TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG

Automatic Policy Refinement Using OWL-S and Semantic Infrastructure Information

Torsten Klie, Lars Wolf

Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Benjamin Ernst

Resco GmbH
Hamburg

Outline

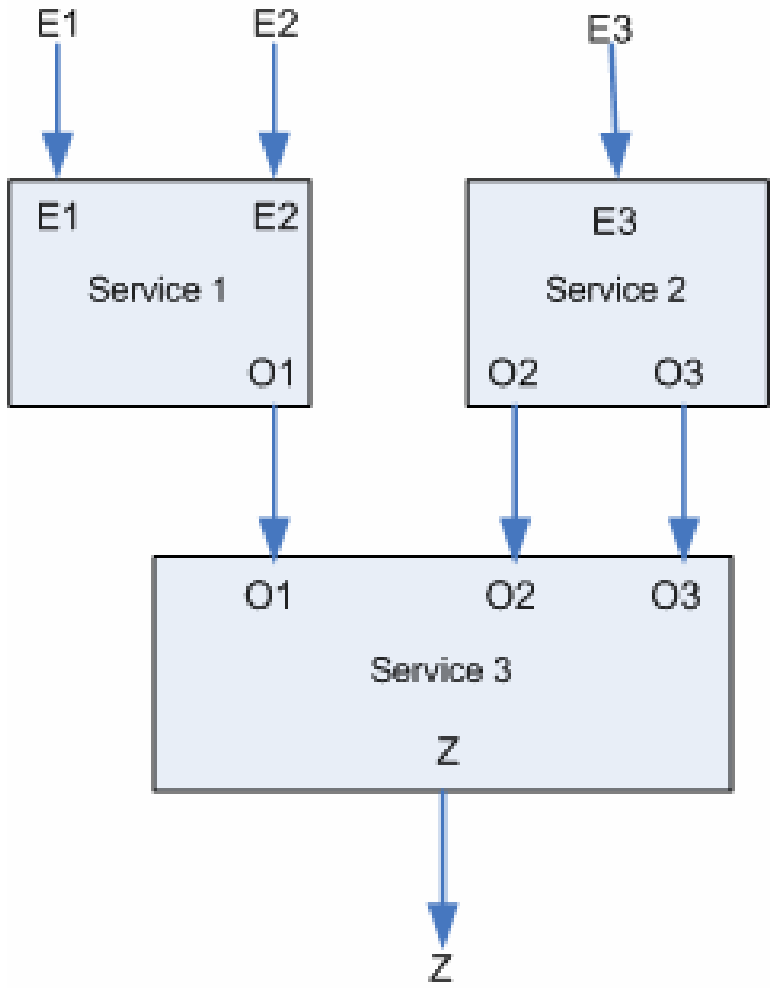
- **Introduction**
  - Motivation
- **Policy-based Architecture for Autonomic Communications**
- **Web Services Composition with SEMPR and NINO**
  - Policy Refinement with Web Service Composition
  - NINO – a **N**etwork **IN**frastructure **O**ntology
  - SEMPR – a **SEM**antic **P**olicy **R**efinement engine
- **Case Study: Home Networks**
- **Conclusions and Outlook**

