Rapid Ontology Development (RODE) With PIKE

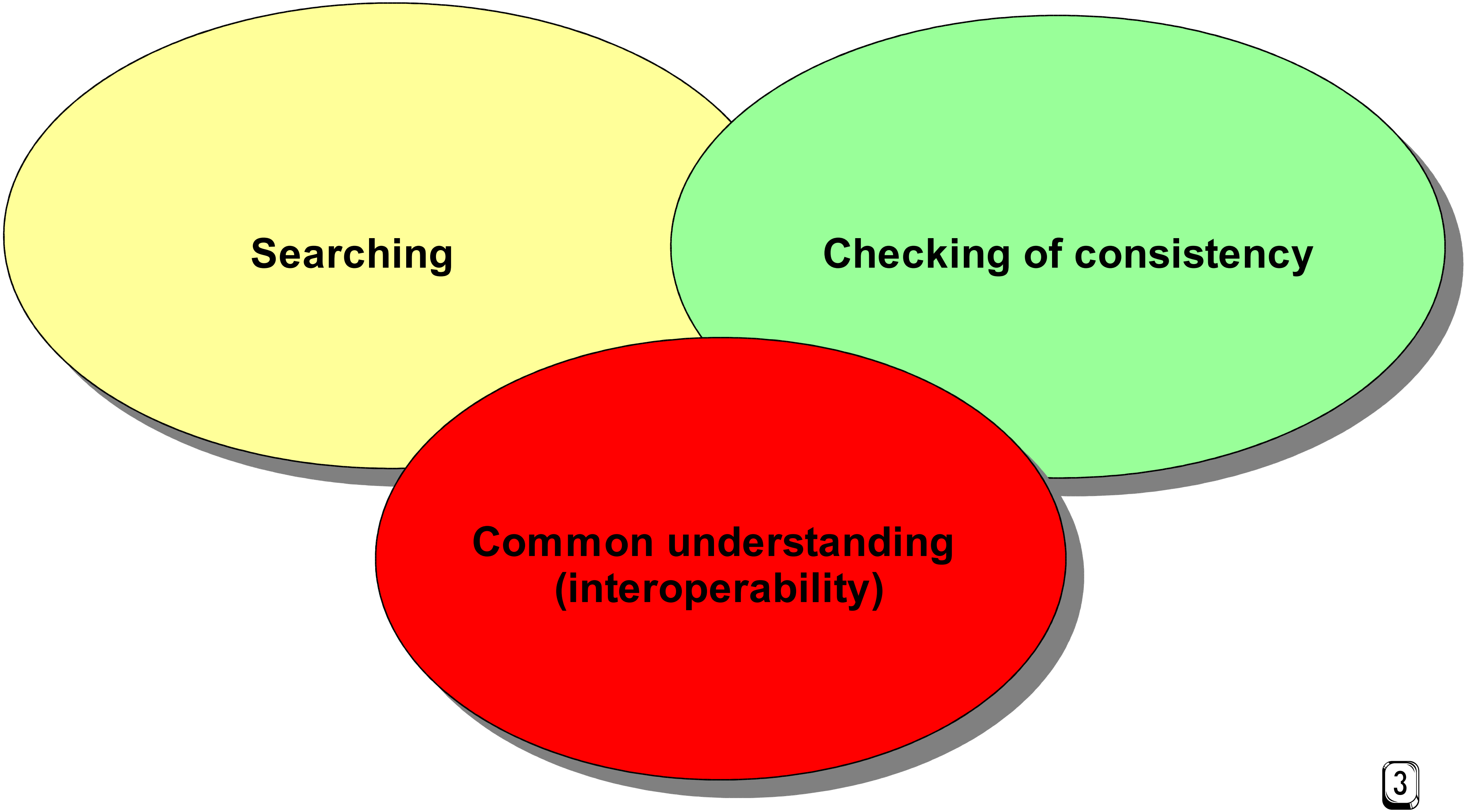
Prof. Uwe Assmann

Martin Nilsson, Leif Stensson, Marcus Comstedt

Research Center for Integrational Software Engineering (RISE)

