Infrastructure Configuration Service - programmer manual

version 0.2.0 - 11 December 2013

PoSecCo

http://www.posecco.eu

Contents

1 Introduction

This document provides an overview of the *Infrastructure Configuration Service* (ICS) considering the developer's point of view. As described in "Infrastructure Configuration Service - user manual" [2], this tool used to transform the logical associations, generated by *LA Generation Service*, into abstract configurations for Data Protection and Filtering in the PoSecCo workflow.

This service consists of a set of different modules with their own user interface. All of this modules and their options are documented in the following sections.

For an introduction on the process and on data models see "Infrastructure Configuration Service - user manual" [2] and D3.6 "Models for generating the desired configurations" [1] for a complete description.

The Sec. 2 is devoted to explaining the software architecture and the Sec. 3 documents the public API of the tool.

Finally, Sec. 4 describes the architecture of optimization module and discusses how to update the tool to support new optimization models.

2 Software architecture

The Infrastructure Configuration Service (ICS) transforms logical associations (LAs) into abstract configurations. ICS is composed by the Data Protection and the Filtering modules that refine LAs into abstract configurations. The Data Protection is executed before the Filtering because the filtering rules depends on data protection abstract configurations. For example, when IPsec is configured, the needed filtering rules must be created to permit related traffic.

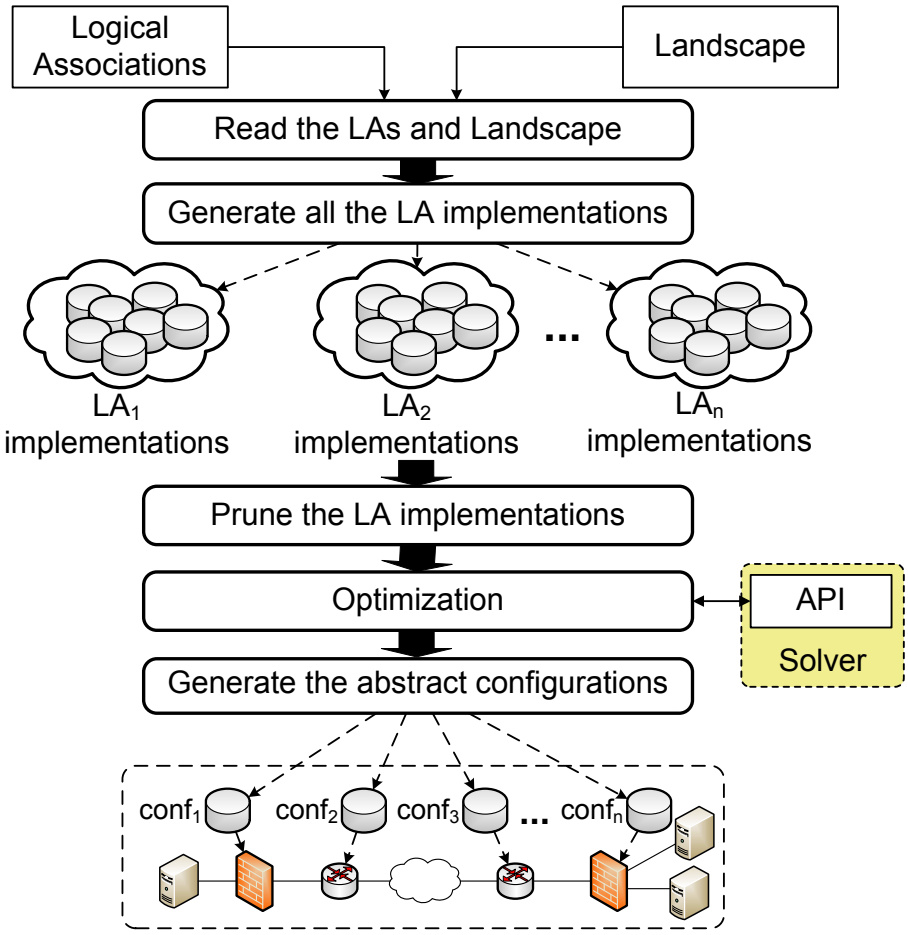

Figure 1: ICS refinement process.

Both modules share the workflow (depicted in Fig. 1) as discussed in [2]. The refinement process can be resumed as follow:

Generation of Data Protection configurations At the beginning, the module loads the PoSecCo ontology which contains LAs (generated by LA Generation Service) and landscape description. Then it generates the set of LAs implementations, performs pruning and optimization tasks. The optimal solution is transformed into a Data Protection abstract configurations;

Generation of Data Protection configurations similarly to previous stage, this module loads the PoSecCo ontology, it generates the set of LAs implementations, performs pruning and optimization tasks. Finally, the optimal solution is transformed into Filtering abstract configurations.

The following paragraphs provides a briefly overview of the internal implementation of the Data Protection and the Filtering modules.

General implementation details

The Data Protection and the Filtering modules are written using the Java programming language and packaged into a single plug-in. In addition the technologies listed in Table 1 are extensively used in the project.

Name	Website of the project
Eclipse plug-in framework	http://eclipse.org/
Remote Application Platform toolkit	http://eclipse.org/rap/
OWL API ontology library	http://owlapi.sourceforge.net/
Pellet reasoner	http://clarkparsia.com/pellet/
Hermit reasoner	http://hermit-reasoner.com/
SPARQL-DL query engine	http://www.derivo.de/en/resources/sparql-dl-api.html
JGraphT graph library	http://jgrapht.org/
lp_solve	http://lpsolve.sourceforge.net/5.5/

Table 1: The technologies used in the ICS.

Implementation of the Data Protection Module

This software module implements the process to generate the Data Protection configurations as described in [2].
The software architecture is composed by the following Java classes:

- `ICSChannelProtectionCoordinator`;
- `ProcessLAandLandscapeModule`;
- `GenerateAllImplementationsModule`;
- `ImplementationsPruningModule`;
- `OptimizationModule`;
- `GenerateAbstractConfigurationsModule`.

`ICSChannelProtectionCoordinator` coordinates the process and the other modules, offers the public API (described into Sec. 3);

`ProcessLAandLandscapeModule` performs a set of analysis on landscape and on logical associations to generate the set of artifacts required by the next steps. The required inputs are:

- `LandscapeDescription`: the description of the landscape;
- `LogicalAssociationsRawContainer`: are the set of logical associations passed as input by LA Generation Service;
- `LogicalAssociationsRawDependencyContainer`: contains the set of dependencies among the logical associations (e.g., considering the First Matching Rule strategy and Deny Take Precedence strategy). These dependencies are provided by the Analysis Service.

The provided outputs are:

- `PhysicalTopologyGraph`: the physical topology of the landscape;
- `ChprotNodeContainer`: the set of channel protection devices;
- `LogicalAssociationsInputContainer`: this set contains the low-level logical associations (i.e., the information regarding IP addresses, ports and protocols). This process is required to derive ip addresses, ports and protocols when they are not specified by the LA Generation Service;
- `LAsChprot`: the set of channel protection logical associations.

`GenerateAllImplementationsModule` generates the channel protection implementations. Each implementation contains a set of nodes that may be used to enforce a logical association (using available technologies e.g., IPsec, SSL/TLS, etc.). The required inputs are:

- `LAsChprot`: the set of channel protection logical associations;
- `PhysicalTopologyGraph`: the physical topology of the landscape;
- `LandscapeDescription`: the description of the landscape;
- `ChprotNodeContainer`: the set of channel protection devices.

The provided output is:

- `LAChprotImplContainer`: the set of channel protection implementations.

`ImplementationsPruningModule` performs pruning on a set of channel protection implementations. The pruning task reduces the set of LA implementations which may be discarded either according to some heuristics or by allowing users to explicitly discard some of them. The required inputs are:

- `LAChprotImplContainer`: the set of channel protection implementations;
- `PruningCriterion`: the criterion to perform pruning.

The provided output is:

- `LAChprotImplContainer`: the reduced set of channel protection implementations.

`OptimizationModule` builds the optimization model and identifies the optimal solution. The required inputs are:

- `OptimizationProfile`: the optimization profile (i.e., the optimization target function);
- `SolverParameter`: the set of solver parameters (type of the solver e.g., lp_solve or CPLEX, described in Sec. 4);
- `LAChprotImplContainer`: the set of channel protection implementations;
- `LAsChprot`: the set of channel protection logical associations;
- `WeightsOracle`: provides the set of weights for landscape elements and LAs.

The provided outputs are:

- `ChprotOptModel`: the channel protection optimization model (described in Sec. 4).

This module and the optimization model are detailed in Sec. 4.

`GenerateAbstractConfigurationsModule` generates the channel protection configurations. Each channel protection device has a configuration that contains the rules. It takes the following inputs:

- `ChprotOptModel`: the channel protection optimization model class (described in Sec. 4);
- `LAChprotImplContainer`: the set of channel protection implementations;
- `LAsChprot`: the set of channel protection logical associations.

And provides the following outputs:

- `ChprotConfigurations`: the set of configured channel protection devices and related configurations.

Most of the inputs and outputs used by these modules are modelled using XML schemas (described in Sec. 4). Then using a particular Java technology (i.e., JAXB) the schemas are transformed into Java classes.

Implementation of the Filtering Module

This software module implements the process to generate the Filtering configurations as described in [2].
The software architecture is composed by the following Java classes:

- `ICSFilteringCoordinator`;
- `ProcessLAandLandscapeModule`;
- `GenerateAllImplementationsModule`;
- `ImplementationsPruningModule`;
- `OptimizationModule`;
- `GenerateAbstractConfigurationsModule`.

`ICSFilteringCoordinator` coordinates the process and the other modules, offers the public API (described into Sec. 3);

`ProcessLAandLandscapeModule` performs a set of analysis on landscape and on logical associations to generate the set of artifacts required by the next steps. The required inputs are:

- `LandscapeDescription`: the description of the landscape;
- `LogicalAssociationsRawContainer`: are the set of logical associations passed as input by LA Generation Service;
- `LogicalAssociationsRawDependencyContainer`: contains the set of dependencies among the logical associations (e.g., considering the First Matching Rule strategy and Deny Take Precedence strategy). These dependencies are provided by the Analysis Service.

The provided outputs are:

- `PhysicalTopologyGraph`: the physical topology of the landscape;
- `FilteringDeviceContainer`: the set of filtering devices;
- `LogicalAssociationsInputContainer`: this set contains the low-level logical associations (i.e., the information regarding IP addresses, ports and protocols). This process is required to derive ip addresses, ports and protocols when they are not specified by the LA Generation Service;
- `LAsFiltering`: the set of filtering logical associations.

