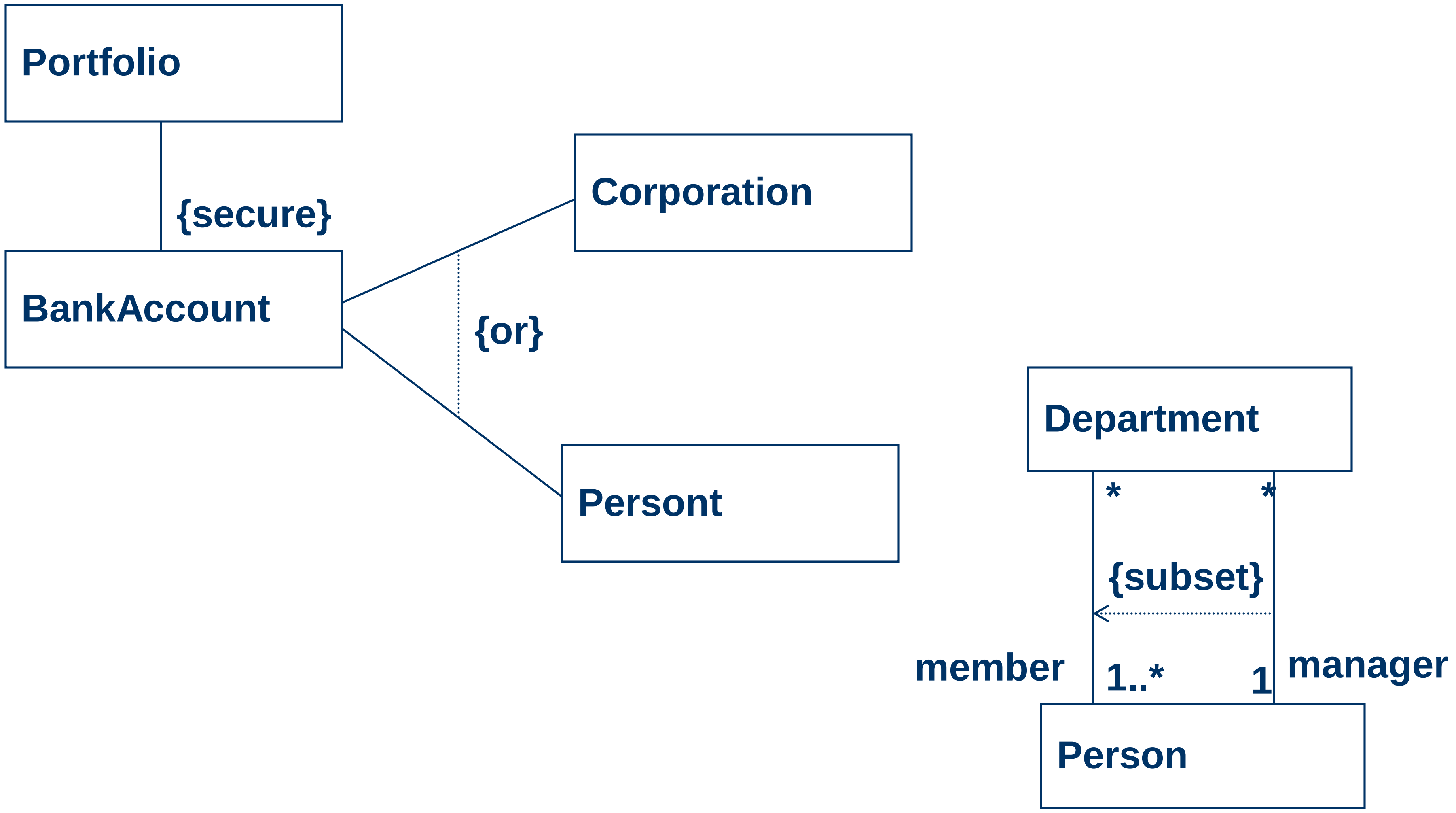
Association class

Are used instead of link-attributes if

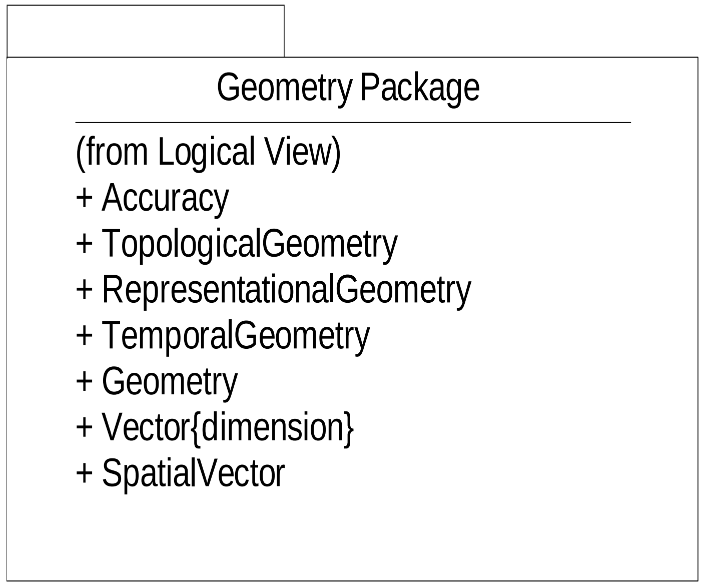
- The association are related to other objects
- Operations are attached to the association



Constraints on relationships

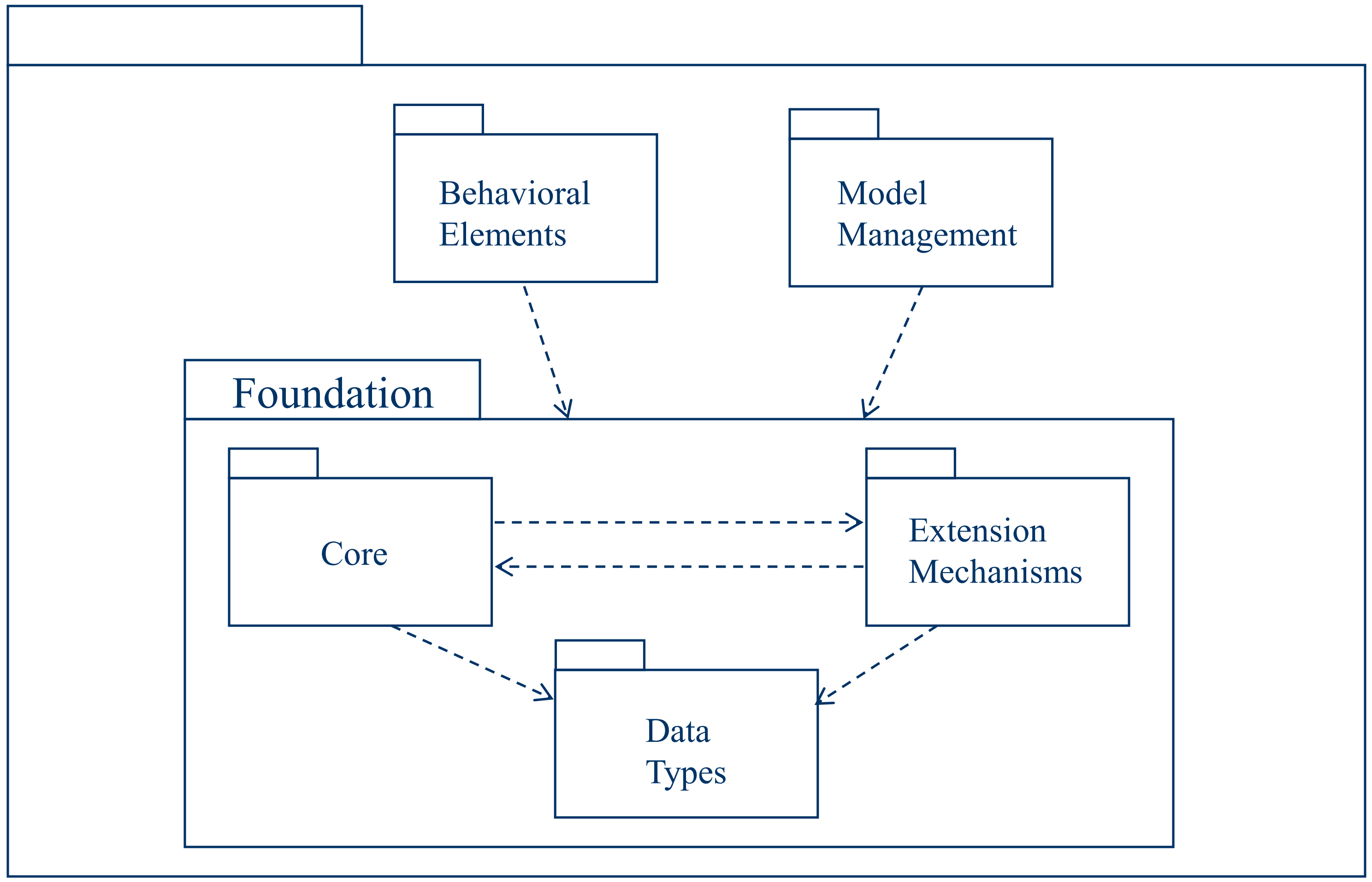


Packages in UML



- This is a grouping of model elements and diagrams.
- Package dependencies usually summarize dependencies among the contents of the packages.
- Packages can contain other Packages.
- Packages can show the Class/Entities found in a given Package.

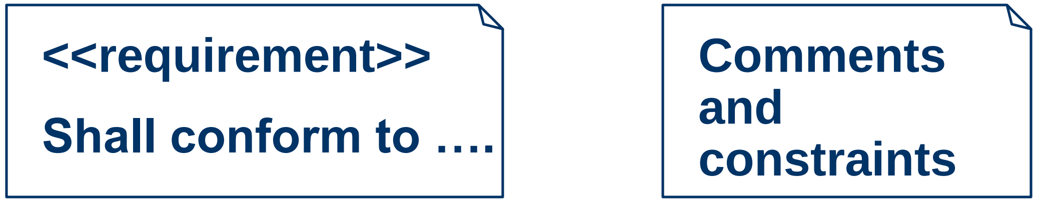
Containment and dependency



Common Mechanisms

- Adornments: notes
- Extensibility mechanisms: stereotypes, tagged values, constraints

Notes:



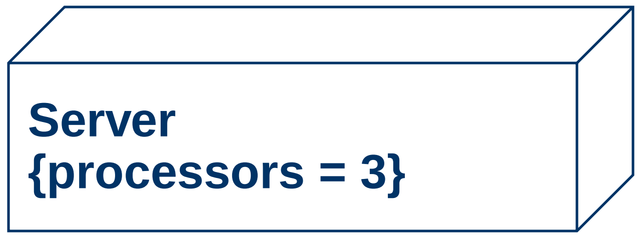
Stereotypes

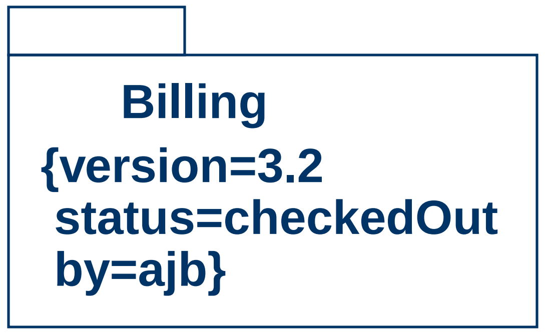
- Used to define derivative modeling concepts based on existing generic modeling concepts
- Defined by:
  - base (meta-)class = UML meta-class or stereotype
  - constraints
  - required tags (0..*)
    - often used for modeling pseudo-attributes
  - icon
- A model element can have at most one stereotype

Tagged values - properties

Name (tag) separator (=) value (of the tag)

properties on an element - relevant for code generation or configuration management (Can be applied to all UML elements)