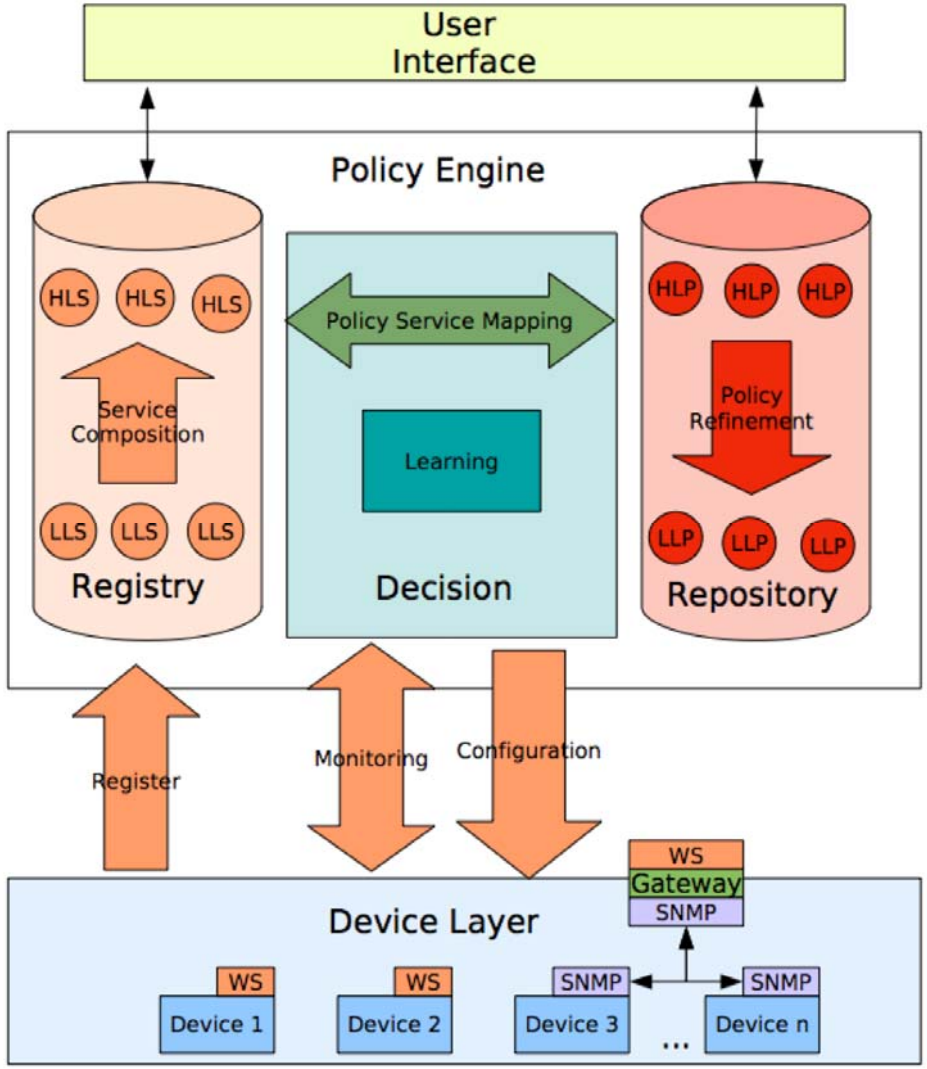
Motivation

- **Policy-based Network Management**
  - Reduce complexity
  - Govern the behavior of the network with rules
  - Policies exist on different levels of abstraction
  - Policy refinement: still a big challenge
- **New Management Technology: Web Services**
  - Used in standardization approaches
    - OASIS WSDM
    - DMTF WS Management
    - IETF NETCONF (optional)
  - Advantages: several; esp. composability
  - Use Web service composition to do policy refinement

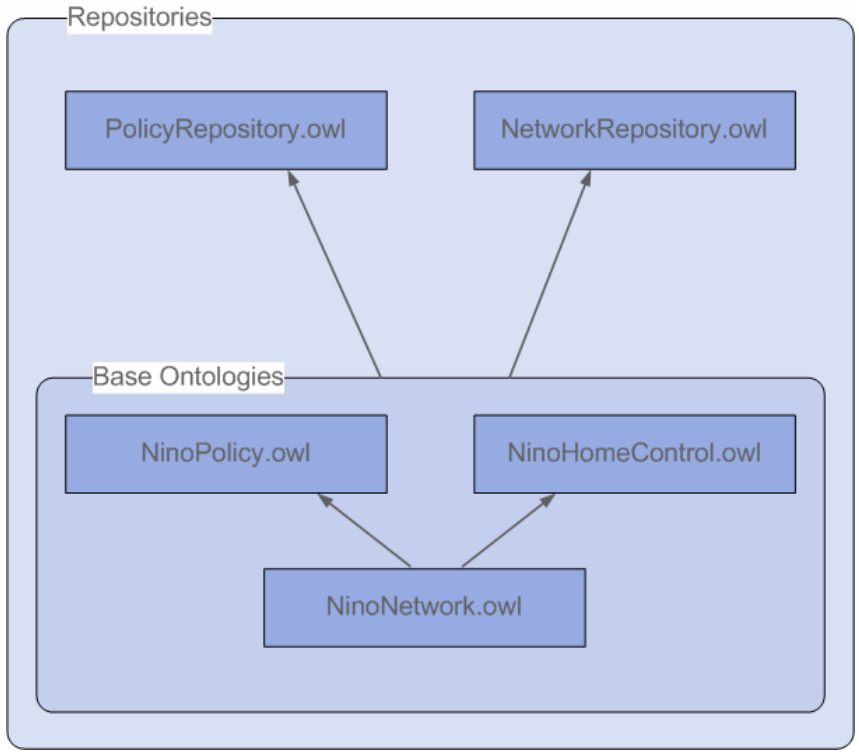
Web Services Composition

- **Combining services: aggregated tasks**
- **Web services composition**
  - Synthesis: Select the services
    - Template-based (BPEL)
    - (Semi-)Automatic
  - Orchestration: Executed composed service
- **OWL-S**
  - Ontology for semantic description of Web services
  - Uses OWL to describe functional properties (IOPEs: Input, Output, Preconditions, Effects)



