Project Number: 257909

SPRINT

Software Platform for Integration of Engineering and Things

Collaborative project
Information and Communication Technologies

D3.2 Definition of the Semantic Services Integration Layer

Start date of project: October, 1st 2010
Duration: 42 months

Deliverable	SSI Foundations and Definitions		
Confidentiality	PU	**Deliverable type**	R
Project	SPRINT	**Date**	2014-02-07
Status	FINAL	**Version**	1.0
Contact Person	Michael Wagner	**Organisation**	Fraunhofer FOKUS
Phone	+49 30 3463 7391	**E-Mail**	Michael.Wagner @fokus.fraunhofer.de

PROPRIETARY RIGHTS STATEMENT

THIS DOCUMENT CONTAINS INFORMATION, WHICH IS PROPRIETARY TO THE SPRINT CONSORTIUM. NEITHER THIS DOCUMENT NOR THE INFORMATION CONTAINED HEREIN SHALL BE USED, DUPLICATED OR COMMUNICATED BY ANY MEANS TO ANY THIRD PARTY, IN WHOLE OR IN PARTS, EXCEPT WITH THE PRIOR WRITTEN CONSENT OF THE SPRINT CONSORTIUM THIS RESTRICTION LEGEND SHALL NOT BE ALTERED OR OBLITERATED ON OR FROM THIS DOCUMENT

THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE SVENTH FRAMEWORK PROGRAMME UNDER GRANT AGREEMENT N° 257909

AUTHORS TABLE

Name	Company	E-Mail
Michael Wagner	Fraunhofer FOKUS	Michael.Wagner@fokus.fraunhofer.de
Daniel Hedberg	Wolfram	dhedberg@wolfram.com
Tom Ritter	Fraunhofer FOKUS	Tom.Ritter@fokus.fraunhofer.de
Christos Sofronis	ALES s.r.l.	christos.sofronis@ales.eu.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
1.0	2014-02-07	Final Version	all

CONTENT

Content of Figures

Content of Tables

Content of Appendix

1 Introduction

This document is the final public version of the restricted "D3.2 Definition of the Semantic Services Integration Layer" to be made public at the end of the project.

1.1 Overview and Purpose of the Document

The purpose of this document is to define and present the approach for the realization of the semantic layer of the integration platform of SPRINT. In order to do so, relevant requirements are identified and presented, and an architectural overview is given, followed by the description of the semantic mediation and an example. The purpose is to give a general guideline on how to realize semantic mediation in an interoperable way. This is why the approach focuses as much as possible on the application of existing semantic web technologies. The definition in this document shows how to apply them in order to achieve goal defined by the requirements from the system engineering domain. Additionally necessary extensions due to limitations of the existing technologies are defined.

This document is based on the input from

- D.2.1 SRINT Requirements
- D.3.1 SSI Foundations and Definitions

and conforms to the input from

- D.5.1 High and Low level design description of SD and SII
- D_5_4.RE.Architectural principles for Internet of System Design and the IoT.

1.2 Scope of the Document

The scope of the document is on the definition of the Semantic Services Integration Layer (SSI).

What is inside the scope of the document:

- Identification of relevant requirements
- General overview on the approach
- Definition of the semantic mediation language
- Semantic mediation example for understanding
- Integration in the Sprint platform (generic interface)

2 SPRINT Requirements

This section contains all relevant requirement input for this document. This includes high level requirements, use cases and technical requirements.

2.1 High Level Requirements

This section lists all in WP2 identified requirements relevant for Task 3.2 and this document.

ID	Description	
6.1.1	The SPRINT Engineering Environment ontology shall support, and enable the integration of, Engineering Tools from, but not limited to, the following domains: Requirements Engineering, Systems and Software Modelling, Model Simulation, Testing\Validation, Formal Verification, and Contract Authoring	
6.1.3	The SEE shall include a set of Management Tools that will allow for Cross Tool Data Browsing, and will provide Cross Tool Data net views.	
6.1.4	SEE shall support the integration of tools that are deployed using the following paradigms: Client-Server, Web Access, File Based (e.g. Rhapsody)	
6.1.7	The SPRINT platform shall be based on meta-modelling technologies for semantic mediation.	
6.1.8	The SPRINT platform shall adopt standard or de-facto standard technologies wherever available.	
6.1.9	SEE shall support the integration of third party applications (e.g. report generators and model analysis) that can access System Engineering Data, including cross tool and cross domain data, as well as create new data (e.g. system documentation and verification reports).	
6.2.2	Meta-models description shall support standards technologies such as EMF (Eclipse Modelling Framework), Ecore, Resource Description Framework (RDF) and (Web Ontology Language) OWL.	
6.2.3	The semantic integration of design models shall support a 'light' version of HRC.	
6.2.4	Shared Resources shall have the following services available to them :(1) Queries(2) Administration(3) Storage	
6.2.5	All resources shall be accessible via a URI and have a single owner.	
6.3.2	Simulation tools in SPRINT shall support interleaving semantics for the components execution.	
6.3.3	Design element shall be exchanged between tools by having their resources share the same semantic model.	
6.3.4	Tools shall share the same semantic models for resources of a common understanding when design elements need to be exchanged.	
6.4.1	SPRINT Contract tools (authoring and analysis) shall support CSL (SPEEDS CSL or other agreed contract specification means)	
6.4.6	The integration of physical devices and System designs shall be done using	

ID	Description	
	semantic mediation.	
8.2.1	The SEE shall include a service that checks consistency of I\O between a super system and its subsystems.	
8.2.2	The SPRINT Engineering Environment shall include a service that checks the consistency of I\O between models of neighbouring systems (connected subsystems in a super-system).	
8.2.3	The SEE shall include a service that will enable a system engineer to view the relevant Block Diagrams representing the environment(s) in which the system is to be integrated.	
8.2.4	Modelling Tools participating in the SPRINT Project (System Architect, Rhapsody, MathModellica and Simulink) shall each have the capability of representing instances, or objects that are specified on a remote server, and modelled by a different design tool.	
8.2.5	Modelling Tools participating in the SPRINT Project (System Architect, Rhapsody, MathModellica and Simulink) shall each have the capability of creating within the tool, a black box representation of a component that has been modelled, as a black box, using a different tool, based on data available on the SSI\SII.	

Table 2-1: SPRINT requirements considered in this deliverable

2.2 Goal

The goal of semantic mediation is the integration of information across different tools on a certain abstract level to allow cross domain cross tool development without notice to the particular developer. In an ideal scenario multiple system engineers would work on the same project with different tools without noticing the participation of the difference of the tools. The only thing they should notice is the presence of the multitude of develops. This is visualized in Figure 2-1: Semantic Mediation Experience where exemplary two system engineers collaborate in the same project via the Internet of System Engineering (IoSE). In this setting, ideally the Rhapsody utilizing system engineer should not necessarily know that the other engineer in the same project is using a different tool than him. As shown in the lower part of the picture, the model fetched through the IoSE utilizing semantic mediation should be presented to him as it would be also a rhapsody model. So that he can work with the Modelica input in Rhapsody as if it would be a native rhapsody model.

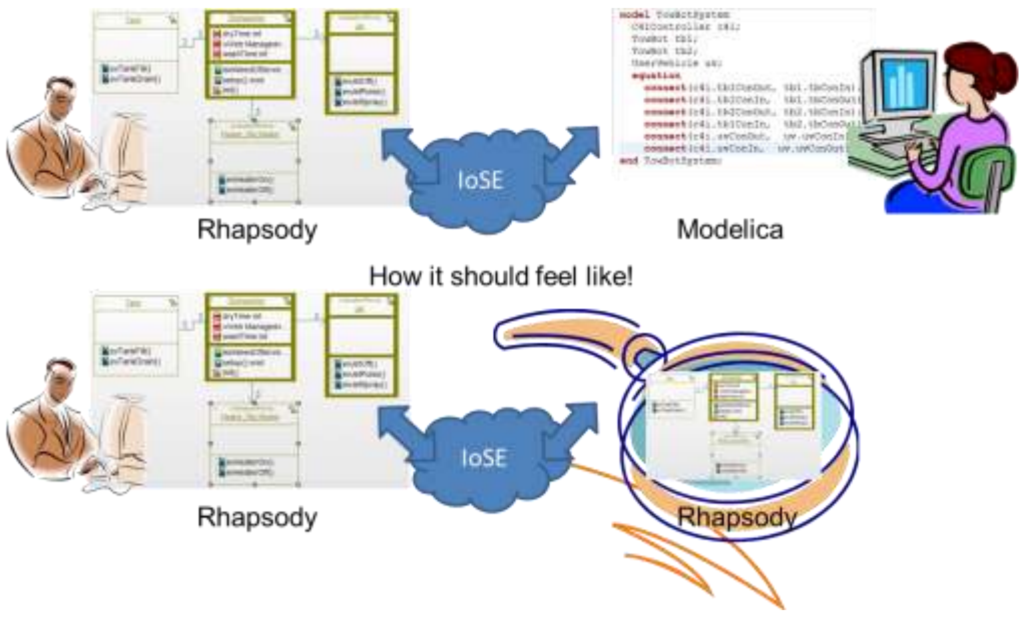


Figure 2-1: Semantic Mediation Experience

2.3 Technological Requirements

The content of this section is mostly based on the technological recommendation given in D.3.1 SSI Foundations and Definitions section 4.

2.3.1 Modelling framework

For the use of modelling framework RDF(S) and OWL as well as possible extensions have been recommended. Because:

- Since multiple classification is allowed, data can more easily be reused without redundancies
- They allows a higher degree of abstraction than for example MOF (triplets represent a very abstract and simplistic from of data representation)
- They are more fine grained and therefore more universal
- They have better support for OWA, which is necessary in the addressed setting
- They are near to the internet and evolving internet technologies (W3C)

2.3.2 Information integration approach

The sea of information approach scenario is advised because it is more flexible and extensible. Key points for this recommendation are:

- The extensibility of the approach
- Good support through the recommended modelling framework (Multi viewpoint support down to object level)
- Good coverage for most of the needed semantic mediation needs with simple and available mechanisms
- Consistency support

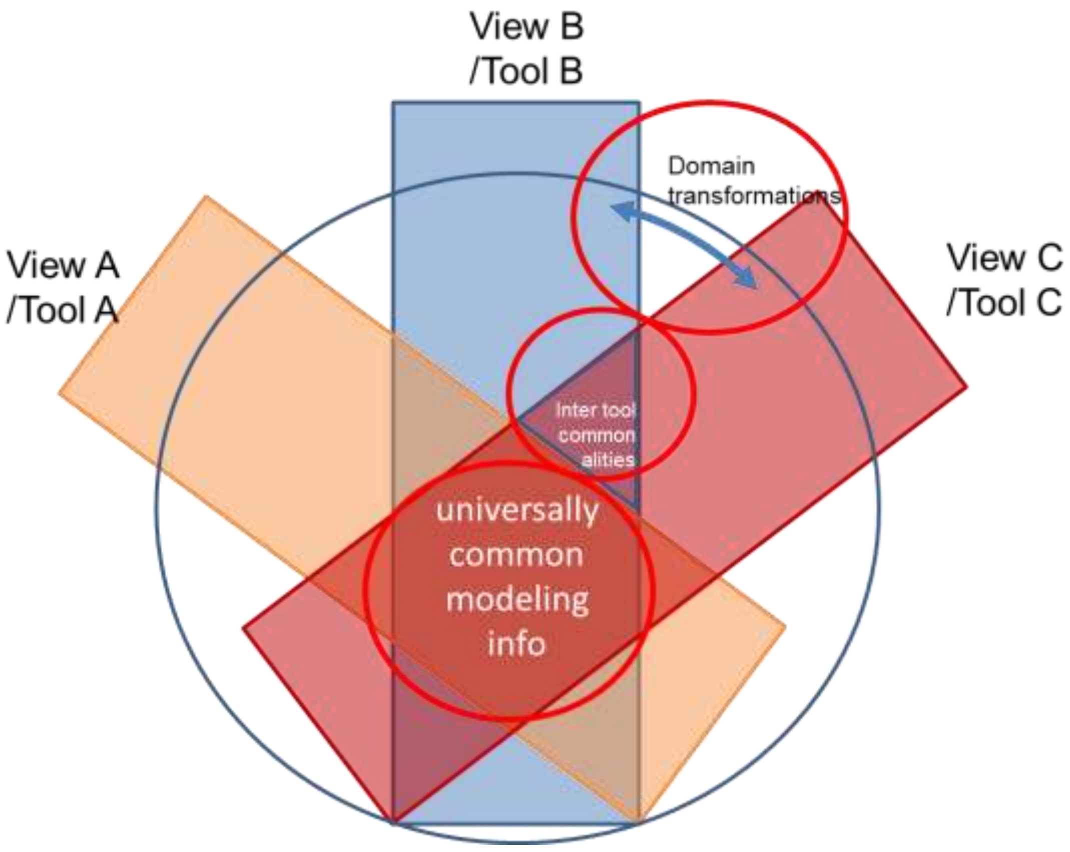
However, this approach could be enhanced with legacy functionality form the transformation chain approach. Legacy functionality support is especially necessary to support existing model transformations or to support deeper semantic mediation with very intricate rules. However, this is not in the scope of semantic mediation.

3 Semantic mediation principles

This section gives a general overview on semantic mediation motivation, the technological background and the process of application.

3.1 Semantic Mediation Motivation

In the Sprint deliverable “3.1 SSI Foundations and Definitions” [D3.1] we assessed the current situation in systems engineering with regard tool data integration. The result of this assessment was, that approaches like Common Meta-Model proposed by EU Project Speeds [Speeds] and ARTEMIS-JU Project CESAR [CESAR] or transformation chains as proposed by ModelBus [ModelBus] and in the ARTEMIS-JU Project iFEST [iFEST] had certain drawbacks which made application and management hazardous. Some of these drawbacks where inflexibility, architectural scalability, lag of extensibility, synchronization overhead, management, replication of information[1] and the amount of storage[2]. A proposed alternative to these approaches was the sea of information approach which promised to overcome the mentioned drawbacks of the other approaches. In order to achieve this, the realization of the approach would utilize technology form the language engineering domain[3]. Here people research now for many years in order to be capable to deal with very complex and diverse languages and their semantics. The combination of this technology and the approach we call now semantic mediation. Since we try to mediate between the different engineering languages based on their semantics[4] the approach is called semantic mediation.



[1] And its synchronization
[2] Basically unnecessarily consumed storage, since the information is already available in another information representation.
[3] Natural Language Processing
[4] On the level where there is semantic overlap.

Figure 3-1: Semantic Overlap

In the end of the application of this approach a semantic network is created. This semantic network can be seen as a network of languages linked to each other or one common language having multiple representations. The tooling should allow detecting inconsistencies in this network.

