Policy Modelling

Gregorio Martínez
University of Murcia (Spain)

DESEREC

Dependability and Security by Enhanced Reconfigurability

What is a Policy??

- Set of rules that determine the behaviour of the network, services and applications

> **IF**
> certain ***condition(s)*** are present
> **THEN**
> specific ***action(s)*** are taken

- Example:

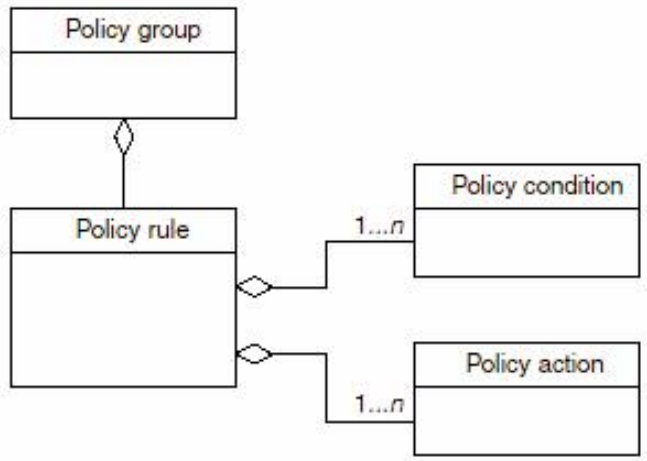
  If *((trafficToOrFrom NetworkA)* and *(dayOfMonth is last10Days))*
  then *securityLevel = high*

Policy Rules

Basic building block of a policy (according to IETF and DMTF)

Composed by:

- One of more conditions (when the policy rule is applicable)
- One of more actions (what the network entity, service or application should do)



Representing Policies

- Ideally based on standard information models and schemas for interoperability issues
- Although most current existing products (CISCO Policy Manager, HP OpenView, …) use proprietary schemas/languages

- Current possibilities (in chronological order):
  - Academic approaches: Ponder, KeyNote, etc.
  - Main standards (from the IETF and DMTF)
    - CIM (Common Information Model)
    - PCIM (Policy Core Information Model) and PCIMe (PCIM extensions)

SPL

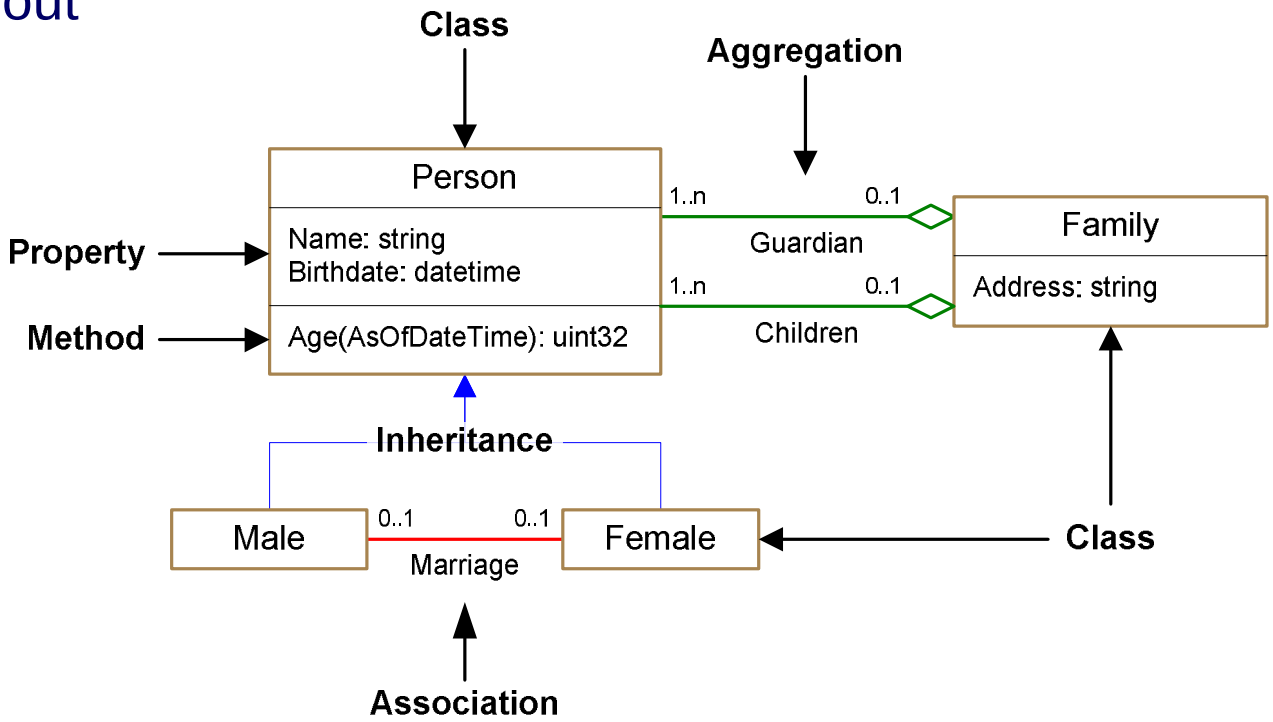
- Language for policy specification
- Defines the desired behaviour of the networked systems and applications
- SPL is based on the standard CIM of DMTF and XML technologies
  - http://www.dmtf.org/standards/cim
- A set of grouping classes has been extracted from the model in order to represent these types of policies
  - It allows grouping, priority and classification of the policies
- Allows representing several types of policies
  - The syntax of each one is defined in his own SPL schema
  - SPL schemas are derived from xCIM Schema
- Uses references to SDL system model
  - The description of the system components should have been created previously by the system administrator using SDL

- Main features:
  - **Based on the CIM Policy Model**
    - This model provides policy-based management by enabling an administrator to represent policies in a vendor-independent and device-independent way
  - **Supports filtering, authentication, authorization, channel protection and operational policies**
    - These are the currently defined types, but SPL can be extended to represent additional types
  - **Provides an XML schema for each type of policy**
    - SPL is composed of five independent schemas, one for each type of policy which is currently defined

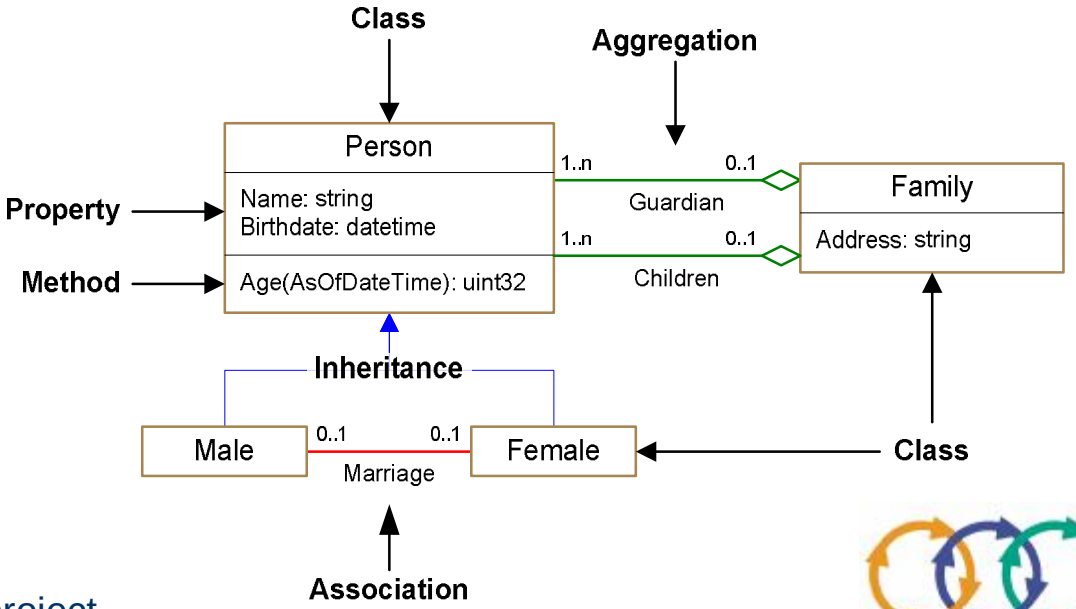
- The object-oriented modelling used by **CIM is based on UML**
- Therefore, it is independent of any:
  - Hardware architecture
  - Operating system
  - Programming language
  - …
- The object-oriented modelling uses meta-schema to describe the model (to represent something in real world)
- Schema: Group of classes with single owner
  - It is used for administration and class naming

UML (brief description)

- Class: Collection or set of objects that have similar properties and fulfil similar purposes
  - A class can contain properties and methods
- Property: Describes the data of the class (also known as attribute)
- Method: Describes the behaviour of the class and process that class carried out



- Inheritance: Describes the relationship of class that derived from parent class, or *superclass*
  - In CIM convention, inheritance uses blue lines
- Aggregation: Defines the relationship of an entity that is made up of some other entities, or a part-of relationship
  - In CIM convention, aggregation uses green lines
- Association: Describes the relationship between two classes or two instances
  - In CIM convention, aggregation uses red lines



