OntoMaven Repositories and OMG API4KP

Ralph Schäfermeier    Adrian Paschke
Freie Universität Berlin

Berlin Semantic Web Meetup
1 OMG API4KP – An API for Knowledge Platforms
   - Scope of the Standard
   - Scenario: Semantic Annotation and Discovery of Biodiversity Data
   - The API4KP Metamodel
   - Foundations: Monads
   - Architecture Styles

2 Repositories with OntoMaven
   - Motivation
   - Architecture
   - Selected Plug-Ins
   - The Terminology Server Scenario with Distributed Knowledge Repositories
Outline

1. OMG API4KP – An API for Knowledge Platforms
   - Scope of the Standard
     - Scenario: Semantic Annotation and Discovery of Biodiversity Data
     - The API4KP Metamodel
     - Foundations: Monads
     - Architecture Styles

2. Repositories with OntoMaven
   - Motivation
   - Architecture
   - Selected Plug-Ins
   - The Terminology Server Scenario with Distributed Knowledge Repositories
A Meta-API for Knowledge (Reasoning) Platforms (KPs)

- A Meta-API is a platform-independent model (PIM) for a family of APIs in specific languages, also called PSMs (platform-specific models).

- API4KP provides a PIM for the external APIs of KPs.

**Figure: A Monolithic KP Architecture.**
Languages in Scope

- Knowledge Representation and Reasoning Languages
  - Resource Description Framework (RDF)
  - RDF Schema (RDFS)
  - Web Ontology Language (OWL)
  - Common Logic (CL)
  - Event-Condition-Action RuleML (ECA)
  - ...

- Data Representation Languages
  - Extensible Messaging and Presence Protocol (XMPP)
  - JSON
  - ...

Outline

1. OMG API4KP – An API for Knowledge Platforms
   - Scope of the Standard
   - Scenario: Semantic Annotation and Discovery of Biodiversity Data
   - The API4KP Metamodel
   - Foundations: Monads
   - Architecture Styles

2. Repositories with OntoMaven
   - Motivation
   - Architecture
   - Selected Plug-Ins
   - The Terminology Server Scenario with Distributed Knowledge Repositories
GFBio Terminology Server

- **Web Services Client**
  - HTTP request
  - Serialization format: RDF/XML, JSON, JSON LD, ...

- **API4KP**
  - External Web Services requestor

- **HTTP Server**

- **Semantic Web Repository (Virtuoso)**
  - SKOS
  - OWL
  - RDF

- **External Web Services**
  - requestor

- **GF Bion Terminology Server**
  - WoRMS
  - PESI
  - Catalogue of Life
  - GeoNames
Outline

1. **OMG API4KP – An API for Knowledge Platforms**
   - Scope of the Standard
   - Scenario: Semantic Annotation and Discovery of Biodiversity Data
   - The API4KP Metamodel
   - Foundations: Monads
   - Architecture Styles

2. **Repositories with OntoMaven**
   - Motivation
   - Architecture
   - Selected Plug-Ins
   - The Terminology Server Scenario with Distributed Knowledge Repositories
A metamodel is a model about models. Entities in the universe of a KP fall principally into these classes:

- Knowledge Sources - Models of the World
- Knowledge Environments - Models of Relationships
- Knowledge Operations - the Meta-API, Models of APIs
- Knowledge Events
Entities in the universe of a KP fall principally into these classes:

- **Knowledge Sources**: source of machine-readable information with semantics.
- **Knowledge Environments**
- **Knowledge Operations**
- **Knowledge Events**
Entities in the universe of a KP fall principally into these classes:

- Knowledge Sources
  - Mutable or Immutable (a.k.a Knowledge Resources)
  - By level of abstraction (Knowledge Source Level)
  - Basic or Structured

- Knowledge Environments
- Knowledge Operations
- Knowledge Events
Entities in the universe of a KP fall principally into these classes:

- Knowledge Sources
  - Mutable or Immutable (a.k.a Knowledge Resources)
  - By level of abstraction (Knowledge Source Level)
    - Item - physical source, e.g. on hard disk or in memory
    - Manifestation - concrete syntax, e.g. OWL axiom in Manchester syntax
    - Expression - abstract syntax, e.g. OWL axiom
    - Asset - equivalence class of Knowledge Expressions, with some equivalence relation e.g. logical equivalence

- Basic or Structured
  - Knowledge Environments
  - Knowledge Operations
  - Knowledge Events
Entities in the universe of a KP fall principally into these classes:

- Knowledge Sources
- Knowledge T Environments: **mathematical structure of mappings,** where the domain and codomains of the mappings, called members of the knowledge environment, are instances of class T.
- Knowledge Operations
- Knowledge Events
Entities in the universe of a KP fall principally into these classes:

- Knowledge Sources
- Knowledge Environments
- Knowledge Operations: functionality (possibly with side-effects, i.e. effects beyond the output value returned) having a knowledge source, environment or operation type in its signature.
- Knowledge Events
Entities in the universe of a KP fall principally into these classes:

- Knowledge Sources
- Knowledge Environments
- Knowledge Operations
- Knowledge Events: successful evaluation or execution of a knowledge operation by a particular application at a particular time
Outline

1. OMG API4KP – An API for Knowledge Platforms
   - Scope of the Standard
   - Scenario: Semantic Annotation and Discovery of Biodiversity Data
   - The API4KP Metamodel
   - Foundations: Monads
   - Architecture Styles

2. Repositories with OntoMaven
   - Motivation
   - Architecture
   - Selected Plug-Ins
   - The Terminology Server Scenario with Distributed Knowledge Repositories
The various knowledge representations in the scenario require a variety of structures.

- Descriptions of I/O programs for persistent Knowledge Sources, e.g. the database of the Case History KP.
- Unordered collections for declarative Knowledge Sources,
- Ordered collections for order-prioritized Knowledge Sources,
- Concurrent sequences for streamed Knowledge Sources,
- Descriptions of side-effects for active Knowledge Sources,
- Failure-aware types for reliable Knowledge Sources,
- Descriptions of state transitions for stateful Knowledge Sources,
The various knowledge representations in the scenario require a variety of structures.

- Descriptions of I/O programs for persistent Knowledge Sources,
- Unordered collections for declarative Knowledge Sources, *e.g.* the OWL ontology in the Stream Processing KP,
- Ordered collections for order-prioritized Knowledge Sources,
- Concurrent sequences for streamed Knowledge Sources,
- Descriptions of side-effects for active Knowledge Sources,
- Failure-aware types for reliable Knowledge Sources,
- Descriptions of state transitions for stateful Knowledge Sources,
Structures Needed in API4KP

The various knowledge representations in the scenario require a variety of structures.

- Descriptions of I/O programs for persistent Knowledge Sources,
- Unordered collections for declarative Knowledge Sources,
- Ordered collections for order-prioritized Knowledge Sources, e.g. the ECA rulebase in the CDS KP.
- Concurrent sequences for streamed Knowledge Sources,
- Descriptions of side-effects for active Knowledge Sources,
- Failure-aware types for reliable Knowledge Sources,
- Descriptions of state transitions for stateful Knowledge Sources,
The various knowledge representations in the scenario require a variety of structures.

- Descriptions of I/O programs for persistent Knowledge Sources,
- Unordered collections for declarative Knowledge Sources,
- Ordered collections for order-prioritized Knowledge Sources,
- Concurrent sequences for streamed Knowledge Sources, e.g. the input and output streams of the Stream Processing KP,
- Descriptions of side-effects for active Knowledge Sources,
- Failure-aware types for reliable Knowledge Sources,
- Descriptions of state transitions for stateful Knowledge Sources,
The various knowledge representations in the scenario require a variety of structures.

- Descriptions of I/O programs for persistent Knowledge Sources,
- Unordered collections for declarative Knowledge Sources,
- Ordered collections for order-priorititized Knowledge Sources,
- Concurrent sequences for streamed Knowledge Sources,
- Descriptions of side-effects for active Knowledge Sources, e.g. the effects generated by the CDS KP.
- Failure-aware types for reliable Knowledge Sources,
- Descriptions of state transitions for stateful Knowledge Sources,
The various knowledge representations in the scenario require a variety of structures.

- Descriptions of I/O programs for persistent Knowledge Sources,
- Unordered collections for declarative Knowledge Sources,
- Ordered collections for order-prioritized Knowledge Sources,
- Concurrent sequences for streamed Knowledge Sources,
- Descriptions of side-effects for active Knowledge Sources,
- Failure-aware types for reliable Knowledge Sources, e.g. a failed communication from the CDS KP.
- Descriptions of state transitions for stateful Knowledge Sources,
The various knowledge representations in the scenario require a variety of structures.

