A Solution for Goal-oriented Policy Refinement in NFV-MANO Systems

Michel Bonfim, Fred Freitas, Stênio Fernandes

Federal University of Pernambuco (UFPE)

NMRG 57th meeting
IETF 106, Singapore

Centro de Informática U·F·P·E

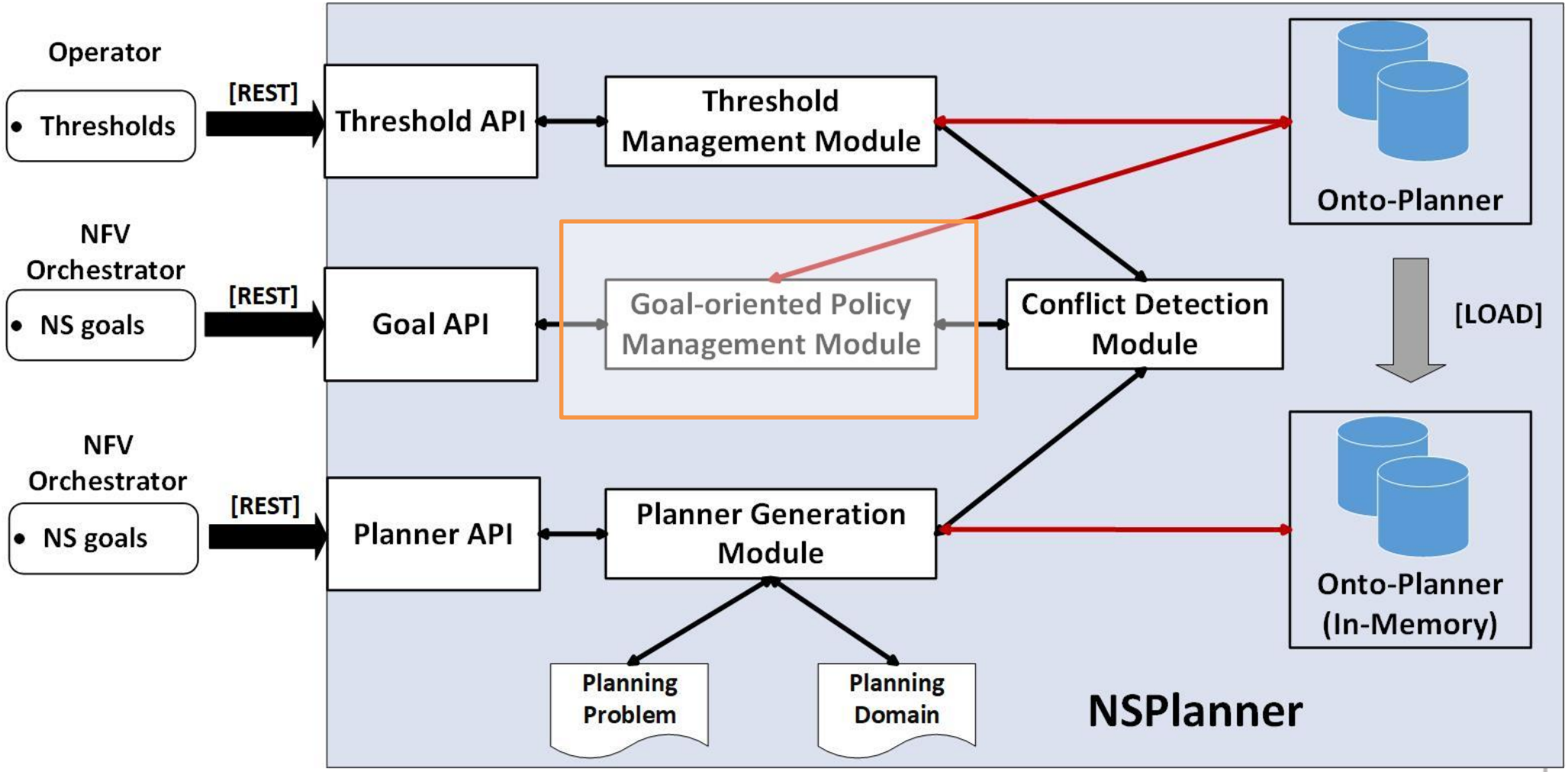
Universidade Federal de Pernambuco

CIn.ufpe.br

Introduction

- **Policy Refinement:** the process of transforming high-level policies into directly enforceable, low-level policies.
- **Problem Statement:** A fully automated refinement process in NFV systems is still an **open issue**.
- **NSPlanner:** A goal-oriented policy refinement procedure for NFV-MANO systems.
  - It uses a well-funded **HTN planner** to perform Goal-oriented policy refinement procedures.
  - It proposes the use of one ontology in OWL 2, called **Onto-Planner**.