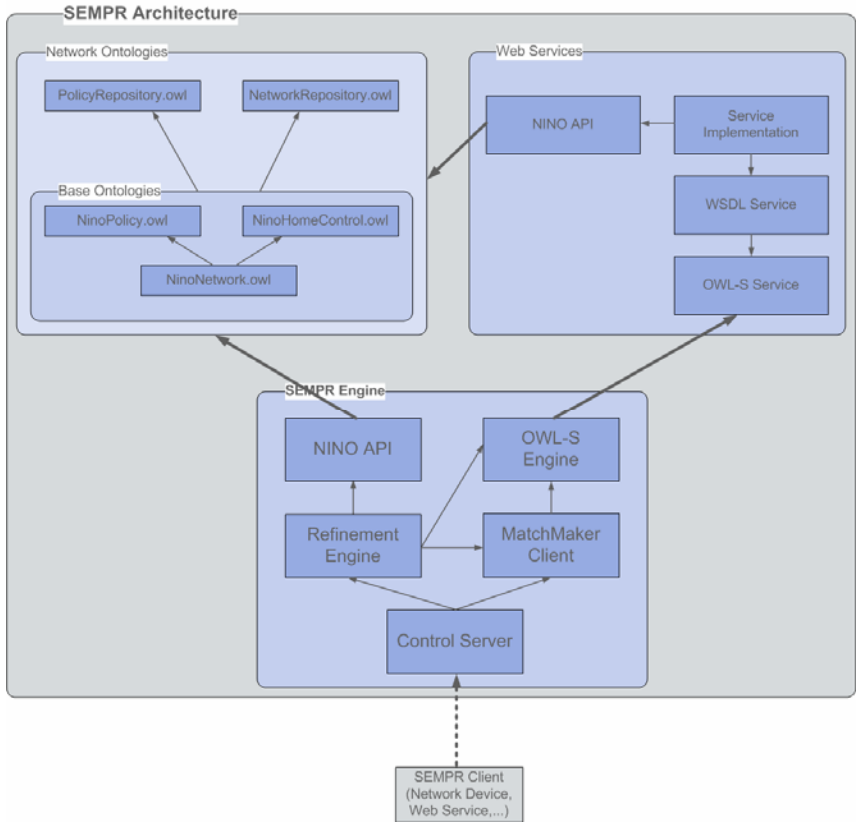

User Interface
Policy Engine
HLS
HLS
HLS
Service Composition
LLS
LLS
LLS
Registry
Policy Service Mapping
Learning
Decision
HLP
HLP
HLP
Policy Refinement
LLP
LLP
LLP
Repository
Register
Monitoring
Configuration
WS
Gateway
SNMP
Device Layer
WS
Device 1
WS
Device 2
SNMP
Device 3
SNMP
Device n
...

- **Basis: semantic Web services in OWL-S**
  - LLS: Management services offered by the devices
  - HLS: Composed services
- **NINO: Network Infrastructure Ontology**
  - OWL
  - Repositories
    - Network repository
    - Policy repository
  - Base ontologies (extensible)
    - Network ontology
    - Policy ontology
    - Home control ontology (used in the case study)
    - User control ontology (used in another case study)