3.2 Technological Background

As recommended in “3.1 SSI Foundations and Definitions” [D3.1] and defined in “D5.4 Architectural principles for Internet of System Design and the IoT” the semantic mediation solution should utilize certain kinds of technology. The requirement to use RESTful HTTP for the communication plays only a second order role. The main affecting requirements and recommendations are on the data format and structuring. In this regard both documents point to the Resource Description Framework (RDF) [RDF] which the solution should be conform to. This means that the technology chosen for creating and managing the semantic network should best be based on RDF and be compliant to RDF in order to avoid unnecessary conversion work. However, this is not really limiting because the Web Ontology Language (OWL) which is the most favourable technology according to D3.1 complies with this requirement. In the internet of systems engineering it is important to rely on web technology and web standards like OWL. This technology with its Open World Assumption (OWA) should allow creating the semantic network step by step as new knowledge is coming in, avoiding wrong assumptions through incompleteness or unknown facts. It is also part of this document and the work in Task 3.2 to find out if this technology is sufficient to fulfil the requirements posed by the internet of systems engineering or if and how it needs to be extended.

3.3 General Process Overview

Since the semantic mediation and the SSI has to adhere to the RESTfull paradigm the two major operations provided for information exchange between tools are GET and the POST. In a standard semantic mediation scenario multiple tools participate with their own respective vocabularies as shown in Figure 3-2: Semantic Mediation General Overview. The POST operation performed by a tool is mainly not responsible for data conversion between tools but for data update and consistency. This means that the POST operation usually writes information more or less unchanged into the repository and modifies only other models with regard to consistency aspects (deletion of obsolete elements or properties). To do so the tool converts if necessary its information in an RDF representation conformant to its vocabulary and sends it via RESTful POST operation to the server. With this input, the server invokes the semantic mediation POST operation on the semantic mediation engine and gets as results the changes on the models to be stored in the repository. The GET operation realizes the main part of the semantic mediation. This is done through execution of semantic mediation rules in a chained manner. The tool executes a RESTful GET operation requesting a mediated model to the server. The server passes this information to the semantic mediation engine. The semantic mediation engine resolves dependencies between the rules for rule execution. These rules may create temporary or persisted instances of domain vocabularies which represent abstraction of domains and are introduced by the semantic mediation engineer for the purpose of abstraction and semantic mediation rule reduction.

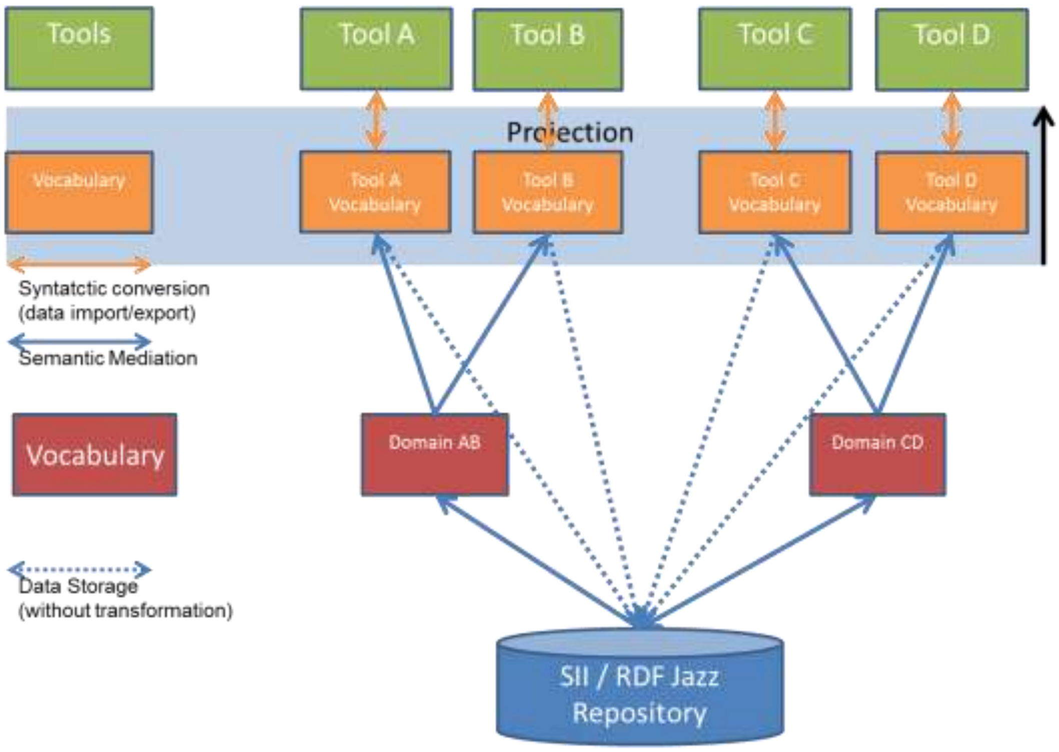


Figure 3-2: Semantic Mediation General Overview

At the end the semantic mediation has all the models and possible representation mediated (most likely in memory). Since the requesting tool does only understand it's own vocabulary and with regard to the network bandwidth its does not make sense to send all the mediated information to the tool, even though it should be capable of handling the information. Because of this, the information is semantically projected in order to focus on the tool specific vocabulary and to reduce the amount of information. The result of this projection is given to the server who responds with this information to the initial RESTful GET request of the tool.

3.4 General Operations

In the general process of semantic mediation two or at least one operations are invoked which are described in more detail in this section. Both operations GET and POST follow the REST style of software architecture proposed by the SII.

3.4.1 GET Operation

The RESTful get operation on the server receives

- the URL of the requested information,
- the mime representation type which is not of relevance for semantic mediation engine (since everything is dealt with as RDF/XML internally),
- the etag representing the last known state of the resource to the client (which is not so much of importance in the case of get),
- and the information about vocabulary understood by the client tool.

The get operation of the semantic mediation is defined as follows:

$$(\mathrm{RDFS}[0..*] \times \mathrm{RDFS}[1..1] \times \mathrm{RDF}[1..1] \times \mathrm{OWL}[0..*]) \rightarrow (\mathrm{RDF}[1..1])$$

Where the inputs:

- RDFS[0..*] : participatingSchema, the list of schemata need to be understood by the semantic mediation. This includes schemata of model information and domain model schemata utilized by the semantic mediationRules.
- RDFS[1..1] : destinationSchema, this represents the schemata understood by the tool. The result model will only contain model information adhering to this schema or its subsequent imports.
- RDF[1..1] : sourceModel, is the main source of the instance model information (A-Box).[5]
- OWL[0..*] : mediationRules, is the list of semantic mediation rules to be applied during the mediation. These rules may only rely on schemata defined in participatingSchema or on the destinationSchema.

Project to the output:

- RDF[1..1] : resultModel, is the resulting mediated model without information not understood by the destionationSchema.

3.4.2 POST Operation

The RESTful post operation on the server receives

- the URL of the requested information,
- the mime representation type which is not of relevance for semantic mediation engine (since everything is dealt with as RDF/XML internally),
- the etag representing the last known state of the resource to the client (Here the server has to ensure that the client has up to date information. This may require the client to update its information.),
- and the model information itself (adhering to the tools vocabulary).

The post operation of the semantic mediation is defined as follows:

$$(\mathrm{RDFS}[0..*] \times \mathrm{RDF}[1..1] \times \mathrm{RDF}[1..1] \times \mathrm{OWL}[0..*]) \to (\mathrm{RDF}[1..1])$$

Where the inputs:

- RDFS[0..*] : participatingSchema, the list of schemata need to be understood by the semantic mediation. This includes schemata of model information and domain model schemata utilized by the semantic mediationRules.
- RDF[1..1] : originalSourceModel, is the main original instance of the source model from the repository (A-Box) .[6]
- RDF[1..1] : updatedSourceModel, is the updated source instance model form the client tool (A-Box).
- OWL[0..*] : mediationRules, is the list of semantic mediation rules to be applied during the mediation. These rules may only rely on schemata defined in participatingSchema or on the destinationSchema.

Project to the output:

- RDF[1..1] : resultModel, is the resulting mediated model. This model and its elements may adhere do any of the participatingSchema.

[5] For now we assume only one model, though ther might be support for multiple models in the future depending on the aproach of storage.

[6] For now we assume only one model, though ther might be support for multiple models in the future depending on the aproach of storage.

4 Description of the Language for the Semantic Mediation

Because of the tool support and the scope of the language the Web Ontology Language (OWL) is proposed as semantic mediation language.[7] OWL has been designed to deal with multiple languages and its instances. It can be utilized to achieve and maintain consistency between languages utilizing an open world assumption approach allowing for iterations. This section will describe the most relevant OWL concepts for semantic mediation in detail [OWL, 2009] and how they work together with the semantic projection.[8]

4.1 Semantic Definition of the Mediation Language

This subsection will describe the most relevant OWL concepts for semantic mediation in detail.

4.1.1 equivalentClass Rule

owl:equivalentClass is an OWL build in property that links an class description to another class description in a way that both class extension contain exactly the same set of individuals.

Example:

```
< rdf:Description rdf:about="#US_President">
  <equivalentClass rdf:resource="#PrincipalResidentOfWhiteHouse"/>
</rdf:Description>
```

Usage:

The usage in semantic mediation is to project to types (classes) or subsets of types (classes with utilization of subClass) onto each other. The most basic projection is a one-to-one projection of two classes.

4.1.2 equivalentProperty Rule

owl:equivalentProperty defined in OWL is used to describe that two properties have the same property extension. Meaning that equivalent properties have same values.

Example:

```
<rdf:Description rdf:about="#title">
  <equivalentProperty rdf:resource="#name"/>
</rdf:Description>
```

Usage:

The usage in semantic mediation is the projection of two properties on one another. Sometimes only subparts of these property triplets should be projected. In this case one utilizes equivalentProperty in combination with subProperty and subClass.

4.1.3 subClassOf Rule

rdfs:subClassOff has the same meaning in OWL as in RDFS. If class Y is a subclass of X then all individuals of Y are also individuals of X.

Example:

```
<rdf:Description rdf:ID="Opera">
```

[7] This may later change or be extended.
[8] This section will not cover the whole OWL since there are plenty of books and the standard to cover it.

```
  <rdfs:subClassOf rdf:resource="#MusicalWork" />
</rdf:Description>
```

Usage:

The subClass rule is mostly used to filter out individuals based on certain properties (e.g. *Parents is subclass of People* where *Parents = People who have children*).

4.1.4 subPropertyOf

rdfs:subPropertyOf defines that the property is a subproperty of some other property.

Example:

```
<rdf:Description rdf:ID="hasMother">
  <rdfs:subPropertyOf rdf:resource="#hasParent" />
</rdf:Description>
```

Usage:

This can be used in order to create a mapping of a subset of this property which is the case if a property of one domain is mapped to two properties of another domain. It is usually used in conjunction with the RDFS subClassOf rule.

4.1.5 sameAs

The owl:sameAs property linkes one Individual with another individual. It can be used to describe that two URLs (references/identifiers) point to the same thing (have the same identity).

Example:

```
<rdf:Description rdf:about="true">
    <owl:sameAs rdf:resource="#on"/>
</rdf:Description>
```

Usage

This is especially useful for mapping individuals like enumeration literals of two different vocabularies.

4.2 Semantic Projection

The ability to relate language constructs of one language with the language constructs of another language and the ability to utilizing this knowledge in order to infer on an instance model is only one half of the semantic mediation approach. Since we do not want to waste unnecessarily network bandwidth and since the tool can only deal with the information it understands it is necessary to project the inferred model information to the domain language understood by the tool. This is done by removing all irrelevant information from the model.

4.2.1 Instance and Type Projection

In a projection all instances of an inferred model which do in any kind related via the rdf:type property to a type known to the tool are preserved in the model all others a will not appear since they are not understood by the destination domain. Additionally, only the most concrete rdf:type property relations understood by the destination domain are preserved. This means that super types which can be inferred by the tool from the actual type are removed. The same applies to sub types which are not understood by the destination domain. An example projection from BSO to Modelica is shown in Figure 4-1: Example Instance and Type Projection form BSO to Modelica.

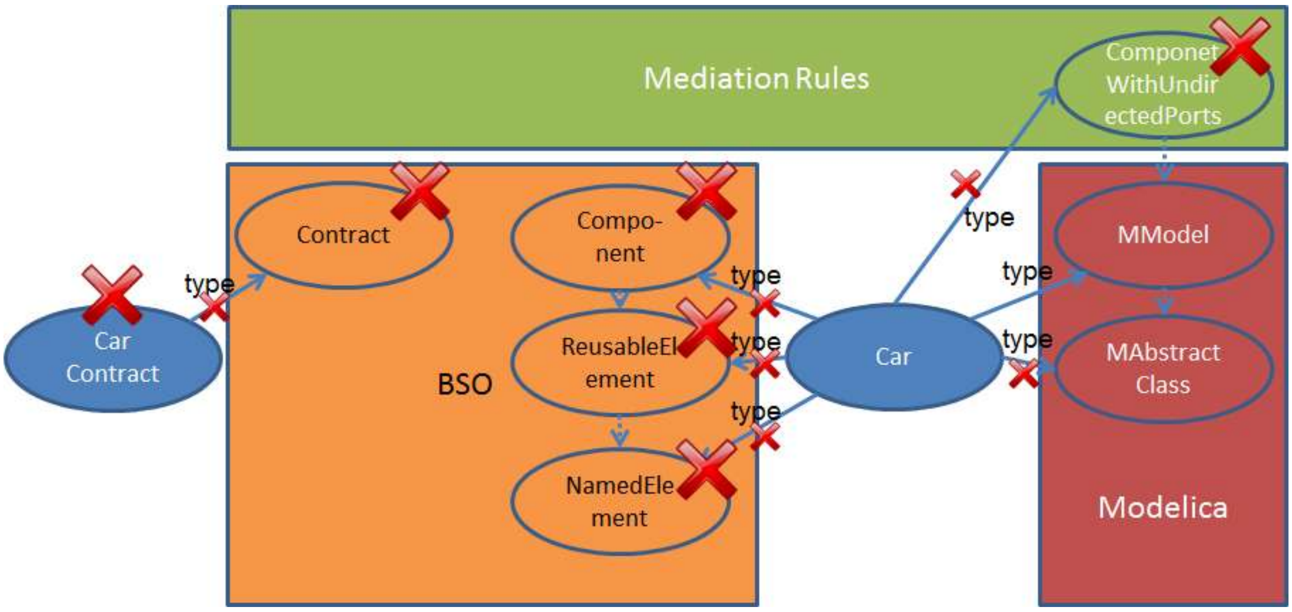


Figure 4-1: Example Instance and Type Projection form BSO to Modelica

The example shows how the car contract has to be removed since it refers to no type in the Modelica domain. Also all the BSO domain and Mediation Rule (e.g. ComponentWithUndirectedPorts) elements and its relations are removed since they are not understood by the tool. The remaining element is the car of type MModel and the Modelica Domain elements[9]. The relation to MAbstarctClass is also removed since it is redundant information already contained in the relation to MModel.