`GenerateAllImplementationsModule` generates the filtering implementations. Each filtering implementation contains a set of filtering devices that could enforce a logical association. The required inputs are:

- `LAsFiltering`: the set of filtering logical associations;
- `PhysicalTopologyGraph`: the physical topology of the landscape;
- `LandscapeDescription`: the description of the landscape;
- `FilteringDeviceContainer`: the set of filtering devices.

The provided outputs are:

- `LogicalAssociationsPaths`: the set of alternative paths (each path contains the set of devices and related interfaces). Each path is associated to a logical association;
- `LAFilteringImplContainer`: the set of filtering implementations.

`ImplementationsPruningModule` performs pruning on a set of filtering implementations. The pruning task reduces the set of LA implementations which may be discarded either according to some heuristics or by allowing users to explicitly discard some of them. The required inputs are:

- `LAFilteringImplContainer`: the set of filtering implementations;
- `PruningCriterion`: the criterion to perform pruning.

The provided output is:

- `LAFilteringImplContainer`: the reduced set of filtering implementations.

`OptimizationModule` builds the optimization model and identifies the optimal solution. The required inputs are:

- `OptimizationProfile`: the optimization profile (i.e., the optimization target function);
- `SolverParameter`: the set of solver parameters (type of solver e.g., lp_solve or CPLEX, described in Sec. 4);
- `LAFilteringImplContainer`: the set of filtering implementations;
- `LAsFiltering`: the set of filtering logical associations;
- `WeightsOracle`: provides the set of weights for landscape elements and LAs.

The provided output is:

- `FilteringOptModel`: the filtering optimization model (described in Sec. 4).

This module and the optimization model are detailed in Sec. 4.

`GenerateAbstractConfigurationsModule` generates the filtering configurations. Each filtering device has a filtering configuration that contains the rules. The required inputs are:

- `FilteringOptModel`: the filtering optimization model class (described in Sec. 4);
- `LAFilteringImplContainer`: the set of filtering implementations;
- `LAsFiltering`: the set of filtering logical associations;
- `LogicalAssociationsPaths`: the set of alternative paths (each path contains the set of devices and associated interfaces). Each path is associated to a logical association.

The provided output is:

- `FilteringConfigurations`: the set of configured filtering devices and related configurations.

Most of the inputs and outputs used by these modules are modelled using XML schemas (described in Sec. 4). Then using a particular Java technology (i.e., JAXB) the schemas are transformed into Java classes.

Metrics

The Figure 2 shows the Infrastructure Configuration Service dependency graph, where nodes denote packages and edges denote dependencies, the edge's weight is the number of the underlying code dependencies.

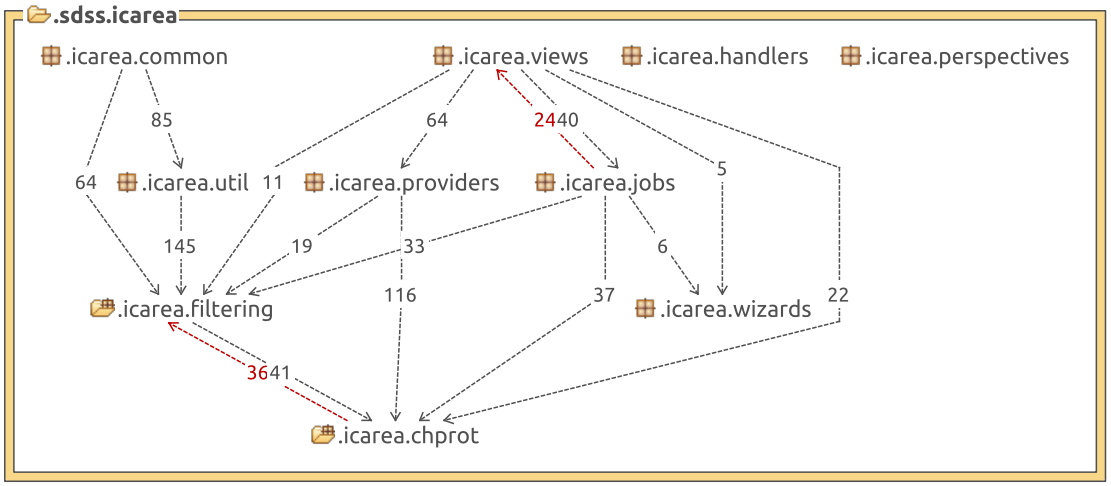

Figure 2: The Infrastructure Configuration Service dependency graph.

The Table 2 shows a series of metrics that can be used to estimate the complexity of the Infrastructure Configuration Service.

Metric	Value
Number of plug-ins	10
Number of packages	45
Number of classes	325
Number of methods	1969
Number of lines	22243
Number of lines (Drools rules)	21705
Average McCabe cyclomatic complexity	1.77

Table 2: The Infrastructure Configuration Service implementation metrics.

3 Public API

Data Protection API

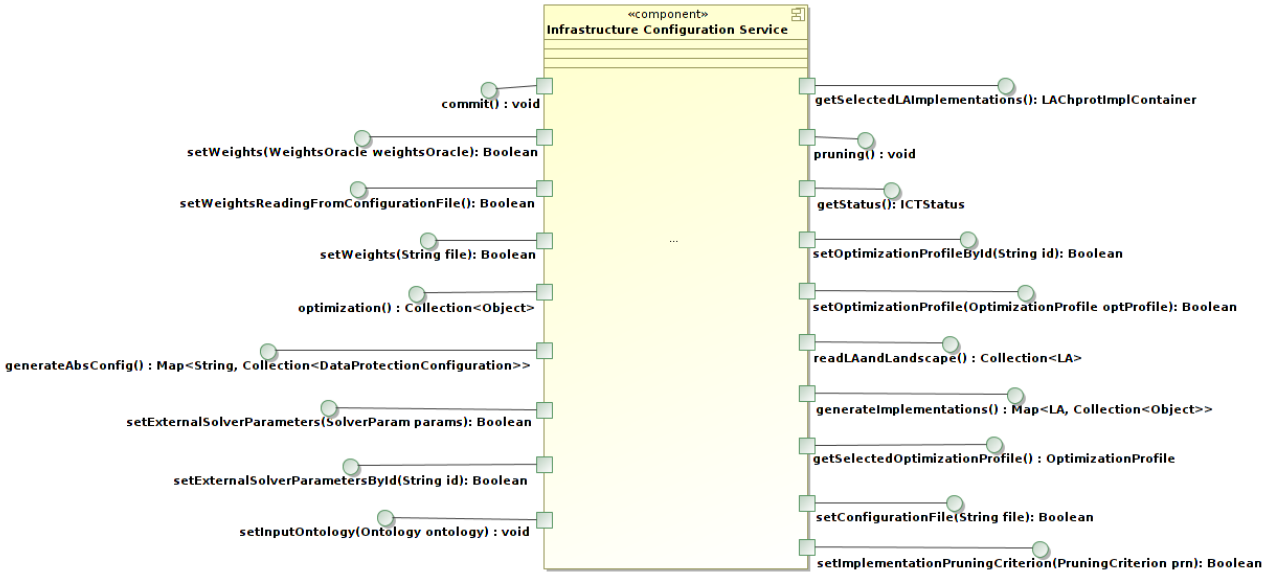

Figure 3: Infrastructure Configuration Service - Data protection API

Fig. 3 sketches the API exposed by the Infrastructure Configuration Service. The available methods are:

`ICTStatus getStatus()` return the status of the process;

`Boolean setOptimizationProfile(OptimizationProfile opt)` set the optimization profile that contains the optimization target functions;

`Boolean setOptimizationProfileById(String id)` set the optimization profile using the `id` to identify profile stored into configuration file;

`OptimizationProfile getSelectedOptimizationProfile()` return the selected optimization profile;

`Boolean setConfigurationFile(String file)` specify the configuration `file` that contains the following data: (2) the list of available target function types used to express the optimization objective(s); the list of predefined optimization profiles; (3) the information on the external repository; (4) the list of external solvers and available solver parameters;

`Boolean setImplementationPruningCriterion(PruningCriterion prn)` set the criterion to perform the pruning. The argument is a complex object (not implemented yet) used to define the set of technologies (and the related properties, e.g., end-to-end for IPsec) and heuristics parameters (e.g., maximum LA implementation costs, maximum device risk, minimum device performance);

`Boolean setWeights(WeightsOracle weightsOracle)` set the weights used for the devices and optimization objectives composition. The argument object contains: the reference to device(s) or to objectives composition and the value of a weight (expressed as a double);

`Boolean setWeights(String file)` set the weights reading from `file`;

`Boolean setWeightsReadingFromConfigurationFile()` set the weights reading from file (stored into configuration file);

`Boolean setExternalSolverParameters(SolverParam params)` set the solver to adopt and the related parameters. The argument contains the solver to adopt (as solver name), the timeout in seconds (as an integer number), and a list of string used to specify solver parameters. This list depends on the adopted solver;

`Boolean setExternalSolverParametersById(String id)` set the solver to adopt and the related parameters;

`void setInputOntology(Ontology ontology)` set the input `ontology` that contains landscape and LAs;

`Collection<LA> readLAandLandscape()` read LAs and landscape, create internal representations and returs the set of LAs;

`Map<LA, Collection<Object>> generateImplementations()` generate and return all the LA implementations;

`void pruning()` prune the LA implementations;

`LAChprotImplContainer getSelectedLAImplementations()` return the selected LA implementations that satisfy the pruning criteria;

`Collection<Object> optimization()` identify the optimal solution considering the available LA implementations and the optimization profile;

`Map<String, Collection<DataProtectionConfiguration>> generateAstractConfiguration` generate and return the abstract configurations;

`void commit()` inform the workflow manager that abstract configurations are available and can be stored in MoVE.