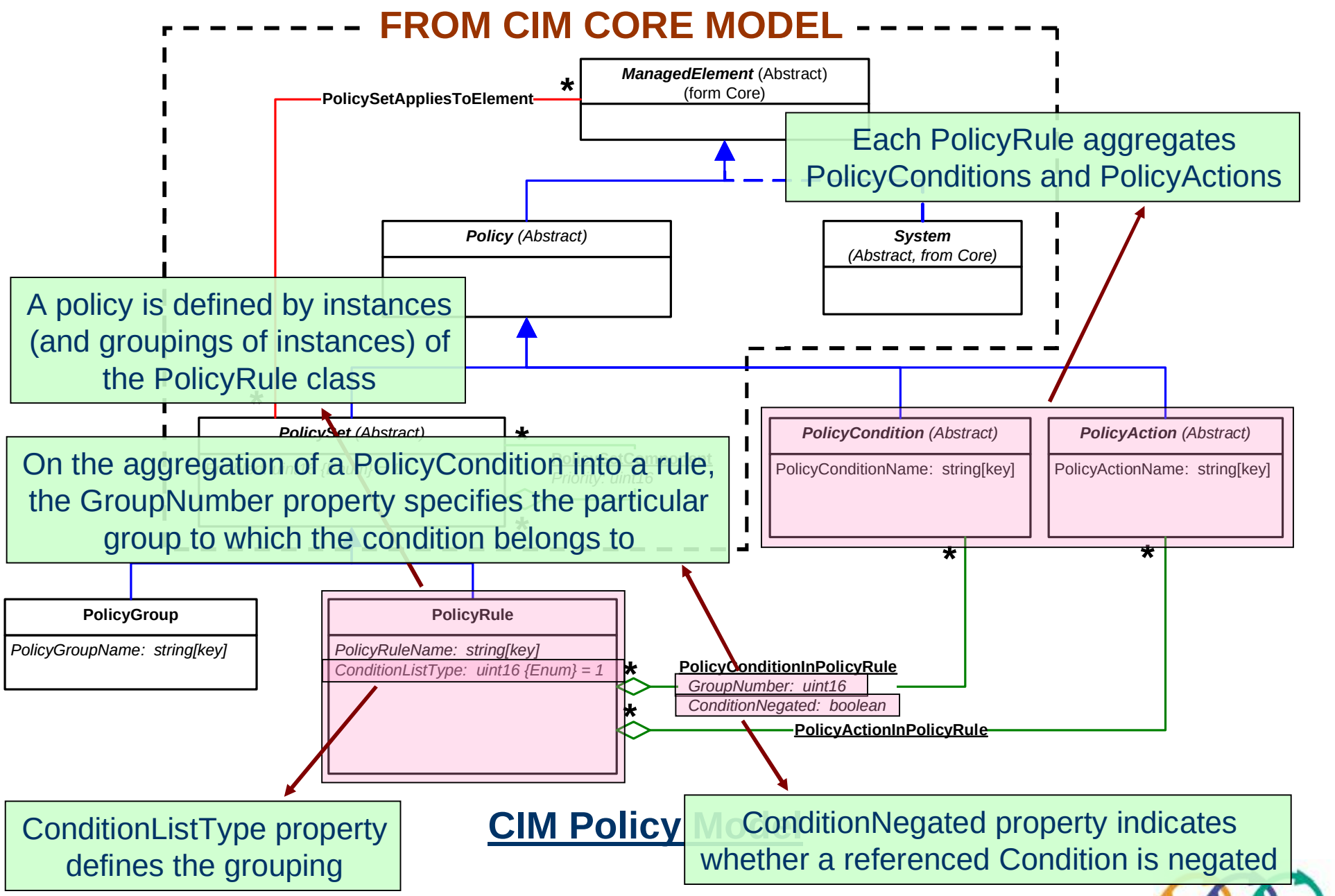
- Policies have the standard “rule” form:

> ***if*** *condition(s)* ***then*** *action(s)*

- For instance: How to model a simple filter rule?

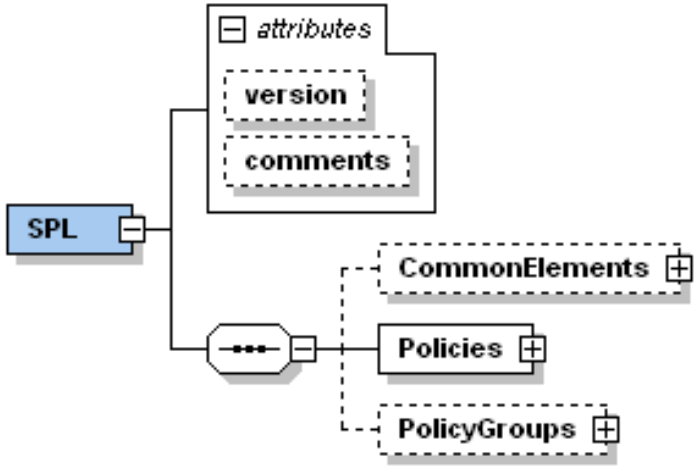
> ***if*** *((TrafficTo equal HostA)* ***and*** *(TrafficPort equal 80))*
> ***then*** *incomingTraffic = permit*

- SPL follows the object-oriented data model of CIM
  - Classes: policy rules, conditions, actions…
  - Associations: a policy rule has one or more conditions, and one or more actions
- SPL is defined on CIM schema v2.11



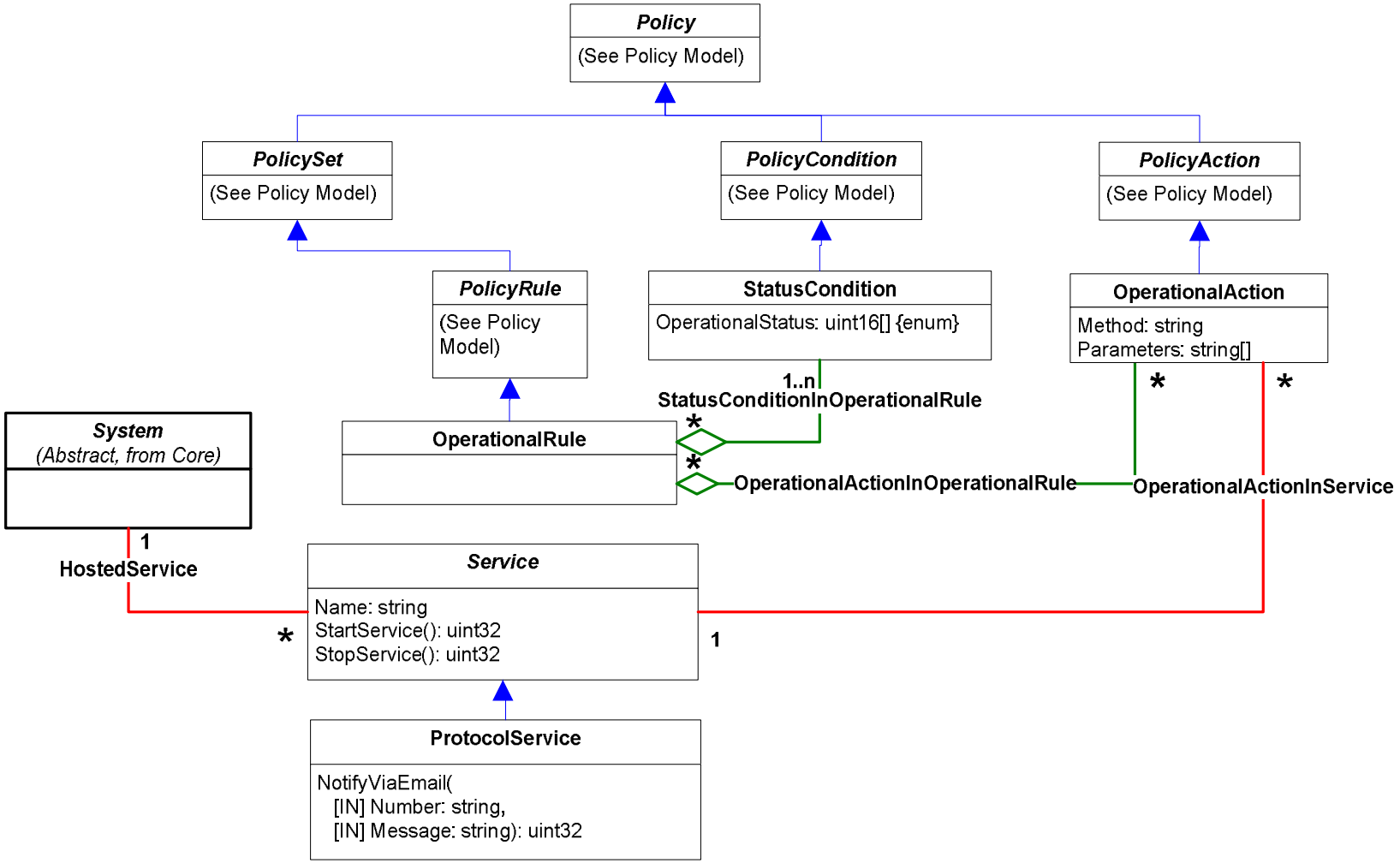
SPL Language Architecture

- Three main groups:



- **Common elements**
  - Concepts that are globally defined and common to some types of Policies and Policy Groups
- **Policies**
  - The security policies describing the dynamic behaviour of the domain being managed
- **Policy Groups**
  - The group of policies and the relation between them

SPL definition for operational policies (reuses some CIM classes):



MOF (Managed Object Format)

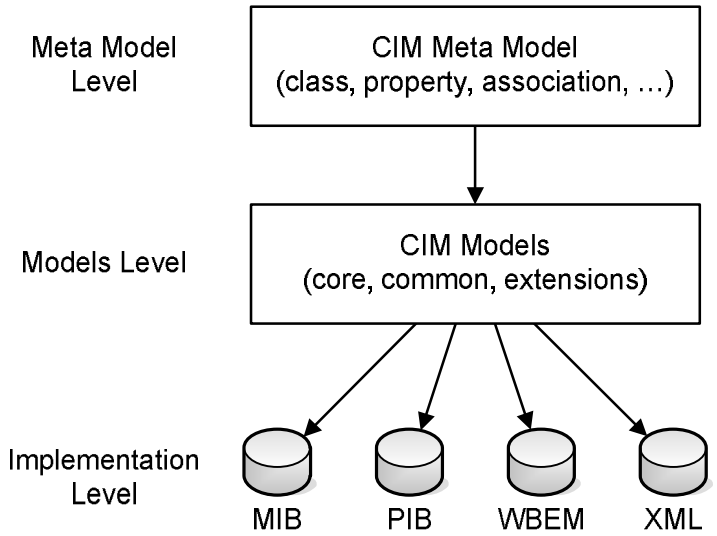
- Language used by DMTF to describe CIM
- Derived from Interface Definition Language (IDL) to describe the management information

- MOF syntax is a way to describe object definitions in text form:
  - Classes
  - Associations
  - Properties
  - Methods, etc.
- This language can be recognized by a compiler
  - A MOF file can be encoded in either Unicode or UTF-8 format

```
// ==========================================
// ManagedElement (in MOF)
// ==========================================
[Abstract, Version ("2.7.0"), Description (
    "ManagedElement is an abstract class that provides a common"
    "superclass (or top of the inheritance tree) for the"
    "non-association classes in the CIM Schema.") ]

class CIM_ManagedElement {
    [MaxLen (64), Description (
        "The Caption property is a short textual description (one-"
        "line string) of the object.") ]
    string Caption;

    [Description (
        "The Description property provides a textual description of "
        "the object.") ]
    string Description;

    [Description (
        "A user-friendly name for the object…") ]
    string ElementName;
};
```

Mapping: CIM to XML Schema (xCIM)

- The CIM Schema is independent of any implementation
- CIM can be represented as several structured specifications



- The XML Schema is used to describe the CIM classes
- CIM classes and instances are valid XML documents for that schema
  - Each CIM class generates its own XSD fragment
- CIM element names are mapped to XML attribute or element values

Mapping: CIM to XML Schema (xCIM)

- Intuitive mapping between the CIM model and the XML schema
- For instance:
  - A CIM class and its properties can be mapped directly a complexType element

```
class CIM_A {
    string A1;
    string A2;
    string A3;
};
```

```
<xs:element name="xCIM_A" type="CIM_A"/>
<xs:complexType name="CIM_A">
    <xs:sequence>
        <xs:element name="A1" type="string"/>
        <xs:element name="A2" type="string"/>
        <xs:element name="A3" type="string"/>
    </xs:sequence>
</xs:complexType>
```

Mapping: CIM to XML Schema (xCIM)

- A CIM instance can be expressed into XML using the XSD generated from CIM class.

```
instance of CIM_A
{
    A1="value_A1";
    A2="value_A2";
    A3="value_A3";
};
```

→

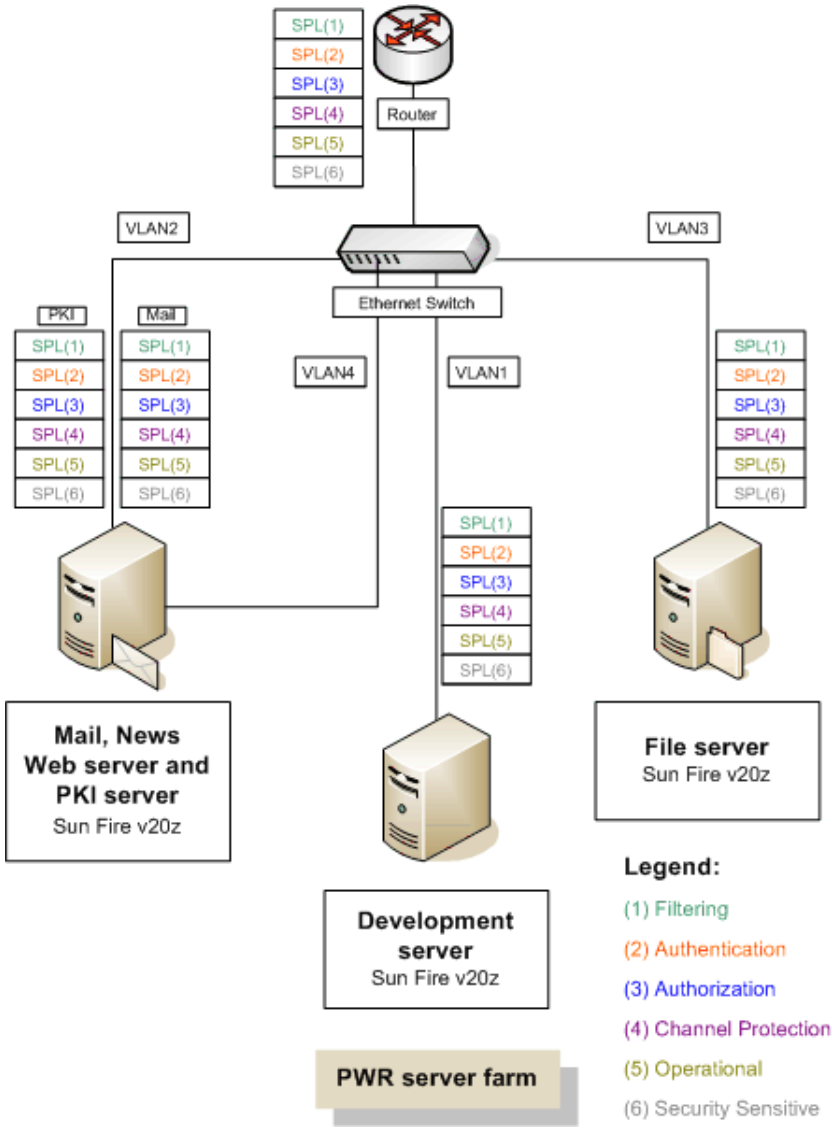
```
<xCIM_A>
    <A1>value_A1</A1>
    <A2>value_A2</A2>
    <A3>value_A3</A3>
</xCIM_A>
```

- We can perform a similar translation to the rest of CIM elements:
  - Inheritance, object identification, associations, and so on

SPL example of applicability

- Typical network scenario:
  - Routers
  - Switches
  - Firewalls
  - Servers
- We can define policies to:
  - Filtering
  - Authentication
  - Authorization
  - Channel protection
  - …

A DESEREC's goal is to provide new types of policies to manage **dependability** issues



Types of policies in DESEREC

In DESEREC two main types of policies are needed:

- **Configuration policies**
  - They specify an operational planning, defining how the system should work
  - They are translated into a full configuration which is applied by WP3
- **Reaction policies**
  - They specify how to monitor the system for incidents, and what to do if they happen
  - They are related to both WP4 and WP3

Configuration policies	Reaction policies
• Routing and filtering policies • Authentication policies • Authorization policies • Channel protection policies	• Monitoring policies • Reconfiguration policies

