Seventh Framework Programme

CallFP7-ICT-2013-10

Project Acronym:	**S-CASE**
Grant Agreement N^{o}:	**610717**
Project Type:	**COLLABORATIVE PROJECT**
Project Full Title:	**Scaffolding Scalable Software Services**

D2.2 Definition of Platform Independent and Platform Specific Service Models

Nature:	**R**
Dissemination Level:	**PU**
Version #:	**1.0**
Date:	**30 January 2015**
WP number and Title:	**WP 2 Automated model-driven mapping and transformation**
Deliverable Leader:	**AUTH**
Author(s):	Christoforos Zolotas (AUTH), Kyriakos Chatzidimitriou (AUTH)
Reviewer(s):	Isabel Matranga (ENG), Ciro Formisano (ENG), Robert Magnus (ATS)
Status:	**Submitted** (Draft, Peer-Reviewed, Submitted, Approved)

Document History

Version[1]	Issue Date	Status[2]	Content and changes
0.1	8 Dec 2014	Draft	Initial Template Document
0.2	10 Dec 2014	Draft	Completed Related Work section
0.3	11 Dec 2014	Draft	Completed Introduction to S-CASE MDE Engine section
0.4	18 Dec 2014	Draft	Completed the Definition of the Ecore PIM Meta-model
0.5	24 Dec 2014	Draft	Completed the Definition of the Ecore PSM Meta-model
0.6	30 Dec 2014	Draft	Completed the Definition of the ATL PIM to PSM Transformation
0.7	6 Jan 2015	Draft	Added figures to sections 4 and 5
0.8	14 Jan 2015	Draft	Revised document content
0.9	15 Jan 2015	Draft	Completed Indroduction section and sent document for Peer-Review
0.95	27 Jan 2015	Peer-Reviewed	Received Peer-Review feedback and took any suggested actions
1.0	30 Jan 2015	Submitted	Final submitted version

Peer Review History[3]

Version	Peer Review Date	Reviewed By
0.91	26 January 2015	Robert Magnus (ATS)
0.92	27 January 2015	Isabel Matranga and Ciro Formisano (ENG)

This project has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under grant agreement no 610717.

[1] Please use a new number for each new version of the deliverable. Use “0.#” for Draft and Peer-Reviewed. “x.#” for Submitted and Approved”, where x>=1.Add the date when this version was issued and list the items that have been added or changed.

[2] A deliverable can be in one of these stages: Draft, Peer-Reviewed, Submitted and Approved.

[3] Only for deliverables that have to be peer-reviewed

Table of contents

List of Figures

List of Tables

Abbreviations and Acronyms

ATL: Atlas Transformation Language (a declarative M2M transformation language)

CIM: Computational Independent Model

EMF: Eclipse Modelling Framework

M2M: Model to Model transformation

MDE: Model Driven Engineering

MOF: Meta-Object Facility

MVC: Model View Controller design pattern

OCL: Object Constraint Language

OMG: Object Management Group

PIM: Platform Independent Model

PSM: Platform Specific Model

QVT: Query View Transformation language

REST: Representational State Transfer

RMM: Richardson Maturity Model

URI: Uniform Resource Identifier

XMI: XML Metadata Interchange

XML: Extensible Markup Language

Executive Summary

The overall objective of WP2 is to provide upper and lower CASE-related functionalities to the S-CASE system in order to allow the successful, automated handling of software-artefact meta-information, as well as its transformation to a RESTful web service upon developer request.

REST architectural style has specific features a web service must present in order to be considered RESTful. Richardson's Maturity Model (RMM) captures these principal features as the decomposition of web services to simple resources that are addressed via a unique URI, the proper use of the HTTP verbs by respecting their semantics as well as the semantic interconnection of a web service's resources, known as the *hypermedia.*

Although there are numerous frameworks in the domain of RESTful development, most of them do not satisfy all of the RMM's criteria and therefore produce 2nd level RMM services. This is because, either most of them do no embed web service resources interconnection at all, or in the few cases that some *hypermedia* are introduced, are of limited use or require developer intervention.

In this context, this document presents advances and achievements within WP2 with respect to the state of the art, which are:

- The full incorporation of the REST architectural style to the S-CASE produced web services, including automated semantic interweaving of their resources with hypermedia links.
- The application of the Model Driven Engineering paradigm to RESTful services development, which provides a comprehensive mechanism that is extensible so as to support multiple platforms instead of a specific one (as most other frameworks do).
- Transparent transformations within the MDE phases by using the declarative programming paradigm with ATL, which leads to increased visibility and traceability of the whole process.
- A list of fully formalized OCL constraints that provides validation capabilities for every S-CASE produced web service, in terms of compliance to the REST architectural style as well as their structural and behavioural consistency in all the intermediary stages.

The PIM meta-model, which this document introduces, embeds a slight variation of the well-established *Model View Controller* design paradigm in order create an abstract resource model. The *Model* part of each such resource represents its data. The *Controller,* its web API that respects the HTTP verb semantics as REST requires. The *Views* or preferably *Representations* represent the allowed media types for client-server interactions. Moreover, every produced service embeds the *Repository* design pattern in order to offer uniform access to its database data for the rest of the automatically produced system as well as a high-level storage handle for any custom addendum by S-CASE developers.

The PSM meta-model, which specifies the abstract PIM design of a web service, embeds several widespread web frameworks, such as *JAXB* for JSON/XML to object mapping, *JPA* for the implementation of the service's relational schema and *JAX-RS* to expose its RESTful API. Finally, it goes beyond simple project structure scaffolding by producing any build/configuration files needed as well.

1 Introduction

1.1 WP2 Description and Objectives

The overall objective of WP2 is to provide upper and lower CASE-related functionality to the S-CASE system in order to allow the successful storing and querying of meta-information for software artefacts, as well as the transformation of this meta-information to RESTful web services upon developer request. The most important objectives of WP2 as a whole are to:

1. Define the structure and constraints of the Platform Independent Model (PIM) following the MDE paradigm so as to be able to create the abstract design of envisioned RESTful services that satisfy the multi-modal user requirements.
2. Define the structure and constraints of the Platform Specific Model (PSM) based on the MDE paradigm so as to be able to specify the abstract PIM design to a concrete set of technologies that will implement the desired functionality.
3. Design the interfaces to the S-CASE ontology in order to support automated storing and retrieval of software artefacts to be used for querying, matching and mining tasks.
4. Develop an automated CIM to PIM Model-to-Model (M2M) transformation mechanism.
5. Develop an automated PIM to PSM M2M transformation mechanism.
6. Develop mechanisms for transforming 3rd party services into models and into S-CASE services.
7. Design and develop mining mechanisms for discovering associations between models.

This deliverable is primarily concerned with the objectives of Task 2.2. These are objectives 1, 2 and 5 from the above list. Specifically, the goal of this task is to define the (formal) scheme for generating the abstract S-CASE Platform Independent Model, as well as the corresponding specialized Platform Specific one. The S-CASE PIM and PSM will serve as **a single point of reference for mapping requirements to functionality, denoting associations between related software artefacts, and describing web service architecture details** within the S-CASE framework. The PIM will be designed in order to **foster the incorporation of all needed information** as described by the S-CASE ontology and as imposed by the REST architectural style, while the PSM design will fully specify the PIM design with **a set of concrete widespread technologies and standards**, in order to lead **to automated code generation**.

Regarding the formal scheme of the PIM and PSM models, this deliverable introduces **two Ecore meta-models that enable the interweaving of parsed software artefacts so as to produce a RESTful web service and its data storage**. The outcome of this task will provide the basis for building the transformation mechanism of the S-CASE ontology to MOF models (T.2.3).

1.2 Deliverable Structure

This deliverable is divided into six main sections. The first three lay the ground regarding terminology and context that is used in the 4th, 5th and 6th section where the PIM, PSM and the transformation between them are defined, respectively.

More specifically, section 1 discusses the general goals of WP2 of the S-CASE project as well as the more specific goals related to Task 2.2 and this deliverable. Then, it identifies the intended audience of this document and the reading guidelines to various stakeholder groups. Section 2 presents state-of-the art frameworks for RESTful web services development and discusses the way the S-CASE MDE engine is related to them, as well as how our work advances beyond the state of the art. Next, section 3 completes the introductory part by presenting the MDE paradigm as applied by S-CASE. Additionally, it provides a brief outline of the various frameworks used in PIM, PSM and PIM to PSM transformation definitions.

The second part of this document, sections 4, 5 and 6, delve deep into the definition of the Task 2.2 related artefacts. Therefore, section 4 fully defines the PIM meta-model both structurally and behaviourally, taking into account the design goals and restrictions imposed by the S-CASE project. Section 5 introduces the definition of a PSM meta-model that is compatible to the PIM one. This is also defined structurally and behaviourally. Finally, section 6 describes conceptually the definition of the way the transformation of PIM meta-model instances to PSM meta-model ones is performed.

The document concludes with a list of references and several appendices that point to the actual implementations of the PIM and PSM meta-model definitions introduced in sections 4 and 5, as well as the full transformation code following the ATL declarative paradigm.

1.3 Target Audience

This document is considerably technical and therefore it primarily targets stakeholders interested in understanding the way S-CASE automates the development of RESTful web services or even extending it.

Readers who wish to gain an overview of the goals related to WP2 of the S-CASE project should mainly focus on the introductory sections 1, 2 and 3, which discuss the overall state of the art on RESTful services development and an overview of the way S-CASE attempts to automate it.

Readers who wish to understand the PIM and PSM meta-models should focus on Sections 4 and 5 that syntactically and semantically define all the aspects of these two meta-models. Moreover, anyone who wishes to define and implement a new PSM meta-model in order to expand the pool of the S-CASE target execution platforms, and thus its user groups, should focus on understanding the PIM meta-model definition. The full Ecore/OCL definition of the PIM and PSM meta-models, as well as the full ATL transformation definition found in appendix should be of great help for anyone in this direction.

2 Related work

This section provides an overview of the related projects and frameworks for RESTful services development. Initially it provides a short description of the REST architectural style so as to lay the minimum background in respect to which other tools and S-CASE's MDE engine should be compared. The last subsection provides an overview of the way S-CASE relates to these frameworks and of the relevant advances made against the state of the art regarding WP2.

2.1 REST Architectural Style

In the year 2000, Roy Fielding introduced a new web architectural style in order to cope with the increasing complexity and heterogeneity of web services. The *Representational State Transfer* or *REST* [1] exhibits some distinct characteristics against which a service can be checked for compliance. *Richardson's Maturity Model* (RMM) [10] captures the principal REST characteristics that a web service must have in order to be considered RESTful. According to RMM, a web service is RESTful if:

- It comprises resources each of which is identified by a unique URI. This means that any two distinct resources must have different URIs. This REST constraint refers to level 1 of RMM (figure 1).
- Every such URI is accessed with the common HTTP verbs and these verbs must be used in the intended way. This means that in any case, the POST HTTP verb is used to create a new resource, the GET verb is used to retrieve an existing resource (or a list of resources), PUT is used to create or update an existing resource and DELETE to delete one. This REST constraint refers to level 2 of RMM.
- Thirdly, the service resources are interconnected with hypermedia links, which provide the clients the next possible actions from any given point of interaction with it. That is, once a client makes a request to a RESTful service, along with its response, it will receive a list of links that allow the client to ponder its next options without prior knowledge.

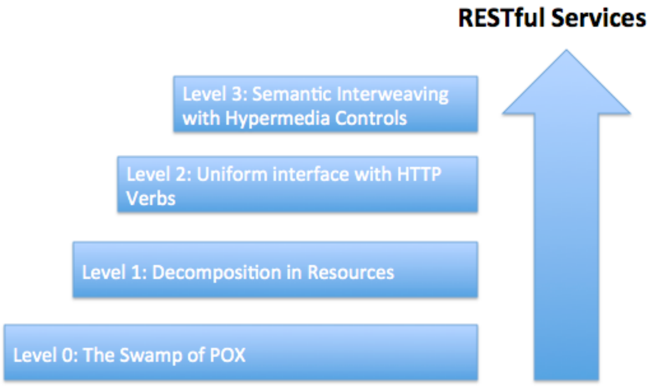

RESTful Services
Level 3: Semantic Interweaving with Hypermedia Controls
Level 2: Uniform interface with HTTP Verbs
Level 1: Decomposition in Resources
Level 0: The Swamp of POX

Figure 1 Web service friendliness Richardson's Maturity Model

2.2 Ruby on Rails

Ruby on Rails (Rails) [2] is an open-source framework for building web applications with the *Ruby* programming language. It was initially released in 2005 and has evolved and gained a lot of popularity since then. Some of the main features that make Rails popular are:

- The Ruby language is of simple nature, which makes it an easier starting point against more complex languages such as PHP and Java.
- It embeds some well-known software engineering paradigm/patterns such as the MVC pattern, the active record pattern and the convention over configuration paradigm.
- It is open source.
- It provides speed and agility, lowering costs and time needed to build services

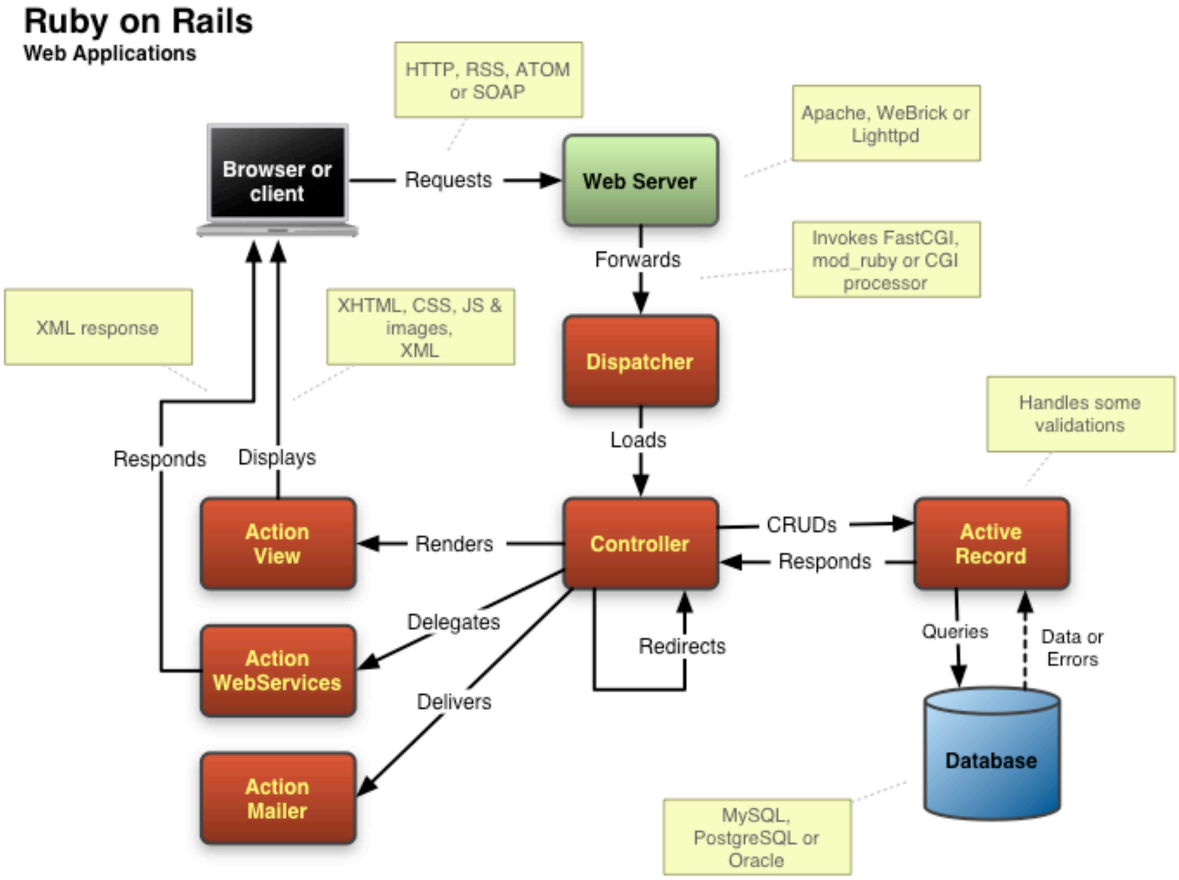


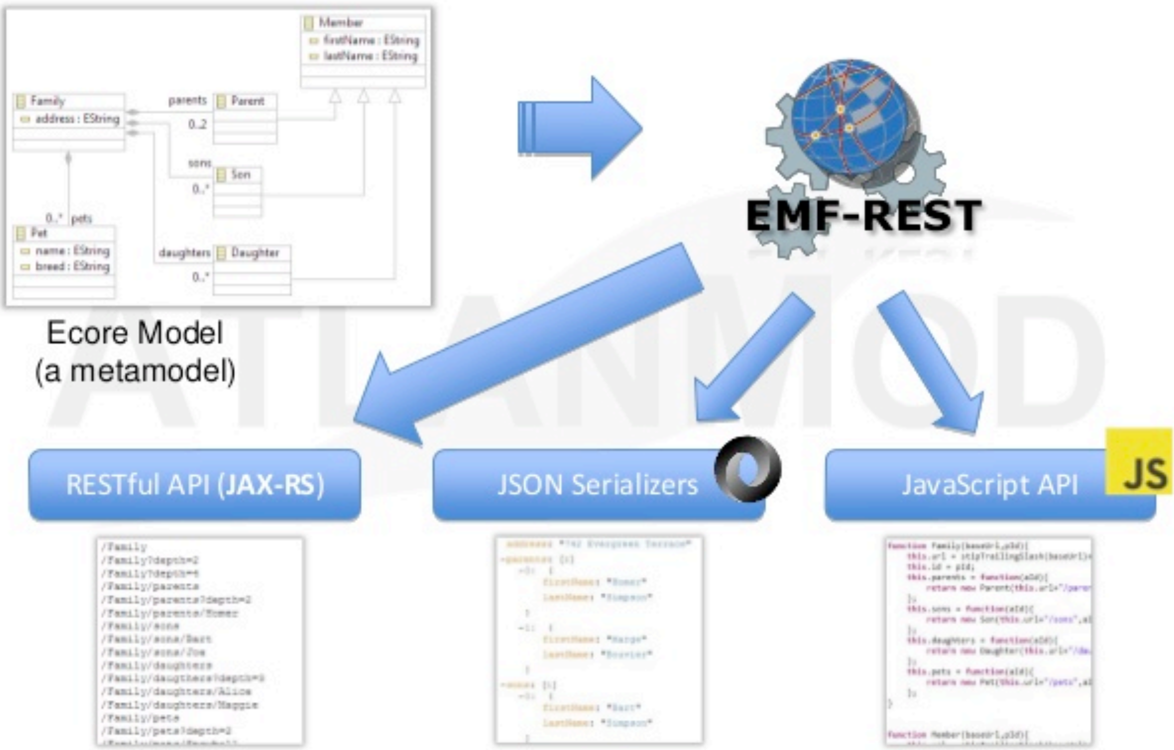
Figure 2 Basic structure of a Rails service (http://binaryhash.com/ruby-on-rails**)**

As figure 2 demonstrates, the core of a Rails service is built around the MVC design pattern. Some of its core elements are therefore *Models, Controllers* and *Views.* Models store data and map to tables of the service database. Controllers receive and respond to external requests by providing a set of actions. Although not compulsory, Rails emphasizes RESTful actions, which means that controllers preferably respond to create/new, edit/update, destroy, show/index. Finally, the *Views* are by default *erb* files, which are compiled to HTML.

Ruby in Rails has also received some criticism primarily regarding performance and scalability, which has made some companies switch from Rails implementations to other frameworks/solutions. However, it still retains a substantial portion of the RESTful development framework pie and some well-known web services use it, at least at some level, such as Twitter, Groupon and Scridb.

2.3 EMF-Rest

EMF-Rest [3] is an Eclipse-based framework that enables web service development by using Ecore meta-models. As figure 3 demonstrates, the developer starts by modelling his/her web service using the Ecore meta-model in the form of the familiar class diagram. Once this is done, the EMF-Rest framework scaffolds the corresponding service and provides a RESTful API by means of JAX-RS, JSON serializers and JavaScript API. Figure 4 demonstrates a RESTful API produced from an example "Family" Ecore meta-model, whilst at the left there is a JSON response produced by such a service. As it is observed in the JSON response EMF-REST produces no Hypermedia Controls and therefore, it does not achieve the maximum web-friendliness as it is defined in RMM.



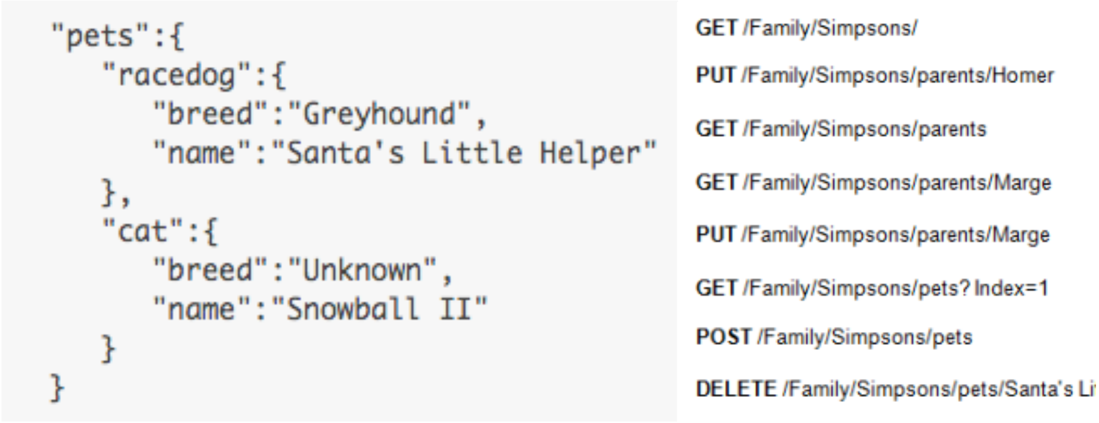
```
/Family
/Family?depth=2
/Family?depth=4
/Family/parents
/Family/parents?depth=2
/Family/parents/Homer
/Family/sons
/Family/sons/Bart
/Family/sons/Joe
/Family/daughters
/Family/daugthers?depth=1
/Family/daughters/Alice
/Family/daughters/Maggie
/Family/pets
/Family/pets?depth=2
```

```
 address: "742 Evergreen Terrace"
-parents: [2]
   -0: {
         firstName: "Homer"
         lastName: "Simpson"
     }
   -1: {
         firstName: "Marge"
         lastName: "Bouvier"
     }
-sons: [1]
   -0: {
         firstName: "Bart"
         lastName: "Simpson"
     }
```

```
function Family(baseUrl,pId){
    this.url = stripTrailingSlash(baseUrl)+
    this.id = pId;
    this.parents = function(aId){
        return new Parent(this.url+"/paren
    };
    this.sons = function(aId){
        return new Son(this.url+"/sons",a
    };
    this.daughters = function(aId){
        return new Daughter(this.url+"/da
    };
    this.pets = function(aId){
        return new Pet(this.url+"/pets",a
    };
}

function Member(baseUrl,pId){
```

Figure 3 AtlanMod overview presentation of EMF-REST at EclipseCon Europe 2013 Symposium



```
"pets":{
    "racedog":{
        "breed":"Greyhound",
        "name":"Santa's Little Helper"
    },
    "cat":{
        "breed":"Unknown",
        "name":"Snowball II"
    }
}
```

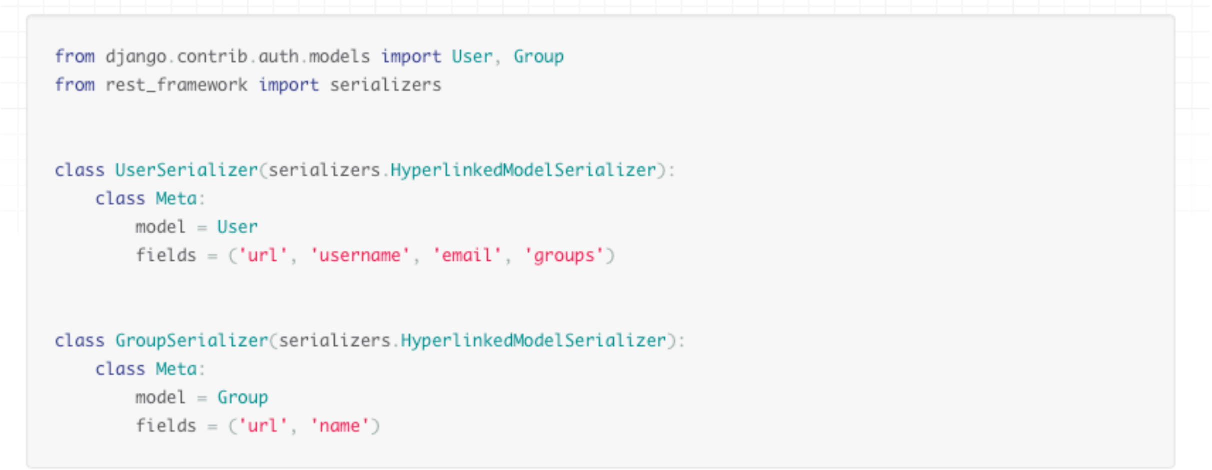
```
GET /Family/Simpsons/
PUT /Family/Simpsons/parents/Homer
GET /Family/Simpsons/parents
GET /Family/Simpsons/parents/Marge
PUT /Family/Simpsons/parents/Marge
GET /Family/Simpsons/pets? Index=1
POST /Family/Simpsons/pets
DELETE /Family/Simpsons/pets/Santa's Li
```

Figure 4 Left: example JSON response. Right: example of RESTful API

2.4 Django-REST-Framework

Django-REST-Framework [4] is another web application framework for scaffolding REST web services. This one is based on the python language. The principal elements/activities that a developer has to set-up in order to build a REST API are *Models, Serializers, Views* and naming URLs. In the rest of this subsection an overview of the development process using Django-REST-Framework is provided.

Initially, the developer has to define some sort of *Model* that represents the data with which the REST interface will interact with and then creates some serializers to serialize/deserialize it (figure 5). Then the developer has to define views, which define the way clients can interact with the API (requests/responses). The final step in order to include *hyperlinking* is to setup a python file with URL templates to be used. Figure 6 demonstrates a server response to a GET request. As it is stated in the documentation though, Django does not automatically produce Hypermedia Controls.

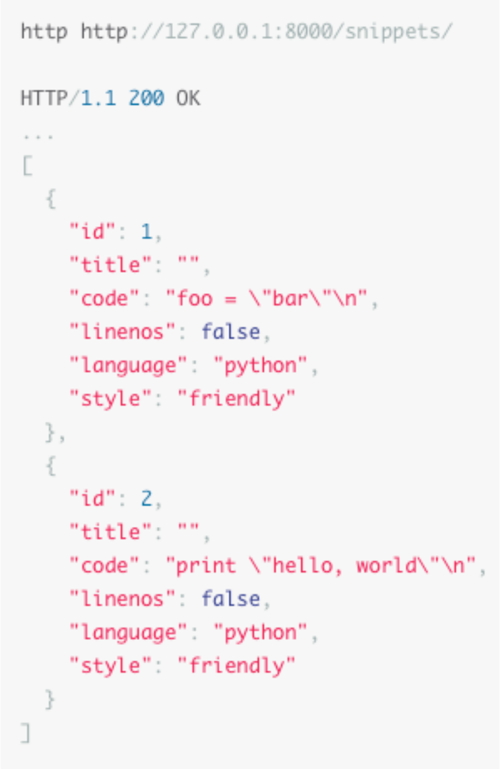


```
from django.contrib.auth.models import User, Group
from rest_framework import serializers


class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ('url', 'username', 'email', 'groups')


class GroupSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Group
        fields = ('url', 'name')
```

Figure 5 Django-REST-Framework serializers



```
http http://127.0.0.1:8000/snippets/

HTTP/1.1 200 OK
...
[
  {
    "id": 1,
    "title": "",
    "code": "foo = \"bar\"\n",
    "linenos": false,
    "language": "python",
    "style": "friendly"
  },
  {
    "id": 2,
    "title": "",
    "code": "print \"hello, world\"\n",
    "linenos": false,
    "language": "python",
    "style": "friendly"
  }
]
```

Figure 6 Service response to a GET HTTP request

2.5 S-CASE MDE Engine In Relation to State of the Art

The previous sections presented an overview of an indicative subset of the major attempts to facilitate the development of RESTful services that have been made so far. Some of them tend to achieve higher levels of web-friendliness in terms of RMM, whilst others attempt to achieve higher automation levels. The three indicative frameworks presented, as well as most of the other major attemps, belong to the second level of the RMM since they break down the service into a set of resources, each of which is addressed uniquely with a URI, and respect the semantics of the HTTP verbs as well. However, in most cases they do not interweave their resources with hypermedia controls and thus fail to achieve the maximum web-friendliness. Even in the few cases in which hypermedia links are introduced in some way, their usage is either limited or requires developer intervention since it is not automated.

In contrast, since the S-CASE goal is to automate the development of RESTful services as much as possible, a lot of effort has been made to ensure that the produced services will belong to the 3rd level of RMM. In this context, the achievements of the designed and developed S-CASE MDE engine in respect of the state of the art are:

1. The full incorporation of the REST architectural style to the produced S-CASE web services. The generated resources are semantically interweaved to each other with hypermedia controls. Thus, whenever clients interact with them, they receive, additionally to the expected HTTP response, a complete list with hypermedia links that list all the possible next actions that the client can take in its next request. These links, according to the definition of the PIM meta-model, include the URI of every resource with which an interaction has become viable. Additionally, they include the

HTTP verb, which can be used to interact with each resource, as well as the relation of every target resource with the action that has been just taken.