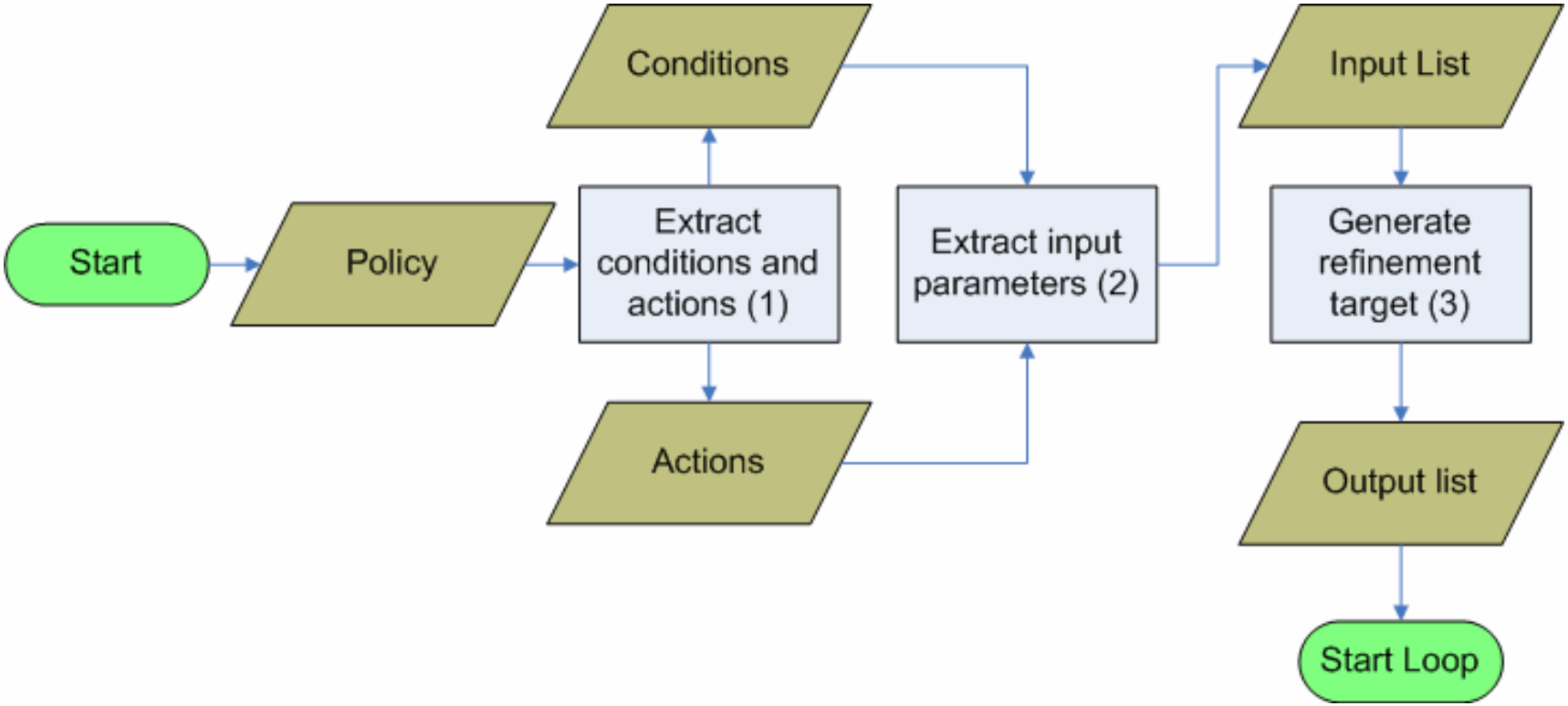


SEMPR Engine

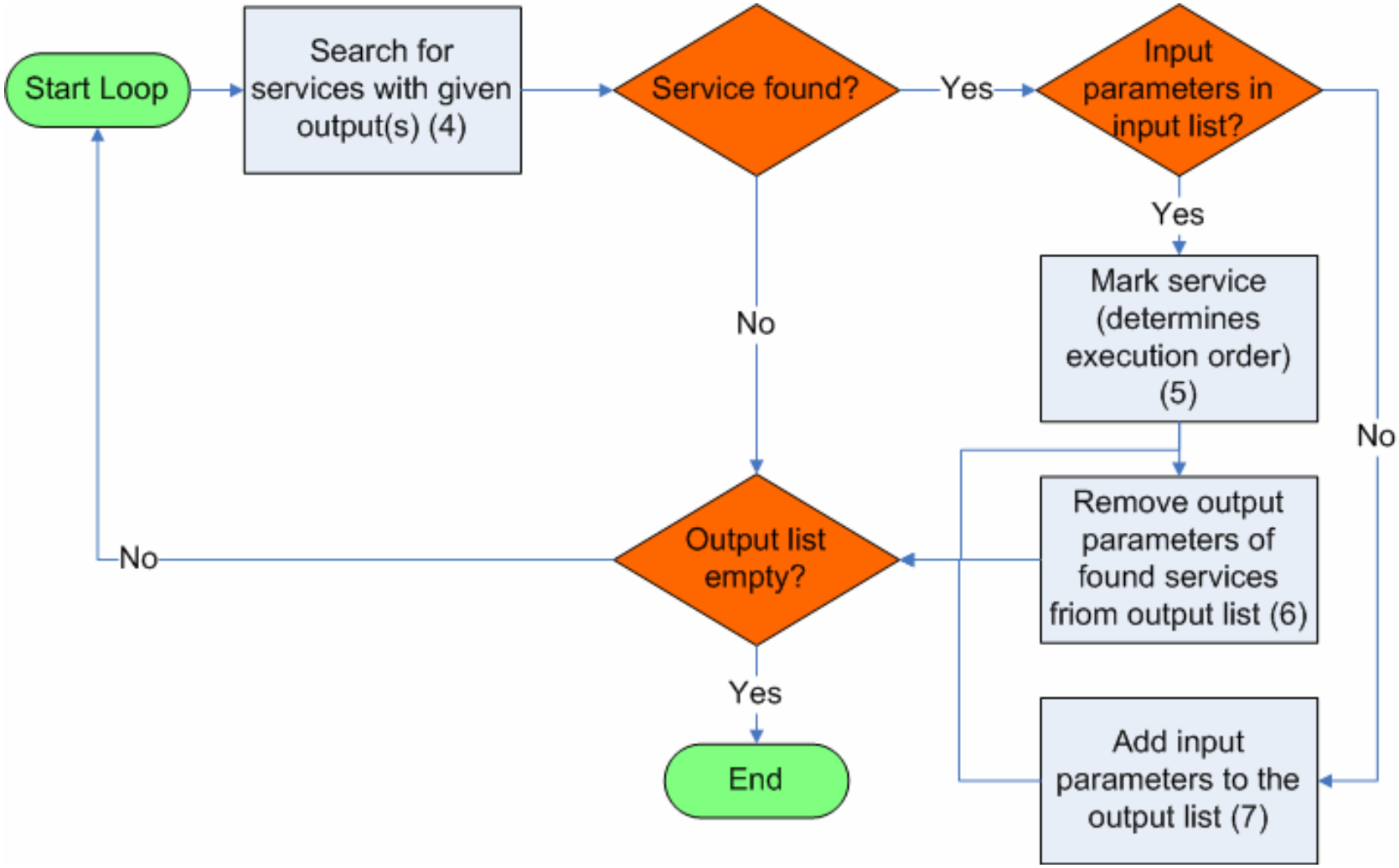
- **Core component of the SEMPR architecture**
- **NINO API**
  - Links SEMPR and NINO
  - Functions for reading policies and network device descriptions
- **Refinement Engine**
  - Get policies via NINO API
  - Refinement with matchmaker client
  - Tell OWL-S engine to execute service
- **Matchmaker Client**
  - Provides OWL-S services matching given IOPEs
- **OWL-S Engine**
  - Executes composite Web service
- **Control Server**
  - Controls SEMPR
  - Initiates refinement process
  - Adds/removes new devices /services /policies