Filtering API

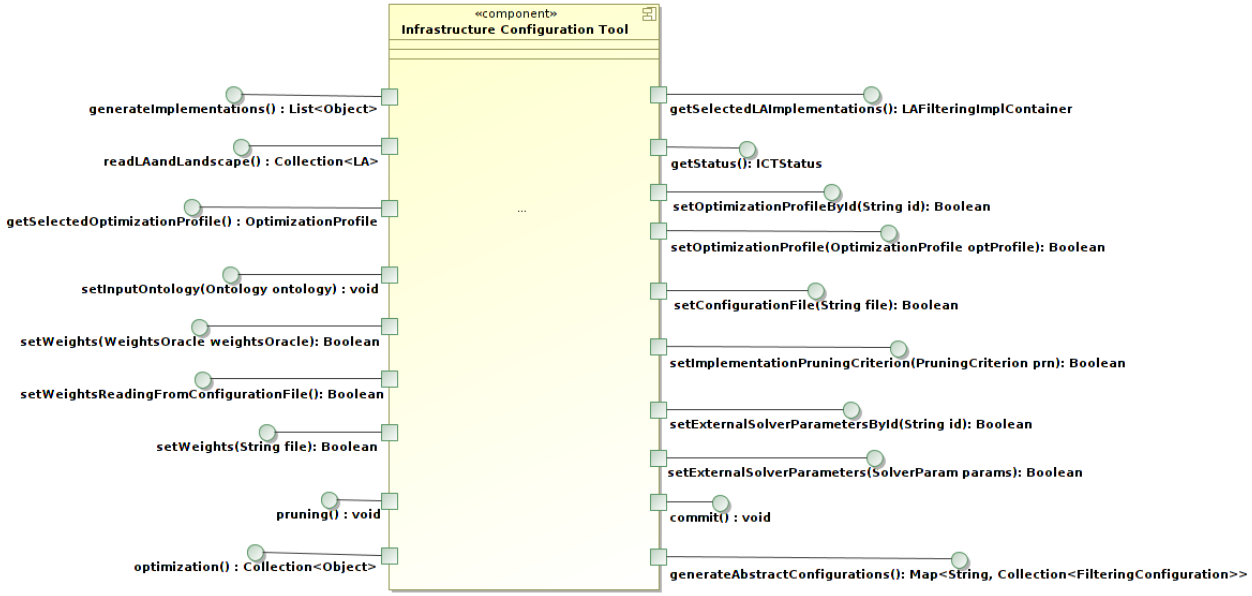

Figure 4: Infrastructure Configuration Service - Filtering API

Fig. 4 sketches the API exposed by the Infrastructure Configuration Service. The available methods are:

`ICTStatus getStatus()` return the status of the process;

Boolean setOptimizationProfile(OptimizationProfile opt) set the optimization profile that contains the optimization target functions;

Boolean setOptimizationProfileById(String id) set the optimization profile using the `id` to identify profile stored into configuration file;

OptimizationProfile getSelectedOptimizationProfile() return the selected optimization profile;

Boolean setConfigurationFile(String file) set the configuration `file` that contains the following data: the list of available target function types used to express the optimization objective(s); (2) the list of predefined optimization profiles; (3) the information on the external repository; (4) the list of external solvers and available solver parameters;

Boolean setImplementationPruningCriterion(PruningCriterion prn) set the criterion to perform the pruning. The argument is a complex object (not implemented yet) used to define the set of technologies (and the related properties, e.g., end-to-end for IPsec) and heuristics parameters (e.g., maximum LA implementation costs, maximum device risk, minimum device performance);

Boolean setWeights(WeightsOracle weightsOracle) set the weights used for the devices and optimization objectives composition. The argument object contains: the reference to device(s) or to objectives composition and the value of a weight (expressed as a double);

Boolean setWeights(String file) set the weights reading from `file`;

Boolean setWeightsReadingFromConfigurationFile() set the weights reading from file (stored into configuration file);

Boolean setExternalSolverParameters(SolverParam params) set the solver to adopt and the related parameters. The argument contains the solver to adopt (as solver name), the timeout in seconds (as an integer number), and a list of string used to specify solver parameters. This list depends on the adopted solver;

Boolean setExternalSolverParametersById(String id) set the solver to adopt and the related parameters;

void setInputOntology(Ontology ontology) set the input `ontology` that contains landscape and LAs;

Collection<LA> readLAandLandscape() read LAs and landscape, create internal representations and return the set of LAs;

Map<LA, Collection<Object>> generateImplementations() generate and return all the LA implementations;

void pruning() prune the LA implementations;

LAFilteringImplContainer getSelectedLAImplementations() return the selected LA implementations that satisfy the pruning criteria;

Collection<Object> optimization() identify the optimal solution considering the available LA implementations and the optimization profile;

Map<String, Collection<FilteringConfiguration>> generateAstractConfigurations() generate and return the abstract configurations;

void commit() inform the workflow manager that abstract configurations are available and can be stored in MoVE.

4 Optimization models design

Data Protection optimization model design

The `OptimizationModule` (shown in Fig. 5) is composed by the following submodules:

- `BuildAbstractModel`;
- `OptModelBuilder`;
- `OptModelSolver`;
- `SolMapping`.

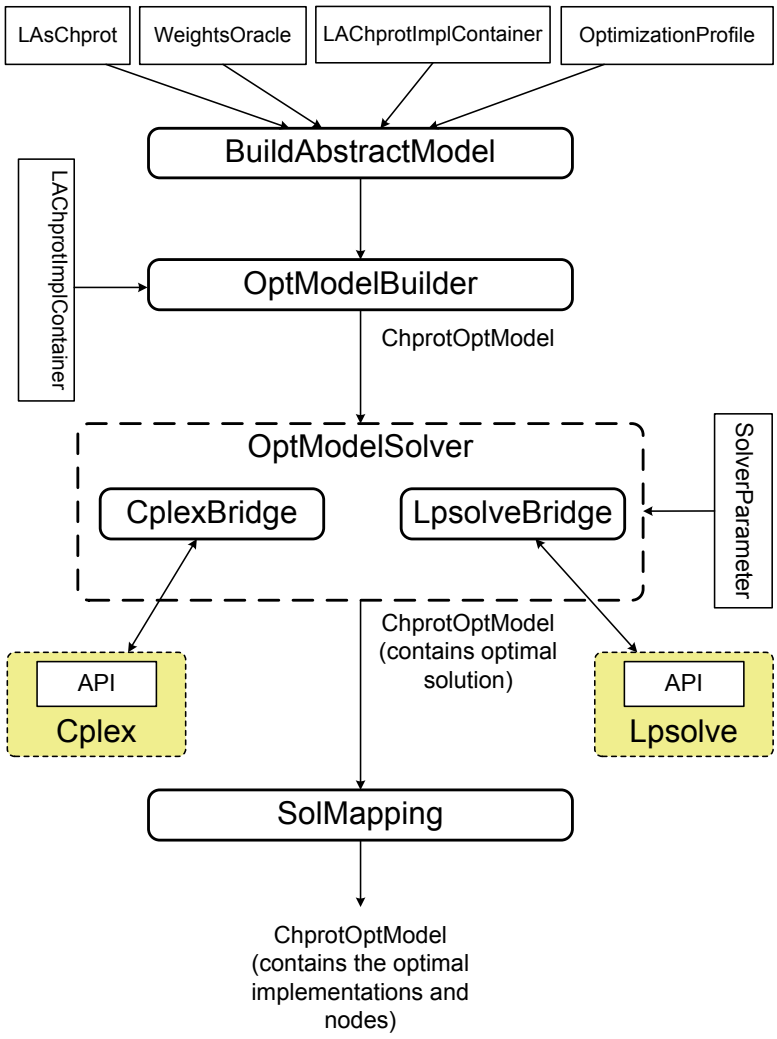

Figure 5: Data Protection optimization module

`BuildAbstractModel` creates and populates an abstract model. This submodule offers the following methods:

- **`setContext`**`(LAsChprot lacontainer, OptimizationProfile optProfile, WeightsOr woracle, LAChprotImplContainer implc)`: set the context passing required objects. The context method provides the inputs for a specific task, in this case to create and populate the abstract model;
- **`run`**`()`: create and populate the knowledge base with the required objects and start the inference process;

- **getOutput**(): return ChprotOptModel.

OptModelBuilder create an object representation of the mathematical model starting from the abstract model. The mathematical model is built using a set of Drools rules (*ChProtOptBuilder.drl*). In particular, the submodule creates and populates the knowledge base managed by the set of rules and by Drools engine. This submodule offers the following methods:

- **setContext**(ChprotOptModel opt, LAChprotImplContainer implContainer): set the context (i.e., the inputs) passing required objects;
- **run**(): create and populate the knowledge base with the objects contained in the abstract model and starts the inference process;
- **getOutput**(): return the ChprotOptModel populated with the mathematical model.

OptModelSolver is an abstract submodule that transforms the mathematical model into a problem representation for a specific solver (we support CPLEX and lp_solve). The transformation process is performed by a *bridge* object that directly interacts with a solver by using the specific API. Therefore we provide two different *bridges*: CplexBridge and LpsolveBridge. The offered methods are:

- **setContext**(ChprotOptModel optModel, SolverParameter solverParams): set the context (i.e., the inputs) with the model and parameters for the specific solver;
- **run**(): build and solve the mathematical model by using the specific solver API;
- **getOutput**(): return the model (ChprotOptModel) that contains the optimal solution.

SolMapping after solving the model using one of the above solvers, this submodule updates the ChprotOptModel identifying which set of channel protection devices implement a particular logical association (LAsChprot) using a particular technology (e.g., SSL/TLS, IPsec, etc.). This process is performed using solMapping.drl, another set of Drools rules. The offered methods are:

- **setContext**(ChprotOptModel opt, LAChprotImplContainer implContainer): set the context with the model and the channel protection implementations;
- **run**(): create and populate the knowledge base and starts the inference process that fill the model to identify which set of channel protection devices implement a LAsChprot;
- **getOutput**(): return the updated model ChprotOptModel.

Optimization model anatomy

To represent our optimization model we start from XML Schema and then, using the JAXB *xjc* compiler we generate all the related java classes. This approach makes it possible to use XML schema to formalize the model and automatic mechanisms of JAXB (marshalling/unmarshalling) to create, read and save objects as XML. The optimization model is composed by two parts:

- *abstract* part: that contains high level information on optimization goal(s) and constraints on devices, links, LAs, LAs implementations;
- *implementation* part: that contains low level information to represent the mathematical model.

The main schema, ChprotOptModel (Listing 1), describes the optimization model and contains reference to other elements.

