

Integration of the Semantic Web with Meta Object Facilities

Work in progress supported by the U.S. General
Service Administration's Open Source eGov Reference
Architecture (OsEra) Project

Cory Casanave, Data Access Technologies
Mohamed Keshk, Data Access Technologies

Overview

- Enhance “modeling world” capabilities with ontologies
- Map existing MOF models to RDF/OWL
 - Map Meta Meta Model
 - Automate – Map Meta models (E.G. UML)
 - Automate – Map Models (E.G. Finance model)
- Semantic integration (Semantic core)
 - Reference ontology for meta concepts
 - Integrate information in different views/languages

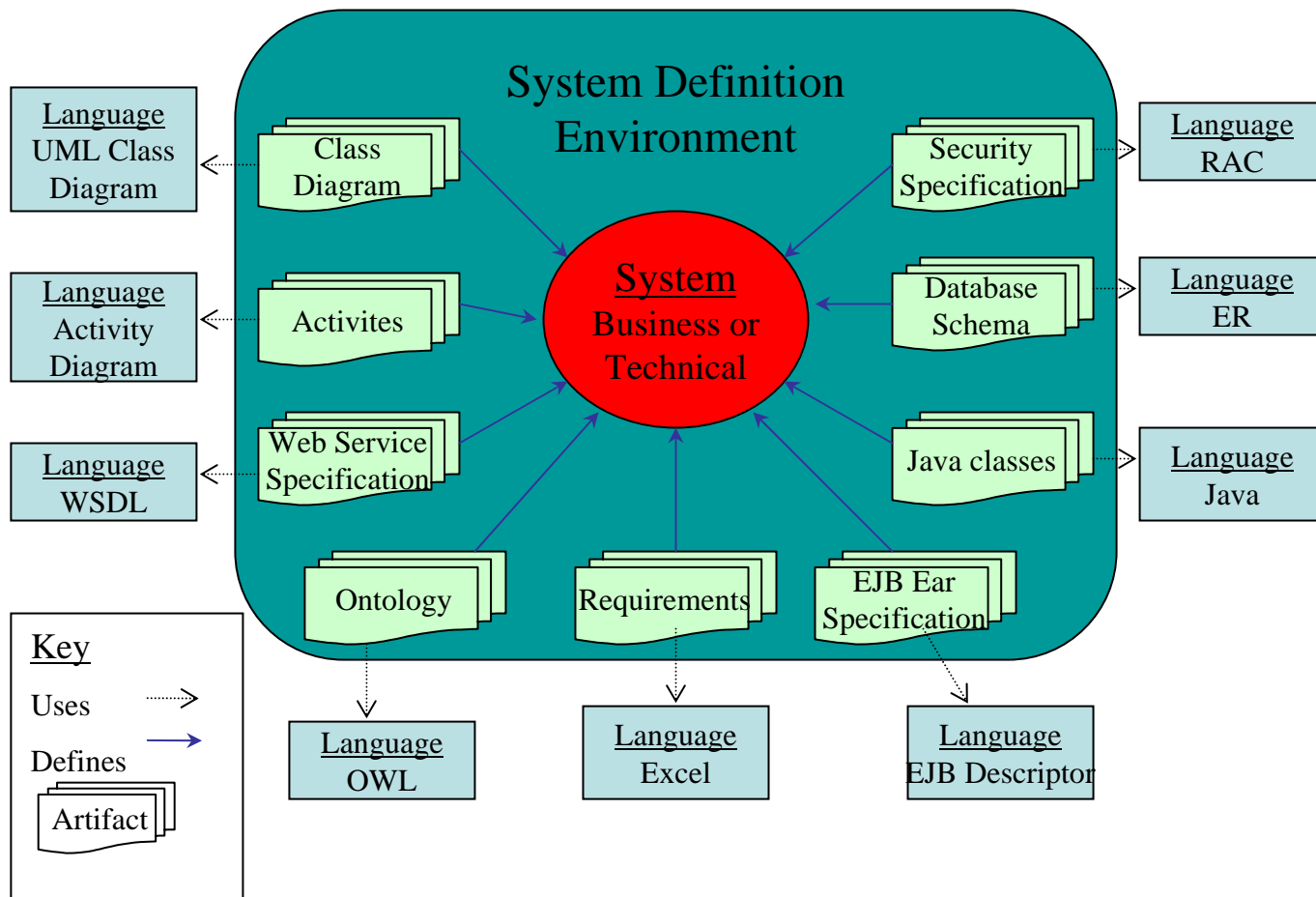
OsEra

- Open Source Egov Reference Architecture
- Sponsored by GSA
- Preliminary state – not really public
- Goals
 - Government enterprise architecture focus
 - “Model to integrate” MDA tool set
 - Semantic grounding and integration of architectures
 - Provisioning of runtime components
 - Runtime framework

Semantic Technologies Objective

- More work is needed to bridge between the “Modeling World” and the Ontology World
 - ODM Being the most significant work so far
 - EMF to OWL is “Reverse ODM”
- Vocabularies, Ontologies, Processes, Information Models
 - Information about enterprises and systems is being defined and collected in a plethora of languages, paradigms and infrastructures
- This does not meet the needs of the users
 - Who need their information connected, regardless of the source or “religion”
 - Integrated view of the enterprise
 - Enormous problem in government!
- We can start to address this at the “repository” and “meta” levels
- RDF and OWL seem ideally suited to this task

Users “meta” integration problem



Too many ways to talk about the same thing, redundant and conflicting semantics.

Ontologies Need Structured Modeling

- We define “structured modeling” to include MOF, UML, Process Models, Information Modeling (E.G. E-R models), Enterprise Architectures, Corba-IDL, etc.
 - It can also be thought to include programming languages
 - E.G. Most of the Non-ontology modeling world
- A vast amount of information exists in these environments.
- Use of these paradigms represents mainstream best practice – there are lots of practitioners
- The structured modeling tools are more mature and suited to specific problems.
- Model Driven Architecture has started to bind structured models with the software development process, providing even more leverage
 - Doesn’t require changing the runtime infrastructure as some Semantic Web Approaches are suggesting – separation of technology concerns
- Ontologies can’t ignore this wealth of knowledge, tools, expertise and industry momentum.

Structured Modeling Needs Ontologies

- The semantic web infrastructure provides a great way to
 - Publish models as web resources
 - Query over models
 - Analyze models and the intersection of multiple models
 - “Semantically ground” models
- RDF/Ontology based models are more resilient to change without “refactoring”
- Ontologies are better able to connect models that were not designed together – integrating and adapting architectures, processes, interfaces and information
- The open, distributed and federated “meta object facility” has yet to emerge as readily available and mainstream
- Semantic web infrastructure is picking up industry steam, tools and infrastructure are coming available

Meta Object Facilities

- “MOF” is a current standard of the OMG
- Provides for a “model driven” repository of anything that can be described in it’s structural model
- UML is defined in MOF
- Three layer architecture;
 - M0 – Instances – “Ney York”
 - M1 – Models – “Cities”
 - M2 – Meta Models – “Class” – E.G. UML Constructs
 - M3 – The MOF meta meta model – “Mof Classes”
- Comes in full (CMOF) and lite (EMOF)
- Uses “XMI” – XML Model Interchange
- Becoming basis for much structured modeling interchange
- ODM is described in MOF
- MOF meta model is a subset of UML

Question we asked

- Would RDF make a better “MOF” than “MOF”???
 - Providing web based federation
 - More flexible modeling semantics
 - E.G. “Same as” and instances with more than one type
 - Link to inference, transformation and rules (Some support in MOF standards, but generally not implemented)
 - Query mechanisms – emerging (Not in MOF)
 - Rule systems emerging (Not in MOF)
 - Ability to “link” architectures (Not easily done in MOF)
 - Distributed and federated – Not widely available for MOF
 - Lots of emerging support (MOF is invisible)
- But, we can't lose all the capability, meta models and information in these MOF like environments. RDF and OWL are not up to being an MDA environment.

Note; This is the opposite of what is specified in ODM!

The “big win” potential

- Architectures developed using structured modeling tend to be islands
 - Can we bring these together into a coherent view of the problem domain – E.G. a true multi-view enterprise model?
 - Can we embrace multiple structured modeling languages as well as integrate Ontologies?
 - Can we use the semantic web stack to support integration of enterprises and their technology islands?

