DogOnt - Ontology Modeling for Intelligent Domotic Environments

Dario Bonino, Fulvio Corno

Politecnico di Torino
Dipartimento di Automatica ed Informatica
Torino - Italy

October 28, 2008

Outline

DOMOTICS

DOMus inf**O**rma**TICS**: Information technology in the home (domus is Latin for home).

- Remote lighting and appliance control have been used for years (see X10),
- Nowadays domotics is another term for the digital home, including: the networks and devices that add comfort and convenience as well as security;
- Controlling heating, air conditioning, food preparation, TVs, stereos, lights, appliances, entrance gates and security system all fall under the domotics umbrella.

Domotic Systems - drawbacks (1/2)

Many vendors on the market, each with separate, not compatible, solutions

- Different technologies (bus, powerline, wireless)
- Different protocols (KNX, MyOpen, X10, LonWorks)
- Different device features
- Different sophistication of device firmware (from simple relay to full software-based operation)

Domotic Systems - drawbacks (2/2)

Nowadays Domotic Systems are rooted on simple electric automation

- Only simple automation is supported
  - Simple scenarios
  - Fixed, programmed behaviors
  - Simple comfort, security and energy saving policies
- No support for more complex interactions
  - Adaptation to user preferences
  - Context detection
  - Structural verification
  - Static and dynamic reasoning on the house state

Starting considerations

- The sparseness of domotics solutions, the differences in languages, communication means and protocols is very similar to the “old web”
- Semantic Web technologies can help solving
  - Interoperation issues
  - Integration of different technologies
- and can support home intelligence through
  - Reasoning
  - Context Modeling
  - ...