NSPlanner Architecture



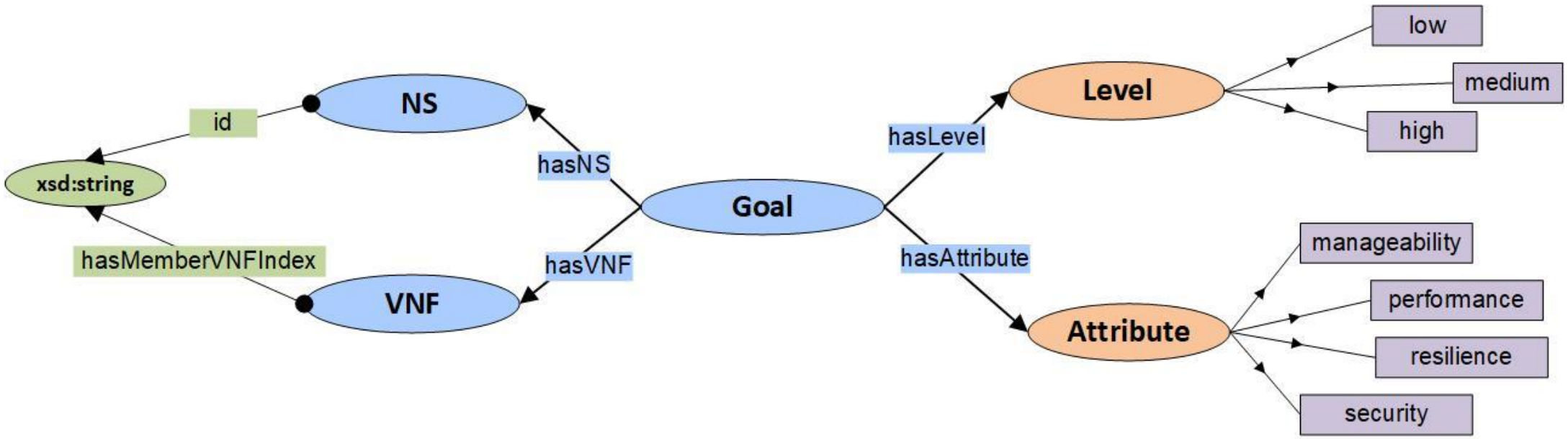
High-level Goal Language

```
1 Language -> <Elements> must receive <Level> <Attributes>
  Elements -> <Element> | <Element><Connective><Elements>
3 Element -> vnf-member-index
  Level -> high | medium | low
5 Attributes -> <Attribute> | <Attribute><Connective><Attributes>
  Attribute -> resiliency | manageability | security | performance
7 Connective -> and
```

EXAMPLE:

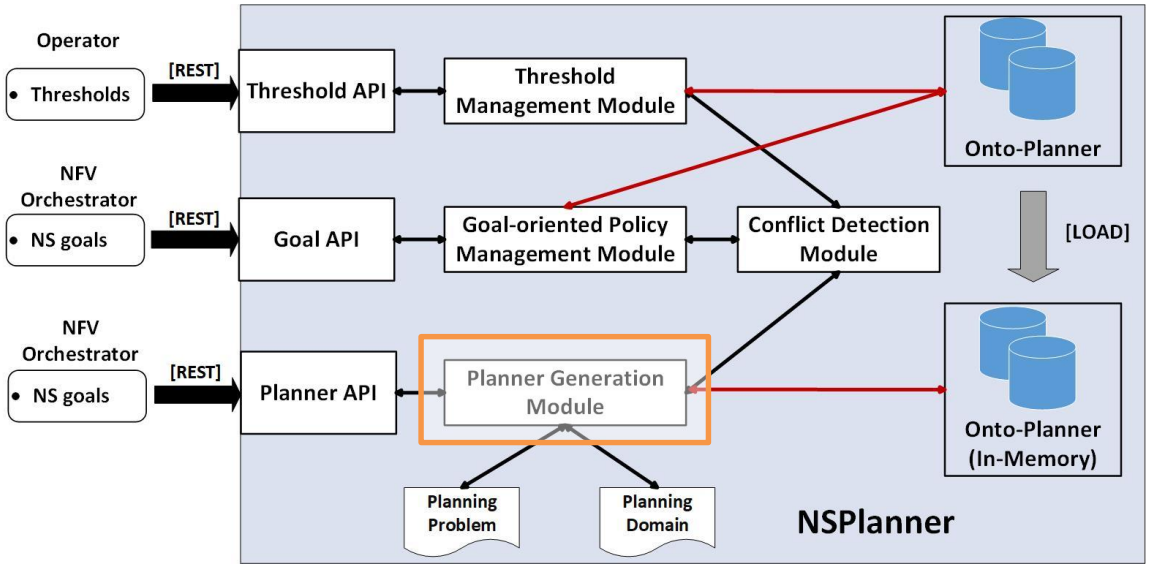
1 and 2 must receive high performance and resilience