4.2.2 Property Projection

The property projection is doing basically the same as the instance projection only for properties. All triples relation to properties unknown in the destination domain are removed. Also triplet information which is redundant since the same information is present in triples of sub properties is removed. Additionally all triplets which relate to elements not projected and henceforth not understood by the destination domain are also removed.

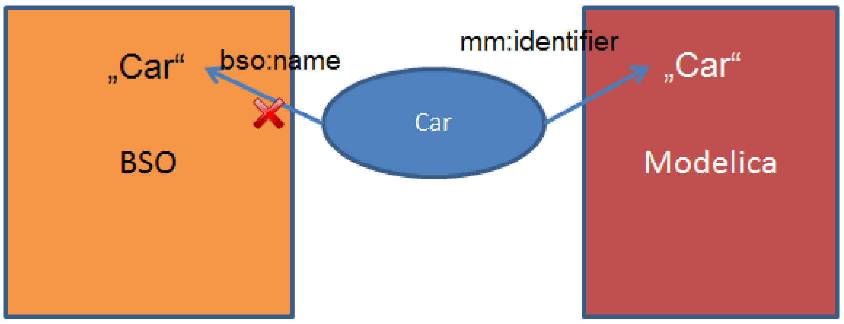


Figure 4-2: Example Instance and Type Projection form BSO to Modelica

[9] Without any additional information (only for referencing).

This simple example just shows how the name of an element in BSO is projected into Modelica. Just the triplet with the bso:name property is removed.

5 Case Study

This section contains practical examples how to apply semantic mediation in the tool environment. There is one basic example and one extended example where the extended example is based on the basic one. This constellation of examples has been chosen in order to show the extensibility of the semantic mediation, which is one of its main advantages.

5.1 Example for semantic mediation between 2 tools

This section contains the basic example for the semantic mediation between two tools. Rhapsody and SystemModeler have been chosen as tools since they are available to project participants and have relevance for the prototype.

5.1.1 Example Overview

One important characteristic of semantic mediation is the flexibility. But flexibility needs also structure and a good architecture in order to avoid spoiling the benefit of less management overhead in information integration. Because of this we have chosen to establish in our example a common structural syntax which will help us in abstracting the process of mediation. The same technique has also been proven good praxis in model transformation chains.

In the first place we have the two tools Rhapsody and SystemModeler between which we want to exchange data. Both have the capability to exchange data in RDF. The RDF/RDFS vocabulary for Rhapsody and Modelica are also given. Additionally, we have an abstraction vocabulary the BSO for language elements in the system engineering domain. In order to enable semantic mediation we define semantic mediation rules between Rhapsody and BSO as well as between BSO and Modelica.

Components of the Example:

Tools:

- Rhapsody
- Wolfram SystemModeler

Vocabularies:

- Rhapsody Vocabulary
- Modelica Vocabulary
- BSO Vocabulary (Basic Structure Ontology)

Semantic Mediation Rules:

- Rhapsody <-> BSO
- BSO <-> Modelica
- (direct way, Rhapsody <-> Modelica also possible but not advised)

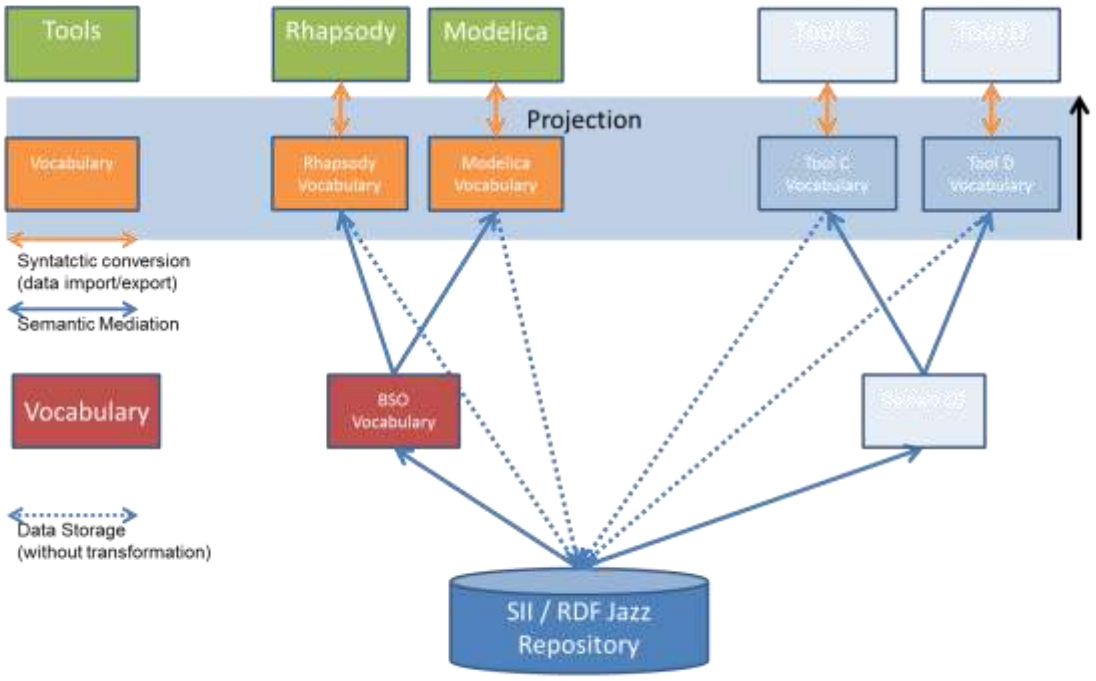


Figure 5-1: Semantic Mediation Rhapsody and Modelica Overview

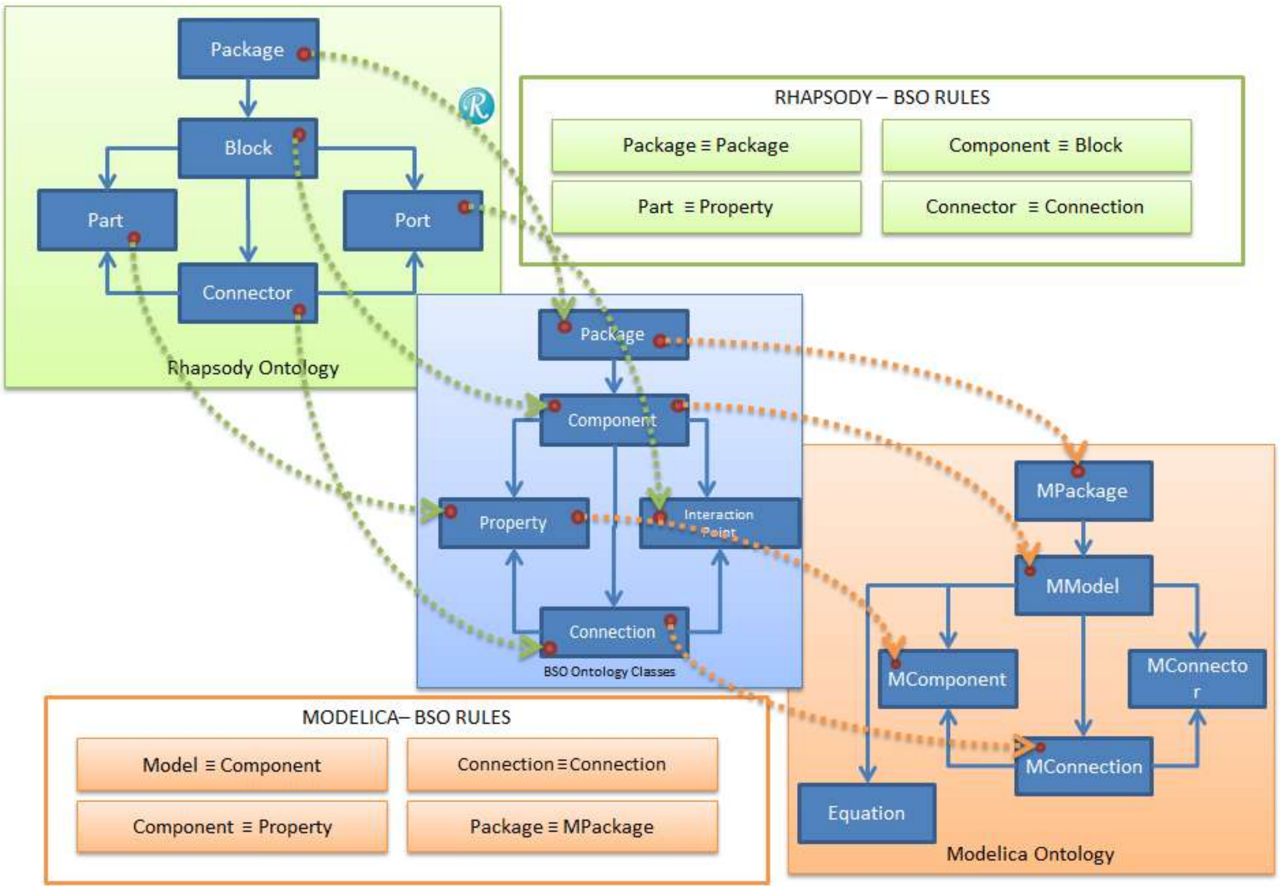


Figure 5-2: Example Semantic Mediation Rhapsody and Modelica Rule Visualization

Figure 5-2: Example Semantic Mediation Rhapsody and Modelica Rule Visualization shows an exemplary visualization of rules between Modelica and Rhapsody domain utilizing BSO.

5.1.2 Ontologies

In this section the participating ontologies are described. This includes the tool ontologies and the BSO ontology for mediation abstraction. This is a simplified example; in real world scenarios multiple ontologies for mediation abstraction might be introduced.

5.1.2.1 Rhapsody Ontology

The Rhapsody model consist out of classes, object properties and data properties described in more detail in the following sections. The classes build on top of OSLC AM. Properties from dc:terms are utilized as specified in OSLC AM.

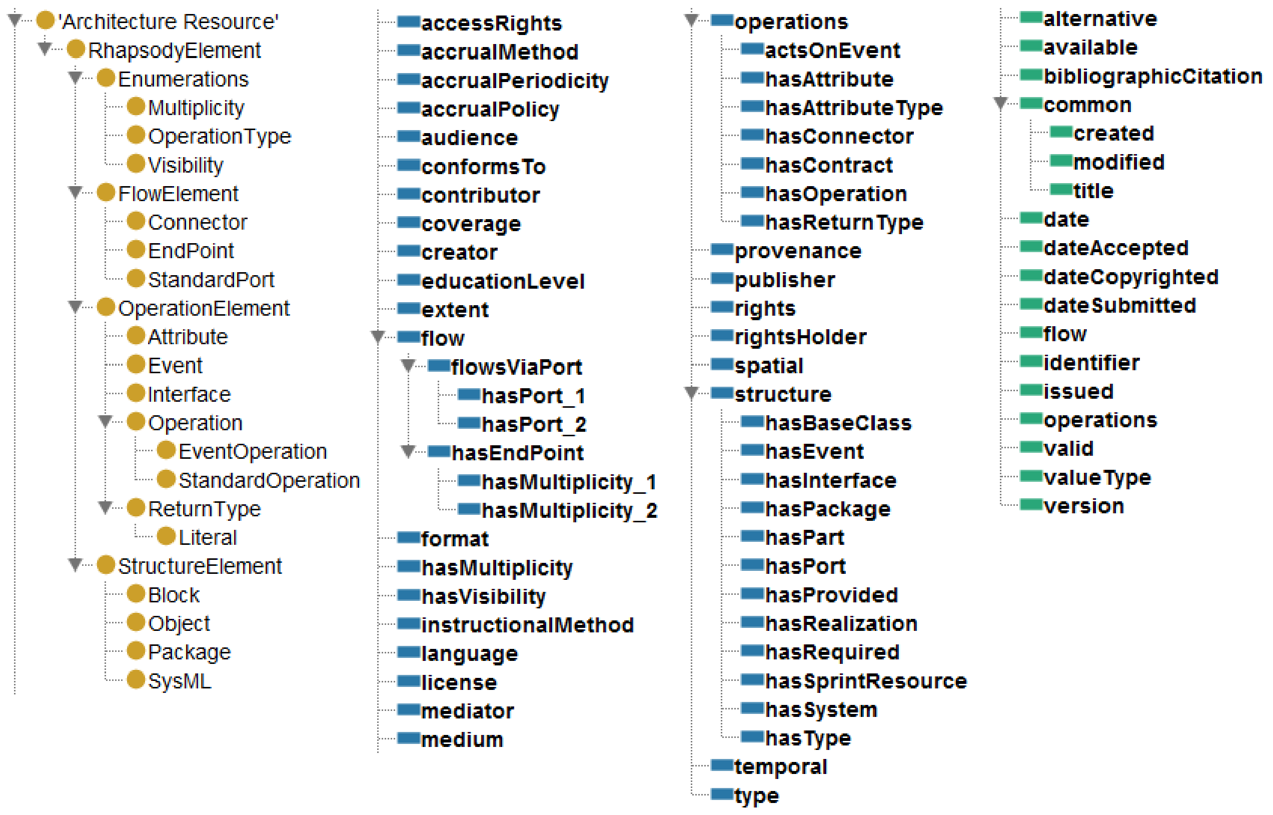


Figure 5-3: Rhapsody Ontology

5.1.2.1.1 Rhapsody Resource Types

- SysML is the project, consisting of Packages.
- A Package holds Blocks, Objects like parts, Connectors, Attributes, Events, Interfaces and PrimitiveOperations.
- Object is an instance of a Block.
- Block is a type, holds Blocks, Objects like parts, Connectors, Attributes, Events, Interfaces, StandardPorts and Operations.
- Literal is a type of an attribute or the return type of a primitive operation.
- StarndardPort is a port, it may provide and it may require Interfaces.
- Connector represents a connection between ports, associated with two end point ports.
- EndPoint is a port and part of a Connection.
- Attribute of a Block, Interface and Package.