PELAB, IDA

The Future Semantic Web

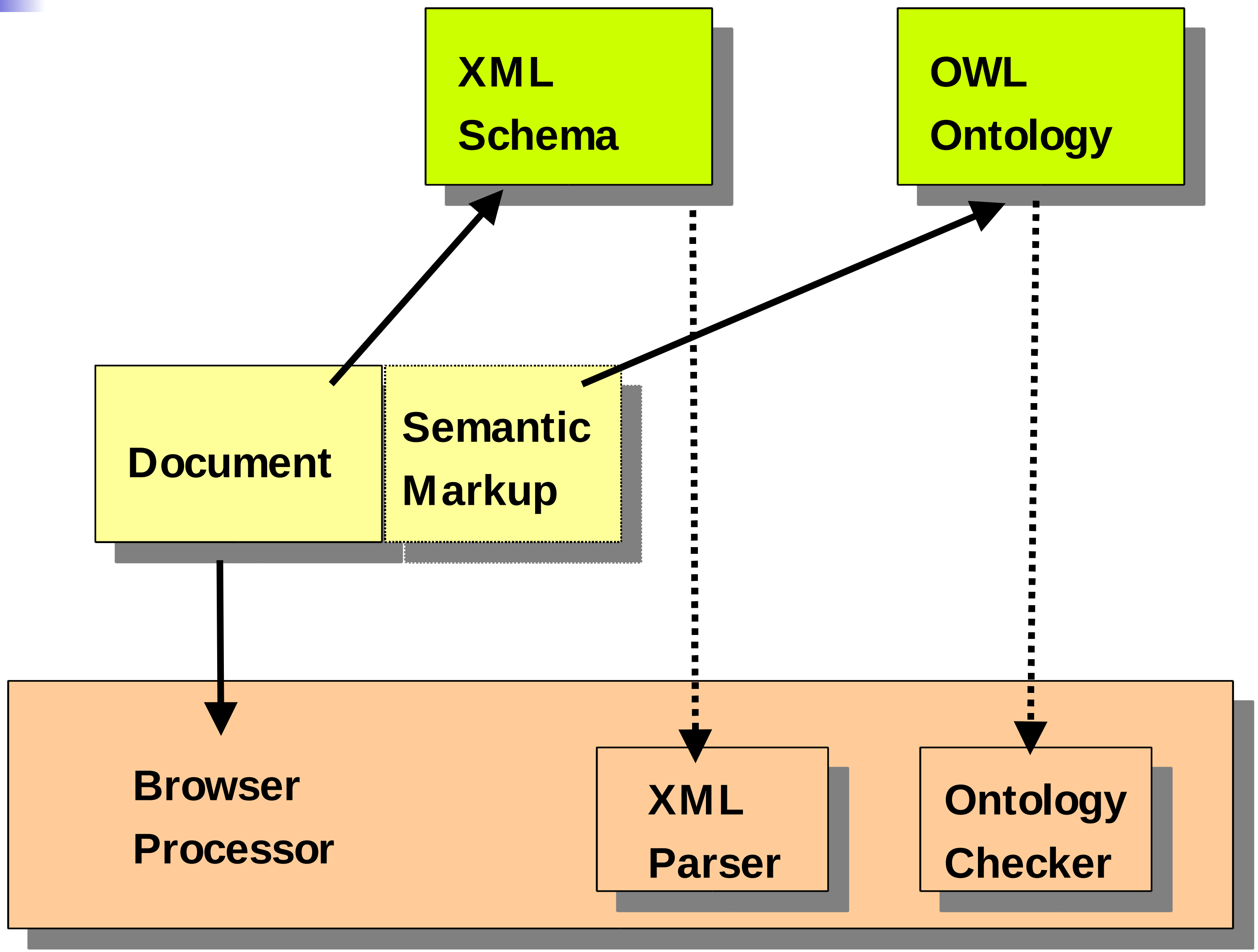


Searching

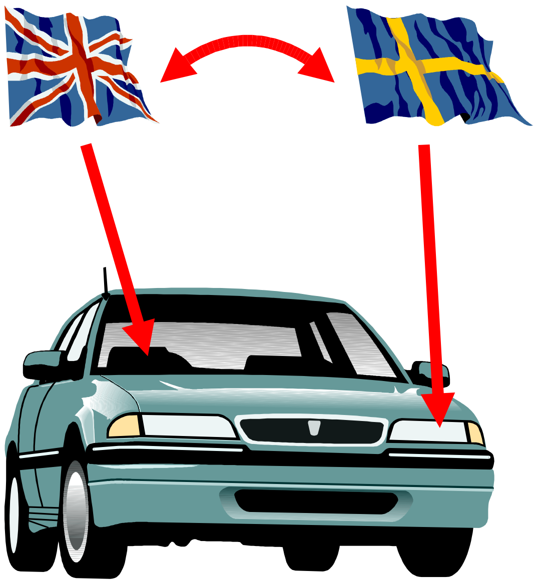
Checking of consistency

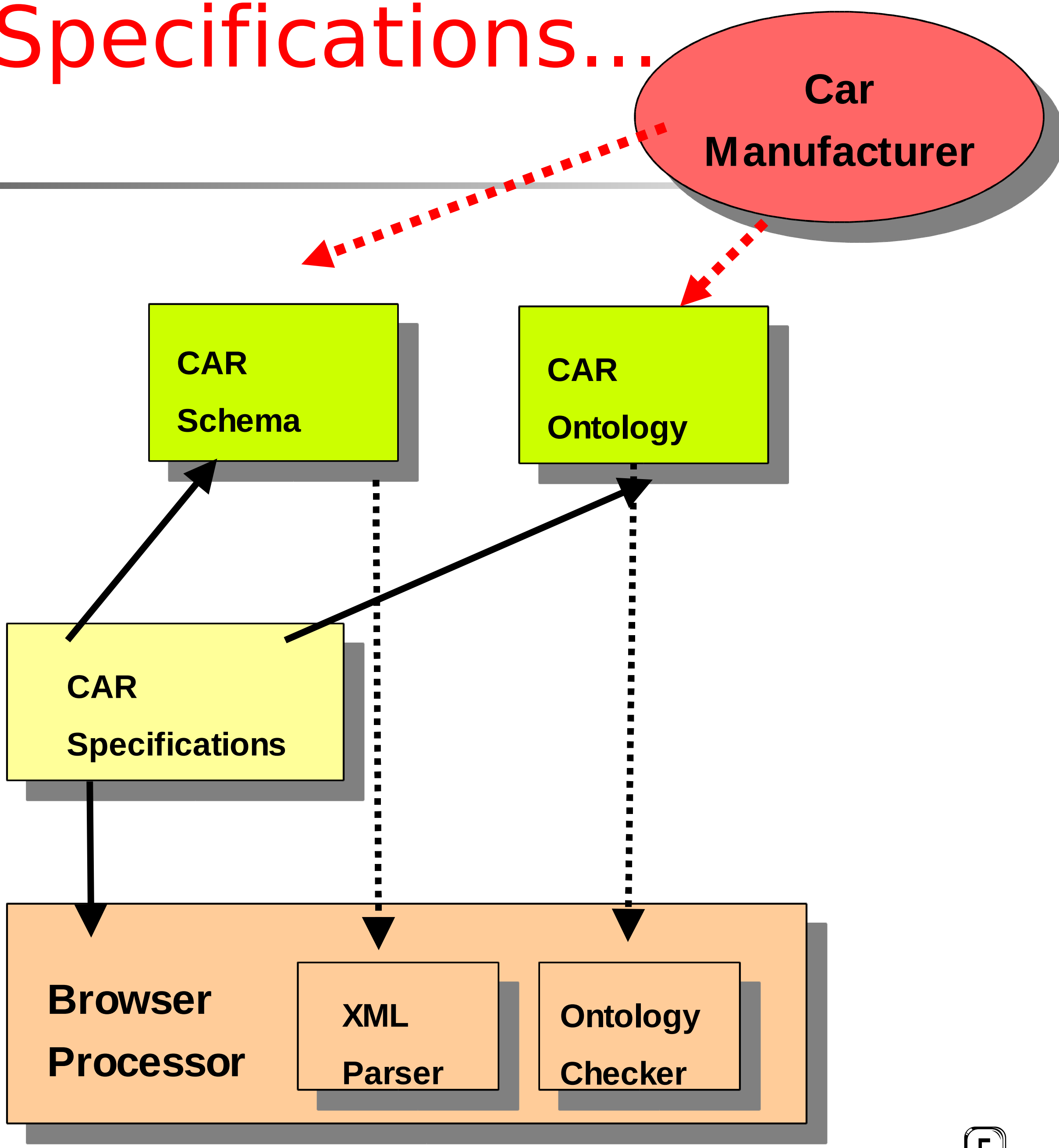
Common understanding
(interoperability)

Standardized Document Processing Architecture



Car Data Specifications…





Car Manufacturer

CAR Schema

CAR Ontology

CAR Specifications

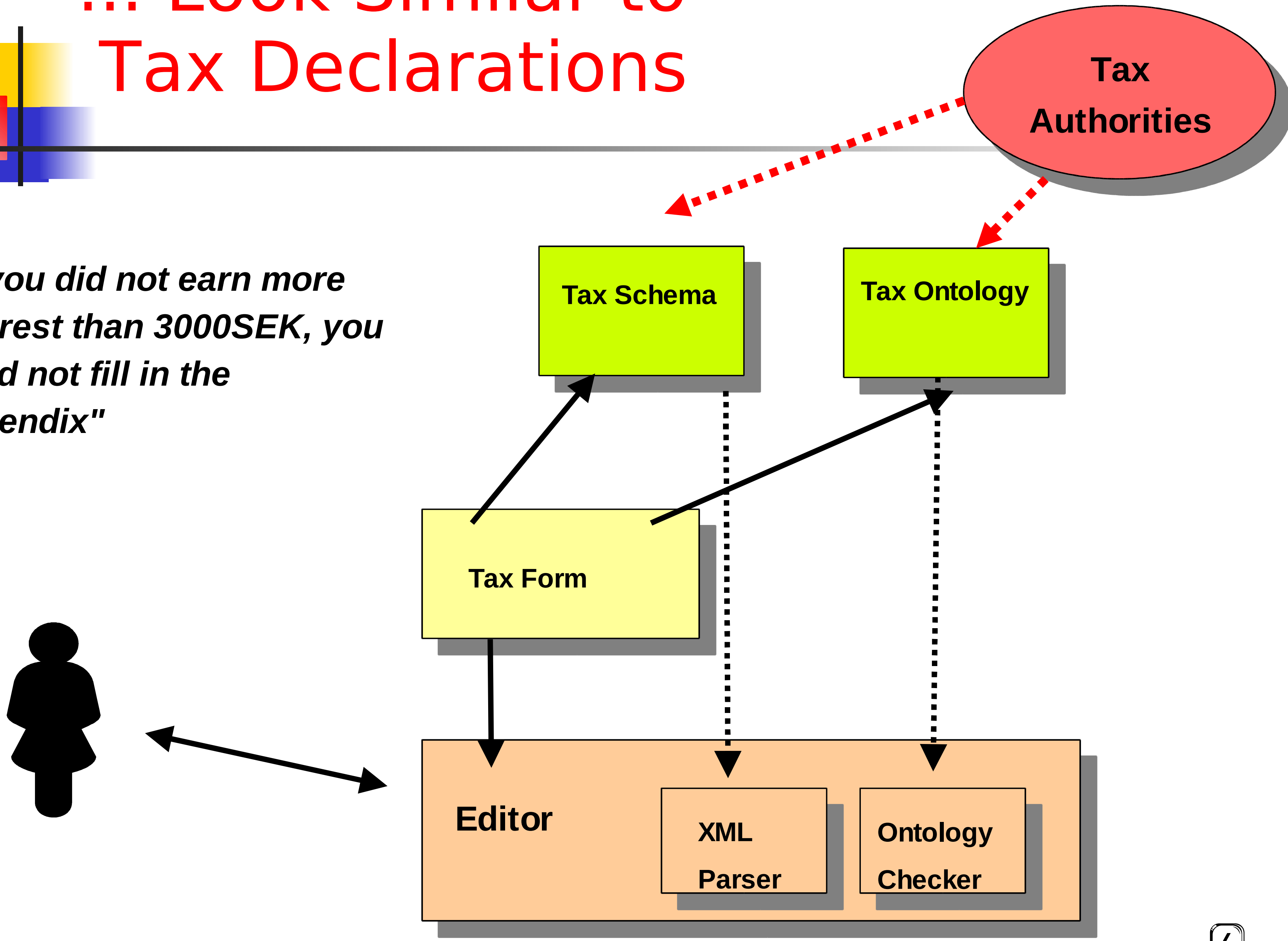
Browser Processor

XML Parser

Ontology Checker

... Look Similar to Tax Declarations

"If you did not earn more interest than 3000SEK, you need not fill in the appendix"