Describing Goals

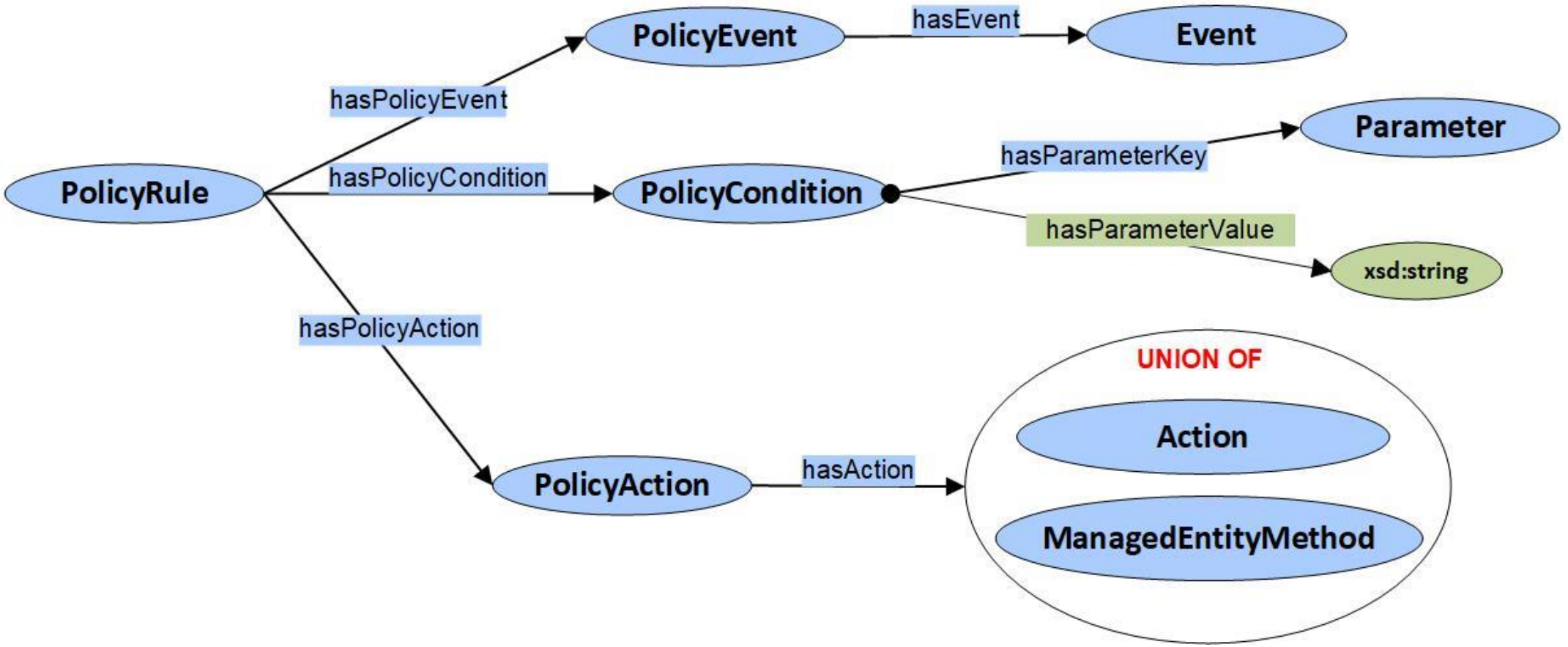


Policy Refinement Procedure

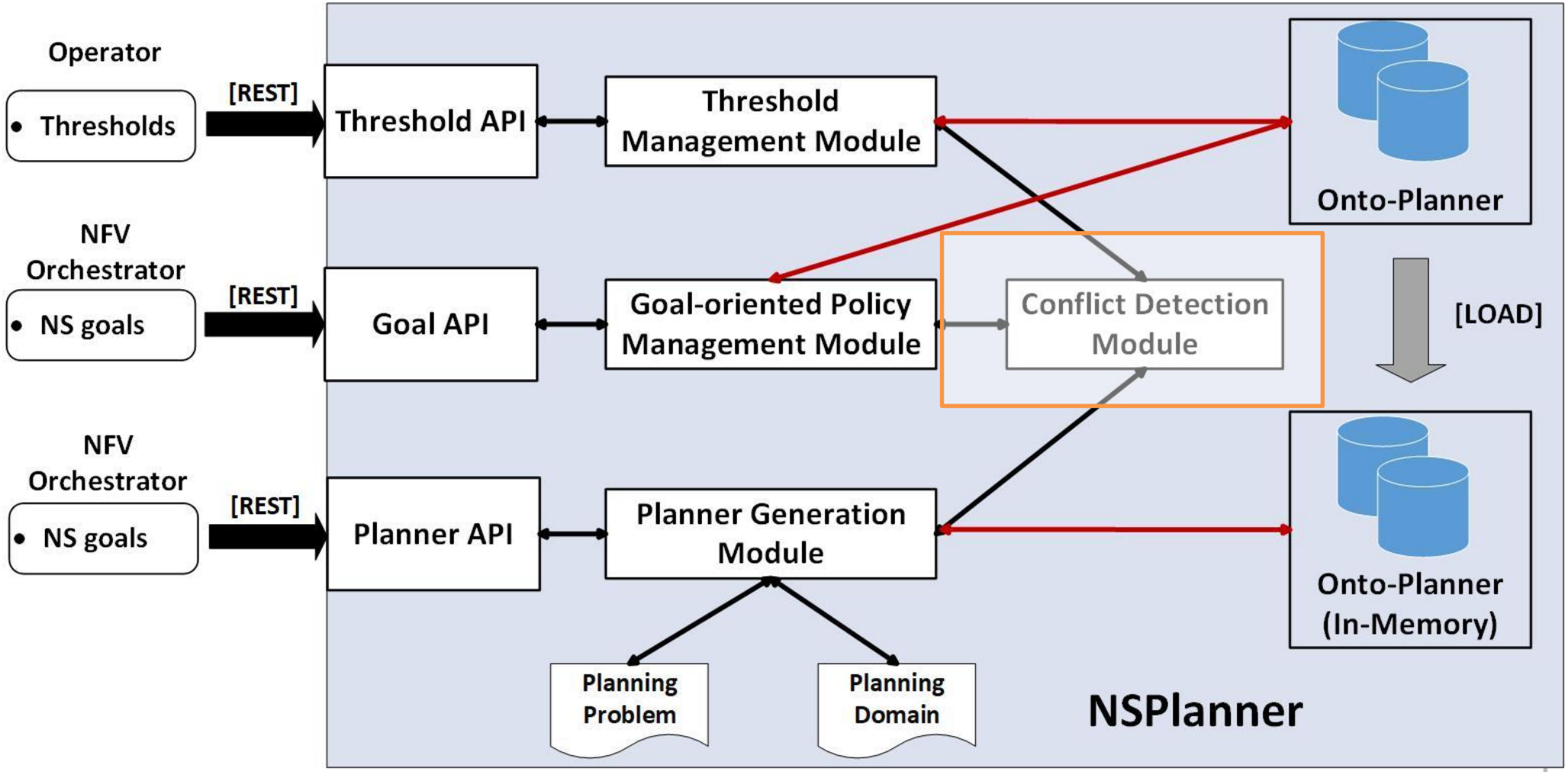
1. It instantiates a copy of Onto-Planner in memory;
2. It extracts the current state of the domain (actions, metrics, alarms) and requested goal information;
3. It builds the planning problem document from above data;
4. It performs the policy refinement;
5. It creates the enforceable policies (ECA rules) and enforceable alarms descriptions into the Onto-Planner copy;
6. It performs DL reasoning to provide inconsistency verification in Onto-Planner copy.