- Interface is an interface, holds Attributes and Operations.
- Event is used in Reception (kind of an Operation).
- EventOperation is an operation in a Block and Interface, operates on Events.
- A PrimitiveOperation is an operation in a Block, Package and Interface.
- A TriggeredOperation same as PrimitivOperation, is an operation in a Block, Package and Interface.
- Reception operation in a Block and Interface, operates on Events.
- Multiplicity of connections. A connection can have AnyMultiplicity, ExactlyOnMultiplicity or OneOrMoreMultiplicity
- Visibility is the protection level of an Attribute, an Operation, a StandartPort or Enumeration. Capabilities are Private, Protected and Public.
- Realization is an implementation of an Interface. Blocks and OperationElements hold Realizations.
- OperationElement is the base type for all elements used in operations (Attributes, Events, Operations, Realizations and ReturnTypes)
- ReturnType of an operation. Literals are ReturnType for PrimitivOperation

5.1.2.1.2 Rhapsody Relations

- hasType: Object->Block: Indicates whether an Object has a type or not
- hasProvided:StandardPort->Interface:
- hasRequired:StandardPort->Interface
- hasContract:StandardPort->Interface
- hasPort_1:Connector->StandardPort: A list of StandartPorts connected with this port.
- hasPort_2:Connector->StandardPort: Same as HasPort_1.
- hasOnEvent: Reception->Event: A list of Events, the reception will react on.
- hasPackage: SysML->Package: A list of Packages.
- hasOperation: {Interface, Block}->Operation; Package->PrimitiveOperation: A list of Operations/ PrimitivOperations defined in this Interface, Block / Package
- hasAttribute:{Interface, Block, Package}->Attribute: A list of defined Attributes.
- hasReturnType:{PrimitiveOperation, TriggeredOperation}->{Block|Event|Literal}: The ReturnType for the PrimitiveOperation/TriggeredOperation
- hasAttributeType:Attribute->{Block|Event|Literal}: The Type of the Attribute
- hasConnector:{Block,Package}->Connector: A list of Connectores, which are connected with this Block/Package
- hasInterface:{Block,Package}->Interface: A list of interfaces defined in this Block/Package
- hasBlock:{Block,Package}->Block: A list of Blocks defined in this Block/Package.
- hasPart: {Block,Package}->Object: A list of parts defined in this Block/Package. Parts are instances of Blocks.
- hasPort: Block->StandardPort: A list of Ports defined in this Block
- hasEvent: {Block,Package)->Event: A list of Events defined in this Block/Package

5.1.2.1.3 Rhapsody Properties

- version-> SysML -> string: The version of the SysML specification
- hasVisibiliy: Attribute -> Visibility: The visibility of an Attribute
- hasMultiplicity_1: Connector -> Multiplicity: The multiplicity of connections linked with this Connector.
- hasMultiplicity_2: Connector -> Multiplicity: Same as hasMultiplicity_1.
- hasSprintResource: RhapsodyElement -> am:Resource: The resource that contains the element.

5.1.2.2 Modelica Ontology

The figure below is showing the structure of the Modelica meta model in a UML notation (ecore diagram) used to derive the Modelica ontology. Detailed descriptions of the elements and their meaning are given in the following sub sections.

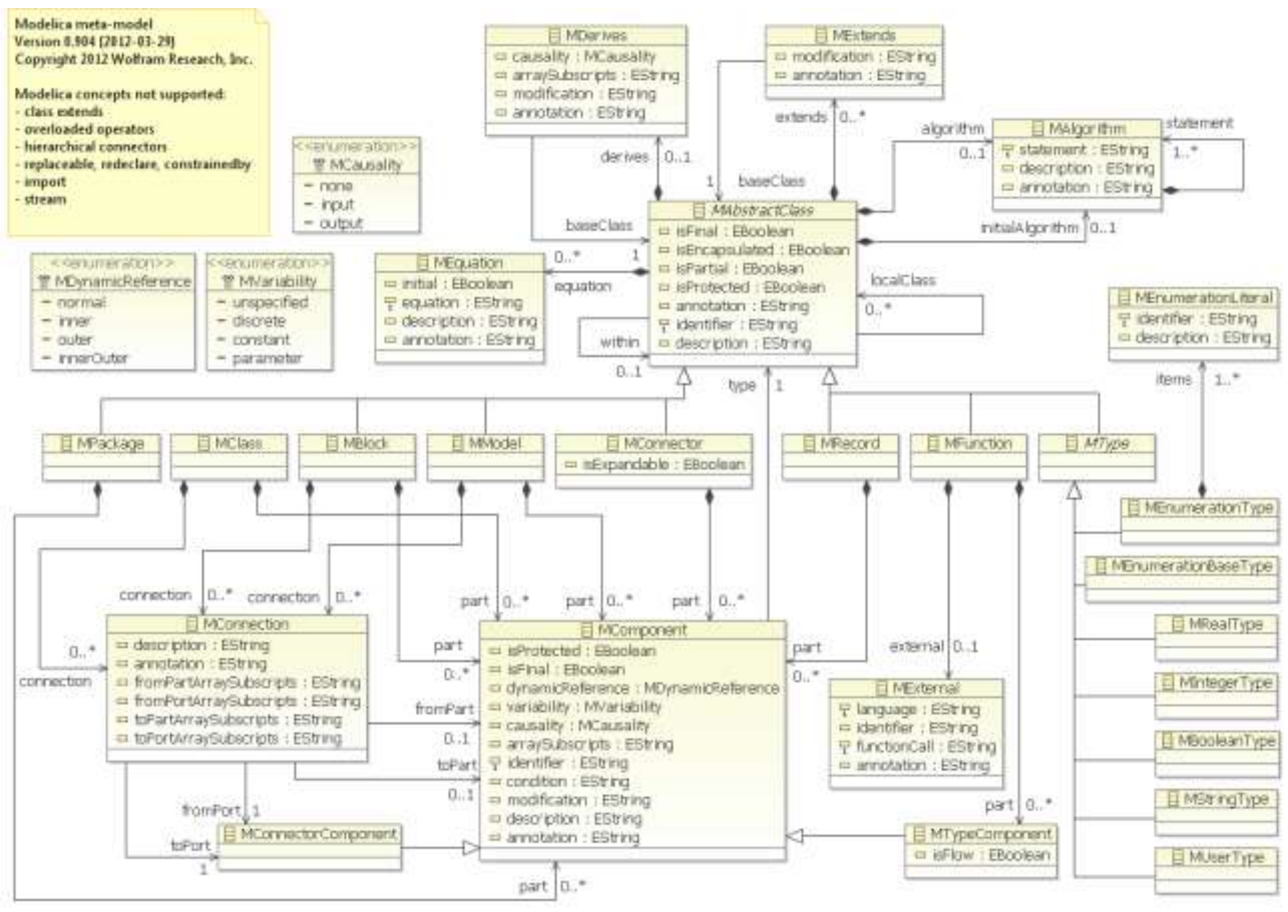


Figure 5-4: Modelica Meta Model

The main element in this model is the MAbstractClass which main concepts are MPackage, MClass, MBlock, MModel, MConnector, MRecors, MFunction, and MType. Modelica supports also data types reflected in MEnumerationType, MStringType, MBooleanType, MRealType and MIntegerType. The similar mechnisims to the SysML part element is represented in Modelica though the component, represented in the meta model through MComponent. The Modelica ecore meta model structure shown in Figure 5-4: Modelica Meta Model has been converted into an OWL ontology representation as to be seen in Figure 5-5: Modelica Ontology. This allows the usage of the model structure in combination with RDF model instances (see: Modelica Example Model)

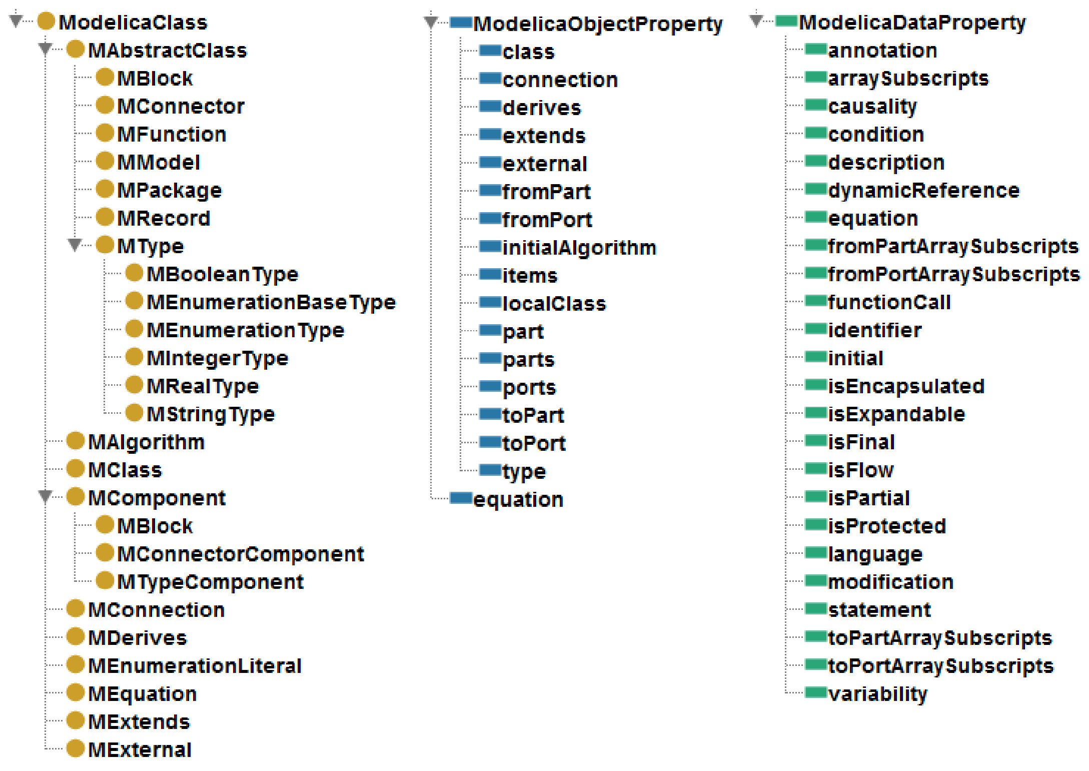


Figure 5-5: Modelica Ontology

5.1.2.3 Modelica Resource Types

- MAbstractClass the abstract base type for all Modelica classes.
- MClass is the general Modelica class.
- MModel is identical to MClass with no restrictions or enhancements.
- MBlock same as MModel with the restriction that each connector component of a block must have prefixes input and/or output for all connector variables.
- MConnector is a special kind of MClass and does not allow equations in the definition or in any of its components. Enhanced to allow connect(..) to components of connector classes.
- MFunction is a specialized Modelica class. A function has many of the properties of a general class, e.g. being able to inherit other functions, or to redeclare or modify elements of a function declaration. Holds algorithms and MComponentType for input and output.
- MConnector is a connector class used to interface points of a component. The MConnector is an endpoint of a MConnection.
- MPackage inherits MAbstractClass and is used for structural purpose. May only contain declarations of classes and constants. Enhanced to allow import of elements of packages.
- MRecord does only allow public sections in the definition or in any of its components (i.e., equation, algorithm, initial equation, initial algorithm and protected sections are not allowed). May not be used in connections. The elements of a record may not have prefixes input, output, inner, outer, or flow. Enhanced with implicitly available record constructor function. Additionally, record components can be used as component references in expressions and in the left hand side of assignments, subject to normal type compatibility rules.
- MAlgorithm is an algorithmic construct and contains a sequence of statements, defined in the algorithm section of a Modelica class.
- MDynamicReference is used to allow a Modelica component to reference another component. Values are normal, inner, outer, and innerOuter.

- MCausality is used to define the computational causality of a function body, the causality of variables in connectors and the interaction with the simulation environment in top level models and blocks. For models and blocks used as components, MCausality can also be used to define that a binding equation has to be provided for a variable when the component is utilized. Values are input, output and none.
- MVariability defines in which situation the variable values of a component are initialized.
- MExtends is used to describe inheritance between Modelica classes, i.e., classes extending the behaviour and properties of other classes.
- MDerives is used to describe short-class definitions in Modelica.
- MEquation is an equation defined in the equation section or initial equation section of a Modelica class, with the exception of connect-equations which are represented by MConnection.
- MType is the abstract base type for all Modelica predefined types, enumerations, array of type, and classes extending from a Modelica type.
- MEnumerationType is a class for custom enumerations.
- MStringType type for strings.
- MBooleanType type for Boolean numbers.
- MRealType type for Real numbers.
- MIntegerType type for integer numbers.
- MUserType is a user defined type extending from a Modelica type.
- MExternal mathematical function, defined in another language.
- MComponent is an instance of a MClass, MBlock, MModel, or MRecord.
- MTypeComponent is an instance of either a predefined type such as MRealType, or a class derived from a predefined type.
- MConnection is a connection between objects and introduced by connect-equations. Holds two MConnectorCompontents for source and target ports.
- MConnectorComponent is an instance of a MConnector and is part of a connection between components.

5.1.2.4 Modelica Relations

- equation: MAbstractClass -> MEquation: The list of equations defined in this MAbstractClass.
- derives: MAbstractClass -> MDerives: Return the base class of this MAbstractClass, if it is extending a class using a short-class definition.
- extends: MAbstractClass -> MExtends: The list of base classes of this MAbstractClass, if it is extending other classes.
- algorithm: MAbstractClass -> MAlgorithm: Return the algorithm if one is defined in MAbstractClass MAbstractClass.
- initialAlgorithm: MAbstractClass -> MAlgorithm: Returns the initial algorithm if one is defined in the MAbstractClass. Initial algorithms are part of the initialization process, assigning consistent values for all variables present in the model.
- localClass: MAbstractClass -> MAbstractClass: The list of classes defined in this MAbstractClass.
- whitin: MAbstractClass -> MAbstractClass: The enclosing class of this MAbstractClass unless this Modelica class is a root class.
- statement: MAlgorithm -> Malgorithm: The list of statements used for the algorithm.
- connection: {MClass, MBlcok, MModel} -> MConnection. The list of connections linked with this MClass/MBlock/MModel.
- fromPort: MConnection -> MConnectorComponent: The source MConnectorComponent of a connection.
- toPort: MConnection -> MConnectorComponent: The target MConnectorComponent of a connection.
- fromPart: MConnection -> MComponent: The source MComponent (if any) of this connection.
- toPart: MConnection -> MComponent: The target MComponent (if any) of this connection.

- part: {MPackage, MClass, MModel, MBlock, MConnector, MRecord} -> MComponent: A list of MComponents defined in this MPackage/MClass/MModel/MBlock/MConnector/MRecord.
- part: MFunction -> MTypeComponent: The list of MTypeComponents defined in this function.
- external: MFunction -> MExternal: Return an MExternal element if this function is an external function.
- item: MEnumerationType -> MEnumerationLiteral: The list of literals defined in this enumeration.