OCL - Object Constraint Language

First order predicate logic, for Boolean expressions - included in UML 1.1

Can be used for:
invariants,
value restrictions,
pre- and postconditions

Expressions with:
and, or, not, implies, exists, for all,
Collections (select, reject, collect, iterate)

Person

self.age > 0

Married people are of age >= 18

self.wife->notEmpty implies self.wife.age >= 18 and

self.husband->notEmpty implies self.husband.age >= 18 and

ISO 19103 – Conceptual modeling with UML

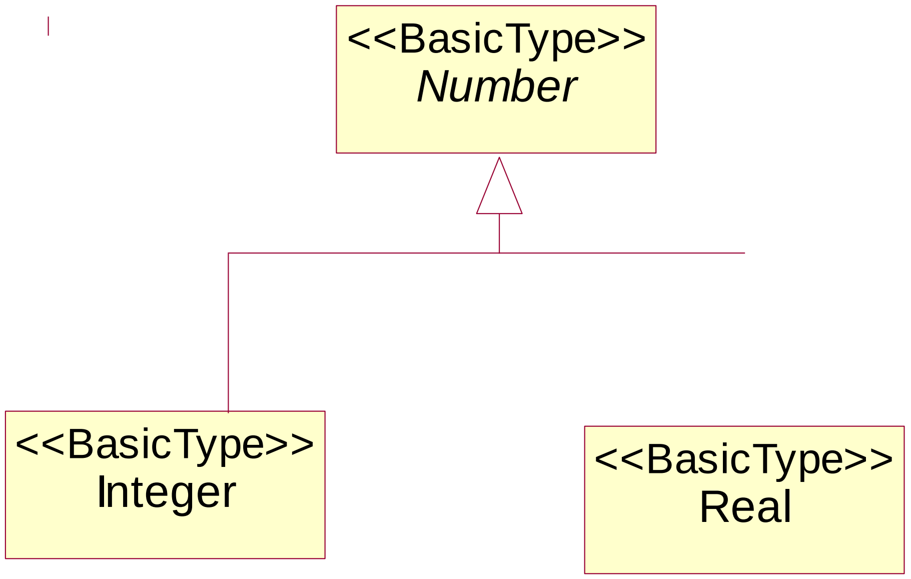
Implementation neutral! Not implementation specific!

- Basic data types
- Stereotypes
- Naming
- Documentation of models
- Information modelling guidelines

Basic Types

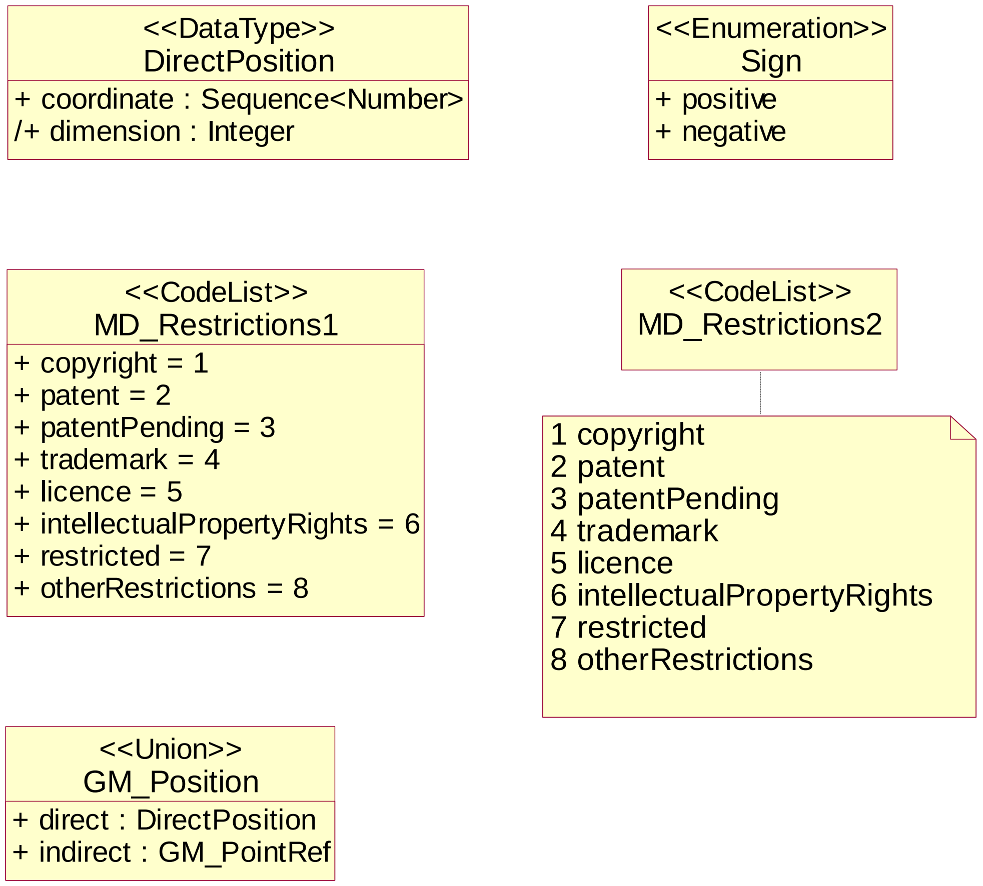
- Primitive
  - Decimal, Integer, Number, Real, Vector, Character, CharacterString, Date, Time DateTime, Binary
  - Sign, Boolean, Logical, Probability
- Collections
  - Set<T>, Bag<T>, Sequence<T>, Dictionary<K,T>
- Measures
  - Angle, Area, Distance, length, Scale, MTime, Velocity, Volume, UnitOfMeasure
- Records, Namespace

Number



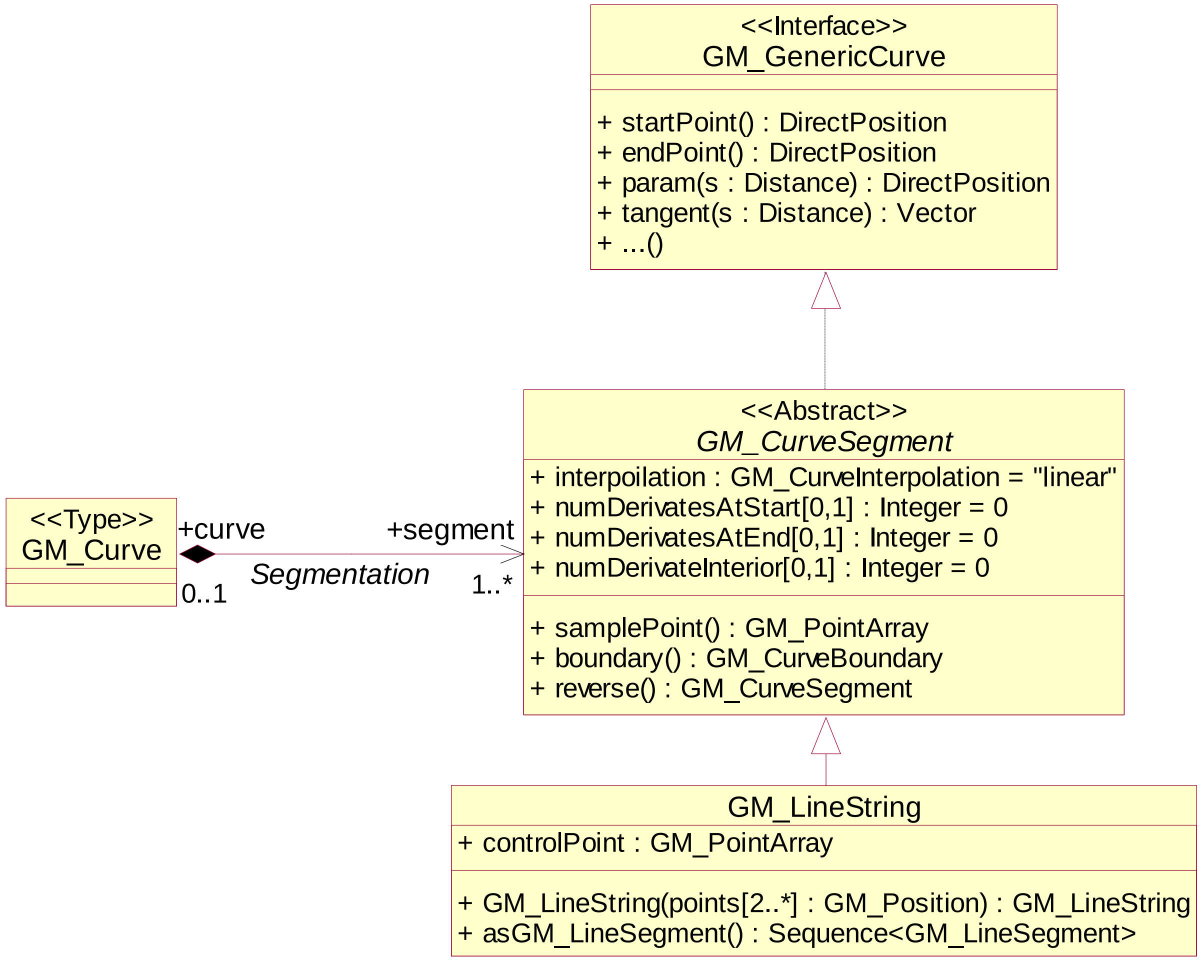
User defined data types

- <<DataType>>
- <<Enumeration>>
- <<CodeList>>
- <<Union>>



Class Stereotypes

- NONE
- <<Interface>>
- <<Type>>
- <<Abstract>>
- <<MetaClass>>


<<Interface>>
GM_GenericCurve
+ startPoint() : DirectPosition
+ endPoint() : DirectPosition
+ param(s : Distance) : DirectPosition
+ tangent(s : Distance) : Vector
+ ...()
<<Abstract>>
GM_CurveSegment
+ interpoilation : GM_CurveInterpolation = "linear"
+ numDerivatesAtStart[0,1] : Integer = 0
+ numDerivatesAtEnd[0,1] : Integer = 0
+ numDerivateInterior[0,1] : Integer = 0
+ samplePoint() : GM_PointArray
+ boundary() : GM_CurveBoundary
+ reverse() : GM_CurveSegment
<<Type>>
GM_Curve
+curve
+segment
Segmentation
0..1
1..*
GM_LineString
+ controlPoint : GM_PointArray
+ GM_LineString(points[2..*] : GM_Position) : GM_LineString
+ asGM_LineSegment() : Sequence<GM_LineSegment>

Class (package, association) names

- All classes must have unique names.
- Names shall start with upper case letter.
- Should not have a name that is based on its external usage, since this may limit reuse.
- Should not contain spaces
  - Separate words in a class should be concatenated, e.g. “XnnnYmmmm”
- Class names should start with bialpha prefixes for each standard part.