2. The application of the Model Driven Engineering paradigm to RESTful services development. That is, the MDE engine of S-CASE follows all the MDE steps (presented in section 3) and thus embodies the corresponding benefits. The S-CASE MDE engine provides a comprehensive mechanism that is extensible to support multiple platforms instead of a single combination of some programming language and two or three 3rd party frameworks like most other frameworks do.
3. The building of transparent transformations within the MDE phases by using the declarative programing paradigm with ATL leads to increased visibility and traceability of the whole transformation process.
4. The developed meta-models for PIM and PSM (Sections 4 and 5) embed a formalized set of OCL constraints that provide validation capability for every produced web service by the S-CASE platform in terms of compliance to the REST architectural style, as well as structural and behavioural consistency of all the intermediary and final artefacts.

3 S-CASE MDE Engine Introduction

3.1 MDE Process Overview

This subsection introduces the cornerstone technology adopted in S-CASE for automation, which is the *Model Driven Engineering* or in short *MDE* [5]. The Object Management Group (OMG) has introduced MDE since 2000, which aspires to change the software design and construction paradigm. Its principal concept is the transition of code-centric software engineering to a model-based one, so as to achieve higher consistency, increased level of automation and productivity. In a nutshell, it aspires to deliver software in less time and at a lower cost. Within S-CASE, MDE is one of the technologies used in order to automate the production of RESTful services. MDE comprises four distinct phases each of which is introduced and briefly explained subsequently.

During the first phase of MDE the developer formulates the *Computational Independent Model* (CIM). This model is the most abstract view of the underlying system. It comprises strictly application domain concepts, properties and entities. During this phase any design, architecture and implementation details are irrelevant. That is, the developer models the system to be built in terms of abstract domain concepts, their properties and relations. In S-CASE the domain of application is the web services that follow the *REST* architectural style (RESTful domain). Therefore, the principal concepts of the S-CASE MDE CIM are *resources, web interfaces, resource properties, resource relations* etc. The CIM meta-model of the S-CASE MDE engine will be defined in deliverable D2.3 *Ontology to MOF models transformation.*

Once the CIM of an envisioned service is formulated, a “model to model” (M2M) transformation takes place that transforms the CIM into the corresponding *Platform Independent Model* or PIM. The PIM introduces the abstract design of the system without any implementation details. Therefore, during the transformation that takes place from the CIM to the corresponding PIM, the abstract architecture of the system is applied to every CIM concept in order to form the complete design that satisfies the system requirements. Section 4 of this document fully defines the language of S-CASE MDE’s PIM, the *PIM meta-model*, along with the architecture it introduces.

The third phase of the MDE process involves the transformation of the PIM to one of its possible *Platform Specific Models* (PSM). The PSM conforms to the design that PIM introduces but specifies the way it will be implemented by introducing specific implementation language details, as well as several external frameworks, libraries and/or programming language specific design patters or idioms. All these together form the target platform that the PSM models. Thus, the PIM to PSM transformation specifies each PIM element using the concrete technologies that form the PSM. It must be noted that there is a one-to-many relationship between a PIM model and the compatible PSMs. That is, every abstract design of a system that the PIM introduces can be implemented with numerous distinct mixtures of specific technologies that form different PSMs. Section 5 of this document fully defines the language of S-CASE MDE PSM, the *PSM meta-model,* along with the concrete technologies that are used in it. Thereafter, section 6, conceptually defines the PIM to PSM M2M transformation that is implemented with the *Atlas Transformation Language* (ATL).

Finally, the MDE engine produces the code by filling code templates with the meta-data that is available in the PSM. Some parts of the code may be fully implemented, whilst others may need extra intervention by developers. The code templates in par with the code generator of S-CASE MDE engine will be defined in D2.3 along with the CIM. Figure 7 summarizes the MDE phases and transformations.

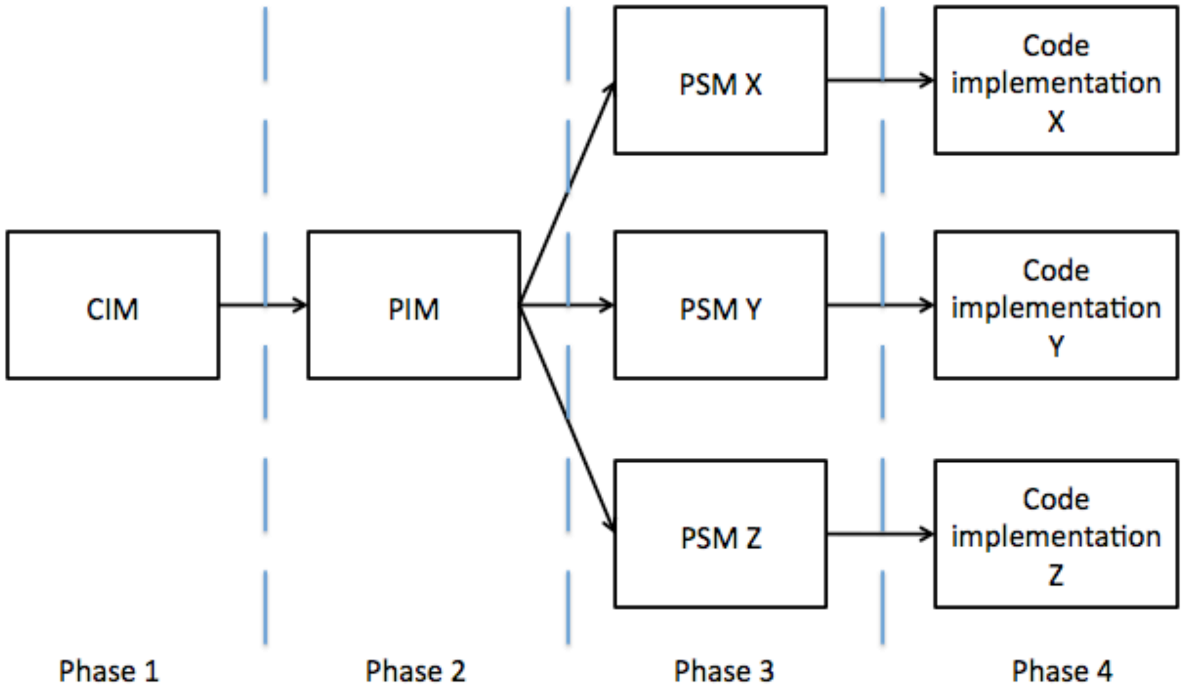


Figure 7 The four phases of Model Driver Engineering

3.2 Associated Technologies and Frameworks

This section briefly presents the technologies used to define and implement the PIM and PSM meta-models as well as the M2M transformation from the PIM to the PSM.

3.2.1 Ecore meta-model

In the definition of *Meta-Object Facility 2.0* MOF, apart from the *Complete Meta-Object Facility* or *CMOF*, OMG introduced an "essential" subset of it, the *Essential Meta-Object Facility* or *EMOF* that aims to model primarily structural aspects of systems, whilst the CMOF models behavioural ones as well. The Eclipse framework implementation of the EMOF is the Ecore meta-model [6]. The Ecore meta-model is the cornerstone of many modelling applications and plugins within the Eclipse ecosystem.

Since S-CASE aims to offer developers tools within the Eclipse framework, the Ecore meta-model has been selected as the core upon which the S-CASE profiles will be created. That is, the CIM, PIM and PSM meta-models are all extensions of the Ecore meta-model. Therefore, S-CASE produced artefacts can be used with other tools and Eclipse plugins that conform to

Ecore. This selection impacts positively on the potential S-CASE users group due to the large Eclipse user pool.

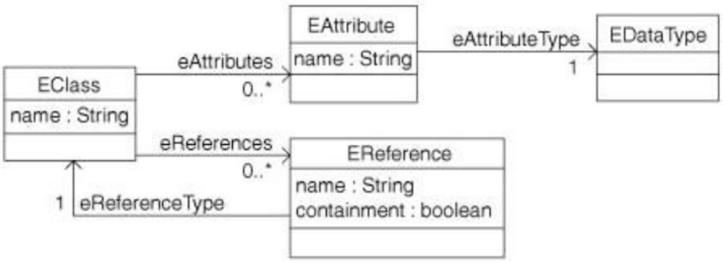


Figure 8 Simplified subset of the Ecore meta-model

Figure 8 depicts a simplified subset of the Ecore meta-model. The *EClass* element is used to represent a modeled class. It has a name, and zero or more *EAttributes or EReferences.* The *EAttribute* is used to represent a modeled attribute of an *EClass* and has a *name* and a *type.* Thereafter, *EReferences* represent associations between *EClasses* either of containment type or not and lastly, the *EDataType* represents either primitive or custom datatypes. The S-CASE CIM, PIM and PSM meta-models extend these Ecore elements in order to define the S-CASE profiles.

3.2.2 Object Constraint Language (OCL)

The *Object Constraint Language* (OCL) [7] is a formal language used to describe expressions on MOF models. Such expressions typically describe invariant constraints that must hold within the system being modelled or queries to its objects. OCL can be used primarily:

- As a query language
- To specify invariants on classes and types of a class model
- To describe pre- and post-conditions on operations and methods

Within the S-CASE MDE engine, OCL is used to specify invariants that form concrete validation schemes of CIM, PIM and PSM meta-model instances. Sections 4 and 5 of this document specify all the OCL constraints that an instance model has to conform to in order to be a valid one of the S-CASE MDE PIM or PSM meta-model respectively.

3.2.3 Eclipse Modelling Framework (EMF)

The *Eclipse Modelling Framework* (EMF) [6] is a framework that unifies many of the Eclipse ecosystem facilities. As figure 9 depicts, EMF unifies the *Java* language with the *XMI* specification and the UML one as well. This means that instances of any of the three languages can be interpreted transparently to any of the others offering thus, a robust, reliable connection among different representations of the same model. Therefore an

Eclipse EMF user does not need any more to, for example, manually convert UML diagrams to Java instances and therefore risk introducing inconsistency among them.

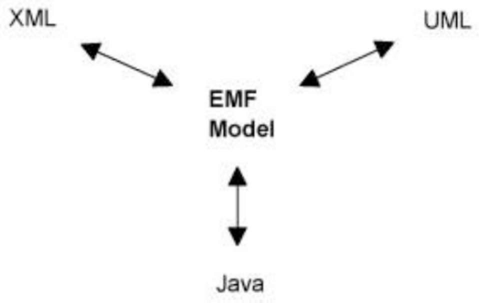


Figure 9 The Eclipse Modelling Framework unifies Java, XML and UML

Among others, EMF embeds a model validation scheme against an Ecore meta-model. The MDE engine of S-CASE, therefore, uses the EMF in order to validate instances of CIM, PIM and PSM meta-models. This happens against both structural and behavioural aspects. The structural ones are imposed by the structure of the CIM, PIM and PSM Ecore meta-models of S-CASE, whilst the behavioural ones are embedded in them as OCL invariant constraints using the OCLinEclipse facility.

3.2.4 Atlas Transformation Language (ATL)

The *Atlas Transformation Language* (ATL) [8] is a hybrid language (declarative and imperative) that is used in M2M transformation definitions. The AtlanMod research group introduced ATL as a response to the OMG's QVT request for proposal. In the context of MDE, ATL provides developers the means to specify the way to produce a number of target models from a set of source models. Figure 10 demonstrates the mechanism by means of which the ATL language performs M2M transformations.

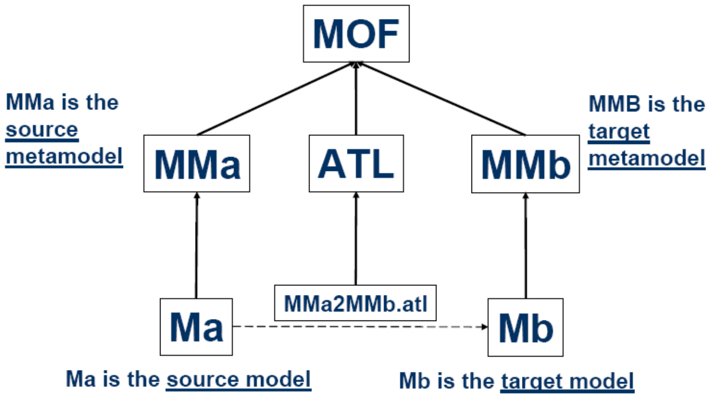


Figure 10 ATL Operational Context

ATL conforms itself to the MOF meta-meta-model and performs transformations among models that are instances of meta-models that also conform to MOF. For example, following a set of ATL rules, a source model *Ma,* which conforms to the MMa meta-model, is transformed, element by element, to an *Mb* model that conforms to the *MMb* meta-model. In the same way S-CASE MDE Engine uses sets of ATL rules to transform elements of CIM, PIM and PSM meta-models to elements of other models. This is possible since the CIM, PIM and PSM meta-models are defined using the Ecore meta-model, which is an implementation of the EMOF that in turn is a subset of CMOF. Section 6 of this document conceptually defines all the implemented ATL rules in order to perform the PIM to PSM transformation within the S-CASE MDE engine.

ATL as a hybrid language provides a variety of ways to define such rules. These are:

- *Matched Rules:* The matched rules are declarative rules. Such rules enable to match some elements from the source model and generate a number of distinct target model elements. The ATL transformation engine calls such rules implicitly whenever a source model element, which is "matched" in a rule, is found and then the target element is generated. In order to provide some execution order control, ATL provides also a subcategory of the matched rules, the *Lazy* ones. These are also declarative, but are called only by other rules explicitly. Both of them is usually more suitable in order to perform transformations to target model elements that do have source model counterparts.
- *Called Rules:* The called rules on the other hand are imperative rules. These rules must be invoked by other rules and do not have any source element as input. Such rules are usually suitable in order to perform transformation rules that generate target model elements that do not have source model counterparts.

3.3 S-CASE MDE Engine Architecture Overview

The S-CASE MDE engine comprises four distinct components, each of which matches each MDE phase. Therefore, there exist four components, one for the CIM formulation, one for the PIM, one for the PSM and one that generates the actual code of the envisioned system RESTful service. Figure 11 demonstrates this structure.

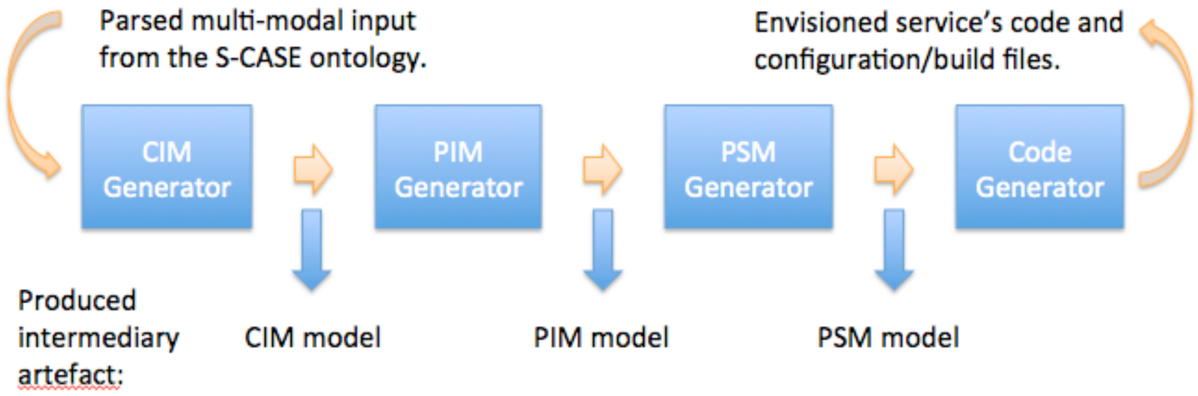

Parsed multi-modal input from the S-CASE ontology.
Envisioned service's code and configuration/build files.
CIM Generator
PIM Generator
PSM Generator
Code Generator
Produced intermediary artefact:
CIM model
PIM model
PSM model

Figure 11 S-CASE MDE engine component level decomposition

The *CIM Generator* module reads the parsed multi-modal input from the S-CASE ontology and generates the CIM model of the envisioned system to be produced. Deliverable *D2.3 Ontology to MOF models transformation mechanism* will define the CIM meta-model to which the CIM model must conform. The output of the CIM generator is the CIM model of the envisioned system in *XMI* format that conforms to the CIM meta-model.

The CIM model, in *XMI* format, is the input of the *PIM Generator* module. The principal activity of this module is to read the envisioned system’s CIM model and fully define its abstract design by applying to it the REST architectural style. The outcome of this process is the envisioned system’s PIM model that conforms to the PIM meta-model, which section 4 of this document defines. The PIM model is exported in XMI format as well.

The *PSM Generator* module parses the produced PIM model and then specifies its abstract design. That is, for every abstract element of the PIM model, PSM Generator introduces a PSM one that is semantically equivalent but is defined with specific technologies such as a specific programming language, third party frameworks etc. and conforms to the PSM meta-model that section 5 of this document defines. Once again, the output is in XMI format. Finally, section 6 of this document defines the PIM to PSM transformation with ATL using its full hybrid declarative/imperative capacity.

The produced PSM contains all the needed information to automate the generation of code as much as possible. This task is handled by the *Code Generator* module, which produces all the source code files as well as the build files, the configuration files and the whole project structure. S-CASE will achieve a varying automation level depending on the nature of the envisioned system. Once S-CASE scaffolds the maximum of the envisioned system, the S-CASE user will have to intervene and complete the rest of it.

4 Ecore PIM Meta-model

4.1 PIM Ecore Meta-model Design Goals and Constraints

This section relates the underlying definition of the PIM meta-model with the design goals and constraints imposed by the S-CASE requirements and specifications as they are defined in *D1.3.1 S-CASE technical specification* as well as in *D1.1 S-CASE requirements* deliverables. Afterwards, this section provides the PIM meta-model's design rational so as to satisfy the aforementioned goals and constraints.

Design Goals and Constraints

The principal goal of the PIM meta-model is to introduce the envisioned system design so as to satisfy the S-CASE user requirements. Since S-CASE scaffolds RESTful web services, the PIM applies to every CIM concept the REST architectural style. It has to do so by being fully compatible with Richardson's Maturity Model (RMM), which has been presented in section 2.1 of this document.

Moreover, the S-CASE requirements that are related with the PIM meta-model, which can be found in the D1.3.1 deliverable, are a subset of PIM module requirements. In summary the functional and non-functional requirements, which the design of the PIM meta-model must satisfy are the following:

1. The PIM meta-model should enable storage and retrieval of web service data in a uniform way.
2. The S-CASE system must produce systems that follow the REST architectural style.
3. The S-CASE system must produce the envisioned local RDBMS schema of the system.
4. The S-CASE system must produce systems that embed the composition of other RESTFul or SOAP web services if needed.

PIM Meta-model Design Rational

The Ecore PIM meta-model, that is defined in this document, models a web service resource as a three-part component entity as figure 12 demonstrates. These three parts are a slight variation of the popular *Model View Controller* (MVC) design pattern so as to better fit the REST nature. This variation, the *Model Representation Controller* (MRC), comprises one *Model* element, one or more *Representation* elements and one *Controller* element*.*

The *Model* element models the data of a service resource. The data is modelled in terms of their name, type and multiplicity. The *Representation* element represents the various web media types, such as "application/JSON", that a specific resource can be formatted during its interaction with a client. The *Controller* element represents the web API of the resource. Every such controller has a unique URI through which clients can access it, as REST requires. Actually, in the PIM meta-model two types of such "abstract" resources are defined. The Model element one models a simple *Java-Bean*-like data model whilst the other is a wrapper for the composition of SOAP/REST third party services as required by the S-CASE specifications.

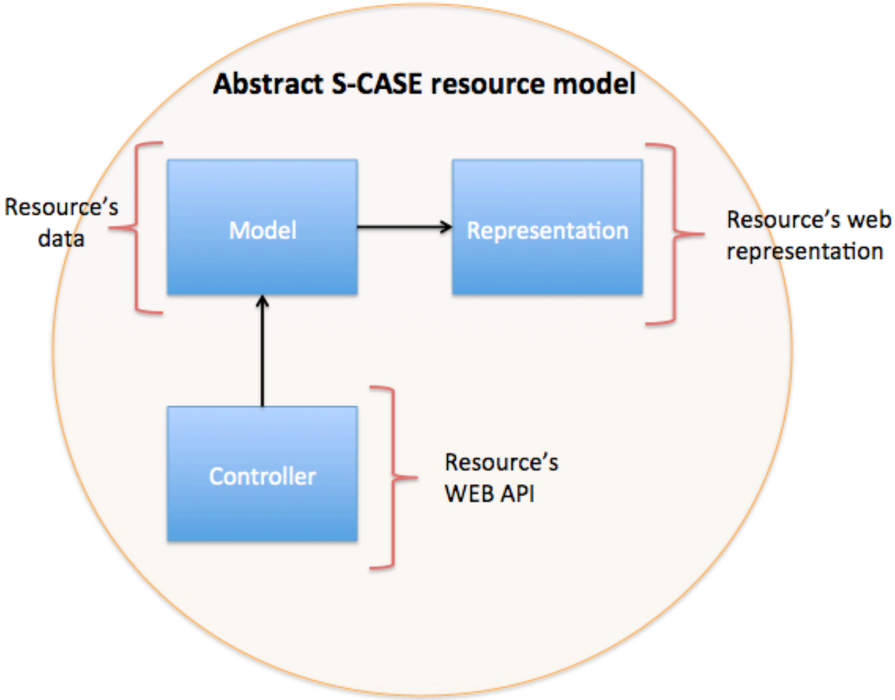


Figure 12 Abstract Resource Model

With reference to the PIM meta-model definition, explained in detail in the following section, every controller exposes distinct interaction entries for each of the HTTP verbs, each of which respects their aforementioned semantics, as REST requires. Additionally, following the widespread software engineering principle of separating interfaces from their implementation, these entry points only formulate the API of each resource and do not handle the actual requests. Instead, the request handling is delegated to specific per HTTP verb request handlers as Figure 13 demonstrates. Apart from accessing the local database of the service, in order to serve the request, these handlers also calculate all the hypermedia links that will be sent to the client. These links inform the client about the next valid actions it can take after each request, by pointing the client to the controllers of other resources achieving thus the semantic interweaving and interconnection of service resources that the REST architectural style requires. Thus, since the services produced by S-CASE are compartmentalized to uniquely addressed resources, which respect the HTTP verbs semantics, and are interconnected with hypermedia links, they fall into the 3rd level of the RMM and are RESTful.

In order to satisfy the requirement for a uniform storage access, the PIM meta-model embeds the *Repository* pattern. Therefore, it introduces, for each service, a unique database controller, which offers high-level access to and from the local database, hiding from the rest of the system any low level details that the usual interaction with relational databases requires. Additionally, apart from the automatically produced code, this database controller offers a high-level storage handler for any custom changes or additions that the S-CASE developer wishes to make in the final product. Finally, the local database is automatically created since the PIM meta-model maps each appropriate *Model* element to a relational database table respecting as well its referential integrity as it is explained in the next subsections.

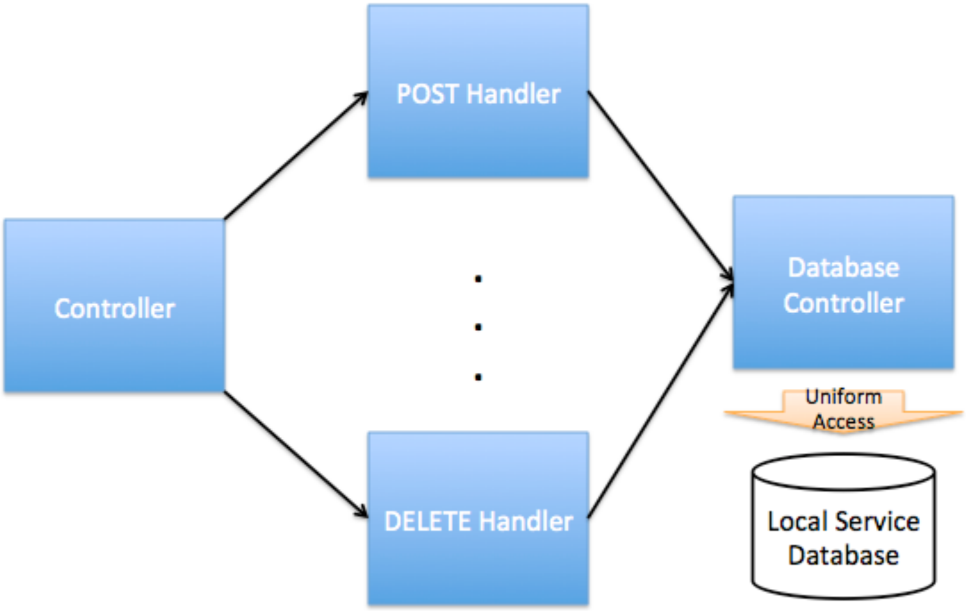


Figure 13 Per HTTP verb handling delegation and uniform storage access.

4.2 PIM Ecore Meta-model Definition

4.2.1 Introduction

In order to fully define the PIM meta-model, its various aspects must be explained, demonstrated as well as documented. Each of the remaining subsections of this section precisely defines the structure of each PIM meta-model element, its properties and their behavioural constraints.

Each such subsection begins with a meta-model element overview that provides a high level description. Depending on the complexity of each element, one or more Ecore class diagrams may be included to clearly demonstrate the relations of it with the rest of the elements of the PIM meta-model. Due to space limitations, the full PIM meta-model diagram can be found at https://github.com/s-case/mde.

For each element two tables are provided. The first table defines the properties of each PIM meta-model. Therefore, they provide the *name*, the *type* as well as an explanation of every such property. The second table defines the relations of the meta-model element. This table comprises the name of the related elements, the type of relation (e.g. composition, association etc.) and the multiplicity of that relation. The last column of such tables, defines the structural constraints of this element within the PIM meta-model. Finally, the last part of each element subsection includes a list of the OCL constraints conceptual description that apply to it. These constraints define the behavioural constraints of each element as well as the well-formness rules it must comply with.

4.2.2 PIM Meta-model Elements

This section lists all the PIM meta-model elements defined using the format presented in subsection 4.2.1.

4.2.2.1 PIM Meta-model Custom Data Types

This subsection defines the custom data types that PIM meta-model elements use in their definitions.

MediaType

The *MediaType* data type is an enumeration that models all the possible input and output media types that are supported in any case by S-CASE. These can be found in the following table:

Table 1 Supported Media Types

MediaType	Relative HTTP Header value	Default Value	Explanation
JSON	application/JSON	Yes	When JSON media type is selected for a resource the related WEB API expects as input and or outputs the data in application/JSON format
XML	application/XML	No	When XML media type is selected for a resource the related WEB API expects as input and or outputs the data in application/XML format

CRUDVerb

The *CRUDVerb* data type is an enumeration, which models the four CRUD verbs, *Create, Read, Update, Delete*. They are defined in the following table:

Table 2 CRUDVerb Data Type

CRUDVerb	Default Value	Explanation
CREATE	Yes	The CREATE verb is used to model the abstract form of the HTTP POST verb within the PIM meta-model. Wherever it is used, it respects the REST architectural style and thus denotes the action of creating a new resource.
READ	No	The READ verb is used to model the abstract form of the HTTP GET verb within the PIM meta-model. Wherever it is used, it respects the REST architectural style and thus denotes the action of retrieving either a specific existing

		resource or a list of existing resources.
UPDATE	No	The UPDATE verb is used to model the abstract form of the HTTP PUT verb within the PIM meta-model. Wherever it is used, it respects the REST architectural style and thus denotes the action of updating a specific existing resource. Moreover, since S-CASE produces web services that produce automatically resource identifiers, the UPDATE verb is not used to also create resources.
DELETE	No	The DELETE verb is used to model the abstract form of the HTTP DELETE verb within the PIM meta-model. Wherever it is used, it respects the REST architectural style and thus denotes the action of deleting a specific existing resource as well as any subsequence resources related to it.

RDBMSVerb

The *RDBMSVerb* data type is an enumeration that models the four verbs used to interact with relational database schemas. These are the *INSERT, SELECT, UPDATE* and *DELETE* verbs. They are defined in the following table:

Table 3 CRUDVerb Data Type

RDBMSVerb	Default Value	Explanation
INSERT	Yes	Whenever a PIM meta-model element has a property with the *INSERT RDBMSVerb,* its primary goal is to perform an INSERT activity to the services local database.
SELECT	No	Whenever a PIM meta-model element has a property with the *SELECT RDBMSVerb,* its primary goal is to perform a SELECT activity from the services local database.
UPDATE	No	Whenever a PIM meta-model element has a property with the *UPDATE RDBMSVerb,* its primary goal is to perform an UPDATE activity to the services local database.
DELETE	No	Whenever a PIM meta-model element has a property with the *DELETE RDBMSVerb,* its primary goal is to perform a **cascade** DELETE activity to the services local database.

