Towards an Ontology-Based Framework for a Behavior-Oriented Integration of the IoT

Domenico Cantone, Carmelo Fabio Longo,
Marianna Nicolosi-Asmundo, Daniele Francesco Santamaria,
Corrado Santoro

Deparment of Mathematics and Computer Science,
University of Catania

The 20th Workshop From Objects to Agents (WOA 2019)
Parma, Italy
26-28 June, 2019

Introduction

- The Internet of things (IoT) is a system of interrelated devices, applications, and users, provided with the ability to transfer data over a network.
- IoT is the extension of Internet connectivity to physical devices and everyday objects.
- In IoT systems, objects acquire, share, and act on data collected from the environment.
- One of the founding principles of the IoT is a transparent communication system among devices.
- Transparency is necessary to enable (real) IoT.

Introduction

The connectivity, networking and communication protocols used by these web-enabled devices largely depend on the specific IoT applications deployed (IoTAgenda, 2019):

- General purpose and Plug&Play IoT systems are hard to realize.

In the case of domotic assistants, clients are tight to providers:

- They buy from a compatibility list of "certified" devices.
- Building certified devices requires a registration to the provider (and usually commercial licensing).
    - E.g., Alexa Gadgets Toolkit (Beta).

Can I build a device by myself without licensing?.

Can I build a platform-independent device?.

Introduction

- Semantic web promotes common data formats and exchange protocols on the Web, and languages for them.
- Applications that automatically process information, instead of just presenting it, and exchange information with other applications, need appropriate languages with formally defined syntax and semantics.
- The W3C indicates the Web Ontology Language (OWL), a family of knowledge representation languages relying on Description Logics (DLs) (Allemang, 2011), as a standard for representing ontologies.
- With the aid of OWL and OWL-complaint reasoners, we can infer and process implicit information present in the data thus obtaining new facts. Automated reasoning systems allow one to also verify the consistency of the model and query the data-set.

Introduction

Integration of ontologies with IoT had a deep treatment in several contexts.

- Hendler, 2001. Agents and the semantic web.
- Wang et al., 2012. A Comprehensive Ontology for Knowledge Representation in the Internet of Things.
- Hadzic et al., 2014. Ontology-Based MultiAgent Systems.
- Akshay et al., 2014. A Unified Semantic Knowledge Base for IoT.
- Fritzsche et al., 2017. Ontologies within semantic interoperability ecosystems.
- Bajaj et al., 2017. A study of existing Ontologies in the IoT-domain, 2017.
- Seydoux et al., 2017. Capturing the Contributions of the Semantic web to the IoT: a Unifying Vision.

Motivation

Ontologies for IoT are mainly focused on:

- Mapping sensors.
- Mapping services.
- Mapping MAS.

But in a system of interrelated devices:

- How do devices know what other devices may do?.
- Is there any transparent communication protocol among devices which abstracts their implementation?.
- Are devices plug-and-play, in the sense that no third-part intervention is required?.
- How users take part in such a system?.

Our approach

- Providing a sufficiently general modelling of Agents.
    - Agents are capable of performing actions.
    - Component are only subjected to those actions.
    - Such paradigm can include other paradigms.
- Providing a sufficiently general modelling of agents behaviors and requests of actions.
    - Agents expose behaviors.
    - Agents may request actions (tasks).
    - Agents know if and how a task is performed.
    - Behaviors change without breaking the system.
    - Defining general agent behaviors which can be adopted.
- Agents exchange information.
    - Regardless the content.
    - Regardless the agent implementation.
    - Agents decide which action should be performed.

Our approach

- We define an ontology with all the mentioned features.
  - OASIS - An ontology for Agents, Systems, and Integration of services.
  - Agents exchange RDF fragments of OASIS.
- We apply OASIS to a domotic case study.
  - Agents sent requests.
  - Agents expose their behaviors.
  - SPARQL queries are built ad-hoc to select agents, devices, and actions to perform.
  - A suitable RDF fragment is sent to the selected agent.
  - The selected agent performs the action requested.
  - The selected agent updates its caller with the execution status of the action.