Outline

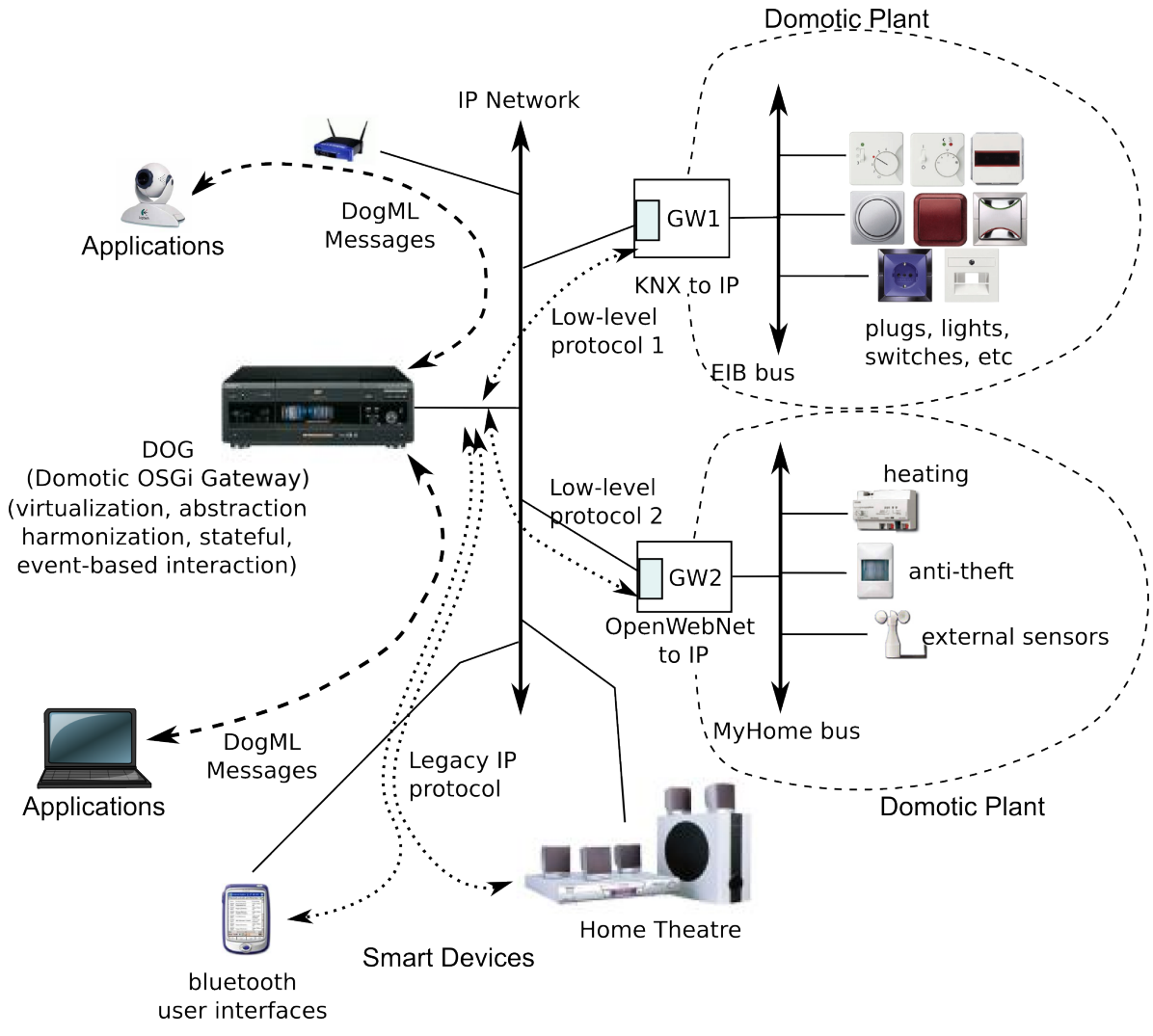
Objectives

Evolving Domotic Systems into Intelligent Domotic Environments (IDEs) supporting Interoperation, Integration and Intelligence.

- By
  - Adding a single (cheap) device for
    - interoperating different domotic plants
    - implementing complex behaviors
  - **Modeling environments in a semantic-rich, technology independent way**
  - Providing suitable querying and reasoning mechanism over the environment model

Outline

Anatomy of an IDE



DOG

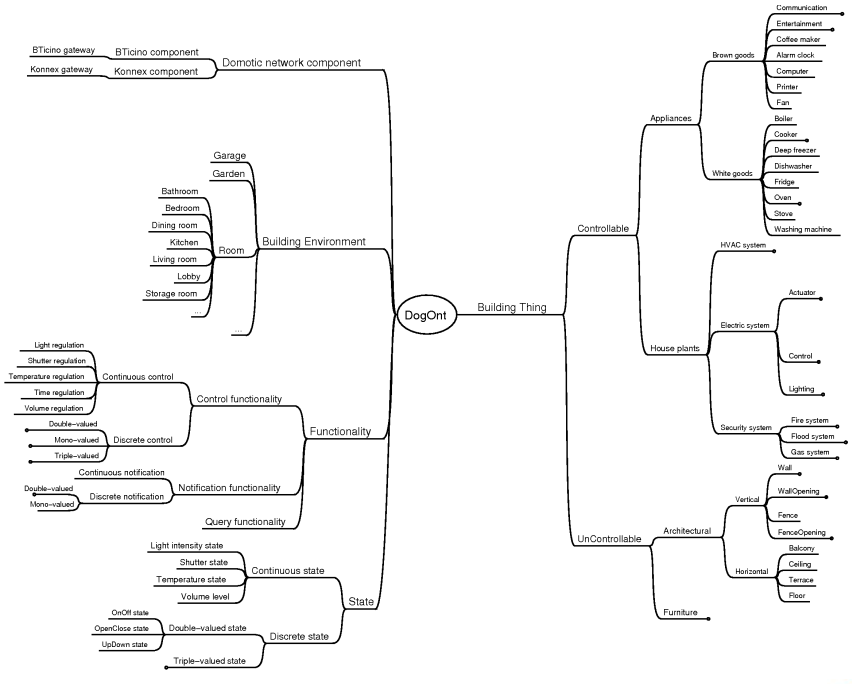
- DOG is a Domotic House Gateway [ICTAI2008] designed for transforming commercial Domotic Systems into Intelligent Domotic Environments.
- It provides
  - Interoperation between different domotic networks through proper drivers;
  - technology independent, ontology-based, house and device modeling
  - advanced, inter-network, rule-based scenario definition and operation
  - reasoning on the house model
- **DogOnt** is the ontology model lying at the bases of DOG