Listing 1: ChprotOptModel.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html
```

```
   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema elementFormDefault="qualified"
  targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
  xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="LinearExpression.xsd" />
  <xs:include schemaLocation="LAsChprot.xsd" />
  <xs:include schemaLocation="ALChprotConstraint.xsd" />
  <xs:include schemaLocation="ILConstraint.xsd" />
  <xs:include schemaLocation="Variable.xsd" />
  <xs:include schemaLocation="ChprotNode.xsd" />

  <xs:element name="ChprotOptModel">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="chprotObjectiveFunction"
          type="posecco_channelprotection:ChprotObjectiveFunction" />
        <xs:element name="optFunctionality"
          type="posecco_channelprotection:OptFunctionality" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="containsLAChprot" type="posecco_channelprotection:LAChprot" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="containsALConstraint"
          type="posecco_channelprotection:ALChprotConstraint" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="containsILConstraint" type="posecco_channelprotection:ILConstraint" />
        <xs:element maxOccurs="unbounded" minOccurs="0" name="binvariable"
          type="xs:string" />
        <xs:element maxOccurs="unbounded" minOccurs="0" name="intvariable"
          type="xs:string" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="floatvariable" type="xs:string" />
        <xs:element maxOccurs="unbounded" minOccurs="0" name="variable"
          type="posecco_channelprotection:Variable" />
        <xs:element maxOccurs="unbounded" minOccurs="0" name="chprotNode"
          type="posecco_channelprotection:ChprotNode" />
        <xs:element name="objFunctionValue" type="xs:double" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:complexType name="ChprotObjectiveFunction">
    <xs:sequence>
      <xs:element name="chprotOptimizationGoal"
        type="posecco_channelprotection:ChprotOptimizationGoal" />
      <xs:element maxOccurs="unbounded" minOccurs="1"
        name="containsLinearExpression"
        type="posecco_channelprotection:LinearExpression" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="ChprotOptimizationGoal">
    <xs:restriction base="xs:string">
      <xs:enumeration value="MIN" />
      <xs:enumeration value="MAX" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OptFunctionality">
    <xs:restriction base="xs:string">
      <xs:enumeration value="MIN_IMPLEMENTATION_COST_AND_MAX_THROUGHPUT" />
      <xs:enumeration value="MIN_RISK" />
      <xs:enumeration value="MAX_THROUGHPUT_WITH_VPN_PROFILE" />
    </xs:restriction>
```

```
  </xs:simpleType>
</xs:schema>
```

The *abstract* part contains the following attributes:

- `ALChprotConstraint`: adopts a particular abstract constraint defined into the schema of Listing 2;
- `OptFunctionality`: expresses the optimization type (e.g., maximize throughput and minimize implementation cost);
- `containsLAChprot`: adopts a LA channel protection that is expressed using the schema of Listing 3.

The *implementation* part, i.e., the mathematical model, contains the following attributes:

- `containsILConstraint`: adopts a particular implementation constraint defined into the schema of Listing 17;
- `ChprotObjectiveFunction`: expresses the main goal (MIN/MAX) of the target function and contains a set of linear expressions (to manage sub-objective);
- `variable`: expresses a variable and it is defined into the schema of Listing 19;
- `binvariable`: used to define the name of a binary variable;
- `intvariable`: used to define the name of an integer variable;
- `floatvariable`: used to dene the name of an float variable;
- `objFunctionValue`: represent the value calculated by the solver.

`ALChprotConstraint` defined the structure of a `LAChprotImplConstraint` and its extension `LAChprotImplRe` In particular `LAChprotImplRequired` makes it possible to specify when a particular `LAChprotImpl` is required or not. The schema of `ALChprotConstraint` is defined into the Listing 2;

`LAsChprot` defines the structure of the logical association in our model (Listing 3);

`ILConstraint` defines the constraint structure inside our model. It has the following attributes:

- `containsTerm` is a `Term` (Listing 20);
- `containsConstraintOperator` is a `ConstraintOperator` (Listing 21);
- `containsLinearExpression` is a `LinearExpression` (Listing 22).

`ChprotNode` defines the structure of the related object. The schema is defined into Listing 9;

`Variable` defines the structure of the related object. It is composed by and id and a value. The schema is defined into Listing 19;

`Weight` defines the structure of the related object, it has an identifier and a value. The schema is defined into Listing 10;

`Link` defines the structure of the related object. The schema is defined into Listing 23. It is composed by the following attributes:

- `hasSourceId`: the source identifier of an landscape object;
- `hasDestinationId`: the destination identifier of an landscape object;
- `hasSourceInterfaceId`: the source identifier of the object netwokr interface;
- `hasDestinationInterfaceId`: the destination identifier of the object netwokr interface;
- `weight`: the cost, e.g., to express performance.

`Term` defines the structure of the related object. It has an identifier, a type (that can be integer or binary) and a weight. The schema is defined into Listing 20;

`LinearExpression` defines the structure of the related object. It is composed by a set of terms that are implicitly bounded by an operator. The schema is defined into Listing 22;

`ConstraintOperator` defines the structure of the related object. An operator could be one of the following: `Equal`, `Less Equal` or `Greater Equal`. The schema is defined into Listing 21;

`LAChprotImpl` defines the structure of a LA channel protection implementation. Each `LAChprotImpl` must specify which `ChprotNode` adopts and which `Link` will be used. The schema is defined into Listing 12;

`OptimizationProfile` defines the structure of the optimization profile. The schema is defined into Listing 25;

`SolverParameter` defines the structure of the parameters passed as argument to the solver. The schema is defined into Listing 25.

Listing 2: ALChprotConstraint.xsd

frame

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="LAChprotImpl.xsd" />

   <xs:element name="ALChprotConstraintContainer">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="chprotConstraint"
               type="posecco_channelprotection:ALChprotConstraint" minOccurs="0"
               maxOccurs="unbounded" />
         </xs:sequence>
      </xs:complexType>
   </xs:element>

   <xs:complexType name="ALChprotConstraint" abstract="true">
      <xs:attribute name="id" type="xs:string" />
   </xs:complexType>

   <xs:complexType name="LAChprotImplConstraint">
      <xs:complexContent>
         <xs:extension base="posecco_channelprotection:ALChprotConstraint">
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>



   <xs:complexType name="LAChprotImplRequired">
      <xs:complexContent>
```

```
      <xs:extension base="posecco_channelprotection:LAChprotImplConstraint">
        <xs:sequence>
          <xs:element name="isRequired" type="xs:boolean" />
          <xs:element name="LAChprotImpl"
            type="posecco_channelprotection:LAChprotImpl" minOccurs="1"
            maxOccurs="unbounded" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>



</xs:schema>
```

Listing 3: LAsChprot.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="LAChprotImpl.xsd" />
   <xs:include schemaLocation="Weight.xsd" />

   <xs:element name="LAsChprot">
     <xs:complexType>
       <xs:sequence>
         <xs:element name="la" type="posecco_channelprotection:LAChprot"
           minOccurs="0" maxOccurs="unbounded" />
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:complexType name="LAChprot">
     <xs:sequence>
       <xs:element name="traffic" type="posecco_channelprotection:Weight" />
       <xs:element name="cardinality" type="xs:int" />
       <xs:element name="source" type="xs:string" />
       <xs:element name="sourceType" type="xs:string" />
       <xs:element name="networkElementSourceId" type="xs:string" />
       <xs:element name="sourceITResourceId" type="xs:string" />
       <xs:element name="destination" type="xs:string" />
       <xs:element name="destinationType" type="xs:string" />
       <xs:element name="networkElementDestinationId" type="xs:string" />
       <xs:element name="destinationITResourceId" type="xs:string" />
       <xs:element name="order" type="xs:int" />
       <xs:element name="connectionStateType" type="xs:string" />
       <xs:element name="IPVersion" type="xs:string" /> <!-- ipv4 or ipv6 -->
       <xs:element name="L3ProtocolType" type="xs:string" /> <!-- e.g., icmp -->
       <xs:element name="L4ProtocolType" type="xs:string" />
       <xs:element name="L4SourcePort" type="xs:string" />
       <xs:element name="L4DestinationPort" type="xs:string" />
       <xs:element name="L7ProtocolType" type="xs:string" />
       <xs:element name="L7Method" type="xs:string" />
       <xs:element name="url" type="xs:string" />
```

```
      <xs:element name="technology" type="xs:string" default="nd"/>
      <xs:element name="isRelatedTo" type="posecco_channelprotection:LAChprot"
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="hasLAImplId" type="xs:string" minOccurs="0"
        maxOccurs="unbounded" />
      <xs:element name="hasPrecedence" type="xs:string"
        minOccurs="0" maxOccurs="unbounded" /> <!-- chprot LA that precedes this -->
      <xs:element name="laRawId" type="xs:string" />
      <xs:element name="confidentiality" type="xs:boolean" />
      <xs:element name="integrity" type="xs:boolean" />
      <xs:element name="done" type="xs:boolean" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>


</xs:schema>
```

Listing 4: ILConstraint.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="Term.xsd" />
   <xs:include schemaLocation="ConstraintOperator.xsd" />
   <xs:include schemaLocation="LinearExpression.xsd" />

   <xs:element name="ILConstraintContainer">
     <xs:complexType>
       <xs:sequence>
         <xs:element name="ILconstraint" type="posecco_channelprotection:ILConstraint"
           minOccurs="0" maxOccurs="unbounded" />
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:complexType name="ILConstraint">
     <xs:sequence>
       <xs:element name="containsTerm" type="posecco_channelprotection:Term" />
       <xs:element name="containsConstraintOperator"
         type="posecco_channelprotection:ConstraintOperator" />
       <xs:element name="containsLinearExpression"
         type="posecco_channelprotection:LinearExpression" />
     </xs:sequence>
     <xs:attribute name="id" type="xs:string" />
   </xs:complexType>


</xs:schema>
```

Listing 5: Variable.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
```

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">

  <xs:element name="VariablesContainer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="variable" type="posecco_channelprotection:Variable"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Variable">
    <xs:sequence>
      <xs:element name="value" type="xs:double" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>

</xs:schema>
```

Listing 6: Term.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">
  <xs:include schemaLocation="Weight.xsd" />

  <xs:element name="TermContainer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="term" type="posecco_channelprotection:Term"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Term">
    <xs:sequence>
      <xs:element name="type" type="xs:string" />
      <xs:element name="value" type="xs:double" />
      <xs:element name="weight" type="posecco_channelprotection:Weight" />
    </xs:sequence>
```

```
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>

</xs:schema>
```

Listing 7: ConstraintOperator.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">

  <xs:simpleType name="ConstraintOperator">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Equal" />
      <xs:enumeration value="Lower_Equal" />
      <xs:enumeration value="Greater_Equal" />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Listing 8: LinearExpression.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">
  <xs:include schemaLocation="Term.xsd" />
  <xs:include schemaLocation="Weight.xsd" />


  <xs:element name="LinearExpressionContainer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="linearExpression"
          type="posecco_channelprotection:LinearExpression" minOccurs="0"
          maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="LinearExpression">
    <xs:sequence>
      <xs:element name="containsTerm" type="posecco_channelprotection:Term"
```

```
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="hasWeight" type="posecco_channelprotection:Weight" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>


</xs:schema>
```

Listing 9: ChprotNode.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="Weight.xsd" />
   <xs:include schemaLocation="LAChprotImpl.xsd" />

   <xs:element name="ChprotNodeContainer">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="node" type="posecco_channelprotection:ChprotNode"
               minOccurs="0" maxOccurs="unbounded" />
         </xs:sequence>
      </xs:complexType>
   </xs:element>