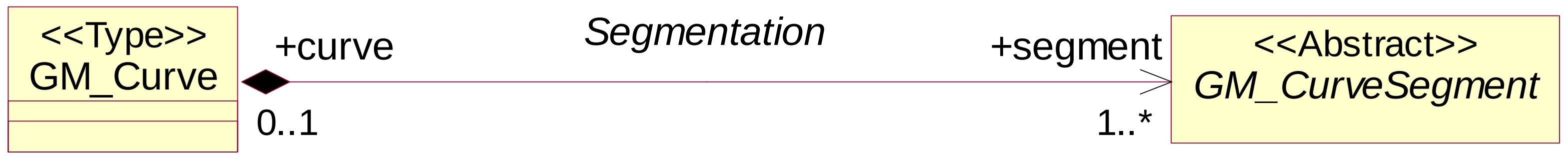
GM_CurveSegment
MD_Citation
GM_FeatureAttribute

Attribute, operation and role names

- Shall start with lower case letter
- Concatenate words shall begin with capital letter
- Do not repeat class names in attribute names
- Keep names technical, meaningful and short, if possible

computePartialDerivates
compute Partial Derivates
compute_partial_derivatives

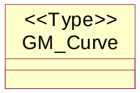
Associations



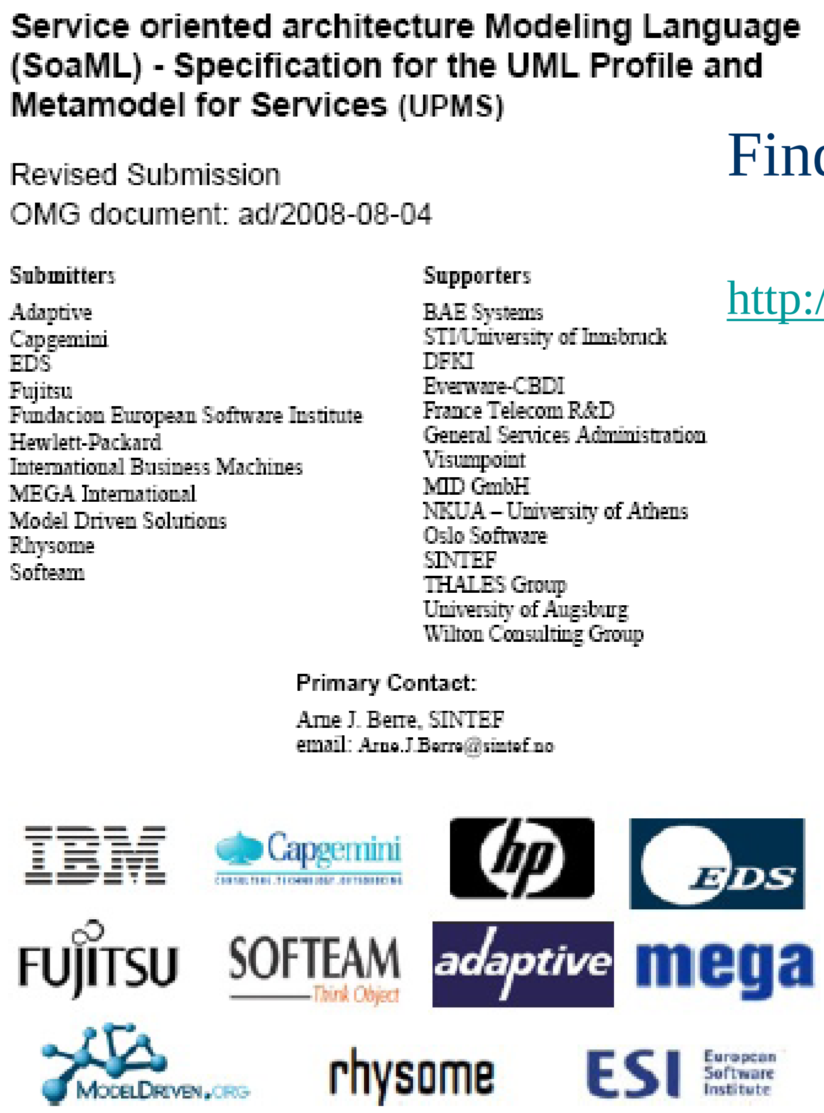
- Multiplicity shall be defined for both associations ends
- All associations ends (roles) representing the direction of a relationship must be named or else the association itself must be named
- The role name must be unique within the context of a class and its supertypes.
- The direction of an association must be specified

Documentation of models

- Diagrams
  - Package dependency diagrams
  - Class diagrams
  - Class context diagrams
- Text
  - Class
    - General description, semantics, supertypes/subtypes
  - Attributes, Associations
  - Operations
    - preconditions, input/output parameters, return value, post conditions, exceptions, constraints
  - Constraints



NB! Font size > 8pt



Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS)

Revised Submission
OMG document: ad/2008-08-04

Submitters	**Supporters**
Adaptive	BAE Systems
Capgemini	STI/University of Innsbruck
EDS	DFKI
Fujitsu	Everware-CBDI
Fundacion European Software Institute	France Telecom R&D
Hewlett-Packard	General Services Administration
International Business Machines	Visumpoint
MEGA International	MID GmbH
Model Driven Solutions	NKUA – University of Athens
Rhysome	Oslo Software
Softeam	SINTEF
	THALES Group
	University of Augsburg
	Wilton Consulting Group

Primary Contact:

Arne J. Berre, SINTEF
email: Arne.J.Berre@sintef.no

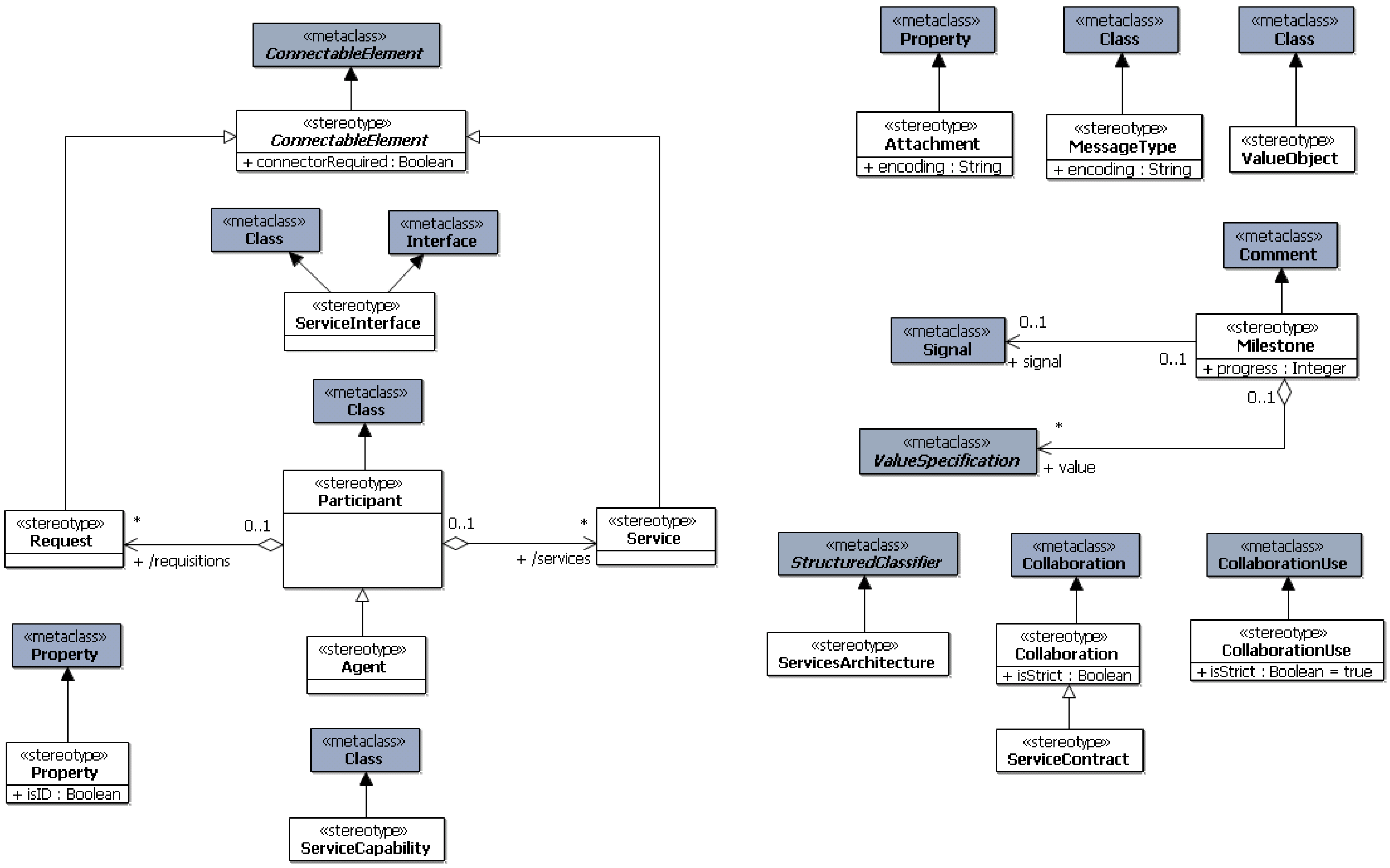
Find the document here:

http://www.omg.org/cgi-bin/doc?ad/08-08-04.pdf

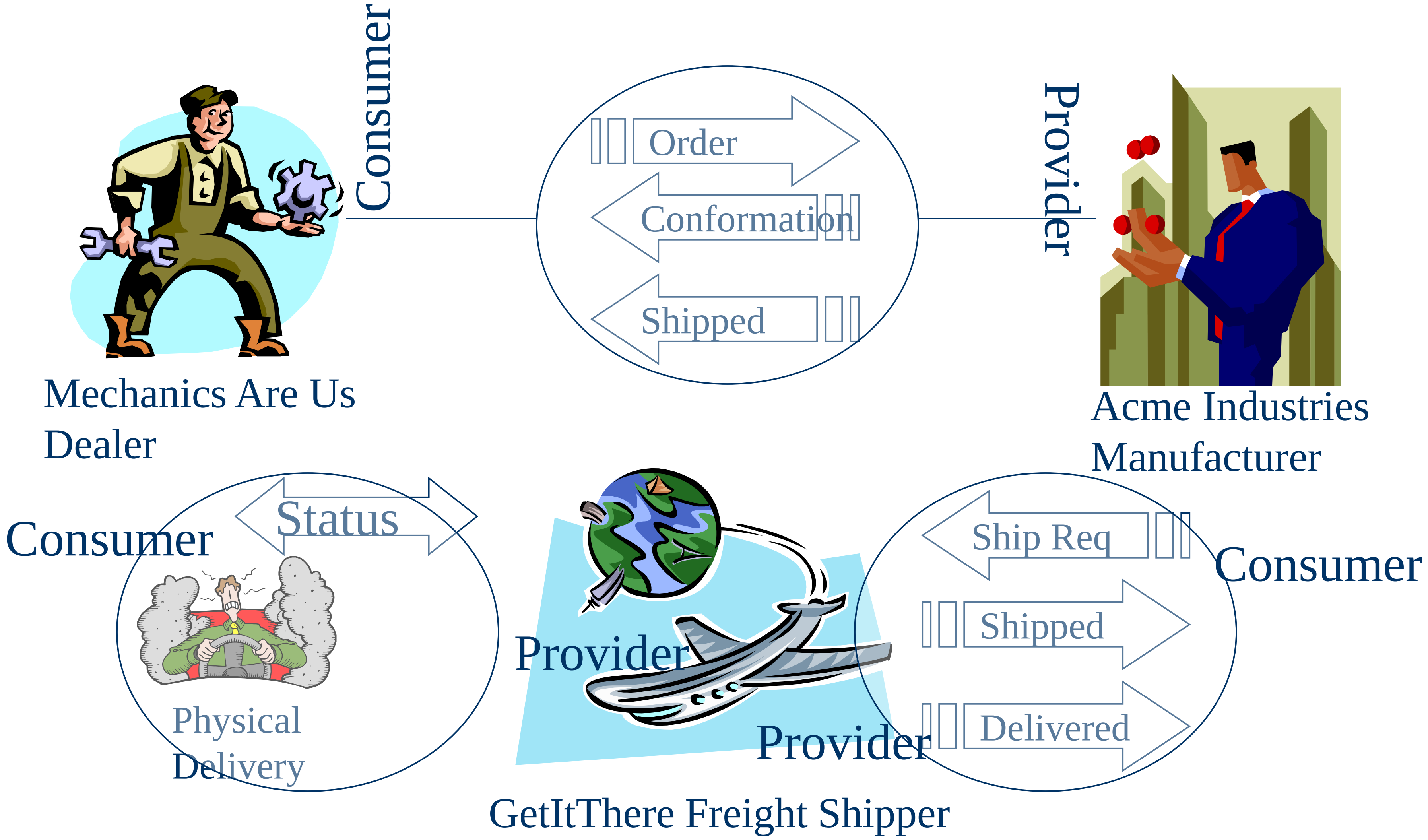
See also: www.soaml.or

Revised version of SoaML per November 10th, 2008 and January 31st, 2009

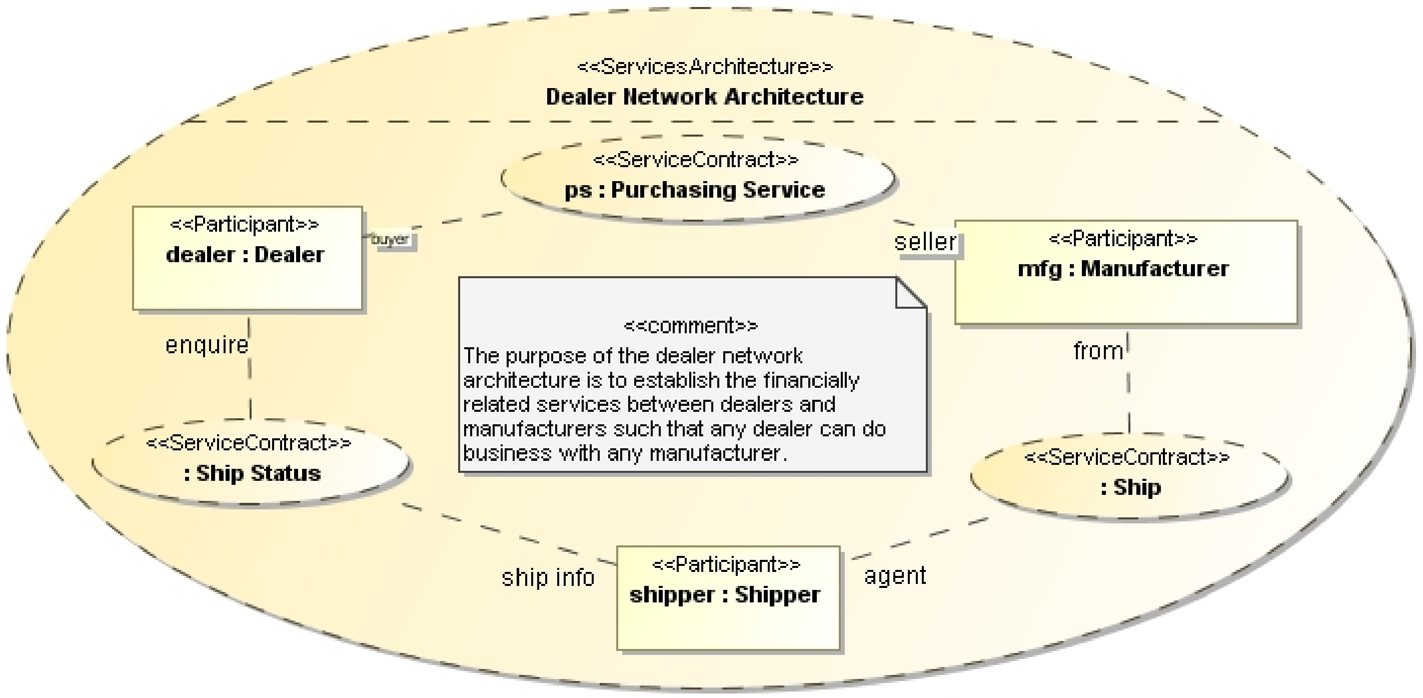
SoaML UML Profile & Metamodel



Example: Marketplace Services



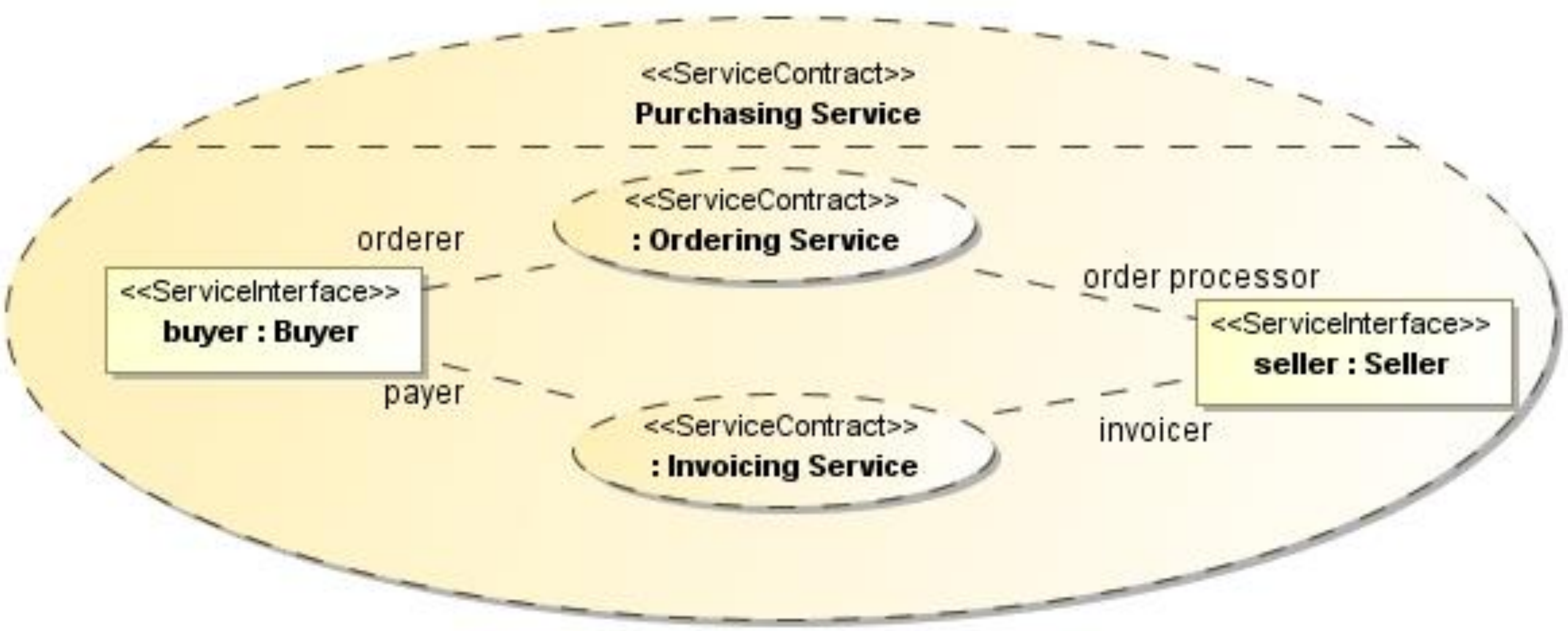
Services Architecture



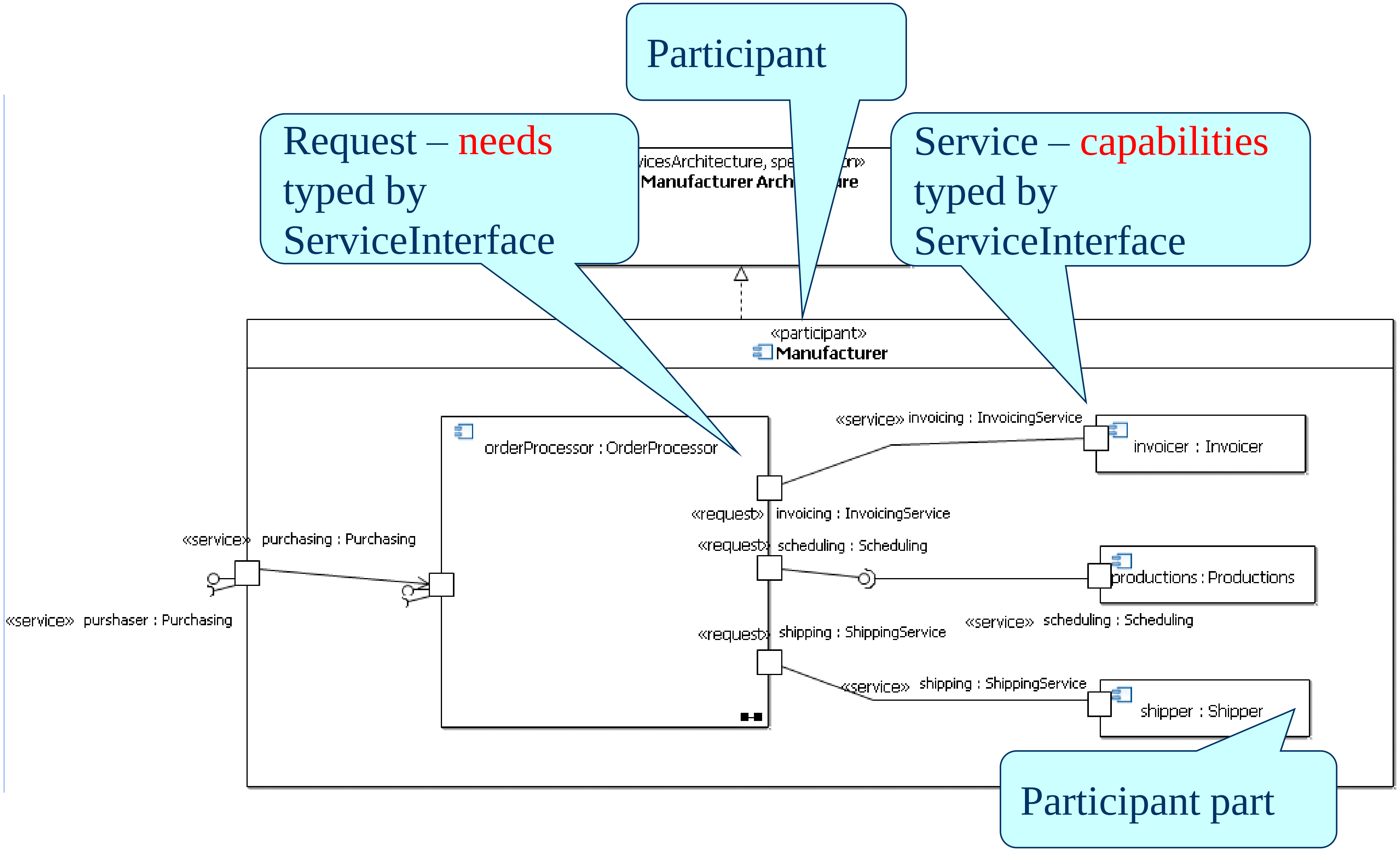
<<comment>>
The purpose of the dealer network architecture is to establish the financially related services between dealers and manufacturers such that any dealer can do business with any manufacturer.

A ServicesArchitecture (or SOA) is a network of participant roles *providing* and *consuming services* to fulfill a purpose. The services architecture defines the requirements for the types of participants and service realizations that fulfill those roles.