Outline

DogOnt

DogOnt is an ontology model designed for supporting Interoperation, Integration and Intelligence in domotic environments

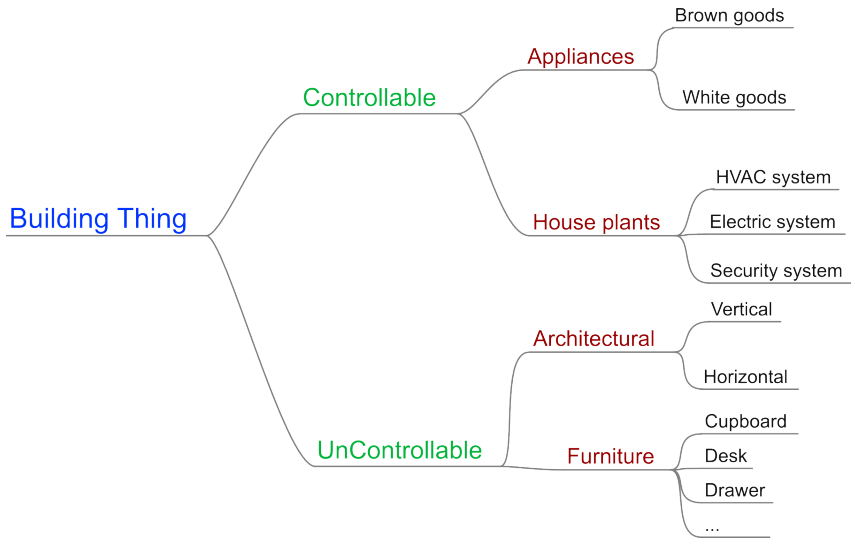
- Building Thing
- Building Environment
- State
- Functionality
- Domotic Network Component



Environment Modeling (1/2)

BuildingThing

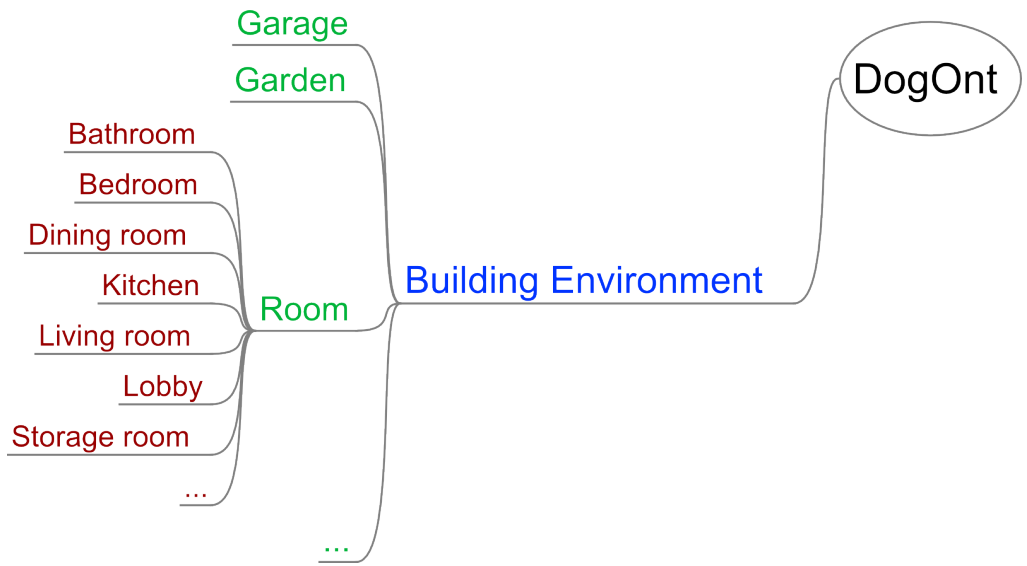
- Models all the elements of a Building Environment divided into
  - Controllable
  - UnControllable
- The UnControllable sub-tree allows to model
  - Furniture elements
  - Walls, floors, ceilings and other architectural elements (Architectural sub-tree)



Environment Modeling (2/2)

BuildingEnvironment

- Models rooms and architectural spaces composing a house
  - Rooms
  - External spaces such as garages, garden, etc.