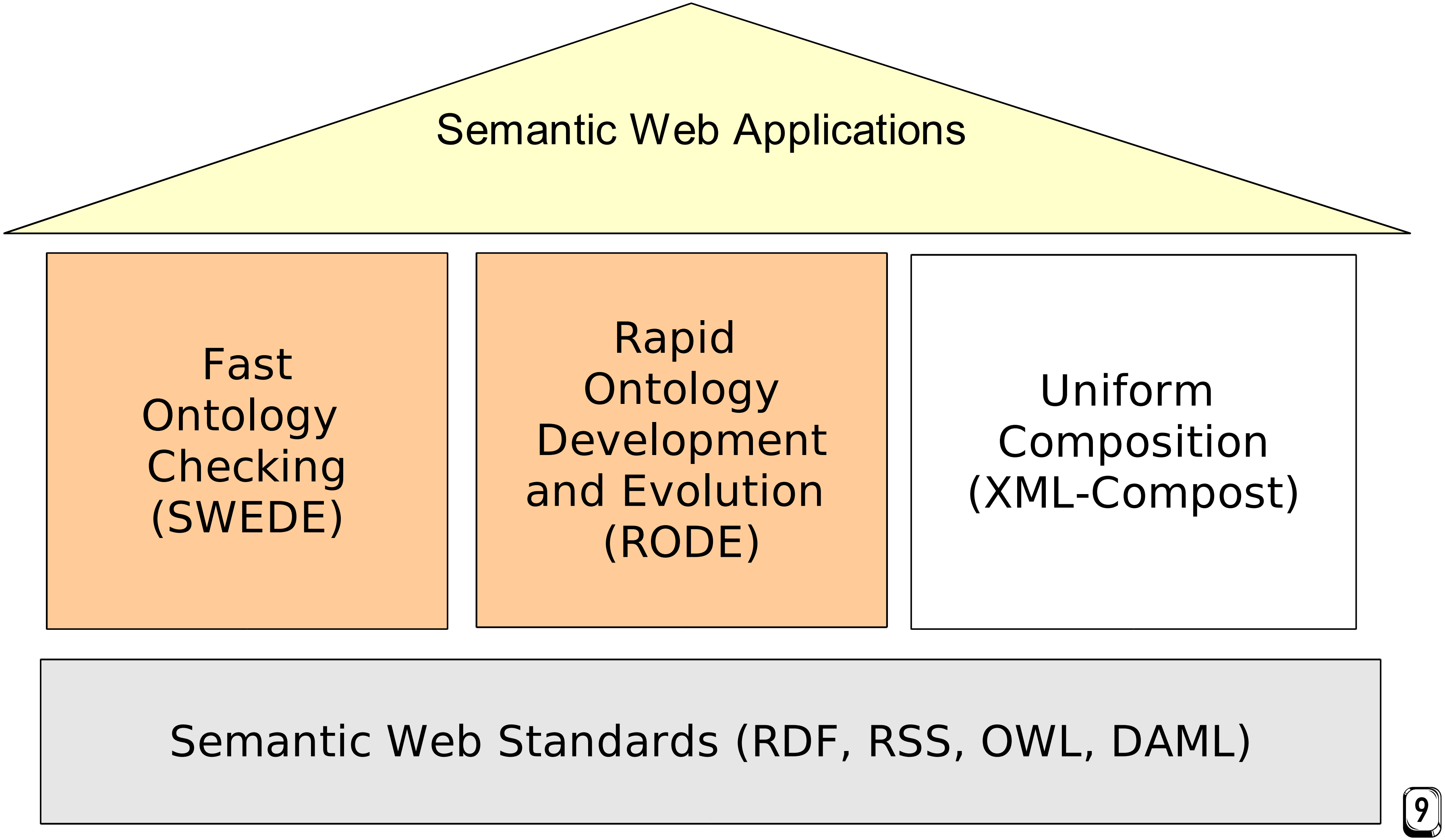
The Problems of the Future Semantic Web

- Ontology-based development - how?
  - The Semantic Web has rather static ontologies (models), but in software engineering, everything flows
  - Models *change*
  - Models are developed out of each other in different abstraction levels
- Slow document checking
  - How to load a document (OWL instance) of 200MB into Prolog?
    - Conversion time
    - Memory consumption
    - Speed of checking

The Solutions for the Future Semantic Web

- Rapid Ontology Development (RODE)
  - Brigde OWL ontologies with a RAD language (rapid ontology engineering environment with Pike)
  - Demonstrator RODE
- Development environment with fast in-line ontology checking
  - Translate an OWL ontology into the classes as check code
  - Demonstrator SWEDE environment:

Semantic Web Applications



Semantic Web Applications

Fast Ontology Checking (SWEDE)

Rapid Ontology Development and Evolution (RODE)

Uniform Composition (XML-Compost)

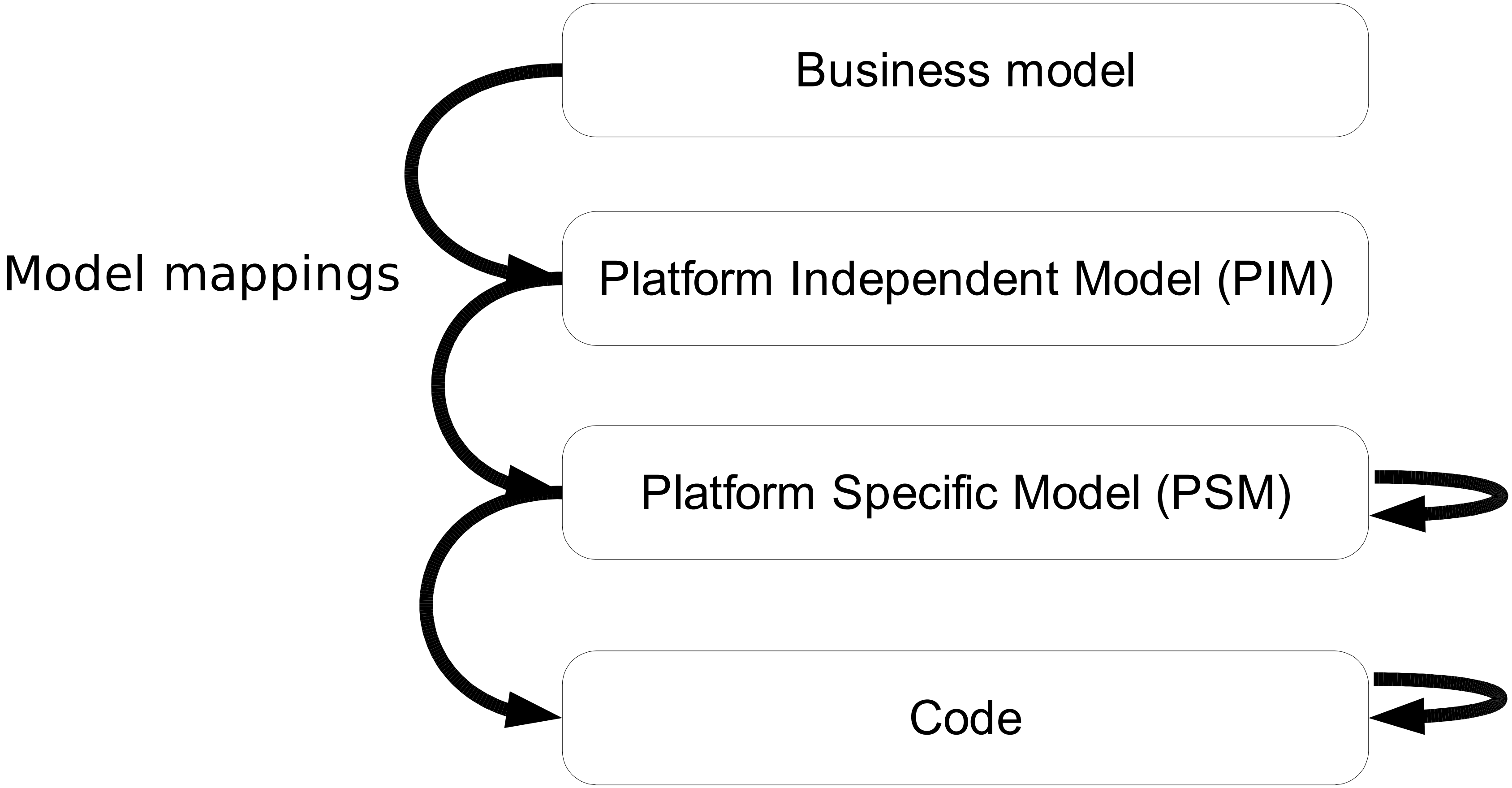
Semantic Web Standards (RDF, RSS, OWL, DAML)

Rapid Ontology Development (RODE)

Model-Driven Architecture (MDA)

- MDA (http://www.OMG.org/mda) attracts engineers
- Split the models for systems software into platform-independent and platform specific models (PIM vs. PSM).
  - The PIM focus on the logical architecture
  - The PSM adds platform specific details and timing constraints.
- Promises to simplify the designs
  - Derive implementation models from design models (semi-) automatically.
- However, tool support for MDA is missing
- OMG expects MDA to be their major activity area for the next 10 years