5.1.2.5 Modelica Properties

- causality: {MDerives, MComponent} -> MCausality: A causality can be input, output or none
- arraySubscripts: {MDerives, MComponent} -> EString: The array subscripts if the Modelica element is an array.
- modification: {MDerives, MExtends, MComponent } -> EString: Modifications to the Modelica class or Modelica component.
- description: {MDerives, MExtends, MEquation, MAbstractClass, MAlgorithm, MEnumerationLiteral, MConnection, MComponent } -> EString: A short description of this Modelica element.
- annotation: {MDerives, MEquation, MAbstractClass, MAlgorithm, MExternal, MConnection, MComponent } -> EString: Annotations for the Modelica element.
- equation: MEquation -> EString: The list of equations defined in this Modelica class.
- initial: MEquation -> EBoolean: True if this equation is an initial equation that is part of the initialization process, assigning consistent values for all variables present in the model.
- isFinal: {MAbstractClass, MComponent}-> EBoolean: True if this Modelica element is final and cannot be modified.
- isProtected: {MAbstractClass, MComponent} -> EBoolean: True if the protection level for this Modelica element is protected. Root Modelica classes cannot be protected.
- isPartial: MAbstractClass -> EBoolean: True if this Modelica class is a partial (incomplete) class.
- isEncapsulated: MAbstractClass –> EBoolean: True if this Modelica class is encapsulated.
- isFlow: MTypeComponent -> EBoolan: True if this component (which must be of a subtype of Real) is a flow-variable.
- identifier: {MAbstractClass, MEnumerationLiteral, MExternal, MComponent} -> EString: A unique and atomic (non-composed) name for this Modelica element.
- statement: MAlgorithm -> EString: An (algebraic) statement in an algorithm.
- language: MExternal -> EString: The (computer) language the external function is writtin in.
- functionCall: MExternal -> EString: The function call of the external function.
- fromPartArraySubscripts: MConnection -> EString: The array subscripts (parameter-expressions) if the source component of the connection is an array of components.
- toPartArraySubsrcipts: MConnection -> EString: The array subscripts (parameter-expressions) if the target component of the connection is an array of components.
- fromPortArraySubsrcipts: MConnection -> EString: The array subscripts (parameter-expressions) if the source connector of the connection is an array of connectors.
- toPortArraySubsrcipts: MConnection -> EString: The array subscripts (parameter-expressions) if the target connector of the connection is an array of connectors.
- condition: MComponent -> EString: A boolean scalar parameter-expression associated with the component. If true, the component is to be ignored.
- dynamicReference: A dynamic reference can be inner, outer and innerouter.
- variability: MComponent -> MVariability: A variability can be unspecified, parameter, constant or discrete.

5.1.2.6 Basic Structural Ontology

The figure below is showing the structure of the Basic Structural Ontology (BSO) meta model in a UML notation (ecore diagram) used to derive the Basic Structural Ontology. Detailed descriptions of the elements and their meaning are given in the following sub sections. The ontology representation has been derived the same way as the Modelica ontology and with the same purpose.

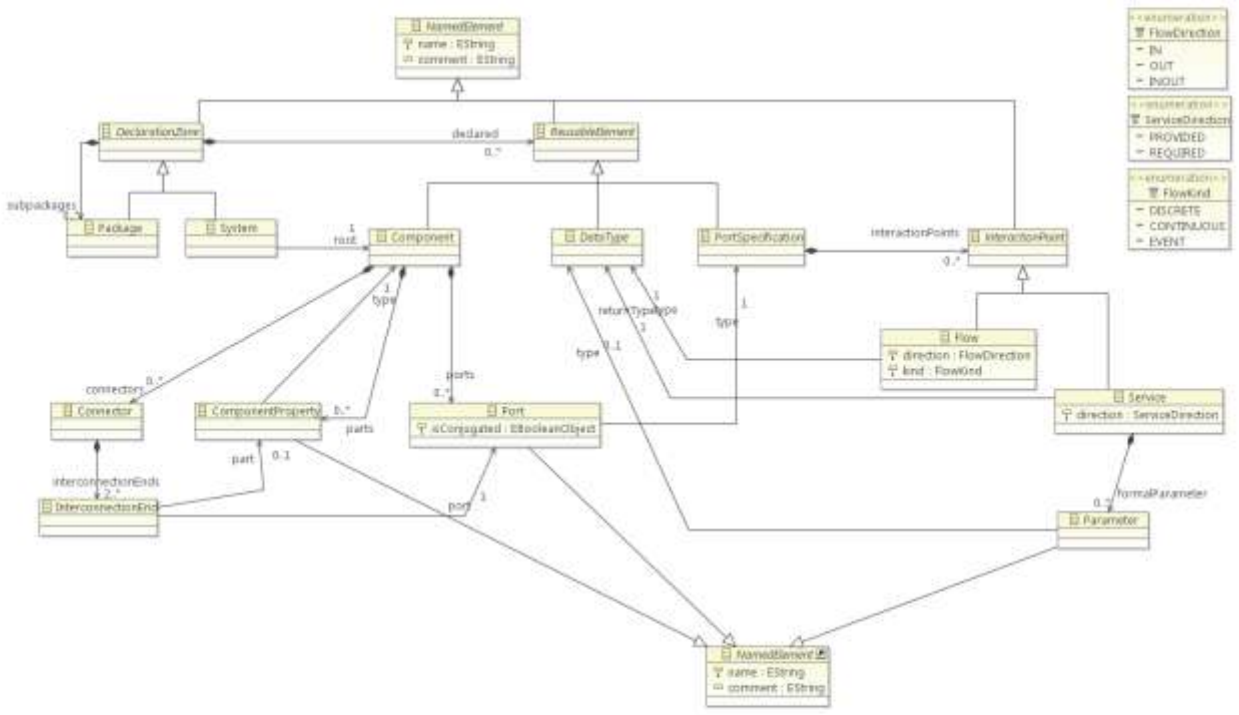
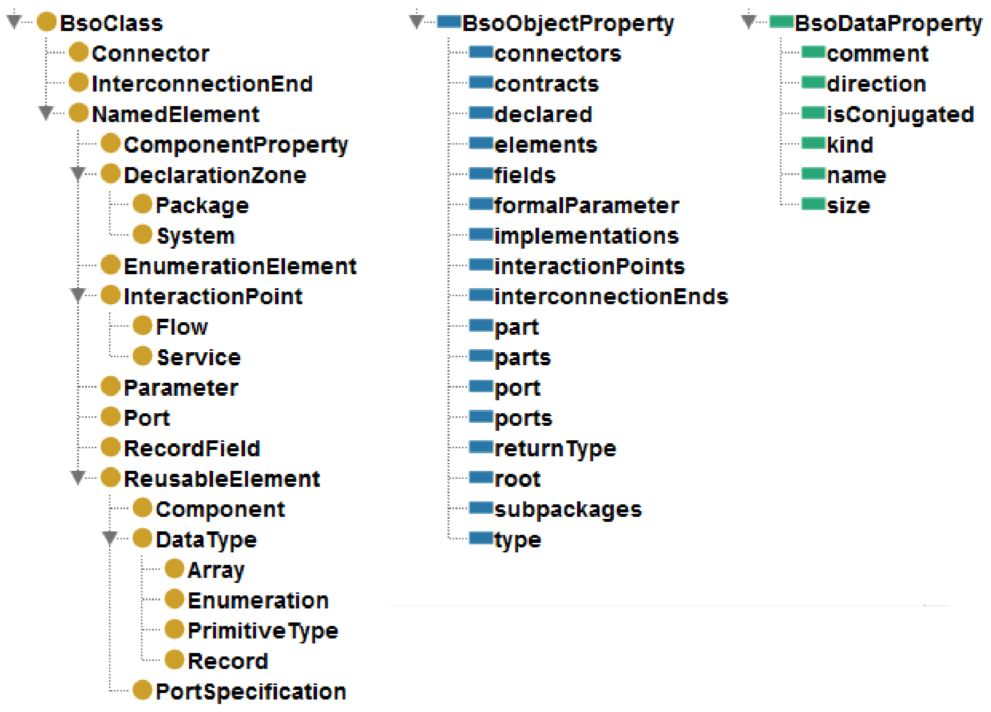


Figure 5-6: Basic Structural Ontology Meta Model



```
BsoClass                          BsoObjectProperty          BsoDataProperty
  Connector                         connectors                 comment
  InterconnectionEnd                contracts                  direction
  NamedElement                      declared                   isConjugated
    ComponentProperty               elements                   kind
    DeclarationZone                 fields                     name
      Package                       formalParameter            size
      System                        implementations
    EnumerationElement              interactionPoints
    InteractionPoint                interconnectionEnds
      Flow                          part
      Service                       parts
    Parameter                       port
    Port                            ports
    RecordField                     returnType
    ReusableElement                 root
      Component                     subpackages
      DataType                      type
        Array
        Enumeration
        PrimitiveType
        Record
      PortSpecification
```

Figure 5-7: Basic Structural Ontology

5.1.2.6.1 BSO Resource Types

- System is a DeclarationZone that has a root RichComponent. The root represents the highest-level component of the system. This means that the system can be instantiated by instantiating its root component and all the sub-components (parts) of the root, recursively.
- Component describes a type of sturctual units that share the same characterization of features, constraints, and dynamics
- Port is owned by a RichComponent. It contains a set of interaction points of the owning rich component. A port is typed by a PortSpecification, which specifies the set of interaction points in terms of flows or services.
- ComponentProperty specifies a part/sub-component of a RichComponent. It can be also seen as instance of a RichComponent within another (different) RichComponent. The former is the one referenced by the type reference whereas the latter is the owning RichComponent.
- Connector Represents the link between two ports; hierarchical or between two instances belonging on the same component.
- PortSpecification. The aggregation point of InteractionPoints, which is used to type Port objects.
- Flow can be descrete, continuous or event
- FlowKind is the datatype of the information carried by a flow
- Service is the declaration of a service that is provided or required by a RichComponent through a PortSpecification.
- ServiceDirection specifies whether the service is provided or required
- Datatype is for the characterization of the flows and services of the ports.
- DeclarationZone is mainly a container for subpackages and ReusableElements. In particular the System and Package classes specialize this class. The class inherits from UniquelyIdentifiableElement and all its instances should be able to be uniquely identified in a given context.
- Flow. One of the two InteractionPoint specializations (the otherone being the Service). A flow is characterized by its data type, its kind, and its direction. The data type defines the type of values that appear on the flow. The kind defines when values are available on the flow, and how they change. And the direction defines whether the owning rich component does input/output/both on the flow.
- InteractionPoint denotes the single interaction point in a Port (a Port is typed by a PortSpecification that contains the interaction points). It can be either a Flow or a Service
- InterconnectionEnd represents the end of a connector. It can be used to represnt both hierarchical (from owning RichComponent to one of its parts) and same-level (between parts of the same RichComponent) connections
  - For the hierarchical the RichComponent and the Port shall be used;
  - For the internal connection the RichComponentProperty and the Port shall be used.
- NamedElement provides a name and optional comment to the inherited class.
- Package is a DeclarationZone that is not a System. The main difference between the System and the Package is that the former can be considered as the main reference for a System specification since it provides a reference to the top-level RichComponent
- Parameter. Instances of Parameter, are formal parameters in the model. Parameters are named elements with a DataType that indicates the intended type of the actual elements
- ReusableElement is an element that has a name and that is directly owned by a DeclarationZone. It is reusable in the sense that it can be used in different contexts (e.g., multiple instantiation of a RichComponent).

5.1.2.6.2 BSO Relations

- Type: {ComponentProperty, Flow, Paramter, Port}->{Component, DataType, DataType, PortSpecification}: The type of the class carried with the flow
- declared: {Package, System, DeclarationZone}->{PortSpecification, Component, ReusableElement}: A list of PortSpecification, Component or ReusableElement objects

- subpackages:{Package, System, DeclarationZone } -> Package: The list of subpackages of this declaration zone.
- connectors: Component->Connector: The list of connections of the RichComponent. There are two types of connections:
    1. the hierarchical ones that connect the ports of this RichComponent and the ports of the instances contained by it and
    2. the connections between ports of the contained parts (instances of other RichComponents)
- part:Component->ComponentProperty: The set of instances/parts that are contained by this RichComponent. This is basically part of the Gray box specification.
- interconnectionEnds:Connector->InterconnectionEnd: The ends that are connected with this connector.
- ports:Component->Port: The ports of the interface of the RichComponent
- interactionPoints:PortSpecification-> interactionPoints: The set of Flow and Services that are contained by this PortSpecification.
- formalParameter: Service -> Paramter: The set of formal parameters (input arguments) of the service invocation.
- returnType: Service ->DataType: The datatype of the values returned by this service.
- Root: System -> Component: The list of subpackages of this declaration zone

5.1.2.6.3 BSO Properties

- Comment: { ComponentProperty, Component, DataType, DeclarationZone, Flow, InteractionPoint, NamedElement, Package, Paramter, Port, PortSpecification, ReusableElement, Service,System } ->EString: An optional comment of the instance
- Name:{ComponentProperty, Component, DataType, DeclarationZone Flow, InteractionPoint, NamedElement, Package, Paramter, Port, PortSpecification, ReusableElement, Service, System } -> EString: The String representation of the object's name. The field is required and should be unique (additional post-processing check needed).
- Direction: {Flow, Service}->{FlowDirection, ServiceDirection}: A flow can be input, output or bidirectional.
- Kind: {Flow}->FlowKind: a flow can be discrete, continuous or event.
- Part: InterconnectionEnd -> ComponentProperty
- Port: InterconnectionEnd -> Port
- isConjugated: Port -> EBooleanObject: PortSpecification. This allows using single PortSpecification typing for ports that are connected.

5.1.3 Example Models

This sections shows for better understanding example models in the different tool notations. It also shows in a practical way what we want to achieve with semantic mediation and how it should look like. In the example models we concentrate on components (e.g. Bocks), connectors (e.g. Flows) and ports since this are the main interoperable engineering elements.

5.1.3.1 Rhapsody Example Model

The Rhapsody model consist out of a main bock definition M which and two sub lock definition M1 and M2. Both block definitions M1 and M2 are used as parts in the block definition of M. In order to show the directionality support each sub block definition has an in port and an out port (e.g. p1a, p1b, p21, p2b) with different types (e.g. int and double). The ports of the parts of these sub block definitions which are used in M are connected appropriately with regard to type and direction.