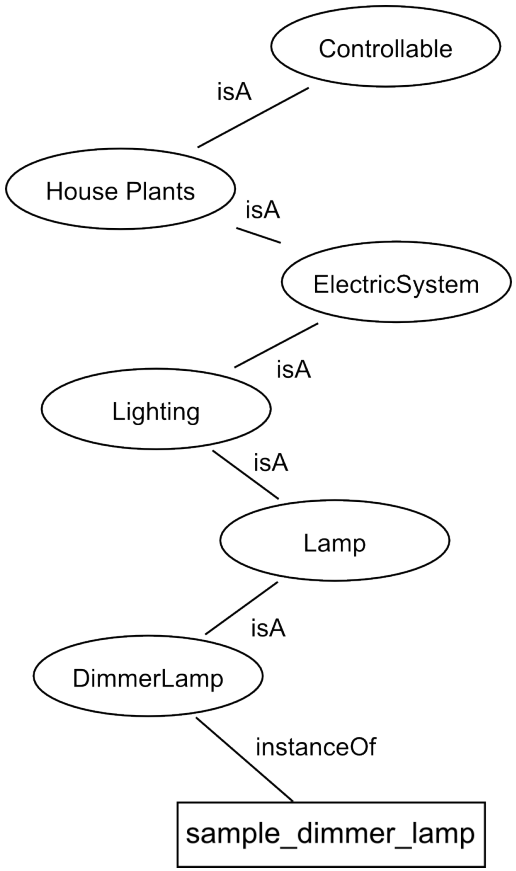
Device Modeling

- Devices are modeled independently from specific technologies
- 3 Modeling axes:
  - **Typology** - describes the type of device, separating appliances and devices belonging to house plants
  - **Functionality** - describes the tasks that a device can accomplish, by defining the available commands
  - **State** - describes the conditions in which a device can be (e.g. a Lamp can be ON or OFF)
- Technology specific aspects are modeled through separate classes
  - **NetworkComponent** - the root concept for modeling every network specific information, its sub-classes reflect the different networks supported by DOG.

Typology

Controllable devices taxonomy

- Appliances
  - Brown Goods (TV, HiFi,...)
  - White Goods (Fridge, Dishwasher,...)
- HousePlants
  - Electric
  - HVAC (Heating Ventilation & Air Conditioning)
  - Security



Functionalities (1/3)