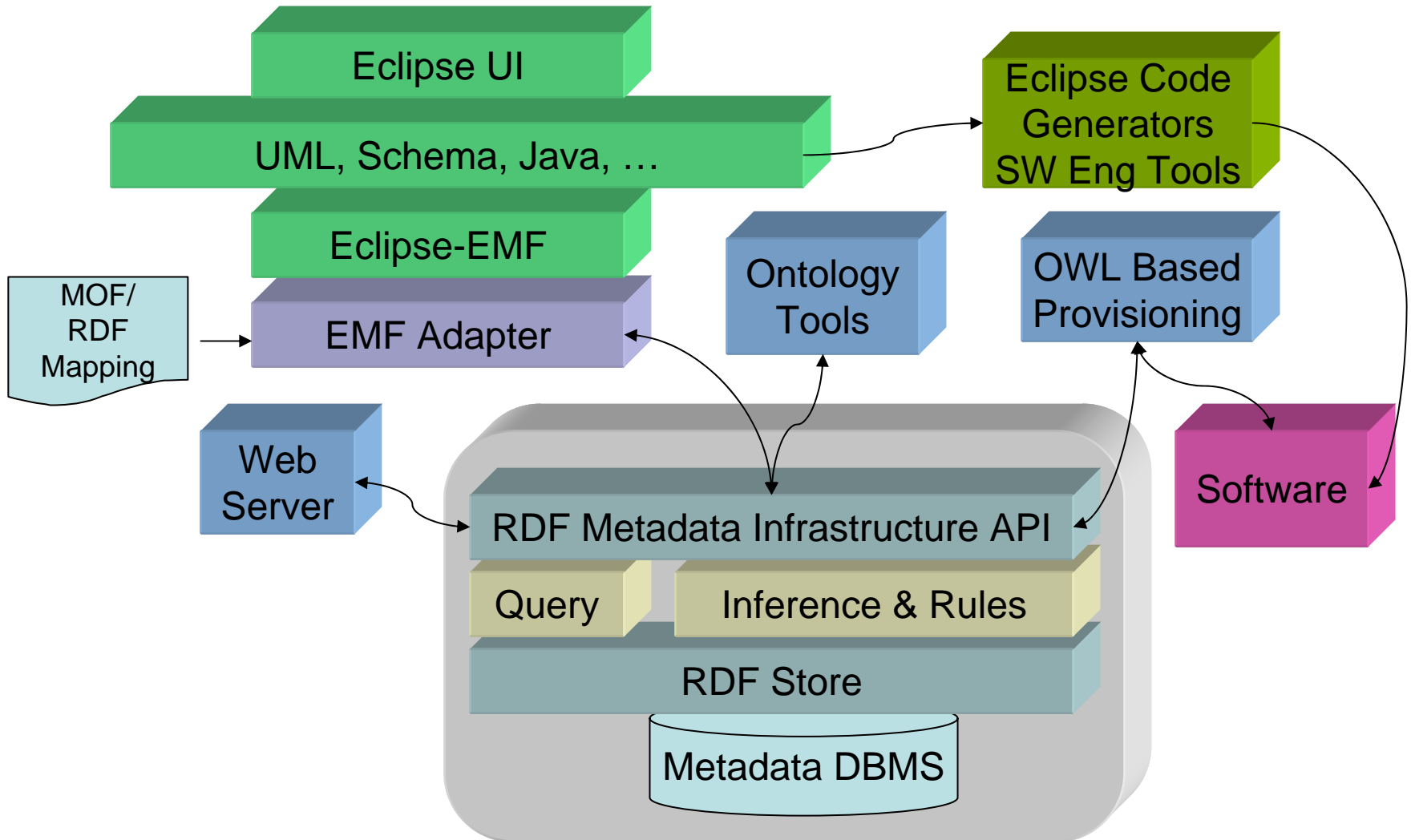
Separation of Concerns

- Infrastructural integration – MOF & RDF
 - Define mapping of MOF “meta layers” into RDF & OWL
 - Basis is MOF meta-meta model (Currently EMF)
 - Implement “Mappings”
 - Implement RDF as the persistence infrastructure for MOF based on these mappings
- Semantic Integration – Semantic Core
 - Look at concepts expressed in languages – UML, OWL, BPMN, EDOC, Etc.
 - Produce a “mid ontology” of modeling concepts
 - Use as semantic bridge between models expressed in different languages – avoid N*N integration
 - Provide mechanisms to “ground” structural models with Ontologies
 - Do not require as “the one and only” just, a grounding for a set of related models

OsEra MOF/RDF Experiment

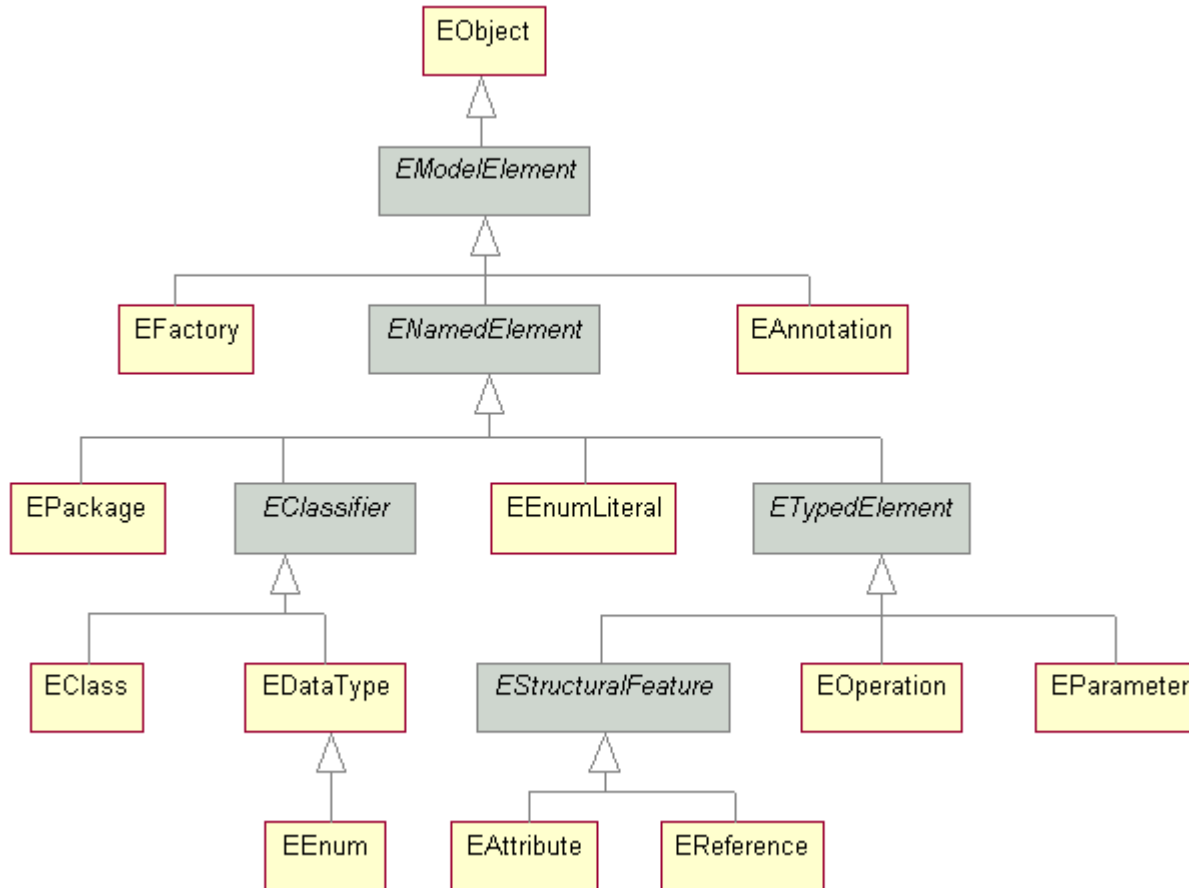
- Eclipse is an open source environment supporting a “lite” MOF called EMF
- Map the EMF meta meta model to RDF
 - ODM UML->OWL mapping is a base
- Use as a pattern for the other meta layers
- Implement transforms between “MOF in MOF” and “MOF in RDF”
- Using a RDF repository (E.G. Sesame or Jena) implement RDF as the persistence layer for EMF
 - Result – RDF becomes the persistent representation of ANY meta model in EMF
 - UML, J2EE, XML Schema, BPEL, Java... The list goes on and on
- Publish repositories as web resources
 - Everything in eclipse environment becomes a RDF web resource
- Integrate RDF query – MOF models become query-ready – across multiple distributed models!
- Note;
 - these will not be “semantically integrated” any more than they were in MOF
 - EMF is not full MOF, but is a good starting point and will prove concept.