MDA



Model mappings

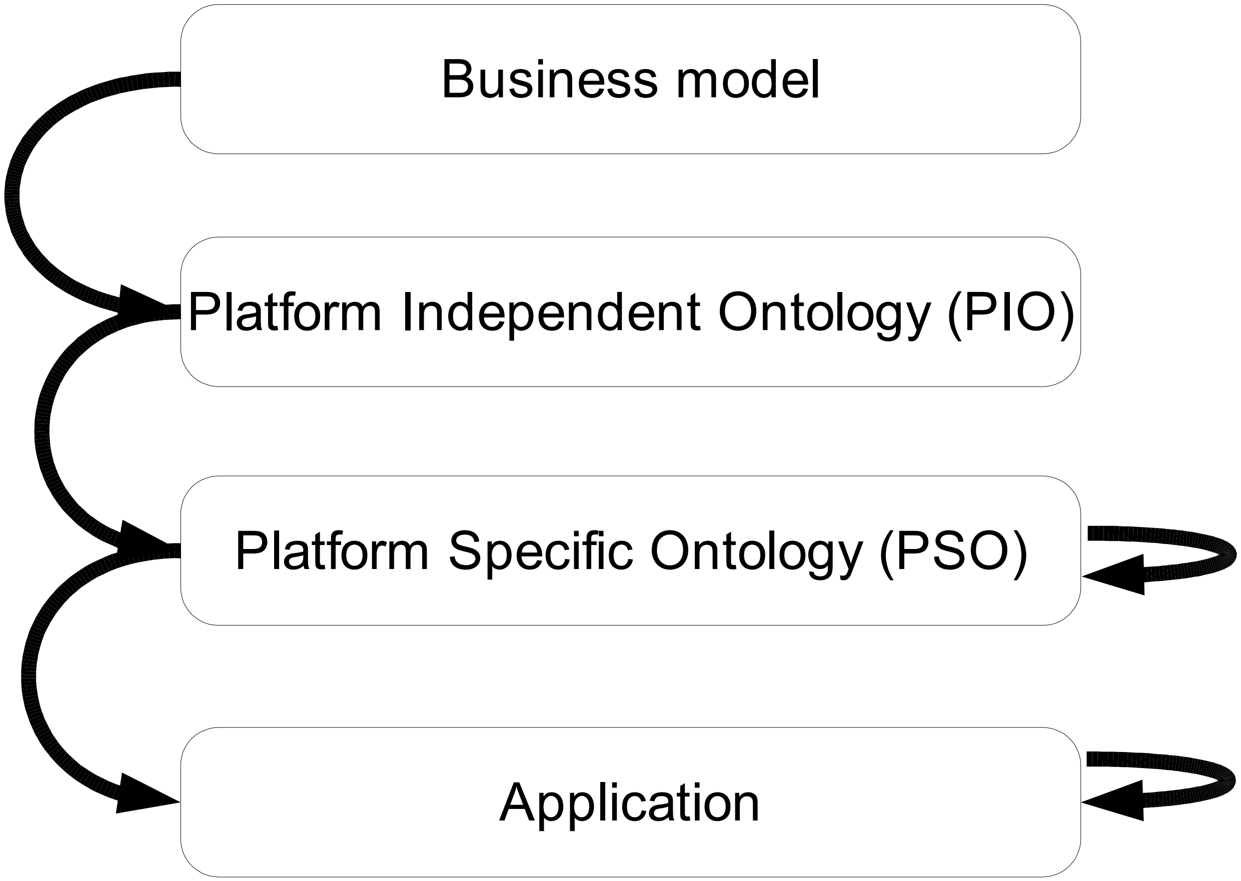
Business model

Platform Independent Model (PIM)

Platform Specific Model (PSM)

Code

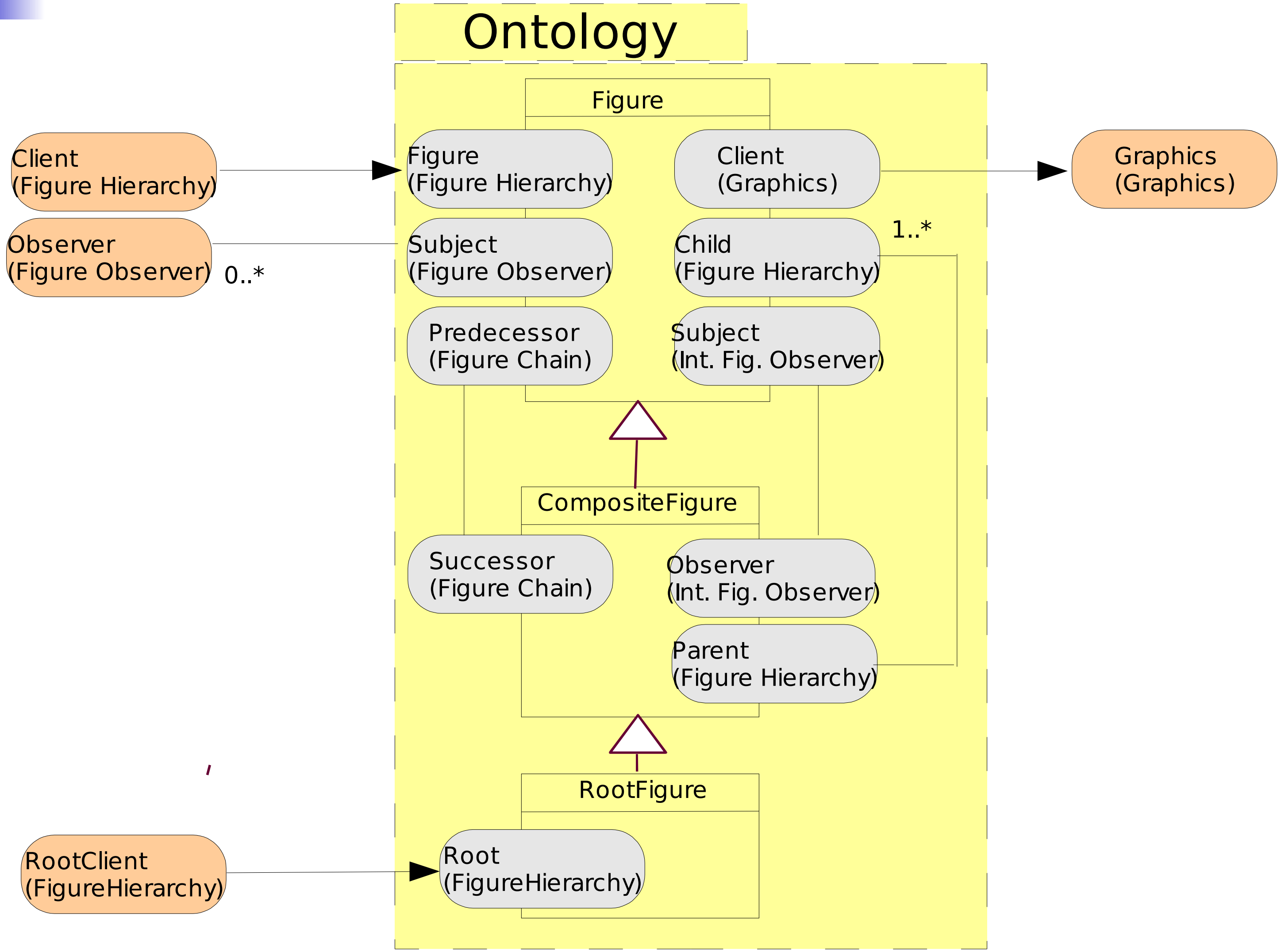
MDA for Ontology Development


Business model
Platform Independent Ontology (PIO)
Platform Specific Ontology (PSO)
Application

Rapid Ontology Development

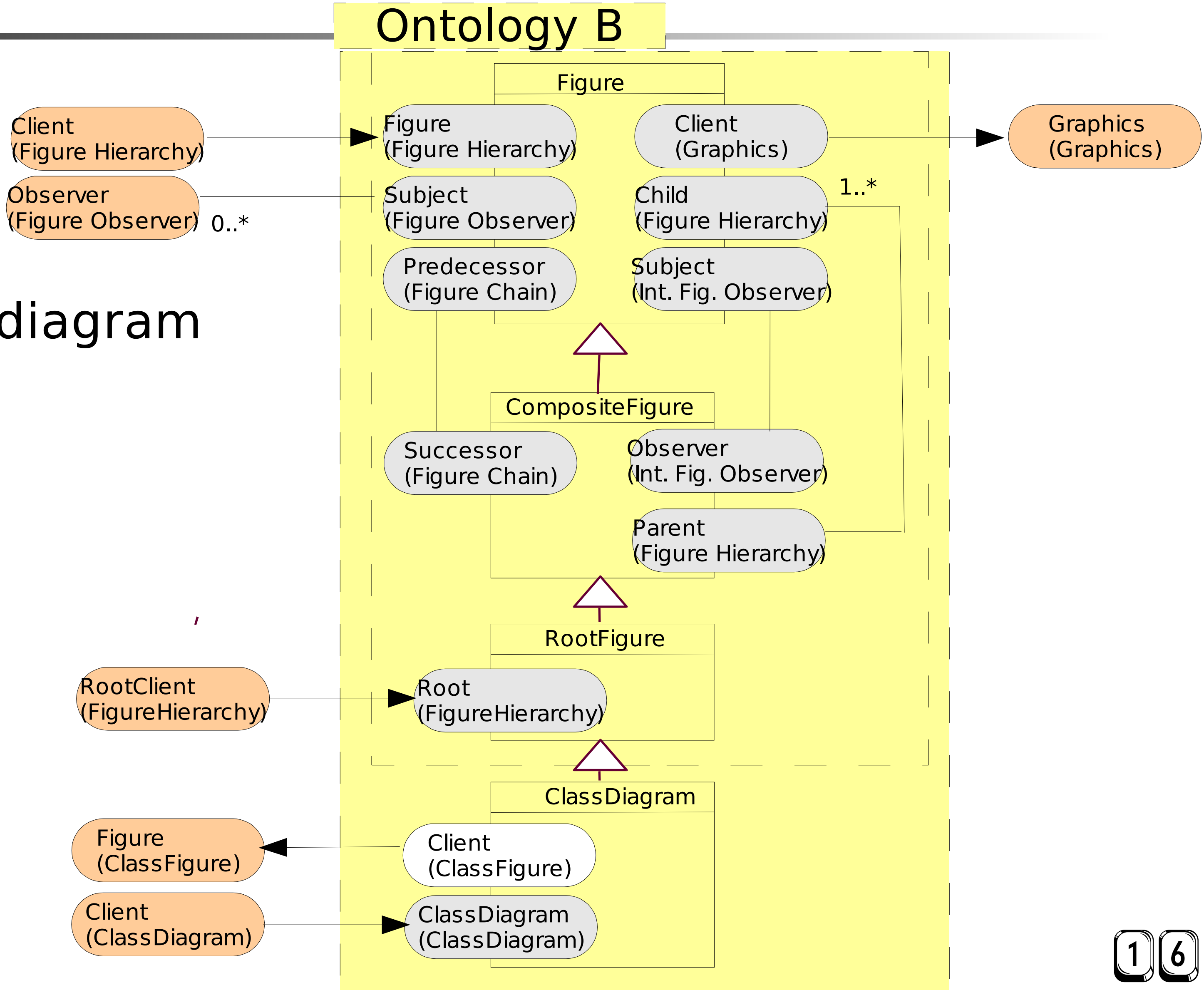
- Problem: Several types of ontologies will be needed in the development process
- Abstract ontologies, platform independent
- Detailed ontologies, platform specific
- Or: design ontologies vs implementation ontologies
- Ontology engineering will be a discipline