OASIS

OASIS
An Ontology for Agents, Systems, and Integration of Services

OASIS

OASIS mainly maps:

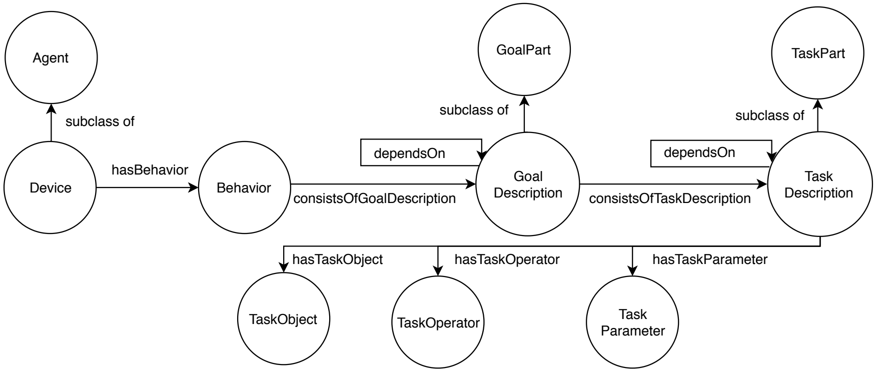
- Agent behaviors.
- Configurations of agents and components.
- Requests of tasks from agent to agent.
- Execution of tasks.

Agent behaviors

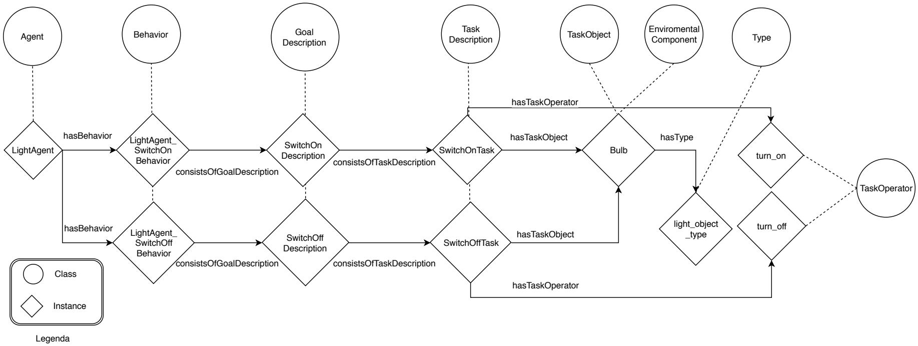
Agent behaviors:

- Agents are described by one or more behaviors.
- A behavior is a set of, possibly dependent, goals.
- A goal is a set of, possibly dependent, tasks.
- A task consists of an action, an object, and, possibly, a parameter.

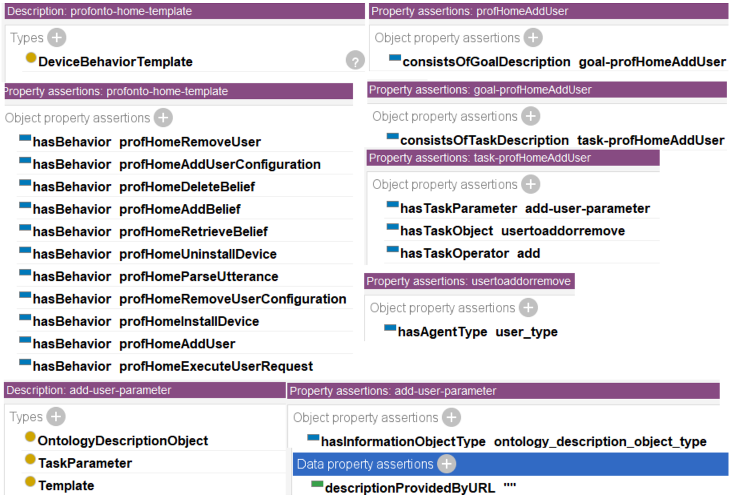
Agent Behavior: general schema



Agent Behavior: example



Agent Behavior: example



Description: profonto-home-template

Types

DeviceBehaviorTemplate

Property assertions: profonto-home-template

Object property assertions

hasBehavior profHomeRemoveUser
hasBehavior profHomeAddUserConfiguration
hasBehavior profHomeDeleteBelief
hasBehavior profHomeAddBelief
hasBehavior profHomeRetrieveBelief
hasBehavior profHomeUninstallDevice
hasBehavior profHomeParseUtterance
hasBehavior profHomeRemoveUserConfiguration
hasBehavior profHomeInstallDevice
hasBehavior profHomeAddUser
hasBehavior profHomeExecuteUserRequest