LinkType

The *LinkType* data type is an enumeration that models the three possible hypermedia link types supported by S-CASE. These are the *PARENT, SIBLING* and *CHILD* data types. They are defined in the following table:

Table 4 CRUDVerb Data Type

LinkType	Default Value	Explanation
PARENT	Yes	Whenever a PIM meta-model *Hypermedia* element has a property with the *PARENT LinkType* it models a link that connects a specific resource A with a possible action at one of the service resources that have A as related resource.
SIBLING	No	Whenever a PIM meta-model *Hypermedia* element has a property with the *SIBLING LinkType* it models a link that connects a specific resource with one of its possible actions.
CHILD	No	Whenever a PIM meta-model *Hypermedia* element has a property with the *CHILD LinkType* it models a link that connects a specific resource A with a possible action of one of A's related resources.

4.2.2.2 RESTfulServicePIM Element

Overview

RESTfulServicePIM is the root element of the PIM meta-model. There may exist exactly one *RESTfulServicePIM* element in any PIM produced by S-CASE. It comprises the abstract form of all the elements that form a RESTful service and is associated with them by means of a composition association. This suggests that any other element of the PIM must be contained by this specific root element. The rest of this subsection provides information on its properties, relations and their structural and behavioural restrictions. Figure 14 demonstrates a simplified diagram of a RESTfulServicePIM element with its properties and relations.

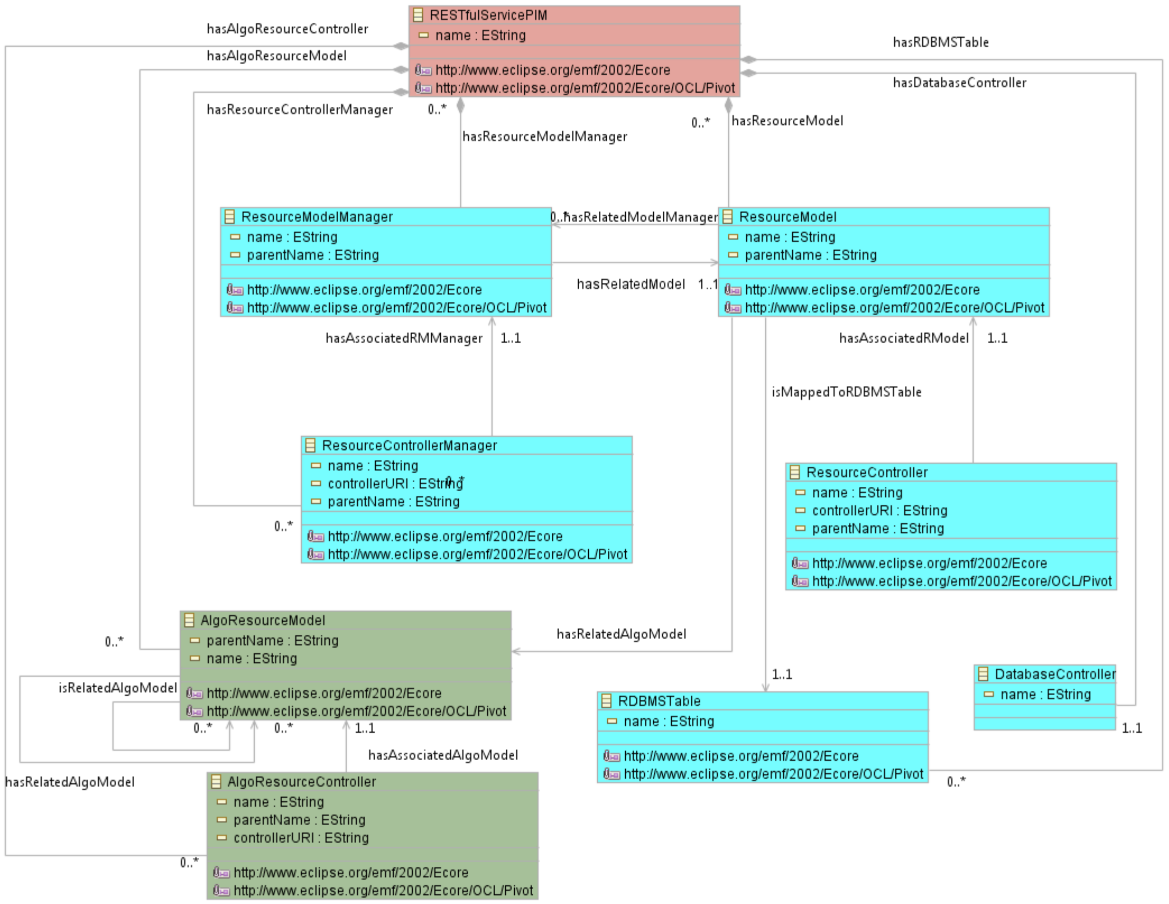

RESTfulServicePIM
name : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
hasAlgoResourceController
hasAlgoResourceModel
hasResourceControllerManager
0..*
hasResourceModelManager
0..*
hasResourceModel
hasRDBMSTable
hasDatabaseController
ResourceModelManager
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..hasRelatedModelManager
hasRelatedModel 1..1
ResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
hasAssociatedRMManager 1..1
hasAssociatedRModel 1..1
isMappedToRDBMSTable
ResourceControllerManager
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*
ResourceController
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
AlgoResourceModel
parentName : EString
name : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*
isRelatedAlgoModel
hasRelatedAlgoModel
0..* 0..* 1..1
hasRelatedAlgoModel
hasAssociatedAlgoModel
1..1
RDBMSTable
name : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
DatabaseController
name : EString
1..1
0..*
AlgoResourceController
name : EString
parentName : EString
controllerURI : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*

Figure 14 Simplified PIM meta-model: RESTfulServicePIM element

Properties

The properties of the RESTfulServicePIM are depiceted in the following table.

Table 5 RESTfulServicePIM’s Properties

Name	Type	Multiplicity	Explanation
Name	EString	1	This is the provided service name by the S-CASE developer. Among others, it is used to produce the service folder structure and build files.

Relations

Table 6 RESTfulServicePIM’s Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
ResourceModel	composition	0..*	*RESTfulServicePIM* may contain zero or more *ResourceModel elements*.Any *ResourceModel that* exists in PIM belongs in

			the list of the associated *ResourceModels* of the *RESTfulServicePIM element.*
ResourceController	composition	0..*	*RESTfulServicePIM* may contain zero or more *ResourceController elements*. Any *ResourceController that* exists in PIM belongs in the list of the associated *ResourceControllers* of the *RESTfulServicePIM element.*
ResourceModelManager	composition	0..*	*RESTfulServicePIM* may contain zero or more *ResourceModelManager elements*. Any *ResourceModelManager that* exists in PIM belongs in the list of the associated *ResourceModelManager* of the *RESTfulServicePIM element.*
ResourceControllerManager	composition	0..*	*RESTfulServicePIM* may contain zero or more *ResourceControllerManager elements*. Any *ResourceControllerManager that* exists in PIM belongs in the list of the associated *ResourceControllerManager* of the *RESTfulServicePIM element.*
AlgoResourceModel	composition	0..*	*RESTfulServicePIM* may contain zero or more *AlgoResourceModel elements*. Any *AlgoResourceModel that* exists in PIM belongs in the list of the associated *AlgoResourceModel* of the *RESTfulServicePIM element.*
AlgoResourceController	composition	0..*	*RESTfulServicePIM* may contain zero or more *AlgoResourceController elements*. Any *AlgoResourceController that* exists in PIM belongs in the list of the associated *AlgoResourceController* of the *RESTfulServicePIM element.*
DatabaseController	composition	1	*RESTfulServicePIM* must contain exactly one *DatabaseController element*.
RDBMSTable	composition	0..*	*RESTfulServicePIM* may contain zero or more *RDBMSTable elements*. Any *RDBMSTable that* exists in PIM belongs in the list of the associated *RDBMSTable* of the *RESTfulServicePIM element.*

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *RESTfulServicePIM* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *controllersHaveUniqueURIs:* As it is defined in the PIM meta-model, *ResourceControllers, ResourceControllerManagers* and *AlgoResourceControllers* have a *URI* property. Since this meta-model follows the REST architectural style all these controllers must have unique *URI* by means of which they can be addressed. This OCL constraint ensures the uniqueness of all the automatically produced *URI*s.

- *uniquePIMComponentNames:* This OCL constraint validates the uniqueness of the name among any PIM component that will end to be a class once code is produced. This guarantees that no naming conflicts will be encountered during the code production phase. The name uniqueness is validating among the *ResourceModel, ResourceController, ResourceModelManager, ResourceControllerManager, AlgoResourceModel* and *AlgoResourceController* PIM meta-model elements.

- *uniqueRModelRDBMSTableMapping:* This constraint ensures that any two distinct *ResourceModels* are mapped to two distinct *RDBMSTable*s.

- *validateDatabaseReferentialIntegrity:* This constraint ensures that for any two related resources the proper foreign key columns are included in the corresponding *RDBMSTables.*

- *correctlyMatchingRControllerCRUDActivitiesWithRDBMSActivities:* This constraint ensures that for any *CRUDActivity* of a *ResourceController* PIM meta-model element there exists exactly one *RDBMSActivity* in the *DatabaseController* one.

- *correctlyMatchingRCManagerCRUDActivitiesWithRDBMSActivities:* This constraint ensures that for any *CRUDActivity* of a *ResourceControllerManager* PIM meta-model element there exists exactly one *RDBMSActivity* in the *DatabaseController*.

4.2.2.3 ResourceModel Element

Overview

The *ResourceModel* element models the data of a resource. This can be seen as the *Model* part of the *Model-View-Controller* design pattern or a simple Java Bean. This element aims to fully define every resource property by providing its name and type as well as providing functions that access them (*setters/getters).* Figure 15 demonstrates a simplified Resource Model diagram with its properties and relations.

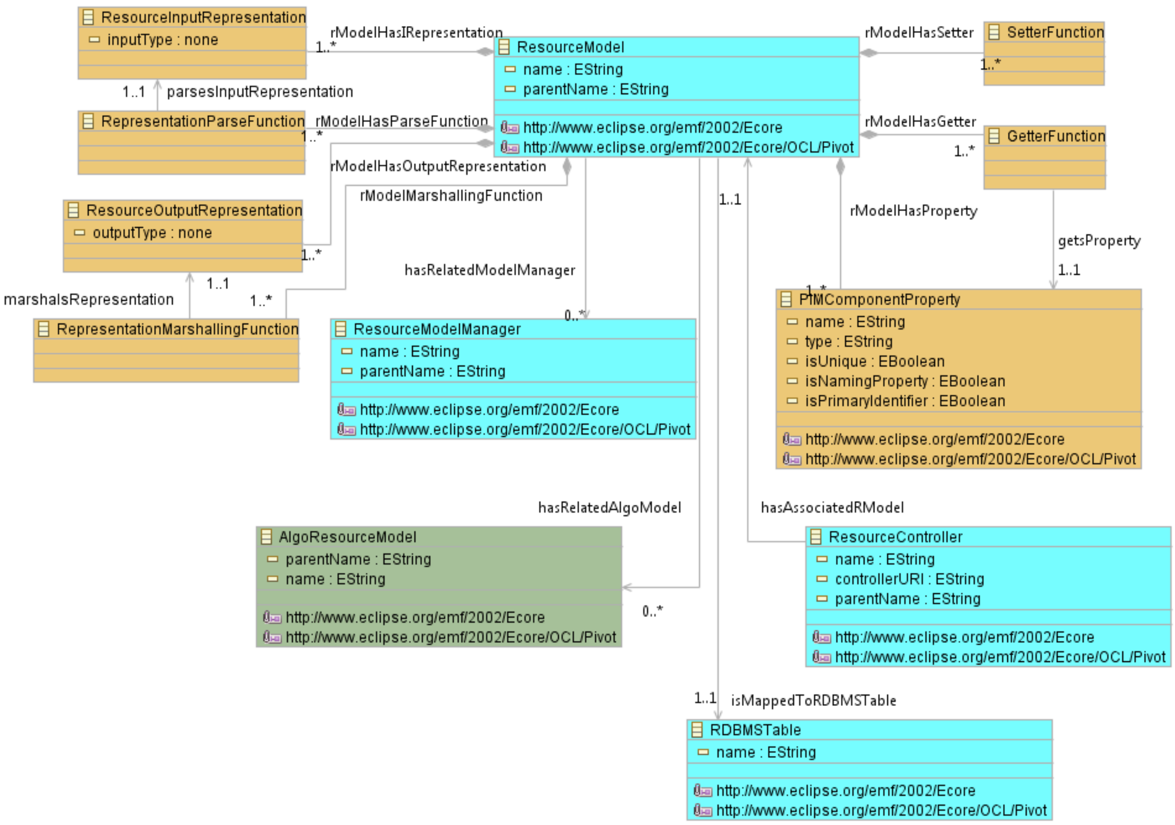

ResourceInputRepresentation
inputType : none
rModelHasIRepresentation
1..*
ResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
rModelHasSetter
SetterFunction
1..*
1..1
parsesInputRepresentation
RepresentationParseFunction
rModelHasParseFunction
1..*
rModelHasGetter
GetterFunction
1..*
rModelHasOutputRepresentation
rModelMarshallingFunction
ResourceOutputRepresentation
outputType : none
1..*
1..1
rModelHasProperty
getsProperty
1..1
hasRelatedModelManager
marshalsRepresentation
1..1
1..*
0..*
1..*
PIMComponentProperty
name : EString
type : EString
isUnique : EBoolean
isNamingProperty : EBoolean
isPrimaryIdentifier : EBoolean
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
RepresentationMarshallingFunction
ResourceModelManager
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
hasRelatedAlgoModel
hasAssociatedRModel
AlgoResourceModel
parentName : EString
name : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*
ResourceController
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
1..1
isMappedToRDBMSTable
RDBMSTable
name : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot

Figure 15 PIM meta-model simplified diagram: ResourceModel element

Properties

Table 7 ResourceModel's Properties

Name	Type	Multiplicity	Explanation
Name	EString	1	This is the name of the *ResourceModel.*
parentName	EString	1	This is the name of the parent *CIM Resource* of this *ResourceModel.*

Relations

Table 8 ResourceModel's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
PIMComponentProperty	composition	1..*	The *ResourceModel* element must be associated with at least one *PIMComponentProperty.* Since *ResourceModels* model the data of the resource, the existence of a *ResourceModel* that does not have any, would be a contradiction since there would be no data to model.
SetterFunction	composition	1..*	The *ResourceModel* element must be associated with at least one *SetterFunction*.

			Since *ResourceModels* have at least one *PIMComponentProperty* there must be at least one *SetterFunction* to alter its value.
GetterFunction	composition	1..*	The *ResourceModel* element must be associated with at least one *GetterFunction*. Since *ResourceModels* have at least one *PIMComponentProperty* there must be at least one *GetterFunction* to retrieve its value.
AlgoResourceModel	association	0..*	The *ResourceModel* element may be associated with zero or more *AlgoResourceModels*. This association models the fact that a specific *AlgoResourceModel* may be related resource of a specific *ResourceModel.*
ResourceInputRepresentation	composition	1..*	The *ResourceModel* element must be associated with at least one *ResourceInputRepresentation* element because there must be defined at least one input media type in which clients will be able to send data modelled by this *ResourceModel* to the envisioned service.
RepresentationParseFunction	composition	1..*	The *ResourceModel* element must be associated with at least one *RepresentationParseFunction* since it is associated with at least one *ResourceInputRepresentation* and therefore at least one function with parsing capabilities is needed.
ResourceOutputRepresentation	composition	1..*	The *ResourceModel* element must be associated with at least one *ResourceOutputRepresentation* element because there must be defined at least one input media type in which the envisioned service will send data modelled by this *ResourceModel* to any of its clients.
ResourceMarshalingFunction	composition	1..*	The *ResourceModel* element must be associated with at least one *RepresentationMarshallingFunction* since it is associated with at least one *ResourceOutputRepresentation* and therefore at least one function with marshalling capabilities is needed.
ResourceModelManager	association	0..*	The *ResourceModel* element may be associated with zero or more *ResourceModelManager* elements. This association models the fact that a specific *ResourceModel* may have some related *ResourceModelManager*.
RDBMSTable	association	1	The *ResourceModel* element must be associated with exactly one *RDBMSTable* element which will hold the modelled data of the *ResourceModel* in the envisioned services

			local database.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *ResourceModel* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *uniqueNamingProperty:* This OCL constraint checks whether there exists exactly one *PIMComponentProperty* of this *ResourceModel* that is naming property.

- *uniquePrimaryIdentifier:* This constraint checks whether a *ResourceModel* has exactly one *PIMComponentProperty* that is primary identifier.

- *existantSettersForAllProperties:* This constraint checks whether there exists exactly one *SetterFunction* for every *PIMComponentProperty* this *ResourceModel* has.

- *existantGettersForAllProperties:* This constraint checks whether there exists exactly one *GetterFunction* for every *PIMComponentProperty* this *ResourceModel* has.

- *uniqueLinklistProperty:* This constraint checks whether every *ResourceModel* has exactly one property named "linklist" which will be used in runtime in order to group all the *HypermediaLinks* that will be sent back to client as the next possible actions.

4.2.2.4 ResourceController Element

Overview

The *ResourceController* element models the web interface of a resource. This happens in respect of the REST architectural style, which means that every *ResourceController* is assigned a unique *URI* and the CRUDVerbs are used as intended when accessing that *URI*. Figure 16 demonstrates a simplified diagram of a Resource Controller element with its properties and relations.

Properties

Table 9 ResourceController's Properties

Name	Type	Multiplicity	Explanation
Name	EString	1	This is the name of the *ResourceController.*

parentName	EString	1	This is the name of the parent *CIM Resource* of this *ResourceController.*
controllerURI	EString	1	This EString contains the *URI* that is automatically assigned to every *ResourceController.*

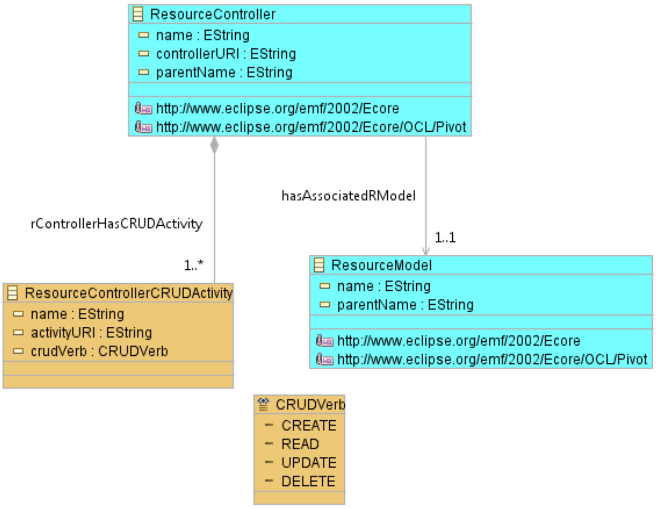


Figure 16 PIM meta-model simplified diagram: ResourceController element

Relations

Table 10 ResourceController's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
ResourceModel	association	1	The *ResourceController* element must be associated with exactly one *ResourceModel*. The *ResoruceController* is the WEB API that interacts with clients by receiving and sending data modelled by the associated *ResourceModel.*
ResourceControllerCRUDActivity	composition	1..*	The *ResourceController* element must have at least one composition association with a *ResourceControllerCRUDActivity.* Since *ResourceControllerCRUDActivities* are the intended functions to handle client requests, absence of any of them would lead to inability of clients to interact with that specific resource.

Behavioural Restrictionss

This subsection lists all the behavioural restrictions of the properties and relations of a *ResourceController* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *rControllerHasUniqueCRUDActivities:* This OCL constraint checks whether every *ResourceController* element has at most one *ResourceControllerCRUDActivity* for each *CRUDVerb.* This means that there can be no duplicates. Every request with a specific *CRUDVerb* is handled by exactly one WEB API function.

- rControllerNotAllowedCRUDActivityVerbs: This constraint verifies that this ResourceController has no ResourceControllerCRUDActivity with the CRUDVerb "CREATE".

4.2.2.5 ResourceModelManager Element

Overview

The *ResourceModelManager* element is always coupled with a *ResourceModel* one. It has the responsibility to create new instances of its associated *ResourceModel* (like a factory pattern) as well as retrieve all of them and to send a list of their *HypermediaLinks* back to the client. Figure 17 demonstrates a simplified diagram of the properties and relations of a ResourceModelManager element.

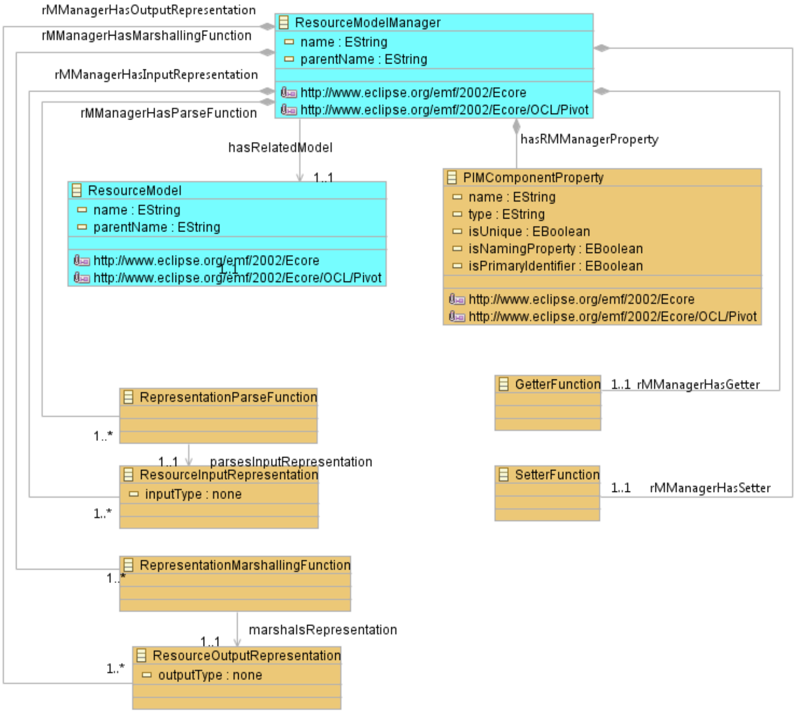

rMManagerHasOutputRepresentation
rMManagerHasMarshallingFunction
rMManagerHasInputRepresentation
rMManagerHasParseFunction
ResourceModelManager
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
hasRelatedModel
hasRMManagerProperty
1..1
ResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
PIMComponentProperty
name : EString
type : EString
isUnique : EBoolean
isNamingProperty : EBoolean
isPrimaryIdentifier : EBoolean
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
RepresentationParseFunction
1..*
GetterFunction
1..1 rMManagerHasGetter
1..1
parsesInputRepresentation
ResourceInputRepresentation
inputType : none
1..*
SetterFunction
1..1 rMManagerHasSetter
RepresentationMarshallingFunction
1..*
marshalsRepresentation
1..1
ResourceOutputRepresentation
outputType : none
1..*

Figure 17 PIM meta-model simplified diagram: ResourceModelManager element

Properties

Table 11 ResourceModelManager's Properties

Name	Type	Multiplicity	Explanation
Name	EString	1	This is the name of a *ResourceModelManager* element

parentName	EString	1	This is the name of the parent *CIM Resource* of this *ResourceModelManager.*

Relations

Table 12 ResourceModelManager’s Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
ResourceModel	association	1	The *ResourceModelManager* element must have exactly one associated *ResourceModel.* This association models the coupling between instances of a specific *ResourceModel* and its `managing’ *ResourceModelManager.*
PIMComponentProperty	composition	1	The *ResourceModelManager element must have exactly one PIMComponentProperty*. This property is always named `linklist’ and is used to send back to the client a list of *HypermediaLinks* with all the possible next actions.
SetterFunction	composition	1	The *ResourceModelManager* element must have exactly one *SetterFunction* that will update the value of its unique `linklist’ *PIMComponentProperty*.
GetterFunction	composition	1	The *ResourceModelManager element must have exactly one GetterFunction that will retrieve the value of its unique `linklist’ PIMComponentProperty*.
ResourceInputRepresentation	composition	1..*	The *ResourceModelManager* element must be associated with at least one *ResourceInputRepresentation* element because there must be defined at least one input media type in which clients will be able to send data modelled by the associated *ResourceModel* to the envisioned service.
RepresentationParseFunction	composition	1..*	The *ResourceModelManager* element must be associated with at least one *RepresentationParseFunction* since it is associated with at least one *ResourceInputRepresentation* and therefore at least one function with parsing capabilities is needed.
ResourceOutputRepresentation	composition	1..*	The *ResourceModelManager* element must be associated with at least one *ResourceOutputRepresentation* element because there must be defined at least one input media type in which the envisioned service will send data modelled by the associated *ResourceModel* to any of its clients.

ResourceMarshalingFunction	composition	1..*	The *ResourceModelManager* element must be associated with at least one *RepresentationMarshallingFunction* since it is associated with at least one *ResourceOutputRepresentation* and therefore at least one function with marshalling capabilities is needed.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *ResourceModelManager* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *existentSettersForAllProperties:* This OCL constraint checks whether there exists exactly one *SetterFunction* for every *PIMComponentProperty* a *ResourceModelManager* has.

- *existentGettersForAllProperties:* This constraint checks whether there exists exactly one *GetterFunction* for every *PIMComponentProeprty* a ResourceModelManager has.

- *uniqueLinklistProperty:* This constraint verifies that every *ResourceModelManager* contains exactly one property with the name *`linklist'.*

4.2.2.6 ResourceControllerManager Element

Overview

The *ResourceControllerManager* element models the web interface of a *ResourceModelManager*. This happens in respect of the REST architectural style, which means that every *ResourceControllerManager* is assigned a unique *URI* and the *CRUDVerbs* are used as intended when accessing that *URI*. Figure 18 demonstrates a simplifed diagram of the ResourceControllerManager element with its properties and relations.

Properties

Table 13 ResourceControllerManager's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *ResourceControllerManager.*
parentName	EString	1	This is the name of the parent *CIM Resource* of this *ResourceControllerManager.*
controllerURI	EString	1	This EString contains the *URI* that is automatically assigned to every *ResourceControllerManager.*

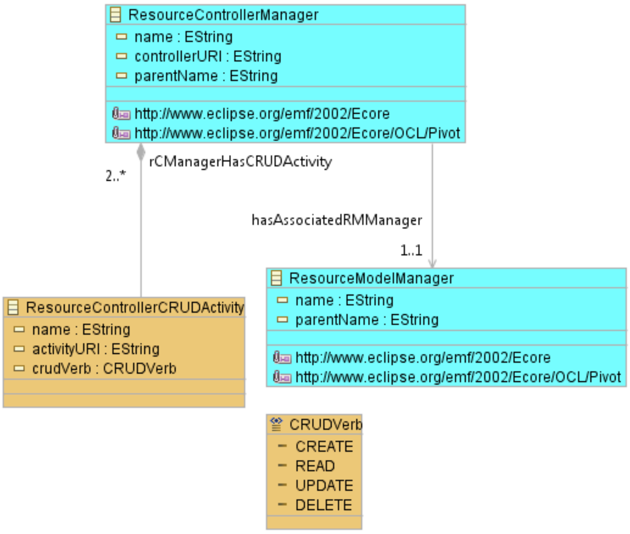


Figure 18 PIM meta-model simplified diagram: ResourceControllerManager element

Relations

Table 14 ResourceControllerManager's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
ResourceModelManager	association	1	The *ResourceControllerManager* element must be associated with exactly one *ResourceModelManager*. The *ResoruceControllerManager* is the WEB API that interacts with data modelled by the associated *ResourceModelManager's ResourceModel.*
ResourceControllerCRUDActivity	composition	2	The *ResourceControllerManager* element must have at least two *ResourceControllerCRUDActivities.* Since *ResourceControllerCRUDActivities* are the intended functions to handle client requests, absence of any of them would lead to inability of clients to interact with that specific resource.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *ResourceControllerManager* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *rCManagerHasUniqueCREATEActivity:* This OCL constraint checks whether a *ResourceControllerManager* has exactly one *ResourceControllerCRUDActivity* with *CRUDVerb* 'CREATE'.

- *rCManagerHasUniqueREADActivity:* This OCL constraint checks whether a *ResourceControllerManager* has exactly one *ResourceControllerCRUDActivity* with *CRUDVerb* 'READ'.

- *rCManagerNotAllowedCRUDActivityVerbs:* This OCL constraing checks whether a *ResourceControllerManager* element has only *ResourceControllerCRUDActivities* with the allowed *CRUDVerbs* and not any with 'UPDATE' or 'DELETE'.

4.2.2.7 AlgoResourceModel Element

Overview

AlgoResourceModel PIM meta-model elements model any resource that does not fall in the category of the ones modelled by *ResourceModels.* Such resources embed some sort of algorithm rather than pure data modelling as *ResourceModels* do. These models, along with their corresponding controllers, wrap as well, compositions of SOAP/REST third party services. Figure 19 demonstrates a simplified diagram of an *AlgoResourceModel* element with its properties and relations.