RODE Example
A Platform-Independent Ontology

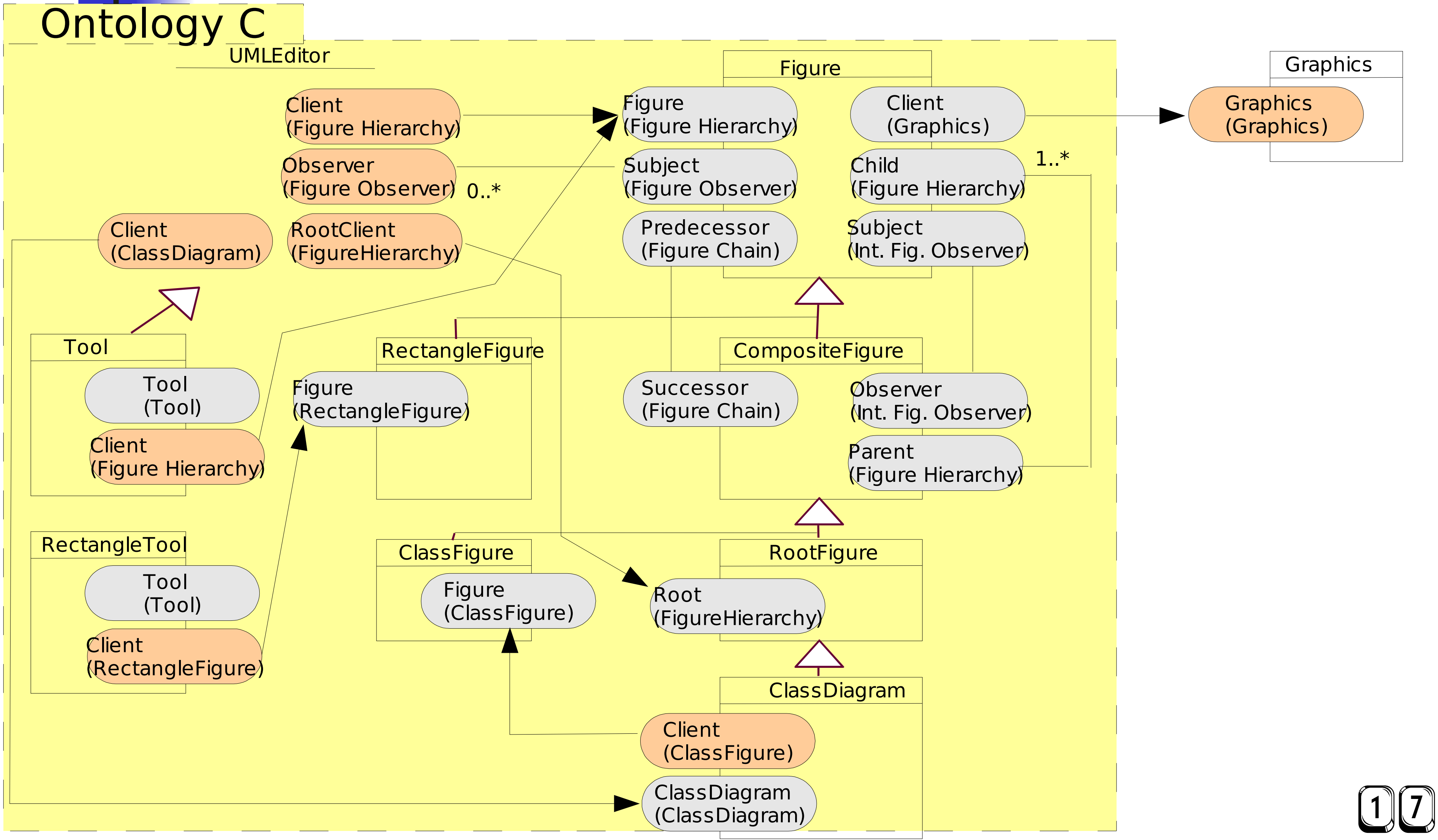


A Derived Application-Oriented Ontology

- For class diagram editors


Ontology B
Figure
Client (Figure Hierarchy)
Figure (Figure Hierarchy)
Client (Graphics)
Graphics (Graphics)
Observer (Figure Observer)
0..*
Subject (Figure Observer)
Child (Figure Hierarchy)
1..*
Predecessor (Figure Chain)
Subject (Int. Fig. Observer)
CompositeFigure
Successor (Figure Chain)
Observer (Int. Fig. Observer)
Parent (Figure Hierarchy)
RootFigure
RootClient (FigureHierarchy)
Root (FigureHierarchy)
ClassDiagram
Figure (ClassFigure)
Client (ClassFigure)
Client (ClassDiagram)
ClassDiagram (ClassDiagram)

A Derived Ontology for UML Editors



What Are Platforms In MDA?

- Abstract machines
  - Libraries, such as JDK, .NET
- Implementation languages
  - Java, Eiffel, C#
- Component models
  - CORBA, etc.
- Set of predefined types (vocabulary)
- Ontology of a domain (e.g., medicine)
- Constraints
  - Time
  - Memory
  - Energy
- Platforms are described by UML profiles

What Are UML Profiles in MDA?

- UML dialect of a platform
  - With new stereotypes and tagged values
  - With metamodel
- Domain specific languages
  - With own vocabulary
  - Every entry in metamodel is a term
- Examples
  - EDOC Enterprise Distributed Objects Computing
  - Middleware: Corba, .NET, EJB
  - Embedded and real time systems: time, performance, schedulability

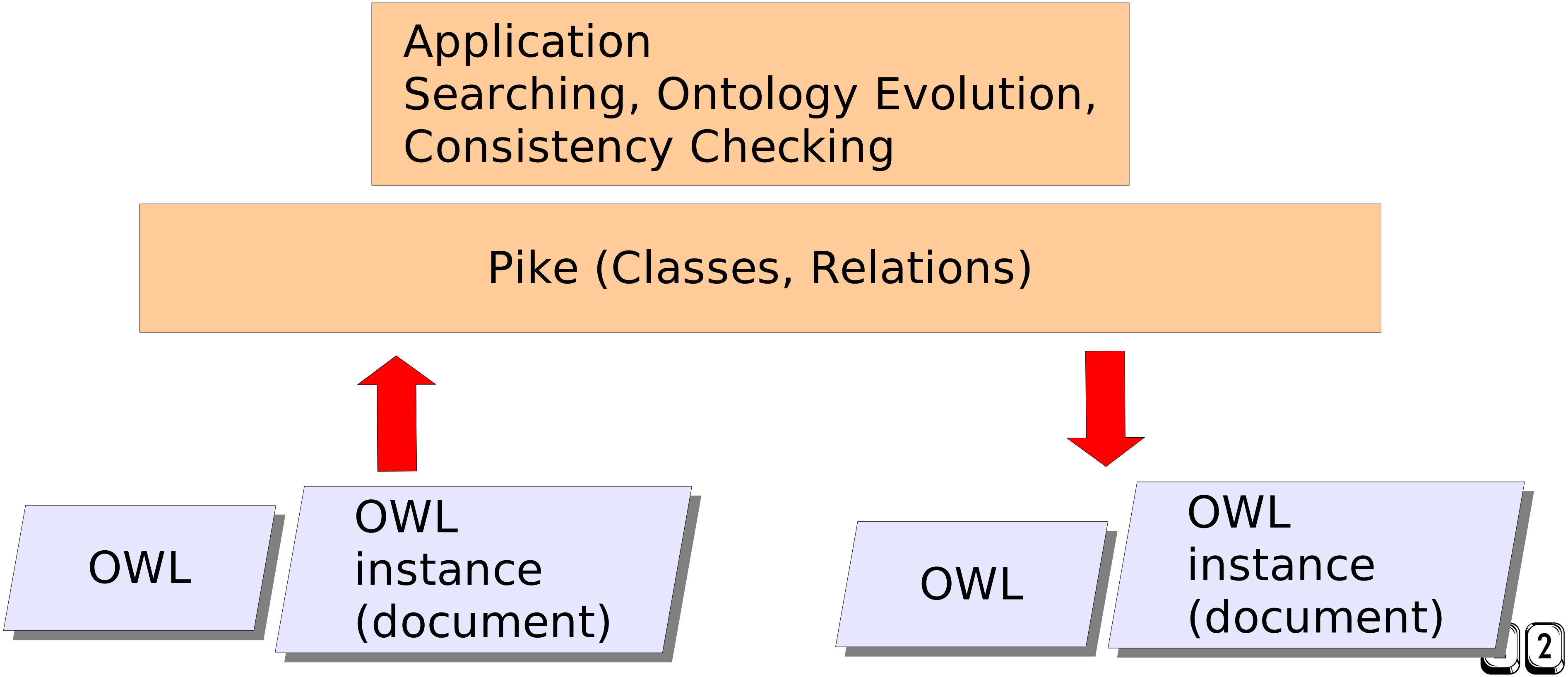
Profiles Are…

- Ontologies in UML
  - If domain is large enough
  - If there are enough users
- Also profiles should be represented in OWL

Rapid Ontology Development and Evolution (RODE)