Property assertions: profHomeAddUser

Object property assertions

consistsOfGoalDescription goal-profHomeAddUser

Property assertions: goal-profHomeAddUser

Object property assertions

consistsOfTaskDescription task-profHomeAddUser

Property assertions: task-profHomeAddUser

Object property assertions

hasTaskParameter add-user-parameter
hasTaskObject usertoaddorremove
hasTaskOperator add

Property assertions: usertoaddorremove

Object property assertions

hasAgentType user_type

Description: add-user-parameter

Types

OntologyDescriptionObject
TaskParameter
Template

Property assertions: add-user-parameter

Object property assertions

hasInformationObjectType ontology_description_object_type

Data property assertions

descriptionProvidedByURL ""

We have recently introduced behavior templates (class Template) for managing general models of behaviors and of task objects.

Configuration and connections

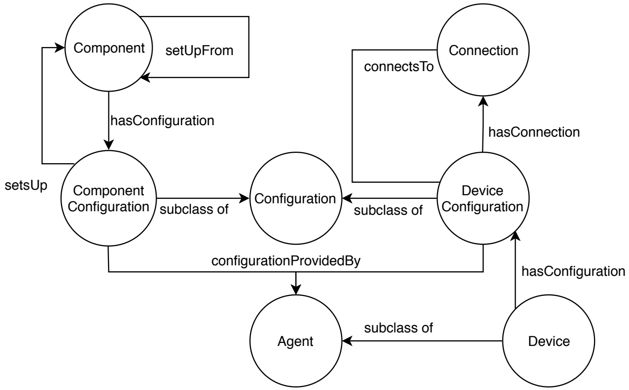
Configurations:

- Configurations are defined for components and agents.
- A configuration of an object defines the personal projection of the object in the perceptional world of the configuring agent.
- Basically, a configuration introduces a “copy” of the configurable object having all the features of the configured object and interacting with the configuring agent.

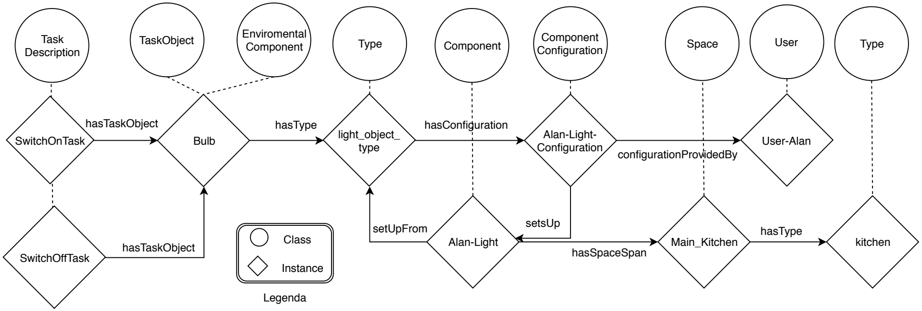
Connections:

- define the entry point of an agent.
- are used to communicate with agents.
- defines the information used to communicate.