   <xs:complexType name="ChprotNode">
      <xs:sequence>
         <xs:element name="ImplCostIpsec" type="posecco_channelprotection:Weight" />
         <xs:element name="ImplCostSsl" type="posecco_channelprotection:Weight" />
         <xs:element name="PerfIpsec" type="posecco_channelprotection:Weight" />
         <xs:element name="PerfSsl" type="posecco_channelprotection:Weight" />
         <xs:element name="capability" type="xs:string" minOccurs="0"
            maxOccurs="unbounded" />
         <xs:element name="chprotService" type="posecco_channelprotection:ChprotService"
            minOccurs="0" maxOccurs="unbounded" />
         <xs:element name="chprotNodeImplementation"
            type="posecco_channelprotection:ChprotNodeImplementation" minOccurs="0"
            maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" />
   </xs:complexType>

   <xs:complexType name="ChprotService">
      <xs:sequence>
         <xs:element name="capability" type="xs:string" minOccurs="1"
            maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" />
      <xs:attribute name="L4port" type="xs:string" />
      <xs:attribute name="L4proto" type="xs:string" />
      <xs:attribute name="L7proto" type="xs:string" />
      <xs:attribute name="url" type="xs:string" />
      <xs:attribute name="serviceMustBeInstalled" type="xs:boolean" />
```

```
  </xs:complexType>

  <xs:complexType name="ChprotNodeImplementation">
    <xs:sequence>
      <xs:element name="LAChprotImplId" type="xs:string" />
      <xs:element name="chprotImplPart"
        type="posecco_channelprotection:LAChprotImplPart" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Listing 10: Weight.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">

  <xs:element name="WeightsContainer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="weight" type="posecco_channelprotection:Weight"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Weight">
    <xs:sequence>
      <xs:element name="value" type="xs:double" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>

</xs:schema>
```

Listing 11: Link.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">
```

```
  <xs:include schemaLocation="Weight.xsd" />

  <xs:element name="LinkContainer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="link" type="posecco_channelprotection:Link"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Link">
    <xs:sequence>
      <xs:element name="weight" type="posecco_channelprotection:Weight" />
      <xs:element name="hasSourceId" type="xs:string" />
      <xs:element name="hasSourceInterfaceId" type="xs:string" />
      <xs:element name="hasDestinationId" type="xs:string" />
      <xs:element name="hasDestinationInterfaceId" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>

</xs:schema>
```

Listing 12: LAChprotImpl.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/model/core"
   xmlns:posecco_channelprotection="http://posecco.eu/sdss/icarea/chprot/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="ChprotNode.xsd" />
   <xs:include schemaLocation="Link.xsd" />
   <xs:include schemaLocation="LAsChprot.xsd" />
   <xs:include schemaLocation="Weight.xsd" />

   <xs:element name="LAChprotImplContainer">
     <xs:complexType>
       <xs:sequence>
         <xs:element name="laChprotImpls" type="posecco_channelprotection:LAChprotImpl"
           minOccurs="0" maxOccurs="unbounded" />
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:complexType name="LAChprotImpl">
     <xs:sequence>
       <xs:element name="technology" type="xs:string" />
       <xs:element name="mode" type="xs:string" /> <!-- end-to-end,
         site-to-site, remote-access -->
       <xs:element name="confidentiality" type="xs:boolean" default="true" />
       <xs:element name="integrity" type="xs:boolean" default="true" />
       <xs:element name="hasPart"
         type="posecco_channelprotection:LAChprotImplPart" minOccurs="0"
         maxOccurs="unbounded" />
       <xs:element name="LAChprot" type="posecco_channelprotection:LAChprot" minOccurs="0"
```

```
        maxOccurs="unbounded" />
      <xs:element name="motivation" type="xs:string" />
      <xs:element name="selected" type="xs:boolean" default="false"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>

  <xs:complexType name="LAChprotImplPart">
    <xs:sequence>
      <xs:element name="LAChprot" type="posecco_channelprotection:LAChprot" />
      <xs:element name="destination" type="posecco_channelprotection:ChprotNode" />
      <xs:element name="source" type="posecco_channelprotection:ChprotNode" />
      <xs:element name="pep1" type="posecco_channelprotection:ChprotNode" />
      <xs:element name="pep1ExternalIPv4Address" type="xs:string" minOccurs="0"
        maxOccurs="unbounded" /> <!-- pep1 external IPv4 address -->
      <xs:element name="pep1InternalIPv4Address" type="xs:string" minOccurs="0"
        maxOccurs="unbounded" /> <!-- pep1 external IPv4 address -->
      <xs:element name="pep2" type="posecco_channelprotection:ChprotNode" />
      <xs:element name="pep2ExternalIPv4Address" type="xs:string" minOccurs="0"
        maxOccurs="unbounded" /> <!-- pep2 external IPv4 address -->
      <xs:element name="pep2InternalIPv4Address" type="xs:string" minOccurs="0"
        maxOccurs="unbounded" /> <!-- pep2 external IPv4 address -->
      <xs:element name="service" type="posecco_channelprotection:ChprotService" />
      <xs:element name="adoptsLink" type="posecco_channelprotection:Link"
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="traffic" type="posecco_channelprotection:Weight" />
      <xs:element name="performance" type="posecco_channelprotection:Weight" />
      <xs:element name="risk" type="posecco_channelprotection:Weight" />
      <xs:element name="economic-cost" type="posecco_channelprotection:Weight" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>