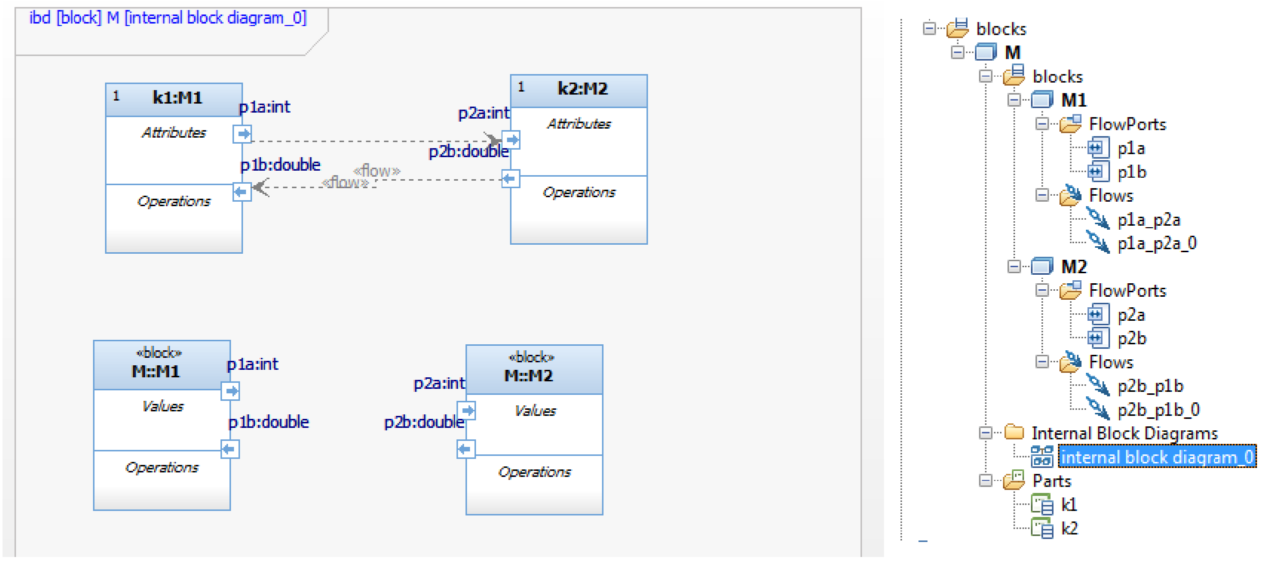


Figure 5-8: Rhapsody Example Model

For detailed understanding RDF examples of the models are shown

5.1.3.1.1 Package

A package has for example a tile and can contain blocks.

```
<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-66341366-c102-4a08-a2b4-e2f74f8767cb"
  dc:title="Default"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Package"/>
  <rhp_model:hasBlock rdf:resource="http://com.ibm.rhapsody/sprint/GUID-0208a299-1530-4e1f-aa76-03c71b1c38f2"/>
  </rdf:Description>
```

5.1.3.1.2 Block

A block for example can contain flows and flowports.

```
<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-fff2e05a-168f-4246-9a76-52b14647f403"
  dc:title="M1"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Block"/>
  <rhp_model:hasFlow rdf:resource="http://com.ibm.rhapsody/sprint/GUID-b10f270b-d2b3-462b-a339-5dc597c7683d"/>
  <rhp_model:hasFlow rdf:resource="http://com.ibm.rhapsody/sprint/GUID-1ebf7201-d4a9-4644-a46a-99bc7d398fc0"/>
  <rhp_model:hasFlowPort rdf:resource="http://com.ibm.rhapsody/sprint/GUID-437271cd-5f5c-4325-bf65-552f7bc349da"/>
  <rhp_model:hasFlowPort rdf:resource="http://com.ibm.rhapsody/sprint/GUID-16b7ddaa-6d51-434d-8566-564628902614"/>
  </rdf:Description>
```

5.1.3.2 Modelica Example Model

The Modelica model represents basically the same as the Rhapsody only the element types are different. The Modelica model consists of a model M and two blocks M1 and M2. Both M1 and M2 contain connector components (e.g. p1a, p1b, p2a, p2b). M uses the blocks M1 and M2 as components k1 and k2. The connector components are connected appropriately via connections.

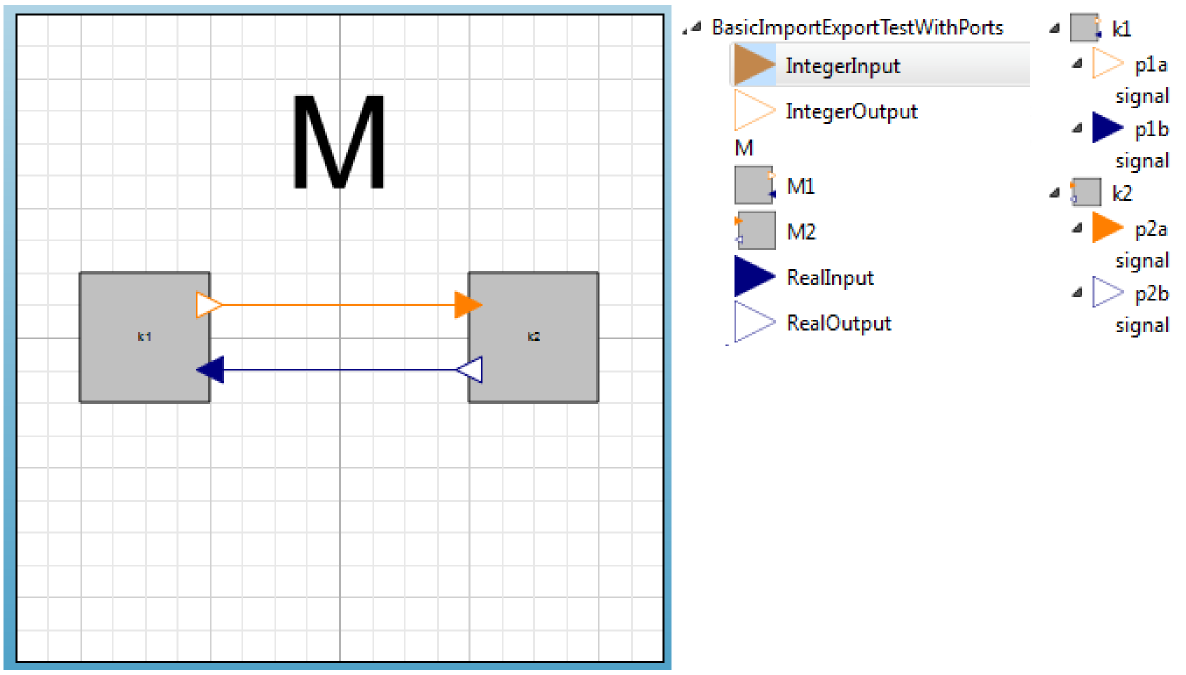


Figure 5-9: Modelica Example Model

For detailed understanding RDF an example of the model in turtle are shown

5.1.3.2.1 Model

A model can have for example parts and connections.

```
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                <http://www.mathcore.com/sm-rdf/MModel> .
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>
        <http://www.mathcore.com/sm-rdf/identifier>
                "M" .
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>
        <http://www.mathcore.com/sm-rdf/within>
                <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts> .
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>
        <http://www.mathcore.com/sm-rdf/part>
                <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k1> .
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>
        <http://www.mathcore.com/sm-rdf/part>
                <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k2> .
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>
        <http://www.mathcore.com/sm-rdf/connection>
                <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-1> .
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>
        <http://www.mathcore.com/sm-rdf/connection>
                <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-2> .
```

5.1.4 Semantic Mediation Rules

This section contains the semantic mediation rules for the mediation between Rhapsody and Modelica via BSO.

5.1.4.1 Rhapsody and BSO

This Section contains the sematic mediation rules between Rhapsody and BSO.

5.1.4.1.1 rhp:SysML2bso:System

Each rhp:SysML is translated to bso:System and back (one-to-one). There is only one rhp:SysML resource in a model and only one bso:System in a model at any given time.

Realized through OWL concept equivalent class (SysML≡System):

```
<rdf:Description rdf:about="http:// www.sprint-iot.eu/ontologies/bso/#System" >
        <equivalentClass rdf:resource="http://com.ibm.rhapsody/sprint/SysML"/>
</rdf:Description>
```

5.1.4.1.2 rhp:Package2bso:package

Each rhp:Package is translated to bso:package and back (one-to-one).

Realized through OWL concept equivalent class (Package≡Package):

```
<rdf:Description rdf:about="http:// www.sprint-iot.eu/ontologies/bso/#Package" >
        <equivalentClass rdf:resource="http://com.ibm.rhapsody/sprint/Package"/>
</rdf:Description>
```

5.1.4.1.3 rhp:hasPackage2bso:subpackages

Each [rhp:SysML rhp:hasPackage rhp:Package] is translated to [bso:System bso:subpackages bso:Package] relation and back (one-to-one).

Realized through OWL concept equivalent object property (hasPackage≡subpackages):

```
<rdf:Description rdf:about=" http:// www.sprint-iot.eu/ontologies/bso/#subpackages.DeclarationZone" >
        < equivalentProperty rdf:resource="http://com.ibm.rhapsody/sprint/hasPackage"/>
</rdf:Description>
```

5.1.4.1.4 rhpBlock2bso:Component

Each rhpBlock is translated to bso:Component and back (one-to-one).

Realized through OWL concept equivalent class (Block≡Component):

```
<rdf:Description rdf:about="http:// www.sprint-iot.eu/ontologies/bso/#Component" >
        <equivalentClass rdf:resource="http://com.ibm.rhapsody/sprint/Block"/>
</rdf:Description>
```

5.1.4.1.5 rhp:Object2bso:ComponentProperty

Each rhp:Object is translated to bso:ComponentProperty and back (one-to-one).

Realized through OWL concept equivalent class (Object≡ComponentProperty):

```
<rdf:Description rdf:about="http:// www.sprint-iot.eu/ontologies/bso/#ComponentProperty" >
        <equivalentClass rdf:resource="http://com.ibm.rhapsody/sprint/Object"/>
</rdf:Description>
```

5.1.4.1.6 rhp:hasType2bso:type

Each relation [rhp:Object rhp:hasType rhp:Block] is translated to [bso:ComponentProperty bso:type bso:Component] relation and back (one-to-one).

Realized through OWL concept equivalent object property (hasType≡type):

```
<rdf:Description rdf:about=" http:// www.sprint-iot.eu/ontologies/bso/#type" >
        < equivalentProperty rdf:resource="http://com.ibm.rhapsody/sprint/hasType"/>
</rdf:Description>
```

5.1.4.1.7 rhp:hasPart2bso:parts

Each relation relation [rhp:Block rhp:hasPart rhp:Object] is translated to [bso:Component bso:parts bso:ComponentProperty] relation and back (one-to-one).

Realized through OWL concept equivalent object property (hasPart≡parts):

```
<rdf:Description rdf:about=" http:// www.sprint-iot.eu/ontologies/bso/#parts.Component" >
        < equivalentProperty rdf:resource="http://com.ibm.rhapsody/sprint/hasPart"/>
</rdf:Description>
```

5.1.4.1.8 rhp:hasBlock2bso:declares

Each relation [rhp:Container rhp:hasBlock rhp:Block] is translated to [bso:Container bso:declares bso:Component] relation and back (one-to-one).

Realized through OWL concept equivalent object property (hasBlock≡declares):

```
<rdf:Description rdf:about=" http:// www.sprint-iot.eu/ontologies/bso/#parts.declares" >
        < equivalentProperty rdf:resource="http://com.ibm.rhapsody/sprint/hasBlock"/>
</rdf:Description>
```

5.1.4.1.9 rhp:hasPort2bso:ports

Each relation [rhp:Block rhp:hasPort rhp:StandardPort] is translated to [bso:Component bso:ports bso:Port] relation and back (one-to-one).

Realized through OWL concept equivalent object property (hasPort≡ports):

```
<rdf:Description rdf:about=" http:// www.sprint-iot.eu/ontologies/bso/#ports.Component" >
	< equivalentProperty rdf:resource="http://com.ibm.rhapsody/sprint/hasPort"/>
</rdf:Description>
```

5.1.4.1.10 rhp:Connector2bso:Connector

Each rhp:Connector is translated to bso:Connector and back (one-to-one).

Realized through OWL concept equivalent class (Connector≡Connector):

```
<rdf:Description rdf:about="http:// www.sprint-iot.eu/ontologies/bso/#Connector" >
	<equivalentClass rdf:resource="http://com.ibm.rhapsody/sprint/Connector"/>
</rdf:Description>
```

5.1.4.1.11 rhp:hasConnector2bso:connectors

Each relation [rhp:Block rhp:hasConnector rhp:Connector] is translated to [bso:Component bso:connectors bso:Connector] relation and back (one-to-one).

Realized through OWL concept equivalent object property (hasConnector≡ connectors):

```
<rdf:Description rdf:about=" http:// www.sprint-iot.eu/ontologies/bso/#connectors" >
	< equivalentProperty rdf:resource="http://com.ibm.rhapsody/sprint/hasConnector"/>
</rdf:Description>
```

5.1.4.1.12 dcTitle2bso:name

Each property wsm:name is translated to bso:name relation and back (one-to-one).

Realized through OWL concept equivalent property (name≡name):

```
<rdf:Description rdf:about=" http:// www.sprint-iot.eu/ontologies/bso/#name " >
	< equivalentProperty rdf:resource=" http://purl.org/dc/terms/title"/>
</rdf:Description>
```

5.1.4.2 BSO and Modelica

This section contains the semantic mediation rules between BSO and Modelica.

5.1.4.2.1 bso:Package2wsm:MPackage

Each wsm:MPackage is translated to bso:Package and back (one-to-one).

Realized through OWL concept equivalent class (MPackage≡Package):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#Package" >
        <equivalentClass rdf:resource="http://www.mathcore.com/sm-rdf/MPackage"/>
</rdf:Description>
```

5.1.4.2.2 bso:Connector2wsm:MConnection

Each wsm:MConnection is translated to bso:Connector and back (one-to-one).

Realized through OWL concept equivalent class (MConnection≡Connector):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#Connector " >
        <equivalentClass rdf:resource="http://www.mathcore.com/sm-rdf/MConnection"/>
</rdf:Description>
```

5.1.4.2.3 bso:Component 2wsm:MModel

Each wsm:MModel is translated to bso:Component and back (one-to-one).

Realized through OWL concept equivalent class (MModel≡Component):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#Component" >
        <equivalentClass rdf:resource="http://www.mathcore.com/sm-rdf/MModel"/>
</rdf:Description>
```

5.1.4.2.4 bso:ComponentProperty2wsm:MComponent

Each wsm:MComponent is translated to bso:ComponentProperty and back (one-to-one).

Realized through OWL concept equivalent class (MComponent≡ComponentProperty):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#ComponentProperty " >
        <equivalentClass rdf:resource="http://www.mathcore.com/sm-rdf/MComponent"/>
</rdf:Description>
```

5.1.4.2.5 bso:Connector2wsm:MConnection

Each wsm:MConnection is translated to bso:Connector and back (one-to-one).

Realized through OWL concept equivalent class (MConnection≡Connector):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#Connector" >
        <equivalentClass rdf:resource="http://www.mathcore.com/sm-rdf/MConnection"/>
</rdf:Description>
```

5.1.4.2.6 bso:Port2wsm:MConnectorComponent

Each wsm:MConnectorComponent is translated to bso:Port and back (one-to-one).

Realized through OWL concept equivalent class (MConnectorComponent≡Port):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#Port " >
        <equivalentClass rdf:resource="http://www.mathcore.com/sm-rdf/MConnectorComponent"/>
</rdf:Description>
```

5.1.4.2.7 bso:PortSpecification2wsm:MConnector

Each wsm:MConnector is translated to bso:PortSpecification and back (one-to-one).

Realized through OWL concept equivalent class (MConnector≡PortSpecification):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#PortSpecification" >
        <equivalentClass rdf:resource="http://www.mathcore.com/sm-rdf/MConnector"/>
</rdf:Description>
```

5.1.4.2.8 bso:Flow2wsm:MTypeComponent

Each wsm:MTypeComponent is translated to bso:Flow and back (one-to-one).

Realized through OWL concept equivalent class (MTypeComponent≡Flow):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#Flow " >
        <equivalentClass rdf:resource="http://www.mathcore.com/sm-rdf/MTypeComponent"/>
</rdf:Description>
```

5.1.4.2.9 bso:name2wsm:name

Each property wsm:name is translated to bso:name relation and back (one-to-one).