Component configurations: general schema



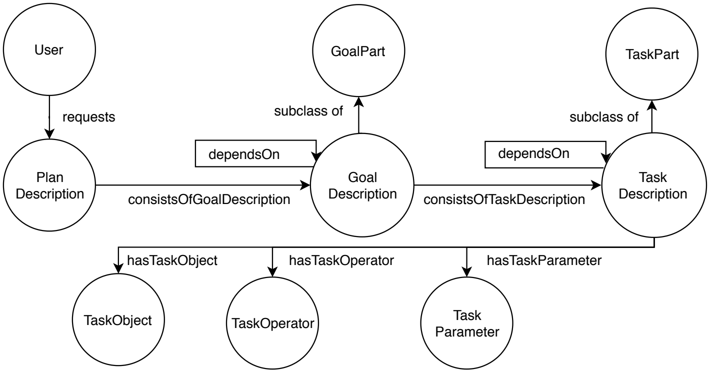
Component configurations: example



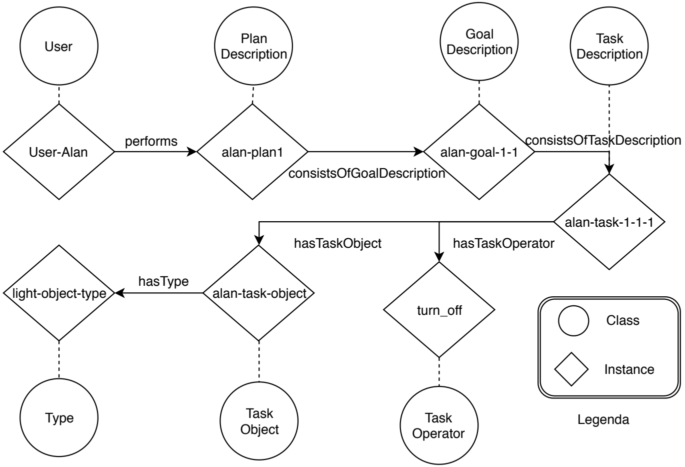
Agent requests

- Requests are used to demand for the execution of actions.
- They can be sent to a request dispatcher agent (such as an assistant) or directly to the agent through a connection.
- Requests follow the model of agent behaviors.

Agent requests: general schema



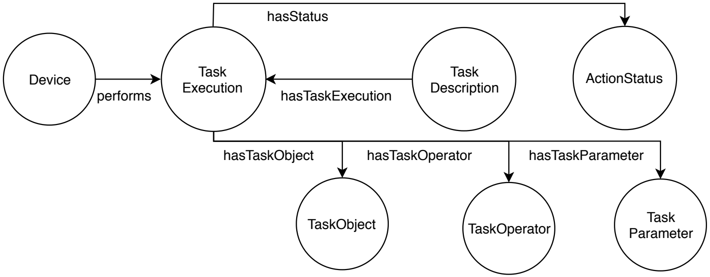
Agent requests: example



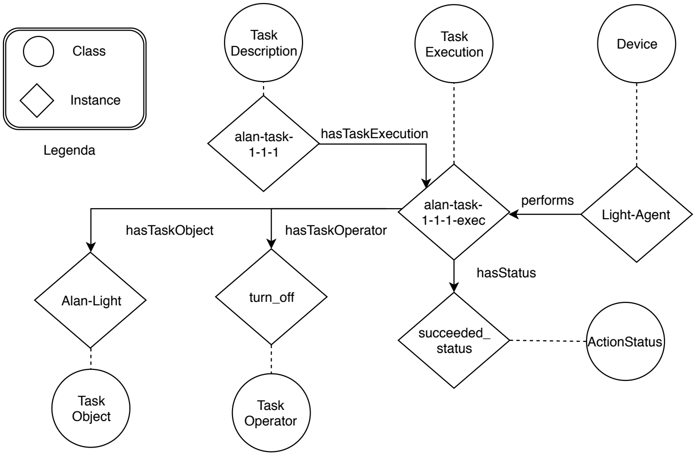
Execution of actions

- A request is associated to the attempt of executing it.
- An attempt of executing requests is represented by a task execution.
- A task execution follow the model of agent behaviors.
- The result of the attempt is transmitted to the agent which decides how to operate depending on the result.

Executing tasks: general schema



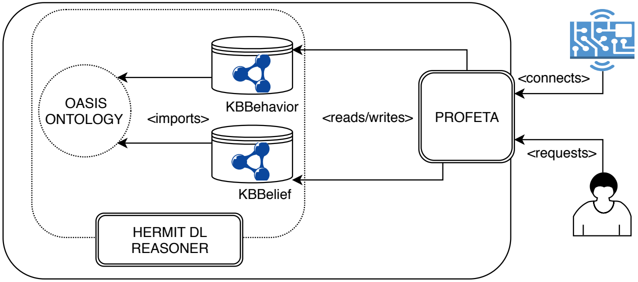
Executing tasks: example



A domotic framework: Prof-onto

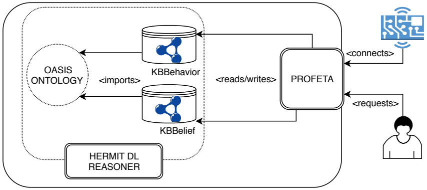
- Prof-Onto is a domotic framework based on OASIS.
- Prof-Onto implements a prototype of domotic assistant that:
  - accepts user requests.
  - decides which agent performs the requested action.
- User utterances are parsed in order to obtain OASIS agent requests (work in progress) by exploiting the NLP tool Spacy (https://spacy.io) and a BDI rule system called Profeta (Longo et al. 2017).