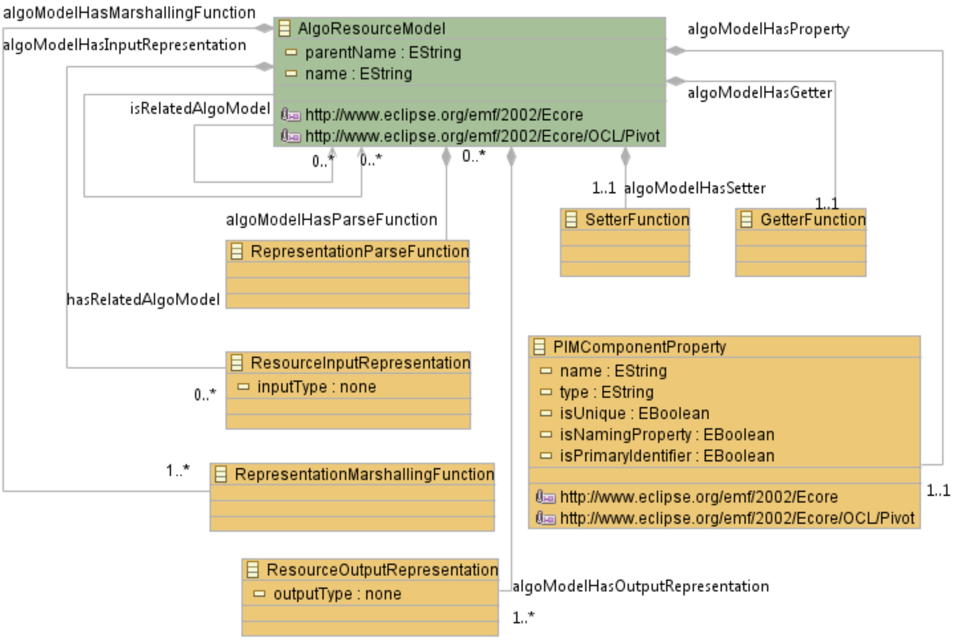

algoModelHasMarshallingFunction
algoModelHasInputRepresentation
AlgoResourceModel
parentName : EString
name : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
algoModelHasProperty
algoModelHasGetter
isRelatedAlgoModel
0..*
0..*
0..*
1..1
algoModelHasSetter
1..1
algoModelHasParseFunction
SetterFunction
GetterFunction
RepresentationParseFunction
hasRelatedAlgoModel
ResourceInputRepresentation
inputType : none
0..*
PIMComponentProperty
name : EString
type : EString
isUnique : EBoolean
isNamingProperty : EBoolean
isPrimaryIdentifier : EBoolean
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
1..*
RepresentationMarshallingFunction
1..1
ResourceOutputRepresentation
outputType : none
algoModelHasOutputRepresentation
1..*

Figure 19 PIM meta-model simplified diagram: AlgoResourceModel element

Properties

Table 15 AlgoResourceModel’s Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *AlgoResourceModel* element.
parentName	EString	1	This is the name of the parent *CIM Resource* of this *AlgoResourceModel.*

Relations

Table 16 AlgoResourceModel’s Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
AlgoResourceModel	association	0..*	The *AlgoResourceModel* may be associated with zero or more other *AlgoResourceModels* by the means of the association “hasRelatedAlgoModel”. This association models the fact that some *AlgoResourceModels* may have some related *AlgoResourceModels.*
AlgoResourceModel	association	0..*	The *AlgoResourceModel* may be associated with zero or more other *AlgoResourceModels* by the means of the association “isRelatedAlgoModel”. This association models the fact that some *AlgoResourceModels* may have the related *AlgoResourceModels* of zero or more other ones.
PIMComponentProperty	composition	1	The *AlgoResourceModel element* must have exactly one *PIMComponentProperty*. This property is always named `linklist’ and is used to send back to the client a list of *HypermediaLinks* with all the possible next actions.
SetterFunction	composition	1	The *AlgoResourceModel* element must have exactly one *SetterFunction* that will update the value of its unique `linklist’ *PIMComponentProperty*.
GetterFunction	composition	1	The *AlgoResourceModel element must have exactly one GetterFunction that will retrieve the value of its unique `linklist’ PIMComponentProperty*.
ResourceInputRepresentation	composition	1..*	The *AlgoResourceModel* element must be associated with at least one *ResourceInputRepresentation* element because there must be defined at least one input media type in which clients will be able to send data modelled by this Algo*ResourceModel* to the envisioned service.

RepresentationParseFunction	composition	1..*	The *AlgoResourceModel* element must be associated with at least one *RepresentationParseFunction* since it is associated with at least one *ResourceInputRepresentation* and therefore at least one function with parsing capabilities is needed.
ResourceOutputRepresentation	composition	1..*	The *AlgoResourceModel* element must be associated with at least one *ResourceOutputRepresentation* element because there must be defined at least one input media type in which the envisioned service will send data modelled by this Algo*ResourceModel* to any of its clients.
ResourceMarshalingFunction	composition	1..*	The *AlgoResourceModel* element must be associated with at least one *RepresentationMarshallingFunction* since it is associated with at least one *ResourceOutputRepresentation* and therefore at least one function with marshalling capabilities is needed.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of an *AlgoResourceModel* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *existentSettersForAllProperties:* This OCL constraint checks whether there exists exactly one *SetterFunction* for every *PIMComponentProperty* an *AlgoResourceModel* has.

- *existentGettersForAllProperties:* This constraint checks whether there exists exactly one *GetterFunction* for every *PIMComponentProeprty* an *AlgoResourceModel* has.

- *uniqueLinklistProperty:* This constraint verifies that every *AlgoResourceModel* contains exactly one *property with the name `linklist'.*

4.2.2.8 AlgoResourceController Element

Overview

The *AlgoResourceController* PIM meta-model element models the *WEB API* of an *AlgoResourceModel.* This happens in respect of the REST architectural style, which means that every Algo*ResourceController* is assigned a unique *URI* and the *CRUDVerbs* are used as intended when accessing that *URI*. Figure 20 demonstrates a simplified diagram of an *AlgoResourceController* with its properties and relations.

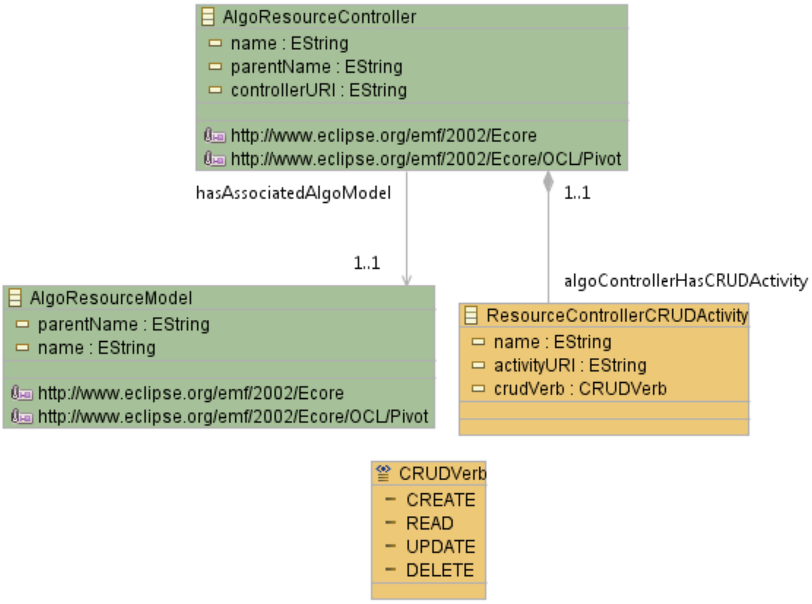


Figure 20 PIM meta-model simplified diagram: AlgoResourceController

Properties

Table 17 AlgoResourceController's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *AlgoResourceController* element.
parentName	EString	1	This is the name of the parent *CIM Resource* of this *AlgoResourceController.*
controllerURI	EString	1	This EString stores the unique *URI* that is assigned automatically to every *AlgoResourceController* element.

Relations

Table 18 AlgoResourceController's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
AlgoResourceModel	association	1	The *AlgoResourceController* element must be associated with exactly one *AlgoResourceModel*. The *AlgoResourceController* is the WEB API that interacts with data modelled by the associated *AlgoResourceModel.*
ResourceControllerCRUDActivity	composition	1	The *AlgoResourceController* element must have exactly one *ResourceControllerCRUDActivity*. Since

			ResourceControllerCRUDActivities are the intended functions to handle client requests, absence of any of them would lead to inability of clients to interact with that specific resource.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of an *AlgoResourceController* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *algoControllerNotAllowedCRUDVerbs:* This OCL constraint verifies that this *AlgoResourceController* has no *ResourceControllerCRUDActivities* with the *CRUDVerb* 'UPDATE' or 'DELETE'.

- *algoControllerHasUniqueCRUDActivity:* This OCL constraint checks whether this *AlgoResourceController* has exactly one *ResourceControllerCRUDActivity.*

4.2.2.9 PIMComponentProperty Element

Overview

The *PIMComponentProperty* PIM meta-model element models properties that instances of PIM meta-model elements may have. Such elements are the *ResourceModel,* the *ResourceModelManager* and the AlgoResourceModel. The rest of this subsection defines properties, structural relations and constraints as well as the behavioural constraints of a *PIMComponentProperty*.

Properties

Table 19 PIMComponentProperty's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *PIMComponentProperty* element
type	EString	1	This is the type of the *PIMComponentProperty* element
isUnique	EBoolean	1	This denotes the multiplicity of the *PIMComponentProperty.* If the EBoolean value is true then the property has multiplicity one. Otherwise there may be multiple instances of this property (array).
isNamingProperty	EBoolean	1	This denotes whether a *PIMComponentProperty* element is a naming property as well or not. This attribute is useful for the client in order to pick one specific *ResourceModel* instance within a list with

			many ones and retrieve it following the provided *HypermediaLink*.
isPrimaryIdentifier	EBoolean	1	This denotes whether a *PIMComponentProperty* element is a primary identifier or not. This attribute is used to calculate the URIs within the *HypermediaLinks* that are sent to clients as well as to identify specific records in the local envisioned service database.

Relations

Table 20 PIMComponentProperty's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
RDBMSColumn	association	0..1	The *PIMComponentProperty* element may be associated with zero or one *RDBMSColumn* elements. All the *PIMComponentProperties* of *ResourceModels* that describe resource data have exactly one associated *RDBMSColumn. The restPIMComponentProperties*, such as the mandatory helper `linklist' one, do not have associated *RDBMSColumn*.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *PIMComponentProperty* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *primaryIdentifierMapsToPrimaryKey:* This OCL constraint verifies that every *PIMComponentProperty* element that is a primary identifier is always mapped to the according primary key column within the relational database of the envisioned service.

4.2.2.10 PIMComponentFunction Element

Overview

The *PIMComponentFunction* PIM meta-model element is an abstract model of some PIM component function. It brings together common properties that are shared among other more specific function classes within the PIM meta-model.

Properties

Table 21 PIMComponentFunction's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *PIMComponentFunction.* It will be used as a function name at the code generation phase.

Relations

Table 22 PIMComponentFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
FunctionParameter	composition	0..*	The *PIMComponentFunction* element may have zero or more associated *FunctionParameters.*

Behavioural Restrictions

The *PIMComponentFunction* element has no behavioural constraints.

4.2.2.11 SetterFunction Element

Overview

The *SetterFunction* PIM meta-model element models a PIM component function that is dedicated in updating a specific PIM component property. It is a specialization of the *PIMComponentFunction* element.

Properties

The *SetterFunction* element has no additional properties other than those that are inherited from the *PIMComponentFunction* element.

Relations

Table 23 SetterFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
PIMComponentProperty	association	1	The *SetterFunction* element must be associated with exactly one *PIMComponentProperty* element. This association models the fact that this *SetterFunction* intention is to update the value of the unique associated *PIMComponentProperty.* Therefore, it has a unique *FuntionParameter* that has the same name, type and multiplicity with the *PIMComponentProperty* and the *bIsReturnParameter* EBoolean attribute set to false.

Behavioural Restrictions

The *SetterFunction* element has no behavioural constraints.

4.2.2.12 GetterFunction Element

Overview

The *GetterFunction* PIM meta-model element models a PIM component function that is dedicated in retrieving the value of a specific PIM component property. It is a specialization of the *PIMComponentFunction* element.

Properties

The *GetterFunction* element has no additional properties other than those that are inherited from the *PIMComponentFunction* element.

Relations

Table 24 GetterFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
PIMComponentProperty	association	1	The *GetterFunction* element must be associated with exactly one *PIMComponentProperty* element. This association models the fact that this *GetterFunction* intention is to retrieve the value of the unique associated *PIMComponentProperty.* Therefore, it has a unique *FuntionParameter* that has the same name, type and multiplicity with the *PIMComponentProperty* and the *bIsReturnParameter* EBoolean attribute set to true.

Behavioural Restrictions

The *GetterFunction* element has no behavioural restrictions.

4.2.2.13 FunctionParameter Element

Overview

The *FunctionParameter* PIM meta-model element models any parameter of a *PIMComponentFunction.*

Properties

Table 25 FunctionParameter's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *FunctionParameter* element.
type	EString	1	This is the type of the *FunctionParameter* element.
isUnique	EBoolean	1	This denotes the multiplicity of the

			FuntionParameter. If the EBoolean value is true then the parameter has multiplicity one. Otherwise there may be multiple instances of this parameter (array).
bIsReturnParameter	EBoolean	1	This denotes whether a *FunctionParameter* element is a *PIMComponentFunction* return parameter or an argument. If it has the true value then it is a return parameter. Otherwise, it is an argument.

Relations

The *FunctionParameter* element has no relations with other PIM meta-model elements.

Behavioural Restrictions

The *FunctionParameter* element has no behavioural constraints.

4.2.2.14 ResourceOutputRepresentation Element

Overview

The *ResourceOutputRepresentation* PIM meta-model element models any output representation that may be used to format data that is sent within request responses to clients.

Properties

Table 26 ResourceOutputRepresentation's Properties

Name	Type	Multiplicity	Explanation
inputType	MediaType	1	This attribute denotes the media type of a *ResourceOutputRepresentation.* As it is defined in the MediaType S-CASE data type definition, the possible options are *JSON* and *XML.*

Relations

The *ResourceOutputRepresentation* does not have any relations with other PIM meta-mode elements.

Behavioural Restrictions

The *ResourceOutputRepresentation* element does not have behavioural constraints.

4.2.2.15 ResourceInputRepresentation Element

Overview

The *ResourceInputRepresentation* PIM meta-model element models any input representation that may be used to format data that is sent from clients.

Properties

Table 27 ResourceInputRepresentation's Properties

Name	Type	Multiplicity	Explanation
outputType	MediaType	1	This attribute denotes the media type of a *ResourceInputRepresentation.* As it is defined in the MediaType S-CASE data type definition, the possible options are *JSON* and *XML.*

Relations

The *ResourceInputRepresentation* does not have any relations with other PIM meta-mode elements.

Behavioural Restrictions

The *ResourceInputRepresentation* element does not have behavioural constraints.

4.2.2.16 RepresentationParseFunction Element

Overview

The *RepresentationParseFunction PIM meta-model* element models *ResourceModel, ResourceModelManager* and *AlgoResourceModel* functions that parse one of their specific *ResourceInputRepresentation*.

Properties

The *RepresentationParseFunction* element does not have any properties.

Relations

Table 28 RepresentationParseFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
ResourceInputRepresentation	association	1	The *RepresentationParseFunction* element must have exactly one associated *ResourceInputRepresentation.* This association denotes the fact that every parse function is

			able to parse exactly one input media type.

Behavioural Restrictions

The *RepresentationParseFunction* has no behavioural restrictions.

4.2.2.17 RepresentationMarshallingFunction Element

Overview

The *RepresentationMarshallingFunction* PIM meta-model element models *ResourceModel, ResourceModelManager* and *AlgoResourceModel* functions that marshal one of their specific *ResourceOutputRepresentation*.

Properties

The *RepresentationMarshallingFunction* element does not have any properties.

Relations

Table 29 RepresentationMarshallingFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
ResourceOutputRepresentation	association	1	The *RepresentationMarshallingFunction* element must have exactly one associated *ResourceOutputRepresentation.* This association denotes the fact that every marshalling function is able to marshal exactly one output media type.

Behavioural Restrictions

The *RepresentationMarshallingFunction* has no behavioural restrictions.

4.2.2.18 ResourceControllerCRUDActivity Element

Overview

The *ResourceControllerCRUDActivity* PIM meta-model element models a component of some controller-type element's web *API.* That is, every *ResourceControllerCRUDActivity* is dedicated to handle client requests of exactly one specific *CRUDVerb* either of a *ResourceController, AlgoResourceController or ResourceControllerManager* web *API.*

Properties

Table 30 ResourceControllerCRUDActivity Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *ResoruceControllerCRUDActivity* element
activityURI	EString	1	This is the *URI* of the *ResourceControllerCRUDActivity.* It is relative to the according controller-type element URI. Together, along with the services URL form the full URI at which a client should send its requests.
crudVerb	CRUDVerb	1	This denotes the specific *CRUDVerb* requests that this specific *ResourceControllerCRUDActivity* accepts. As it is defined at *CRUDVerb* S-CASE data type definition, the four possible opionts are *CREATE, READ, UPDATE* and *DELETE.*

Relations

Table 31 ResourceControllerCRUDActivity's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
CRUDActivityHandler	composition	1	The *ResourceControllerCRUDActivity* element must have exactly one association with a *CRUDActivityHandler element.*
RDBMSActivity	assotiation	0..1	The *ResourceControllerCRUDActivity* element may have zero or one association with an *RDBMSActivity.* More specifically, the *ResourceControllerCRUDActivities* of *ResourceControllers* and *ResourceControllerManagers* have exactly one associated *RDBMSActivity,* whilst the ones of *AlgoResourceControllers* zero.

Behavioural Restrictions

The *ResourceControllerCRUDActivity* element has no behavioural constraints.

4.2.2.19 CRUDActivityHandler Element

Overview

The *CRUDActivityHandler* PIM meta-model element models the actual implementation of the required actions that need to be taken in order to serve a client request that is forwarded to it by its overlying *ResourceControllerCRUDActivity.* Usually, these include some sort of interaction with the local envisioned service database and then the generation of the list with *HypermediaLinks* that will be sent back to client alongside its request response.

Properties

Table 32 CRUDActivityHandler's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *CRUDActivityHandler* element.
crudVerb	CRUDVerb	1	This is the *CRUDVerb* type of requests that his *CRUDActivityHandler* element handles.

Relations

Table 33 CRUDActivityHandler's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
DatabaseController	association	1	The *CRUDActivityHandler* element must have exactly one association with a *DatabaseController* one. Actually, this association denotes the fact that the *CRUDActivityHandler* uses the *DatabaseController* to access the local relational schema.
CreateHypermediaFunction	composition	1	The *CRUDActivityHandler* element must have exactly one *CreateHypermediaFunction* element.

Behavioural Restrictions

The *CRUDActivityHandler* element has no behavioural constraints.

4.2.2.20 CreateHypermediaFunction Element

Overview

The *CreateHypermediaFunction* PIM meta-model element models the function of each overlying *CRUDActivityHandler*, which creates the full list of hypermedia links to be sent back to the client.

Properties

The *CreateHypermediaFunction* element has no properties.

Relations

Table 34 CreateHypermediaFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
HypermediaLink	composition	1..*	The *CreateHypermediaFunction* element may be associated with one or more *HypermeidaLinks.* In fact, this association denotes that the *CreateHypermediFunction* element creates the associated hypermedia links.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *CreateHypermediaFunction* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

- *algoControllerActivityAddsHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction of the AlgoResourceController's* unique *ResourceControllerCRUDActivity,* of some allowed *CRUDVerb* adds exactly one *HypermediaLink* with the same *CRUDVerb, LinkType* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerCreateActivityAddsCreateHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb CREATE*, adds exactly one *HypermediaLink* with the *CREATE CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerReadActivityAddsCreateHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *CREATE CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerCreateActivityAddsReadHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb CREATE*, adds exactly one *HypermediaLink* with the *READ CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerReadActivityAddsReadHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *READ CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerUpdateActivityAddsUpdateHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb UPDATE*, adds exactly one *HypermediaLink* with the *UPDATE CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerReadActivityAddsUpdateHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *UPDATE CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerReadActivityAddsDeleteHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *DELETE CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerUpdateActivityAddsReadHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb UPDATE*, adds exactly one *HypermediaLink* with the *READ CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerReadActivityAddsReadHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *READ CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerUpdateActivityAddsDeleteHypermediaLinkToSelf:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb UPDATE*, adds exactly one *HypermediaLink* with the *DELETE CRUDVerb* of type *SIBLING* and target controller being itself, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerCreateActivityAddsReadHypermediaLinkToRRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb CREATE*, adds exactly one *HypermediaLink* with the *READ CRUDVerb* of type *CHILD* and target controller being the related *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerCreateActivityAddsUpdateHypermediaLinkToRRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb CREATE*, adds exactly one *HypermediaLink* with the *UPDATE CRUDVerb* of type *CHILD* and target controller being the related *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerCreateActivityAddsDeleteHypermediaLinkToRRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb CREATE*, adds exactly one *HypermediaLink* with the *DELETE CRUDVerb* of type *CHILD* and target controller being the related *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerReadActivityAddsReadHypermediaLinkToRRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *READ CRUDVerb* of type *CHILD* and target controller being the related *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerReadActivityAddsUpdateHypermediaLinkToRRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *UPDATE CRUDVerb* of type *CHILD* and target controller being the related *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerReadActivityAddsDeleteHypermediaLinkToRRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *DELETE CRUDVerb* of type *CHILD* and target controller being the related *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client.

- *rCManagerCreateActivityAddsReadHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb CREATE*, adds exactly one *HypermediaLink* with the *READ CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

- *rCManagerCreateActivityAddsUpdateHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb CREATE*, adds exactly one *HypermediaLink* with the *UPDATE CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

- *rCManagerCreateActivityAddsDeleteHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb CREATE*, adds exactly one *HypermediaLink* with the *DELETE CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

- *rCManagerReadActivityAddsReadHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *READ CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

- *rCManagerReadActivityAddsUpdateHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *UPDATE CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

- *rCManagerReadActivityAddsDeleteHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceControllerManager's* unique *ResourceControllerCRUDActivity* with

CRUDVerb READ, adds exactly one *HypermediaLink* with the *DELETE CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

- *rControllerReadActivityAddsCreateHypermediaLinkToRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *CREATE CRUDVerb,* of type *CHILD* and target controller being a related *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every related *ResourceControllerManager.*

- *rControllerReadActivityAddsReadHypermediaLinkToRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *READ CRUDVerb,* of type *CHILD* and target controller being a related *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every related *ResourceControllerManager.*

- *rControllerUpdateActivityAddsCreateHypermediaLinkToRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb UPDATE*, adds exactly one *HypermediaLink* with the *CREATE CRUDVerb,* of type *CHILD* and target controller being a related *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every related *ResourceControllerManager.*

- *rControllerUpdateActivityAddsReadHypermediaLinkToRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb UPDATE*, adds exactly one *HypermediaLink* with the *READ CRUDVerb,* of type *CHILD* and target controller being a related *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every related *ResourceControllerManager.*

- *rControllerReadActivityAddsCreateHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *CREATE CRUDVerb,* of type *PARENT* and target controller being the parent *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerReadActivityAddsReadHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *READ CRUDVerb,* of type *PARENT* and target controller being the parent *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerUpdateActivityAddsCreateHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb UPDATE*, adds exactly one *HypermediaLink* with the *CREATE CRUDVerb,* of type *PARENT* and target controller being the parent *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client.
- *rControllerUpdateActivityAddsReadHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb UPDATE*, adds exactly one *HypermediaLink* with the *READ CRUDVerb,* of type *PARENT* and target controller being the parent *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerDeleteActivityAddsCreateHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb DELETE*, adds exactly one *HypermediaLink* with the *CREATE CRUDVerb,* of type *PARENT* and target controller being the parent *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client.

- *rControllerDeleteActivityAddsReadHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb DELETE*, adds exactly one *HypermediaLink* with the *READ CRUDVerb,* of type *PARENT* and target controller being the parent *ResourceControllerManager* element, to the list of *HypermediaLinks* to be returned to the client.

- *algoControllerActivityAddsCreateHypermediaLinkToRAlgoController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the Algo*ResourceController's* unique *ResourceControllerCRUDActivity* with an allowed *CRUDVerb*, adds exactly one *HypermediaLink* with the *allowed CRUDVerb,* of type *CHILD* and target controller being a related Algo*ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every related *AlgoResourceController.*

- *algoControllerActivityAddsCreateHypermediaLinkToParentAlgoController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the Algo*ResourceController's* unique *ResourceControllerCRUDActivity* with an allowed *CRUDVerb*, adds exactly one *HypermediaLink* with the *allowed CRUDVerb,* of type *PARENT* and target controller being a parent Algo*ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *AlgoResourceController.*

- *rControllerReadAddsHypermediaLinkToRAlgoController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb READ*, adds exactly one *HypermediaLink* with the *allowed CRUDVerb,* of type *CHILD* and target controller being a related *AlgoResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every related *AlgoResourceController.*

- *rControllerUpdateAddsHypermediaLinkToRAlgoController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the *ResourceController's* unique *ResourceControllerCRUDActivity* with *CRUDVerb UPDATE*, adds exactly one *HypermediaLink* with the *allowed CRUDVerb,* of type *CHILD* and target controller being a related *AlgoResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every related *AlgoResourceController.*

- *algoControllerAddsReadHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the Algo*ResourceController's* unique *ResourceControllerCRUDActivity* with an allowed *CRUDVerb*, adds exactly one *HypermediaLink* with the *READ CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

- *algoControllerAddsUpdateHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the Algo*ResourceController's* unique *ResourceControllerCRUDActivity* with an allowed *CRUDVerb*, adds exactly one *HypermediaLink* with the *UPDATE CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

- *algoControllerAddsDeleteHypermediaLinkToParentRController:* This OCL constraint verifies that the *CreateHypermediaFunction* of the Algo*ResourceController's* unique *ResourceControllerCRUDActivity* with an allowed *CRUDVerb*, adds exactly one *HypermediaLink* with the *DELETE CRUDVerb,* of type *PARENT* and target controller being a parent *ResourceController* element, to the list of *HypermediaLinks* to be returned to the client. This is checked against every parent *ResourceController.*

4.2.2.21 HypermediaLink Element

Overview

The *HypermediaLink* PIM meta-model element models the hypermedia links or HATEOAS of the REST architectural style.

Properties

Table 35 HypermediaLink's Properties

Name	Type	Multiplicity	Explanation
linkCRUDVerb	CRUDVerb	1	Every *HypermediaLink* that is sent to clients, among others includes the required *CRUDVerb* to be used to access the target resource in a new client request.
linkType	LinkType	1	This denotes the relationship between the resource that responses with this hypermedia link (source) and the target resource. If the value of 'linkType' is *SIBLING* then the *HypermediaLink* points to another available action of the same source resource. If the value is *CHILD* then it points to an available action of some target resource that is related resource of the source resource. Finally, if the value of the 'linkType' is *PARENT* then the link points to an available action of some target resource of which this source resource is related.

Relations

Table 36 HypermediaLink's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
AlgoResourceController	association	0..1	The *HypermediaLink* may be associated with zero or one *AlgoResourceControllers*. If this link points the client to an *AlgoResourceController* then it is associated with exactly one of them. Otherwise, it is associated with zero of them.
ResourceController	association	0..1	The *HypermediaLink* may be associated with zero or one *ResourceControllers*. If this link points the client to a *ResourceController* then it is associated with exactly one of them. Otherwise, it is associated with zero of them.
ResourceControllerManager	association	0..1	The *HypermediaLink* may be associated with zero or one *ResourceControllerManager*. If this link points the client to a *ResourceControllerManager* then it is associated with exactly one of them. Otherwise, it is associated with zero of them.

Behavioural Restrictions

The *HypermediaLink* element has no behavioural constraints.

4.2.2.22 DatabaseController Element

Overview

The *DatabaseController* PIM meta-model element brings together all the *RDBMSActivities* that are needed by the envisioned service for all its interactions with its local RDBMS database.

Properties

Table 37 DatabaseController's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *DatabaseController* element.

Relations

Table 38 DatabaseController's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
RDBMSActivity	composition	1..*	The *DatabaseController element may be associated with one ore more RDBMSActivities.*

Behavioural Restrictions

The *DatabaseController* element has no behavioural constraints.

4.2.2.23 RDBMSActivity Element

Overview

The *RDBMSActivity* PIM meta-model element models one function of the overlying *DatabaseController* element which is dedicated in performing exactly one type of interaction with exactly one table of the envisioned service local database.

Properties

Table 39 RDBMSActivity's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *RDBMSActivity* element.
rdbmsVerb	RDBMSVerb	1	The *RDBMSVerb* of an *RDBMSActivity* denotes the type of action that it is intended to perform to the envisioned system's local database. If it has the value *INSERT* then it creates a new row in a database table. If it has the value *SELECT* then it either retrieves a specific entry or a list from a database table. If it has the value *UPDATE* then it updates the content of a database table row. Finally, if it has the value *DELETE* then it performs a *CASCADE DELETE* action in the local database.

Relations

Table 40 RDBMSActivity's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
RDBMSTable	association	1	The *RDBMSActivity* must be associated with exactly one *RDBMSTable.* This association denotes the fact that every *RDBMSActivity* alters exactly one *RDBMSTable.* The only case where more tables may be altered is the one of a *CASCADE DELETE.*

Behavioural Restrictions

The *RDBMSActivity* element has no behavioural constraints.

4.2.2.24 RDBMSTable Element

Overview

The *RDBMSTable* PIM meta-model element model one relational database table.

Properties

Table 41 RDBMSTable's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *RDBMSTable.* It is used to define the relational schema.