</xs:schema>
```

Listing 13: ToolConfiguration.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/chprot/coordinator/model/core"
   xmlns:posecco_channelprotection_coordinator="http://posecco.eu/sdss/icarea/
chprot/coordinator/model/core"
   elementFormDefault="qualified">
   <!-- <xs:include schemaLocation="OptimizationProfile.xsd" /> <xs:include
     schemaLocation="SolverParameter.xsd" /> -->

   <xs:element name="ToolConfiguration">
     <xs:complexType>
       <xs:sequence>
         <xs:element name="predefinedOptimizationProfile"
           type="posecco_channelprotection_coordinator:OptimizationProfile"
           minOccurs="0" maxOccurs="unbounded" />
         <xs:element name="externalRepoInfo"
           type="posecco_channelprotection_coordinator:ExternalRepositoryInfo" />
         <xs:element name="predefinedExternalSolverInfo"
           type="posecco_channelprotection_coordinator:SolverParameter"
```

```
        minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="internalModelsInfo"
        type="posecco_channelprotection_coordinator:InternalModelsInfo" />
    </xs:sequence>

    <xs:attribute name="configurationId" type="xs:string" />
  </xs:complexType>
</xs:element>

<xs:complexType name="OptimizationProfile">
  <xs:sequence>
    <xs:element name="optimizationMainObjective" type="xs:string" />
    <xs:element name="optimizationObjective"
      type="posecco_channelprotection_coordinator:OptimizationObjective"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="profileId" type="xs:string" />
  <xs:attribute name="type" type="xs:string" /> <!-- filtering or chprot -->
</xs:complexType>

<xs:complexType name="OptimizationObjective">
  <xs:sequence>
    <xs:element name="subtargetFunctionName" type="xs:string" />
    <xs:element name="subtargetFunctionWeight" type="xs:double" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" />
</xs:complexType>

<xs:complexType name="SolverParameter">
  <xs:sequence>
    <xs:element name="solverName" type="xs:string" />
    <xs:element name="solverPath" type="xs:string" />
    <xs:element name="timemout" type="xs:string" />
    <xs:element name="solverParams" type="xs:string"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" />
</xs:complexType>

<xs:complexType name="ExternalRepositoryInfo">
  <xs:sequence>
    <xs:element name="url" type="xs:string" />
    <xs:element name="protocol" type="xs:string" />
    <xs:element name="port" type="xs:string" />
    <xs:element name="username" type="xs:string" />
    <xs:element name="password" type="xs:string" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="InternalModelsInfo">
  <xs:sequence>
    <xs:element name="WeightsOracleFile" type="xs:string" />
    <xs:element name="OptimizationProfileFile" type="xs:string" />
    <xs:element name="LogicalAssociationsRawContainerFile"
      type="xs:string" />
    <xs:element name="LogicalAssociationsInputContainerFile"
      type="xs:string" />
    <xs:element name="LogicalAssociationsRawDependencyContainerFile"
      type="xs:string" />
    <xs:element name="LAsChprotFile" type="xs:string" />
    <xs:element name="ChprotNodeFile" type="xs:string" />
    <xs:element name="LandscapeDescriptionFile" type="xs:string" />
    <xs:element name="LAChprotImplContainerFile" type="xs:string" />
    <xs:element name="ChprotOptModelFile" type="xs:string" />
    <xs:element name="ChprotConfigurationsFile" type="xs:string" />
    <xs:element name="OntologyInputFile" type="xs:string" />
```

```
      <xs:element name="OntologyInputLASplittedFile" type="xs:string" />
      <xs:element name="OntologyOutputFile" type="xs:string" />
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Filtering optimization model design

The `OptimizationModule` (shown in Fig. 6) is composed by the following submodules:

- `BuildAbstractModel`;
- `OptModelBuilder`;
- `OptModelSolver`;
- `SolMapping`.

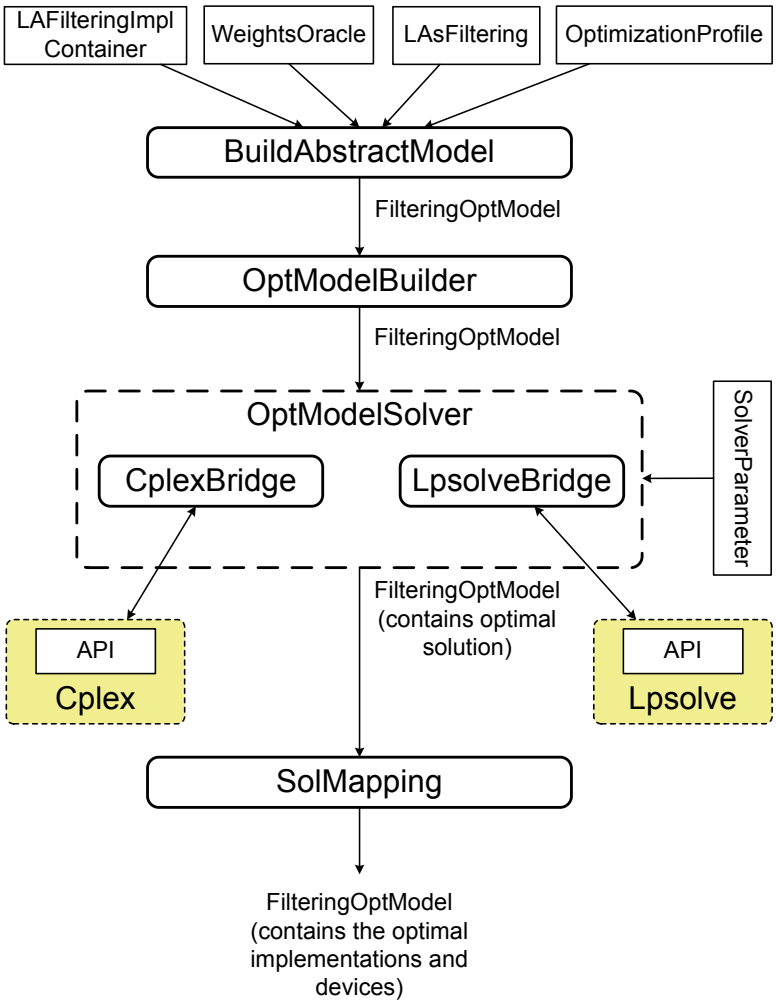

Figure 6: Filterijg Optimization module

`BuildAbstractModel` creates and populates an abstract model. This submodule offers the following methods:

- **`setWeights`**`(LAFilteringImplContainer implc, WeightsOracle wo)`: set the weights for FilteringDevice, Link and return `LAFilteringImplContainer`;

- **buildModel**(LAsFiltering lasFiltering, OptimizationProfile optProfile): create the abstract model and the FilteringOptModel (populated with LAFiltering). Return FilteringOptModel.

OptModelBuilder creates an object representation of the mathematical model starting from the abstract model. The mathematical model is built using a set of Drools rules (*Filtering_max_performance.drl*). In particular, the submodule creates and populates the knowledge base managed by the set of rules and by Drools engine. This submodule offers the following methods:

- **setContext**(FilteringOptModel opt, LAFilteringImplContainer implContainer): set the context passing required objects. The context method provides the inputs for a specific task, in this case to build the abstract model;
- **run**(): create and populate the knowledge base with the objects contained in the abstract model and starts the inference process;
- **getOutput**(): return the model enriched (FilteringOptModel) with the mathematical model.

OptModelSolver is an abstract submodule that transforms the mathematical model into a problem representation for a specific solver (we support CPLEX and lp_solve). The transformation process is performed by a *bridge* object that directly interacts with a solver using specific API. Therefore we provide two different *bridges*: CplexBridge and LpsolveBridge. The offered methods are:

- **setContext**(FilteringOptModel optModel, SolverParameter solverParams): set the context (i.e., the inputs) with the model and parameters for the specific solver;
- **run**(): build the mathematical model for specific solver and solves it;
- **getOutput**(): return the enriched model (FilteringOptModel) expressing the optimal solution.

SolMapping after solving the model using one of the above solvers, this submodule enriches the FilteringOptModel identifying which set of filtering devices implement a particular logical association (LAsfiltering). This process is performed using solMapping.drl, another set of Drools rules. The offered methods are:

- **setContext**(FilteringOptModel opt, LAFilteringImplContainer implContainer): set the context with the model and the filtering implementations;
- **run**(): create the knowledge base, populate it and start the inference process that fill the model to identify which set of filtering devices implement a LAsFiltering;
- **getOutput**(): returns the updated model FilteringOptModel.

Optimization model anatomy

To represent our optimization model we start from XML Schema and then, using the JAXB *xjc* compiler we generate all the java classes. This approach makes it possible to use XML schema to formalize the model and automatic mechanisms of JAXB (marshalling/unmarshalling) to create, read and save objects as XML. The optimization model is composed by two parts:

- *abstract* part: that contains high level information on optimization goal(s) and constraints on devices, links, LAs, LAs implementations;
- *implementation* part: that contains low level information to represent the mathematical model.

The main schema, FilteringOptModel (Listing 14), describes the optimization model and contains reference to other elements.

Listing 14: FilteringOptModel.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema elementFormDefault="qualified"
   targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
   xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="LAsFiltering.xsd" />
  <xs:include schemaLocation="ALFilteringConstraint.xsd" />
  <xs:include schemaLocation="ILConstraint.xsd" />
  <xs:include schemaLocation="FilteringDevice.xsd" />
  <xs:include schemaLocation="Variable.xsd" />
  <xs:include schemaLocation="LinearExpression.xsd" />

  <xs:element name="FilteringOptModel">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="filteringObjectiveFunction"
          type="posecco_filtering:FilteringObjectiveFunction" />
        <xs:element name="optFunctionality" type="posecco_filtering:OptFunctionality" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="containsLADeny" type="posecco_filtering:LAFilteringDeny" />
        <xs:element maxOccurs="unbounded" minOccurs="1"
          name="containsLAAllow" type="posecco_filtering:LAFilteringAllow" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="containsALConstraint"
          type="posecco_filtering:ALFilteringConstraint" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="containsLAFilteringImplRequired"
          type="posecco_filtering:LAFilteringImplRequired" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="containsILConstraint" type="posecco_filtering:ILConstraint" />
        <xs:element maxOccurs="unbounded" minOccurs="0"
          name="filteringDevice" type="posecco_filtering:FilteringDevice" />
        <xs:element maxOccurs="unbounded" minOccurs="0" name="binvariable"
          type="xs:string" />
        <xs:element maxOccurs="unbounded" minOccurs="0" name="intvariable"
          type="xs:string" />
        <xs:element maxOccurs="unbounded" minOccurs="0" name="floatvariable"
          type="xs:string" />
        <xs:element maxOccurs="unbounded" minOccurs="0" name="variable"
          type="posecco_filtering:Variable" />
        <xs:element name="objFunctionValue" type="xs:double" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:complexType name="FilteringObjectiveFunction">
    <xs:sequence>
      <xs:element name="filteringOptimizationGoal"
        type="posecco_filtering:FilteringOptimizationGoal" />
      <xs:element maxOccurs="unbounded" minOccurs="1"
        name="containsLinearExpression" type="posecco_filtering:LinearExpression" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="FilteringOptimizationGoal">
    <xs:restriction base="xs:string">
```

```
      <xs:enumeration value="MIN" />
      <xs:enumeration value="MAX" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OptFunctionality">
    <xs:restriction base="xs:string">
      <xs:enumeration value="MAX_LINK_PERFORMANCE" />
      <xs:enumeration value="MAX_FILTERING_PERFORMANCE" />
      <xs:enumeration value="MIN_SECURITY_RISK" />
      <xs:enumeration value="FILTERING_SORTING_RULES_CONSIDERING_THROUGHPUT" />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

The *abstract* part contains the following attributes:

- `containsLAFilteringImplRequired`: expresses, using the schema of Listing 15, specifies when a filtering logical association must be adopted during optimization. This is an high level constraint that is required by the administrator.
- `optFunctionality`: expresses the optimization type (e.g., maximize filtering performance);
- `containsLADeny`: adopts a particular LA filtering deny that is expressed using the schema of Listing 16;
- `containsLAAllow`: adopts a particular LA filtering allow that is expressed using the schema of Listing 16;
- `containsALConstraint`: adopts a particular abstract constraint defined into the schema of Listing 15.

The *implementation* part, i.e., the mathematical model, contains the following attributes:

- `containsILConstraint`: adopts a particular implementation constraint defined into the schema of Listing 17;
- `filteringDevice`: are the set of available filtering devices that can be used to implement the logical associations. This element is defined into the schema of Listing 18;
- `filteringObjectiveFunction`: expresses the main goal (MIN/MAX) of the target function and contains a set of linear expressions (to manage sub-objective);
- `variable`: expresses a variable and it is defined into the schema of Listing 19;
- `binvariable`: used to define the name of a binary variable;
- `intvariable`: used to define the name of an integer variable
- `objFunctionValue`: represent the value calculated by the solver.

`ALFilteringConstraint` defined the structure of a `LAFilteringImplConstraint` and its extensions `LAFilteringImplRequired`, `LAFilteringRedundancyConstraint`, `FilteringDeviceRequ` `FilteringLinkConstraint`. In particular `LAFilteringRequired` allows to specify when a particular `LAFilteringImpl` is required or not. On the contrary, `FilteringDeviceRequired` is used to specify that a particular `LAFiltering` must be implemented or not on a specific `FilteringDevice`. The schema of `ALFilteringConstraint` is defined into the Listing 15;

`LAsFiltering` defines the structure of the logical association in our model (Listing 16);

`ILConstraint` defines the constraint structure inside our model. It has the following attributes:

- `containsTerm` is a `Term` (Listing 20);
- `containsConstraintOperator` is a `ConstraintOperator` (Listing 21);
- `containsLinearExpression` is a `LinearExpression` (Listing 22).

`FilteringDevice` defines the structure of the related object. It is composed by an id and a weight (the cost, e.g., to express performance). The schema is defined into Listing 18;

`Variable` defines the structure of the related object. It is composed by and id and a value. The schema is defined into Listing 19;

`Link` defines the structure of the related object. The schema is defined into Listing 23. It is composed by the following attributes:

- `hasSourceId`: the source identifier of an landscape object;
- `hasDestinationId`: the destination identifier of an landscape object;
- `hasSourceInterfaceId`: the source identifier of the object netwokr interface;
- `hasDestinationInterfaceId`: the destination identifier of the object netwokr interface;
- `weight`: the cost, e.g., to express performance.

`Term` defines the structure of the related object. It has an identifier, a type (that can be integer or binary) and a weight. The schema is defined into Listing 20;

`LinearExpression` defines the structure of the related object. It is composed by a set of terms that are implicitly bounded by an operator. The schema is defined into Listing 22;

`ConstraintOperator` defines the structure of the related object. An operator could be one of the following: `Equal`, `Less Equal` or `Greater Equal`. The schema is defined into Listing 21;

`LAFilteringImpl` defines the structure of a LA filtering implementation. Each `LAFilteringImpl` must specify which `FilteringDevice` adopts and which `Link` will be used. The schema is defined into Listing 24;

`OptimizationProfile` defines the structure of the optimization profile. The schema is defined into Listing 25;

`SolverParameter` defines the structure of the parameters passed as argument to the solver. The schema is defined into Listing 25.

Listing 15: ALFilteringConstraint.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
   xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="LAsFiltering.xsd" />
   <xs:include schemaLocation="FilteringDevice.xsd" />
   <xs:include schemaLocation="LAFilteringImpl.xsd" />
   <xs:include schemaLocation="ConstraintOperator.xsd" />
```

```
<xs:element name="ALFilteringConstraintContainer">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="filteringConstraint"
        type="posecco_filtering:ALFilteringConstraint" minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name="ALFilteringConstraint" abstract="true">
  <xs:attribute name="id" type="xs:string" />
</xs:complexType>

<xs:complexType name="LAFilteringImplConstraint">
  <xs:complexContent>
    <xs:extension base="posecco_filtering:ALFilteringConstraint">
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="LAFilteringImplRequired">
  <xs:complexContent>
    <xs:extension base="posecco_filtering:LAFilteringImplConstraint">
      <xs:sequence>
        <xs:element name="isRequired" type="xs:boolean" />
        <xs:element name="LAFilteringImpl" type="posecco_filtering:LAFilteringImpl"
          minOccurs="1" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="LAFilteringRedundancyConstraint">
  <xs:complexContent>
    <xs:extension base="posecco_filtering:ALFilteringConstraint">
      <xs:sequence>
        <xs:element name="noImpl" type="xs:int" />
        <xs:element name="relatedToLAFiltering" type="posecco_filtering:LAFiltering" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="FilteringDeviceRequired">
  <xs:complexContent>
    <xs:extension base="posecco_filtering:ALFilteringConstraint">
      <xs:sequence>
        <xs:element name="isRequired" type="xs:boolean" />
        <xs:element name="requiresLA" type="posecco_filtering:LAFiltering" />
        <xs:element name="requiresFilteringDevice"
        type="posecco_filtering:FilteringDevice" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="FilteringLinkConstraint">
  <xs:complexContent>
    <xs:extension base="posecco_filtering:ALFilteringConstraint">
      <xs:sequence>
        <xs:element name="value" type="xs:string" />
        <xs:element name="type" type="xs:string" />
        <xs:element name="hasConstraintOperator"
        type="posecco_filtering:ConstraintOperator" />
      </xs:sequence>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>



</xs:schema>
```