Prof-Onto: architecture



Prof-Onto: architecture

- The ontological core of Prof-Onto is based on OASIS and developed in JAVA.
- Prof-Onto maintains two main knowledge bases, one for the behaviors and one for the believes.
- Hermit reasoner is used for consistency checking and for inferencing.
- Profeta (now evolved in Phidias) is a BDI rule-based system developed in Python, used to manage communication and to engage agents.
- The entire communication protocol in based on sending RDF fragments.
- User utterances requires two requests. The first one is a request of mapping the utterance into an OASIS agent request; the second one is the actual request.



Prof-Onto: architecture

Phase I:

- A device sends a request of installation and transmits its behavior.
- The framework selects the agent that performs the installation (i.e., the assistant).
- The assistant installs the device by adding it into the KBBehavior knowledge base.
- Analogously, users add themselves to the environment.

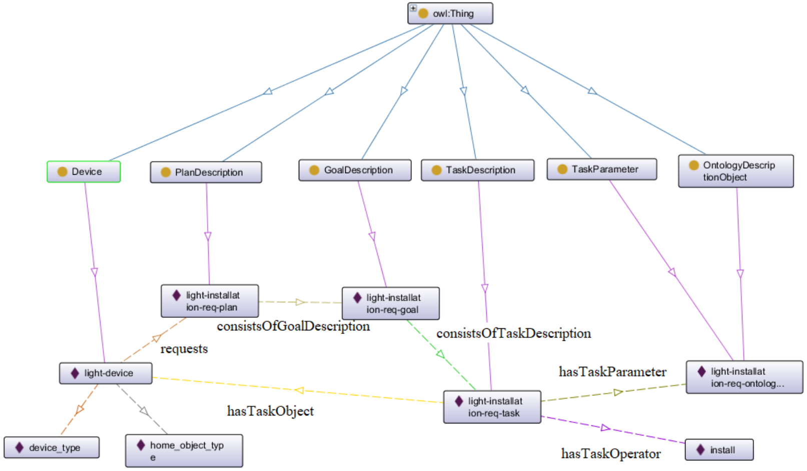
Phase II:

- Users configure devices as in Phase I.

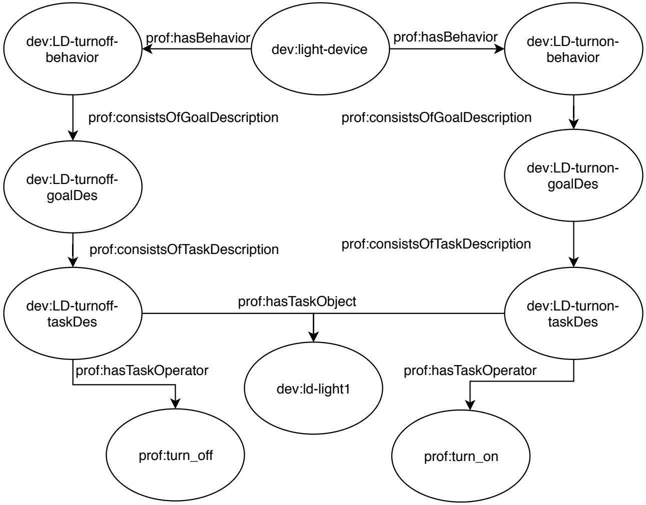
Phase III

- Users send requests to the framework which selects the device.
- The device performs the request and updates the execution status.