Metadata Technology Layers



EMF Modeling in OWL

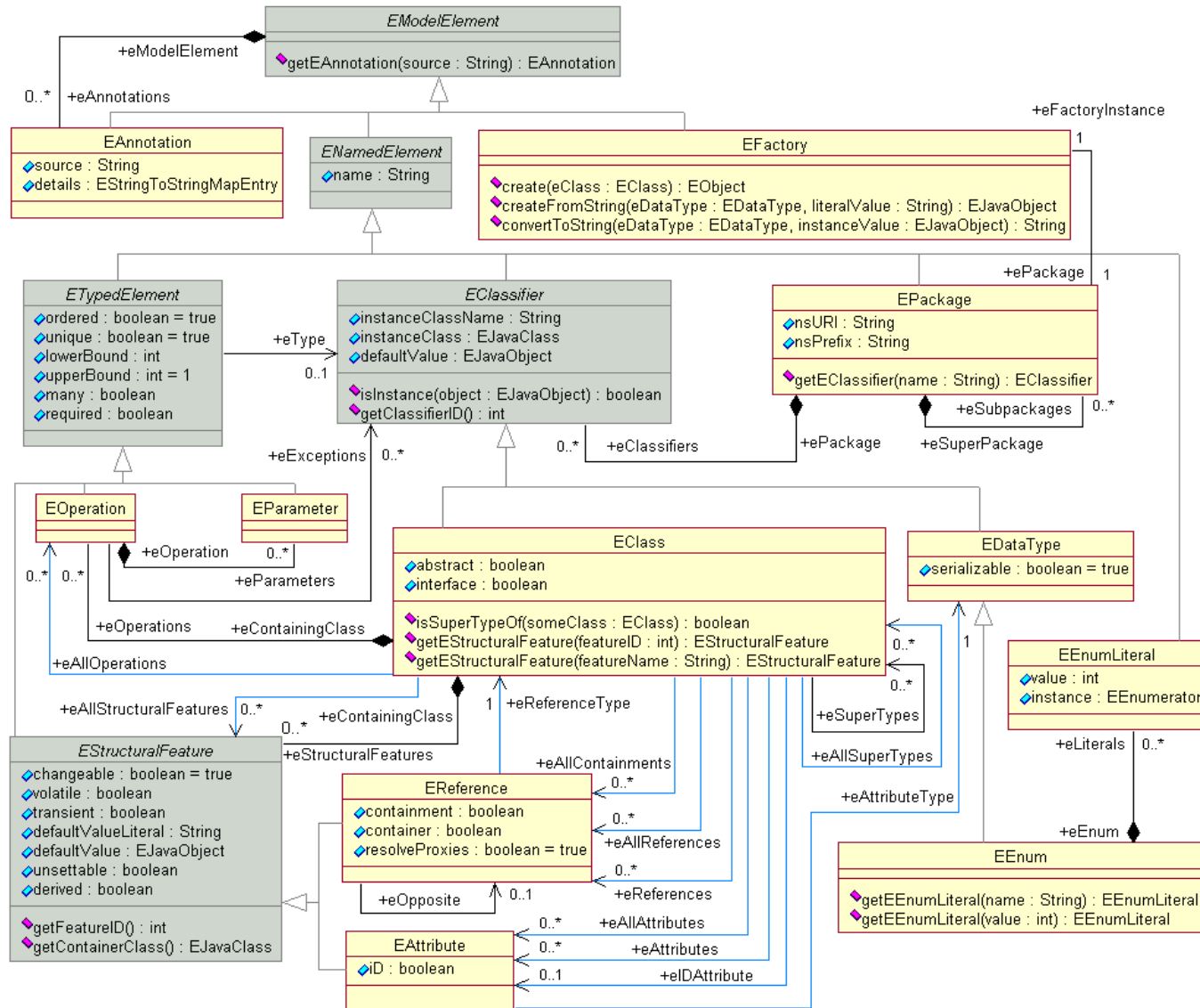
Ecore Class Hierarchy



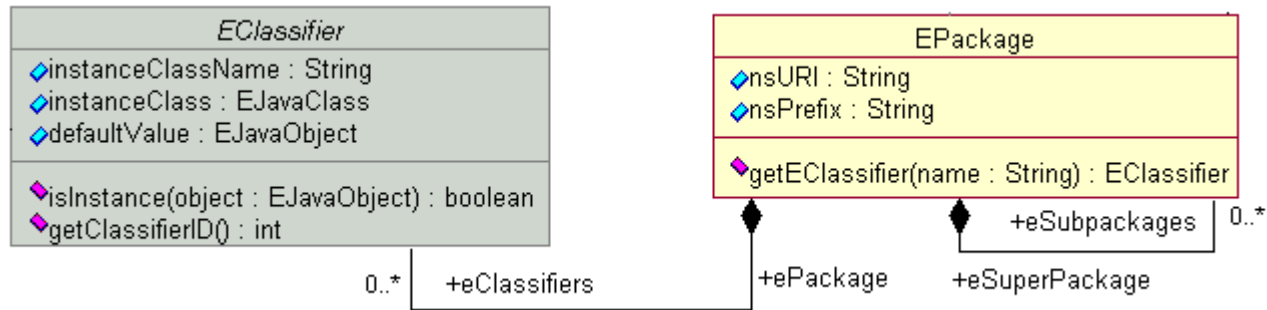
Meta-Meta-Layer

- `<owl:Class rdf:about="&EMF;EObject"/>`
-
- `<owl:Class rdf:about="&EMF;EModelElement">`
- `<rdfs:subClassOf rdf:resource="&EMF;EObject"/>`
- `</owl:Class>`
-
-
- `<owl:Class rdf:about="&EMF;EFactory">`
- `<rdfs:subClassOf rdf:resource="&EMF;EModelElement"/>`
- `</owl:Class>`
-
- `<owl:Class rdf:about="&EMF;EAnnotation">`
- `<rdfs:subClassOf rdf:resource="&EMF;EModelElement"/>`
- `</owl:Class>`
-
- `<owl:Class rdf:about="&EMF;ENamedElement">`
- `<rdfs:subClassOf rdf:resource="&EMF;EModelElement"/>`
- `</owl:Class>`

ECore Relationships



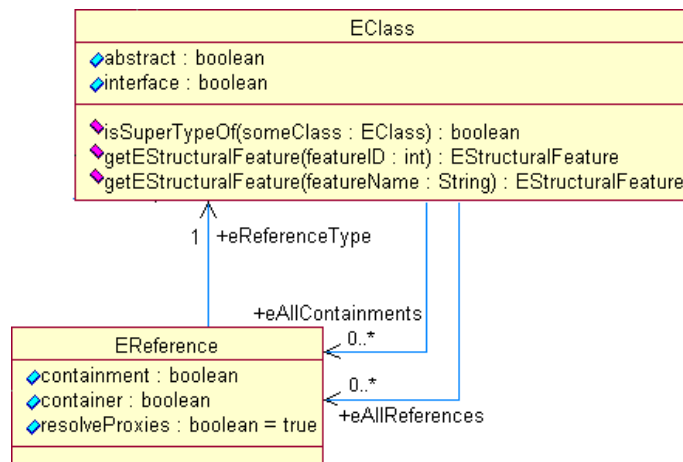
Composition Example



- `<owl:ObjectProperty rdf:about="&EPackage;eClassifiers">`
- `<rdfs:domain rdf:resource="&EMF;EPackage"/>`
- `<rdfs:range rdf:resource="&EMF;EClassifier"/>`
- `<owl:inverseOf rdf:resource="&EClassifier;ePackage"/>`
- `</owl:ObjectProperty>`