Describing Policy Rules



Managing Policy Conflicts



Managing Policy Conflicts

- **Goal Conflicts:**

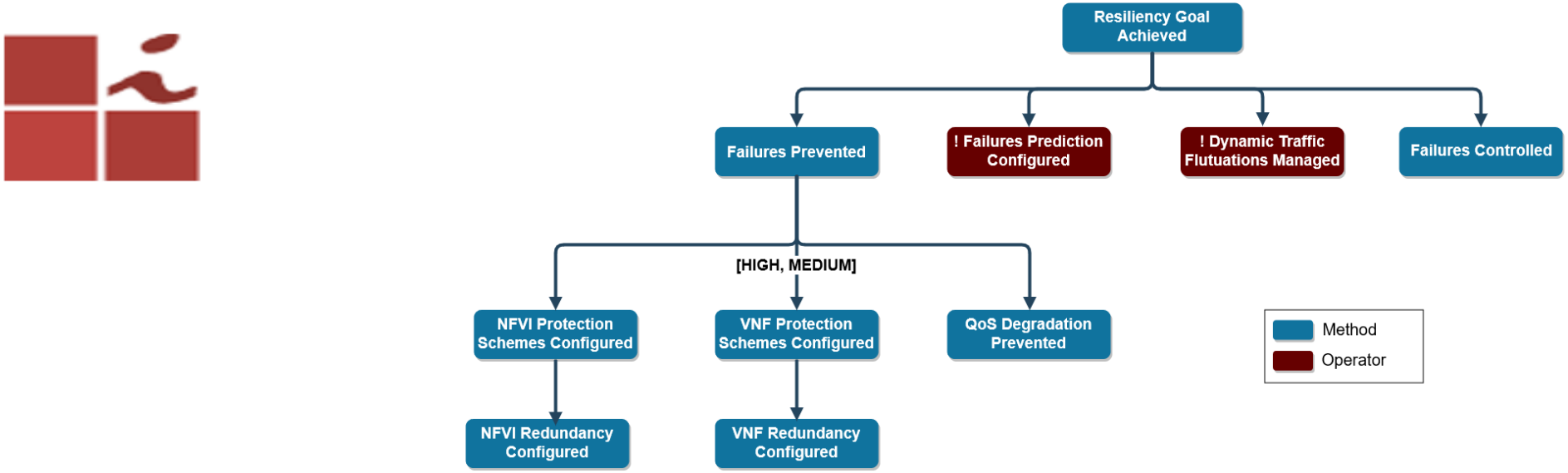
$$Goal(g1) \wedge Goal(g2) \wedge differentFrom(g1, g2)$$
$$\wedge hasNS(g1, ns1) \wedge hasNS(g2, ns2) \wedge id(ns1, idt) \wedge id(ns2, idt)$$
$$\wedge hasVNF(g1, vnf1) \wedge hasVNF(g2, vnf2)$$
$$\wedge hasMemberVNFIndex(vnf1, index) \wedge hasMemberVNFIndex(vnf2, index)$$
$$\wedge Attribute(a) \wedge hasAttribute(g1, a) \wedge hasAttribute(g2, a)$$
$$\rightarrow intersectsWith(g1, g2)$$