Types of policies in DESEREC

Configuration policies

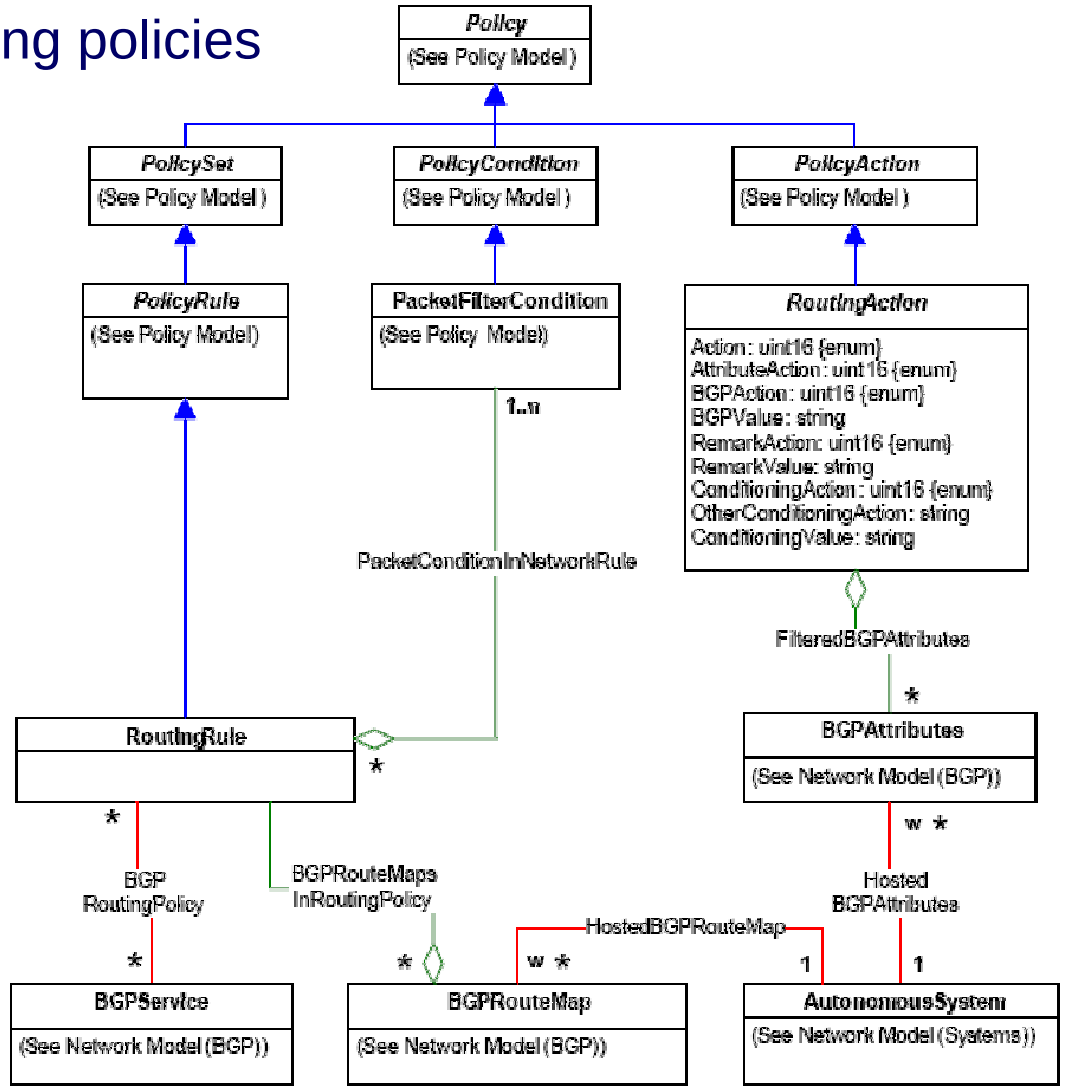
- **Routing and filtering policies**
  - Rules that define the routing and filtering criteria used in a network element (i.e. policy target)
  - The routing policies are bound to control traffic flow
    - Change metrics and path attributes, deny or prefer certain routes, etc.
  - The filtering policies define the filtering requirements used in a network element
    - Source/destination address
    - Source/destination port
    - Protocol type: TCP, IP, ICMP, etc.
    - Others: it can be extended to other rules types

They can be used to specify a high-level operational plan for routers and firewalls, which can be translated into their appropriate configurations

Types of policies in DESEREC

Configuration policies

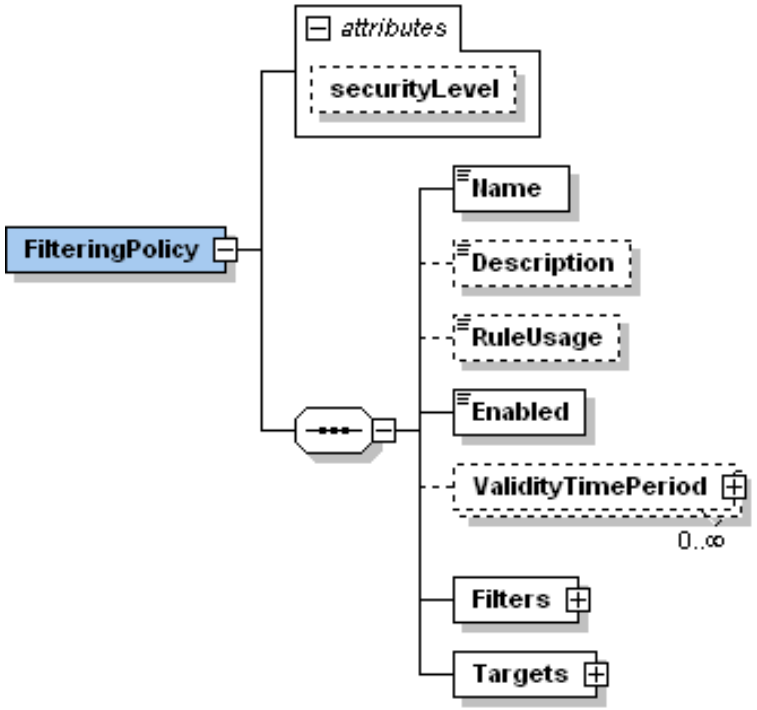
- **Routing** and filtering policies



Types of policies in DESEREC

Configuration policies

- Routing and **filtering policies**

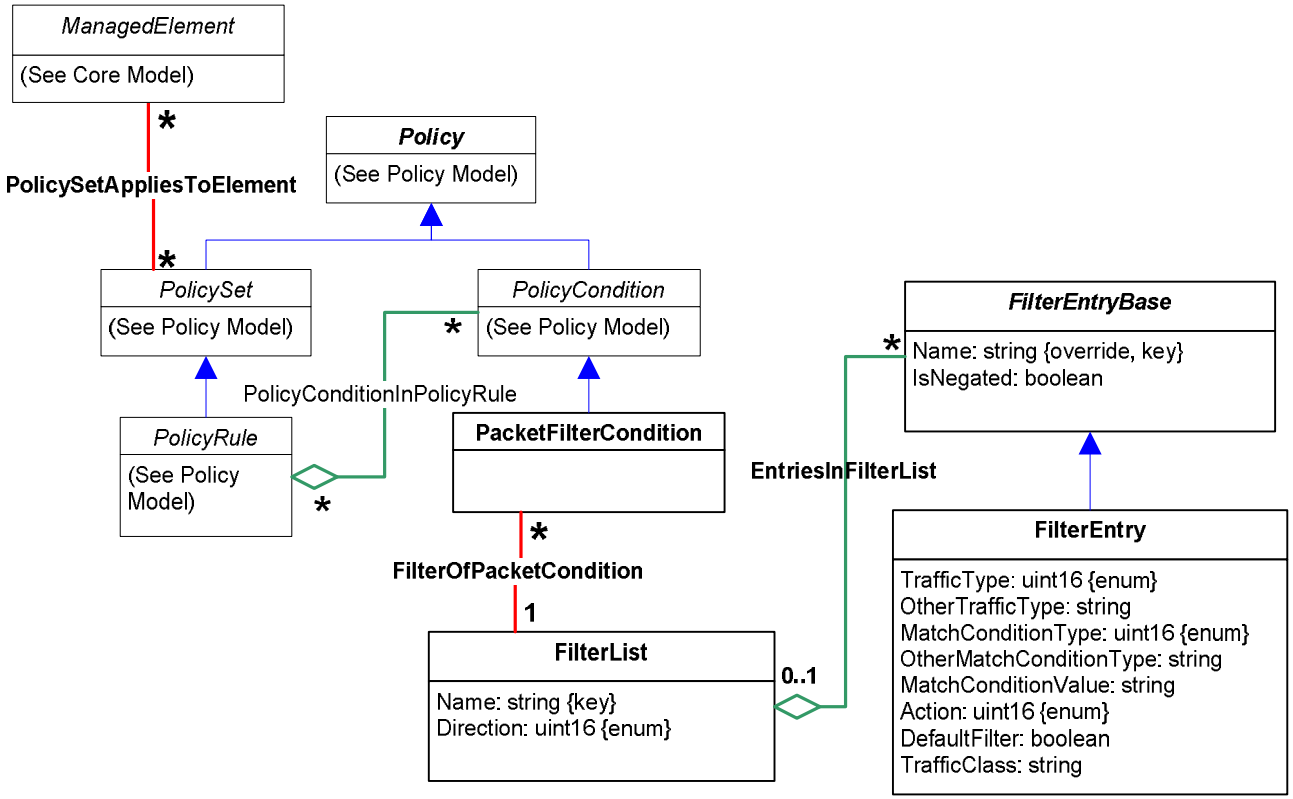


- The **Filters** tag describes the various filters applied to the system
- A **Filter** is used by network devices to identify routes by aggregating a set of entries into a unit
- There are no actions associated with this policy.
  - The actions are implicitly defined for each Filter: Deny / Permit