Relations

Table 42 RDBMSTable's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
RDBMSColumn	composition	2..*	The *RDBMSTable* element may be associated with two or more *RDBMSColumn* elements. This denotes the fact that any *RDBMSTable* must have at least a *primary key* column as well as at least one other column to store some sort of data.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of an *RDBMSTable* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PIM meta-model that can be found in appendix A.1.

uniquePrimaryKey: This OCL constraint verifies that every *RDBMSTable* has exactly one *RDBMSColumn* that is primary key.

4.2.2.25 RDBMSTableColumn Element

Overview

The *RDBMSTableColumn* PIM meta-model element models a column of a relational database table.

Properties

Table 43 RDBMSTableColumn's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *RDBMSTableColumn* element. It is used to define the envisioned system's local database.
type	EString	1	This is the type of the *RDBMSTableColumn* element. It is used to define the envisioned system's local databse.
isForeignKey	EBoolean	1	This EBoolean denotes whether one *RDBMSTableColumn* is a foreign key or not. If it is true then it is a foreign key. Otherwise, it is not a foreign key.
isPrimaryKey	EBoolean	1	This EBoolean denotes whether one RDBMSTableColumn is a primary key or not. If its value is true then it is a primary key. Otherwise, it is not a primary key.

Relations

Table 44 RDBMSTableColumn's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
RDBMSTable	association	0..*	The *RDBMSTableColumn* element may be associated with zero or more *RDBMSTables.* This association denotes that if an *RDBMSTableColumn* is a foreign key then it must reference some *RDBMSTable.* Otherwise, it does not reference any *RDBMSTable.*

Behavioural Restrictions

The *RDBMSTableColumn* element has not behavioural constraints.

5 Ecore PSM Meta-Model

5.1 PSM Meta-model Profile Design Goals

This section relates the underlying definition of the PSM meta-model with the design goals and constraints imposed by the S-CASE requirements and specifications as they are defined in *D1.3.1 S-CASE technical specification* as well as in *D1.1 S-CASE requirements* deliverables. Afterwards, this section provides the PSM meta-model design rational so as to satisfy the aforementioned goals and constraints.

Design Goals and Constraints

The S-CASE requirements that are related with the PSM meta-model, which can be found in the *D1.3.1* deliverable, are a subset of the PSM module requirements. In summary the functional and non-functional requirements, which the design of the PSM meta-model must satisfy, are the following:

1. The PSM meta-model should enable storage and retrieval of web service data in a uniform way.
2. The S-CASE system must produce systems that follow the REST architectural style.
3. The S-CASE system must produce systems that embed web service API technology (at least JAX-RS 2.0) in order to embrace open source solutions.
4. The S-CASE system must produce systems that embed XML Schema – object-mapping technology (at least JAXB) in order to embrace cloud and open source solutions.
5. The S-CASE system must produce systems that embed the HTTP 1.1 protocol in order to embrace cloud and open source solutions.
6. The S-CASE system must produce systems that embed object- relational mapping technology at least JPA) in order to embrace open source solutions.
7. The S-CASE system must produce systems that embed composition of other RESTFul or SOAP web services if needed.

PSM Meta-model Design Rational

The Ecore PSM meta-model that is defined in this document closely follows the design introduced by the PIM meta-model so as to ensure the production of RESTful services as it is explained in section 4, but it further enriches it with specific technologies and patterns that realize the abstract design of PIM by using concrete technologies. Due to such specialization, the PSM meta-model, models a web service resource as a two-part component entity and each of the two parts has additional constraints in comparison with their PIM meta-model counterparts.

The variation of MVC presented in section 4, the *Model Representation Controller* (MRC) one, in the PSM meta-model case also comprises of one *Model* element and one *Controller* but has no *Representation* elements as figure 21 demonstrates. This is because the JAXB framework is used to annotate the *Model* element instead, as S-CASE requirements require. Thus, the third party JAXB library performs the XML schema-object mapping of the *Model* element data to/from the desired media type such as *JSON* or *XML.* Moreover, the definition of the PSM meta-model enforces the annotation of the *Model* element with JPA standard annotations as well. Using this framework the relational database schema introduced in the PIM is transformed to the according SQL one. The available SQL schemas are thus, the whole family that JPA supports.

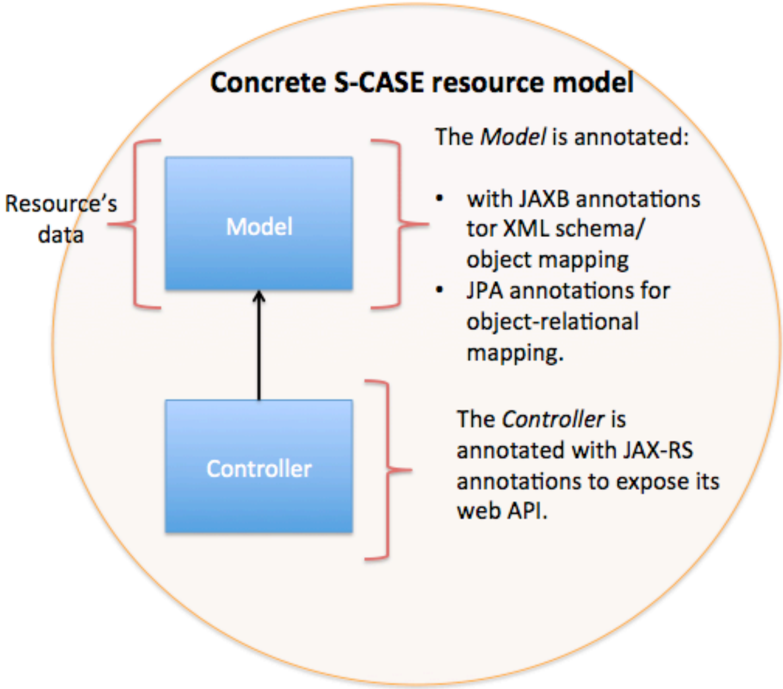

Concrete S-CASE resource model
Resource's data
Model
Controller
The *Model* is annotated:
• with JAXB annotations tor XML schema/ object mapping
• JPA annotations for object-relational mapping.
The *Controller* is annotated with JAX-RS annotations to expose its web API.

Figure 21 Concrete Resource Model

The *Controller* element represents, likewise the PIM case, the web API of the resource. Every such controller retains the unique URI that is assigned during PIM formulation. Additionally, each *Controller* element has JAX-RS framework annotations. These annotations expose the web API of the resource to the network by means of the underlying servlet such as *Jetty or Tomcat.* Through this framework, the systems that S-CASE produces are capable to interact with their clients using the HTTP 1.1 protocol.

5.2 PSM Ecore Meta-model Definition

5.2.1 Introduction

In order to fully define the PSM meta-model, various aspects of it must be explained, demonstrated as well as documented. Each of the following subsections of section 5 precisely define the structure of each PSM meta-model element, its properties and their behavioural constraints. Each subsection has the same format as the one defined in 4.2.1 for the PIM meta-model elements definition. Moreover, the full PSM meta-model diagram can be found at https://github.com/s-case/mde due to space limitations.

5.2.2 PSM Meta-model Elements

This section lists all the PSM meta-model elements defined using the format presented in subsection 4.2.1.

5.2.2.1 PSM Meta-model Custom Data Types

This subsection defines the custom data types that PSM meta-model elements use in their definitions.

MediaType

The *MediaType* data type is an enumeration, which models all the possible input and output media types that are supported in any case by S-CASE. These can be found in the following table:

Table 45 Supported Media Types

MediaType	Relative HTTP Header value	Default Value	Explanation
JSON	application/JSON	Yes	When JSON media type is selected for a resource the related WEB API expects as input and or outputs the data in application/JSON format
XML	application/XML	No	When XML media type is selected for a resource the related WEB API expects as input and or outputs the data in application/XML format

HTTPVerb

The *HTTPVerb* data type is an enumeration, which models the four HTTP verbs, *Post, Get, Put, Delete*. They are defined in the following table:

Table 46 CRUDVerb Data Type

HTTPVerb	Default Value	Explanation
POST	Yes	The POST verb is used to model the HTTP POST verb within the PSM meta-model. Wherever it is used, it respects REST's architectural style and thus denotes the action of creating a new resource.
GET	No	The GET verb is used to model the HTTP GET verb within the PSM meta-model. Wherever it is used, it respects

		REST's architectural style and thus denotes the action of retrieving either a specific existing resource or a list of existing resources.
PUT	No	The PUT verb is used to model the HTTP PUT verb within the PSM meta-model. Wherever it is used, it respects REST's architectural style and thus denotes the action of updating a specific existing resource. Moreover, since S-CASE produces web services that automatically create resource identifiers, the PUT verb is not used to also create resources.
DELETE	No	The DELETE verb is used to model the HTTP DELETE verb within the PSM meta-model. Wherever it is used, it respects REST's architectural style and thus denotes the action of deleting a specific existing resource as well as any subsequence related resources of it.

LinkType

The *LinkType* data type is an enumeration, which models the three possible hypermedia link types supported by S-CASE. These are the *PARENT, SIBLING* and *CHILD* ones. They are defined in the following table:

Table 47 CRUDVerb Data Type

LinkType	Default Value	Explanation
PARENT	Yes	Whenever a PSM meta-model *Hypermedia* element has a property with the *PARENT LinkType* it models a link that connects a specific resource A with a possible action at one of the service's resources that have A as related resource.
SIBLING	No	Whenever a PSM meta-model *Hypermedia* element has a property with the *SIBLING LinkType* it models a link that connects a specific resource with a possible action of it self.
CHILD	No	Whenever a PSM meta-model *Hypermedia* element has a property with the *CHILD LinkType* it models a link that connects a specific resource A with a possible action of one of the A's related resources.

5.2.2.2 RESTfulServicePSM Element

Overview

RESTfulServicePSM is the root element of the PSM meta-model. There may exist exactly one *RESTfulServicePSM* element in any PSM produced by S-CASE. It comprises the technologically specialized form of all the elements that form a RESTful service and is associated with them by means of a composition association. This suggests that any other element of PSM must be contained by this specific root element. Its properties, relations and their structural and behavioural restrictions are described in the rest of this subsection. Figure 22 demonstrates a simplified diagram with its properties and relations.

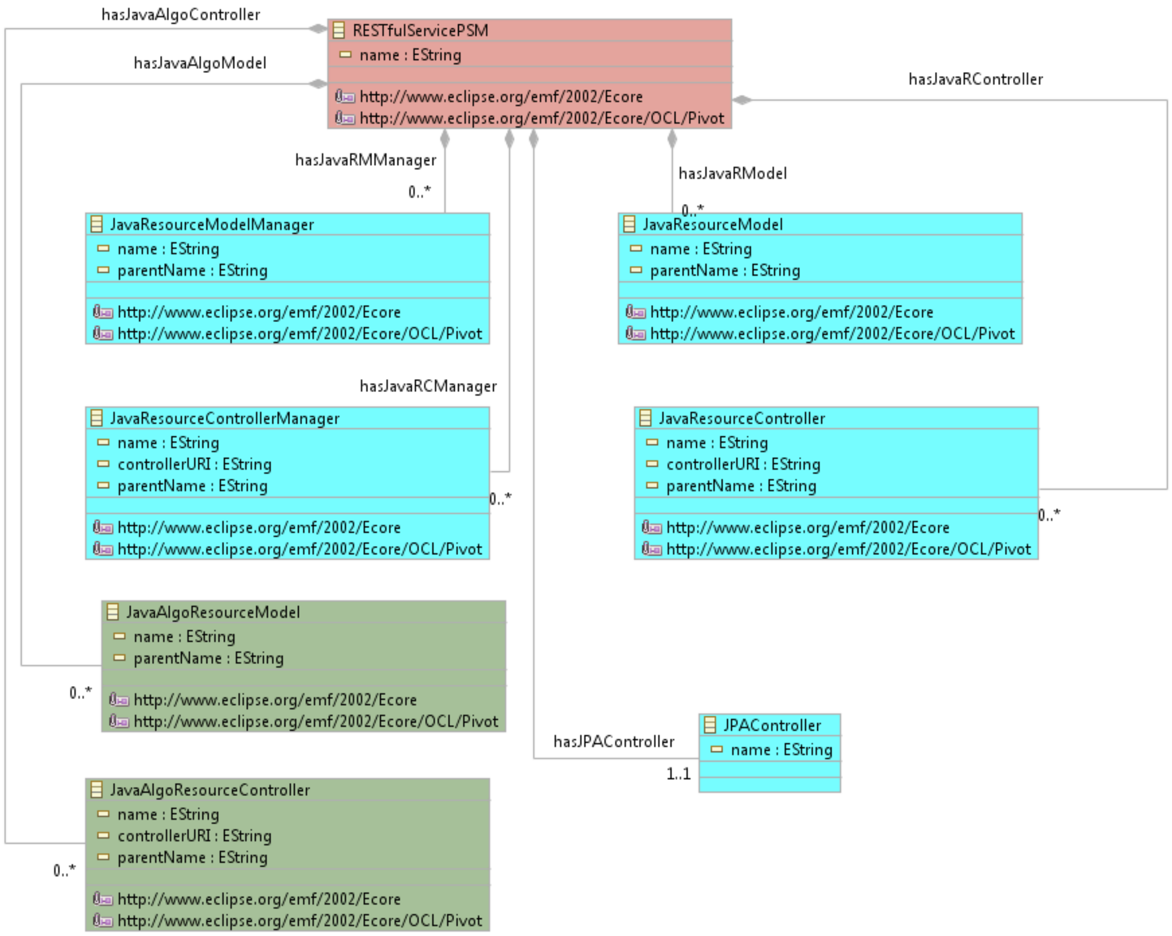

hasJavaAlgoController
RESTfulServicePSM
name : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
hasJavaAlgoModel
hasJavaRController
hasJavaRMManager
0..*
hasJavaRModel
0..*
JavaResourceModelManager
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
hasJavaRCManager
JavaResourceControllerManager
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*
JavaResourceController
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*
JavaAlgoResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*
hasJPAController
JPAController
name : EString
1..1
JavaAlgoResourceController
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*

Figure 22 PSM meta-model simplified diagram: RESTfulServicePSM element

Properties

Table 48 RESTfulServicePSM's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the provided service name by the S-CASE developer. Among others, it is used to produce the service folder structure and build files.

Relations

Table 49 RESTfulServicePIM's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JavaResourceModel	composition	0..*	The *RESTfulServicePSM* may contain zero or more *JavaResourceModels.* Any *JavaResourceModel* that exists in PSM belongs in the list oft he associated *JavaResourceModels* of the *RESTfulServicePSM element.*
JavaResourceController	composition	0..*	The *RESTfulServicePSM* may contain zero or more *JavaResourceController* elements. Any *JavaResourceController* that exists in PSM belongs in the list of the associated *JavaResourceControllers* of the *RESTfulServicePSM* element.
JavaResourceModelManager	composition	0..*	The *RESTfulServicePSM* element may contain zero or more *JavaResourceModelManager* elements. Any *JavaResourceModelManager* that exists in PSM belongs in the list of the associated *JavaResourceModelManagers* of the *RESTfulServicePSM* element.
JavaResourceControllerManager	composition	0..*	The *RESTfulServicePSM* element may contain zero or more *JavaResourceControllerManager* elements. Any *JavaResourceControllerManager* that exists in PSM belongs in the list of the associated *JavaResourceControllerManagers* of the *RESTfulServicePSM* element.
JavaAlgoResourceModel	composition	0..*	The *RESTfulServicePSM* element may contain zero or more *JavaAlgoResourceModel* elements. Any *JavaAlgoResourceModel* that exists in PSM belongs in the list of the associated *JavaAlgoResourceModels* of the *RESTfulServicePSM* element.
JavaAlgoController	composition	0..*	The *RESTfulServicePSM* element may contain zero or more *JavaAlgoController* elements. Any *JavaAlgoController* that exists in PSM belongs in the list of the associated *JavaAlgoControllers* of the *RESTfulServicePSM* element.
JPAController	composition	1	The *RESTfulServicePSM* must contain exactly one *JPAController* element.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *RESTfulServicePSM* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *allModelsHaveRResourcePropertiesWithProperJPAAnnotations:* This OCL constraint verifies that every *JavaResourceModel* element that exists in PSM has the needed by the *JPA* standard, *Java* properties for every related resource of it, so as to produce the underlying relational schema. It also verifies that each one of such properties also has the appropriate *JPAAnnotations* "@OneToMany" and "@OnDelete".

- *allModelsHavePResourcePropertiesWithProperJPAAnnotations:* This constraint verifies that every *JavaResourceModel* element that exists in PSM has the needed by the *JPA* standard, *Java* properties for every resource of which it is related, so as to produce the underlying relational schema. It also verifies that each one of such properties also has the appropriate *JPAAnnotations* "@ManyToOne", "@JoinColumn" and "@ForeignKey".

- *rMPropertiesHaveColumnAnnotation:* This OCL constraint verifies that every *JavaResourceModel* element that exists in PSM has the needed by the *JPA* standard, "@Column" annotation, so as to produce the underlying relational schema.

- *rMPropertiesSettersHaveProperXMLTransientAnnotation:* This OCL constraint verifies that the *JavaResourceModels* functions have the appropriate *JAXB* "@XmlTransient" annotations if needed.

- *RControllerUniqueHTTPVerbsPerParent:* This OCL constraint verifies that any *JavaResourceController* has at most one *HTTPActivity* for each *HTTPVerb* that is allowed, that is *GET, PUT* and *DELETE,* per resource of which is it related.

- *RCManagerHasUniqueHTTPVerbsPerParent:* This OCL constraint verifies that any *JavaResourceControllerManager* element has at most one *HTTPActivity* for each *HTTPVerb* that is allowed, that is *POST* and *GET,* per resource of which is it related.

5.2.2.3 JavaResourceModel Element

Overview

The *JavaResourceModel* element models the data of a resource with a *Java* class. This can be seen as the *Model* part of the *Model-View-Controller* design pattern or a simple Java Bean. This element aims to fully define every resource property by providing its name and type as well as provide functions that access them (*setters/getters).* Figure 23 demonstrates a simplified diagram with its properties and relations.

Properties

Table 50 JavaResourceModel's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JavaResourceModel* element. Among others, this is used to define the corresponding *Java* class.
parentName	EString	1	This is the name of the parent *CIM Resource* of this *JavaResourceModel.*

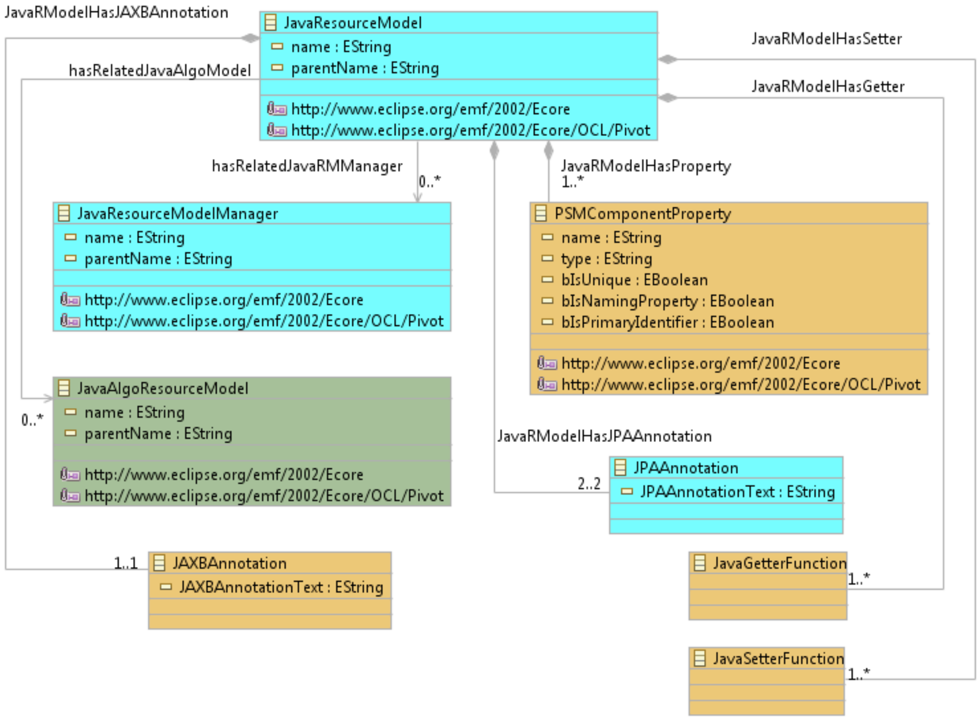

JavaRModelHasJAXBAnnotation
JavaResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaRModelHasSetter
hasRelatedJavaAlgoModel
JavaRModelHasGetter
hasRelatedJavaRMManager
0..*
JavaRModelHasProperty
1..*
JavaResourceModelManager
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
PSMComponentProperty
name : EString
type : EString
bIsUnique : EBoolean
bIsNamingProperty : EBoolean
bIsPrimaryIdentifier : EBoolean
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaAlgoResourceModel
0..*
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaRModelHasJPAAnnotation
JPAAnnotation
2..2
JPAAnnotationText : EString
1..1
JAXBAnnotation
JAXBAnnotationText : EString
JavaGetterFunction
1..*
JavaSetterFunction
1..*

Figure 23 PSM meta-model simplified diagram: JavaResourceModel element

Relations

Table 51 JavaResourceModel's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
PSMComponentProperty	composition	1..*	The *JavaResourceModel* element must be associated with at least one or more *PSMComponentProperty* elements. Since *JavaResourceModels* model the data of the resource, the existence of a *JavaResourceModel* that does not have any *PSMComponentProperties* , would be a contradiction since there would be no data to model.
JavaSetterFunction	composition	1..*	The *JavaResourceModel* element must be associated with at least one *JavaSetterFunction*. Since *JavaResourceModels* have at least one *PSMComponentProperty* there must be at least one *JavaSetterFunction* to alter its value.
JavaGetterFunction	composition	1..*	The *JavaResourceModel* element must be associated with at least one *JavaGetterFunction*. Since *JavaResourceModels* have at least one *PSMComponentProperty* there must be at least one *JavaGetterFunction* to retrieve its value.
JavaAlgoResourceModel	association	0..*	The *JavaResourceModel* element may be associated with zero or more *JavaAlgoResourceModels*. This association denotes the fact that a specific *JavaAlgoResourceModel* may be related resource of a specific *JavaResourceModel.*
JavaResourceModelManager	association	0..*	The *JavaResourceModel* element may be associated with zero or more *JavaResourceModelManager* elements. This association denotes the fact that a specific *JavaResourceModel* may have some related *JavaResourceModelManager*.
JAXBAnnotation	composition	1	The *JavaResourceModel* element must be associated with exactly one *JAXBAnnotation* element. This association denotes the fact that every *JavaResourceModel* must have the *JAXB* "@XMLRootElement" annotation.
JPAAnnotation	composition	2	The *JavaResourceModel* element must be associated with exactly two *JPAAnnotation* elements. This association denotes the fact that every *JavaResourceModel* must have exactly one "@Entity" and exactly one "@Table" JPA annotation.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *JavaResourceModel* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *uniqueXMLRootElementAnnotation:* This OCL constraint verifies that every *JavaResourceModel* element has exactly one "@XmlRootElement" annotation.

- *uniqueNamingProperty:* This OCL constraing verifies that every *JavaResourceModel* element has exactly one *PSMComponentProperty* that is naming property.

- *neededRModelJPAAnnotationsExist:* This OCL constraint verifies that every *JavaResourceModel* element has exactly one "@Entity" JPA annotation as well as exactly one "@Table" one.

- *uniquePrimaryIdentifier:* This OCL constraint verifies that every *JavaResourceModel* element has exactly one *PSMComponentProperty* that is primary identifier.

- *uniqueSetterForEveryProperty:* This OCL constraint verifies that every *JavaResourceModel* element has exactly one *JavaSetterFunction* for every *PSMComponentProperty* it has.

- *uniqueGetterForEveryProperty:* This OCL constraint verifies that every *JavaResourceModel* element has exactly one *JavaGetterFunction for every* PSMComponentProperty it has.

- *uniqueLinkListProperty:* This OCL constraint verifies that every *JavaResourceModel* element has exactly one *PSMComponentProperty* with the name "linklist" and that *PSMComponentProperty* has the "@Transient" JPA annotation.

- *properCollectionJPAAnnotations:* This OCL constraint verifies that every *JavaResourceModel* element's *PSMComponentProperties* that have multiplicity greater than one, have exactly one "@ElementCollection" JPA annotation, exactly one "@CollectionTable" as well as exactly one "@ForeignKey".

5.2.2.4 JavaResourceController Element

Overview

The *JavaResourceController* element models the web interface of a resource with a *Java* class. This happens in respect of the REST architectural style, which means that every *JavaResourceController* is assigned a unique *URI* and the *HTTPVerbs* are used as intended when accessing that *URI*. Figure 24 demonstrates a simplified diagram with its properties and relations.

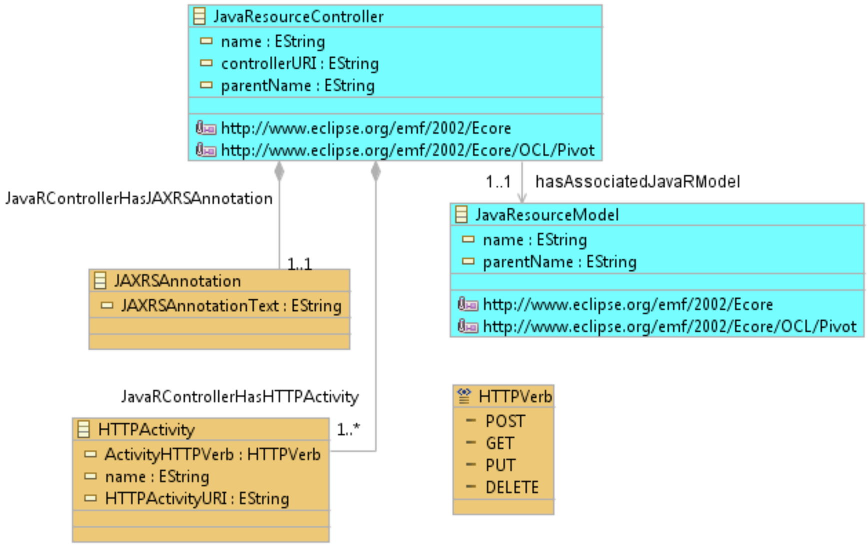

JavaResourceController
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaRControllerHasJAXRSAnnotation
1..1
hasAssociatedJavaRModel
JavaResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
1..1
JAXRSAnnotation
JAXRSAnnotationText : EString
JavaRControllerHasHTTPActivity
HTTPActivity
1..*
ActivityHTTPVerb : HTTPVerb
name : EString
HTTPActivityURI : EString
HTTPVerb
POST
GET
PUT
DELETE

Figure 24 PSM meta-model simplified diagram: JavaResourceController element

Properties

Table 52 JavaResourceController's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JavaResourceController* element. Among others, this is used to define the corresponding *Java* class.
parentName	EString	1	This is the name of the parent *CIM Resource* of this *JavaResourceController.*
controllerURI	EString	1	This EString contains the *URI* that is automatically assigned to every *JavaResourceController.*

Relations

Table 53 JavaResourceController's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JavaResourceModel	association	1	The *JavaResourceController* element must be associated with exactly one *JavaResourceModel*. The *JavaResourceController* is the WEB API that

			interacts with clients by receiving and sending data modelled by the associated *JavaResourceModel.*
HTTPActivity	composition	1..*	The *JavaResourceController* element must have at least one composition association with an *HTTPActivity* element. Since *HTTPActivities* are the intended functions to handle client requests, absence of any of them would lead to inability of clients to interact with that specific resource.
JAXRSAnnotation	composition	1	The *JavaResourceController* element must be associated with exactly one *JAXRSAnnotation* element. This association denotes the fact that every *JavaResourceController* element must have exactly one "@Path" *JAXRS* annotation.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *JavaResourceController* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *controllerURIPathAnnotation:* This OCL constraint verifies that every *JavaResourceController* has exactly one "@Path" *JAXRSAnnotation*.

- *RControllerNotAllowedVerb*: This OCL constraint verifies that every *JavaResourceController* does not have any *HTTPActivities* with the *HTTPVerb POST.*

- RControllerHasMinimumActivity: This OCL constraint verifies that every *JavaResourceController* has at least one *HTTPActivity.*

- *RControllerActivitiesHaveJAXRSAnnotations:* This OCL constraint verifies that every *JavaResourceController's HTTPActivity* has exactly one "@GET" or "@PUT" or "@DELETE" *JAXRSAnnotation* if and only if it is of that type and it also verifies that is has exactly one "@Path" *JAXRSAnnotation.*

5.2.2.5 JavaResourceModelManager Element

Overview

The *JavaResourceModelManager* element is always coupled with a *JavaResourceModel* one. It has the responsibility to create new instances of its associated *JavaResourceModel* (like a factory pattern) as well as retrieve all of them and send a list of *PSMHypermediaLinks* to them, back to the client. Figure 25 demonstrates a simplified diagram with its properties and relations.