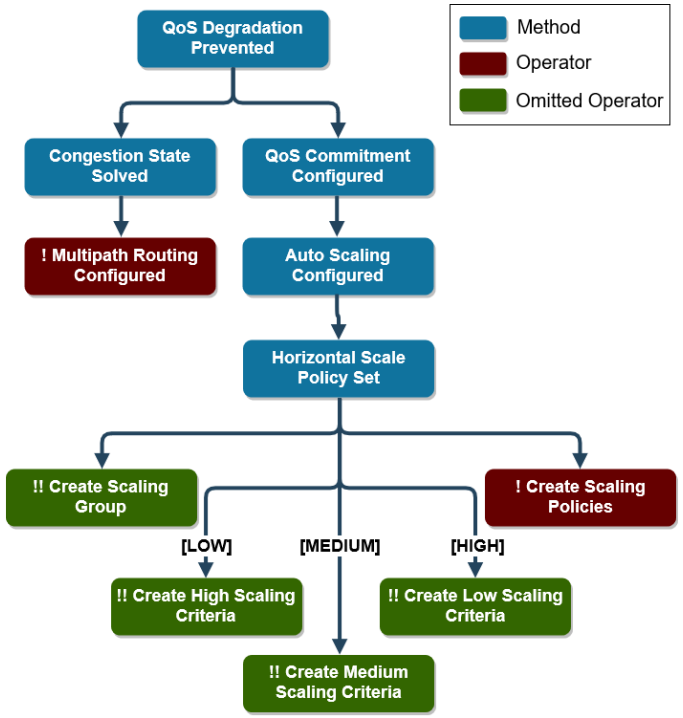
- **ECA Rule Conflicts:**

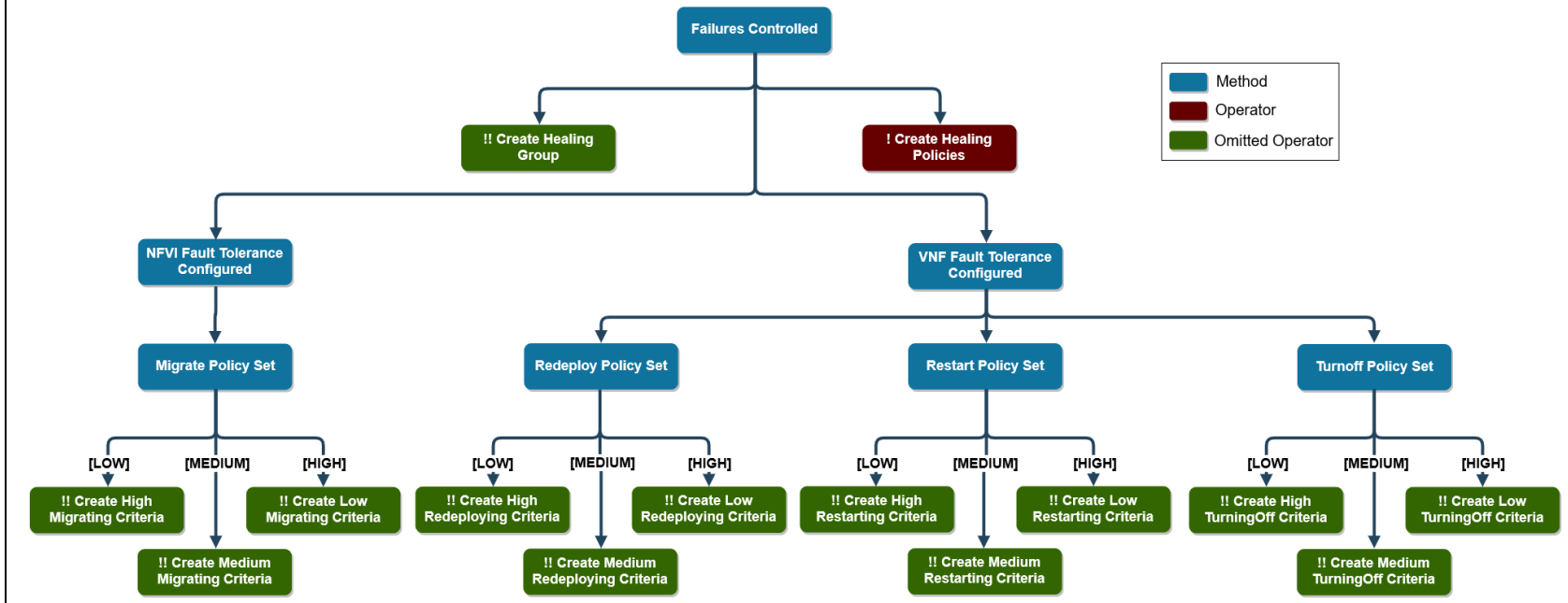
$$PolicyRule(r1) \wedge PolicyRule(r2)$$
$$\wedge hasPolicyEvent(r1, pe1) \wedge hasPolicyEvent(r2, pe2)$$
$$\wedge hasEvent(pe1, e) \wedge hasEvent(pe2, e)$$
$$\wedge hasPolicyCondition(r1, pc1) \wedge hasPolicyCondition(r2, pc2)$$
$$\wedge hasParameterKey(pc1, key) \wedge hasParameterKey(pc2, key)$$
$$\wedge hasParameterValue(pc1, value) \wedge hasParameterValue(pc2, value)$$
$$\wedge hasPolicyAction(r1, pa1) \wedge hasPolicyAction(r2, pa2)$$
$$\wedge hasAction(pa1, a1) \wedge hasAction(pa2, a2) \wedge conflictsWith(a1, a2)$$
$$\rightarrow conflictsWith(a2, a1)$$