- `<owl:ObjectProperty rdf:about="&EClassifier;ePackage">`
- `<rdfs:domain rdf:resource="&EMF;EClassifier"/>`
- `<rdfs:range rdf:resource="&EMF;EPackage"/>`
- `<rdf:type rdf:resource="&owl;FunctionalProperty"/>`
- `<owl:inverseOf rdf:resource="&EPackage;eClassifiers"/>`
- `</owl:ObjectProperty>`

Association Example



- `<owl:ObjectProperty rdf:about="&EClass;eAllContainments">`
- `<rdfs:domain rdf:resource="&EMF;EClass"/>`
- `<rdfs:range rdf:resource="&EMF;EReference"/>`
- `</owl:ObjectProperty>`
-
- `<owl:ObjectProperty rdf:about="&EClass;eAllReferences">`
- `<rdfs:domain rdf:resource="&EMF;EClass"/>`
- `<rdfs:range rdf:resource="&EMF;EReference"/>`
- `</owl:ObjectProperty>`
-
- `<owl:ObjectProperty rdf:about="&EReference;eReferenceType">`
- `<rdfs:domain rdf:resource="&EMF;EReference"/>`
- `<rdfs:range rdf:resource="&EMF;EClass"/>`
- `<rdf:type rdf:resource="&owl;FunctionalProperty"/>`
- `</owl:ObjectProperty>`

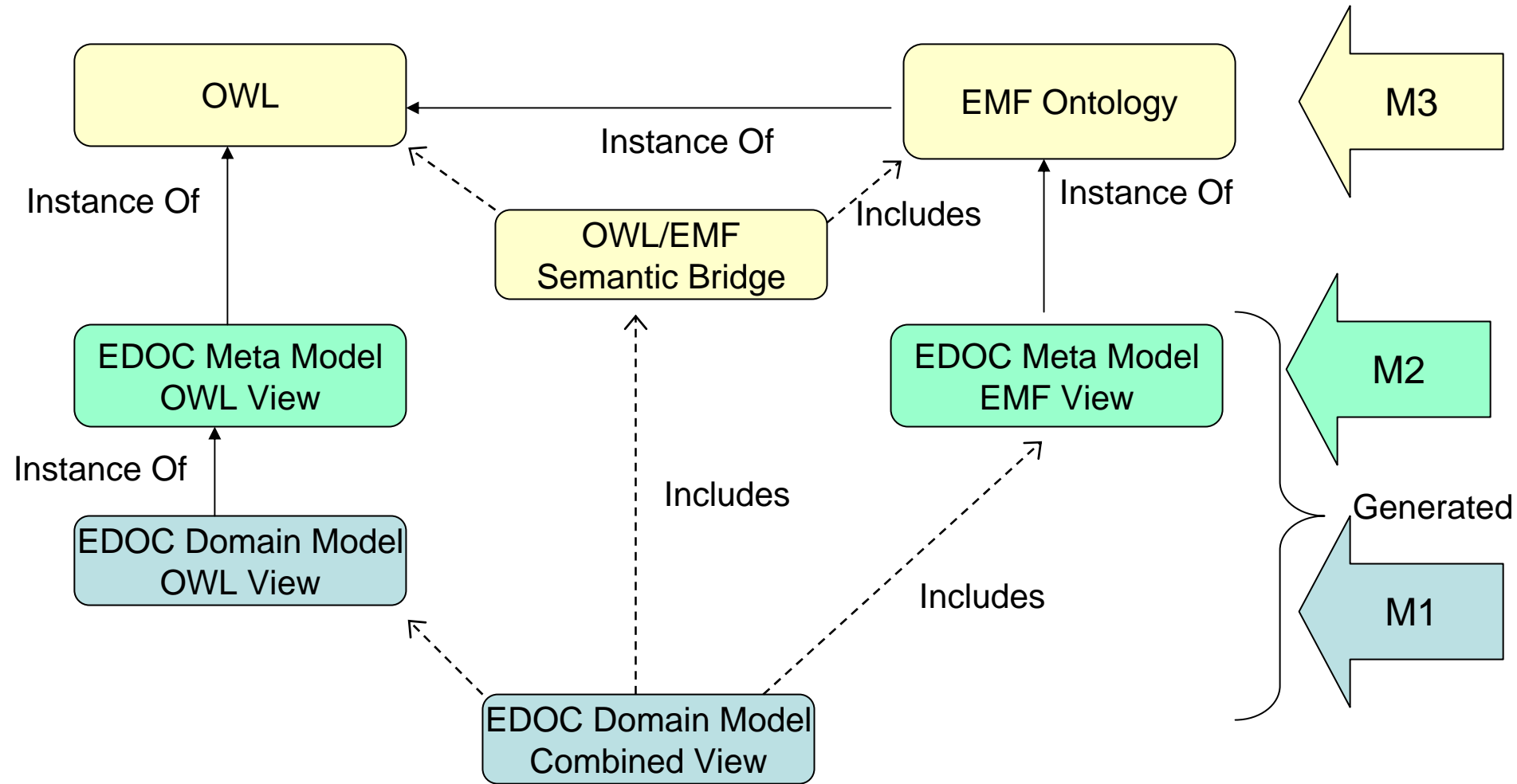
Modeling Issues & Workaround

- Representing a class as an instance at the same time
 - Since EMF follows MDA paradigm of having every element at certain layer as an instance of one element in the upper layer, So, it's a trade off to follow that modeling style in OWL. In this case, we maintain strong and consistent semantics by using the natural instance-of (type-of) relationship, but this will end up using OWL full. Consequently, we would lose using plug-in reasoners to infer hidden information because these reasoners only support OWL DL, but not OWL Full.
 - The other alternative is to use OWL DL, with some ontology design patterns (best practices) to weak instance-of (type-of) relationship into Sub-Class relationship. In this case, semantics wouldn't be as consistent as type-of relationship style, but we would be able to leverage a variety of available reasoners.
 - As a conclusion, a compromise is required. So, we decided to stick with type-of relationship approach (for now), and use a rule engine to extract all kinds of entailments we need by using a COT reasoner.
- Every attribute in UML class should have corresponding cardinality restriction or "FunctionalProperty" in OWL to confirm that its multiplicity is "1".
- UML attributes with default value has no direct OWL representation; this would be fixed using a rule engine to enforce the default value when creating an individual (instance) of the class.
- Namespace concepts are different – Use multiple OWN namespaces.