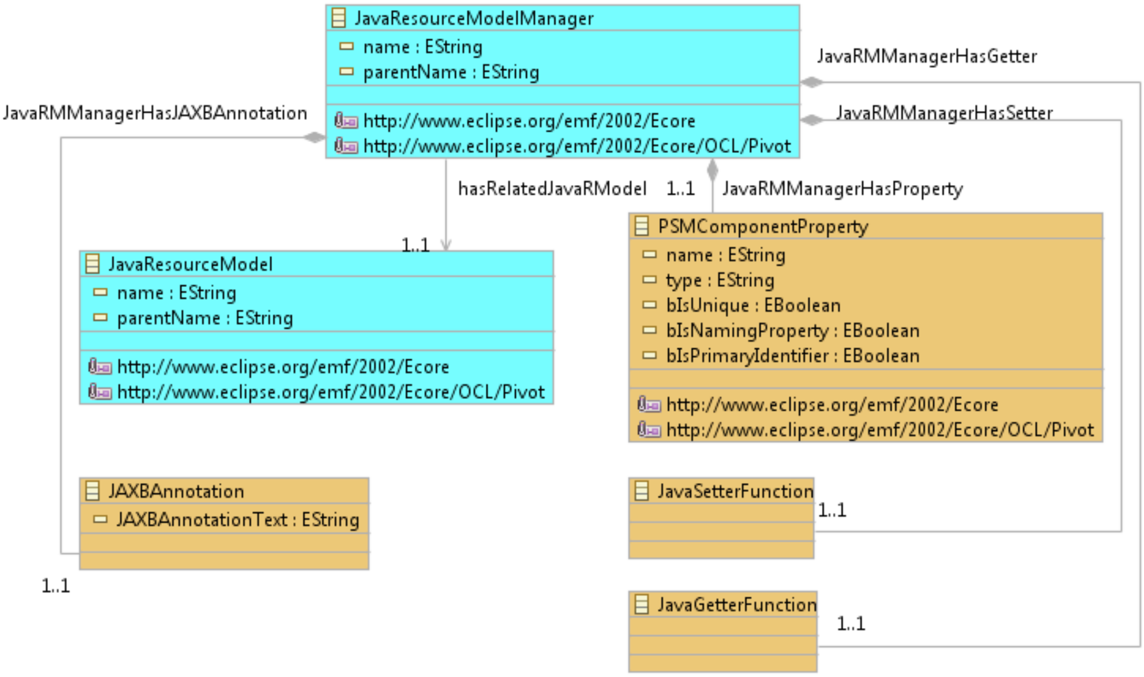

JavaResourceModelManager
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaRMManagerHasGetter
JavaRMManagerHasSetter
JavaRMManagerHasJAXBAnnotation
hasRelatedJavaRModel
1..1
JavaRMManagerHasProperty
1..1
JavaResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
PSMComponentProperty
name : EString
type : EString
bIsUnique : EBoolean
bIsNamingProperty : EBoolean
bIsPrimaryIdentifier : EBoolean
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JAXBAnnotation
JAXBAnnotationText : EString
1..1
JavaSetterFunction
1..1
JavaGetterFunction
1..1

Figure 25 PSM meta-model simplified diagram: JavaResourceModelManager element

Properties

Table 54 JavaResourceModelManager's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JavaResourceModelManager* element. Among others, it is used to define the name of the *Java* class.
parentName	EString	1	This is the name of the parent *CIM Resource* of this *JavaResourceModelManager*

Relations

Table 55 JavaResourceModelManager's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JavaResourceModel	association	1	The *JavaResourceModelManager* element must have exactly one associated *JavaResourceModel.* This association models the coupling between instances of a specific *JavaResourceModel* and its `managing' *JavaResourceModelManager.*
PSMComponentProperty	composition	1	The *JavaResourceModelMaanger element must have exactly one*

			PSMComponentProperty. This property is always named `linklist’ and is used to send back to client a list of *PSMHypermediaLinks* with all the possible next actions.
JavaSetterFunction	composition	1	The *JavaResourceModelManager* element must have exactly one *JavaSetterFunction* that will update the value of the unique `linklist’ *PSMComponentProperty* of it.
JavaGetterFunction	composition	1	The *JavaResourceModelManager element must have exactly one JavaGetterFunction that will retrieve the value of the unique `linklist’ PSMComponentProperty* of it.
JAXBAnnotation	composition	1	The *JavaResourceModelManager* element must be associated with exactly one *JAXBAnnotation* element. This association denotes the fact that every *JavaResourceModelManager* must have the *JAXB* “@XMLRootElement” annotation.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *JavaResourceModelManager* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *uniqueXMLRootElementAnnotation:* This OCL constraint verifies that every *JavaResourceModelManager* element must have exactly one “@XmlRootElement” *JAXBAnnotation.*

- *uniqueSetterForEveryProperty:* This OCL constraint verifies that every *JavaResourceModelManager* element must have exactly one *JavaSetterFunction* for every *PSMComponentProperty* it has.

- *uniqueGetterForEveryProperty:* This OCL constraint verifies that every *JavaResourceModelManager* element must have exactly one *JavaGetterFunction* for every *PSMComponentProperty* it has.

- *uniqueLinkListProperty:* This OCL constraint verifies that every *JavaResourceModelManager* element must have exacvtly one *PSMComponentProperty* with the name “linklist”.

5.2.2.6 JavaResourceControllerManager Element

Overview

The *JavaResourceControllerManager* element models the web interface of a *JavaResourceControllerManager* with a *Java* class. This happens in respect of the REST architectural style, which means that every *JavaResourceControllerManager* is assigned a unique *URI* and the *HTTPVerbs* are used when accessing that *URI*. Figure 26 demonstrates a simplified diagram with its properties and relations.

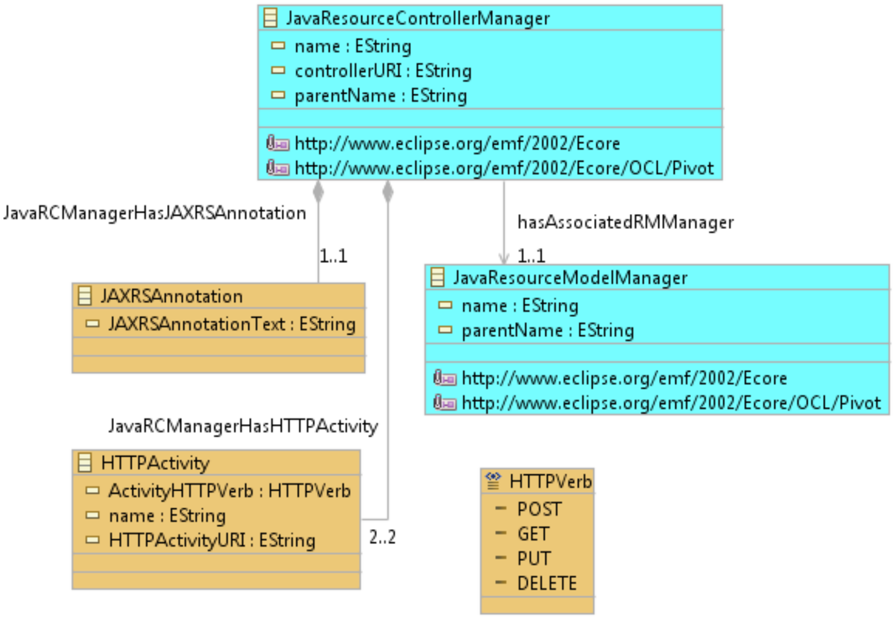

JavaResourceControllerManager
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaRCManagerHasJAXRSAnnotation
1..1
hasAssociatedRMManager
1..1
JAXRSAnnotation
JAXRSAnnotationText : EString
JavaResourceModelManager
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaRCManagerHasHTTPActivity
HTTPActivity
ActivityHTTPVerb : HTTPVerb
name : EString
HTTPActivityURI : EString
2..2
HTTPVerb
POST
GET
PUT
DELETE

Figure 26 PSM meta-model simplified diagram: JavaResourceControllerManager element

Properties

Table 56 JavaResourceControllerManager's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JavaResourceControllerManager* element. Among others, it is used to define the according *Java* class.
parentName	EString	1	This is the name of the parent *CIM Resource* of this *JavaResourceControllerManager.*
controllerURI	EString	1	This *EString* contains the *URI* that is automatically assigned to every *JavaResourceControllerManager.*

Relations

Table 57 JavaResourceControllerManager's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JavaResourceModelManager	association	1	The *JavaResourceControllerManager* element must be associated with exactly one *JavaResourceModelManager*. The *JavaResourceControllerManager* is the WEB API that interacts with data modelled by the associated *JavaResourceModelManager's JavaResourceModel.*
HTTPActivity	composition	2	The *JavaResourceControllerManager* element must have at least two *HTTPActivities.* Since *HTTPActivities* are the intended functions to handle client requests, absence of any of them would lead to inability of clients to interact with that specific resource.
JAXRSAnnotation	composition	1	The *JavaResourceControllerManager* element must be associated with exactly one *JAXRSAnnotation* element. This association denotes the fact that every *JavaResourceControllerManager* element must have exactly one "@Path" *JAXRS* annotation.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *JavaResourceControllerManager* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *controllerURIPathAnnotation:* This OCL constraint verifies that every *JavaResourceControllerManager* element has exactly one *"@Path" JAXRSAnnotation.*

- *RCManagerNotAllowedHTTPActivityVerbs:* This OCL constraint verifies that every *JavaResourceControllerManager* element does not have any *HTTPActivities* with *HTTPVerb PUT* or *DELETE.*
- *RCManagerActivitiesHaveJAXRSAnnotations:* This OCL constraint verifies that every *JavaResourceControllerManager's HTTPActivity* has exactly one "@POST" or "@GET" *JAXRSAnnotation* if and only if it is of that type and it also verifies that is has exactly one "@Path" *JAXRSAnnotation.*

5.2.2.7 PSMComponentProperty Element

Overview

The *PSMComponentProperty* PSM meta-model element models properties that instances of PSM meta-model elements may have with *Java* class properties. Such elements are the *JavaResourceModel,* the *JavaResourceModelManager* and the *JavaAlgoResourceModel*. The rest of this subsection defines *PSMComponentProperty's* properties, structural relations and constraints as well as behavioural constraints.

Properties

Table 58 PSMComponentProperty's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *PSMComponentProperty* element. Among others, it is used to define the according *Java* class property name.
type	EString	1	This is the type of the *PSMComponentProperty* element. Among others, it is used to define the corresponding Java class property type.
bIsUnique	EBoolean	1	This denotes the multiplicity of the *PSMComponentProperty.* If the EBoolean value is true then the property has multiplicity one. Otherwise there may be multiple instances of this property (array). Among others, it is used to define the multiplicity of the according *Java* class property.
bIsNamingProperty	EBoolean	1	This denotes whether a *PSMComponentProperty* element is a naming property as well or not. This attribute is useful for the client in order to pick one specific *JavaResourceModel* instance within a list with many ones and retrieve it following the provided *PSMHypermediaLink*.
bIsPrimaryIdentifier	EBoolean	1	This denotes whether a *PSMComponentProperty* element is a primary identifier or not. This attribute is used to calculate the URIs within the *PIMHypermediaLinks* that are sent to clients as well as to identify specific records in the local envisioned service database.

Relations

Table 59 PSMComponentProperty's relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JPAAnnotation	composition	1..*	The *PSMComponentProperty* element may be associated with one ore more *JPAAnnotations.*

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *PSMComponentProperty* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *primaryIdentifierHasProperAnnotations:* This OCL constraint verifies that every *PSMComponentProperty* element that is a primary identifier has exactly one "@Id" *JPAAnnotation* as well as exactly one "@GeneratedValue" *JPAAnnotation.*

5.2.2.8 JavaFunction Element

Overview

The *JavaFunction* PSM meta-model element is an abstract model of some PSM component function. It brings together common properties that are shared among other more specific function classes within the PSM meta-model.

Properties

Table 60 JavaFunction's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JavaFunction*. Among others, it is used to define the according *JavaFunction's* name.

Relations

Table 61 JavaFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
FunctionParameter	composition	0..*	The *JavaFunction* element may have zero or more associated *FunctionParameters.*

Behavioural Restrictions

The *JavaFunction* element has no behavioural constraints.

5.2.2.9 JavaGetterFunction Element

Overview

The *JavaGetterFunction* PSM meta-model element models a PSM component function that is dedicated in retrieving the value of a specific PSM component property with a *Java* function. It is a specialization of the *JavaFunction* element.

Properties

The *JavaGetterFunction* element has no additional properties other than those that are inherited from the *JavaFunction* element.

Relations

Table 62 JavaGetterFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
PSMComponentProperty	association	1	The *JavaGetterFunction* element must be associated with exactly one *PSMComponentProperty* element. This association models the fact that this *JavaGetterFunction* intention is to retrieve the value of the unique associated *PSMComponentProperty.* Therefore, it has a unique *FuntionParameter* that has the same name, type and multiplicity with the *PSMComponentProperty* and the *bIsReturnParameter* EBoolean attribute set to true.

Behavioural Restrictions

The *JavaGetterFunction* element has no behavioural restrictions.

5.2.2.10 JavaSetterFunction Element

Overview

The *JavaSetterFunction* PSM meta-model element models a PSM component function that is dedicated in updating a specific PSM component property. It is a specialization of the *JavaFunction* element.

Properties

The *JavaSetterFunction* element has no additional properties other than those that are inherited from the *JavaFunction* element.

Relations

Table 63 JavaSetterFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
PSMComponentProperty	association	1	The *JavaSetterFunction* element must be associated with exactly one *PSMComponentProperty* element. This association models the fact that this *JavaSetterFunction* intention is to update the value of the unique associated *PSMComponentProperty.* Therefore, it has a unique *FuntionParameter* that has the same name, type and multiplicity with the *PSMComponentProperty* and the *bIsReturnParameter* EBoolean attribute set to false.

Behavioural Restrictions

The *JavaSetterFunction* element has no behavioural constraints.

5.2.2.11 FunctionParameter Element

Overview

The *FunctionParameter* PSM meta-model element model any parameter of a *JavaFunction.*

Properties

Table 64 FunctionParameter's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *FunctionParameter* element. Among others, it is used to define the name of the according *Java* function parameter.
type	EString	1	This is the type of the *FunctionParameter* element. *Among others,* it is used to define the type of the according *Java* function parameter.
bIsUnique	EBoolean	1	This denotes the multiplicity of the *FuntionParameter.* If the EBoolean value is true then the parameter has multiplicity one. Otherwise there may be multiple instances of this parameter (array). Among others, it is used to define the multiplicity of the according *Java* function parameter.
bIsReturnParameter	EBoolean	1	This denotes whether a *FunctionParameter* element is a *JavaFunction* return parameter or an argument. If it has the true value then it is a return parameter. Otherwise, it is an argument.

Relations

The *FunctionParameter* element has no relations with other PSM meta-model elements.

Behavioural Restrictions

The *FunctionParameter* element has no behavioural constraints.

5.2.2.12 HTTPActivityFunctionParameter Element

Overview

The *HTTPActivityFunctionParameter* element is a specialization of the *FunctionParameter* PSM meta-model one. It models the additional attributes that arguments of *HTTPActivities* have. These are the extra *JAXRSAnnotations.*

Properties

The *HTTPActivityFunctionParameter* element has no additional properties other than those inherited from the *FunctionParameter* element.

Relations

Table 65 HTTPActivityFunctionParameter's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JAXRSAnnotation	composition	0..1	The *HTTPActivityFunctionParameter* must have zero or one associated *JAXRSAnnotation* elements. This association denotes the fact that some *HTTPActivityFunctionParameters* may have the "@PathParam" *JAXRSAnnotation.*

Behavioural Restrictions

The *HTTPActivityFunctionParameter* element has no behavioural constraints.

5.2.2.13 JPAAnnotation Element

Overview

The *JPAAnnotation* PSM meta-model element models a *JPA* standard annotation. Such annotations may apply to *JavaResourceModels* and their properties. These annotations are used by the *JPA* framework in order to perform *Object-Relation-Mapping (ORM)* and produce the underlying envisioned database relational schema of the system.

Properties

Table 66 JPAAnnotation's Properties

Name	Type	Multiplicity	Explanation
JPAAnnotationText	EString	1	This EString stores the text of the *JPAAnnotation* element.

Relations

The *JPAAnnotation* element has no properties.

Behavioural Restrictions

The *JPAAnnotation* element has no behavioural constraints.

5.2.2.14 JAXBAnnotation Element

Overview

The *JAXBAnnotation* PSM meta-model element models annotations of the *JAXB* framework. These annotations are used on *JavaResourceModels, JavaResourceModelManagers* as well as *JavaAlgoResourceModels* so as to allow the *JAXB* framework to perform the transformation from the object model to the desired media type format such as "application/JSON" or "application/XML"

Properties

Table 67 JAXBAnnotation's Properties

Name	Type	Multiplicity	Explanation
JAXBAnnotationText	EString	1	This EString stores the *JAXB* annotation text.

Relations

The *JAXBAnnotation* element is has no relations with other PSM meta-model elements.

Behavioural Restrictions

The *JAXBAnnotation* element has no behavioural constraints.

5.2.2.15 JavaAlgoResourceModel Element

Overview

JavaAlgoResourceModel PSM meta-model elements model any resource that does not fall in the category of the ones modelled by *JavaResourceModels,* with a *Java* class*.* Such resources embed some sort of algorithm rather than pure data modelling as *JavaResourceModels* do. Figure 27 demonstrates a simplified diagram with its properties and relations.

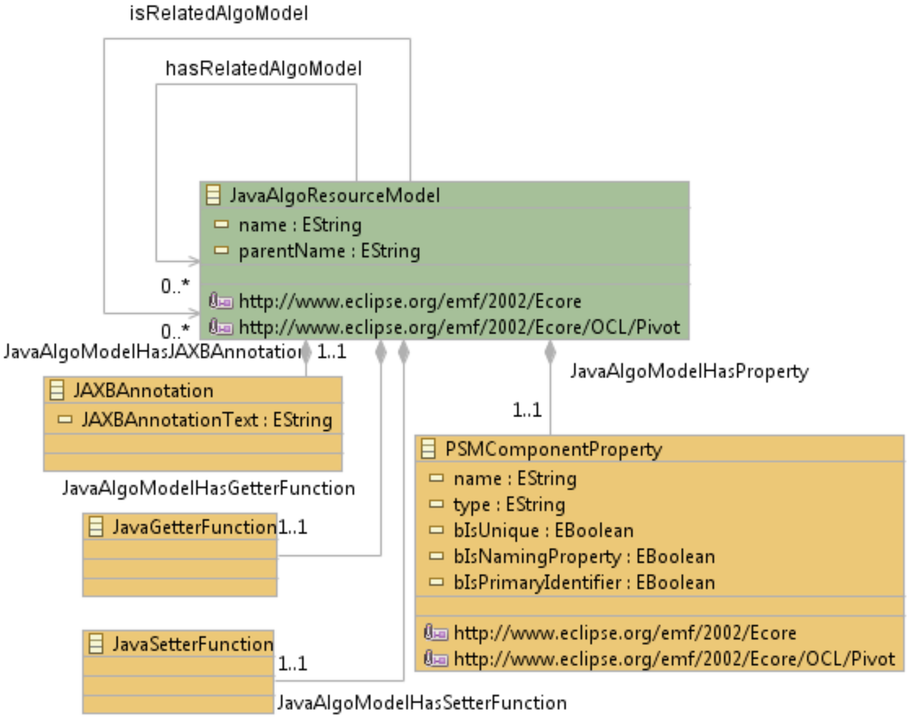

isRelatedAlgoModel
hasRelatedAlgoModel
JavaAlgoResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
0..*
0..*
JavaAlgoModelHasJAXBAnnotation
1..1
JAXBAnnotation
JAXBAnnotationText : EString
JavaAlgoModelHasProperty
1..1
PSMComponentProperty
name : EString
type : EString
bIsUnique : EBoolean
bIsNamingProperty : EBoolean
bIsPrimaryIdentifier : EBoolean
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
JavaAlgoModelHasGetterFunction
JavaGetterFunction
1..1
JavaSetterFunction
1..1
JavaAlgoModelHasSetterFunction

Figure 27 PSM meta-model simplified diagram: JavaAlgoResourceModel element

Properties

Table 68 JavaAlgoResourceModel's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JavaAlgoResourceModel* element. Among others, it is used for the definition of the according *Java* class.
parentName	EString	1	This is the name of the parent *CIM Resource* of this JavaAlgoResourceModel.

Relations

Table 69 JavaAlgoResourceModel's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JavaAlgoResourceModel	association	0..*	The *JavaAlgoResourceModel* may be associated with zero or more other *JavaAlgoResourceModels* by the means of the association "hasRelatedAlgoModel". This association models the fact that some *JavaAlgoResourceModels* may have some related *JavaAlgoResourceModels.*
JavaAlgoResourceModel	association	0..*	The *JavaAlgoResourceModels* may be associated with zero or more other *JavaAlgoResourceModels* by the means of the association "isRelatedAlgoModel". This association models the fact that some *JavaAlgoResourceModels* may have be related *JavaAlgoResourceModels* of zero or more other ones.
PSMComponentProperty	composition	1	The *JavaAlgoResourceModel element must have exactly one PSMComponentProperty*. This property is always named `linklist' and is used to send back to the client a list of *PSMHypermediaLinks* with all the possible next actions.
JavaSetterFunction	composition	1	The *JavaAlgoResourceModel* element must have exactly one *JavaSetterFunction* that will update the value of its unique `linklist' *PSMComponentProperty*.
JavaGetterFunction	composition	1	The *JavaAlgoResourceModel element must have exactly one JavaGetterFunction that will retrieve the value of its unique `linklist' PSMComponentProperty*.
JAXBAnnotation	composition	1	The *JavaAlgoResourceModel* element must have exactly one *JAXBAnnotation* element. This association denotes the fact that every *JavaAlgoResourceModel* must have exactly one "@XmlRootElement" *JAXBAnnotation.*

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *JavaAlgoResourceModel* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *uniqueXMLRootElementAnnotation:* This OCL constraint verifies that every *JavaAlgoResourceModel* element has exactly one "@XmlRootElement" *JAXBAnnotation*.

- *uniqueSetterForEveryProperty:* This OCL constraint verifies that every *JavaAlgoResourceModel* element has exactly one *JavaSetterFunction* for every *PSMComponentProperty* it has.

- *uniqueGetterForEveryProperty:* This OCL constraint verifies that every *JavaAlgoResourceModel* element has exactly one *JavaGetterFuncion* for every *PSMComponentProperty* it has.

- *uniqueLinkListProperty:* This OCL constraint verifies that every *JavaAlgoResourceController* has exactly one *PSMComponentProperty* that has the name "linklist".

5.2.2.16 JavaAlgoResourceController Element

Overview

The *JavaAlgoResourceController* PSM meta-model element models the *WEB API* of a *JavaAlgoResourceController.* This happens in respect of the REST architectural style, which means that every JavaAlgoResourceController is assigned a unique *URI* and the *HTTPVerbs* are used as intended when accessing that *URI*. Figure 28 demonstrates a simplified diagram with its properties and relations.

Properties

Table 70 JavaAlgoResourceController's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JavaAlgoResourceController* element. Among others, it is used to define the name of the according *Java* class.
parentName	EString	1	This is the name of the parent *CIM Resource* of this *JavaAlgoResourceController.*
controllerURI	EString	1	This Estring stores the unique *URI* that is assigned automatically to every *JavaAlgoResourceController* element.

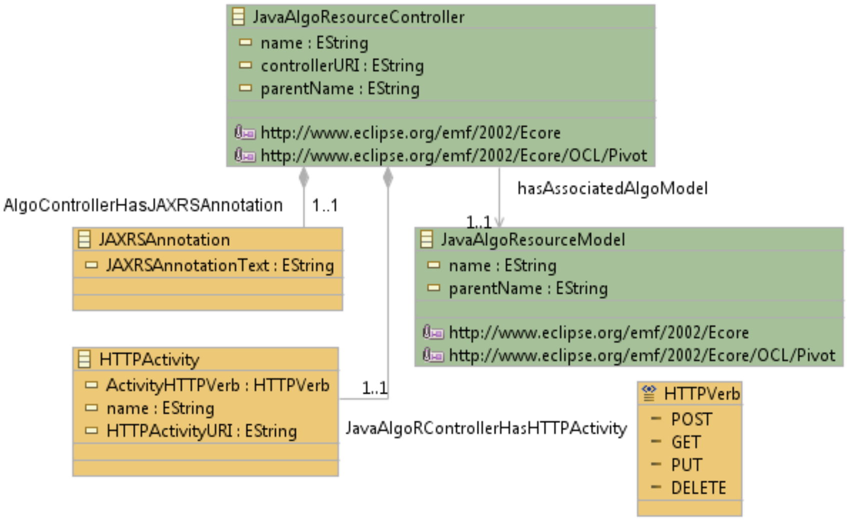

JavaAlgoResourceController
name : EString
controllerURI : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
AlgoControllerHasJAXRSAnnotation
1..1
hasAssociatedAlgoModel
1..1
JAXRSAnnotation
JAXRSAnnotationText : EString
JavaAlgoResourceModel
name : EString
parentName : EString
http://www.eclipse.org/emf/2002/Ecore
http://www.eclipse.org/emf/2002/Ecore/OCL/Pivot
HTTPActivity
ActivityHTTPVerb : HTTPVerb
name : EString
HTTPActivityURI : EString
1..1
JavaAlgoRControllerHasHTTPActivity
HTTPVerb
POST
GET
PUT
DELETE

Figure 28 PSM meta-model simplified diagram: JavaAlgoResourceController element

Relations

Table 71 JavaAlgoResourceController's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JavaAlgoResourceModel	association	1	The *JavaAlgoResourceController* element must be associated with exactly one *JavaAlgoResourceModel*. The *JavaAlgoResourceController* is the WEB API that interacts with data modelled by the associated *JavaAlgoResourceModel.*
HTTPActivity	composition	1	The *JavaAlgoResourceController* element must have exactly one *HTTPActivity.* Since *HTTPActivities* are the intended functions to handle client requests, absence of any of them would lead to inability of clients to interact with that specific resource.
JAXRSAnnotation	composition	1	The *JavaAlgoResourceController* element must have exactly one associated *JAXRSAnnotation* element. This association denotes the fact that every *JavaAlgoResourceController* must have exactly one "@Path" *JAXRSAnnotation.*

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *JavaAlgoResourceController* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *controllerURIPathAnnotation:* This OCL constraint verifies that every *JavaAlgoResourceController* element has exactly one "@Path" *JAXRSAnnotation.*

- *AlgoControllerHasUniqueProperHTTPActivity:* This OCL constraint verifies that every *JavaAlgoResourceController* element has exactly one *HTTPActivity* with either the *POST* or *GET HTTPVerb.*

- *AlgoControllerActivitiesHaveJAXRSAnnotations:* This OCL constraint verifies that every *HTTPActivity* of JavaAlgoResourceController elements has exactly one "@POST" or "@GET" *JAXRSAnnotation* if and only if it is of the according type as well as it also verifies that it has exactly one "@Path" *one.*

5.2.2.17 JAXRSAnnotation Element

Overview

The *JAXRSAnnotation* PSM meta-model element models the *JAXRS* framework annotations. Such annotations can be applied to *JavaResourceControllers, JavaResourceControllerManagers, JavaAlgoResourceControllers* and *HTTPActivities.* The *JAXRSAnnotations* are used so as to allow the *JAXRS* framework to route the client requests to the proper *HTTPActivities* for any needed handling as well as sending back the service response to clients.

Properties

Table 72 JAXRSAnnotation's Properties

Name	Type	Multiplicity	Explanation
JAXRSAnnotationText	EString	1	This EString stores the text of the *JAXRSAnnotatin* element.

Relations

The *JAXRSAnnotation* element has no relations with other PSM meta-model elements.

Behavioural Restrictions

The *JAXRSAnnotation* element has no behavioural constraints.

5.2.2.18 HTTPActivity Element

Overview

The *HTTPActivity* PSM meta-model element models a sub-part of any controller-type element's web *API* with a *Java* class function. That is, every *HTTPActivity* is dedicated to handle client requests of exactly one specific *HTTPVerb.*

Properties

Table 73 HTTPActivity's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *HTTPActivity* element. Among others, it is used to define the according *Java* class function name.
HTTPActivityURI	EString	1	This is the *URI* of the *HTTPActivity.* It is relative to the according controller-type element URI. Together, along with the services URL form the full URI at which a client should send its requests.
ActivityHTTPVerb	HTTPVerb	1	This denotes the specific *HTTPVerb* requests that this specific *HTTPActivity* accepts. As it is defined at *HTTPVerb* S-CASE data type definition, the four possible opionts are *POST, GET, PUT* and *DELETE.*

Relations

Table 74 HTTPActivity's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
HTTPActivityHandler	composition	1	The *HTTPActivity* element must have exactly one association with an *HTTPActivityHandler element.*
JPAActivity	association	0..1	The *HTTPActivity* element may have zero or one association with a *JPAActivity.* More specifically, the *HTTPActivities* of *JavaResourceControllers* and *JavaResourceControllerManagers* have exactly one associated *JPAActivity,* whilst the ones of *JavaAlgoResourceControllers* zero.
JAXRSAnnotation	composition	1..*	The *HTTPActivity* must have at least one associated *JAXRSAnnotation* element. This association denotes the fact that every *HTTPActivityElement* has at least the "@Path" *JAXRSAnnotation*, whilst others may also have the "@Consumes" and/or the "@Produces" ones.

Behavioural Restrictions

The *HTTPActivity* PSM meta-model element has no behavioural constraints.