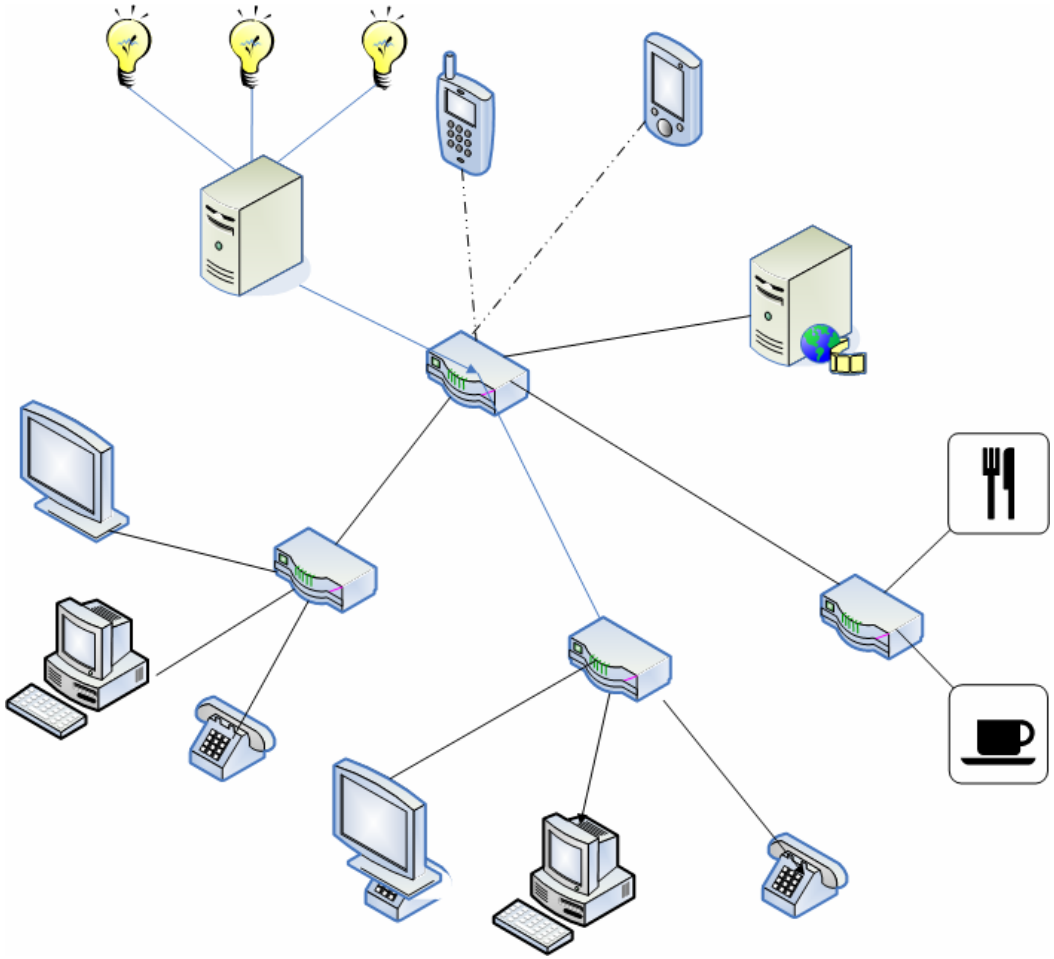
Policy Refinement Algorithm 1/2



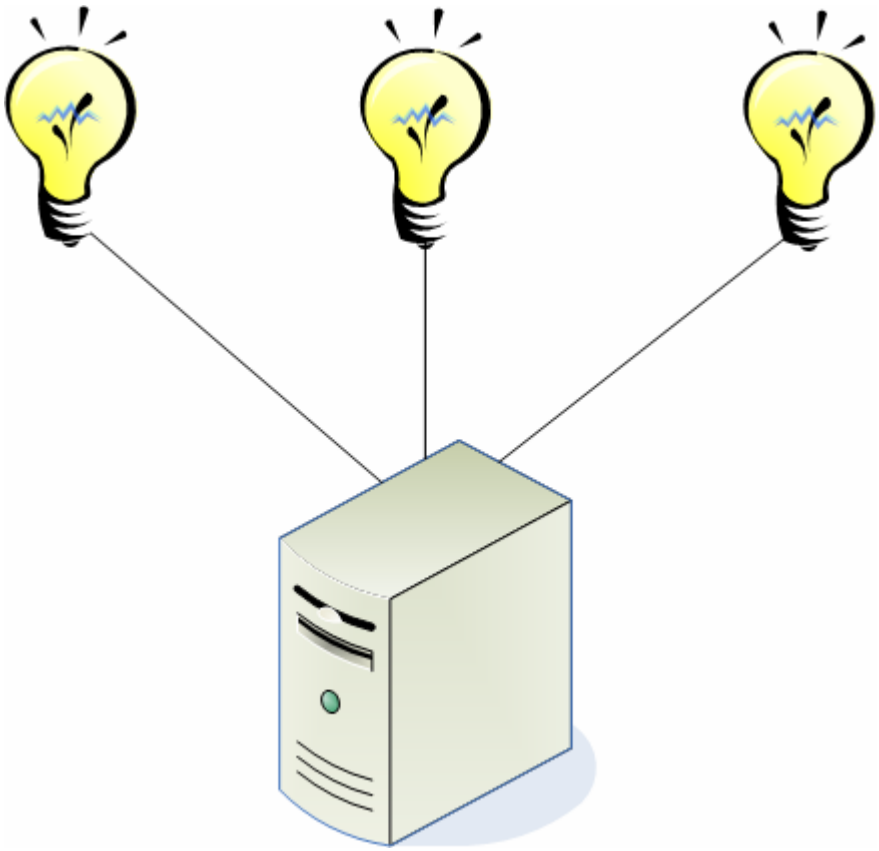
Policy Refinement Algorithm 2/2



Home Area Network



Home Area Network 2



Policy Refinement Example 1/8

- **Condition: `SensorConditionByRoom`**
  - Input: `Room`, `SensorStatus`, `SensorType`
  - Fulfilled if the sensor (of a given type) in the given room is in the given state
- **Action: `LightActionByRoom`**
  - Input: `LightStatus`, `Room`
  - Switches the light in a given room to a given state