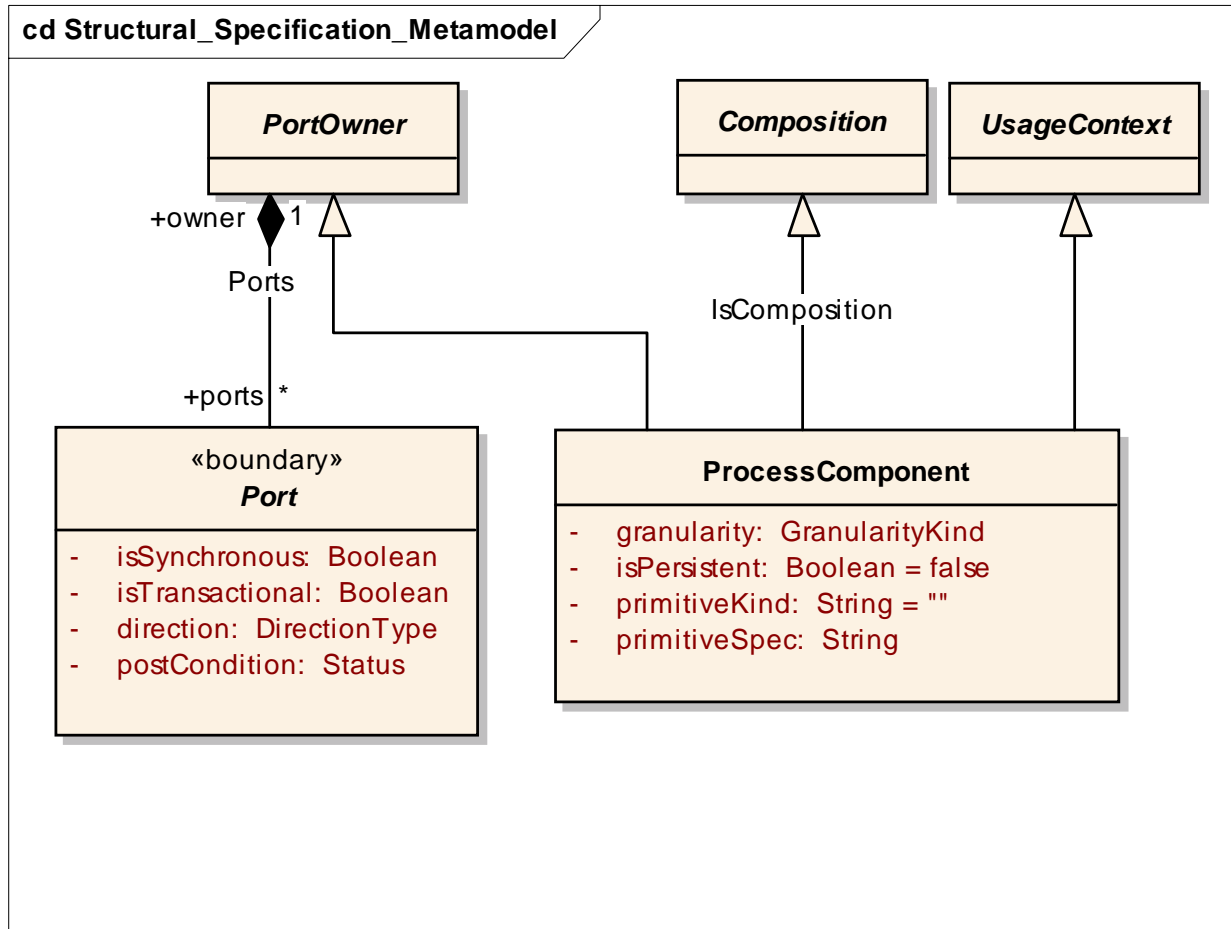
Meta Views

- OWL and EMF each provide an “M3” view, each of these is preserved
 - The “OWL View” presents the model as an OWL user may write it, but loses some EMF semantics
 - The “EMF View” is an instance of the EMF meta model expressed in OWL
 - Both views describe the same meta classes
 - Each view is in an independent ontology (about the same resource) so the required view(s) can be used

View of Meta Views



ProcessComponent Example



Example OWL View

```
<!ENTITY ProcessComponent      "http://www.modeldriven.org/rdf/EDOC/ProcessComponent/">
xmlns:ProcessComponent      = "&ProcessComponent;"
```

```
<!-- ***** Class ProcessComponent ***** -->
```

```
<owl:Class rdf:about="&EDOC;ProcessComponent">
  <rdfs:label>ProcessComponent</rdfs:label>
  <rdfs:subClassOf rdf:resource="&EDOC;Composition"/>
  <rdfs:subClassOf rdf:resource="&EDOC;Package"/>
  <rdfs:subClassOf rdf:resource="&EDOC;PortOwner"/>
  <rdfs:subClassOf rdf:resource="&EDOC;UsageContext"/>
</owl:Class>
```

```
<owl:DatatypeProperty rdf:about="&ProcessComponent;granularity">
  <rdfs:domain rdf:resource="&EDOC;ProcessComponent"/>
  <rdfs:range rdf:resource="&EDOC;GranularityKind"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:about="&ProcessComponent;isPersistent">
  <rdfs:domain rdf:resource="&EDOC;ProcessComponent"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>
```

Example EMF View

```
<!-- ***** Class ProcessComponent ***** -->

<ECore:EClass rdf:about="&EDOC;ProcessComponent">
  <ENamedElement:name
rdf:datatype="&xsd:string">ProcessComponent</ENamedElement:name>
  <EClass:eSuperTypes rdf:resource="&EDOC;Composition"/>
  <EClass:eSuperTypes rdf:resource="&EDOC;Package"/>
  <EClass:eSuperTypes rdf:resource="&EDOC;PortOwner"/>
  <EClass:eSuperTypes rdf:resource="&EDOC;UsageContext"/>
</ECore:EClass>
<ECore:EAttribute rdf:about="&ProcessComponent;isPersistent">
  <ENamedElement:name
rdf:datatype="&xsd:string">isPersistent</ENamedElement:name>
  <ETypedElement:eType rdf:resource="&ECore;EBooleanObject"/>
  <EStructuralFeature:defaultValueLiteral
rdf:datatype="&xsd:string">>false</EStructuralFeature:defaultValueLiteral>
</ECore:EAttribute>
```

EDOC Domain Model Fragment

```
<EDOC:ProcessComponent rdf:ID="ProcessComponent_1">  
  <Choreography:nodes rdf:resource="#PortConnector_1"/>  
  <Choreography:nodes rdf:resource="#PortConnector_2"/>  
  <UsageContext:portsUsed rdf:resource="#PortConnector_1"/>  
  <UsageContext:portsUsed rdf:resource="#PortConnector_2"/>  
  <PortOwner:ports rdf:resource="#FlowPort_1"/>  
  <PortOwner:ports rdf:resource="#FlowPort_2"/>  
</EDOC:ProcessComponent>
```

EDOC model in OWL

EDOC – M3_ECore

- `<owl:Class rdf:about="&ECore;EClass">`
- `<rdfs:subClassOf rdf:resource="&ECore;EClassifier"/>`
- `<rdfs:subClassOf>`
- `<owl:Restriction>`
- `<owl:onProperty rdf:resource="&EClass;eIDAttribute"/>`
- `<owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>`
- `</owl:Restriction>`
- `</rdfs:subClassOf>`
- `</owl:Class>`

- `<owl:Class rdf:about="&ECore;EAttribute">`
- `<rdfs:subClassOf rdf:resource="&ECore;EStructuralFeature"/>`
- `</owl:Class>`

- `<owl:Class rdf:about="&ECore;EReference">`
- `<rdfs:subClassOf rdf:resource="&ECore;EStructuralFeature"/>`
- `<rdfs:subClassOf>`
- `<owl:Restriction>`
- `<owl:onProperty rdf:resource="&EReference;eOpposite"/>`
- `<owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>`
- `</owl:Restriction>`
- `</rdfs:subClassOf>`
- `</owl:Class>`

- `<owl:DatatypeProperty rdf:about="&ENamedElement;name">`
- `<rdfs:domain rdf:resource="&ECore;ENamedElement"/>`
- `<rdfs:range rdf:resource="&xsd;string"/>`
- `<rdf:type rdf:resource="&owl;FunctionalProperty"/>`
- `</owl:DatatypeProperty>`

- `<owl:ObjectProperty rdf:about="&EClass;eSuperTypes">`
- `<rdfs:domain rdf:resource="&ECore;EClass"/>`
- `<rdfs:range rdf:resource="&ECore;EClass"/>`
- `</owl:ObjectProperty>`

- `<owl:ObjectProperty rdf:about="&ETypedElement;eType">`
- `<rdfs:domain rdf:resource="&ECore;ETypedElement"/>`
- `<rdfs:range rdf:resource="&ECore;EClassifier"/>`
- `</owl:ObjectProperty>`

- `<owl:ObjectProperty rdf:about="&EReference;eOpposite">`
- `<rdfs:domain rdf:resource="&ECore;EReference"/>`
- `<rdfs:range rdf:resource="&ECore;EReference"/>`

EDOC – M2_EDOC (EDOC view)

- `<ECore:EClass rdf:about="&EDOC;ProcessComponent">`
- `<ENamedElement:name rdf:datatype="&xsd:string">ProcessComponent</ENamedElement:name>`
- `<EClass:eSuperTypes rdf:resource="&EDOC;Composition"/>`
- `<EClass:eSuperTypes rdf:resource="&EDOC;Package"/>`
- `<EClass:eSuperTypes rdf:resource="&EDOC;PortOwner"/>`
- `<EClass:eSuperTypes rdf:resource="&EDOC;UsageContext"/>`
- `</ECore:EClass>`

- `<ECore:EAttribute rdf:about="&ProcessComponent;granularity">`
- `<ENamedElement:name rdf:datatype="&xsd:string">granularity</ENamedElement:name>`
- `<ETypedElement:eType rdf:resource="&ECore;GranularityKind"/>`
- `</ECore:EAttribute>`