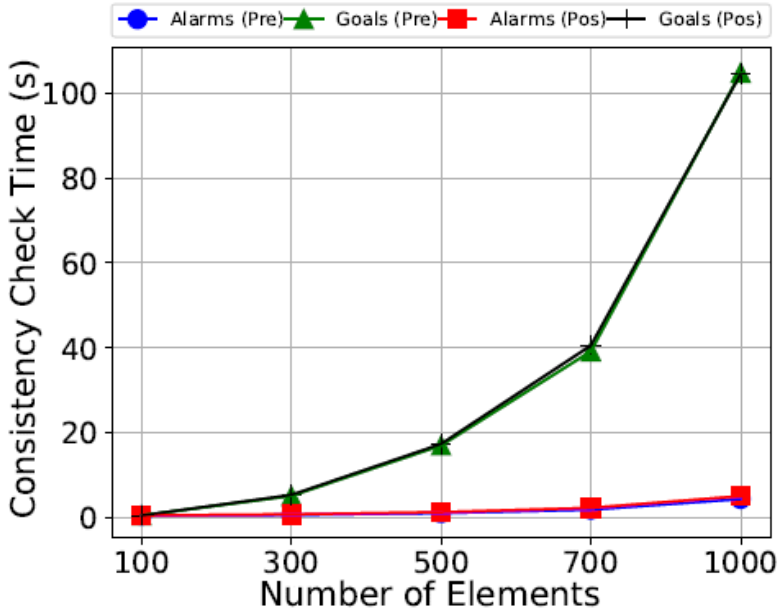
Prototype Implementation

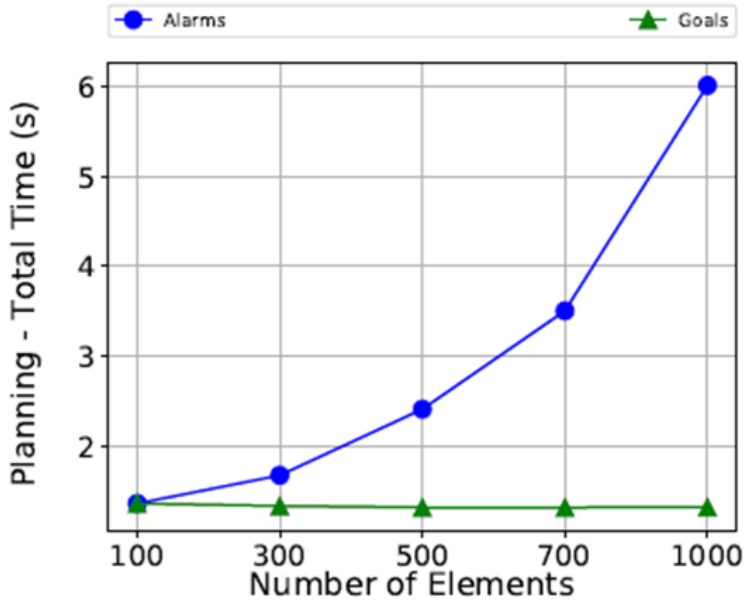
- We developed an NSPlanner prototype in **Java**:
  - We use **Spring Boot** to implement the **RESTful APIs**;
  - We use **OWL API**.
- The Conflict Detection Module uses **HermiT Reasoner**;
- The Planner Management Module uses **SHOP2** as **HTN Planner**:
  - http://www.cs.umd.edu/projects/shop/
- **Use case:** we implemented a planning domain model for the **Resilience Attribute**