Listing 16: LAsFiltering.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
   xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="LAFilteringImpl.xsd" />
   <xs:include schemaLocation="FilteringDevice.xsd" />
   <xs:include schemaLocation="ALFilteringConstraint.xsd" />

   <xs:element name="LAsFiltering">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="la" type="posecco_filtering:LAFiltering"
               minOccurs="0" maxOccurs="unbounded" />
         </xs:sequence>
      </xs:complexType>
   </xs:element>

   <xs:complexType name="LAFiltering" abstract="true">
      <xs:sequence>
         <xs:element name="cardinality" type="xs:int" />
         <xs:element name="source" type="xs:string" />
         <xs:element name="sourceType" type="xs:string" />
         <xs:element name="networkElementSourceId" type="xs:string" />
         <xs:element name="destination" type="xs:string" />
         <xs:element name="destinationType" type="xs:string" />
         <xs:element name="networkElementDestinationId" type="xs:string" />
         <xs:element name="order" type="xs:int" />
         <xs:element name="connectionStateType" type="xs:string" />
         <xs:element name="IPVersion" type="xs:string" /> <!-- ipv4 or ipv6 -->
         <xs:element name="L3ProtocolType" type="xs:string" /> <!-- e.g., icmp -->
         <xs:element name="L4ProtocolType" type="xs:string" />
         <xs:element name="L4SourcePort" type="xs:string" />
         <xs:element name="L4DestinationPort" type="xs:string" />
         <xs:element name="L7ProtocolType" type="xs:string" />
         <xs:element name="L7Method" type="xs:string" />
         <xs:element name="url" type="xs:string" />
         <xs:element name="technology" type="xs:string" default="nd"/>
         <xs:element name="requiredDevice"
            type="posecco_filtering:FilteringDeviceRequired" minOccurs="0"
            maxOccurs="unbounded" />
         <xs:element name="isRelatedTo" type="posecco_filtering:LAFilteringAllow"
            minOccurs="1" maxOccurs="unbounded" />
         <xs:element name="hasLAImplId" type="xs:string" minOccurs="0"
            maxOccurs="unbounded" />
```

```
      <xs:element name="hasPrecedence" type="xs:string"
        minOccurs="0" maxOccurs="unbounded" /> <!-- filtering LA that precedes this -->
      <xs:element name="laRawId" type="xs:string" />
      <xs:element name="done" type="xs:boolean" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>

  <xs:complexType name="LAFilteringAllow">
    <xs:complexContent>
      <xs:extension base="posecco_filtering:LAFiltering">
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="LAFilteringDeny">
    <xs:complexContent>
      <xs:extension base="posecco_filtering:LAFiltering">
        <xs:sequence>
          <xs:element name="numberOfFilteringDevice" type="xs:int" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Listing 17: ILConstraint.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
   xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="Term.xsd" />
   <xs:include schemaLocation="ConstraintOperator.xsd" />
   <xs:include schemaLocation="LinearExpression.xsd" />

   <xs:element name="ILConstraintContainer">
     <xs:complexType>
       <xs:sequence>
         <xs:element name="ILconstraint" type="posecco_filtering:ILConstraint"
           minOccurs="0" maxOccurs="unbounded" />
       </xs:sequence>
     </xs:complexType>
   </xs:element>

   <xs:complexType name="ILConstraint">
     <xs:sequence>
       <xs:element name="containsTerm" type="posecco_filtering:Term" />
       <xs:element name="containsConstraintOperator"
       type="posecco_filtering:ConstraintOperator" />
       <xs:element name="containsLinearExpression"
       type="posecco_filtering:LinearExpression" />
     </xs:sequence>
     <xs:attribute name="id" type="xs:string" />
   </xs:complexType>
```

```
</xs:schema>
```

Listing 18: FilteringDevice.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
   xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
   elementFormDefault="qualified">
   <xs:include schemaLocation="LAsFiltering.xsd" />
   <xs:include schemaLocation="Weight.xsd" />

   <xs:element name="FilteringDeviceContainer">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="filteringDevice" type="posecco_filtering:FilteringDevice"
               minOccurs="0" maxOccurs="unbounded" />
         </xs:sequence>
      </xs:complexType>
   </xs:element>

   <xs:complexType name="FilteringDevice">
      <xs:sequence>
         <xs:element name="weight" type="posecco_filtering:Weight" />
         <xs:element name="implementsLAImpl" type="posecco_filtering:LAFilteringImpl"
            minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" />
   </xs:complexType>

</xs:schema>
```

Listing 19: Variable.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
   xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
   elementFormDefault="qualified">

   <xs:element name="VariablesContainer">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="variable" type="posecco_filtering:Variable"
```

```
            minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:complexType name="Variable">
      <xs:sequence>
        <xs:element name="value" type="xs:double" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" />
    </xs:complexType>

</xs:schema>
```

Listing 20: Term.xsd

```
<!--
    Copyright (c) 2013 Politecnico di Torino.
    All rights reserved. This program and the accompanying materials
    are made available under the terms of the Eclipse Public License v1.0
    which accompanies this distribution, and is available at
    http://www.eclipse.org/legal/epl-v10.html

    Contributors:
        TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
    xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
    elementFormDefault="qualified">
    <xs:include schemaLocation="Weight.xsd" />

    <xs:element name="TermContainer">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="term" type="posecco_filtering:Term"
            minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:complexType name="Term">
      <xs:sequence>
        <xs:element name="type" type="xs:string" />
        <xs:element name="value" type="xs:double" />
        <xs:element name="weight" type="posecco_filtering:Weight" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" />
    </xs:complexType>

</xs:schema>
```

Listing 21: ConstraintOperator.xsd

```
<!--
    Copyright (c) 2013 Politecnico di Torino.
    All rights reserved. This program and the accompanying materials
    are made available under the terms of the Eclipse Public License v1.0
    which accompanies this distribution, and is available at
    http://www.eclipse.org/legal/epl-v10.html

    Contributors:
        TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
  xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
  elementFormDefault="qualified">

  <xs:simpleType name="ConstraintOperator">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Equal" />
      <xs:enumeration value="Lower_Equal" />
      <xs:enumeration value="Greater_Equal" />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

Listing 22: LinearExpression.xsd

```
<!--
  Copyright (c) 2013 Politecnico di Torino.
  All rights reserved. This program and the accompanying materials
  are made available under the terms of the Eclipse Public License v1.0
  which accompanies this distribution, and is available at
  http://www.eclipse.org/legal/epl-v10.html

  Contributors:
      TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
  xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
  elementFormDefault="qualified">
  <xs:include schemaLocation="Term.xsd" />
  <xs:include schemaLocation="Weight.xsd" />


  <xs:element name="LinearExpressionContainer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="linearExpression" type="posecco_filtering:LinearExpression"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="LinearExpression">
    <xs:sequence>
      <xs:element name="containsTerm" type="posecco_filtering:Term"
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="hasWeight" type="posecco_filtering:Weight" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>


</xs:schema>
```

Listing 23: Link.xsd

```
<!--
  Copyright (c) 2013 Politecnico di Torino.
  All rights reserved. This program and the accompanying materials
  are made available under the terms of the Eclipse Public License v1.0
  which accompanies this distribution, and is available at
  http://www.eclipse.org/legal/epl-v10.html
```

```
  Contributors:
      TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
  xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
  elementFormDefault="qualified">
  <xs:include schemaLocation="Weight.xsd" />

  <xs:element name="LinkContainer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="link" type="posecco_filtering:Link"
          minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Link">
    <xs:sequence>
      <xs:element name="weight" type="posecco_filtering:Weight" />
      <xs:element name="hasSourceId" type="xs:string" />
      <xs:element name="hasSourceInterfaceId" type="xs:string" />
      <xs:element name="hasDestinationId" type="xs:string" />
      <xs:element name="hasDestinationInterfaceId" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
  </xs:complexType>

</xs:schema>
```

Listing 24: LAFilteringImpl.xsd

```
<!--
  Copyright (c) 2013 Politecnico di Torino.
  All rights reserved. This program and the accompanying materials
  are made available under the terms of the Eclipse Public License v1.0
  which accompanies this distribution, and is available at
  http://www.eclipse.org/legal/epl-v10.html

  Contributors:
      TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://posecco.eu/sdss/icarea/filtering/model/core"
  xmlns:posecco_filtering="http://posecco.eu/sdss/icarea/filtering/model/core"
  xmlns:posecco_common="http://posecco.eu/sdss/icarea/common/model/core"
  elementFormDefault="qualified">
  <xs:include schemaLocation="FilteringDevice.xsd" />
  <xs:include schemaLocation="Link.xsd" />
  <xs:include schemaLocation="LAsFiltering.xsd" />
  <xs:import schemaLocation="../common/NAT.xsd"
  namespace="http://posecco.eu/sdss/icarea/common/model/core"/>

  <xs:element name="LAFilteringImplContainer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="logicalAssociationFilteringImplementations"
          type="posecco_filtering:LAFilteringImpl" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="LAFilteringImpl">
```

```
    <xs:sequence>
      <xs:element name="adoptsFilteringDevice" type="posecco_filtering:FilteringDevice"
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="adoptsNatDevice" type="posecco_common:NATDevice"
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="adoptsLink" type="posecco_filtering:Link"
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="LAFiltering" type="posecco_filtering:LAFiltering" />
      <xs:element name="pathId" type="xs:string" />
      <xs:element name="motivation" type="xs:string" />
      <xs:element name="performance" type="posecco_filtering:Weight" />
      <xs:element name="risk" type="posecco_filtering:Weight" />
      <xs:element name="economic-cost" type="posecco_filtering:Weight" />
      <xs:element name="deviceRuleOrder" type="posecco_filtering:DeviceRuleOrder"
        minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="LAFilteringImplId" type="xs:string" />
  </xs:complexType>

  <xs:complexType name="DeviceRuleOrder">
    <xs:sequence>
      <xs:element name="priority" type="xs:int" default="0"/> <!-- priority of
      this rule -->
    </xs:sequence>
    <xs:attribute name="DeviceId" type="xs:string" />
  </xs:complexType>

</xs:schema>
```