Realized through OWL concept equivalent property (name≡name):

```
<rdf:Description rdf:about=" http:// www.sprint-iot.eu/ontologies/bso/#name" >
        < equivalentProperty rdf:resource=" http://www.mathcore.com/sm-rdf/name"/>
</rdf:Description>
```

5.1.4.2.10 bso:comment2 wsm:description

Each property wsm:description is translated to bso:comment relation and back (one-to-one).

Realized through OWL concept equivalent property (description≡comment):

```
<rdf:Description rdf:about="http:// www.sprint-iot.eu/ontologies/bso/#comment" >
        < equivalentProperty rdf:resource="http://www.mathcore.com/sm-rdf/description"/>
</rdf:Description>
```

5.1.4.2.11 rhp:hasPort2bso:ports

Each relation [wsm:MModel wsm:connection wsm:MConnection] is translated to [bso:Component bso:connectors bso:Connector] relation and back (one-to-one).

Realized through OWL concept equivalent object property (connection ≡connectors):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#connectors " >
        < equivalentProperty rdf:resource="http://www.mathcore.com/sm-rdf/connection"/>
</rdf:Description>
```

5.1.4.2.12 rhp:parts2bso:parts

Each relation [wsm:MModel wsm:parts wsm:MComponent] is translated to [bso:Component bso:parts bso:ComponentProperty] relation and back (one-to-one).

Realized through OWL concept equivalent object property (parts ≡ parts):

```
<rdf:Description rdf:about="http://www.sprint-iot.eu/ontologies/bso/#parts" >
        < equivalentProperty rdf:resource="http://www.mathcore.com/sm-rdf/parts"/>
</rdf:Description>
```

6 Interaction with other SPRINT platform components

In order to do so the tools have to have the capability to exchange information (fetch and provide data). This data has to conform in its representation to the standards used in the Sprint engineering environment which are RDF (represented in Turtle or XML, XML preferred). In order to represent the model or instance data in RDF it is necessary to provide a respective RDF/RDFS vocabulary. However, all this is not part of the semantic mediation but a precondition and assumed given. These preconditions are fulfilled by the SSI in which the semantic mediation is been embedded. Figure 6-1: SII container and the interception framework [D5.2] shows a schemata on the integration of SSI indo SII and its components. Amore detaild desctiption on the integration is been given in D5.2 High and low level design description of System Design and Semantic Interoperable Integration.

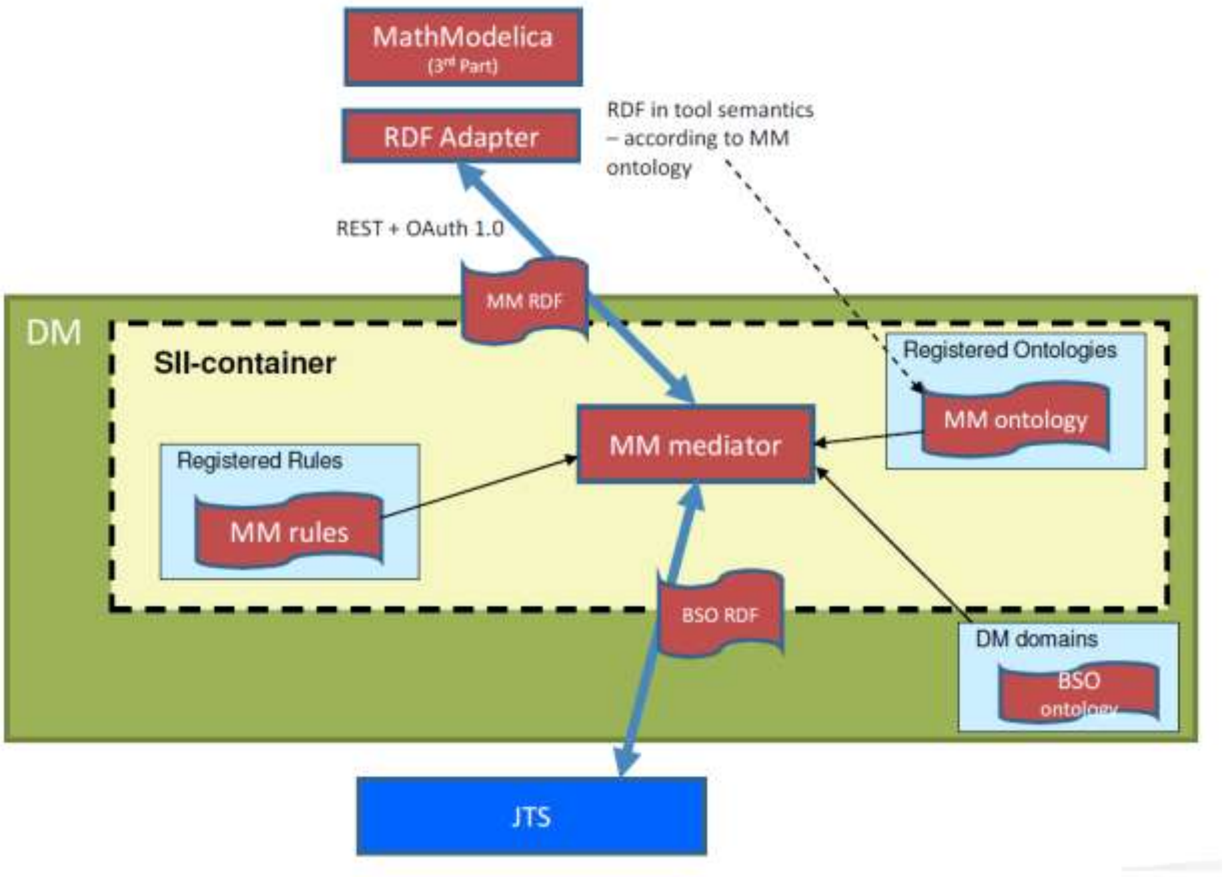


Figure 6-1: SII container and the interception framework [D5.2]

The SSI container will embeds the semantic mediation via an interceptor pattern forwarding the prepared RESTful post and get requests. The SSI container also manages the domain vocabularies and its dependencies as well as the registered rules. This information is then context sensitive provided to the semantic mediation for mediation execution.

7 Experience

This section describes the experiences made during implementation and experimentation.

7.1 Modelling Tools

7.1.1 Protégé 3

This tool is mostly focused on RDF/RDFS and triples. The OWL support is not to be compared with Protégé 4 and it should not really be used for OWL or modelling semantic mediations less than for low level definitions and debugging.

7.1.2 Protégé 4

This tool is very good for OWL definition and management. We encourage the usage of this tool for semantic mediation definition. However, it is not really possible to edit on the level of triplets. Everything is elevated to the level OWL and its definitions. Unfortunately, it seems that there is no tool to support both aspects in a good way. Protégé 4 supports all necessary formats as required by the defined semantic mediation solution. The relevant inference engines are supported in the tool.

7.1.3 NeOn Toolkit

This tool although good integrated into eclipse seems still a little bit immature. Inference does not work as expected.

7.2 Frameworks

7.2.1 Jena

Jena has the advantage of combining low-level RDF triplet API with high-level OWL API. However, it tends to get very fast slow depending on the operations performed and models created[10]. For the better handling of the ontologies one should always try to create absolute URIs:

```
String xmlbase = "http://www.sprint-iot.eu/bso/";
OntModel m = ModelFactory.createOntologyModel(ProfileRegistry.OWL_LANG);
RDFWriter rdfw=m.getWriter("RDF/XML");
rdfw.setProperty("xmlbase", xmlbase);
rdfw.setProperty("relativeURIs", "");
...
FileOutputStream owlFile = new FileOutputStream("test.owl");
m.expandPrefix(xmlbase);
m.write(camera_File, "RDF/XML-ABBREV", "");
```

7.2.2 OWLAPI

This API does not have low-level support for triplets as Jena. This makes it hard to work on the low level which is sometimes necessary, especially when creating the result of the semantic mediation. However, the high-level support for OWL is better than in Jena. The relevant inference engines are supported.

7.3 Languages

[10] Even on in-memory models

7.3.1 SWRL

This language seems to be outdated and no longer maintained[11]. Henceforth has not been chosen for the semantic mediation.

[11] Especially in comparsion to RuleML

8 Abbreviations and Definitions

Architecture	The fundamental organization of a system (maybe on different abstraction levels) embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
Artefact	A physical piece of information that is used or produced by a software development process. Examples of Artefacts include models, source files, scripts, and binary executable files.
Attribute	A piece of information associated with an entity. Used to describe a characteristic of an entity. In the meta-model, an attribute is a placeholder for the actual information that is defined in the models.
Domain-specific Language	Domain-specific language (DSL) is a programming language or specification language dedicated to a particular problem domain.
DSL	Domain-specific Language
Closed World Assumption	The closed world assumption is the presumption that what is not currently known to be true, is false. The opposite of the closed world assumption is the open world assumption. (See: http://en.wikipedia.org/wiki/Closed_world_assumption)
CWA	See: Closed World Assumption
Eclipse Modeling Framework	Is a modeling framework and code generation facility for building tools and other applications based on a structured data model. (See: http://www.eclipse.org/emf/)
eCore	Pendant to EMOF in EMF.
EMF	See: Eclipse Modeling Framework
Entity	An object with an identity that is distinguishable from other entities.
Error	Discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition.
Extensible Markup Language	Is a set of rules for encoding documents in machine-readable form defined by W3C. (See: http://www.w3.org/TR/2008/REC-xml-20081126/)
Instance	An entity that is obtained from another entity by the process of instantiation.
Interface	Abstraction of a service that only defines the operations supported by that service (publicly accessible variables, procedures, or methods), but not their implementation.
Link	Representation of a relation between two objects.
Meta-Object Facility	Is an OMG standard for model-driven engineering. (See: http://www.omg.org/mof/)
Meta-Object Facility 2	(See: www.omg.org/spec/MOF/2.0)
Metamodel	A model for describing (structure of) other models on a meta level.
MathModelica	See: SystemModeler
Model	A semantically closed abstraction of a software or hardware system, i.e. a simplification of reality that gives a complete description a system from a particular perspective.

MOF	See: Meta-Object Facility
MOF2	See: Meta-Object Facility 2
MOF Support For Semantic Structures	Is a standardized extension to MOF by the OMG which modifies MOF 2 in order to support dynamically mutable multiple classifications of elements and to declare the circumstances under which such multiple classifications are allowed, required and prohibited. (See: http://www.omg.org/spec/SMOF/)
Object Constraint Language	Is a declarative language for describing rules that apply to MOF compliant metamodels. (See: www.omg.org/spec/OCL/)
Object Management Group	Is a standardization organization in the modelling area (e.g. programs, systems and business processes). (See: http://www.omg.org)
OCL	See: Object Constraint Language
OMG	See: Object Management Group
Open World Assumption	In formal logic, the open world assumption is the assumption that the truth-value of a statement is independent of whether or not it is known by any single observer or agent to be true. Stating that lack of knowledge does not imply falsity. It is the opposite of the closed world assumption. (See: http://en.wikipedia.org/wiki/Open_world_assumption)
Open Services for Lifecycle Collaboration	Open Services for Lifecycle Collaboration (also known as OSLC or Open Services) is a community and set of specifications for Linked Lifecycle Data. The community's goal is to help product and software delivery teams by making it easier to use lifecycle tools in combination. (See: http://open-services.net/html/Home.html)
OSLC	See: Open Services for Lifecycle Collaboration
OWA	See: Open World Assumption
OWL	See: Web Ontology Language
Rhapsody	Engineering and modelling tool from IBM.
RDF/RDFS	See: Resource Description Framework
Resource Description Framework	It's a family of W3C specifications for conceptual description or modelling of information that is implemented in web resources. (See: http://www.w3.org/TR/rdf-primer/)
SMOF	See: MOF Support For Semantic Structures
SPARQL	SPARQL (SPARQL Protocol And RDF Query Language) is a query language for RDF. http://www.w3.org/TR/rdf-sparql-query/
System	A collection of components organized to accomplish a specific function or set of functions.
SystemModeler	Is a Modelica implementation of Wolfram formerly known as MathModelica.
Traceability	Traceability is a technique used to provide relationships between objects (e.g. in requirements, design and implementation of a system in order to manage the effect of change).
Type system	A tractable syntactic framework for classifying phrases according to the kinds of values they compute. See: http://en.wikipedia.org/wiki/Type_system
UML	See: Unified Modeling Language
UML Profile	A profile in the Unified Modeling Language provides a generic

	extension mechanism for building UML models in particular domains. Profiles are based on additional stereotypes and tagged values that are applied to elements, attributes, methods, links, and link ends. A profile is a collection of such extensions and restrictions that together describe some particular modeling problem and facilitate modeling constructs in that domain.
Unified Modeling Language	From Grady Booch, Ivar Jacobson and Jim Rumbaugh (Co. Rational software) developed, and by the Object Management Group (OMG) confirmed abstract description language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for other non-software systems.
View	A representation of a whole system from the perspective of a related set of concerns.
W3C	See: World Wide Web Consortium
Web Ontology Language	Is a family of knowledge representation languages defined by W3C for authoring ontologies. (See: http://www.w3.org/TR/owl-ref/)
World Wide Web Consortium	Is the main international standardization organization for the World Wide Web. (See: http://www.w3.org)
XML	See: Extensible Markup Language
XML Schema	Language for XML schema description which has Recommendation status by the W3C. (See: http://www.w3.org/TR/xmlschema-1/, http://www.w3.org/TR/xmlschema-2/)
XSD	See: XML Schema

9 References

[MOF, 2006]	Object Management Group: *Meta Object Facility (MOF) Core Specification*, Version 2.0 (OMG Document formal/06-01-01), January 2006. 12
[OWL, 2009]	W3C: *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax,* Recommendation 27 October 2009, http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/
[QVT, 2005]	Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Final Adopted Specification (OMG Document ptc/05-11-01), November 2005.
[D3.1]	Michael Wagner; Fraunhofer FOKUS; *D3.1 SSI Foundations and Definitions*
[D5.2]	Uri Shani; IBM; D5.2 *High and low level design description of System Design and Semantic Interoperable Integration*
[D5.4]	Aviad Sela, Uri Shani; *D5.4 Architectural principles for Internet of System Design and the IoT*

10 Appendix

Appendix I: Complete Modelica Example Model in Turtle