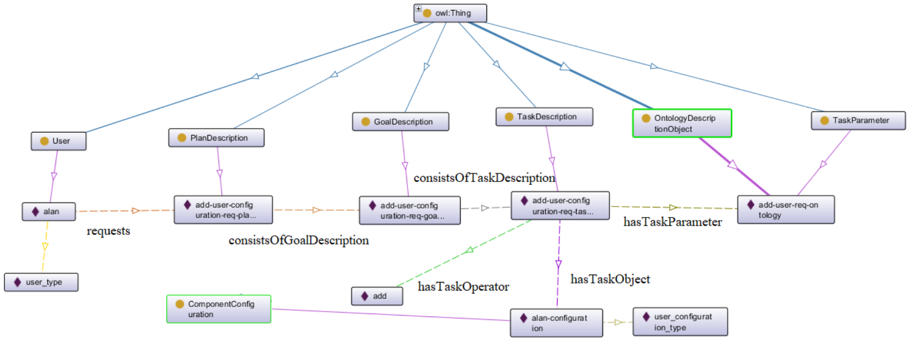
Phase I: requesting installation, example


owl:Thing
Device
PlanDescription
GoalDescription
TaskDescription
TaskParameter
OntologyDescriptionObject
light-installation-req-plan
light-installation-req-goal
consistsOfGoalDescription
consistsOfTaskDescription
requests
light-device
hasTaskParameter
light-installation-req-ontolog...
light-installation-req-task
hasTaskObject
device_type
home_object_type
hasTaskOperator
install

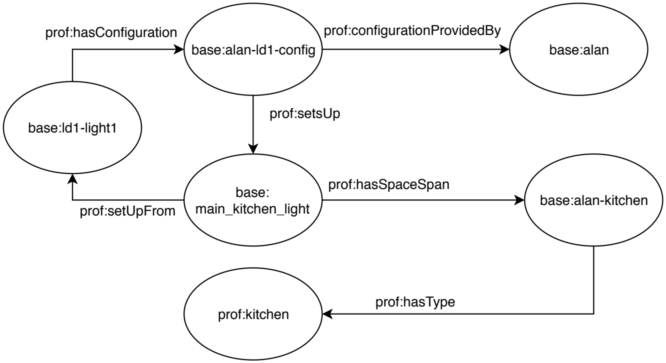
Phase I: exposing behaviors, example



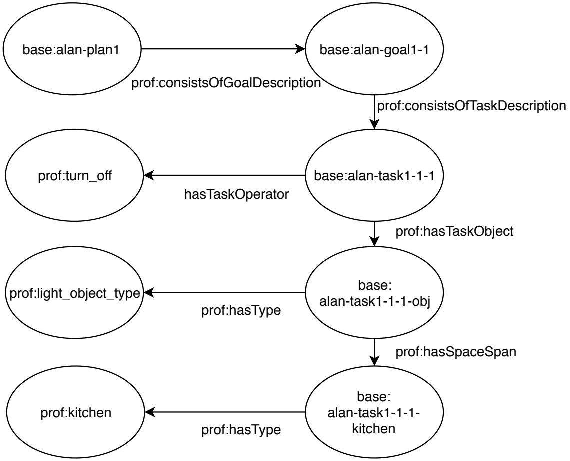
Phase II: requesting the addition of a configuration, example