5.2.2.19 HTTPActivityHandler Element

Overview

The *HTTPActivityHandler* PSM meta-model element models the actual implementation of the required actions that need to be taken in order to serve a client request that is forwarded to it by its overlying *HTTPActivity.* Usually, these include some sort of interaction with the local envisioned service database and then the generation of the list with *PSMHypermediaLinks* that will be sent back to the client alongside its request response.

Properties

Table 75 HTTPActivityHandler's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *HTTPActivityHandler* element. Among others, this is used for the definition of the according *Java* class.
HandlerHTTPVerb	HTTPVerb	1	This is the *HTTPVerb* type of requests that his *HTTPActivityHandler* element handles.

Relations

Table 76 HTTPActivityHandler's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JPAController	association	1	The *HTTPActivityHandler* element must have exactly one association with a *JPAController* one. Actually, this association denotes the fact that the *HTTPActivityHandler* uses the *JPAController* to access the local relational schema.
JavaHypermediaFunction	composision	1	The *HTTPActivityHandler* element must have exactly one *JavaHypermediaFunction* element.

Behavioural Restrictions

The *HTTPActivityHandler* element has no behavioural constraints.

5.2.2.20 JavaHypermediaFunction Element

Overview

The *JavaHypermediaFunction* PSM meta-model element models the function of each overlying *HTTPActivityHandler*, which creates the full list of hypermedia links to be sent back to the client.

Properties

The *JavaHypermediaFunction* element has no properties.

Relations

Table 77 JavaHypermediaFunction's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
PSMHypermediaLink	composition	1..*	The *JavaHypermediaFunction* element may be associated with one or more *PSMHypermediaLinks.* In fact, this association denotes that the *JavaHypermediaFunction* element creates the associated hypermedia links.

Behavioural Restrictions

This subsection lists all the behavioural restrictions of the properties and relations of a *JavaHypermediaFunction* element. The definition of each restriction begins by providing the unique OCL name of the implemented restriction within the PSM meta-model that can be found in appendix A.2.

- *algoControllerActivityAddsHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction of the JavaAlgoResourceController's* unique *HTTPActivity,* of some allowed *HTTPVerb* adds exactly one *PSMHypermediaLink* with the same *HTTPVerb, LinkType* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *JRCManagerPostActivityAddsPostHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb POST*, adds exactly one *PSMHypermediaLink* with the *POST HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *JRCManagerGetActivityAddsPostHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *POST HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *JRCManagerPostActivityAddsGetHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb POST*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *JRCManagerGetActivityAddsGetHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerPutActivityAddsPutHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb PUT*, adds exactly one *PSMHypermediaLink* with the *PUT HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerGetActivityAddsPutHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *PUT HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerGetActivityAddsDeleteHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *DELETE HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerPutActivityAddsGetHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb PUT*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerGetActivityAddsGetHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerPutActivityAddsDeleteHypermediaLinkToSelf:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb PUT*, adds exactly one *PSMHypermediaLink* with the *DELETE HTTPVerb* of type *SIBLING* and target controller being itself, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rCManagerPostActivityAddsGetHypermediaLinkToRRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb POST*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb* of type *CHILD* and target controller being the related *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rCManagerPostActivityAddsPutHypermediaLinkToRRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb POST*, adds exactly one *PSMHypermediaLink* with the *PUT HTTPVerb* of type *CHILD* and target controller being the related *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rCManagerPostActivityAddsDeleteHypermediaLinkToRRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb POST*, adds exactly one *PSMHypermediaLink* with the *DELETE HTTPVerb* of type *CHILD* and target controller being the related *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rCManagerGetActivityAddsGetHypermediaLinkToRRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb* of type *CHILD* and target controller being the related *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rCManagerGetActivityAddsPutHypermediaLinkToRRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *PUT HTTPVerb* of type *CHILD* and target controller being the related *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rCManagerGetActivityAddsDeleteHypermediaLinkToRRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *DELETE HTTPVerb* of type *CHILD* and target controller being the related *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rCManagerPostActivityAddsGetHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb POST*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

- *rCManagerPostActivityAddsPutHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb POST*, adds exactly one *PSMHypermediaLink* with the *PUT HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

- *rCManagerPostActivityAddsDeleteHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb POST*, adds exactly one *PSMHypermediaLink* with the *DELETE HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

- *rCManagerGetActivityAddsGetHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

- *rCManagerGetActivityAddsPutHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceControllerManager's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *PUT HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

- *rCManagerGetActivityAddsDeleteHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the

JavaResourceControllerManager's unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *DELETE HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

- *rControllerGetActivityAddsPostHypermediaLinkToRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *POST HTTPVerb,* of type *CHILD* and target controller being a related *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every related *JavaResourceControllerManager.*

- *rControllerGetActivityAddsGetHypermediaLinkToRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb,* of type *CHILD* and target controller being a related *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every related *JavaResourceControllerManager.*

- *rControllerPutActivityAddsPostHypermediaLinkToRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb PUT*, adds exactly one *PSMHypermediaLink* with the *POST HTTPVerb,* of type *CHILD* and target controller being a related *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every related *JavaResourceControllerManager.*

- *rControllerPutActivityAddsGetHypermediaLinkToRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb PUT*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb,* of type *CHILD* and target controller being a related *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every related *JavaResourceControllerManager.*

- *rControllerGetActivityAddsPostHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *POST HTTPVerb,* of type *PARENT* and target controller being the parent

JavaResourceControllerManager element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerGetActivityAddsGetHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb,* of type *PARENT* and target controller being the parent *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerPutActivityAddsPostHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb PUT*, adds exactly one *PSMHypermediaLink* with the *POST HTTPVerb,* of type *PARENT* and target controller being the parent *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client.
- *rControllerPutActivityAddsGetHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb PUT*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb,* of type *PARENT* and target controller being the parent *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerDeleteActivityAddsPostHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb DELETE*, adds exactly one *PSMHypermediaLink* with the *POST HTTPVerb,* of type *PARENT* and target controller being the parent *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *rControllerDeleteActivityAddsGetHypermediaLinkToParentRCManager:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb DELETE*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb,* of type *PARENT* and target controller being the parent *JavaResourceControllerManager* element, to the list of *PSMHypermediaLinks* to be returned to the client.

- *algoControllerActivityAddsHypermediaLinkToRAlgoController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the JavaAlgoResourceController*'s* unique *HTTPActivity* with an allowed *HTTPVerb*, adds exactly one *PSMHypermediaLink* with the *allowed HTTPVerb,* of type *CHILD* and target controller being a related JavaAlgoResourceController element, to the list of

PSMHypermediaLinks to be returned to the client. This is checked against every related *JavaAlgoResourceController.*

- *algoControllerActivityAddsHypermediaLinkToParentAlgoController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the JavaAlgoResourceController*'s* unique *HTTPActivity* with an allowed *HTTPVerb*, adds exactly one *PSMHypermediaLink* with the *allowed HTTPVerb,* of type *PARENT* and target controller being a parent JavaAlgoResourceController element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaAlgoResourceController.*

- *rControllerGetAddsHypermediaLinkToRAlgoController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb GET*, adds exactly one *PSMHypermediaLink* with the *allowed HTTPVerb,* of type *CHILD* and target controller being a related *JavaAlgoResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every related *JavaAlgoResourceController.*

- *rControllerPutAddsHypermediaLinkToRAlgoController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the *JavaResourceController's* unique *HTTPActivity* with *HTTPVerb PUT*, adds exactly one *PSMHypermediaLink* with the *allowed HTTPVerb,* of type *CHILD* and target controller being a related *JavaAlgoResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every related *JavaAlgoResourceController.*

- *algoControllerAddsGetHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the JavaAlgoResourceController*'s* unique *HTTPActivity* with an allowed *HTTPVerb*, adds exactly one *PSMHypermediaLink* with the *GET HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

- *algoControllerAddsPutHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the JavaAlgoResourceController*'s* unique *HTTPActivity* with an allowed *HTTPVerb*, adds exactly one *PSMHypermediaLink* with the *PUT HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

- *algoControllerAddsDeleteHypermediaLinkToParentRController:* This OCL constraint verifies that the *JavaHypermediaFunction* of the JavaAlgoResourceController*'s* unique *HTTPActivity* with an allowed *HTTPVerb,* adds exactly one *PSMHypermediaLink* with the *DELETE HTTPVerb,* of type *PARENT* and target controller being a parent *JavaResourceController* element, to the list of *PSMHypermediaLinks* to be returned to the client. This is checked against every parent *JavaResourceController.*

5.2.2.21 PSMHypermediaLink Element

Overview

The PSM*HypermediaLink* PSM meta-model element models the hypermedia links or HATEOAS of the REST architectural style with a *Java* class.

Properties

Table 78 PSMHypermediaLink's Properties

Name	Type	Multiplicity	Explanation
LinkHTTPVerb	HTTPVerb	1	Every *PSMHypermediaLink* that is sent to clients, among others includes the required *HTTPVerb* to be used to access the target resource in a new client request.
LinkType	LinkType	1	This denotes the relationship between the resource that responses with this hypermedia link (source) and the target resource. If the value of 'linkType' is *SIBLING* then the *PSMHypermediaLink* points to another available action of the same source resource. If the value is *CHILD* then it points to an available action of some target resource that is related to the resource of the source resource. Finally, if the value of the 'linkType' is *PARENT* then the link points to an available action of some target resource to which this source resource is related.

Relations

Table 79 PSMHypermediaLink's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JavaAlgoResourceController	association	0..1	The *PSMHypermediaLink* may be associated with zero or one *JavaAlgoResourceControllers*. If this link points the client to a *JavaAlgoResourceController* then it is associated with exactly one of them. Otherwise, it is associated with zero of

			them.
JavaResourceController	association	0..1	The *PSMHypermediaLink* may be associated with zero or one *JavaResourceControllers*. If this link points the client to a *JavaResourceController* then it is associated with exactly one of them. Otherwise, it is associated with zero of them.
JavaResourceControllerManager	association	0..1	The *PSMHypermediaLink* may be associated with zero or one *JavaResourceControllerManager*. If this link points the client to a *JavaResourceControllerManager* then it is associated with exactly one of them. Otherwise, it is associated with zero of them.

Behavioural Restrictions

The *PSMHypermediaLink* element has no behavioural constraints.

5.2.2.22 JPAController Element

Overview

The *JPAController* PSM meta-model element brings together all the *JPAActivities* that are needed by the envisioned service for all its interactions with its local RDBMS database. The *JPAController* follows the *Repository* design pattern and provides a unique accessible handler within the service following the *Signleton* design pattern.

Properties

Table 80 JPAController's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JPAController* element. Among others, it is used to define the according *Java* class.

Relations

Table 81 JPAController's Relations

Relation With PIM Element	Type	Multiplicity	Structural Constraints
JPAActivity	composition	1..*	The *JPAController element may be associated with one ore more HTTPActivities.*

Behavioural Restrictions

The *JPAController* element has not behavioural constraints.

5.2.2.23 JPAActivity Element

Overview

The *JPAActivity* PSM meta-model element models one function of the overlying *JPAController* element which is dedicated in performing exactly one type of interaction with exactly one table of the envisioned service's local database, as a *Java* class.

Properties

Table 82 JPAActivity's Properties

Name	Type	Multiplicity	Explanation
name	EString	1	This is the name of the *JPAAcitivty* element. Among others, it is used to define the according *Java* class function name.
JPAActivityHTTPVerb	HTTPVerb	1	The *HTTPVerb* of a *JPAActivity* denotes the type of action that it is intended to perform to the envisioned system's local database. If it has the value *POST* then it creates a new row in a database table. If it has the value *GET* then it either retrieves a specific entry or a list from a database table. If it has the value *PUT* then it updates the content of a database table row. Finally, if it has the value *DELETE* then it performs a *CASCADE DELETE* action in the local database.

Relations

The *JPAActivity* element has not any relations with other PSM meta-model elements.

Behavioural Restrictions

The *JPAActivity* element has no behavioural constraints.

6 PIM to PSM Transformation Definition

6.1 PIM to PSM Transformation Steps

As section 3 already explained, one of the principal activities in Model Driven Engineering is the transformation of one model that conforms to a specific meta-model to another model which conforms to a different meta-model. This section defines the transformation of an envisioned PIM model of a service that conforms to the PIM meta-model, which section 4 defines, to the corresponding PSM model that conforms to the PSM meta-model that section 5 defines. Figure 29 demonstrates this transformation process.

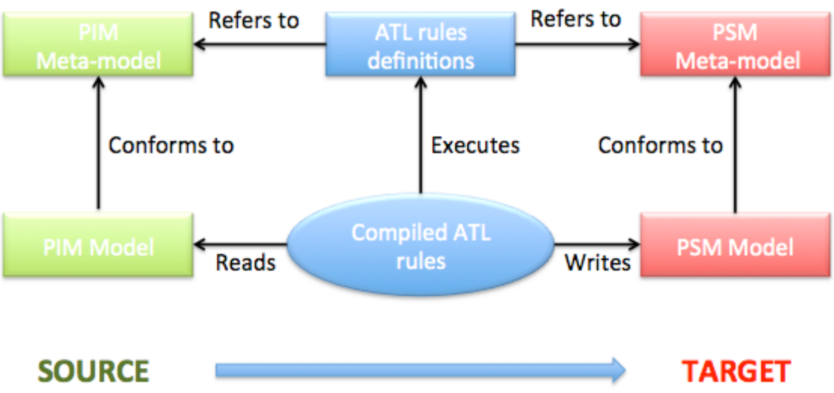


Figure 29 S-CASE PIM to PSM transformation process

Every rule that the next section defines performs one out of two possible actions. It either reads an element of the input PIM model and transforms it to the corresponding output PSM model or further refines already transformed PSM elements by applying extra design patterns or relating them to PSM meta-model elements that have no PIM counterparts. Since ATL is an hybrid declarative/imperative programming language, these rules can be implemented either way. However, since according to Amstel [9] the declarative rules are more efficient, they are always preferred whenever possible.

Therefore, in the first case the MDE engine always uses ATL declarative rules. Moreover, the declarative rules increase the visibility and traceability of the transformation process since every declarative rule simply declares for every PIM meta-model element (along with its properties and relations) its PSM counterpart without defining the step-by-step transformation (imperative programming). Thus, every rule transparently links source with target elements. In the second case, since there are no counterparts between PIM and PSM meta-model elements, the ATL imperative rules are used instead.

6.2 PIM to PSM ATL Rules

This subsection defines all the ATL rules that perform the transformation of any PIM instance service model to its PSM counterpart. The implementation of every rule can be found in appendix B using as key the unique ATL name-identifier that is provided at the caption of each table.

The following subsections define exactly one declarative or imperative ATL rule conceptually. For every rule one overview that describes its goals is provided. Additionally, a three-column table follows every declarative rule. The first column lists the properties and relations of the source (PIM) element that the rule transforms. The second one lists the properties and relations of the target (PSM) element to which the rule transforms the source element and the last column describes the rational and facts about this transformation or points to other ATL rules that are explicitly or implicitly called to perform the transformation activity. In the case of imperative rules, a two-column table is provided instead. The first column lists the properties and the relations of the PSM element to be introduced and the second one describes the way these elements are initialized, since they do not have PIM counterparts.

6.2.1 PIMRestfulService element transformation

The *PIMToPSMService* ATL rule transforms *RESTfulServicePIM* elements, of the PIM meta-model, to *RESTfulServicePSM* ones of the PSM meta-model. Therefore, the *RESTfulServicePSM* is the technological realization of the abstract envisioned service modelled with the *RESTfulServicePIM* element. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 83 PIMtoPSMService ATL rule

RESTfulServicePIM property/relation	RESTfulServicePSM property/relation	Explanation
name	name	The name of the *RESTfulServicePIM* remains unchanged after this transformation.
ResourceModel	JavaResourceModel	Every *ResourceModel* element that is associated with the *RESTfulServicePIM* element is transformed to a *JavaResourceModel* element that is associated with the *RESTfulServicePSM* element. This transformation is defined in the *createJavaRModel* declarative rule.
ResourceController	JavaResourceController	Every *ResourceController* element that is associated with the *RESTfulServicePIM* element is transformed to a *JavaResourceController* element that is associated with the *RESTfulServicePSM* element. This transformation is defined in the *createJavaRController* declarative rule.
ResourceControllerManager	JavaResourceControllerManager	Every *ResourceControllerManager* element that is associated with the *RESTfulServicePIM* element is transformed to a *JavaResourceControllerManager*

		element that is associated with the *RESTfulServicePSM* element. This transformation is defined in the *createJavaRCManager* declarative rule.
ResourceModelManager	JavaResourceModelManager	Every *ResourceModelManager* element that is associated with the *RESTfulServicePIM* element is transformed to a *JavaResourceModelManager* element that is associated with the *RESTfulServicePSM* element. This transformation is defined in the *createJavaRMManager* declarative rule.
AlgoResourceModel	JavaAlgoResourceModel	Every *AlgoResourceModel* element that is associated with the *RESTfulServicePIM* element is transformed to a *JavaAlgoResourceModel* element that is associated with the *RESTfulServicePSM* element. This transformation is defined in the *createJavaAlgoModel* declarative rule.
AlgoResourceController	JavaAlgoResourceController	Every *AlgoResourceController* element that is associated with the *RESTfulServicePIM* element is transformed to a *JavaAlgoResourceController* element that is associated with the *RESTfulServicePSM* element. This transformation is defined in the *createJavaAlgoController* declarative rule.
DatabaseController	JPAController	Every *DatabaseController* element that is associated with the *RESTfulServicePIM* element is transformed to a *JPAController* element that is associated with the *RESTfulServicePSM* element. This transformation is defined in the *createJavaRCManager* declarative rule.

6.2.2 ResourceModel element transformation

The *createJavaRModel* ATL rule transforms *ResourceModel* elements, of the PIM meta-model, to *JavaResourceModel* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 84 createJavaRModel ATL rule

ResourceModel property/relation	JavaResourceModel property/relation	Explanation
parentName	parentName	The *parentName* property of the *ResourceModel* element remains the same through this transformation as it refers to the original *CIM*

		Resource.
name	name	The *name* property of a *ResourceModel* is augmented with the prefix "*Java*" and then assigned to the name property of the *JavaResourceModel*.
-	JAXBAnnotation	As defined in the PSM meta-model, every *JavaResourceModel* element must be associated with exactly one *JAXBAnnotation.* However, that element has no PIM counterpart. Therefore, the "@XmlRootElement" *JAXBAnnotation* is added to the *JavaResourceModel* element by the imperative rule *createJAXBAnnotation.*
PIMComponentProperty	PSMComponentProperty	Every *PIMComponentProperty* element that is associated with a *ResourceModel* element is transformed to a *PSMComponentProperty* element that is associated with the according *JavaResourceModel* element. This transformation is defined in the *createPSMComponentProperty* declarative rule. Moreover, since the PSM meta-model defines extra *PSMComponentProperties* for the *JPA* relational schema for every *JavaResourceModel* that have no PIM counterpart*,* they are added to it by the *createManyToOneJPAProperty* and *createOneToManyJPAProperty* imperative rules.
ResourceModelManager	JavaResourceModelManager	Every *ResourceModelManager* element that is associated with the *ResourceModel* element element is transformed to a *JavaResourceModelManager* element that is associated with the according *JavaResourceModel* element. This transformation is defined in the *createJavaRMManager* declarative rule.
-	JPAAnnotation	As defined in the PSM meta-model, every *JavaResourceModel* element must be associated with exactly two *JPAAnnotations.* However, these elements have no PIM counterpart. Therefore, the "@Entity" and the "@Table" *JPAAnnotations* are added to the *JavaResourceModel* element by the imperative rule *createJPAAnnotation.*
SetterFunction	JavaSetterFunction	Every *SetterFunction* element that is associated with a *ResourceModel* element is transformed to a *JavaSetterFunction* element that is associated with the according *JavaResourceModel* one. This transformation is defined in the *createJavaSetter* declarative rule. Moreover, since the PSM meta-model defines extra *PSMComponentProperties* for the *JPA* relational schema for every *JavaResourceModel* that have no PIM counterpart*,* their *JavaSetterFunctions* are added to it by the *createManyToOneJPASetter* and *createOneToManyJPASetter* imperative rules.
GetterFunction	JavaGetterFunction	Every *GetterFunction* element that is associated

		with a *ResourceModel* element is transformed to a *JavaGetterFunction* element that is associated with the according *JavaResourceModel* one. This transformation is defined in the *createJavaGetter* declarative rule. Moreover, since the PSM meta-model defines extra *PSMComponentProperties* for the *JPA* relational schema for every *JavaResourceModel* that have no PIM counterpart*,* their *JavaGetterFunctions* are added to it by the *createManyToOneJPAGetter* and *createOneToManyJPAGetter* imperative rules.
AlgoResourceModel	JavaAlgoResourceModel	Every *AlgoResourceModel* element that is associated with a *ResourceModel* element is transformed to a *JavaAlgoResourceModel* element that is associated with the according *JavaResourceModel* element. This transformation is defined in the *createJavaAlgoModel* declarative rule.

6.2.3 ResourceController element transformation

The *createJavaRController* ATL rule transforms *ResourceController* elements, of the PIM meta-model, to *JavaResourceController* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 85 createJavaRController ATL rule

ResourceController property/relation	JavaResourceController property/relation	Explanation
parentName	parentName	The *parentName* property of the *ResourceController* element remains the same through this transformation as it refers to the original *CIM Resource.*
name	name	The *name* property of a *ResourceController* is augmented with the prefix "*Java*" and then assigned to the name property of the *JavaResourceController*.
controllerURI	controllerURI	The *controllerURI* property of the *ResourceController* element remains the same through this transformation.
ResourceModel	JavaResourceModel	Every *ResourceModel* element that is associated with a *ResourceController* element is transformed to a *JavaResourceModel* element that is associated with the according *JavaResourceController* element. This transformation is defined in the

		createJavaRModel declarative rule.
ResourceControllerCRUDActivity	HTTPActivity	Every *ResourceControllerCRUDActivity* element that is associated with a *ResourceController* element is transformed to an HTTPActivity element that is associated with the according *JavaResourceController* one. This transformation is defined in the *createHTTPActivity* declarative rule.
-	JAXRSAnnotation	As defined in the PSM meta-model, every *JavaResourceController* element must be associated with exactly one *JAXRSAnnotation.* However, this element has no PIM counterpart. Therefore, the "@Path" *JAXRSAnnotation* is added to the *JavaResourceController* element by the imperative rule *createJAXRSAnnotation.*

6.2.4 ResourceControllerManager element transformation

The *createJavaRCManager* ATL rule transforms *ResourceControllerManager* elements, of the PIM meta-model, to *JavaResourceControllerManager* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 86 PIMtoPSMService ATL rule

ResourceControllerManager property/relation	JavaResourceControllerManager property/relation	Explanation
parentName	parentName	The *parentName* property of the *ResourceControllerManager* element remains the same through this transformation as it refers to the original *CIM Resource.*
name	name	The *name* property of a *ResourceControllerManager* is augmented with the prefix "*Java*" and then assigned to the name property of the *JavaResourceControllerManager*.
controllerURI	controllerURI	The *controllerURI* property of the *ResourceControllerManager* element remains the same through this transformation.
ResourceModelManager	JavaResourceModelManager	Every *ResourceModelManager* element that is associated with a *ResourceControllerManager* element is transformed to an JavaResourceModelManager element that

		is associated with the according *JavaResourceControllerManager* one. This transformation is defined in the *createJavaRMManager* declarative rule.
ResourceControllerCRUDActivity	HTTPActivity	Every *ResourceControllerCRUDActivity* element that is associated with a *ResourceControllerManager* element is transformed to an HTTPActivity element that is associated with the according *JavaResourceControllerManager* one. This transformation is defined in the *createHTTPActivity* declarative rule.
-	JAXRSAnnotation	As defined in the PSM meta-model, every *JavaResourceControllerManager* element must be associated with exactly one *JAXRSAnnotation.* However, this element has no PIM counterpart. Therefore, the "@Path" *JAXRSAnnotation* is added to the *JavaResourceControllerManager* element by the imperative rule *createJAXRSAnnotation.*

6.2.5 ResourceModelManager element transformation

The *createJavaRMManager* ATL rule transforms *ResourceModelManager* elements, of the PIM meta-model, to *JavaResourceModelManager* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 87 createJavaRMManager ATL rule

ResourceModelManager property/relation	JavaResourceModelManager property/relation	Explanation
parentName	parentName	The *parentName* property of the *ResourceModelManager* element remains the same through this transformation as it refers to the original *CIM Resource.*
name	name	The *name* property of a *ResourceModelManager* is augmented with the prefix "*Java*" and then assigned to the name property of the *JavaResourceModelManager*.
PIMComponentProperty	PSMComponentProperty	Every *PIMComponentProperty* element that is associated with a *ResourceModelManager* element is transformed to a *PSMComponentProperty* element that is associated with the according *JavaResourceModelManager* element. This transformation is defined in

		the *createPSMComponentProperty* declarative rule.
SetterFunction	JavaSetterFunction	Every *SetterFunction* element that is associated with a *ResourceModelManager* element is transformed to a *JavaSetterFunction* element that is associated with the according *JavaResourceModelManager* one. This transformation is defined in the *createJavaSetter* declarative rule.
GetterFunction	JavaGetterFunction	Every *GetterFunction* element that is associated with a *ResourceModelManager* element is transformed to a *JavaGetterFunction* element that is associated with the according *JavaResourceModelManager* one. This transformation is defined in the *createJavaGetter* declarative rule.
ResourceModel	JavaResourceModel	Every *ResourceModel* element that is associated with a *ResourceModelManager* element is transformed to a JavaResourceModel element that is associated with the according *JavaResourceModelManager* one. This transformation is defined in the *createJavaRModel* declarative rule.
-	JAXBAnnotation	As defined in the PSM meta-model, every *JavaResourceModelManager* element must be associated with exactly one *JAXBAnnotation.* However, that element has no PIM counterpart. Therefore, the "@XmlRootElement" *JAXBAnnotation* is added to the *JavaResourceModelManager* element by the imperative rule *createJAXBAnnotation.*

6.2.6 AlgoResourceModel element transformation

The *createJavaAlgoModel* ATL rule transforms *AlgoResourceModel* elements of the PIM meta-model to *JavaAlgoResourceModel* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 88 createJavaAlgoModel ATL rule

AlgoResourceModel property/relation	JavaAlgoResourceModel property/relation	Explanation
parentName	parentName	The *parentName* property of the *AlgoResourceModel* element remains the same through this transformation as it refers to the original *CIM Resource.*
name	name	The *name* property of a *AlgoResourceModel* is augmented with the prefix "*Java*" and then assigned to the name property of the *JavaAlgoResourceModel*.
PIMComponentProperty	PSMComponentProperty	Every *PIMComponentProperty* element that is associated with a *AlgoResourceModel* element is transformed to a *PSMComponentProperty* element that is associated with the according *JavaAlgoResourceModel* element. This transformation is defined in the *createPSMComponentProperty* declarative rule.
SetterFunction	JavaSetterFunction	Every *SetterFunction* element that is associated with a *AlgoResourceModel* element is transformed to a *JavaSetterFunction* element that is associated with the according *JavaAlgoResourceModel* one. This transformation is defined in the *createJavaSetter* declarative rule.
GetterFunction	JavaGetterFunction	Every *GetterFunction* element that is associated with a *AlgoResourceModel* element is transformed to a *JavaGetterFunction* element that is associated with the according *JavaAlgoResourceModel* one. This transformation is defined in the *createJavaGetter* declarative rule.
-	JAXBAnnotation	As defined in the PSM meta-model, every *JavaAlgoResourceModel* element must be associated with exactly one *JAXBAnnotation.* However, that element has no PIM counterpart. Therefore, the "@XmlRootElement" *JAXBAnnotation* is added to the *JavaAlgoResourceModel* element by the imperative rule *createJAXBAnnotation.*
AlgoResourceModel	JavaAlgoResourceModel	Every *AlgoResourceModel* element that is associated with an AlgoResourceModel element is transformed to a *JavaAlgoResourceModel* element that is associated with the according *JavaAlgoResourceModel* one. This

		transformation is defined in the *createJavaAlgoModel* declarative rule.
AlgoResourceModel	JavaAlgoResourceModel	Every *AlgoResourceModel* element that has associated an *AlgoResourceModel* element is transformed to a JavaAlgoResourceModel element that is associated with the according *JavaAlgoResourceModel* one. This transformation is defined in the *createJavaAlgoModel* declarative rule.