- Required for Rapid Ontology Development is a powerful RAD language for ontologies
- Results in Rapid Ontology Development and Evolution, (RODE)
- Idea: evolve OWL as Pike data
  - Based on Pike Relation module
  - RSS syndication and RDF processing works
  - OWL soon (end of the year). Then, semantic searching will be possible
  - Full RODE next year

The RODE



Application
Searching, Ontology Evolution,
Consistency Checking

Pike (Classes, Relations)

OWL

OWL
instance
(document)

OWL

OWL
instance
(document)

MDAFA Conference

- June 2004

Why Pike is Suited

- Multiple inheritance
- Mixin inheritance
- Powerful data types
- Iteration concepts

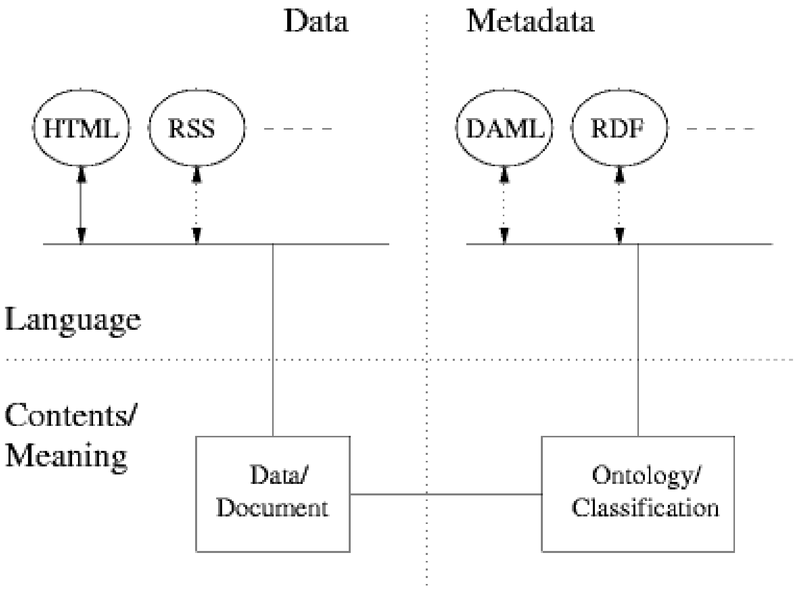
PIKE

- Multiple inheritance
- Open definitions
- Global Relations with inheritance
- Instance lists

OWL

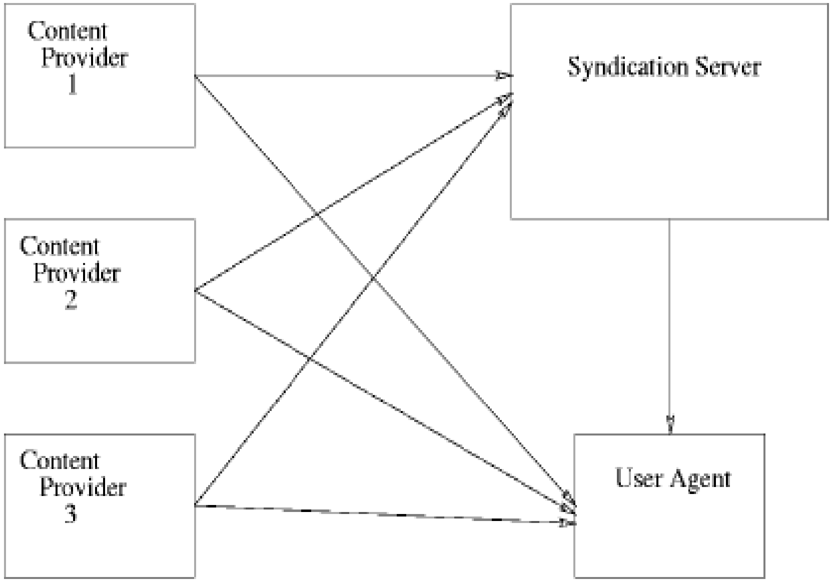
RODE: Semantic Data on the Web (RSS Syndication)

- HTML contain basic documents
- RSS helps making content available
- DAML provides for describing classifications
- RDF provides for describing relations



Syndication

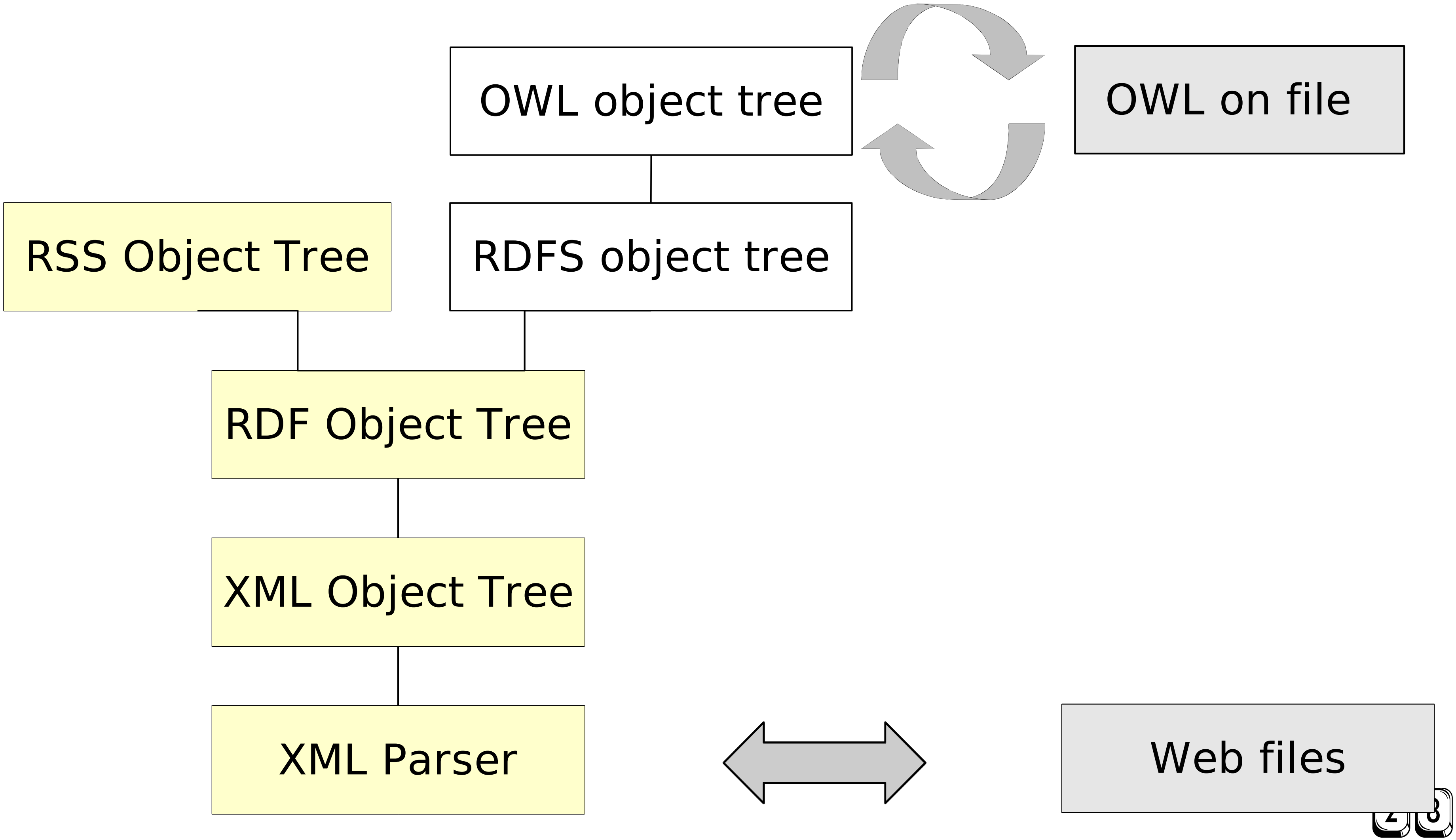
- Data provided by different content provides
- Data used either directly by user, or by intermediate syndication servers that provide the user with the data.



RODE and RSS: Really Simple Syndication

- RSS is an RDF format
- RSS supports Dublin Core metadata markup
- RODE can read, modify and write RSS data
- Users can search on RSS
  - RSS data is a standard data structure, that can be manipulated symbolically in Pike
  - Search and iteration, split, and manipulation is very simple
- RODE enables simple evolution of RSS data

RODE Components



OWL object tree

OWL on file

RSS Object Tree

RDFS object tree

RDF Object Tree

XML Object Tree

XML Parser

Web files

RODE - XML Object Tree

- Offers DOM interface
- Capable of all XSLT transforms

RODE - RDF Object Tree

- Reads and writes XML, 3-tuple and N-triple serialization.

```
> object r = Standards.RDF();
> r->parse_xml(#"<RDF
  xmlns='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:s='http://description.org/schema/'>
  <Description about='http://www.w3.org/Home/Lassila'
               s:Creator='Ora Lassila' /></RDF>");
(1) Result: Standards.RDF(1)
> r->get_n_triples();
(2) Result: "<http://www.w3.org/Home/Lassila>
   <http://description.org/schema/Creator> \"Ora Lassila\" .\n"
```

RODE - RDF Object Tree

- Interactive object manipulation.

```
> object r = Standards.RDF();
> object unnamed_resource = r->Resource();
> r->add_statement( “http://a.com/”, “http://b.com/”,
  unnamed_resource);
(1) Result: Standards.RDF(1);
> r->find_statements(0,0,unnamed_resource);
(5) Result: ({ /* 1 element */
              ({ /* 3 elements */
                  RDF.URIResource(<http://a.com/>),
                  RDF.URIResource(<http://b.com/>),
                  RDF.Resource(_:Resource1)
                })
            })
```

RODE - RDF Object Tree

- Supports set operations between RDF domains.