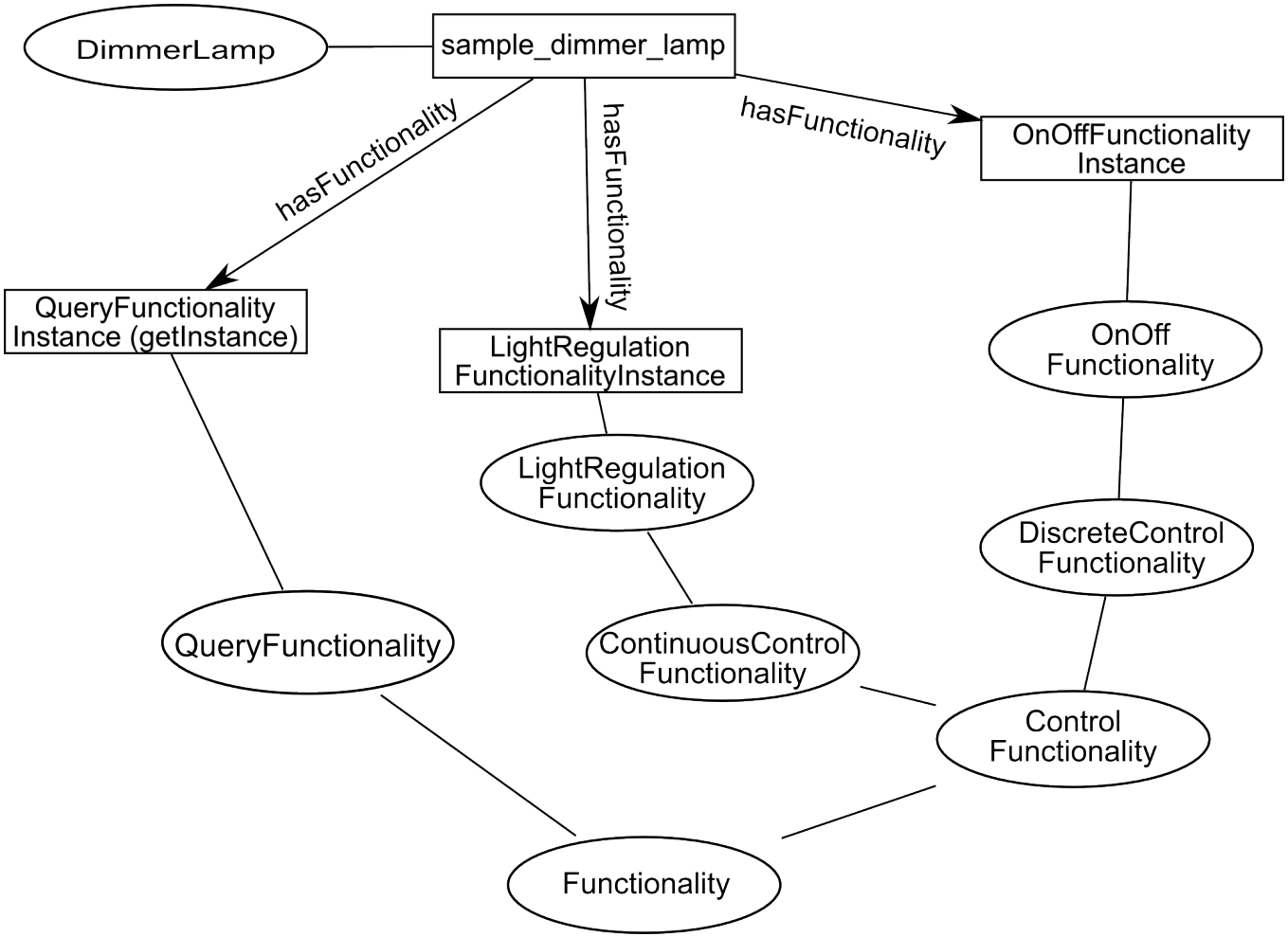
- Control Functionalities
  - Model the ability of a device to be controlled
  - Define the possible commands and their range (needed for continuous functionalities)
  - Almost every Controllable has a control functionality
- Notification Functionalities
  - Model the ability of a device to issue a notification about state/configuration changes
  - Define the possible notifications
  - Typical of Sensors and Buttons/Switches
- Query Functionalities
  - Model the ability of a device to be queried about its state/configuration
  - It's defined **for all** Controllables

Functionalities (2/3)

Every Functionality class is subdivided into

- Continuous Functionalities
  - Model the ability to change device properties in a continuous manner (e.g. dimming the light emitted by a lamp)
- Discrete Functionalities
  - Model the ability to abruptly change device properties (e.g. switching a lamp On)

Functionalities (3/3)