Phase II: configuring component, example



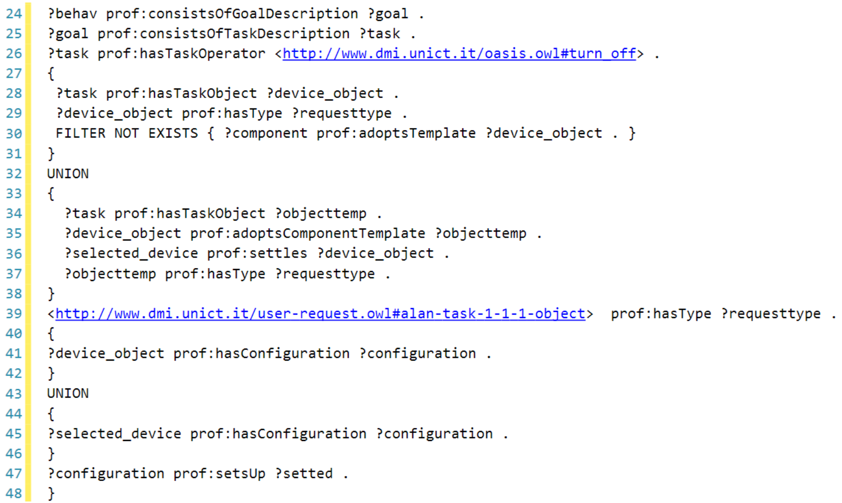
Phase III: requesting tasks, example



Example of generated SPARQL Query - Part I

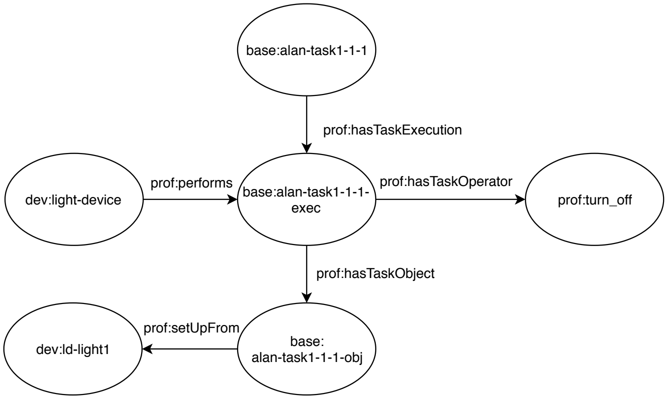
```
CONSTRUCT {
 ?selected_device prof:performs  <http://www.dmi.unict.it/user-request.owl#alan-task-1-1-1_execution> .
 <http://www.dmi.unict.it/user-request.owl#alan-task-1-1-1> prof:hasTaskExecution
                                  <http://www.dmi.unict.it/user-request.owl#alan-task-1-1-1_execution> .
 <http://www.dmi.unict.it/user-request.owl#alan-task-1-1-1_execution> rdf:type prof:TaskExecution .
 <http://www.dmi.unict.it/user-request.owl#alan-task-1-1-1_execution> prof:hasTaskObject ?device_object .
 <http://www.dmi.unict.it/user-request.owl#alan-task-1-1-1_execution> prof:hasTaskOperator
                                                         <http://www.dmi.unict.it/oasis.owl#turn_off> .
  ?device_object   prof:hasType ?requesttype .
  ?selected_device prof:hasConfiguration ?conconf .
  ?conconf prof:hasConnection ?aconnect . }
WHERE {
{
?selected_device prof:hasBehavior ?behav .
 FILTER NOT EXISTS { ?selected_device rdf:type prof:AgentBehaviorTemplate . }
}
UNION {
?selected_device prof:adoptsTemplate ?template . ?template prof:hasBehavior ?behav . }
OPTIONAL
{
  ?selected_device prof:hasConfiguration ?conconf .
  ?conconf prof:hasConnection ?aconnect .
}
```

Example of generated SPARQL Query - Part II



```
?behav prof:consistsOfGoalDescription ?goal .
?goal prof:consistsOfTaskDescription ?task .
?task prof:hasTaskOperator <http://www.dmi.unict.it/oasis.owl#turn_off> .
{
 ?task prof:hasTaskObject ?device_object .
 ?device_object prof:hasType ?requesttype .
 FILTER NOT EXISTS { ?component prof:adoptsTemplate ?device_object . }
}
UNION
{
  ?task prof:hasTaskObject ?objecttemp .
  ?device_object prof:adoptsComponentTemplate ?objecttemp .
  ?selected_device prof:settles ?device_object .
  ?objecttemp prof:hasType ?requesttype .
}
<http://www.dmi.unict.it/user-request.owl#alan-task-1-1-1-object>  prof:hasType ?requesttype .
{
?device_object prof:hasConfiguration ?configuration .
}
UNION
{
?selected_device prof:hasConfiguration ?configuration .
}
?configuration prof:setsUp ?setted .
}
```

Phase III: executing tasks, example



Future Work

OASIS

- Systems and services.
- Mapping sensors from the *Sensor Ontology* to OASIS.
- Applications of OASIS to OntologyBeanGenerator (Briola et al., 2018), JADE (Bellifemine et al., 2007), and CArtAgO (Ricci et al., 2009).
- Set-theoretic representation of OASIS and reasoning (Cantone et al., 2015).

Prof-Onto

- Integration of smart contracts.
- Integration of conditionals.
- Managing human utterances.
- Hardware experimentation.

THANK YOU