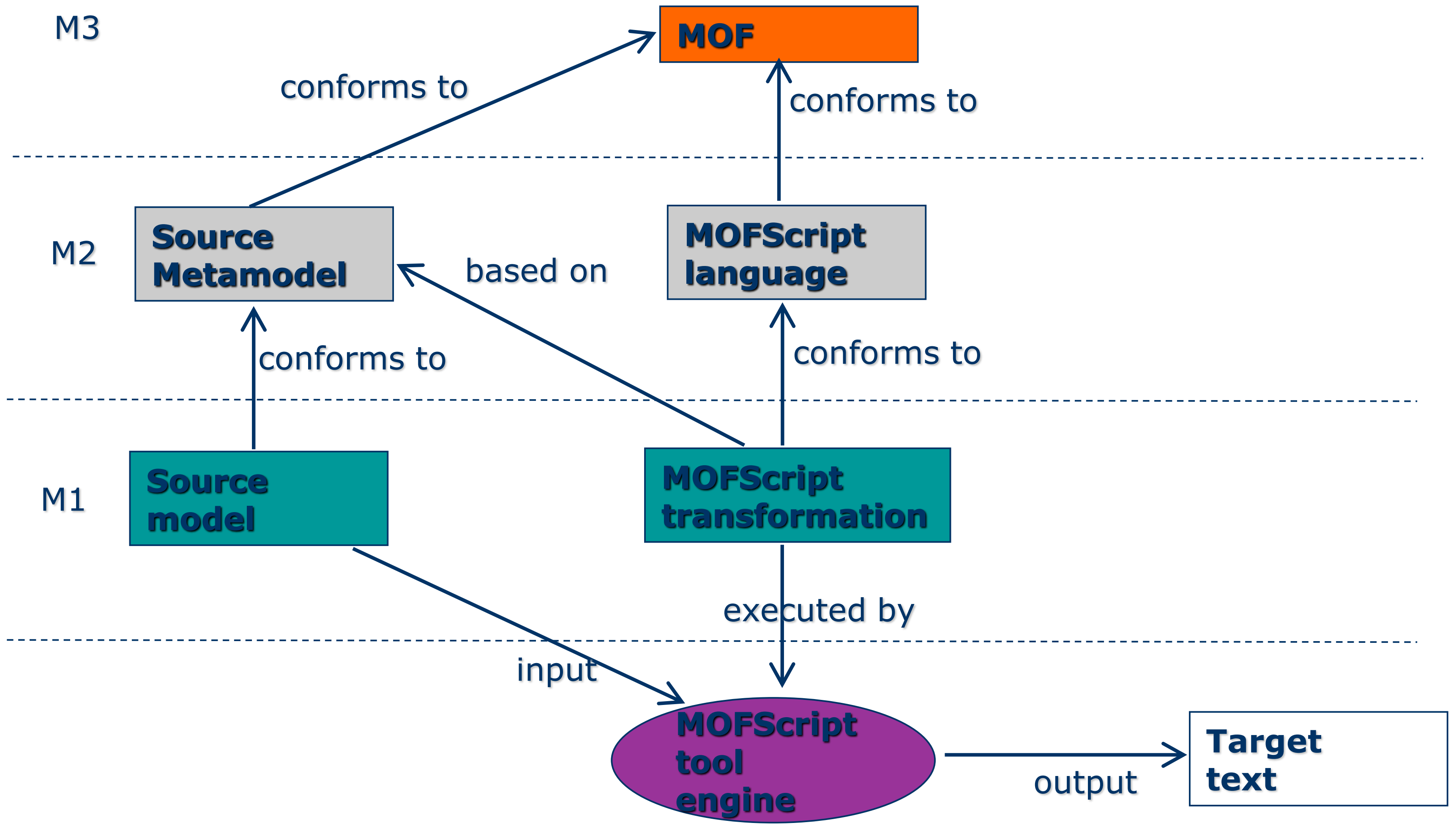
Compound services



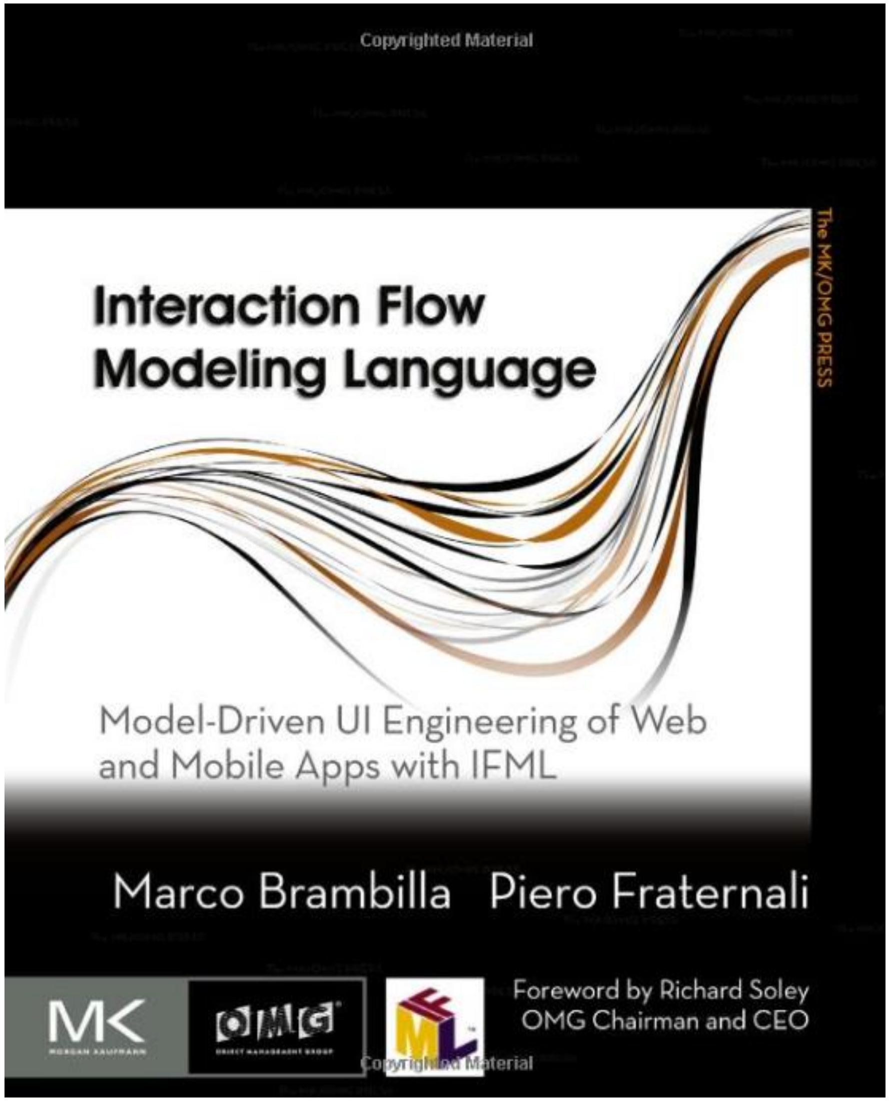
Participants may be assemblies of other Participants (UML 2.0)


Participant
Request – needs typed by ServiceInterface
Service – capabilities typed by ServiceInterface
«servicesArchitecture, spe...on»
Manufacturer Arch...re
«participant»
Manufacturer
orderProcessor : OrderProcessor
«service» invoicing : InvoicingService
invoicer : Invoicer
«request» invoicing : InvoicingService
«service» purchasing : Purchasing
«request» scheduling : Scheduling
productions : Productions
«service» purshaser : Purchasing
«request» shipping : ShippingService
«service» scheduling : Scheduling
«service» shipping : ShippingService
shipper : Shipper
Participant part

MOFScript placed in the 4-layer metamodel architecture (MDE)



IFML – for Model Driven Mobile Apps


Copyrighted Material

Interaction Flow Modeling Language

Model-Driven UI Engineering of Web and Mobile Apps with IFML

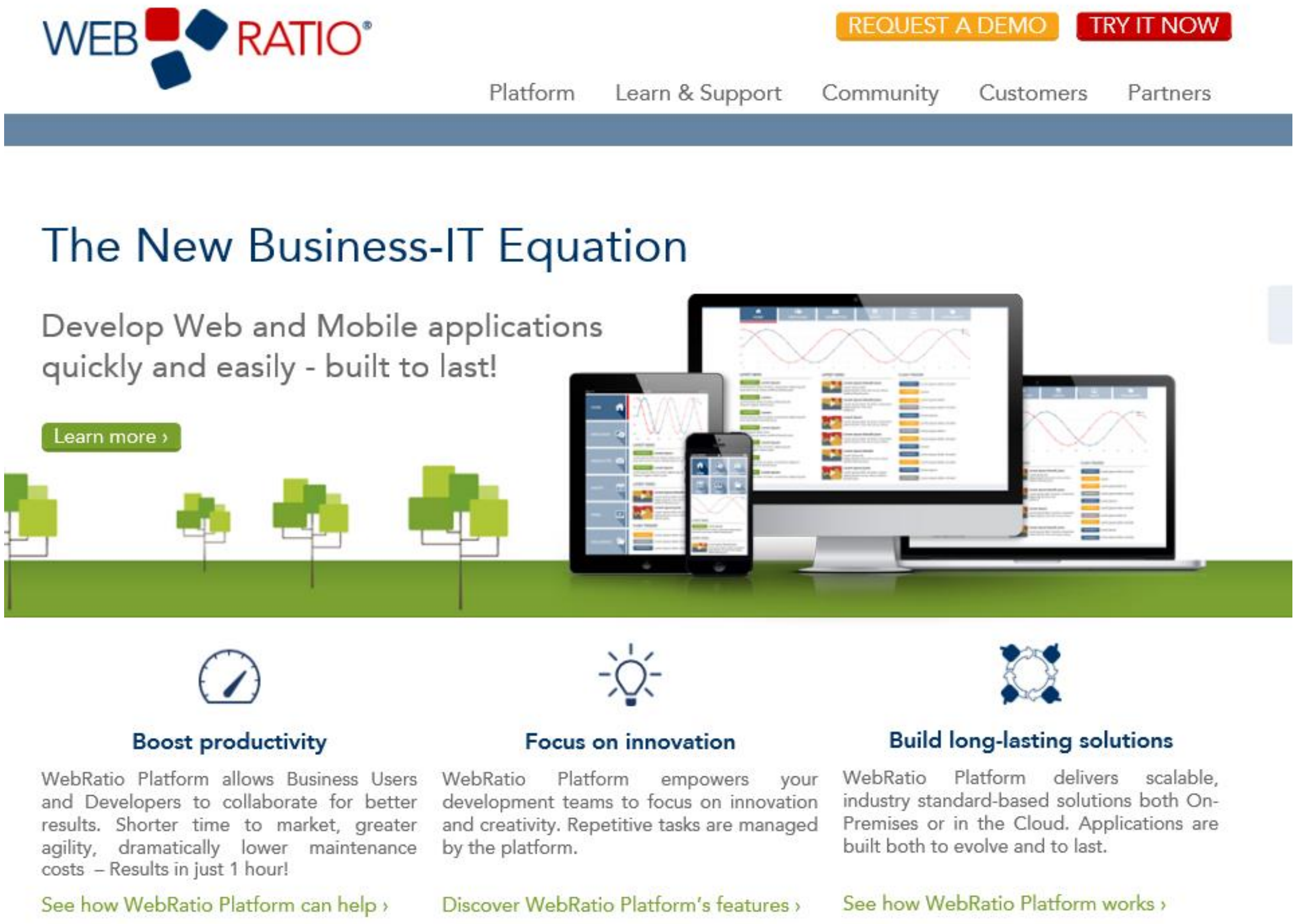
Marco Brambilla Piero Fraternali

Foreword by Richard Soley
OMG Chairman and CEO

The MK/OMG PRESS

Copyrighted Material

WebRatio



REQUEST A DEMO   TRY IT NOW

Platform   Learn & Support   Community   Customers   Partners

The New Business-IT Equation

Develop Web and Mobile applications quickly and easily - built to last!

Learn more ›

Boost productivity

WebRatio Platform allows Business Users and Developers to collaborate for better results. Shorter time to market, greater agility, dramatically lower maintenance costs – Results in just 1 hour!

See how WebRatio Platform can help ›

Focus on innovation

WebRatio Platform empowers your development teams to focus on innovation and creativity. Repetitive tasks are managed by the platform.