Types of policies in DESEREC

Configuration policies

- Routing and **filtering policies**



Types of policies in DESEREC

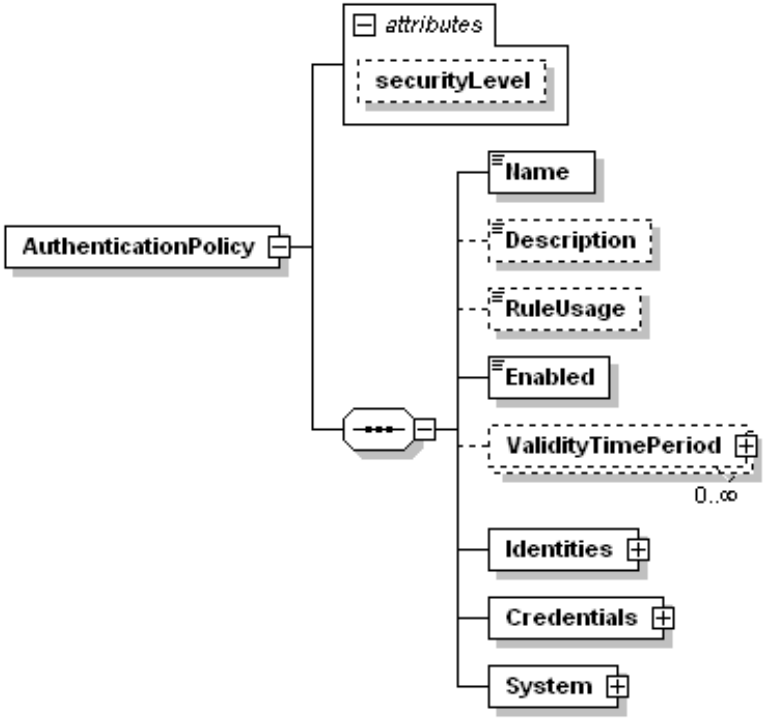
Configuration policies

- **Authentication policies**
  - Rules that define the authentication criteria used for a identity (i.e. policy subject) in a network element (i.e. policy target)
    - The subject of this identity may be a person, a process or a network element
  - Types of authentication that can be supported:
    - Shared secret
    - Account authentication
    - Biometry
    - Identity certificates
    - Kerberos
    - …
  - The *PolicyConditions* in an instance of *AuthenticationRule* describe the various requirements under which the subject is considered as being “authenticated”

Types of policies in DESEREC

Configuration policies

- **Authentication policies**

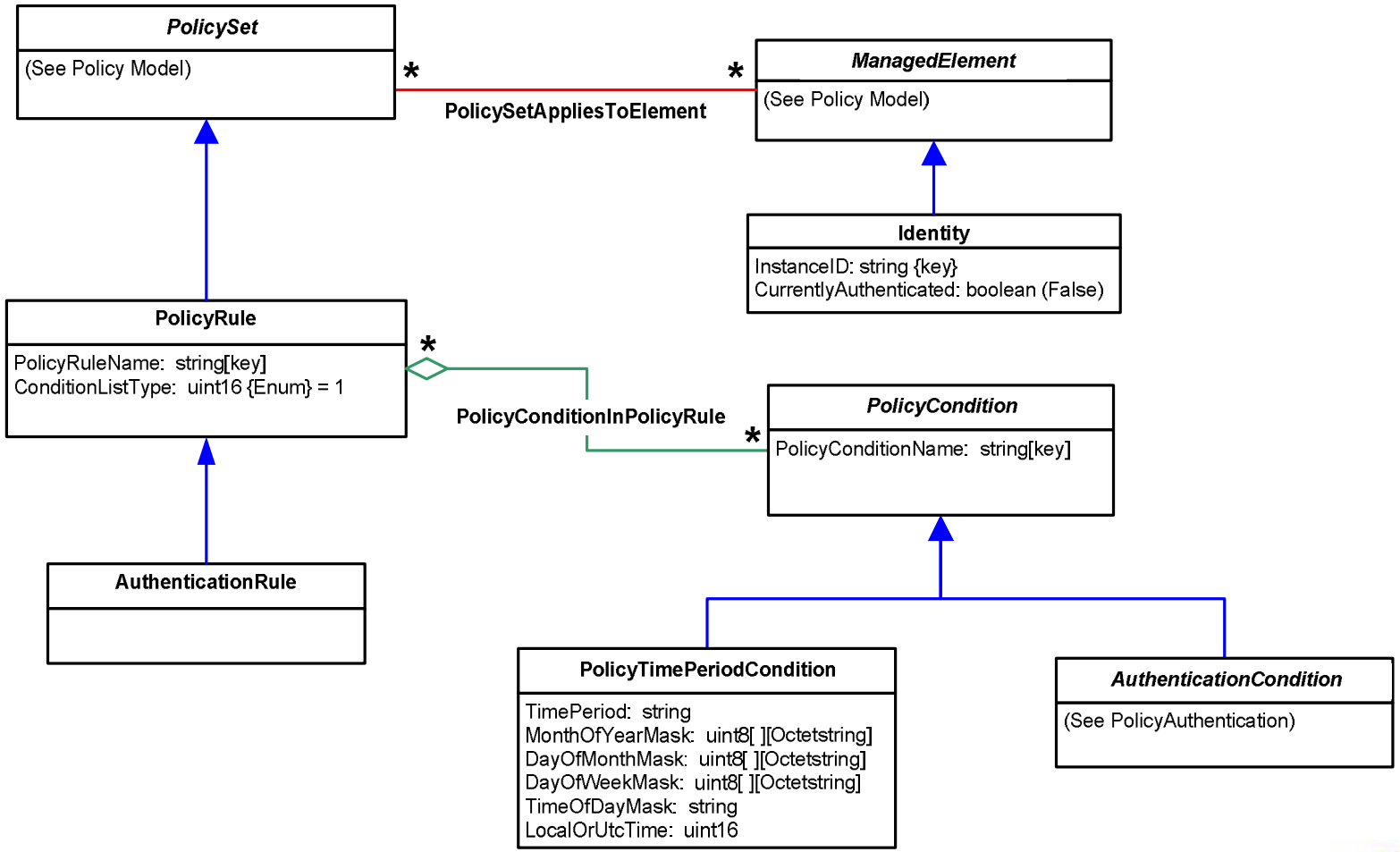


- **Credentials** describe the various requirements under which a Identity is authenticated by the system
- There are no actions associated with this policy
  - **Actions are implicit** γ When the conditions of the rule are met, then the Identity has been authenticated
- The **System** tag represents the system where the policy will be applied

Types of policies in DESEREC

Configuration policies

- **Authentication policies**



Types of policies in DESEREC

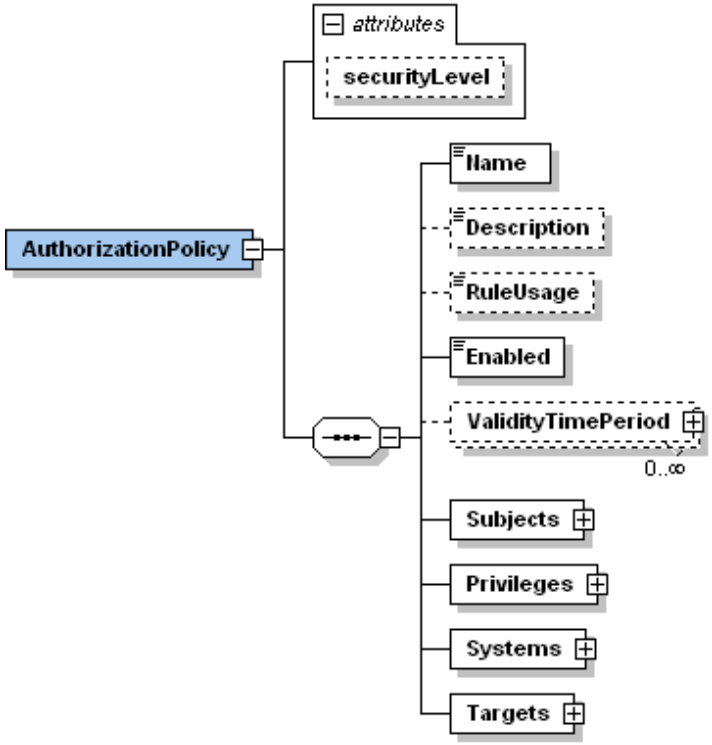
Configuration policies

- **Authorization policies**
  - Rules that define the authorization criteria used in a network element (i.e. policy target) for a identity or role (i.e. policy subject) based on privileges or credentials
  - These policies comprise:
    - Target policy: the network elements (that is, SDL components to which this policy will be applied)
    - Subject policy: the subject identity and/or the roles associated with him.
    - Privilege: authorization granted or denied
  - CIM defines the classes to represent the management concepts that are related to an authorization rule
    - *Privilege* is the base class for all types of activities, which are granted or denied to a subject by a target
    - *AuthorizationRule* is the specific class for the authorization policies