- `<ECore:EAttribute rdf:about="&ProcessComponent;isPersistent">`
- `<ENamedElement:name rdf:datatype="&xsd:string">isPersistent</ENamedElement:name>`
- `<ETypedElement:eType rdf:resource="&ECore;EBooleanObject"/>`
- `<EStructuralFeature:defaultValueLiteral rdf:datatype="&xsd:string">>false</EStructuralFeature:defaultValueLiteral>`
- `</ECore:EAttribute>`

- `<ECore:EAttribute rdf:about="&ProcessComponent;primitiveKind">`
- `<ENamedElement:name rdf:datatype="&xsd:string">primitiveKind</ENamedElement:name>`
- `<ETypedElement:eType rdf:resource="&ECore;EString"/>`
- `<EStructuralFeature:defaultValueLiteral rdf:datatype="&xsd:string"></EStructuralFeature:defaultValueLiteral>`
- `</ECore:EAttribute>`

- `<ECore:EAttribute rdf:about="&ProcessComponent;primitiveSpec">`
- `<ENamedElement:name rdf:datatype="&xsd:string">primitiveSpec</ENamedElement:name>`
- `<ETypedElement:eType rdf:resource="&ECore;EString"/>`
- `</ECore:EAttribute>`

- `<ECore:EReference rdf:about="&ProcessComponent;properties">`
- `<ENamedElement:name rdf:datatype="&xsd:string">properties</ENamedElement:name>`
- `<ETypedElement:eType rdf:resource="&EDOC;PropertyDefinition"/>`
- `<ETypedElement:upperBound rdf:datatype="&xsd:int">-1</ETypedElement:upperBound>`
- `<EReference:containment rdf:datatype="&xsd:boolean">>true</EReference:containment>`
- `<EReference:eOpposite rdf:resource="&PropertyDefinition;component"/>`
- `</ECore:EReference>`

EDOC – M2_EDOC (OWL view)

- `<owl:Class rdf:about="&EDOC;ProcessComponent">`
- `<rdfs:label>ProcessComponent</rdfs:label>`
- `<rdfs:subClassOf rdf:resource="&EDOC;Composition"/>`
- `<rdfs:subClassOf rdf:resource="&EDOC;Package"/>`
- `<rdfs:subClassOf rdf:resource="&EDOC;PortOwner"/>`
- `<rdfs:subClassOf rdf:resource="&EDOC;UsageContext"/>`
- `</owl:Class>`

- `<owl:DatatypeProperty rdf:about="&ProcessComponent;granularity">`
- `<rdfs:domain rdf:resource="&EDOC;ProcessComponent"/>`
- `<rdfs:range rdf:resource="&EDOC;GranularityKind"/>`
- `</owl:DatatypeProperty>`

- `<owl:DatatypeProperty rdf:about="&ProcessComponent;isPersistent">`
- `<rdfs:domain rdf:resource="&EDOC;ProcessComponent"/>`
- `<rdfs:range rdf:resource="&xsd:boolean"/>`
- `</owl:DatatypeProperty>`

- `<owl:DatatypeProperty rdf:about="&ProcessComponent;primitiveKind">`
- `<rdfs:domain rdf:resource="&EDOC;ProcessComponent"/>`
- `<rdfs:range rdf:resource="&xsd:string"/>`
- `</owl:DatatypeProperty>`

- `<owl:DatatypeProperty rdf:about="&ProcessComponent;primitiveSpec">`
- `<rdfs:domain rdf:resource="&EDOC;ProcessComponent"/>`
- `<rdfs:range rdf:resource="&xsd:string"/>`
- `</owl:DatatypeProperty>`

- `<owl:ObjectProperty rdf:about="&ProcessComponent;properties">`
- `<rdfs:domain rdf:resource="&EDOC;ProcessComponent"/>`
- `<rdfs:range rdf:resource="&EDOC;PropertyDefinition"/>`
- `<owl:inverseOf rdf:resource="&PropertyDefinition;component"/>`
- `</owl:ObjectProperty>`

EDOC – M1_EDOC (The Model)

- `<EDOC:Package rdf:ID="Package_1">`
- `<AspectableElement:usage rdf:resource="#AspectUsage_1"/>`
- `<Package:ownedElements rdf:resource="#ProcessComponent_1"/>`
- `</EDOC:Package>`

- `<EDOC:AspectUsage rdf:ID="AspectUsage_1">`
- `<AspectUsage:usedAspect rdf:resource="#osera_aspectSlice"/>`
- `<AspectUsage:aspectConfiguration rdf:resource="#AspectPropertyValue_1"/>`
- `<AspectUsage:aspectConfiguration rdf:resource="#AspectPropertyValue_2"/>`
- `<AspectUsage:aspectConfiguration rdf:resource="#AspectPropertyValue_3"/>`
- `</EDOC:AspectUsage>`

- `<EDOC:ProcessComponent rdf:ID="ProcessComponent_1">`
- `<AspectableElement:usage rdf:resource="#AspectUsage_2"/>`
- `<Choreography:nodes rdf:resource="#PortConnector_1"/>`
- `<Choreography:nodes rdf:resource="#PortConnector_2"/>`
- `<UsageContext:portsUsed rdf:resource="#PortConnector_1"/>`
- `<UsageContext:portsUsed rdf:resource="#PortConnector_2"/>`
- `<PortOwner:ports rdf:resource="#FlowPort_1"/>`
- `<PortOwner:ports rdf:resource="#FlowPort_2"/>`
- `</EDOC:ProcessComponent>`

EDOC – Rules used by the Reasoner

- [r1:
• (?x rdf:type owl:Class), (?x rdf:type ns1:EClass)
• ->
• (owl:Class owl:equivalentClass ns1:EClass)
•]
- [r2:
• (?x rdf:type owl:DatatypeProperty), (?x rdf:type ns1:EAttribute)
• ->
• (owl:DatatypeProperty owl:equivalentClass ns1:EAttribute)
•]
- [r3:
• (?x rdf:type owl:ObjectProperty), (?x rdf:type ns1:EReference)
• ->
• (owl:ObjectProperty owl:equivalentClass ns1:EReference)
•]
- [r4:
• (?x rdfs:subClassOf ?y), (?x ns2:eSuperTypes ?y)
• ->
• (rdfs:subClassOf owl:equivalentProperty ns2:eSuperTypes)
•]
- [r5:
• (?x owl:inverseOf ?y), (?x ns3:eOpposite ?y)
• ->
• (owl:inverseOf owl:equivalentProperty ns3:eOpposite)
•]
- [r6:
• (?x rdfs:range ?y), (?x ns4:eType ?y)
• ->
• (rdfs:range owl:equivalentProperty ns4:eType)
•]

EDOC – Inferred Information

- equivalentClasses:

- -----

- 1 - [<http://www.w3.org/2002/07/owl#ObjectProperty>,
<http://www.w3.org/2002/07/owl#equivalentClass>,
<http://www.modeldriven.org/rdf/EMF/ECore/EReference>]
- 2 - [<http://www.w3.org/2002/07/owl#DatatypeProperty>,
<http://www.w3.org/2002/07/owl#equivalentClass>,
<http://www.modeldriven.org/rdf/EMF/ECore/EAttribute>]
- 3 - [<http://www.w3.org/2002/07/owl#Class>,
<http://www.w3.org/2002/07/owl#equivalentClass>,
<http://www.modeldriven.org/rdf/EMF/ECore/EClass>]

- equivalentProperties:

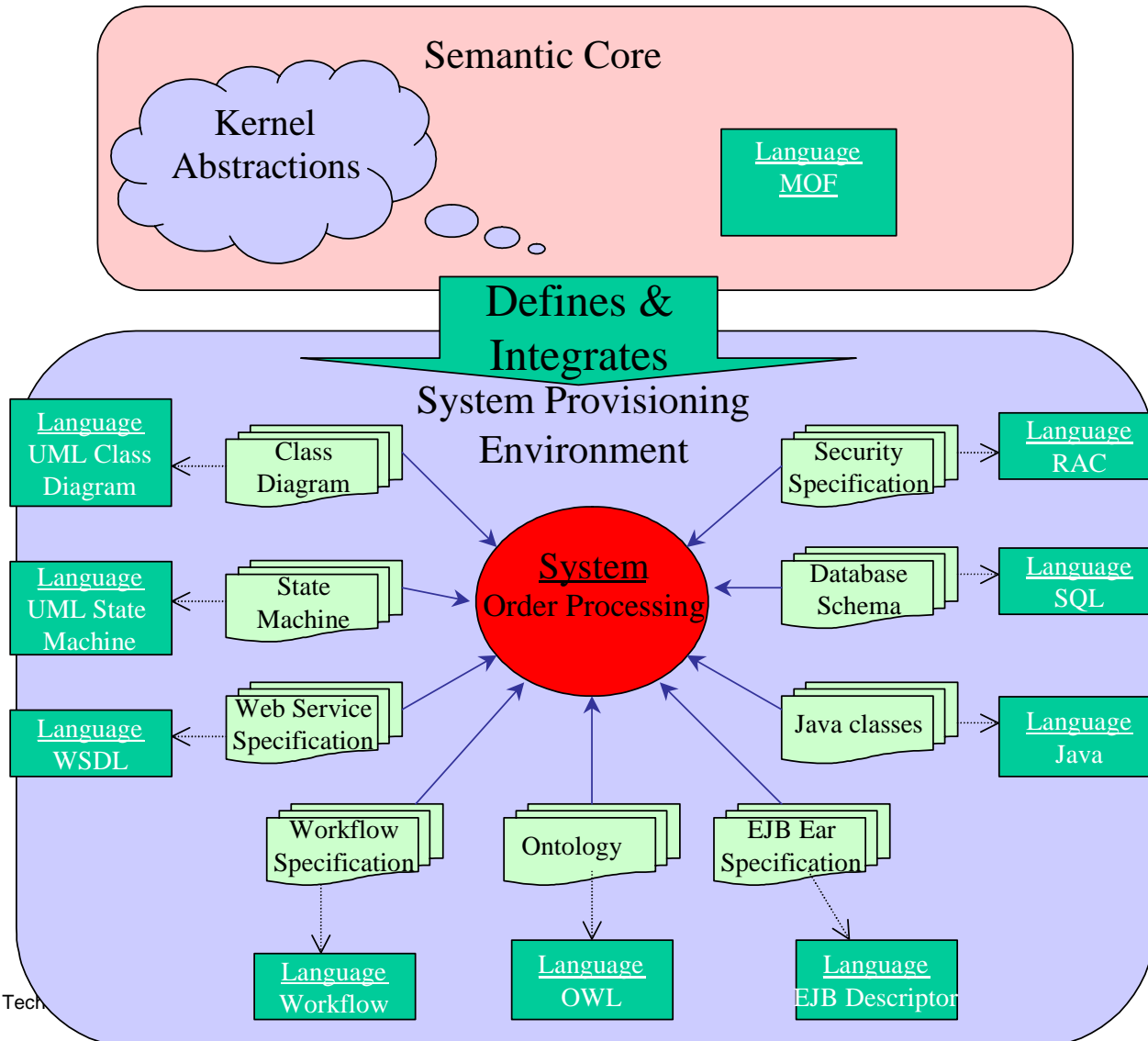
- -----

- 1 - [<http://www.w3.org/2000/01/rdf-schema#subClassOf>,
<http://www.w3.org/2002/07/owl#equivalentProperty>,
<http://www.modeldriven.org/rdf/EMF/ECore/EClass/eSuperTypes>]
- 2 - [<http://www.w3.org/2002/07/owl#inverseOf>,
<http://www.w3.org/2002/07/owl#equivalentProperty>,
<http://www.modeldriven.org/rdf/EMF/ECore/EReference/eOpposite>]
- 3 - [<http://www.w3.org/2000/01/rdf-schema#range>,
<http://www.w3.org/2002/07/owl#equivalentProperty>,
<http://www.modeldriven.org/rdf/EMF/ECore/ETypedElement/eType>]

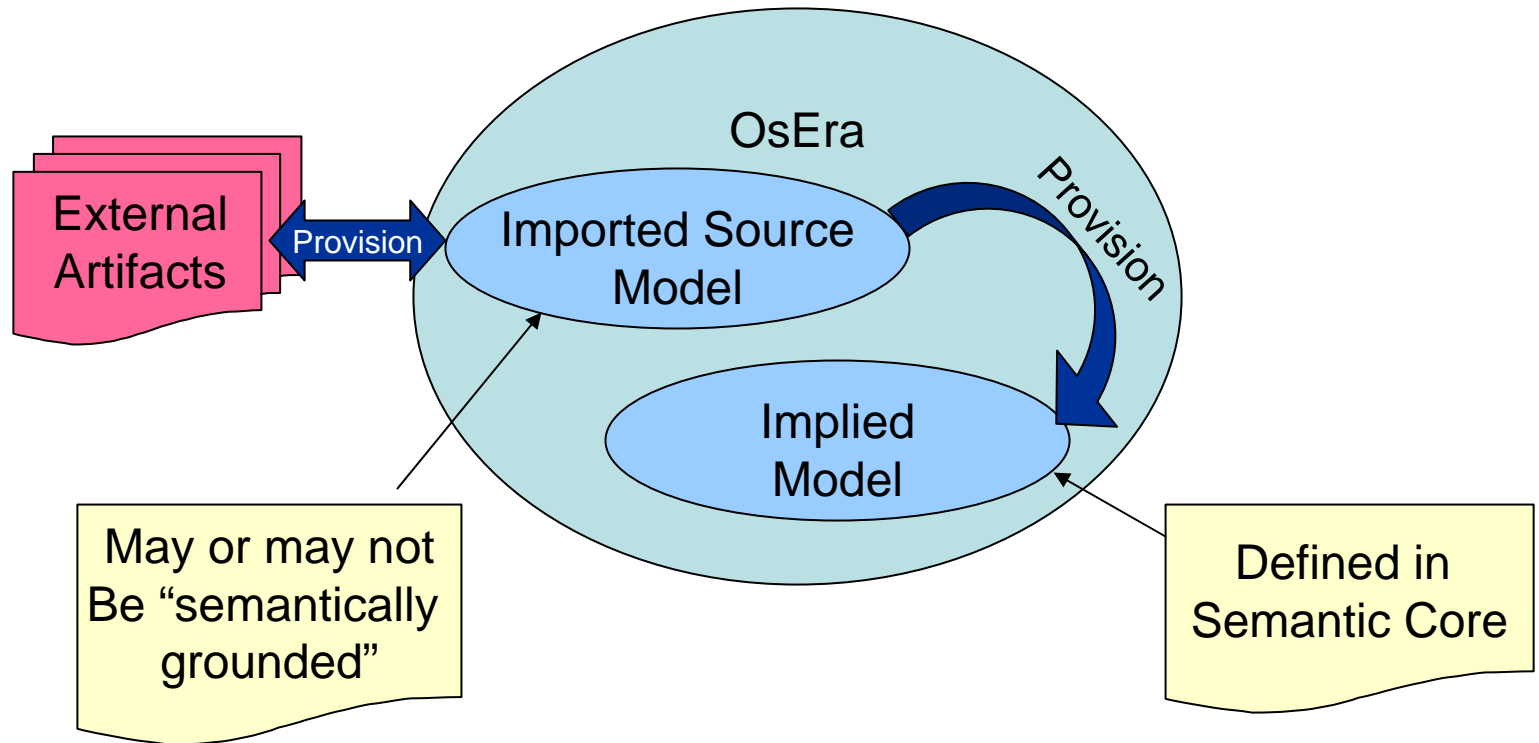
Semantic Integration

- Semantic Core
 - A reference (or “mid”) Ontology/Meta model
 - Normalizes semantics from multiple languages and metadata forms
 - Extensible as new semantics are required – not a “closed set”
 - Starting with process modeling, OWL, UML subset, collaboration modeling, information modeling and FEA (U.S. Federal Enterprise Architecture)
 - First version in place as both an Ontology and Meta Model - semantics not fully grounded
 - Discussion on “www.semanticcore.org”