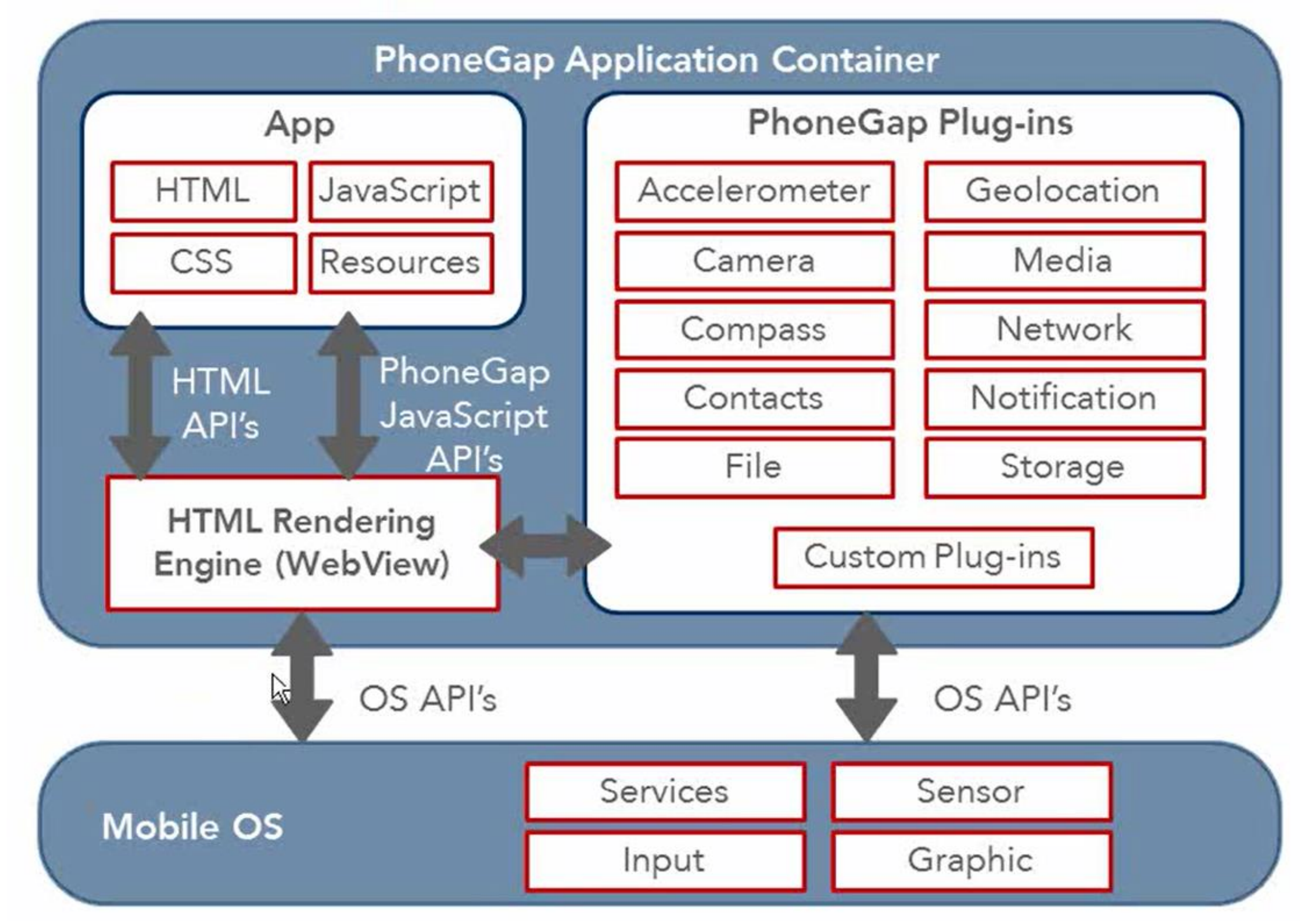
Discover WebRatio Platform's features ›

Build long-lasting solutions

WebRatio Platform delivers scalable, industry standard-based solutions both On-Premises or in the Cloud. Applications are built both to evolve and to last.

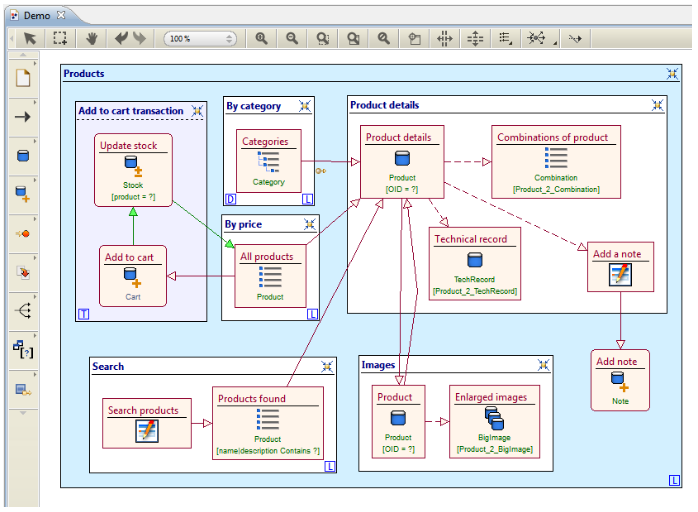
See how WebRatio Platform works ›

Mobile App Architecture



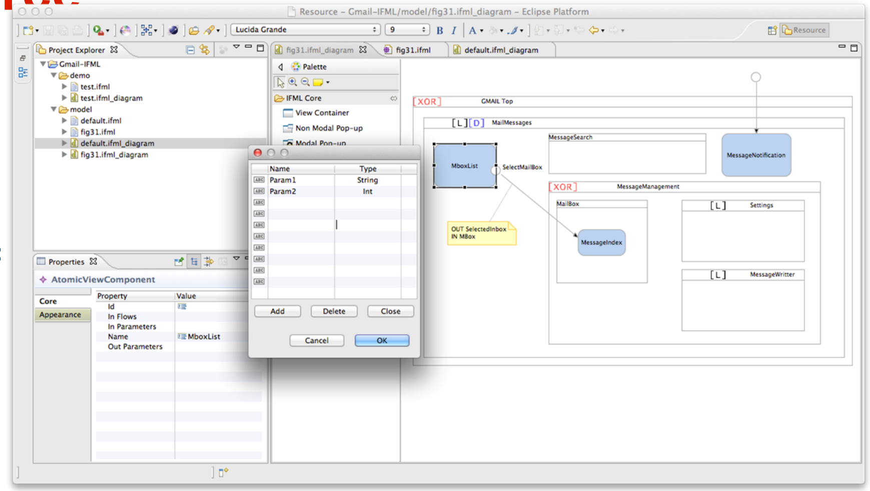
IFML and IFML Meta model

- Consolidated evaluation of WebML
  - WebRatio toolsuite
  - 10 years of experience
  - Will be updated to the new IFML notation

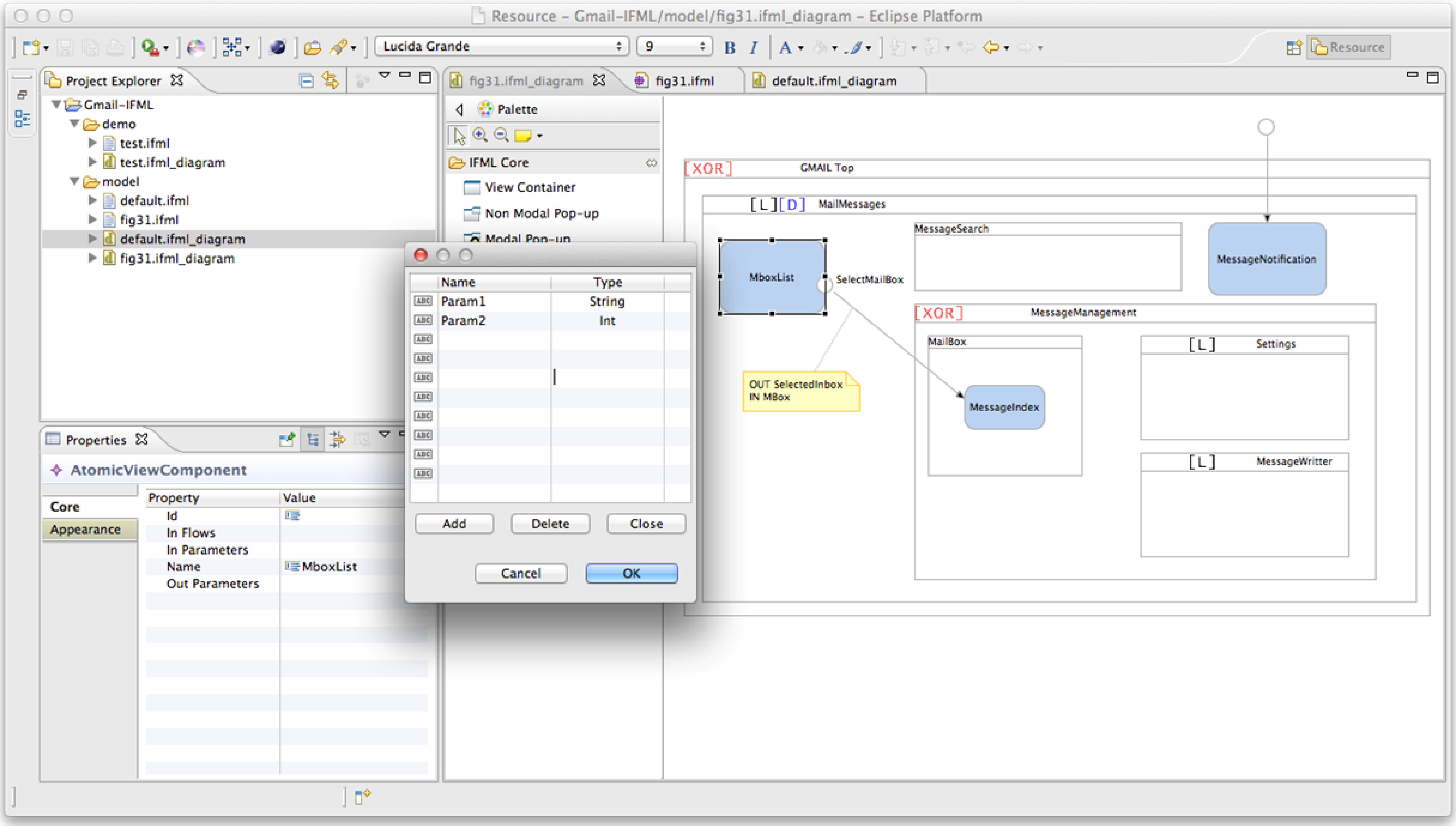


Implementation of new, open-source IFML modeling tool

- Eclipse based
- EMF / GMF
- Integration with UML / fUML / Alf



Proof of concepts



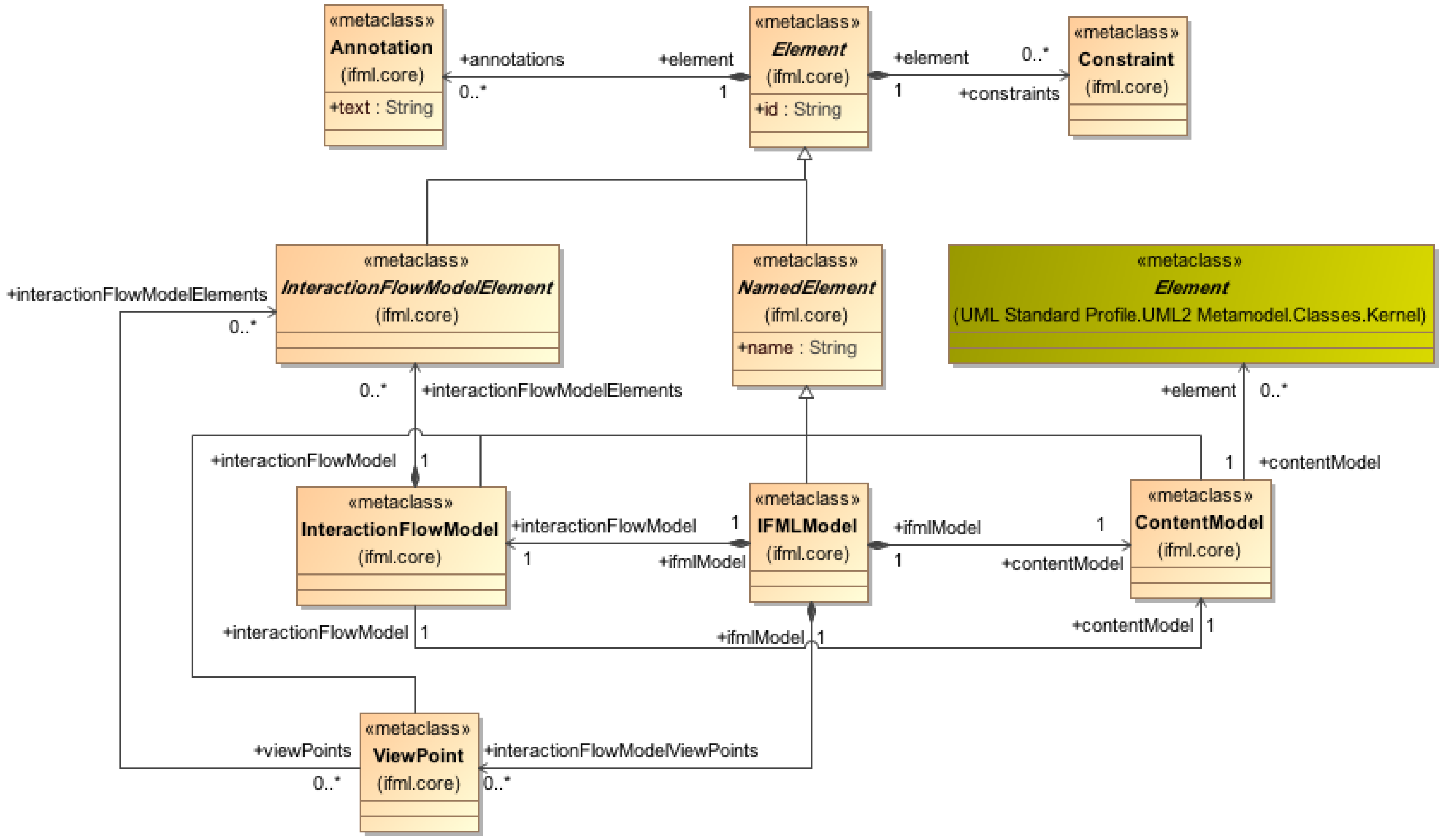
Resource – Gmail-IFML/model/fig31.ifml_diagram – Eclipse Platform