6.2.7 AlgoResourceController element transformation

The *createJavaAlgoController* ATL rule transforms *AlgoResourceController* elements of the PIM meta-model to *JavaAlgoResourceController* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 89 createJavaAlgoController ATL rule

AlgoResourceController property/relation	JavaAlgoResourceController property/relation	Explanation
parentName	parentName	The *parentName* property of the *AlgoResourceController* element remains the same through this transformation as it refers to the original *CIM Resource.*
name	name	The *name* property of an AlgoResourceController is augmented with the prefix "*Java*" and then assigned to the name property of the *JavaAlgoResourceController*.
controllerURI	controllerURI	The *controllerURI* property oft he *AlgoResourceController* element remains the same through this transformation.
ResourceControllerCRUDActivity	HTTPActivity	Every *ResourceControllerCRUDActivity* element that is associated with an AlgoResourceController element is transformed to an HTTPActivity element that is associated with the according *JavaAlgoResourceController* element. This transformation is defined in the *createHTTPActivity* declarative rule.
AlgoResourceModel	JavaAlgoResourceModel	Every *AlgoResourceModel* element that is associated with an AlgoResourceController element is transformed to a *JavaAlgoResourceModel* element that is associated with the according *JavaAlgoResourceController* one. This

		transformation is defined in the *createJavaAlgoModel* declarative rule.
-	JAXRSAnnotation	As defined in the PSM meta-model, every *JavaAlgoResourceController* element must be associated with exactly one *JAXRSAnnotation.* However, this element has no PIM counterpart. Therefore, the "@Path" *JAXRSAnnotation* is added to the *JavaAlgoResourceController* element by the imperative rule *createJAXRSAnnotation.*

6.2.8 DatabaseController element transformation

The *createJPAController* ATL rule transforms *DatabaseController* elements of the PIM meta-model to *JPAController* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 90 createJPAController ATL rule

DatabaseController property/relation	JPAController property/relation	Explanation
name	name	The name of the *JPAController* element is always set to "JPAController" for any service produced by S-CASE.
RDBMSActivity	JPAActivity	Every *RDBMSActivity* element that is associated with a *DatabaseController* element is transformed to an *HTTPActivity* element that is associated with the according *JPAController* one. This transformation is defined in the *craeteJPAActivity* declarative rule.

6.2.9 PIMComponentProperty element transformation

The *createPSMComponentProperty* ATL rule transforms *PIMComponentProperty* elements of the PIM meta-model to *PSMComponentProperty* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 91 createPSMComponentProeprty ATL rule

PIMComponentProperty property/relation	PSMComponentProperty property/relation	Explanation
name	name	The *name* property of the *PIMComponentProperty* element remains the same through this transformation.
type	type	The *type* property of the *PIMComponentProperty* element remains the same through this transformation.
isUnique	bIsUnique	The *isUnique property* of the *PIMComponentProperty* element remains the same through this transformation.
isNamingProperty	bIsNamingProperty	The *isNamingProperty* property of the *PIMComponentProperty* element remains the same through this transformation.
isPrimaryIdentifier	bIsPrimaryIdentifier	The *isPrimaryIdentifier* property of the *PIMComponentProperty* element remains the same through this transformation.
-	JPAAnnotation	As defined in the PSM meta-model, every *PSMComponentProperty* element must be associated with at least one *JPAAnnotation.* However, this element has no PIM counterpart. Therefore, any needed *JPAAnnotations* are added to the *PSMComponentProperty* element by the imperative rule *createJPAAnnotation.* Specifically, if and only if the property is a primary identifier, the transformation adds the “@Id”, “@GeneratedValue” and the “@Column” *JPAAnnotations.* Otherwise, if and only if it is a “linklist” property, the transformation adds the “@Transient” *JPAAnnotation.* Finally, for all the rest types of *PSMComponentProperties,* if their *isUnique* property is true, the transformation adds the “@Column” *JPAAnnotation,* otherwise it adds the *“@ElementCollection”, “@CollectionTable”, “@Column”* and *“@ForeignKey”* ones.

6.2.10 SetterFunction element transformation

The *createJavaSetter* ATL rule transforms *SetterFunction* elements of the PIM meta-model to *JavaSetterFunction* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 92 createJavaSetter ATL rule

SetterFunction property/relation	JavaSetterFunction property/relation	Explanation
PIMComponentProperty	PSMComponentProperty	Every *PIMComponentProperty* element that is associated with a *SetterFunction* element is transformed to a *PSMComponentProperty* element that is associated with the according *JavaSetterFunction* one. This transformation is defined in the *createPSMComponentProperty* declarative rule.
name	name	The *name* property of the *SetterFunction* element remains the same through this transformation.
FunctionParameter	FunctionParameter	Every *FunctionParameter* element that is associated with a *SetterFunction* element is transformed to a *FunctionParameter* element that is associated with the according *JavaSetterFunction* one. This transformation is defined in the *createFunctionParameter* declarative rule.

6.2.11 FunctionParameter element transformation

The *createFunctionParameter* ATL rule transforms *FunctionParameter* elements of the PIM meta-model to *Functionparameter* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 93 createFunctionParameter ATL rule

FunctionParameter property/relation	FunctionParameter property/relation	Explanation
name	name	The *name* property of the *FunctionParameter* element remains the same through this transformation.
type	type	The *type* property of the *FunctionParameter* element remains the same through this transformation.
isUnique	bIsUnique	The *isUnique* property of the *FunctionParameter* element remains the same through this transformation.
bIsReturnparameter	bIsReturnParameter	The *bIsReturnParameter* property of the *FunctionParameter* element remains the same through this transformation.

6.2.12 GetterFunction element transformation

The *createJavaGetter* ATL rule transforms *GetterFunction* elements of the PIM meta-model to *JavaGetterFunction* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 94 createJavaGetter ATL rule

GetterFunction property/relation	JavaGetterFunction property/relation	Explanation
PIMComponentProperty	PSMComponentProperty	Every *PIMComponentProperty* element that is associated with a *GetterFunction* element is transformed to a *PSMComponentProperty* element that is associated with the according *JavaGetterFunction* one. This transformation is defined in the *createPSMComponentProperty* declarative rule.
name	name	The *name* property of the *GetterFunction* element remains the same through this transformation.
FunctionParameter	FunctionParameter	Every *FunctionParameter* element that is associated with a *GetterFunction* element is transformed to a *FunctionParameter* element that is associated with the according *JavaGetterFunction* one. This transformation is defined in the *createFunctionParameter* declarative rule.

6.2.13 RDBMSActivity element transformation

The *createJPAActivity* ATL rule transforms *RDBMSActivity* elements of the PIM meta-model to *JPAActivity* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 95 createJPAActivity ATL rule

RDBMSActivity property/relation	JPAActivity property/relation	Explanation
name	name	The *name* property of the *RDBMSActivity* element remains the same through this transformation.
rdbmsVerb	JPAActivityHTTPVerb	This transformation rule maps every *RDBMSVerb* of an *RDBMSActivity* to exactly

		one *HTTPVerb* of a *JPAActivity* following the rules: 1. INSERT is transformed to POST 2. SELECT is transformed to GET 3. UPDATE is transformed to PUT 4. DELETE is transformed to DELETE

6.2.14 ResourceControllerCRUDActivity element transformation

The *createHTTPActivity* ATL rule transforms *ResourceControllerCRUDActivity* elements of the PIM meta-model to *HTTPActivity* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 96 createHTTPActivity ATL rule

ResourceControllerCRUDActivity property/relation	HTTPActivity property/relation	Explanation
name	name	The *name* property of *ResourceControllerCRUDActivity* element is transformed as follows: 1. create + *CRUDActivity* name is transformed to post + *HTTPActivity* name 2. read + *CRUDActivity* name is transformed to get + *HTTPActivity* name 3. update + *CRUDActivity* name is transformed to put + *HTTPActivity* name 4. delete + *CRUDActivity* name is transformed to delete + *HTTPActivity* name
activityURI	HTTPActivityURI	The *name* property of the *ResourceControllerCRUDActivity* element remains the same through this transformation.
crudVerb	ActivityHTTPVerb	This transformation rule maps every *CRUDVerb* of a *ResourceControllerCRUDActivity* to exactly one *HTTPVerb* of an *HTTPActivity* following the rules: 1. CREATE is transformed to POST 2. READ is transformed to GET 3. UPDATE is transformed to PUT 4. DELETE is transformed to DELETE
CRUDActivityHandler	HTTPActivityHandler	Every *CRUDActivityHandler* element that is associated with a *ResourceControllerCRUDActivity* element is transformed to a *HTTPActivityHandler* element that is associated with the according *HTTPActivity* one. This transformation is defined in

		the *createHTTPActivityHandler* declarative rule.
-	HTTPActivityParameter	As it is defined in PSM meta-model, an *HTTPActivity* may have zero or more *HTTPActivityFunctionParameters.* Since these element do not have PIM counterparts, this rule adds them to the envisioned PSM service’s model by taking into account the type of the *HTTPVerb* this *HTTPActivity* handles as well as the relationships of this resource with other ones.
-	JAXRSAnnotation	As defined in the PSM meta-model, every *HTTPActivity* element must be associated with at least one *JAXRSAnnotation.* However, this element has no PIM counterpart. Therefore, the “@Path” *JAXRSAnnotation* is added to the *HTTPActivity* element by invocation of the imperative rule *createJAXRSAnnotation* with the follow guidline*:* 1. If the *HTTPactivity* is of type *POST* then the transformation rule adds one “@Path” *JAXRS annotation as well as one “@POST”, one “@Produces”* and one *“@Consumes”.* 2. Otherwise, if the *HTTPActivity* is of type *GET* then the transformation rule adds one “@Path” *JAXRSAnnotation* as well as one “@GET” and one “@Produces”. 3. Otherwise, if the *HTTPActivity* is of type *PUT* then the transformation rule adds one “@Path” *JAXRSAnnotation,* one “@PUT”, one “@Produces” and one “@Consumes” one. 4. Finally, if the *HTTActivity* is of type *DELETE* then the transformation rules adds one “@Path” *JAXRSAnnotation*, one “@DELETE” and one “@Produces” one.

6.2.15 CRUDActivityHandler element transformation

The *createHTTPActivityHandler* ATL rule transforms *CRUDActivityHandler* elements of the PIM meta-model to *HTTPActivityHandler* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 97 createHTTPActivityHandler ATL rule

CRUDActivityHandler property/relation	HTTPActivityHandler property/relation	Explanation
name	name	The *name* property of the *HTTPActivityHandler* element remains the same through this transformation.
crudVerb	HandlerHTTPVerb	This transformation rule maps every *CRUDVerb* of a *CRUDActivityHandler* to exactly one *HTTPVerb* of an *HTTPActivityHandler* following the rules: 1. CREATE is transformed to POST 2. READ is transformed to GET 3. UPDATE is transformed to PUT 4. DELETE is transformed to DELETE
CreateHypermediaPIMFunction	JavaHypermediaFunction	Every *CreateHypermediaPIMFunction* element that is associated with an *HTTPActivityHandler* element is transformed to a *JavaHypermediaFunction* element that is associated with the according *HTTPActivityHandler* one. This transformation is defined in the *createHypermediaFunction* declarative rule.

6.2.16 CreateHypermediaPIMFunction element transformation

The *createHypermediaFunction* ATL rule transforms *CreateHypermediaPIMFunction* elements of the PIM meta-model to *JavaHypermediaFunction* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 98 createHypermediaFunction ATL rule

CreateHypermediaPIMFunction property/relation	JavaHypermediaFunction property/relation	Explanation
HypermediaLink	PSMHypermediaLink	Every *HypermediaLink* element that is associated with a *CreateHypermediaPIMFunction* element is transformed to a *PSMHypermediaLink* element that is associated with the according *JavaHypermediaFunction* one. This transformation is defined in the *createPSMHypermediaLink* declarative rule.

6.2.17 HypermediaLink element transformation

The *createPSMHypermediaLink* ATL rule transforms *HypermediaLink* elements of the PIM meta-model to *PSMHypermediaLink* ones of the PSM meta-model. The mapping of properties and relations between the two meta-model elements is defined in the table below:

Table 99 createPSMHypermediaLink ATL rule

HypermediaLink property/relation	PSMHypermediaLink property/relation	Explanation
linkType	LinkType	The *linkType* of the *HypermediaLink* remains the same through this transformation.
linkCRUDVerb	LinkHTTPVerb	This transformation rule maps every *CRUDVerb* of a *HypermediaLink* to exactly one *HTTPVerb* of a *PSMHypermediaLink* following the rules: 1. CREATE is transformed to POST 2. READ is transformed to GET 3. UPDATE is transformed to PUT 4. DELETE is transformed to DELETE
AlgoResourceController	JavaAlgoResourceController	Every *AlgoResourceController* element that is associated with a *HypermediaLink* element is transformed to a *JavaAlgoResourceController* element that is associated with the according *PSMHypermediaLink* one. This transformation is defined in the *createJavaAlgoController* declarative rule. This happens if, and only if, the specific *HypermediaLink* links to an *AlgoResourceController.*
ResourceController	JavaResourceController	Every *ResourceController* element that is associated with a *HypermediaLink* element is transformed to a *JavaResourceController* element that is associated with the according *PSMHypermediaLink* one. This transformation is defined in the *createJavaRController* declarative rule. This happens if, and only if, the specific *HypermediaLink* links to a *ResourceController.*
ResourceControllerManager	JavaResourceControllerManager	Every *ResourceControllerManager* element that is associated with a *HypermediaLink* element is transformed to a *JavaResourceControllerManager* element that is associated with the according

		PSMHypermediaLink one. This transformation is defined in the *createJavaRCManager* declarative rule. This happens if, and only if, the specific *HypermediaLink* links to a *ResourceControllerManager.*

6.2.18 JAXBAnnotation element introduction

The *createJAXBAnnotation* ATL rule introduces *JAXBAnnotation* elements of the PSM meta-model to *PSM* model of the envisioned service. That is, *JAXBAnnotation* elements do not have PIM counterparts. The properties and relations of the rule are defined in the table below:

Table 100 createJAXBAnnotation ATL rule

JAXBAnnotation property/relation	Explanation
JAXBAnnotationText	The property *JAXBAnnotationText* of the *JAXBAnnotation* element is set equal to the *String annotationText* argument passed to this imperative rule by the calling declarative rule.

6.2.19 HTTPActivityFunctionParameter element introduction

The *createHTTPActivityParameter* ATL rule introduces *HTTPActivityFunctionParameter* elements of the PSM meta-model for every *ResourceControllerCRUDActivity* PIM element that is transformed to *HTTPActivity* PSM element. The properties and relations of the rule are defined in the table below:

Table 101 createHTTPActivityParameter ATL rule

HTTPActivityFunctionParameter property/relation	Explanation
name	The *name* of the *HTTPActivityFunctionParameter* element is set equal to the *parameterName* argument passed to this imperative rule by the calling declarative one.
type	The *type* of the *HTTPActivityFunctionParameter* element is set equal to the *parameterType* argument passed to this imperative rule by the calling declarative one.
bIsUnique	The *bIsUnique* property is always set to *true* by this transformation rule since all the *HTTPActivityFunctionParameters* are of unary multiplicity.

bIsReturnParameter	The *bIsReturnParameter* is always set to *false* by this transformation rule since al the *HTTPActivityFunctionParameters* are not return parameters.
JAXRSAnnotation	If the *type* of this *HTTPActivityFunctionParameter* element is equal to "int" then this transformation rule adds a *JAXRSAnnotation* element with the "@PathParam" *JAXRSAnnotation* by invoking the imperative rule *createJAXRSAnnotation.*

6.2.20 JPAAnnotation element introduction

The *createJPAAnnotation* ATL rule introduces *JPAAnnotation* elements of the PSM meta-model to *PSM* model of the envisioned service. That is, *JPAAnnotation* elements do not have PIM counterparts. The properties and relations of the rule are defined in the table below:

Table 102 createJPAAnnotation ATL rule

JPAAnnotation property/relation	Explanation
JPAAnnotationText	The property *JPAAnnotationText* of the *JPAAnnotation* element is set equal to the *String annotationText* argument passed to this imperative rule by the calling declarative rule.

6.2.21 JAXRSAnnotation element introduction

The *createJAXRSAnnotation* ATL rule introduces *JAXRSAnnotation* elements of the PSM meta-model to *PSM* model of the envisioned service. That is, *JAXRSAnnotation* elements do not have PIM counterparts. The properties and relations of the rule are defined in the table below:

Table 103 createJAXRSAnnotation ATL rule

JAXRSAnnotation property/relation	Explanation
JAXRSAnnotationText	The property *JAXRSAnnotationText* of the *JAXRSAnnotation* element is set equal to the *String annotationText* argument passed to this imperative rule by the calling declarative rule.

6.2.22 “@ManyToOne” JavaResourceModel JPA Properties

The *createManyToOneJPAProperty* ATL rule adds *PSMComponentProperty* elements to *JavaResourceModels*. These *PSMComponentProperties* do not have PIM counterparts and model the extra properties needed by the *JPA* standard between associated classes so as to create the corresponding relational database schema. This imperative rule has two input arguments, provided by the calling ones. The first is a *ResourceModel* element and the second another *ResourceModel* (parent*)* to which the first is related. The properties and relations of the rule are defined in the table below:

Table 104 createManyToOneJPAProperty ATL rule

PSMComponentProperty property/relation	Explanation
name	The *name* property of this *PSMComponentProperty* element is set equal to the *parentName* of the parent *ResourceModel* that is provided as argument to this imperative rule.
type	The *type* property of this *PSMComponentProperty* element is set equal to the *name* of the parent *ResourceModel* that is provided as argument to this imperative rule, prefixed with the string “Java”.
bIsUnique	The *bIsUnique* property of this *PSMComponentProperty* element is set always to *true* by this transformation rule since that type of JPA properties are always of unary multiplicity.
bIsNamingProperty	The *bIsNamingProperty* property of this *PSMComponentProperty* element is set always to *false* by this transformation rule since that type of JPA properties cannot be naming properties of a *JavaResourceModel* element.
bIsPrimaryIdentifier	The *bIsPrimaryIdentifier* property of this *PSMComponentProperty* element is set always to *false* by this transformation rule since that type of JPA properties cannot be primary identifiers of a *JavaResourceModel* element.
JPAAnnotation	As defined in the PSM meta-model, every *PSMComponentProperty* element must be associated with at least one *JPAAnnotation.* However, this element has no PIM counterpart. Therefore, any needed *JPAAnnotations* are added to the *PSMComponentProperty* element by the imperative rule *createJPAAnnotation.* Specifically, since this rule creates *PSMComponentProperties* that model JPA “@ManyToOne” properties, it adds one “@ManyToOne(fetch = FetchType.EAGER)” *JPAAnnotation* as well as one *“*@JoinColumn(name = “joinColumnXname”)*”* and one “@ForeignKey(name = “fkNameX”)*”*.

6.2.23 "@OneToMany" JavaResourceModel JPA Properties

The *createOneToManyJPAProperty* ATL rule adds *PSMComponentProperty* elements to *JavaResourceModels*. These *PSMComponentProperties* do not have PIM counterparts and model the extra properties needed by the *JPA* standard between associated classes so as to create the corresponding relational database schema. This imperative rule has two input arguments, provided by the calling ones. The first is a *ResourceModel* element and the second another *ResourceModel (*child*)* that is related to the first one. The properties and relations of the rule are defined in the table below:

Table 105 createOneToManyJPAProperty ATL rule

PSMComponentProperty property/relation	Explanation
name	The *name* property of this *PSMComponentProperty* element is set equal to the *name* of the child *ResourceModel* that is provided as argument to this imperative rule, prefixed with the string "SetOfJava".
type	The *type* property of this *PSMComponentProperty* element is set equal to the *name* of the child *ResourceModel* that is provided as argument to this imperative rule, prefixed with the string "Java".
bIsUnique	The *bIsUnique* property of this *PSMComponentProperty* element is set always to *false* by this transformation rule since that type of JPA properties are always of multiplicity greater than one.
bIsNamingProperty	The *bIsNamingProperty* property of this *PSMComponentProperty* element is set always to *false* by this transformation rule since that type of JPA properties cannot be naming properties of a *JavaResourceModel* element.
bIsPrimaryIdentifier	The *bIsPrimaryIdentifier* property of this *PSMComponentProperty* element is set always to *false* by this transformation rule since that type of JPA properties cannot be primary identifiers of a *JavaResourceModel* element.
JPAAnnotation	As defined in the PSM meta-model, every *PSMComponentProperty* element must be associated with at least one *JPAAnnotation.* However, this element has no PIM counterpart. Therefore, any needed *JPAAnnotations* are added to the *PSMComponentProperty* element by the imperative rule *createJPAAnnotation.* Specifically, since this rule creates *PSMComponentProperties* that model JPA "@OneToMany" properties, it adds one "@OneToMany(fetch = FetchType.LAZY, mappedBy"resourceX", cascade = CascadeType.REMOVE, orphanRemoval = true)" *JPAAnnotation*.

6.2.24 "@ManyToOne" JavaResourceModel setting functions

The *createManyToOneJPASetter* ATL rule adds *JavaSetterFunction* elements to *JavaResourceModels*. These *JavaSetterFunctions* do not have PIM counterparts and model the setting functions of properties needed by the *JPA* standard between associated classes so as to create the corresponding relational database schema. This imperative rule has two input arguments, provided by the calling ones. The first is a *ResourceModel* element and the second another *ResourceModel (*parent*)* to which the first is related. The properties and relations of the rule are defined in the table below:

Table 106 createManyToOneJPASetter ATL rule

JavaSetterFunction property/relation	Explanation
name	The *name* property of this *JavaSetterFunction* is set equal to the *parentName* of the parent *ResourceModel*, prefixed with the string "set".
FunctionParameter	This type of *JavaSetterFunctions* has exactly one *FunctionParameter.* However, since there are no PIM counterparts to these *FunctionParameters*, they are created by calling the imperative rule *createJPASetterParameter.*
JAXBAnnotation	As defined in the PSM meta-model, every *JavaSetterFunction* element must be associated with zero or one *JAXBAnnotations.* However, this element has no PIM counterpart. Therefore, any needed *JAXBAnnotations* are added to the *JavaSetterFunction* element by the imperative rule *createJAXBAnnotation.* Specifically, since this rule creates *JavaSetterFunctions* that model JPA "@ManyToOne" property setting functions, it adds one "@XmlTransient" *JAXBAnnotation*.

6.2.25 "@OneToMany" JavaResourceModel setting functions

The *createOneToManyJPASetter* ATL rule adds *JavaSetterFunction* elements to *JavaResourceModels*. These *JavaSetterFunctions* do not have PIM counterparts and model the setting functions of properties needed by the *JPA* standard between associated classes so as to create the corresponding relational database schema. This imperative rule has two input arguments, provided by the calling ones. The first is a *ResourceModel* element and the second another *ResourceModel (*child*)* that is related to the first one. The properties and relations of the rule are defined in the table below:

Table 107 createOneToManyJPASetter ATL rule

JavaSetterFunction property/relation	Explanation
name	The *name* property of this *JavaSetterFunction* is set equal to the *name* of the child *ResourceModel*, prefixed with the string "setSetOfJava".
FunctionParameter	This type of *JavaSetterFunctions* has exactly one *FunctionParameter.* However, since there are no PIM counterparts to these *FunctionParameters*, they are created by calling the imperative rule *createJPASetterParameter.*
JAXBAnnotation	As defined in the PSM meta-model, every *JavaSetterFunction* element must be associated with zero or one *JAXBAnnotations.* However, this

	element has no PIM counterpart. Therefore, any needed *JAXBAnnotations* are added to the *JavaSetterFunction* element by the imperative rule *createJAXBAnnotation.* Specifically, since this rule creates *JavaSetterFunctions* that model JPA "@OneToMany" property setting functions, it adds one "@XmlTransient" *JAXBAnnotation*.

6.2.26 JavaResourceModel's JPA Properties setting functions parameters

The *createJPASetterFunctionParameter* ATL rule adds *FunctionParameter* elements to *JavaSetterFunctions*. These *FunctionParameters* do not have PIM counterparts and model the *FunctionParameters* of property setting functions needed by the *JPA* standard between associated classes so as to create the corresponding relational database schema. This imperative rule has two input arguments, provided by the calling ones. The first is a *ResourceModel* element and the second is a *boolean* argument. The value of this *Boolean* argument depends on whether this imperative rule is called by the *createOneToManyJPASetter* one, or the *createManyToOneJPASetter.* The properties and relations of the rule are defined in the table below:

Table 108 createJPASetterFunctionparameter ATL rule

FunctionParameter property/relation	Explanation
name	The *name* property of this *FunctionParameter* is set equal to the *parentName* of the *ResourceModel* that is provided as argument, if and only if this rule is called by the *createManyToOneJPASetter* one. Otherwise, the *name* property is set equal to the *name* of the provided *ResourceModel,* prefixed with the string "SetOfJava"
type	The *type* property of this *FunctionParameter* is set equal to the *name* of the provided *ResourceModel*, prefixed with the string "Java"
bIsUnique	The *bIsUnique* property of this *FunctionParameter* is set equal to *true* if and only if this rule is called by the *createManyToOneJPASetter* one. Otherwise, the *bIsUnique* property is set equal to *false.*
bIsReturnParameter	The *bIsReturnParameter* property of this *FunctionParameter* is set equal to false since no *JavaSetterFunction* can have a return parameter.

6.2.27 "@ManyToOne" JavaResourceModel getting functions

The *createManyToOneJPAGetter* ATL rule adds *JavaGetterFunction* elements to *JavaResourceModels*. These *JavaGetterFunctions* do not have PIM counterparts and model the getting functions of properties needed by the *JPA* standard between associated classes so as to create the corresponding relational database schema. This imperative rule has two input arguments, provided by the calling ones. The first is a *ResourceModel* element and the second another *ResourceModel (*parent*)* of which the first is related. The properties and relations of the rule are defined in the table below:

Table 109 createManyToOneJPAGetter ATL rule

JavaGetterFunction property/relation	Explanation
name	The *name* property of this *FunctionParameter* is set equal to the *parentName* of the parent *ResourceModel* that is provided as argument, prefixed with the string "get"
FunctionParameter	This type of *JavaGetterFunctions* has exactly one *FunctionParameter.* However, since there are no PIM counterparts to these *FunctionParameters*, they are created by calling the imperative rule *createJPAGetterFunctionParameter.*

6.2.28 "@OneToMany" JavaResourceModel getting functions

The *createOneToManyJPAGetter* ATL rule adds *JavaGetterFunction* elements to *JavaResourceModels*. These *JavaGetterFunctions* do not have PIM counterparts and model the getting functions of properties needed by the *JPA* standard between associated classes so as to create the corresponding relational database schema. This imperative rule has two input arguments, provided by the calling ones. The first is a *ResourceModel* element and the second another *ResourceModel (*child*)* that is related of the first one. The properties and relations of the rule are defined in the table below:

Table 110 createOneToManyJPAGetter ATL rule

JavaGetterFunction property/relation	Explanation
name	The *name* property of this *FunctionParameter* is set equal to the *parentName* of the child *ResourceModel* that is provided as argument, prefixed with the string "getSetOfJava"
FunctionParameter	This type of *JavaGetterFunctions* has exactly one *FunctionParameter.* However, since there are no PIM counterparts to these *FunctionParameters*, they are created by calling the imperative rule *createJPAGetterFunctionParameter.*

6.2.29 JavaResourceModel's JPA Properties setting functions parameters

The *createJPAGetterFunctionParameter* ATL rule adds *FunctionParameter* elements to *JavaGetterFunctions*. These *FunctionParameters* do not have PIM counterparts and model the *FunctionParameters* of property getting functions needed by the *JPA* standard between associated classes so as to create the corresponding relational database schema. This imperative rule has two input arguments, provided by the calling ones. The first is a *ResourceModel* element and the second is a *boolean* argument. The value of this *Boolean* argument depends on whether this imperative rule is called by the *createOneToManyJPAGetter* one, or the *createManyToOneJPAGetter.* The properties and relations of the rule are defined in the table below:

Table 111 createJPAgetterFunctionParameter ATL rule

FunctionParameter property/relation	Explanation
name	The *name* property of this *FunctionParameter* is set equal to the *parentName* of the *ResourceModel* that is provided as argument, if and only if this rule is called by the *createManyToOneJPAGetter* one. Otherwise, the *name* property is set equal to the *name* of the provided *ResourceModel,* prefixed with the string "SetOfJava"
type	The *type* property of this *FunctionParameter* is set equal to the *name* of the provided *ResourceModel*, prefixed with the string "Java"
bIsUnique	The *bIsUnique* property of this *FunctionParameter* is set equal to *true* if and only if this rule is called by the *createManyToOneJPAGetter* one. Otherwise, the *bIsUnique* property is set equal to *false.*
bIsReturnParameter	The *bIsReturnParameter* property of this *FunctionParameter* is set equal to true since *JavaGetterFunctions* cannot have a return parameter.

References

[1] R.T. Fielding. (2000). Architectural Styles and the Design of Network-based Software Architecture", PhD Dissertation, Department of Information and Computer Science, University of California, Irvine

[2] http://rubyonrails.org

[3] http://emf-rest.com

[4] http://www.django-rest-framework.org

[5] http://www.omg.org/mda/

[6] http://eclipse.org/modeling/emf/

[7] http://www.omg.org/spec/OCL/

[8] https://eclipse.org/atl/

[9] Marcel van Amstel, Steven Boems, Ivan Kurtev and Luis Ferreira Pires. Performance in Model Transformations: Experiments with ATL and QVT. ICMT 2011, pp. 198-212

[10] http://martinfowler.com/articles/richardsonMaturityModel.html

A. Annex A – Ecore Meta-models enriched with OCL Constraints

A.1 PIM Ecore Meta-model enriched with OCL Constraints

Due to space limitations, the Ecore PIM meta-model definition with its OCL Contraints can be found https://github.com/s-case/mde. It should be opened with the OCLinEclipse editor.

A.2 PSM Ecore Meta-model enriched with OCL Constraints

Due to space limitations, the Ecore PSM meta-model definition with its OCL Contraints can be found https://github.com/s-case/mde. It should be opened with the OCLinEclipse editor.

B. Annex B – ATL Transformation Rules Full List

Due to space limitations, the ATL transformation rules implementation can be found https://github.com/s-case/mde. The file should be opened with Eclipse.