```
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MPackage> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.mathcore.com/sm-rdf/identifier>
"BasicImportExportTestWithPorts" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.mathcore.com/sm-rdf/localClass>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.mathcore.com/sm-rdf/localClass>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.mathcore.com/sm-rdf/localClass>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.mathcore.com/sm-rdf/localClass>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.mathcore.com/sm-rdf/localClass>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.mathcore.com/sm-rdf/localClass>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts>   <http://www.mathcore.com/sm-rdf/localClass>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput>   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MConnector> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput>   <http://www.mathcore.com/sm-rdf/identifier>
"IntegerInput" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput>   <http://www.mathcore.com/sm-rdf/within>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput>   <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput#signal> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput>   <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type>   <http://www.mathcore.com/sm-rdf/MConnector> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput>   <http://www.mathcore.com/sm-rdf/identifier>
"IntegerOutput" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput>   <http://www.mathcore.com/sm-rdf/within>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput>   <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput#signal> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MModel> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>   <http://www.mathcore.com/sm-rdf/identifier>   "M" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>   <http://www.mathcore.com/sm-rdf/within>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>   <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k1> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>   <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k2> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>   <http://www.mathcore.com/sm-rdf/connection>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-1> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M>   <http://www.mathcore.com/sm-rdf/connection>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-2> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1>   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MBlock> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1>   <http://www.mathcore.com/sm-rdf/identifier>   "M1" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1>   <http://www.mathcore.com/sm-rdf/within>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1>   <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1a> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1>   <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1b> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2>   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MBlock> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2>   <http://www.mathcore.com/sm-rdf/identifier>   "M2" .
```

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2> <http://www.mathcore.com/sm-rdf/within>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2> <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2a> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2> <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2b> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MConnector> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput> <http://www.mathcore.com/sm-rdf/identifier>
"RealInput" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput> <http://www.mathcore.com/sm-rdf/within>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput> <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput#signal> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MConnector> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput> <http://www.mathcore.com/sm-rdf/identifier>
"RealOutput" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput> <http://www.mathcore.com/sm-rdf/within>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput> <http://www.mathcore.com/sm-rdf/part>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput#signal> .

<http://www.mathcore.com/sm-rdf/Integer> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.mathcore.com/sm-
rdf/MIntegerType> .

<http://www.mathcore.com/sm-rdf/Integer> <http://www.mathcore.com/sm-rdf/identifier> "Integer" .

<http://www.mathcore.com/sm-rdf/Real> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.mathcore.com/sm-
rdf/MRealType> .

<http://www.mathcore.com/sm-rdf/Real> <http://www.mathcore.com/sm-rdf/identifier> "Real" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput#signal> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://www.mathcore.com/sm-rdf/MTypeComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput#signal> <http://www.mathcore.com/sm-rdf/identifier>
"signal" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput#signal> <http://www.mathcore.com/sm-rdf/type>
<http://www.mathcore.com/sm-rdf/Integer> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput#signal> <http://www.mathcore.com/sm-rdf/causality>
"INPUT" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput#signal> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://www.mathcore.com/sm-rdf/MTypeComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput#signal> <http://www.mathcore.com/sm-
rdf/identifier> "signal" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput#signal> <http://www.mathcore.com/sm-rdf/type>
<http://www.mathcore.com/sm-rdf/Integer> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput#signal> <http://www.mathcore.com/sm-
rdf/causality> "OUTPUT" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k1> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k1> <http://www.mathcore.com/sm-rdf/identifier> "k1" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k1> <http://www.mathcore.com/sm-rdf/type>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k2> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k2> <http://www.mathcore.com/sm-rdf/identifier> "k2" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k2> <http://www.mathcore.com/sm-rdf/type>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1a> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MConnectorComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1a> <http://www.mathcore.com/sm-rdf/identifier> "p1a" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1a> <http://www.mathcore.com/sm-rdf/type>
<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerOutput> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1b> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.mathcore.com/sm-rdf/MConnectorComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1b> <http://www.mathcore.com/sm-rdf/identifier> "p1b" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1b> <http://www.mathcore.com/sm-rdf/type> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2a> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.mathcore.com/sm-rdf/MConnectorComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2a> <http://www.mathcore.com/sm-rdf/identifier> "p2a" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2a> <http://www.mathcore.com/sm-rdf/type> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.IntegerInput> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2b> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.mathcore.com/sm-rdf/MConnectorComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2b> <http://www.mathcore.com/sm-rdf/identifier> "p2b" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2b> <http://www.mathcore.com/sm-rdf/type> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput#signal> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.mathcore.com/sm-rdf/MTypeComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput#signal> <http://www.mathcore.com/sm-rdf/identifier> "signal" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput#signal> <http://www.mathcore.com/sm-rdf/type> <http://www.mathcore.com/sm-rdf/Real> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealInput#signal> <http://www.mathcore.com/sm-rdf/causality> "INPUT" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput#signal> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.mathcore.com/sm-rdf/MTypeComponent> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput#signal> <http://www.mathcore.com/sm-rdf/identifier> "signal" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput#signal> <http://www.mathcore.com/sm-rdf/type> <http://www.mathcore.com/sm-rdf/Real> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.RealOutput#signal> <http://www.mathcore.com/sm-rdf/causality> "OUTPUT" .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-1> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.mathcore.com/sm-rdf/MConnection> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-1> <http://www.mathcore.com/sm-rdf/fromPart> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k2> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-1> <http://www.mathcore.com/sm-rdf/fromPort> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2b> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-1> <http://www.mathcore.com/sm-rdf/toPart> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k1> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-1> <http://www.mathcore.com/sm-rdf/toPort> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1b> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-2> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.mathcore.com/sm-rdf/MConnection> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-2> <http://www.mathcore.com/sm-rdf/fromPart> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k1> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-2> <http://www.mathcore.com/sm-rdf/fromPort> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M1#p1a> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-2> <http://www.mathcore.com/sm-rdf/toPart> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#k2> .

<http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M#connection-2> <http://www.mathcore.com/sm-rdf/toPort> <http://www.mathcore.com/sm-rdf/BasicImportExportTestWithPorts.M2#p2a> .

Appendix II: Complete Rhapsody Model in RDF

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rhp_model="http://com.ibm.rhapsody/sprint/"
xmlns:dc="http://purl.org/dc/terms/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-0208a299-1530-4e1f-aa76-03c71b1c38f2"
  dc:title="M"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Block"/>
  <rhp_model:hasPart rdf:resource="http://com.ibm.rhapsody/sprint/GUID-1a2f45d0-5280-4d84-8eaf-4303f5a25158"/>
  <rhp_model:hasPart rdf:resource="http://com.ibm.rhapsody/sprint/GUID-c47c5f00-a0a2-4985-a7d6-881e676b6a81"/>
  <rhp_model:hasBlock rdf:resource="http://com.ibm.rhapsody/sprint/GUID-bd49b29d-1448-45dc-9b31-d2f98d56c43a"/>
  <rhp_model:hasBlock rdf:resource="http://com.ibm.rhapsody/sprint/GUID-fff2e05a-168f-4246-9a76-52b14647f403"/>
  </rdf:Description>
```

```
<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-d64c4fbb-c594-4071-acbc-277bc06d1019"
  dc:title="BasicImportExportTestWithPorts"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/SysML"/>
  <rhp_model:hasPackage rdf:resource="http://com.ibm.rhapsody/sprint/GUID-66341366-c102-4a08-a2b4-e2f74f8767cb"/>
  <rhp_model:hasPackage rdf:resource="http://com.ibm.rhapsody/sprint/OLDID-3-1"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-66341366-c102-4a08-a2b4-e2f74f8767cb"
  dc:title="Default"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Package"/>
  <rhp_model:hasBlock rdf:resource="http://com.ibm.rhapsody/sprint/GUID-0208a299-1530-4e1f-aa76-03c71b1c38f2"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-1a2f45d0-5280-4d84-8eaf-4303f5a25158"
  dc:title="k1"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Object"/>
  <rhp_model:hasType rdf:resource="http://com.ibm.rhapsody/sprint/GUID-fff2e05a-168f-4246-9a76-52b14647f403"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-c47c5f00-a0a2-4985-a7d6-881e676b6a81"
  dc:title="k2"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Object"/>
  <rhp_model:hasType rdf:resource="http://com.ibm.rhapsody/sprint/GUID-bd49b29d-1448-45dc-9b31-d2f98d56c43a"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-fff2e05a-168f-4246-9a76-52b14647f403"
  dc:title="M1"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Block"/>
  <rhp_model:hasFlow rdf:resource="http://com.ibm.rhapsody/sprint/GUID-b10f270b-d2b3-462b-a339-5dc597c7683d"/>
  <rhp_model:hasFlow rdf:resource="http://com.ibm.rhapsody/sprint/GUID-1ebf7201-d4a9-4644-a46a-99bc7d398fc0"/>
  <rhp_model:hasFlowPort rdf:resource="http://com.ibm.rhapsody/sprint/GUID-437271cd-5f5c-4325-bf65-552f7bc349da"/>
  <rhp_model:hasFlowPort rdf:resource="http://com.ibm.rhapsody/sprint/GUID-16b7ddaa-6d51-434d-8566-564628902614"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-bd49b29d-1448-45dc-9b31-d2f98d56c43a"
  dc:title="M2"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Block"/>
  <rhp_model:hasFlow rdf:resource="http://com.ibm.rhapsody/sprint/GUID-36408c3e-e12e-4f8d-b158-ec393fad68a5"/>
  <rhp_model:hasFlow rdf:resource="http://com.ibm.rhapsody/sprint/GUID-c5cd4596-cd46-407e-9561-7e9a8f581c77"/>
  <rhp_model:hasFlowPort rdf:resource="http://com.ibm.rhapsody/sprint/GUID-7bb232fc-44b6-48c4-bf9e-f23b357de119"/>
  <rhp_model:hasFlowPort rdf:resource="http://com.ibm.rhapsody/sprint/GUID-0b75a23a-14dd-4c23-8f43-a1ceb6924bf6"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-16b7ddaa-6d51-434d-8566-564628902614"
  dc:title="p1a"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/FlowPort"/>
  <rhp_model:isReversed rdf:datatype="http://www.w3.org/2001/XMLSchema#xsd:boolean">false</rhp_model:isReversed>
  <rhp_model:hasMultiplicity rdf:resource="http://com.ibm.rhapsody/sprint/ExactlyOneMultiplicity"/>
  <rhp_model:hasType rdf:resource="http://com.ibm.rhapsody/sprint/GUID-6024c6ab-fe9e-402e-954f-27aca6d3b5e4"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-437271cd-5f5c-4325-bf65-552f7bc349da"
  dc:title="p1b"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/FlowPort"/>
  <rhp_model:isReversed rdf:datatype="http://www.w3.org/2001/XMLSchema#xsd:boolean">false</rhp_model:isReversed>
  <rhp_model:hasMultiplicity rdf:resource="http://com.ibm.rhapsody/sprint/ExactlyOneMultiplicity"/>
  <rhp_model:hasType rdf:resource="http://com.ibm.rhapsody/sprint/GUID-02ef7294-6849-47b9-9fb2-cfb87b56d2ad"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-b10f270b-d2b3-462b-a339-5dc597c7683d"
  dc:title="p1a_p2a"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Flow"/>
  <rhp_model:hasPort_1 rdf:resource="http://com.ibm.rhapsody/sprint/GUID-16b7ddaa-6d51-434d-8566-564628902614"/>
  <rhp_model:hasPort_2 rdf:resource="http://com.ibm.rhapsody/sprint/GUID-7bb232fc-44b6-48c4-bf9e-f23b357de119"/>
  <rhp_model:hasFlowDirection>
```

```
    rhp_model:toEnd2Direction
  </rhp_model:hasFlowDirection>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-1ebf7201-d4a9-4644-a46a-99bc7d398fc0"
  dc:title="p1a_p2a_0"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Flow"/>
  <rhp_model:hasPort_1 rdf:resource="http://com.ibm.rhapsody/sprint/GUID-16b7ddaa-6d51-434d-8566-564628902614"/>
  <rhp_model:hasPort_2 rdf:resource="http://com.ibm.rhapsody/sprint/GUID-7bb232fc-44b6-48c4-bf9e-f23b357de119"/>
  <rhp_model:hasFlowDirection>
    rhp_model:toEnd2Direction
  </rhp_model:hasFlowDirection>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-7bb232fc-44b6-48c4-bf9e-f23b357de119"
  dc:title="p2a"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/FlowPort"/>
  <rhp_model:isReversed rdf:datatype="http://www.w3.org/2001/XMLSchema#xsd:boolean">false</rhp_model:isReversed>
  <rhp_model:hasMultiplicity rdf:resource="http://com.ibm.rhapsody/sprint/ExactlyOneMultiplicity"/>
  <rhp_model:hasType rdf:resource="http://com.ibm.rhapsody/sprint/GUID-6024c6ab-fe9e-402e-954f-27aca6d3b5e4"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-0b75a23a-14dd-4c23-8f43-a1ceb6924bf6"
  dc:title="p2b"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/FlowPort"/>
  <rhp_model:isReversed rdf:datatype="http://www.w3.org/2001/XMLSchema#xsd:boolean">false</rhp_model:isReversed>
  <rhp_model:hasMultiplicity rdf:resource="http://com.ibm.rhapsody/sprint/ExactlyOneMultiplicity"/>
  <rhp_model:hasType rdf:resource="http://com.ibm.rhapsody/sprint/GUID-02ef7294-6849-47b9-9fb2-cfb87b56d2ad"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-c5cd4596-cd46-407e-9561-7e9a8f581c77"
  dc:title="p2b_p1b"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Flow"/>
  <rhp_model:hasPort_1 rdf:resource="http://com.ibm.rhapsody/sprint/GUID-0b75a23a-14dd-4c23-8f43-a1ceb6924bf6"/>
  <rhp_model:hasPort_2 rdf:resource="http://com.ibm.rhapsody/sprint/GUID-437271cd-5f5c-4325-bf65-552f7bc349da"/>
  <rhp_model:hasFlowDirection>
    rhp_model:toEnd2Direction
  </rhp_model:hasFlowDirection>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-36408c3e-e12e-4f8d-b158-ec393fad68a5"
  dc:title="p2b_p1b_0"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Flow"/>
  <rhp_model:hasPort_1 rdf:resource="http://com.ibm.rhapsody/sprint/GUID-0b75a23a-14dd-4c23-8f43-a1ceb6924bf6"/>
  <rhp_model:hasPort_2 rdf:resource="http://com.ibm.rhapsody/sprint/GUID-437271cd-5f5c-4325-bf65-552f7bc349da"/>
  <rhp_model:hasFlowDirection>
    rhp_model:toEnd2Direction
  </rhp_model:hasFlowDirection>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-6024c6ab-fe9e-402e-954f-27aca6d3b5e4"
  dc:title="int"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Type"/>
  <dc:description>
    Predefined int
  </dc:description>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/OLDID-3-1"
  dc:title="PredefinedTypesCpp"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Package"/>
  <rhp_model:hasOTHER rdf:resource="http://com.ibm.rhapsody/sprint/GUID-6024c6ab-fe9e-402e-954f-27aca6d3b5e4"/>
  <rhp_model:hasOTHER rdf:resource="http://com.ibm.rhapsody/sprint/GUID-02ef7294-6849-47b9-9fb2-cfb87b56d2ad"/>
  </rdf:Description>

<rdf:Description rdf:about="http://com.ibm.rhapsody/sprint/GUID-02ef7294-6849-47b9-9fb2-cfb87b56d2ad"
  dc:title="double"
  >
  <rdf:type rdf:resource="http://com.ibm.rhapsody/sprint/Type"/>
  <dc:description>
    Predefined double
  </dc:description>
  </rdf:Description>

</rdf:RDF>
```