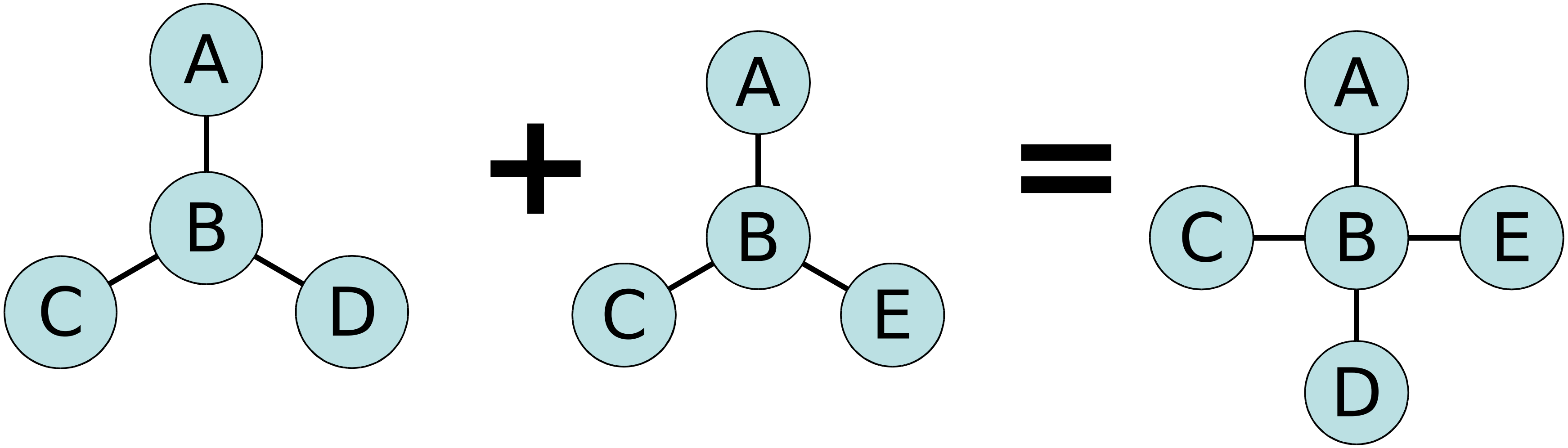
```
> object a=Standards.RDF();

> object b=Standards.RDF();

> a->parse_xml(Stdio.read_file
   ("example_a.rdf"));
(1) Result: Standards.RDF(43)
> b->parse_xml(Stdio.read_file
   ("example_b.rdf"));
(2) Result: Standards.RDF(48)
> a|b;
(3) Result: Standards.RDF(86)
```

RODE - RDF Object Tree

- Supports set operations between RDF domains.



ROD - RSS Object Tree

- Simple RDF application
- Already deployed on several sites
- Real World data
  - e.g., from Runeberg server

Fast Ontology Checking - The SWEDE Framework

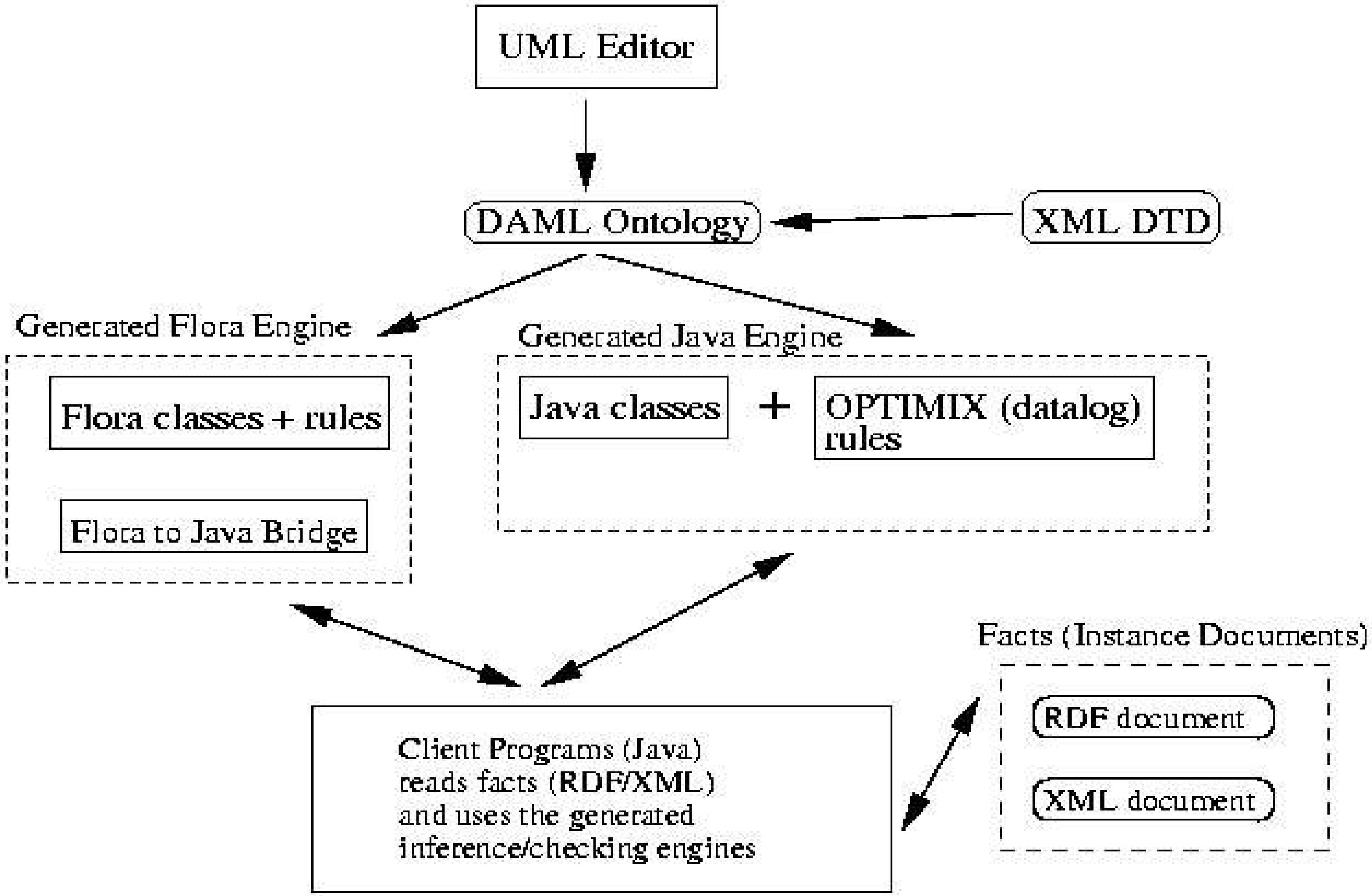
Ontology Development Environment (SWEDE)

- SWEDE (Semantic Web Development Environment) has two parts
  - Development of ontologies in an interactive way with UML tool and constraint editor
  - Checking of ontologies
    - Checking of documents and architectures vs a composition ontology
    - Generation of in-line checkers for applications
    - Faster than usual

Semantic Web Development Environment – SWEDE Goals

- Access ontologies as if they were standard UML models
- UML editing of ontologies
  - Reuse the UML standard for creating ontologies
- Ontology processing
  - Fast checking of documents against ontologies
  - Search on ontology-based data structures
  - Inference engine for constraint checking
- OWL2Optimix is a subtask of SWEDE
  - Goal: get a fast evaluator for OWL
  - Translate to Java and Optimix, one of the advanced compiler generation tools for Datalog

SWEDE-OWL2Optimix Architecture



UML Editor

DAML Ontology

XML DTD

Generated Flora Engine

Flora classes + rules

Flora to Java Bridge

Generated Java Engine

Java classes + OPTIMIX (datalog) rules

Facts (Instance Documents)

RDF document

XML document

Client Programs (Java)
reads facts (RDF/XML)
and uses the generated
inference/checking engines

PIKE Backend for Optimix

- To do..