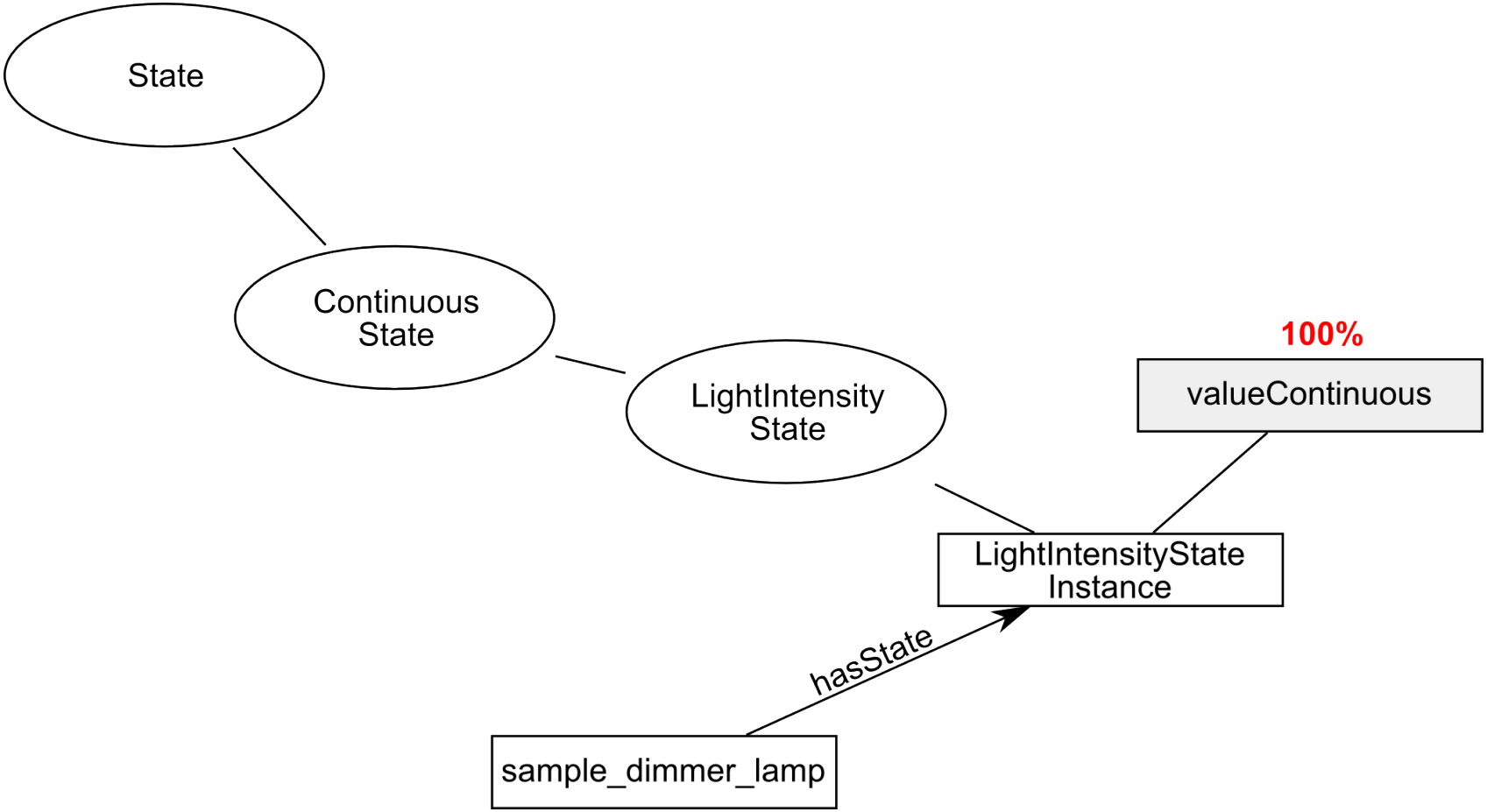


States (1/2)