Resiliency Goal Achieved
Failures Prevented
! Failures Prediction Configured
! Dynamic Traffic Flutuations Managed
Failures Controlled
[HIGH, MEDIUM]
NFVI Protection Schemes Configured
VNF Protection Schemes Configured
QoS Degradation Prevented
NFVI Redundancy Configured
VNF Redundancy Configured
Method
Operator

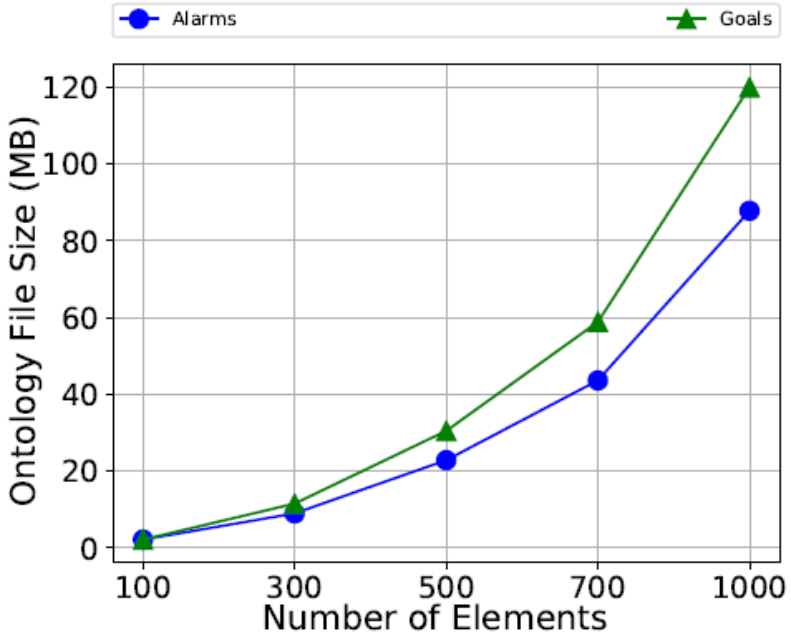

QoS Degradation Prevented
Method
Operator
Omitted Operator
Congestion State Solved
QoS Commitment Configured
! Multipath Routing Configured
Auto Scaling Configured
Horizontal Scale Policy Set
!! Create Scaling Group
! Create Scaling Policies
[LOW]
[MEDIUM]
[HIGH]
!! Create High Scaling Criteria
!! Create Low Scaling Criteria
!! Create Medium Scaling Criteria


Failures Controlled
Method
Operator
Omitted Operator
!! Create Healing Group
! Create Healing Policies
NFVI Fault Tolerance Configured
VNF Fault Tolerance Configured
Migrate Policy Set
Redeploy Policy Set
Restart Policy Set
Turnoff Policy Set
[LOW]
[MEDIUM]
[HIGH]
!! Create High Migrating Criteria
!! Create Low Migrating Criteria
!! Create Medium Migrating Criteria
!! Create High Redeploying Criteria
!! Create Low Redeploying Criteria
!! Create Medium Redeploying Criteria
!! Create High Restarting Criteria
!! Create Low Restarting Criteria
!! Create Medium Restarting Criteria
!! Create High TurningOff Criteria
!! Create Low TurningOff Criteria
!! Create Medium TurningOff Criteria





NSPlanner performs and scales well



Contributions to NMRG

1. Enables goal-oriented policy management:
   - NSPlanner focus on intent-* specific aspects, since goal policies describe desired states in an environment, not a sequence of actions.

2. Provides a decomposition logic to resolve intent to relevant service and select appropriate mgmt function(s).

Thank you! Questions?

Michel Bonfim

<msb6@cin.ufpe.br>

Centro de Informática U·F·P·E

Universidade Federal de Pernambuco