Name	Type
Param1	String
Param2	Int

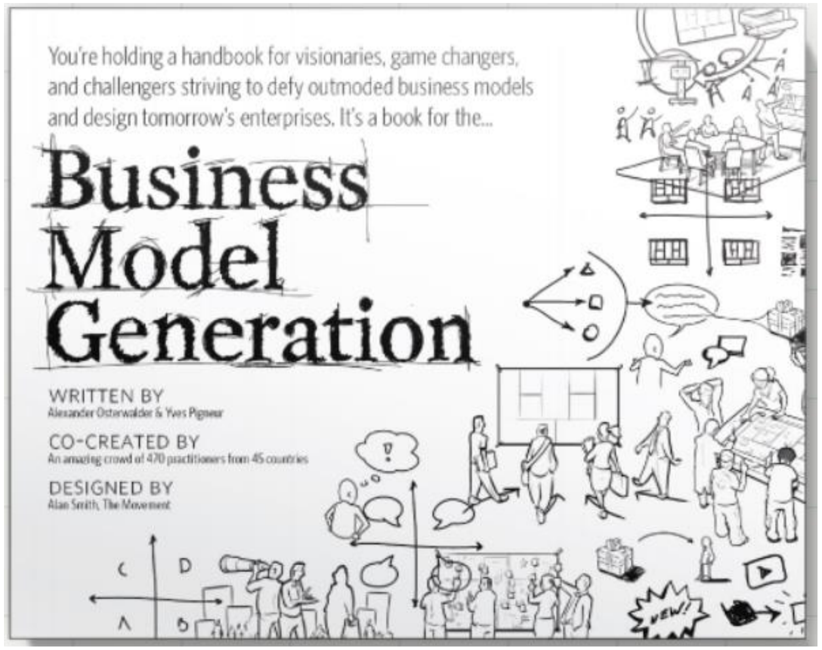
AtomicViewComponent

Property	Value
Id	
In Flows	
In Parameters	
Name	MboxList
Out Parameters	

The metamodel – 1: Core - Content



«metaclass» Annotation (ifml.core) +text : String
+annotations 0..*
+element 1
«metaclass» Element (ifml.core) +id : String
+element 1
0..* +constraints
«metaclass» Constraint (ifml.core)
+interactionFlowModelElements 0..*
«metaclass» InteractionFlowModelElement (ifml.core)
«metaclass» NamedElement (ifml.core) +name : String
«metaclass» Element (UML Standard Profile.UML2 Metamodel.Classes.Kernel)
0..* +interactionFlowModelElements
+element 0..*
+interactionFlowModel 1
1 +contentModel
«metaclass» InteractionFlowModel (ifml.core)
+interactionFlowModel 1
1 +ifmlModel
«metaclass» IFMLModel (ifml.core)
+ifmlModel 1
1 +contentModel
«metaclass» ContentModel (ifml.core)
+interactionFlowModel 1
+ifmlModel 1
+contentModel 1
+viewPoints 0..*
«metaclass» ViewPoint (ifml.core)
+interactionFlowModelViewPoints 0..*



You're holding a handbook for visionaries, game changers, and challengers striving to defy outmoded business models and design tomorrow's enterprises. It's a book for the...

Business Model Generation

WRITTEN BY
Alexander Osterwalder & Yves Pigneur

CO-CREATED BY
An amazing crowd of 470 practitioners from 45 countries

DESIGNED BY
Alan Smith, The Movement

> 1 million copies sold

Obra protegido por derechos de autor
Tiene en sus manos una guía práctica dirigida a visionarios, a personas con ganas de asumir nuevos retos que luchan por dejar atrás modelos de negocio anticuados para diseñar las empresas del futuro. Es un libro para la...