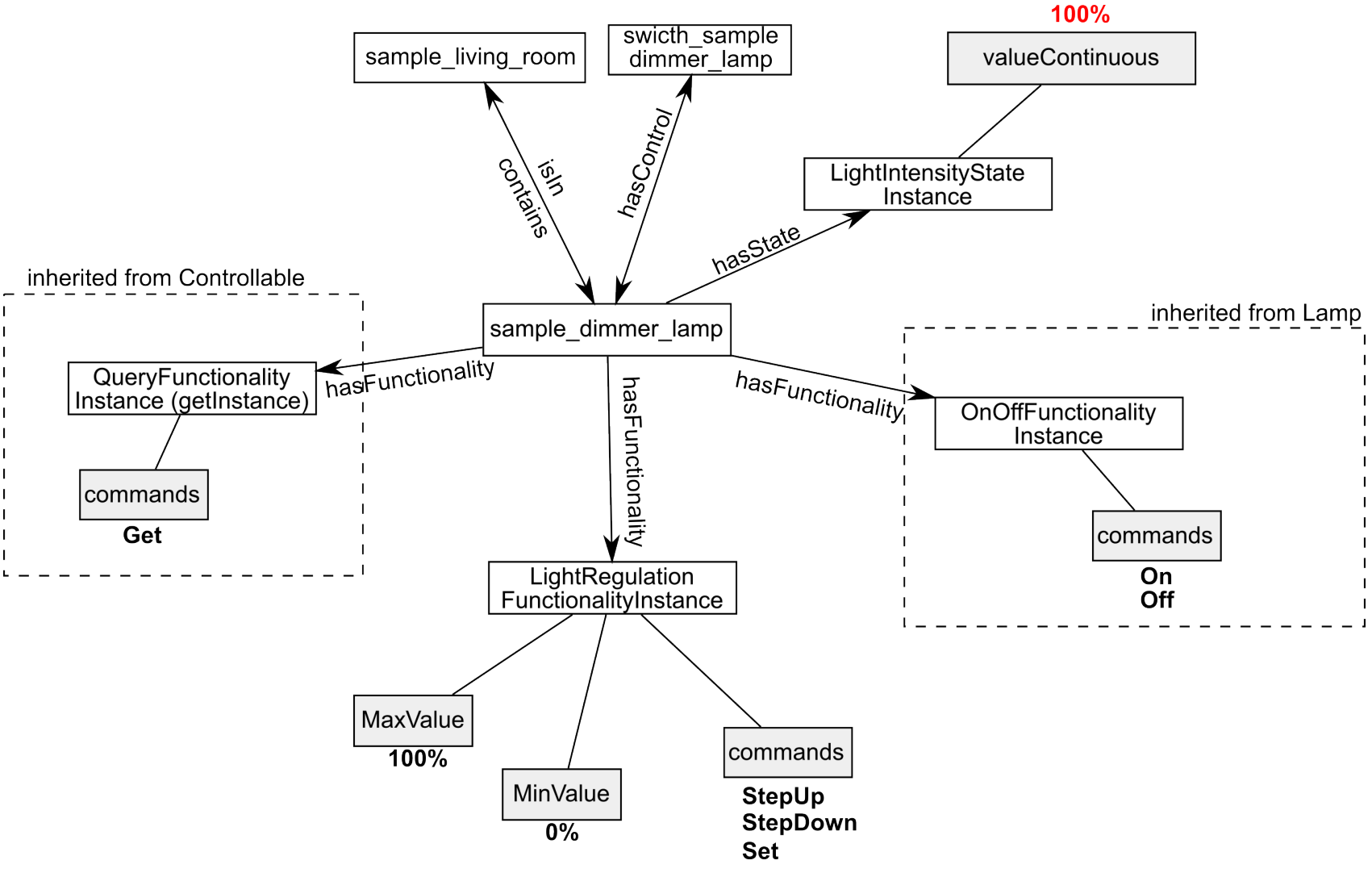
States are classified according to the kind of values they can assume

- Continuous states
  - Model continuously changing qualities (e.g. the current dimming level of a lamp)
  - The current state value is stored in the *continuousValue* property.
- Discrete states
  - Model discretely changing qualities (e.g. the lamp being On or Off)
  - The current state value is stored in the *discreteValue* property.
  - Possible states are listed in the *possibleStates* property.

States (2/2)


State
Continuous State
LightIntensity State
100%
valueContinuous
LightIntensityState Instance
hasState
sample_dimmer_lamp

DimmerLamp modeling example



Outline

DogOnt applications in IDEs (1/2)