Applications

Application: Semantic Web for Product Development

- Semantic Web for interoperability of tools in product development
- Industrial Supporters (case studies)
  - IFS (configuration management)
  - FOI (military tool interoperability)
  - FocalPoint
- Use RODE for developing ontologies
- And SWEDE to check them

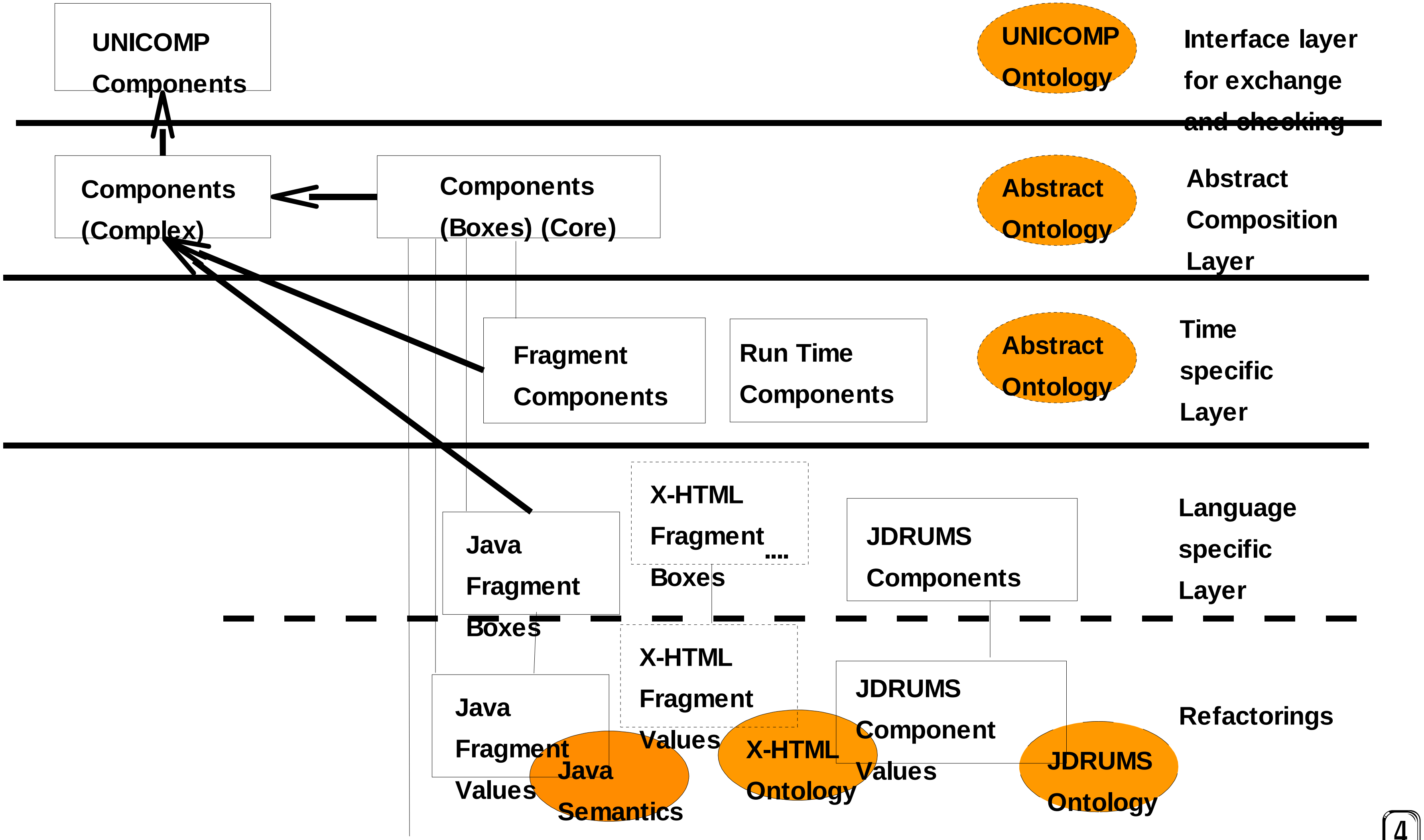
Application: Uniform Composition

- Uniform composition means to compose software and documents uniformly
- COMPOST 2.0 will be the first system
  - 1.0 was for Java only
- Changing
  - parsers
  - semantic descriptions
- Reusing
  - transformations
  - compositions
  - architecture

Ontology Controlled Uniform Composition

- Architectures (both for software and XML documents) can be described uniformly by compositions
  - And checked by a composition ontology
- Control the composition of components by the constraints of a *composition ontology*
- The ontology is split up over layers of the composition framework

Realizing Ontology Controlled Composition with the Layered Architecture of the COMPOST System



UNICOMP Components

UNICOMP Ontology

Interface layer for exchange and checking

Components (Complex)

Components (Boxes) (Core)

Abstract Ontology

Abstract Composition Layer

Fragment Components

Run Time Components

Abstract Ontology

Time specific Layer

Java Fragment Boxes

X-HTML Fragment Boxes

....

JDRUMS Components

Language specific Layer

Java Fragment Values

Java Semantics

X-HTML Fragment Values

X-HTML Ontology

JDRUMS Component Values

JDRUMS Ontology

Refactorings

Future Composition Framework with More Layers

- Every concern makes up a layer
  - Composition Framework Structure
  - Is a Riehle/Züllinghoven framework with layers
  - Every complex object crosscuts all layers and has a core object, role object on every layer
- On every level, there are consistency rules
  - They can be baked into the corresponding role objects
- Division of the ontologies according to the layers
  - Layer-local consistency rules
  - Global consistency rules

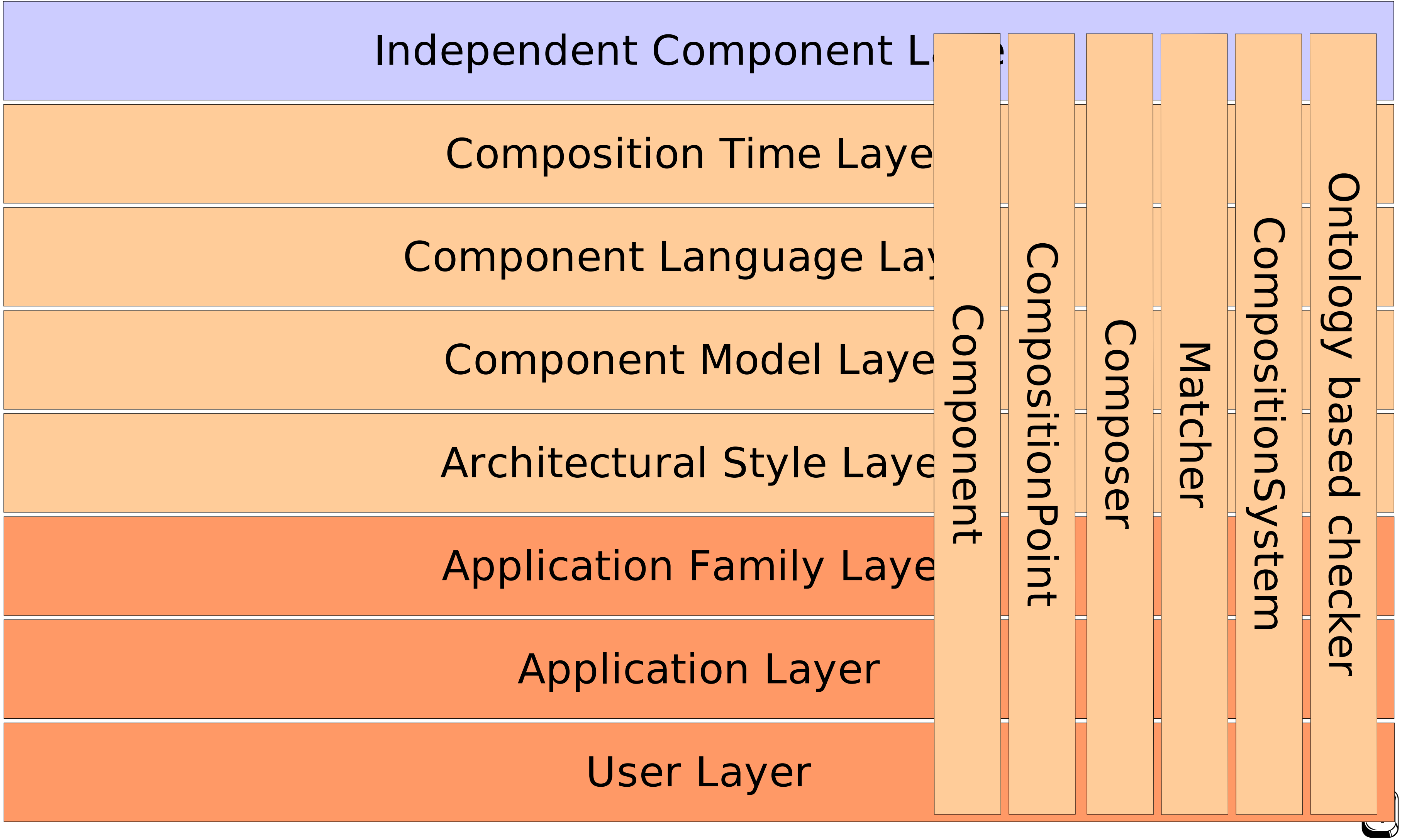
Concern Levels and Framework Layers

- Independent (core layer)
- Composition-time dependent (time layer)
  - forall compositions: same time
  - Staged composition (staging layer): is the result of a composition another composition?
- Language dependent: (component language layer)
  - e.g., forall component languages: same language
  - Mixed systems: language compatibility rules

Concern Levels and Framework Layers (ctd)

- Component and HookModel dependent (component model layer)
  - There can be many component models per component language
- Architectural style dependent (architectural style layer)
- Application family dependent (application family layer)
- Application dependent (application layer)
- [User layer (personalization layer)]

Future Layers and CrossCuts



The End

Translation of Ontologies to RML (ORML)

- [Adrian Pop, CUGS student]
- Pelab's RML is one of the advanced compiler generation tools
  - generating static and dynamic semantics
  - Fast
  - Debugger completed recently
- ORML (DAML2RML) is a subtask of XWizard
  - Goal: get a fast evaluator for DAML&OIL
  - Develop a ontology debugger
- Connection of RuleML must be clearified