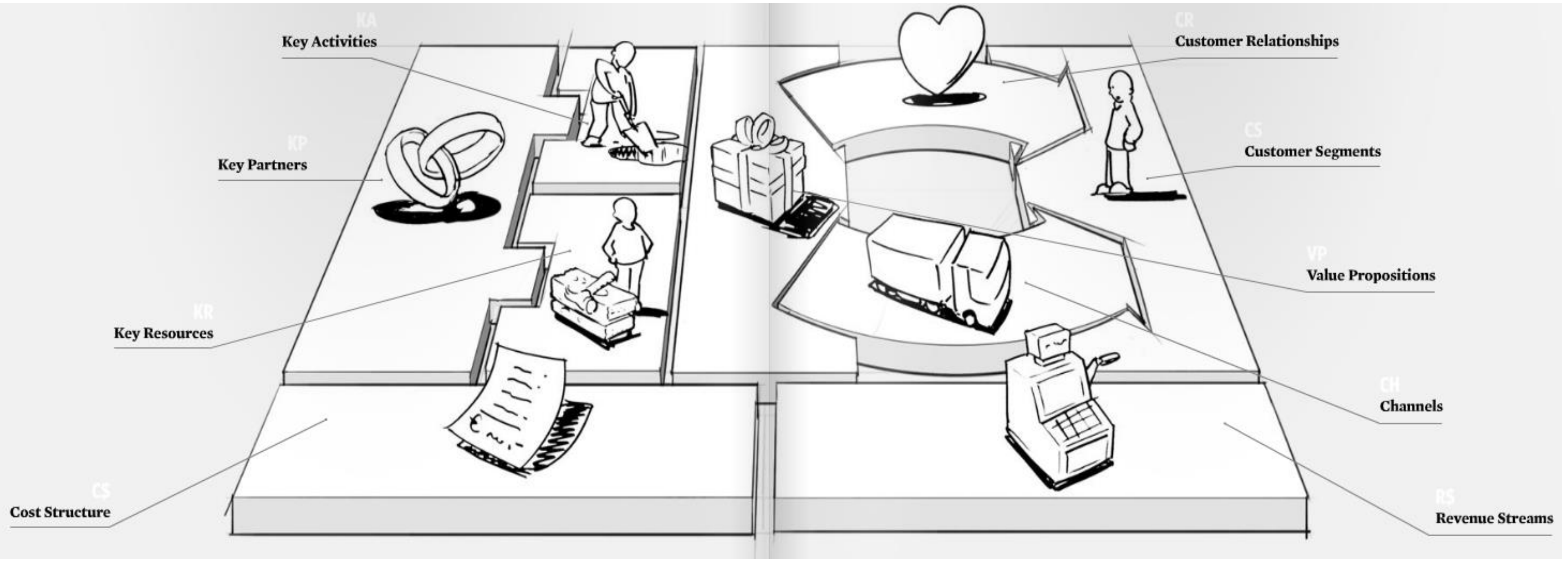
Generación de modelos de negocio

Alexander Osterwalder & Yves Pigneur

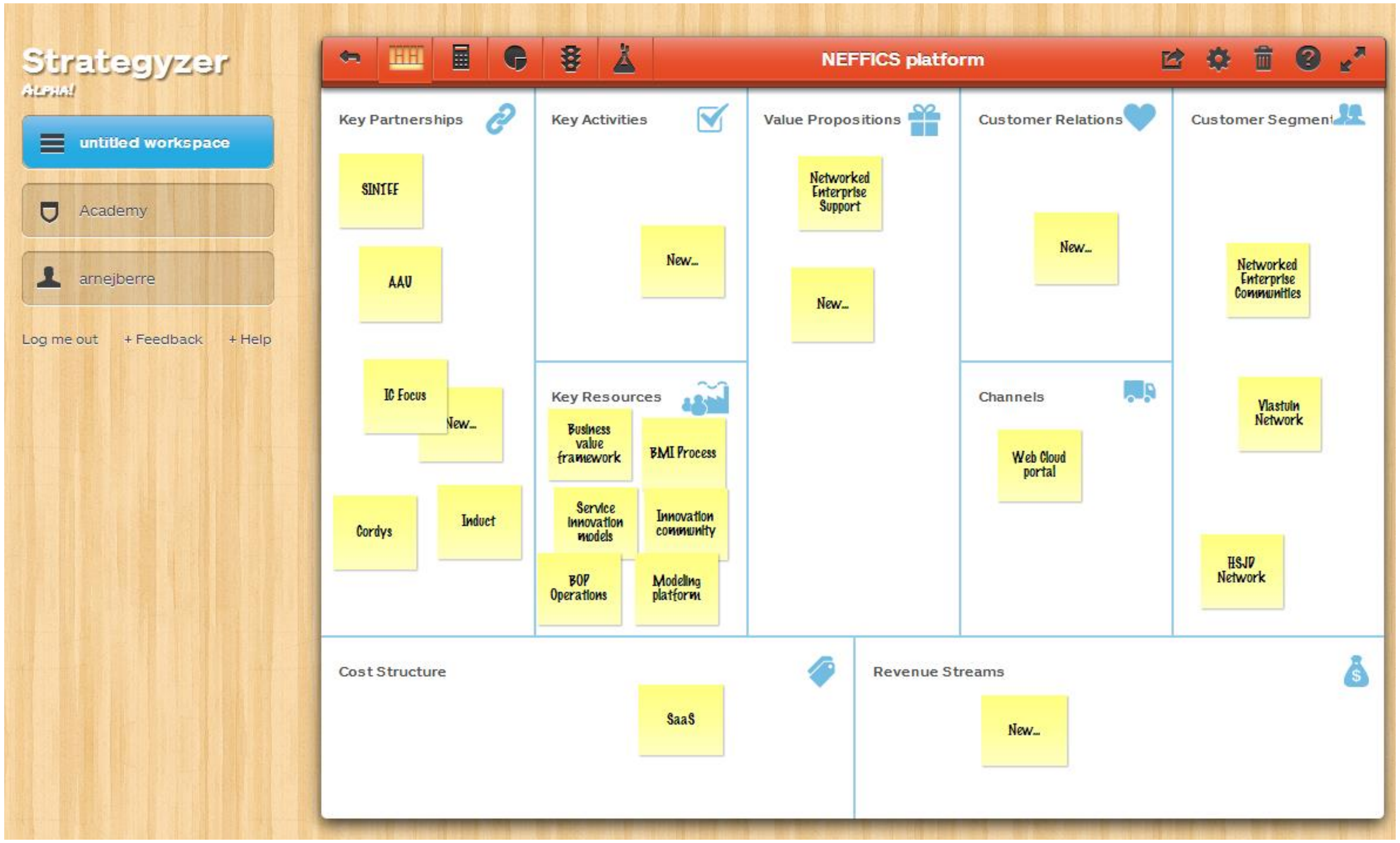
30 languages

Businss Model Innovation

The Business Model Canvas

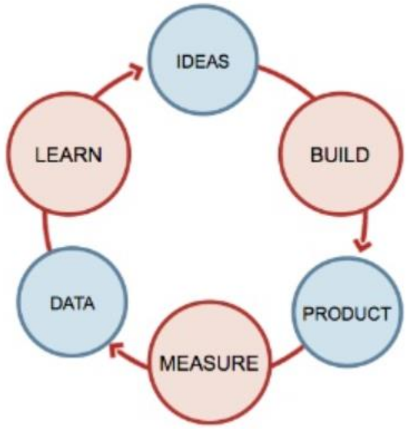


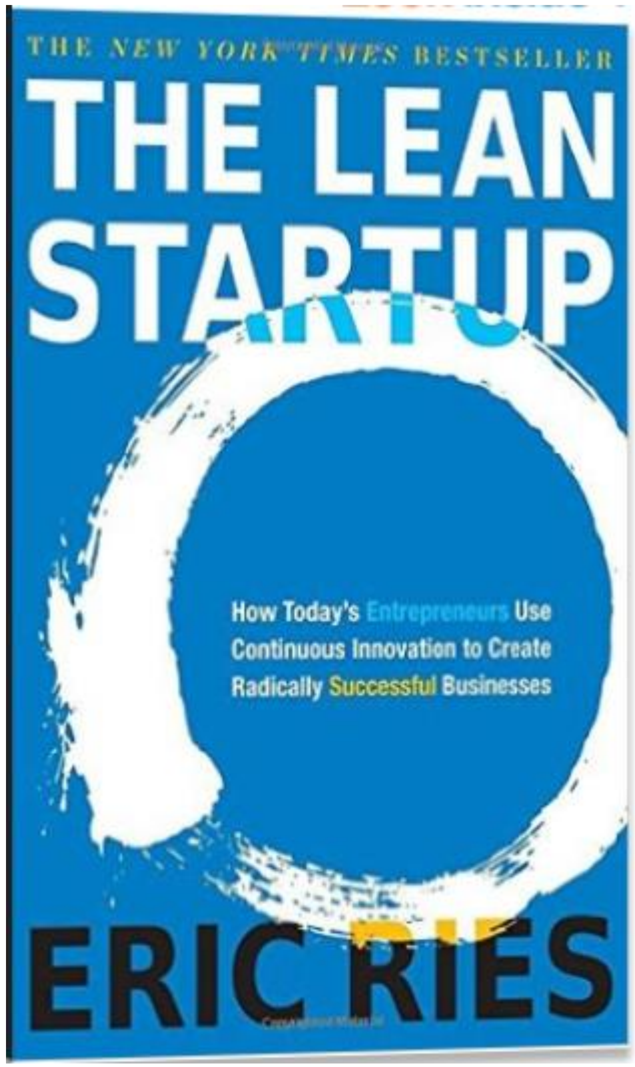
Strategyzer (Osterwalder)



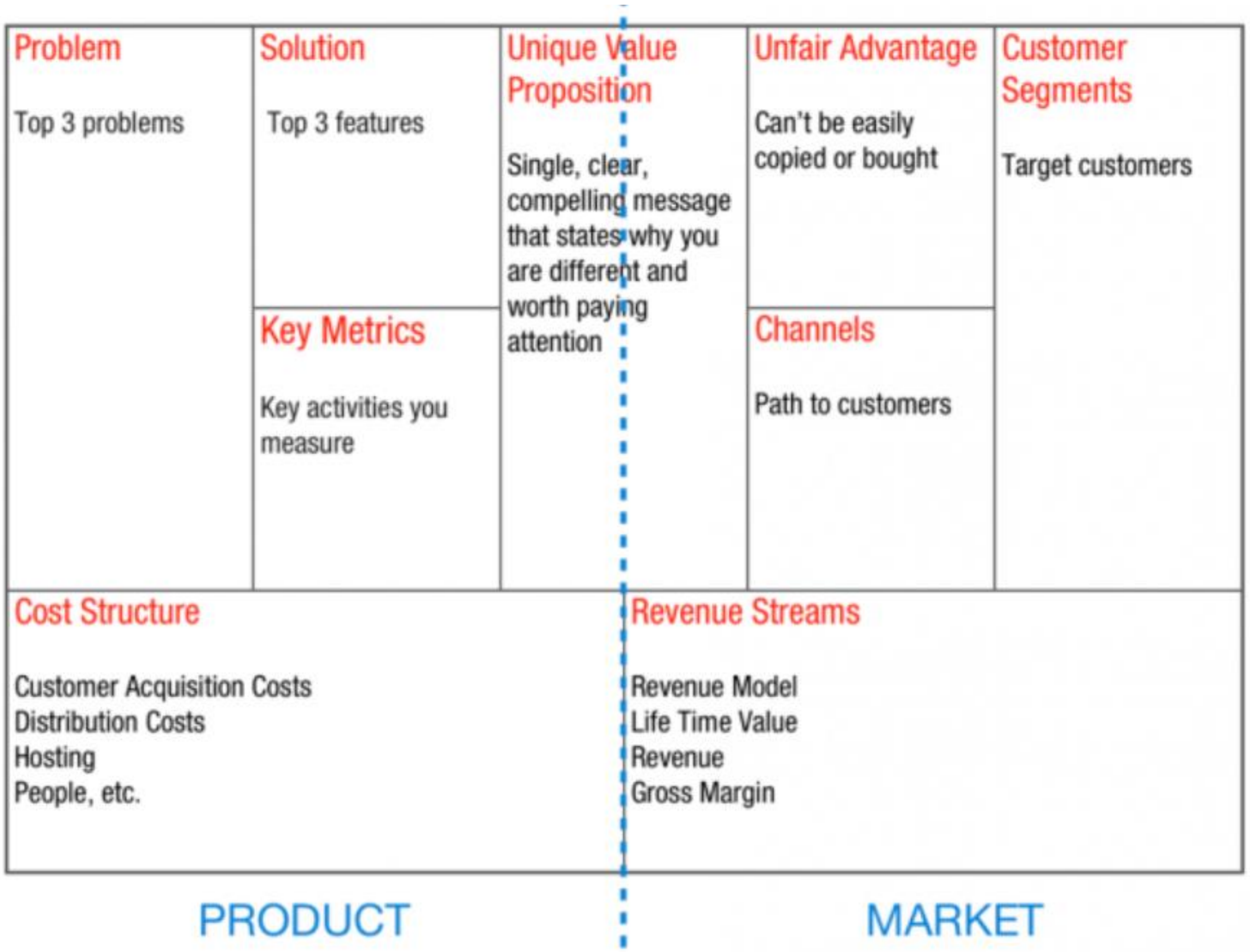
Lean Startup

The core of Lean
is iteration.



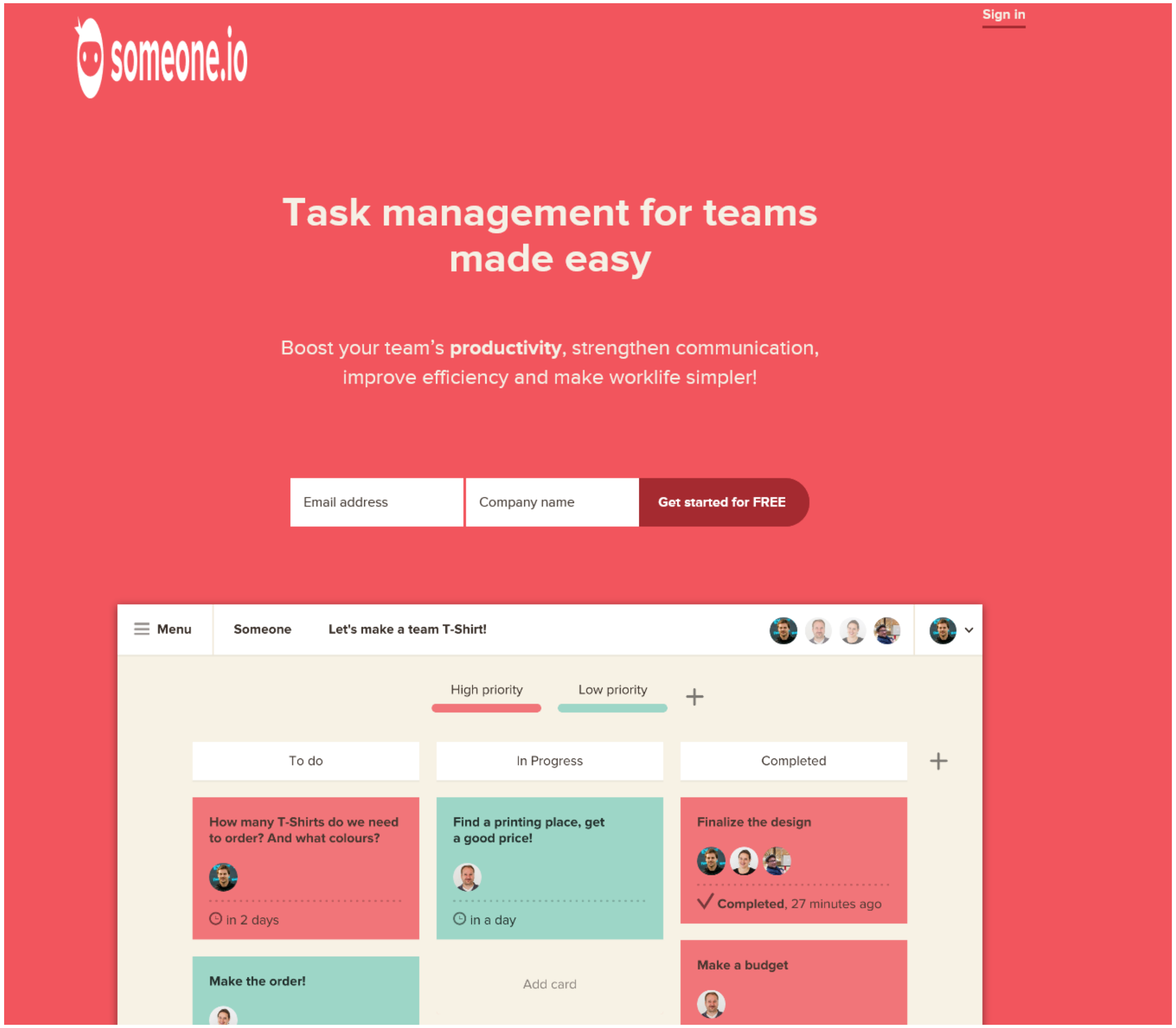

THE NEW YORK TIMES BESTSELLER
THE LEAN STARTUP
How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses
ERIC RIES

Lean Canvas



Problem
Top 3 problems
Solution
Top 3 features
Key Metrics
Key activities you measure
Unique Value Proposition
Single, clear, compelling message that states why you are different and worth paying attention
Unfair Advantage
Can't be easily copied or bought
Channels
Path to customers
Customer Segments
Target customers
Cost Structure
Customer Acquisition Costs
Distribution Costs
Hosting
People, etc.
Revenue Streams
Revenue Model
Life Time Value
Revenue
Gross Margin
PRODUCT
MARKET

UpWave.io – for Scrum



Sign in

someone.io

Task management for teams made easy

Boost your team's **productivity**, strengthen communication, improve efficiency and make worklife simpler!

Lecture January 30, 2017

IFML for User Interfaces for Web Apps/Portals and SmartPhone Apps (using WebRatio)

Lecture February 6, 2017

Business Architecture, Business Engineering and Business Model Canvas, Lean Canvas – user stories and use cases (for Oblig Smart Building)