- Descriptions of I/O programs for persistent Knowledge Sources,
- Unordered collections for declarative Knowledge Sources,
- Ordered collections for order-prioritized Knowledge Sources,
- Concurrent sequences for streamed Knowledge Sources,
- Descriptions of side-effects for active Knowledge Sources,
- Failure-aware types for reliable Knowledge Sources,
- Descriptions of state transitions for stateful Knowledge Sources, e.g. ECA simulations for protocol development.
The various knowledge representations in the scenario require a variety of structures.

- Descriptions of I/O programs for persistent Knowledge Sources,
- Unordered collections for declarative Knowledge Sources,
- Ordered collections for order-prioritized Knowledge Sources,
- Concurrent sequences for streamed Knowledge Sources,
- Descriptions of side-effects for active Knowledge Sources,
- Failure-aware types for reliable Knowledge Sources,
- Descriptions of state transitions for stateful Knowledge Sources,

In functional programming, monads have been created for each of these cases, providing a concept of equivalence that is isolated from side-effects and non-determinism, which is critical for conceptual modeling.
1. OMG API4KP – An API for Knowledge Platforms
   - Scope of the Standard
   - Scenario: Semantic Annotation and Discovery of Biodiversity Data
   - The API4KP Metamodel
   - Foundations: Monads
   - Architecture Styles

2. Repositories with OntoMaven
   - Motivation
   - Architecture
   - Selected Plug-Ins
   - The Terminology Server Scenario with Distributed Knowledge Repositories
Direct Strongly-Coupled API4KB Access
- OntoMaven, RuleMaven
API4KP Distributed Architecture Styles

- Direct Strongly-Coupled API4KB Access
  - OntoMaven, RuleMaven

- Loosely-coupled Remote Invocation via API4KB Interfaces
  - Request-Reply Protocols
  - Remote Procedure Calls
  - Remote Method Invocation
  - Distributed Components (OntoMaven with Aspect-Oriented Component Model)
API4KP Distributed Architecture Styles

- **Direct Strongly-Coupled API4KB Access**
  - OntoMaven, RuleMaven

- **Loosely-coupled Remote Invocation via API4KB Interfaces**
  - Request-Reply Protocols
  - Remote Procedure Calls
  - Remote Method Invocation
  - Distributed Components (OntoMaven with Aspect-Oriented Component Model)

- **Decoupled Indirect Communication**
  - Publish Subscribe and Distributed Event Based Systems
  - Group communication (broadcast/multicast)
  - Shared Resources (Tuple Spaces, Distributed Shared Memory, ...)
  - Asynchronous Messaging (ZeroMQ, Erlang/RabbitMQ, OpenESB, Mule ESB, ...)

Participants

- Tara Athan, Athan Services (athant.com), West Lafayette, Indiana, USA, taraathan@gmail.com
- Roy Bell, Raytheon, Fort Wayne, Indiana, USA, Roy_M_Bell@raytheon.com
- Elisa Kendall, Thematix Partners LLC, New York, New York, USA, ekendall@thematix.com
- Adrian Paschke, AG Corporate Semantic Web, Freie Universitaet Berlin, Germany, paschke@inf.fu-berlin.de
- Ralph Schaefermeier, AG Corporate Semantic Web, Freie Universitaet Berlin, Germany, schaef@inf.fu-berlin.de
- Davide Sottara, Department of Biomedical Informatics, Arizona State University, USA, davide.sottara@asu.edu
Outline

1. OMG API4KP – An API for Knowledge Platforms
   - Scope of the Standard
   - Scenario: Semantic Annotation and Discovery of Biodiversity Data
   - The API4KP Metamodel
   - Foundations: Monads
   - Architecture Styles

2. Repositories with OntoMaven
   - Motivation
   - Architecture
   - Selected Plug-Ins
   - The Terminology Server Scenario with Distributed Knowledge Repositories
Ontology Project Life Cycle Management

- Avoid manual steps
  - Reuse/import, rename, refactor, merge, test, deploy, …
  - Pass documentation and configuration information
Ontology Project Life Cycle Management

- Avoid manual steps
  - Reuse/import, rename, refactor, merge, test, deploy, ...
  - Pass documentation and configuration information
- Work collaboratively, but support local development
  - Remote and local repositories (even without Internet connection)
Avoid manual steps
  - Reuse/import, rename, refactor, merge, test, deploy, …
  - Pass documentation and configuration information

Work collaboratively, but support local development
  - Remote and local repositories (even without Internet connection)

Automatically process resources
  - Manage dependencies and development + deployment properties
Avoid manual steps
- Reuse/import, rename, refactor, merge, test, deploy, . . .
- Pass documentation and configuration information