Semantic Core

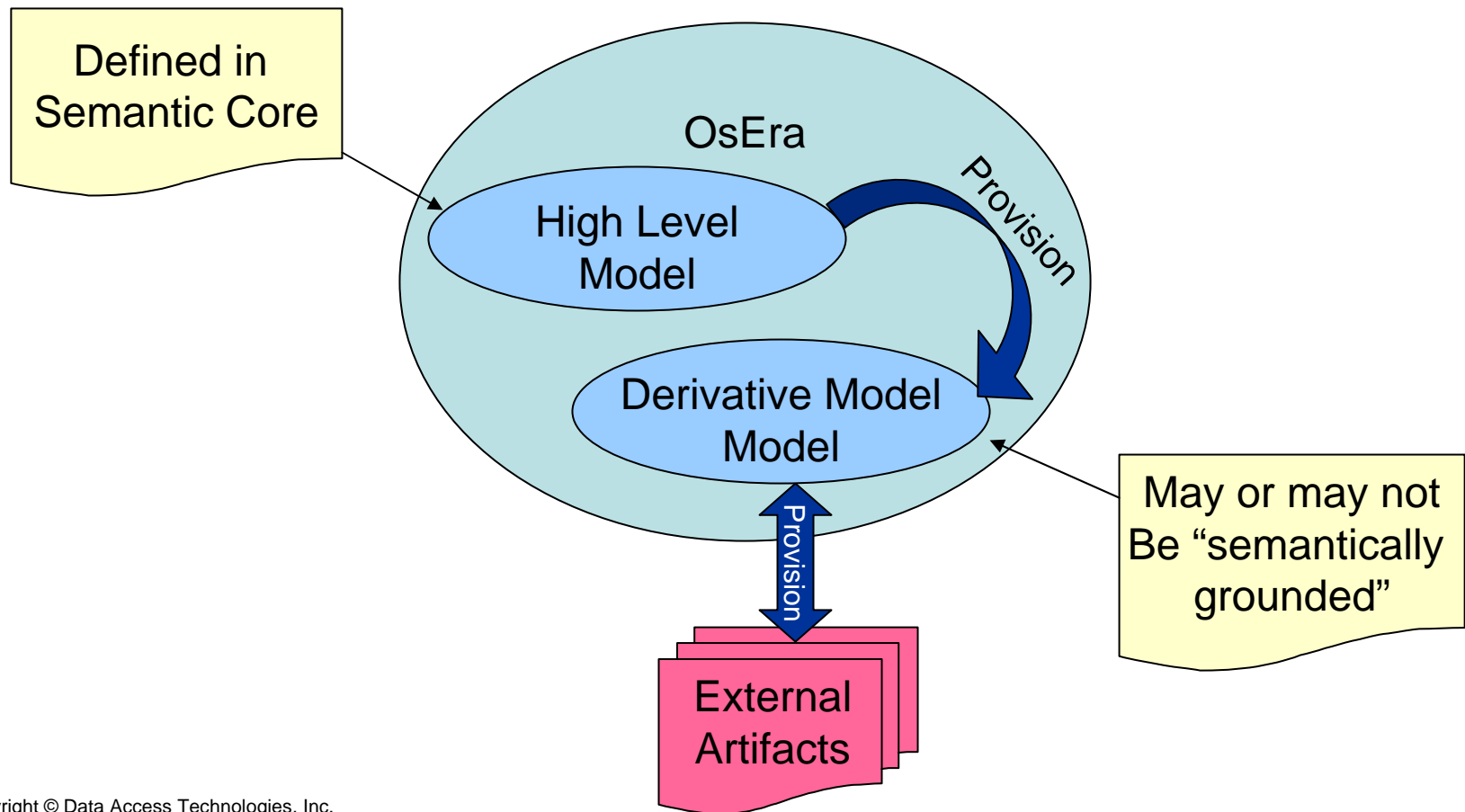


Mapping of external information into semantic core repository

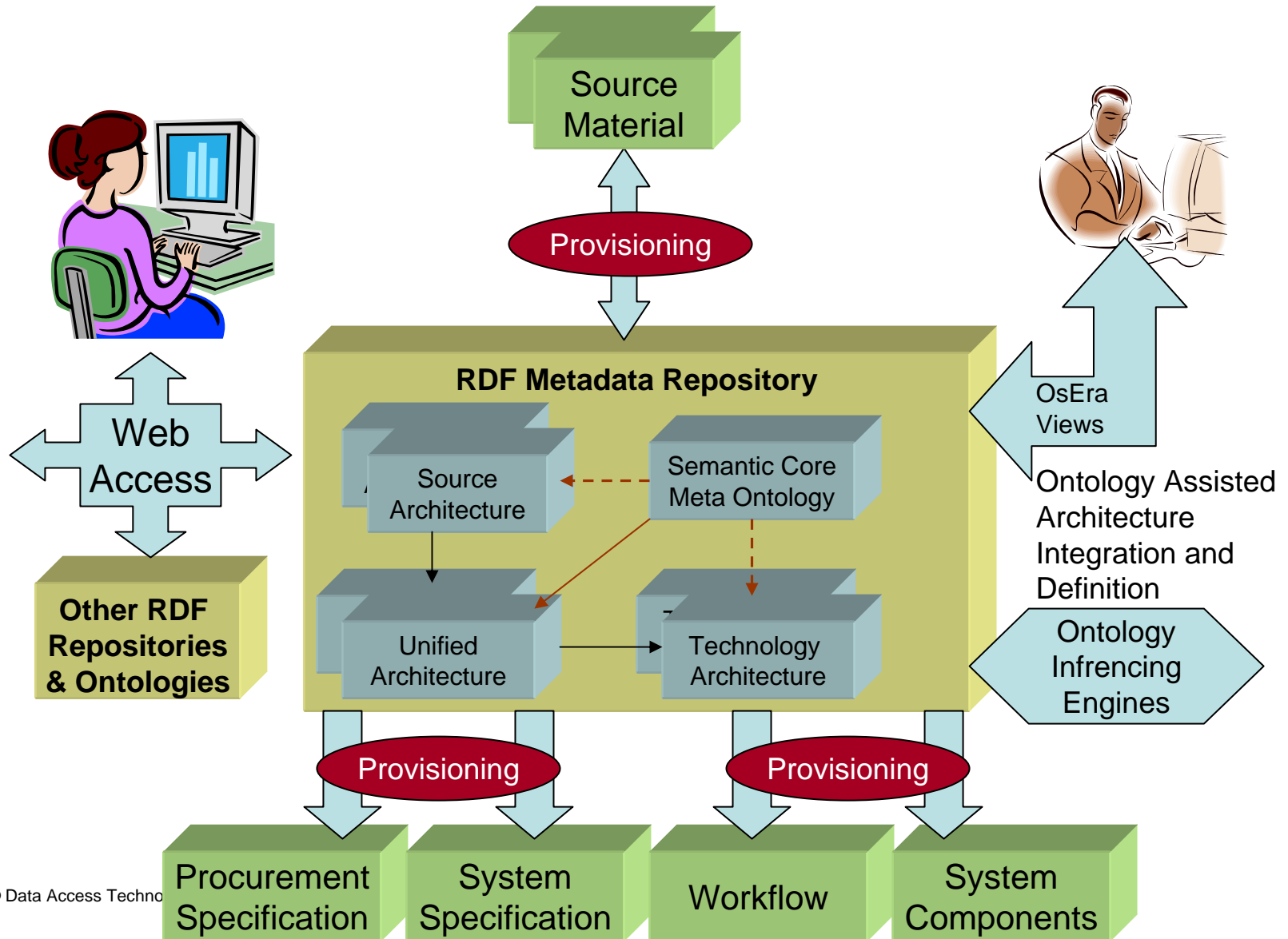


Mapping out of semantic core based models

I use “model” and “ontology” interchangeably



OsEra Metadata Infrastructure



Value Proposition

- Integration of Architectures into a unified view
- Interoperability
- 2-way integration with multiple languages
- Automated assist in grounding and integrating architectures
- MDA Provisioning out to technical infrastructure
 - Note; Does not necessarily require runtime inference

Conclusion

- Lets make the “Modeling camp” and the “Ontology camp” one
- Integrating MOF and RDF is a good start
- Speaks to users need
- Issues with OWL need resolution
- Project is just getting started, but participation welcome –
www.semanticcore.org