DogOnt supports several critical features of IDEs

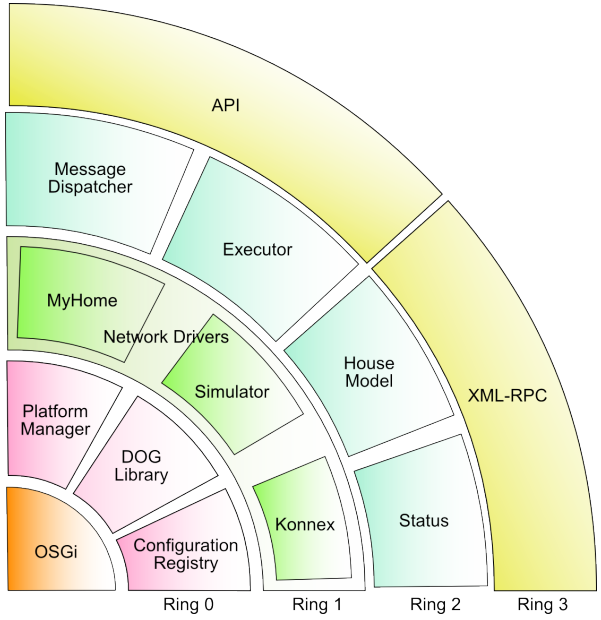
- Device Modeling
  - Allows to define a **central point of configuration** for real devices
  - Abstracts from network-specific issues, exposing systems and objects as a **uniform set of devices, states and functionalities**
  - Enables **syntactic and semantic check of commands** received from external applications/devices

DogOnt applications in IDEs (2/2)

- Device Modeling (continued...)
  - Transitive closure and Classification Reasoning allow to **decouple evolution of the model and domotic systems developments**
  - Supports the definition of top-down **inter-plant** scenarios (e.g. scenarios activated by external applications which involve devices in more than one plant)
  - Provides the basis for interoperation between plants (e.g. allowing a BTicino button to control a KNX light)
    - Frequent issue in Hospitals, Universities, Factories
    - **On-going work on automatic generation of interoperation rules from DogOnt (DogOnt v2.0)**

Example application - DOG

`http://domoticdog.sourceforge.net`



Outline

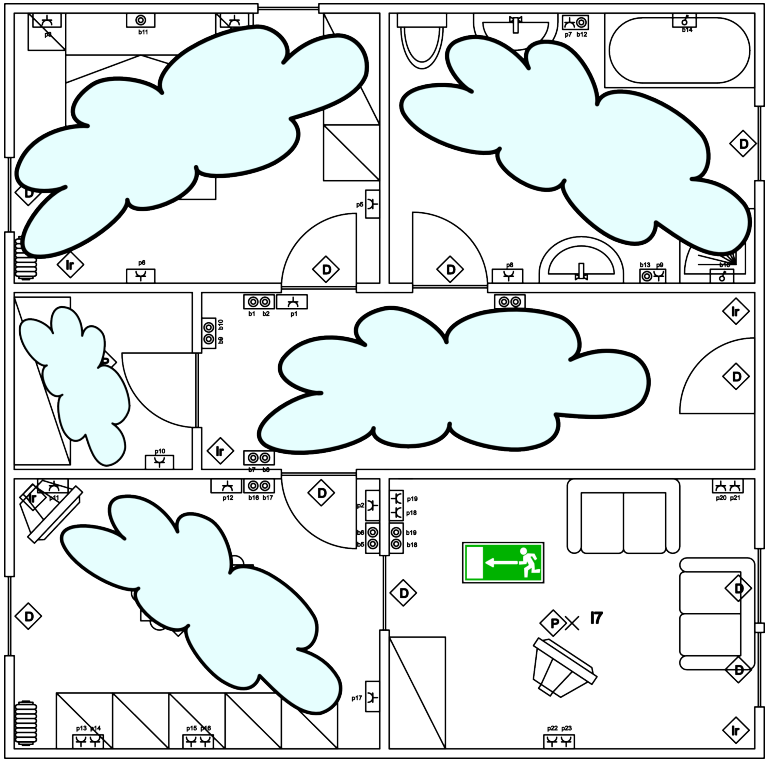
Conclusions

- We defined DogOnt: an ontology for describing domotic environments with a particular focus on objects, states and functionalities
- DogOnt is currently used by the DOG home gateway, allowing to control several, different, domotic plants, at the same time
- Standard reasoning methods allow to decouple the evolution of DogOnt and of the modeled environments
- DogOnt can be used as a basis on which building more complex/intelligent behaviors for commercial domotic systems

On-going work

We are currently working on:

- Structural verification of domotic environments through the evaluation of SWRL constraints on DogOnt model instances
- Dynamic detection/prediction of safety critical situations (smoke propagation, safe exit detection in case of fire) using rule-based reasoning and DogOnt



Questions?

Thank you!
Any Question?

Dario Bonino
dario.bonino@polito.it