Types of policies in DESEREC

Configuration policies

- **Authorization policies**

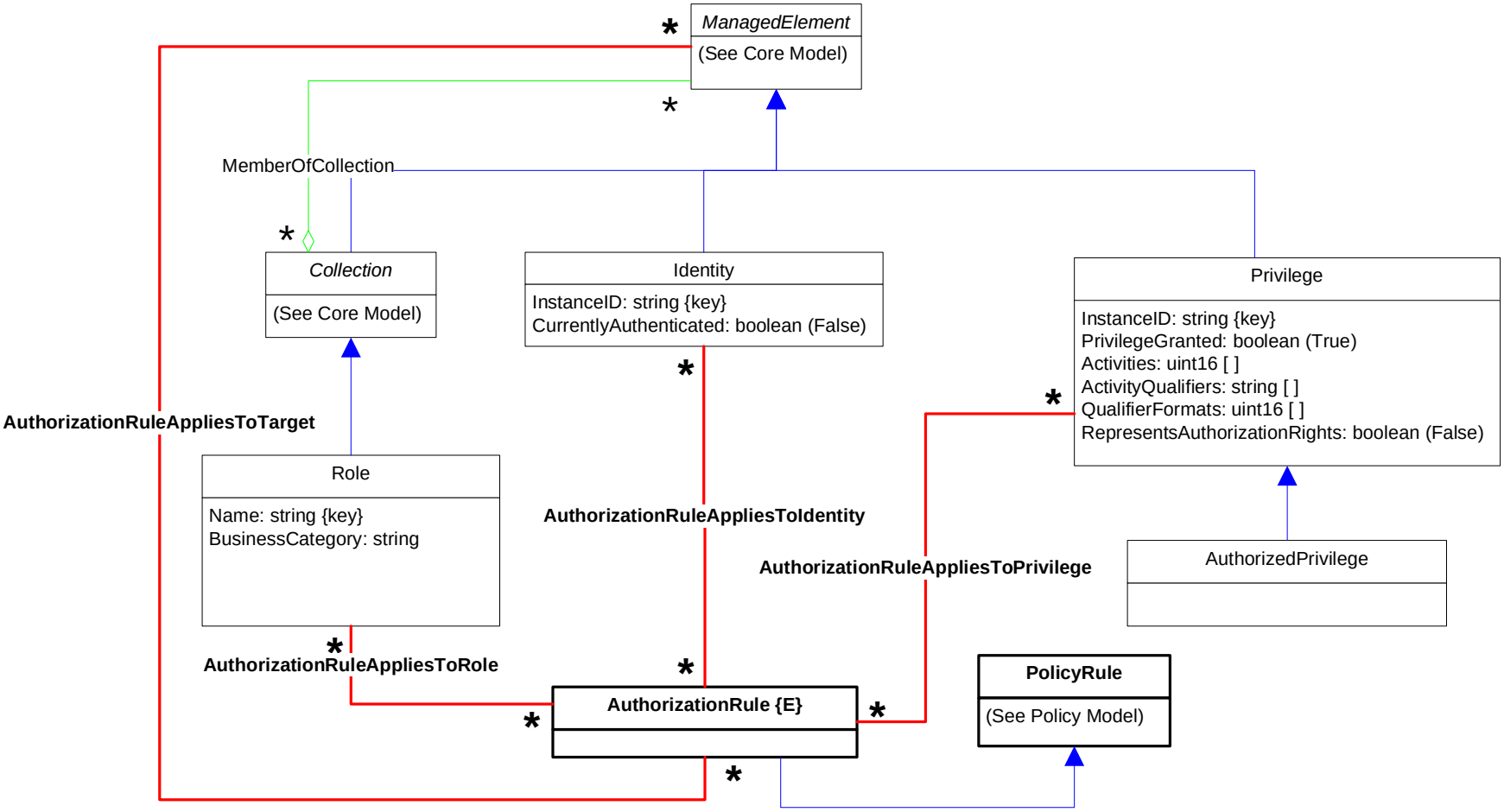


- **Subjects**: Reference to identites or roles

Types of policies in DESEREC

Configuration policies

- **Authorization policies**



Types of policies in DESEREC

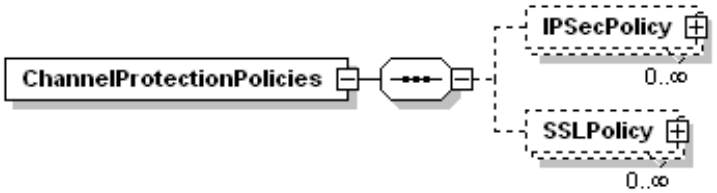
Configuration policies

- **Channel protection policies**
  - Rules that define the encryption criteria used in a network element (i.e. policy target) based on security associations
  - These policies can be:
    - IPsec policies: the security associations can be established statically or using the IKE protocol
    - SSL/TLS policies
  - Instances of *PacketFilterCondition* are used together with *SARule* to define which ciphering configuration should be applied to a particular traffic flow (IKE, IPSec, SSL, TLS)

Types of policies in DESEREC

Configuration policies

- **Channel protection policies**

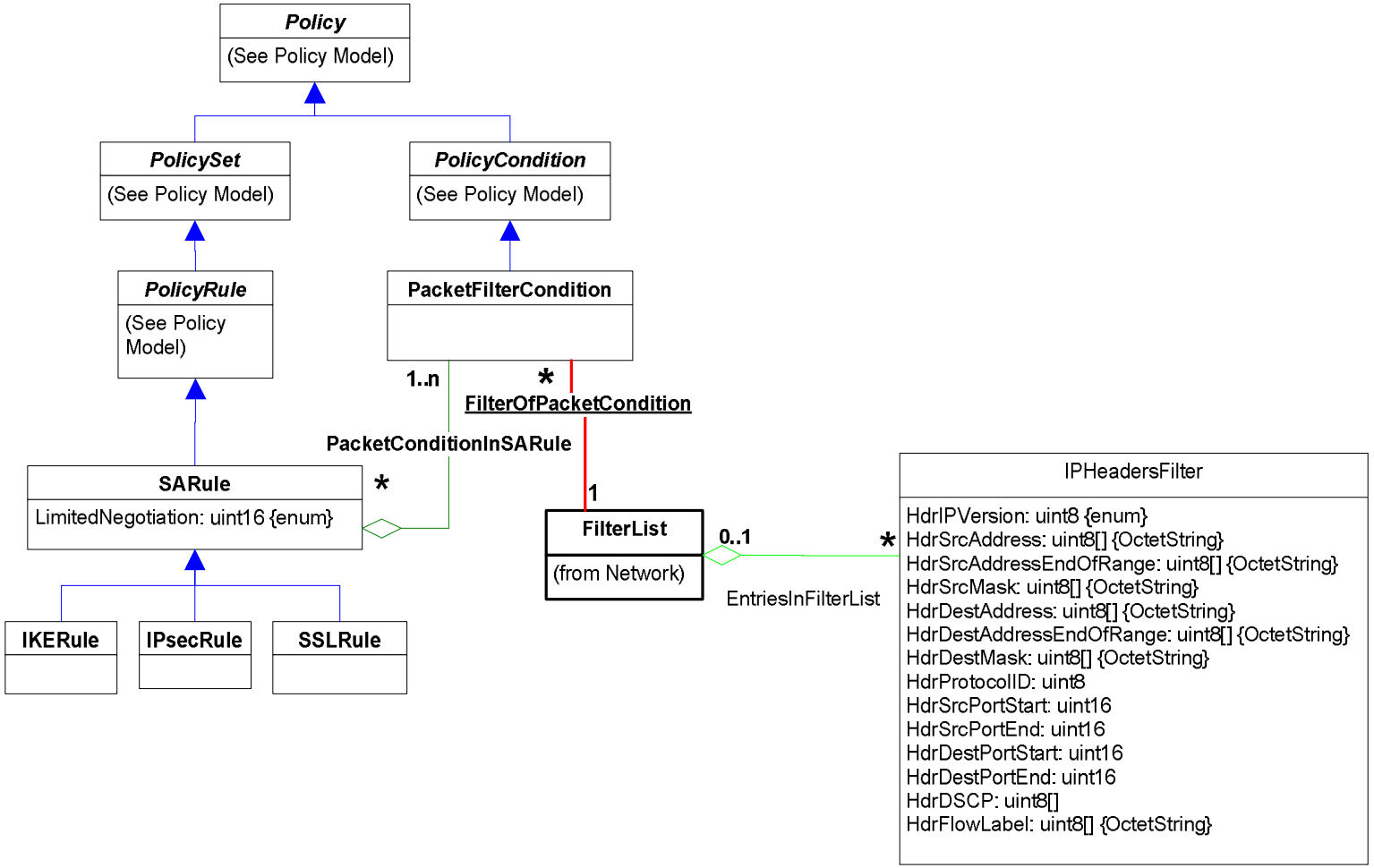


- Two types of channel protection policies has been created:
  - **IPSecPolicy**: It represents the establishment of SAs between endpoints using the IKE protocol (or defining SAs statically)
  - **SSLPolicy**: Represents an SSL communication between different endpoints of the network

Types of policies in DESEREC

Configuration policies

- **Channel protection policies**



Types of policies in DESEREC

Reaction policies

- **Monitoring policies**
  - Rules that define a configuration needed prior the desired monitoring events can be received, such as setting up an SNMP agent to send traps to a specific target upon specific conditions being met
- **Reconfiguration policies**
  - Rules that specify what reconfiguration actions must be taken when the events defined by a monitoring policy are detected

Monitoring and reconfiguration rules work closely to implement DESEREC's reactions for dependability

Types of policies in DESEREC

Monitoring policies

- They can be divided in two great groups:
  - **System monitoring**
    - Supervise critical parameters of a target system
    - These parameters can be detected using, i.e., SNMP software
    - Correct hardware operation, a component goes down, CPU workload, etc.
  - **IDS monitoring**
    - Monitor a critical system in front of internal/external attacks
    - This monitoring can be performed using an IDS software; i.e., SNORT software
    - Ports scan, buffer overflow, bad traffic, etc.