Listing 25: ToolConfiguration.xsd

```
<!--
   Copyright (c) 2013 Politecnico di Torino.
   All rights reserved. This program and the accompanying materials
   are made available under the terms of the Eclipse Public License v1.0
   which accompanies this distribution, and is available at
   http://www.eclipse.org/legal/epl-v10.html

   Contributors:
       TorSec - PoSecCo Team (http://security.polito.it/posecco/sdss) - initial API and
implementation
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://posecco.eu/sdss/icarea/filtering/coordinator/model/core"
   xmlns:posecco_filtering_coordinator="http://posecco.eu/sdss/icarea/
filtering/coordinator/model/core"
   elementFormDefault="qualified">
  <!-- <xs:include schemaLocation="OptimizationProfile.xsd" /> <xs:include
     schemaLocation="SolverParameter.xsd" /> -->

  <xs:element name="ToolConfiguration">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="predefinedOptimizationProfile"
          type="posecco_filtering_coordinator:OptimizationProfile" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element name="externalRepoInfo"
          type="posecco_filtering_coordinator:ExternalRepositoryInfo" />
        <xs:element name="predefinedExternalSolverInfo"
          type="posecco_filtering_coordinator:SolverParameter" minOccurs="0"
          maxOccurs="unbounded" />
        <xs:element name="internalModelsInfo"
          type="posecco_filtering_coordinator:InternalModelsInfo" />
      </xs:sequence>

      <xs:attribute name="configurationId" type="xs:string" />
```

```
  </xs:complexType>
</xs:element>

<xs:complexType name="OptimizationProfile">
  <xs:sequence>
    <xs:element name="optimizationMainObjective" type="xs:string" />
    <xs:element name="optimizationObjective"
      type="posecco_filtering_coordinator:OptimizationObjective"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="profileId" type="xs:string" />
  <xs:attribute name="type" type="xs:string" /> <!-- filtering or chprot -->
</xs:complexType>

<xs:complexType name="OptimizationObjective">
  <xs:sequence>
    <xs:element name="subtargetFunctionName" type="xs:string" />
    <xs:element name="subtargetFunctionWeight" type="xs:double" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" />
</xs:complexType>

<xs:complexType name="SolverParameter">
  <xs:sequence>
    <xs:element name="solverName" type="xs:string" />
    <xs:element name="solverPath" type="xs:string" />
    <xs:element name="timemout" type="xs:string" />
    <xs:element name="solverParams" type="xs:string"
      minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" />
</xs:complexType>

<xs:complexType name="ExternalRepositoryInfo">
  <xs:sequence>
    <xs:element name="url" type="xs:string" />
    <xs:element name="protocol" type="xs:string" />
    <xs:element name="port" type="xs:string" />
    <xs:element name="username" type="xs:string" />
    <xs:element name="password" type="xs:string" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="InternalModelsInfo">
  <xs:sequence>
    <xs:element name="WeightsOracleFile" type="xs:string" />
    <xs:element name="OptimizationProfileFile" type="xs:string" />
    <xs:element name="LogicalAssociationsRawContainerFile"
      type="xs:string" />
    <xs:element name="LogicalAssociationsInputContainerFile"
      type="xs:string" />
    <xs:element name="LogicalAssociationsRawDependencyContainerFile"
      type="xs:string" />
    <xs:element name="LAsFilteringFile" type="xs:string" />
    <xs:element name="LandscapeDescriptionFile" type="xs:string" />
    <xs:element name="FilteringDeviceContainerFile" type="xs:string" />
    <xs:element name="LAFilteringImplContainerFile" type="xs:string" />
    <xs:element name="FilteringOptModelFile" type="xs:string" />
    <xs:element name="FilteringConfigurationsFile" type="xs:string" />
    <xs:element name="ConfigurationOntologyInputFile" type="xs:string" />
    <xs:element name="ConfigurationOntologyOutputFile" type="xs:string" />
    <xs:element name="OntologyInputFile" type="xs:string" />
    <xs:element name="OntologyInputLASplittedFile" type="xs:string" />
    <xs:element name="OntologyOutputFile" type="xs:string" />
  </xs:sequence>
</xs:complexType>
```

```
</xs:schema>
```

5 Extending optimization models

Modify the Data Protection optimization model

Expert users might be interested in adding new features to the optimization model. This section gives a brief procedure to achieve this goal. As defined in the previous section, the optimization model is composed by two parts (abstract and implementation). Since the implementation part contains only low level information (e.g., linear expressions, variables, etc.), the user is primarily interested on changes related to the abstract part. However in this guide we discuss how to provide changes on both parts.

The changes on abstract part require the following steps:

1. **modify XML schema(s)** changes on optimization model requires to edit the related XML schemas, to modify/add object attributes or to add new objects.
2. **generate or regenerate new Java classes** once the changes are implemented on the XML schemas, it is possibile regenerate or generate related Java classes. When a new schema is added (e.g., to create a new object), edit the `generate_classes.sh` adding the name of new schema (e.g., `MyLink.xsd`). Then executing the script to generate Java classes.
3. **modify the `AbstractOptBuilder` module** this fills the abstract part of optimization model following a set of operations on related objects. Therefore adding or modifying attributes or objects requires changes in the operations logic;
4. **modify the `OptModelBuilder` module** this fills the implementation part of the optimization model following a set of Drools rules on related objects. Therefore adding or modifying attributes or objects requires changes in the rules file (e.g., `ChProtOptBuilder.drl`);
5. **modify the `SolMapping` module** (optional) this starts from optimal solution to identify which channel protection devices implement the set of logical associations. Similarly to previous module, this adopts a set of Drools rules to perform its task. Thus, when new mappings among objects are required the user must also modify the set of rules (e.g., `solMapping.drl`).

Design a new Data Protection optimization model

The design and integration of a new optimization model is a long and quite complex task. This section tries to describe the guidelines to achieve this objective. First of all we suggest to analyze the available optimization models to understand the philosophy and the structure. Second, it is a good idea starting from an available model to create a new one trying to reusing, when it is possible, most of the elements.

Anyhow, the guidelines to design and integrate a new optimization model are:

1. design of the new model schema: we suggest to create a core XML schema and a set of other schemas to design related objects and attributes;
2. generate Java classes associated to the model: we suggest to adopt `xjc` tool, included into JAXB, to generate classes starting from the XML schema;
3. create a new module to populate the abstract part of the model: starting from `AbstractOptBuilder` we suggest to add an equivalent class to populate the model;
4. create a new module to populate the implementation part of the model: startin from `OptModelBuilder` we suggest to design the new set of Drools rule to fill the implementation part;
5. create a new module to solve the model: starting from `OptModelSolver` we suggest to design an equivalent class to transform objective function, constraints and variables into specific solver format, using the available API;

6. create a new module to map the optimal solution in the model: starting from `SolMapping` we suggest to design the set of Drools rule to fill abstract object of the model;

7. modify the optimization module to support the new model: starting from `Optimization` module we suggest to modify the source code to support the new model. For example, we need to identify the new model using a string and then calling the related classes to build and solve the model.

Modify the Filtering optimization model

Expert users might be interested in adding new features to the optimization model. This section gives a brief procedure to achieve this goal. As defined in the previous section, the optimization model is composed by two parts (abstract and implementation). Since the implementation part contains only low level information (e.g., linear expressions, variables, etc.), the user is primarily interested on changes related to the abstract part. However in this guide we discuss how to provide changes on both parts.

The changes on abstract part require the following steps:

1. **modify XML schema(s)** changes on optimization model requires to edit the related XML schemas, to modify/add object attributes or to add new objects.

2. **generate or regenerate new Java classes** once the changes are implemented on the XML schemas, it is possibile regenerate or generate related Java classes. When a new schema is added (e.g., to create a new object), edit the `generate_classes.sh` adding the name of new schema (e.g., `MyLink.xsd`). Then executing the script to generate Java classes.

3. **modify the `AbstractOptBuilder` module** this fills the abstract part of optimization model following a set of operations on related objects. Therefore adding or modifying attributes or objects requires changes in the operations logic;

4. **modify the `OptModelBuilder` module** this fills the implementation part of the optimization model following a set of Drools rules on related objects. Therefore adding or modifying attributes or objects requires changes in the rules file (e.g., `Filtering_max_performance.drl`);

5. **modify the `SolMapping` module** (optional) this starts from optimal solution to identify which filtering devices implement the set of logical associations. Similarly to previous module, this adopts a set of Drools rules to perform its task. Thus, when new mappings among objects are required (e.g., change the order of the LAFiltering on a particular `FilteringDevice`), the user must also modify the set of rules (e.g., `solMapping.drl`).

Design a new Filtering optimization model

The design and integration of a new optimization model is a long and quite complex task. This section tries to describe the guidelines to achieve this objective. First of all we suggest to analyze the available optimization model to understand the philosophy and the structure. Second, often it is a good idea starting from available model to create a new one trying to reusing, when it is possible, most of the elements.

Anyhow, the guidelines to design and integrate a new optimization model are:

1. design of the new model schema: we suggest to create a core XML schema and a set of other schemas to design related objects and attributes;

2. generate Java classes associated to the model: we suggest to adopt `xjc` tool, included into JAXB, to generate classes starting from the XML schema;

3. create a new module to populate the abstract part of the model: starting from `AbstractOptBuilder` we suggest to add an equivalent class to populate the model;

4. create a new module to populate the implementation part of the model: startin from `OptModelBuilder` we suggest to design the new set of Drools rule to fill the implementation part;
5. create a new module to solve the model: starting from `OptModelSolver` we suggest to design an equivalent class to transform objective function, constraints and variables into specific solver format, using the available API;
6. create a new module to map optimal solution into model: starting from `SolMapping` we suggest to design the set of Drools rule to fill abstract object of the model;
7. modify the optimization module to support the new model: starting from `Optimization` module we suggest to modify the source code to support the new model. For example, we need to identify the new model using a string and then calling the related classes to build and solve the model.

References

[1] Cataldo Basile, Marco Vallini, Rachid Ouchary, Matteo Casalino, and Theodoor Scholte. D3.6 "models for generating the desired configurations". http://www.posecco.eu/fileadmin/POSECCO/user_upload/deliverables/D3.6_Models_for_generating_the_desired_configurations.pdf.

[2] Rachid Ouchary Cataldo Basile, Marco Vallini. "infrastructure configuration service - user manual". TODO.