Work collaboratively, but support local development
- Remote and local repositories (even without Internet connection)

Automatically process resources
- Manage dependencies and development + deployment properties

Share ontology resources across projects
- Ontology artifacts including metadata, documentation, tests, . . .
Outline

1. OMG API4KP – An API for Knowledge Platforms
   - Scope of the Standard
   - Scenario: Semantic Annotation and Discovery of Biodiversity Data
   - The API4KP Metamodel
   - Foundations: Monads
   - Architecture Styles

2. Repositories with OntoMaven
   - Motivation
   - Architecture
   - Selected Plug-Ins
   - The Terminology Server Scenario with Distributed Knowledge Repositories
OntoMaven – Apache Maven Extension

- **Ontology Artifacts**
  - Managed by `groupId`, `artifactId`, and `version`
OntoMaven – Apache Maven Extension

- **Ontology Artifacts**
  - Managed by `groupId`, `artifactId`, and `version`

- **Ontology Repositories**
  - `Local` and `remote` repositories used to store artifacts (ontologies, plug-ins, resources, test suites etc.)
OntoMaven – Apache Maven Extension

- Ontology **Artifacts**
  - Managed by `groupId`, `artifactId`, and `version`

- Ontology **Repositories**
  - **Local** and **remote** repositories used to store artifacts (ontologies, plug-ins, resources, test suites etc.)

- OntoMaven **Plug-ins**
  - Implement the ontology project life cycle management functionalities
  - **Goals** provide interfaces bundled to life cycle **phases**
OntoMaven – Apache Maven Extension

- **Ontology Artifacts**
  - Managed by `groupId`, `artifactId`, and `version`

- **Ontology Repositories**
  - **Local** and **remote** repositories used to store artifacts (ontologies, plug-ins, resources, test suites etc.)

- **OntoMaven Plug-ins**
  - Implement the ontology project life cycle management functionalities
  - **Goals** provide interfaces bundled to life cycle **phases**

- **Project Ontology Model**
  - `POM.xml` – Declarative project description and configuration
<project>
  <modelVersion>...</modelVersion>
  <groupId>...</groupId>
  <artifactId>...</artifactId>
  <version>...</version>
  ...
</project>
The Maven Repository Architecture

Outline

1  OMG API4KP – An API for Knowledge Platforms
   ■ Scope of the Standard
   ■ Scenario: Semantic Annotation and Discovery of Biodiversity Data
   ■ The API4KP Metamodel
   ■ Foundations: Monads
   ■ Architecture Styles

2  Repositories with OntoMaven
   ■ Motivation
   ■ Architecture
   ■ Selected Plug-Ins
   ■ The Terminology Server Scenario with Distributed Knowledge Repositories
OntoMaven – Selected Plug-Ins

- Import Dependency Management
- Documentation
- Versioning
- Testing with Test Suites
- Aspect-Oriented Development and Module Selection
- ...
Import and Dependency Management Plug-In

Figure: Importation of an ontology into the local repository.
Ontology Artifacts

- Archiva Repository Manager
- Dependencies

**Camera OWL Ontology**

- It is an example for an ontology in owl
- Repository snapshots
- Group ID xfront.com.ontologies
- Artifact ID Camera OWL Ontology
- Version 1.0-20121209.221136-1
- Packaging owl
- Other Versions 1.0-20121209.221136-1

**POM Snippet**

```xml
<dependency>
  <groupId>xfront.com.ontologies</groupId>
  <artifactId>Camera OWL Ontology</artifactId>
  <version>1.0-SNAPSHOT</version>
  <classifier>owl</classifier>
</dependency>
```

**Other Details**

- URL http://example.org
- Organisation CameraOwl
- License Common Public License Version 1.0
Ontology Artifacts in Remote Repository

```xml
<profiles>
  <profile>
    <id>2</id>
    <activation><activeByDefault>true</activeByDefault></activation>
    <repositories>
      <repository>
        <snapshots><enabled>true</enabled></snapshots>
        <id>snapshots</id>
        <name>OntoMaven Snapshot Repository</name>
        <url>http://www.corporate-semantic-web.de/repository/snapshots/</url>
      </repository>
    </repositories>
  </profile>
</profiles>

<dependencies>
  <dependency>
    <groupId>xfront.com.owl.ontologies</groupId>
    <artifactId>Camera-OWL-Ontology</artifactId>
    <version>1.0-SNAPSHOT</version>
    <type>owl</type>
  </dependency>
</dependencies>
```
<description>here’s the description of an ontology</description>