- **Services:**
  - `getLight`
    - Input: `Room`
    - Output: `Light`
  - `getSensor`
    - Input: `Room`, `SensorType`
    - Output: `Sensor`
  - `getHCS`
    - Input: `Light`
    - Output: `NetworkResource`
  - `switchLightBySensor`
    - Input: `Light`, `LightStatus`, `SensorStatus`, `Sensor`, `NetworkResource`
    - Output: -

Policy Refinement Example 2/8


```
<j.0:Policy rdf:ID="Policy_4">
   <j.0:Name
   rdf:datatype="http://www.w3.org/2001/XMLSchema#string">VacPol1</j.0
   :Name>
   <j.0:hasAction><j.0:LightActionByRoom rdf:ID="LightActionByRoom_6">
     <j.0:Name
   rdf:datatype="http://www.w3.org/2001/XMLSchema#string">LighAction2<
   /j.0:Name>
     <j.0:hasLightStatus rdf:resource="#On"/>
     <j.0:hasRoom rdf:resource="#Room_Hall"/>
   </j.0:LightActionByRoom></j.0:hasAction>
   <j.0:hasCondition><j.0:SensorConditionByRoom
   rdf:ID="SensorConditionByRoom_15">
     <j.0:hasRoom rdf:resource="#Room_Hall"/>
     <j.0:hasSensorType rdf:resource="#MovementSensor"/>
     <j.0:Name
   rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SCondByRoom
   1</j.0:Name>
     <j.0:hasSensorStatus rdf:resource="#MovementDetected"/>
    </j.0:SensorConditionByRoom></j.0:hasCondition>
</j.0:Policy>
```

Policy Refinement Example 3/8

- **Condition: `SensorConditionByRoom`**
  - Input: `Room`, `SensorStatus`, `SensorType`
  - Fulfilled if the sensor (of a given type) in the given room is in the given state
- **Action: `LightActionByRoom`**
  - Input: `LightStatus`, `Room`
  - Switches the light in a given room to a given state

- **Preparation: Extracting inputs and outputs**
- **Inputs:**
  - `SensorStatus`
  - `Room`
  - `SensorType`
  - `LightStatus`
- **Outputs:**
  - `SensorConditionByRoom`
  - `LightActionByRoom`
- **Services: n/a**

Policy Refinement Example 4/8

- **0th iteration: Looking for services that can produce the desired output.**
- **Inputs:**
  - `SensorStatus`
  - `Room`
  - `SensorType`
  - `LightStatus`
- **Outputs:**
  - `SensorConditionByRoom`
  - `LightActionByRoom`
  - `NetworkResource`
  - `Light`
  - `Sensor`
- **Services: `switchLightBySensor`**

- **Services:**
  - `getLight`
    - Input: `Room`
    - Output: `Light`
  - `getSensor`
    - Input: `Room`, `SensorType`
    - Output: `Sensor`
  - `getHCS`
    - Input: `Light`
    - Output: `NetworkResource`
  - `switchLightBySensor`
    - Input: `Light`, `LightStatus`, `SensorStatus`, `Sensor`, `NetworkResource`
    - Output: -

Policy Refinement Example 5/8

- **1st iteration: Looking for services that can produce the desired output.**
- **Inputs:**
  - `SensorStatus`
  - `Room`
  - `SensorType`
  - `LightStatus`
- **Outputs:**
  - `SensorConditionByRoom`
  - `LightActionByRoom`
  - `NetworkResource`
  - `Light & Sensor`
- **Services:**
  - `switchLightBySensor`
  - `getLight(1)`
  - `getHCS`
  - `getSensor(1)`

- **Services:**
  - `getLight`
    - Input: `Room`
    - Output: `Light`
  - `getSensor`
    - Input: `Room`, `SensorType`
    - Output: `Sensor`
  - `getHCS`
    - Input: `Light`
    - Output: `NetworkResource`
  - `switchLightBySensor`
    - Input: `Light`, `LightStatus`, `SensorStatus`, `Sensor`, `NetworkResource`
    - Output: -

