SMW for Programming/Teaching Context-Aware AAL Agents

Nicole Merkle (FZI)
29.09.2016
13th SMWCon Frankfurt am Main, Germany
Retrospective—What I talked about the last time?

At SMWCon Fall 2015, Barcelona, Spain:

- Ambient Assisted Living (AAL)
- Web of Things (WoT) of approach.
- Modelling of AAL domain by SMW.
- Sherlock Engine as assistive agent.

Today:

- SMW as programming tool for assistive, context-aware agents, e.g. Sherlock.
Why is this relevant?
Lost in the space of many APIs, Protocols and Programming Languages

Oh my God, so many different APIs.

FS20
EnOcean
KNX
ZWave
Java
eQ3

Developer
Things of Interest

IoT Devices

Agents

Thing Descriptions

URI: http://ont.Actor1
Action: http://ont.SwitchOff
Action: http://ont.SwitchOn
Event: http://ont.LampOn
Event: http://ont.LampOff
Property: http://ont.SwitchActor
Property: „white“

Machine-understandable

Web-Protocols
Rule-Based Sherlock-Agent as User Assistant

I am Sherlock, Sherlock Holmes and I deduce what you intend to do.
A Vision: SMW Programmed Sherlock Agents Everywhere

The User wants to hear music.

The User wants to drink coffee.

The User wants to visit a friend.

I implemented a new Sherlock agent.

AAL environment

AAL Service Provider

Developer

Semantic MediaWiki
I implement a new Sherlock agent.

Light Service Notation

- [[registerFor::EyeTrackingGlass]]
- [[registerFor::Lamp]]
- [[hasSWRLRule::Rule1]]
- [[hasSparqlQuery::Query1]]
- [[hasAction::SwitchLightOn]]

SWRL API
OWL API
SPIN API
WoT API
Rule-based languages

- **Semantic Web Rule Language (abbr. SWRL):**
  - First order logic. Conjunctions of Terms.
  - No Negation.
  - Structure: Premise -> conclusion
  - Terms as Tuples: predicate(?subject, ?object) or Class(?instance)
  - Expressing Triples: subject, predicate, object

- **SPARQL Protocol And RDF Query Language:** Querying RDF(S) graphs
  - SELECT, CONSTRUCT, ASK queries
  - Filter Rules by triples

- **SPARQL Inferencing Notation (abbr. SPIN):** Save SPARQL queries in RDF(S) representation

Idea: Embed SWRL rules SPARQL queries and SPIN in SMW pages
A device of type lamp has the state off and an Ambient light sensor has a state lower than 600 lux. The Assisted Person wears an Eye Tracking Glass and this Eye Tracking Glass has a thing of type lamp in focus. A lamp can be turned on and off.

Conclusion the assisted person wants to turn the lamp on.
SPARQL Query Example

Give me all temperature sensors which are in the Living Room and which state is greater than 25 celsius degree.

```sparql
PREFIX wiki: <http://localhost/mywiki#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?sensor
WHERE {
  ?sensor wiki:hasCelsiusDegree ?degree.
  ?sensor wiki:isInRoom wiki:Living Room.
  FILTER(?degree < 25)
}
```

Embedded queries and rules
SPIN Example

```sparql
{(SPIN
 description=
 query=
 )

 # Show just people with an age over 18 years.
 ASK WHERE {
   ?this my:age ?age .
   FILTER (?age < 18) .
 }
]
```
The Context-sensitive Sherlock Agent

Hello User.

Web of Things Server

IoT Adapter

WoT API

deviceEvent(ETG)

subscribeFor(Lamp)

subscribeFor(ETG)

deviceEvent(ETG)

triggerAction(Action1)

 getService

responseService

SPARQL Endpoint

Ontology

RDF

RDF

RDF

TRIPLE STORE

SERVICE

CREATE SERVICE

GET SERVICE

RESPONSE SERVICE

SWM

Semantic MediaWiki

Developer
Sherlock shall be able to learn

SMW provides static information/knowledge about the user, but the context of the user is dynamic and uncertain.

Conclusion: The preferences and intentions must be learned and the rules must be adjusted. Furthermore, in uncertain situations the user has to be asked.
### Machine Learning Datasets based on SMW knowledge

<table>
<thead>
<tr>
<th>hasState(?lamp, ?state)</th>
<th>hasState(?ambient, ?lux)</th>
<th>hasInFocus(?etg, ?thing)</th>
<th>triggeredAction(?user, ?action)</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>400</td>
<td>Lamp</td>
<td>SwitchOff</td>
</tr>
<tr>
<td>Off</td>
<td>600</td>
<td>Lamp</td>
<td>SwitchOn</td>
</tr>
<tr>
<td>Off</td>
<td>200</td>
<td>Fridge</td>
<td>-</td>
</tr>
<tr>
<td>On</td>
<td>100</td>
<td>Oven</td>
<td>-</td>
</tr>
<tr>
<td>On</td>
<td>700</td>
<td>Lamp</td>
<td>SwitchOff</td>
</tr>
<tr>
<td>Off</td>
<td>250</td>
<td>Lamp</td>
<td>SwitchOn</td>
</tr>
</tbody>
</table>

**Terms of the Premise**  
Evidences (e.g. Sensor States)  

**Conclusion**  
Action
Learning the Datasets by Naive Bayes Classifier

- **Bayes Theorem:**
  - \( P(\text{action1}|\text{evidence}) = \frac{(P(\text{evidence, action1}) \times P(\text{action1}))}{P(\text{evidence})} \)
  - \( P(\text{action2}|\text{evidence}) = \frac{(P(\text{evidence, action2}) \times P(\text{action2}))}{P(\text{evidence})} \)
  - **Action1 if** \( P(\text{action1}|\text{evidence}) > P(\text{action2}|\text{evidence}) \) **else Action2**

- **SWRL Rule:**
  - **Evidence -> Action**

---

**Diagram:**

- **Rule Engine**
  - **Rule**
- **Classifier**
  - **Learned Hypothesis**
  - **User Confirmation**

- Arrows:
  - **requires**
  - **improves**
Summary and Outlook

- **Embedded queries and rules** for programming
- Every Sherlock agent just needs to implement the rule APIs.
- The rules provide the **basis for learning the context** and user intention
- **Machine Learning** can be applied together with rule-based reasoning in order to **solve uncertainty**.
- The **Web of Things** approach provides further information to the agents about the IoT Things in the environment.

Outlook:
- Implementing/Re-using a SMW Bot for adjusting the given rules.