<organization>
<name>Corporate Semantic Web, Freie Universitaet Berlin</name>
$url>http://www.corporate–semantic–web.de</url>
</organization>

<inceptionYear>2013</inceptionYear>

/licenses>
<license>
<name>LGPL–3.0</name>
$url>http://www.gnu.org/licenses/lgpl.txt</url>
</license>
</licenses>

<developers>
<developer>
<name>Adrian Paschke</name>
<email>paschke@inf.fu–berlin.d</email>
<organization>Corporate Semantic Web</organization>
<organizationUrl>http://www.corporate–semantic–web.de/</organizationUrl>
</developer>
</developers>

<roles><role>developer</role></roles>
</developers>
The Documentation of Anonymous-2

Description: Camera OWL Ontology Author: Roger L. Costello Acknowledgments: Many thanks to the following people for their invaluable input: Richard McCullough, Yuzhong Qu, Leo Sauermann, Brian McBride and Jim Farrugia. Modified as a Jena example by Ian Dickinson.

Ontology Format is: RDF/XML

Profile: The Ontology is in OWL 2 DL

Imports:

<table>
<thead>
<tr>
<th>Import</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>found no import</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Classes</td>
<td>13</td>
</tr>
<tr>
<td>Total Number of Datatype Properties</td>
<td>8</td>
</tr>
<tr>
<td>Total Number of Object Properties</td>
<td>7</td>
</tr>
<tr>
<td>Total Number of Annotations Properties</td>
<td>1</td>
</tr>
<tr>
<td>Total Number of Individuals</td>
<td>2</td>
</tr>
</tbody>
</table>
## A-Z of

This is a complete alphabetical A-Z index of all terms, by class (categories or types) and by property.

### Classes:

| PurchaseableItem | Range | Body | Viewer | Money | Camera | Large-Format | Lens | Window | BodyWithNonAdjustableShutterSpeed | Digital | SLR |

### Properties:

| body | shutter-speed | viewFinder | part | lens | cost | compatibleWith | min | focal-length | max | f-stop | aperture | units | currency | size |

### More Details

**PurchaseableItem**

- Has Subclass: Body | Camera | Lens
- DataTypeProperties:
- ObjectProperties: cost
- COMMENT: "Example for an annotated owl class"

**Range**

- Has Subclass:
- DataTypeProperties: min | max | units
Visualization Report
## Versioning Plug-In – SVont

<table>
<thead>
<tr>
<th>Plug-In SVN Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkout</td>
<td>Check out working copy from project</td>
</tr>
<tr>
<td>status</td>
<td>Status of the working copy</td>
</tr>
<tr>
<td>diff</td>
<td>Diff between two revisions or paths</td>
</tr>
<tr>
<td>commit</td>
<td>Commit working copy to repository</td>
</tr>
<tr>
<td>info</td>
<td>Info about repository and working copy</td>
</tr>
</tbody>
</table>

### Diagram

```
Iteration 1
Version 1
Version 2

Iteration 2
Version 1
Version 2

... (Iterations 3 to (n-1))
Version 1
Version 2.1
Version 2.x
Version 3

Iteration n-1
Version 1
Version 2.x
Version 3

Iteration n
Version 1
Version 2.x
Version 3
```
OntoMaven – Plug-In Goal "diff"

-------------------------- DIFF INFORMATION --------------------------
Ontology File : ...\...\...\...\...\...\camera.owl

================================ ACTUAL CHANGES =========================
Axioms were added to the repository, or deleted from the working copy.
SubClassOf(<http://www.xfront.com/ontologies/camera/#Money> owl:Thing)
Declaration(Class(<http://www.xfront.com/ontologies/camera/#Money>))

------------------ MORE INFO ---------------------------------------
The above changes of the OWL classes are dependent on the axioms:
currency <------ DataProperty (Domain)
cost <------ ObjectProperty (Range)
SVont – Subversion for Ontologies

Classical SVN Clients

Ext. SVN Interface

Command Handler

ext. SVN commands

Metadata Rep.

Change Log

Change Detector

Ontology Diff

Consist. Check

SVoNt Server

classical SVN commands

SVoNt Client

SVN Commands

Commit

Revision Rollback

Change Detector

Change Selector

Working Copy

Ontology Diff

Revision Calculator

Class. SVN Interface

Ext. Precommit

SVN Workflow
SVont – Commit Workflow

Client

commit

Ontology File

SVN Server

Precommit Hook

Consistency Check

consistent

no

yes

Change Detection

changes

no

yes

Process Commit

Change Writer

Process Commit

Change Log

SVoNt Server

SVoNt Extension

Error
Test Cases constrain the possible models and approximate the intended models of the knowledge base.