Policy Refinement Example 6/8

- **2nd iteration: Looking for services that can produce the desired output.**
- **Inputs:**
  - `SensorStatus`
  - `Room`
  - `SensorType`
  - `LightStatus`
  - `Light`
  - `Sensor`
- **Outputs:**
  - `SensorConditionByRoom`
  - `LightActionByRoom`
  - `NetworkResource`
- **Services:**
  - `switchLightBySensor`
  - `getLight(1)`
  - `getHCS(2)`
  - `getSensor(1)`

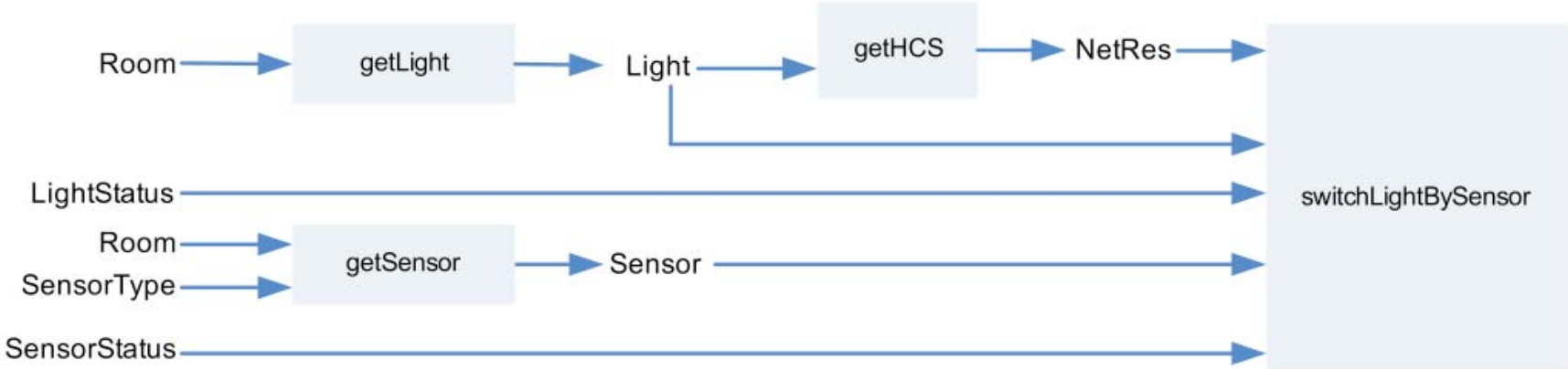
- **Services:**
  - `getLight`
    - Input: `Room`
    - Output: `Light`
  - `getSensor`
    - Input: `Room`, `SensorType`
    - Output: `Sensor`
  - `getHCS`
    - Input: `Light`
    - Output: `NetworkResource`
  - `switchLightBySensor`
    - Input: `Light`, `LightStatus`, `SensorStatus`, `Sensor`, `NetworkResource`
    - Output: -

Policy Refinement Example 7/8

- **3rd iteration: Looking for services that can produce the desired output.**
- **Inputs:**
  - `SensorStatus`
  - `Room`
  - `SensorType`
  - `LightStatus`
  - `Light`
  - `Sensor`
  - `NetworkResource`
- **Outputs:**
  - `SensorConditionByRoom`
  - `LightActionByRoom`
- **Services:**
  - `switchLightBySensor(3)`
  - `getLight(1)`
  - `getHCS(2)`
  - `getSensor(1)`

- **Services:**
  - `getLight`
    - Input: `Room`
    - Output: `Light`
  - `getSensor`
    - Input: `Room`, `SensorType`
    - Output: `Sensor`
  - `getHCS`
    - Input: `Light`
    - Output: `NetworkResource`
  - `switchLightBySensor`
    - Input: `Light`, `LightStatus`, `SensorStatus`, `Sensor`, `NetworkResource`
    - Output: -

Policy Refinement Example 8/8



- **Result of the refinement process sent to OWL-S engine**
  - Parameters
  - Services to execute
  - Execution order
- **OWL-S engine executes the services**

Summary and Outlook

- **Summary**
  - Used Web service composition for refining policies
  - Policies, infrastructure (Web services and devices) described with OWL and OWL-S
  - NINO Ontology
  - SEMPR architecture & Home Network Example
- **Future Work**
  - Use with larger ontologies
  - Use a lighter Web service environment
  - Support for precondition and effects
  - Integrate prototype in our autonomic management architecture
  - Compare (& combine?) with other refinement approaches

The End

- **Questions or comments?**
  - Here and now: speak up!
  - Via e-mail to `tklie@ibr.cs.tu-bs.de`