SPL extension for dependability

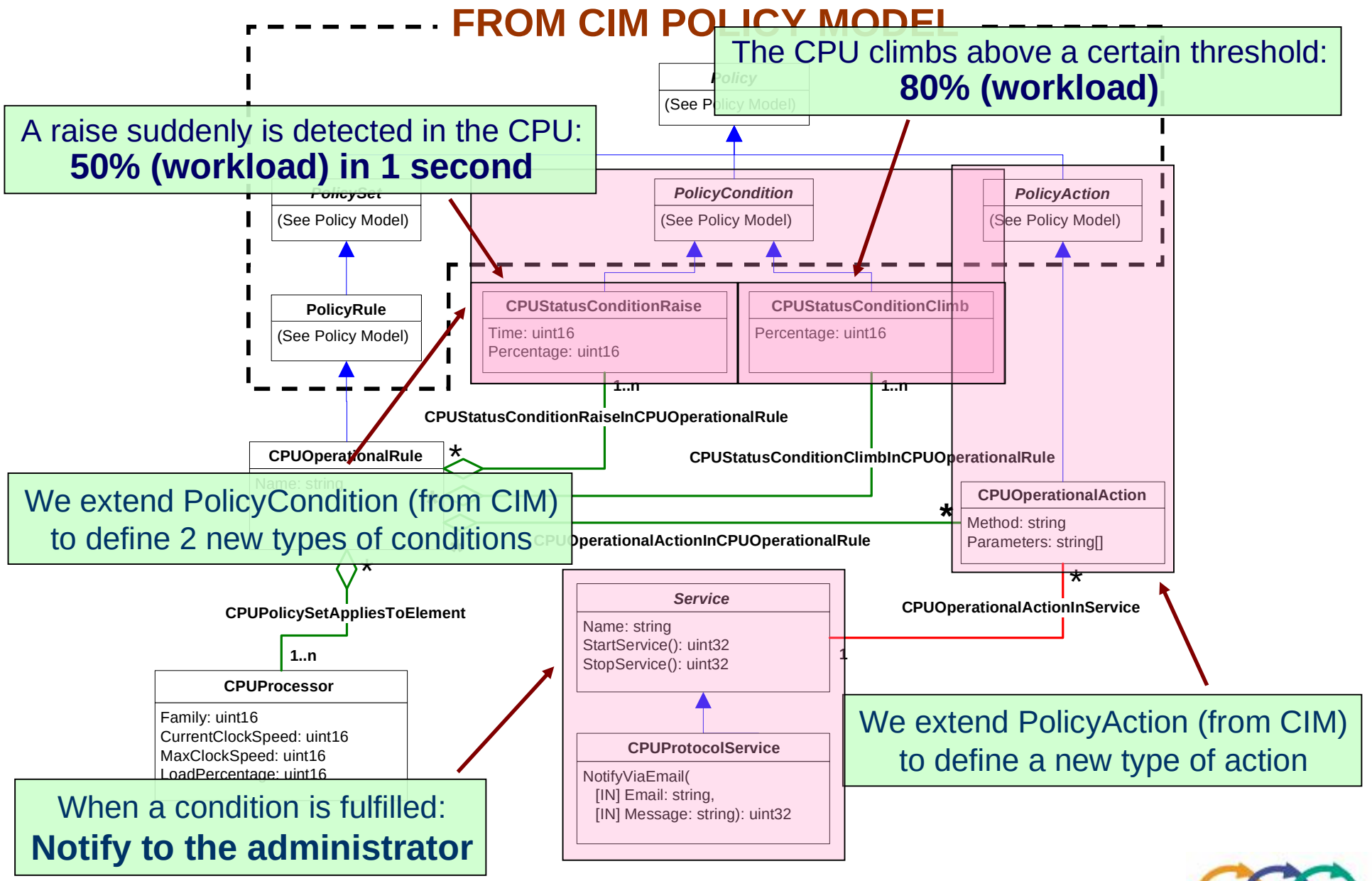
- System dependability in DESEREC is implemented as a policy based, continuous “monitor and reconfigure” loop
  - Dependability relies on the enforcement of reaction policies
- Configuration policies for enforcing the operational plan are supported in the current SPL specification, but some extension is needed for the reaction policies
- Current operational policies only allow defining status-based rules
  - *OperationalRule* class aggregates *StatusCondition*
- In DESEREC, we need additional conditions for reaction policies:
  - Extend *StatusCondition*
  - Extend *PolicyCondition*
  - …
- The same happens with policy actions:
  - Extend *OperationalAction*
  - Extend *PolicyAction*
  - …

SPL extension example

High CPU utilization

- Based on OTE scenario for the Video on Demand service
- High CPU or port utilization may indicate an ongoing attack or problem in the network
- The system administrator must be notified when:
  - A raise suddenly is detected in the CPU, or
  - The CPU climbs above a certain threshold
- The system administrator should be able to establish these values as he considers appropriate, for instance:
  - Raise suddenly = 50% (workload) in 1 second
  - Threshold = 80%

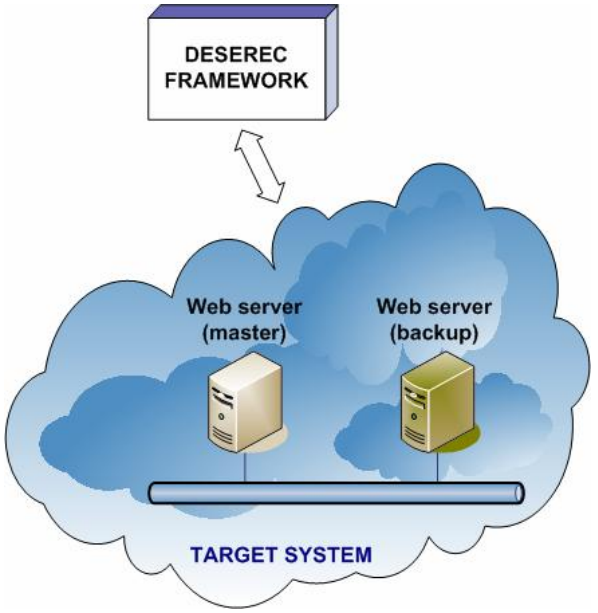
SPL extension example



SPL extension example

Preserving system dependability when a web server goes down

- The master web server is up and running
- If the master web server goes down, the backup one must be active



CONFIGURATION **CONDITION**

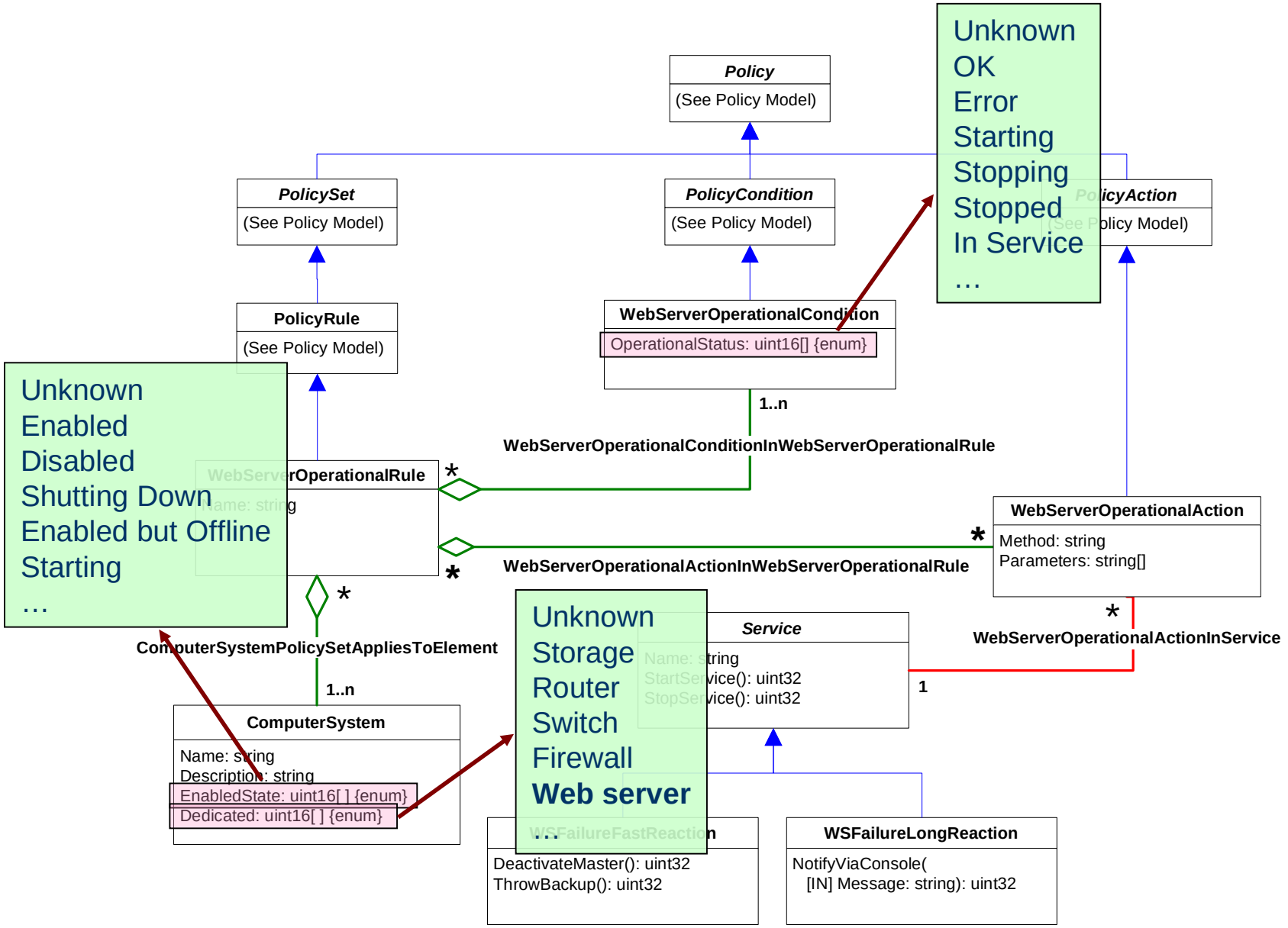
DESEREC framework needs:

- Monitoring of the target system
- To know what happens when the master web server goes down
  - To active the backup one
  - Notify to upper layers in case a reconfiguration of the system is needed

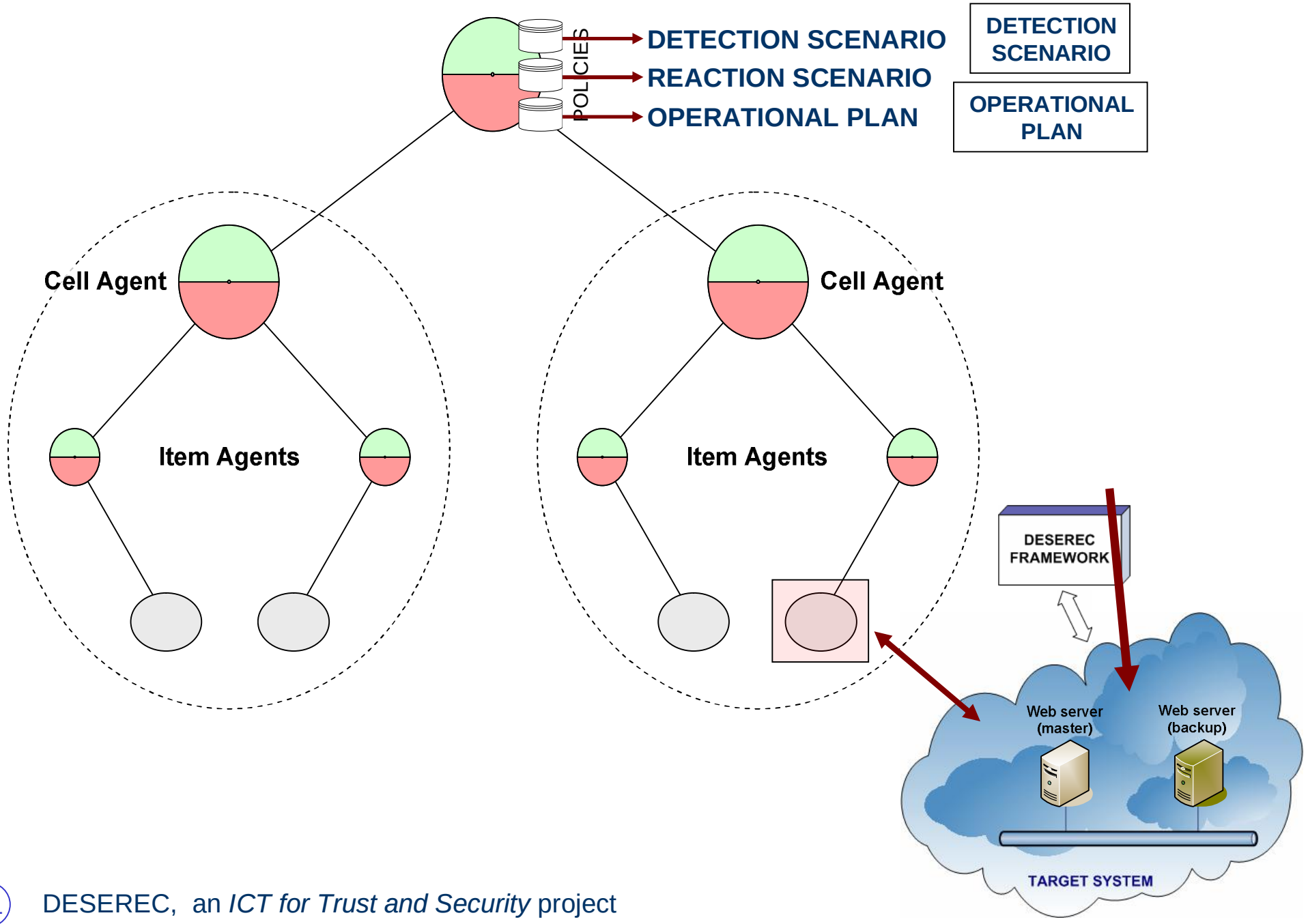
FAST REACTION

LONG-TERM REACTION

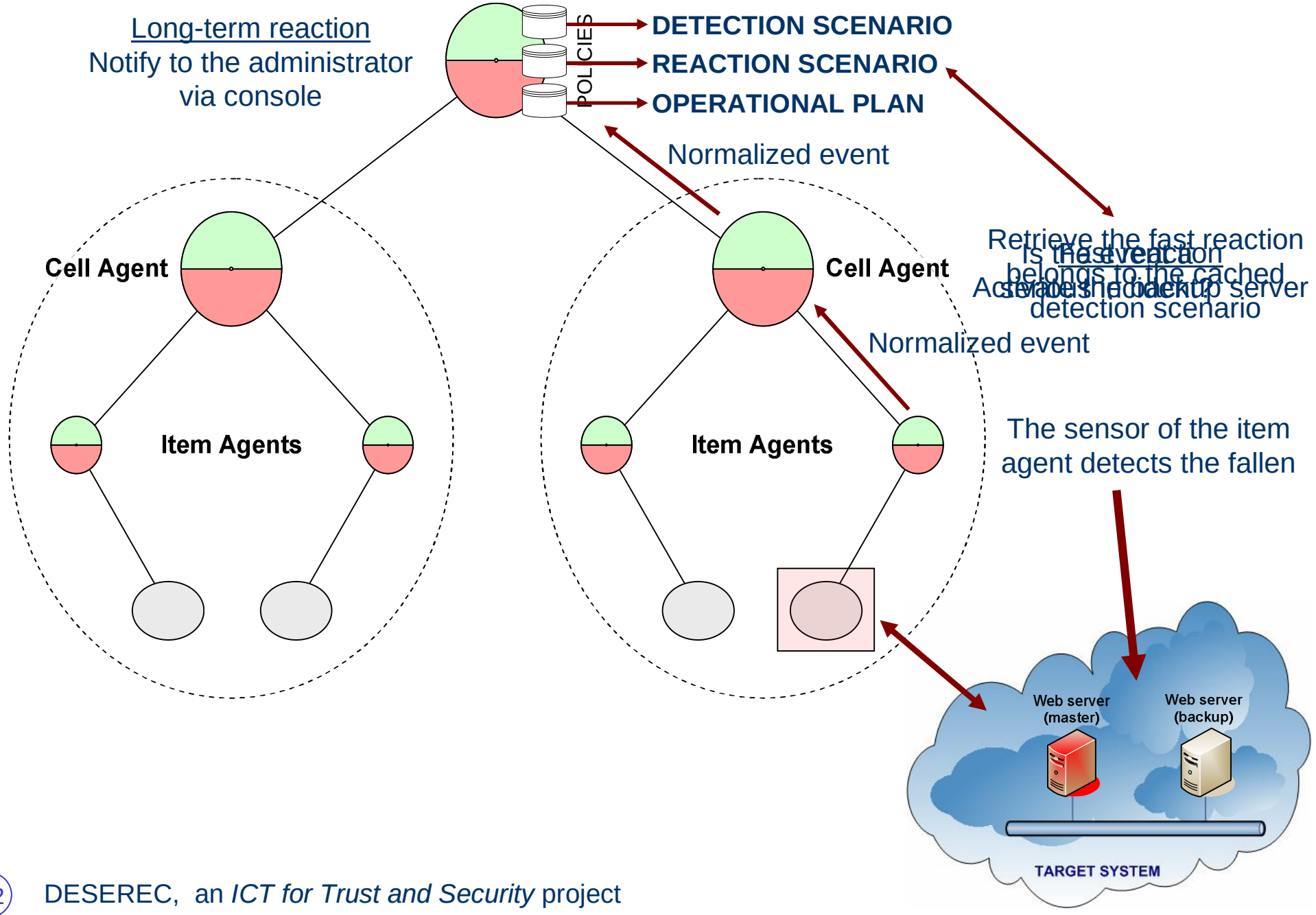
SPL extension example



SPL extension example: Configuration



SPL extension example: When server goes down



Policy Modelling

Gregorio Martínez
University of Murcia (Spain)

Thank you!

DESEREC

Dependability and Security by Enhanced Reconfigurability