- Queries are used to test the rule base.

A test case is defined by $T := \{X, A, N\}$, where

- $X \subseteq L$ assertion base (input data, e.g. facts)
- $A \in L$ a formula denoting a test query
- $N := +, -$ a positive or negative label

Semantics:

$M_0 \models_{TC} (X, A, +)$ iff $\forall m \in M_0 : m \in \Sigma (Mod(X), R) \Rightarrow m \in Mod(A)$

$M_0 \models_{TC} (X, A, -)$ iff $\exists m \in M_0 : m \in \Sigma (Mod(X), R) \Rightarrow m \notin Mod(A)$

- $\models_{TC}$ compatibility relation
- $Mod$ association function between sets of formulas and sets of models
- $\Sigma$ model selection function

$A \notin C_R(X)$ for $T:=\{X, A, +\}$ and $A \in C_R(X)$ for $T:=\{X, A, -\}$

$C_R(X)$ deductive closure of $X$. Decidable inference operator based on formal proofs.
POM.xml Test Plug-In Configuration

```xml
<plugin>
  <groupId>de.csw.MvnOnt</groupId>
  <artifactId>MvnOwlTest</artifactId>
  <version>1.0-SNAPSHOT</version>
  <configuration>
    <owlfile>owl/1a.owl</owlfile>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>owltest</goal>
      </goals>
    </execution>
  </executions>
</plugin>

<plugin>
  <groupId>de.csw.MvnOnt</groupId>
  <artifactId>MvnOwlEntailment</artifactId>
  <version>1.0-SNAPSHOT</version>
  <configuration>
    <premise_file>owl/1a.owl</premise_file>
    <conclusion_file>owl/1aconclusion.owl</conclusion_file>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>owlentailment</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```
Aspect-Oriented Ontologies

- Original ontology
- Query
- Manual annotation/selected editing context in tool
- Aspect definition by quantification
- Ontology with aspect annotations
- Aspect names or descriptions
- Aspect-based module selection
- Ontology module selection as an OntoMaven goal
Declarative Module Selection with Aspects and OntoMaven

Aspect definition by quantification

Query Manual annotation/selected editing context in tool

Original ontology

Aspect-based module selection

Aspect names or descriptions

Ontology with aspect annotations

Ontology module selection as an OntoMaven goal

<userAspects>
  <aspect>http://example.org/reputation#Reputation123</aspect>
  <aspect>http://example.org/provenance#prov_789</aspect>
  <aspect>http://example.org/time#TimeInterval1</aspect>
</userAspects>
1 OMG API4KP – An API for Knowledge Platforms
   ■ Scope of the Standard
   ■ Scenario: Semantic Annotation and Discovery of Biodiversity Data
   ■ The API4KP Metamodel
   ■ Foundations: Monads
   ■ Architecture Styles

2 Repositories with OntoMaven
   ■ Motivation
   ■ Architecture
   ■ Selected Plug-Ins
   ■ The Terminology Server Scenario with Distributed Knowledge Repositories
Terminology Server Scenario with OntoMaven

- Ontologies, vocabularies and external sources configurable as OntoMaven dependencies.
- Easy to add new resources.
- Fast deployment of test instances with different knowledge sources.
- Context-based deployment of knowledge components with aspect-orientation.

FU Berlin, OntoMaven and API4KP, Berlin Semantic Web Meetup Dec 2, 2015
The Terminology Server scenario encompasses a number of architectural paradigms and KR requirements addressed by API4KP. API4KP shows high ontological completeness and clarity with respect to the BWW reference model.
The Terminology Server scenario encompasses a number of architectural paradigms and KR requirements addressed by API4KP.

API4KP shows high ontological completeness and clarity with respect to the BWW reference model.

OntoMaven automates and supports the life cycle management of ontology projects and ontology artifacts.

Declarative description in POM defining project conventions and APIs
- Additional resource directories
- Plug-in configuration
- Artifact libraries
- Repository and deployment configuration
- Aspect-orientation for context-based knowledge resource selection
- ...

API4KP and OntoMaven allow integration and exchange of heterogeneous and distributed knowledge and data sources
- in a declarative fashion
- along with metadata, interface descriptions, and test cases
Related Publications


Related Publications


