Abstracting Interactions with IoT Devices
Towards a Semantic Vision of Smart Spaces

Roberto Yus, Georgios Bouloukakis, Sharad Mehrotra, Nalini Venkatasubramanian
University of California, Irvine
ACM BuildSys, 2019
IoT Application Development

- Constrained to specific devices/protocols
- Difficult to port to other IoT spaces
- Developer needs to understand the devices in the IoT space which makes development challenging

People’s world

Device’s world
App request:

➢ “Decrease temperature of rooms with occupancy above 50% of their capacity.”

User/Space policy:

➢ “Do not capture the location of John and Mary when they are in their offices.”
Challenge: Semantic Gap

App request:
➢ “Decrease temperature of rooms with occupancy above 50% of their capacity.”

User/Space policy:
➢ “Do not capture the location of John and Mary when they are in their offices.”

Which sensors/actuators can we use to answer such request/policy?
Challenge: IoT Heterogeneity

Dozens of devices in the market!

https://www.postscapes.com/iot-thermostats/

Different interaction paradigms and communication protocols

SEMANTIC GAP

Device’s world

https://iotbyhvm.ooo/what-is-coap-protocol/
App request:

- “Decrease temperature of rooms with occupancy above 50% of their capacity.”

1) Translate people’s world request into device’s world request
2) Communicate with specific devices using their protocols
Architecture

Extensible metamodel to define IoT smart spaces
Modeling IoT Spaces

- Defining IoT spaces using an ontology provides **flexibility** and **extensibility**.
  - In addition, **semantic reasoning** to infer non-explicitly defined information (e.g., if occupancy is a property of rooms, it should be also of meeting room 2065).
- Created **OWL meta ontology (semic)** extending the popular sensor ontology (SSN/SOSA)
  - Focus on representing the connection between “people’s world” and “device’s world”.
    - Properties of people/spaces (e.g., location, occupancy, temperature) connected to sensors/actuators based on expected value types and produced value types.
Architecture

Based on domain model applications, pose actions (i.e., requests, commands, or policies)
Defining User Actions

- **User Actions (UA)**, expressed at the semantic-level:
  - Requests for data (UR)
  - Commands (UC)
  - Policies (UP)

- Language for definition of general UAs with following elements:
  - **Entities of interest (E)** → Set of entities $e_i$, either entity classes $\langle e_i,rdfs:subClassOf,semic:Entity \rangle$ or entity instances $\langle e_i,rdf:type,semic:Entity \rangle$.
  - **Properties of interest (P)** → Set of properties $p_i$ $\langle p_i, rdf:type,semic:Property \rangle$.
  - **Conditions (C)** → expression containing properties that has to be satisfied to perform the actions on the entities.
  - *(For UP)* **Interaction to control** (i.e., capture, store, share) and **preferred action** (i.e., accept or deny).

<table>
<thead>
<tr>
<th>UA</th>
<th>Action</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>UR</td>
<td>“retrieve the current location of John and Mary”</td>
<td>$\langle \langle \text{Mary, John}, \text{Location} \rangle \rangle$</td>
</tr>
<tr>
<td>UC</td>
<td>“decrease temp. of rooms with occ. above 50% of their capacity”</td>
<td>$\langle \langle \text{Room, ControlTemp, Occupancy} &gt; 0.5 \times \text{Capacity} \rangle \rangle$</td>
</tr>
<tr>
<td>UP</td>
<td>“do not capture Mary’s and John’s location in private spaces when the occupancy is less than 2 people”</td>
<td>$\langle \langle \text{Mary, John}, \text{Location} \text{subClassOf PrivateSpace, Location.Occupancy} &lt; 2, \text{capture, deny} \rangle \rangle$</td>
</tr>
</tbody>
</table>
Architecture

User Actions get translated into Device-level Actions
Translating User Actions

- **Goal:**
  - Create a plan involving IoT devices to process a UA.
- **Ontology-based translation algorithm that can process policies as well as requests/commands defined at a higher-level.**

**User Action Translation**

1) Flattening
2) Plan Generation
3) Realizability Checking
4) Feasibility Checking
5) Plan Selection

Plans can be infeasible if sensors are not available (e.g., due to privacy policies)

Selection based on metrics (e.g., economical cost, latency, reliability)
Device-Actions get implemented on sensors/actuators based on their features (e.g., communication protocol).
Software components pre-built. To develop wrapper for specific device, developer just includes information about: underlying protocol, parameter, data conversion.
Using SemIoTic

Web application to show occupancy related information of the smart space

Wrappers for different sensors (e.g., Raspberry PI camera, SkySpark HVAC)
Using SemIoTic

SemIoTic (Smart Building)

SemIoTic (Smart Home)

Same application and same request but different underlying sensors used by SemIoTic

Reduction of development effort (in terms of LoC) by 55% to 97%
Thanks!