On Semantics in Onto-DIY

Yan Tang (VUB STARLab, Belgium)
ZhenZhen Zhao (Télécom SudParis, France)

The 3rd Workshop on Semantics & Decision Support (SeDeS’12)
Rome, Italy, 13/09/2012
Summary

• Introduction
  – Use Case and Design of Onto-DIY

• Semantics in Onto-DIY
  – Semantics in Ontologies
  – Semantics in Semantic Decision Tables
  – Semantics in user-centric services

• Conclusion and future work
Introduction

DIY

• Do-It-Yourself
  – 18\textsuperscript{th} century handcraft
  – Mid 1920’s (money$\rightarrow$ time)

• Why DIY?
  – Fun
  – Personal
  – Creativity
  – Social contact

Cover Time Magazine (August 2, 1954) portraying the abundance of tasks for a DIY-homeowner
Introduction
DIY in ICT

- Geeks have a playground in a virtual world
- DIY content creation, online application creation, hardware assemblage etc.
Motivation

• Too difficult to use for non-technical users
• Impossible to use personalized semantics
• Reuse ubiquitous solutions (e.g., context aware applications) is almost impossible
Onto-DIY
An Overview

- Flexible and idea inspiring ontology-based architecture
- Manage data semantics for semantic objects and services in a ubiquitous network
OntoDIY
Background Knowledge
Semantic Decision Tables

- SDT = Semantics + DT (Tang and Meersman, 2007)

<table>
<thead>
<tr>
<th>Condition</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>People move Ear</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Pressure on Crib</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Action</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Screen shows Message</td>
<td></td>
<td></td>
<td>Message1</td>
<td></td>
</tr>
<tr>
<td>iPhone rings</td>
<td></td>
<td></td>
<td></td>
<td>RingTone1</td>
</tr>
</tbody>
</table>

Semantics is expressed through annotations, commitments, definitions, instantiation
Onto-DIY
Naughty Boy Use Case
Before DIY

1. Mary
2. Mary's iPhone
3. 'You bad boy!'
4. 'My new toy!'
5. 'Naughty boy protector'

6. Do not touch the iPhone, James!

7. Nabaztag Rabbit
Onto-DIY
Naughty Boy Use Case

Mary

Mary's iPhone

You bad boy!

My new toy!

James

Do not touch the iPhone, James

Nabaztag

DIY
Onto-DIY
Community of Users for DIY

• Types of users
  – Non-technical
  – Technical
  – Semi-technical
Onto-DIY

DIY Aspects

• Starting point: needs of extending existing hardware/software compositions, e.g.,
  – new ambient objects are plugged in
    • No existing concept type in $\Omega$
    • New object instance with existing concept type
  – wants to use new concepts, not defined in $\Omega$ server
  – wants to define new rules

• Use Meaning Evolution Support Systems to create new concepts in $\Omega$
Semantics in SDT

### Condition

<table>
<thead>
<tr>
<th>People move Ear</th>
<th>Pressure on Crib</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

### Action

<table>
<thead>
<tr>
<th>Screen shows Message</th>
<th>iPhone rings</th>
<th>Message1</th>
<th>RingTone1</th>
</tr>
</thead>
</table>

### SDT Lexons

<table>
<thead>
<tr>
<th>Lexon 1</th>
<th>Lexon 2</th>
<th>Lexon 3</th>
<th>Lexon 4</th>
<th>Lexon 5</th>
<th>Lexon 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Bunny, has, is of, Ear&gt;</td>
<td>&lt;Bunny, has, is of, Name&gt;</td>
<td>&lt;Ear, is moved by, move, People&gt;</td>
<td>&lt;Crib, has, is of, Name&gt;</td>
<td>&lt;Screen, shows, is shown by, Message&gt;</td>
<td>&lt;iPhone, rings with, is rang with, RingTone&gt;</td>
</tr>
</tbody>
</table>

### SDT Commitments

<table>
<thead>
<tr>
<th>Commitment 1</th>
<th>Commitment 2</th>
<th>Commitment 3</th>
<th>Commitment 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>EACH Bunny has EXACT ONE name.</td>
<td>EACH Crib has EXACT ONE name.</td>
<td>EACH Screen shows AT LEAST ONE Message</td>
<td>Each iPhone rings with AT LEAST ONE Ring Tone.</td>
</tr>
</tbody>
</table>

### Instantiation of Decision Items

<table>
<thead>
<tr>
<th>People move Ear</th>
<th>Pressure on Crib</th>
<th>Screen shows Messages</th>
<th>iPhone rings</th>
</tr>
</thead>
<tbody>
<tr>
<td>“People” is James. “Ear” is the ear from the Bunny in the living room.</td>
<td>“Crib” is James’ crib. “Pressure on Crib – Yes” means that James is in his crib.</td>
<td>“Screen” is the smart screen in the living room.</td>
<td>“iPhone” is Mary’s iPhone. She has only one iPhone.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>iPhone</th>
<th>Ring Tone</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone2093113</td>
<td>RingTone1</td>
</tr>
<tr>
<td>iPhone2093113</td>
<td>RingTone2</td>
</tr>
<tr>
<td>iPhone2093113</td>
<td>RingTone3</td>
</tr>
<tr>
<td>iPhoneYan23</td>
<td>RingTone25</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Semantics in SDT

\[\text{IPhone} \sqsubseteq 1\text{has.Name} \cap \exists\text{has.Name} \]
\[\text{IPhone} \sqsubseteq \text{has.Event} \]
\[\text{Shake} \sqsubseteq \text{Motion} \sqsubseteq \text{Event} \]

\[\text{People} \equiv \{\text{james, mary}\} \]
\[\text{IPhone} \equiv \{\text{marysiPhone}\} \]
\[\text{SmartRabbit} \equiv \{\text{marysRabbit}\} \]

\[\text{People} \sqsubseteq \text{shake.IPhone} \]
\[\text{SmartRabbit} \sqsubseteq 1\text{has.Name} \cap \exists\text{has.Name} \]
\[\text{SmartRabbit} \sqsubseteq \exists\text{speaks.Message} \]

\[\text{shake(james, marysiPhone)} \]
\[\text{Message} \equiv \{\text{message1}\} \]
\[\text{speak(marysRabbit, message1)} \]
Semantics in User-Centric Services

• User centric service: apply user-centered design (UCD) process in designing a useful and easy-to-use service,
  – user motivation, user requirements, user behaviors, user interactions

• Propose: develop a semantic service creation assistant,
  – suggest a set of syntactically or semantically related services that can be connected to the existing service
Discover the web service of asking a smart rabbit to speak
• Onto-DIY
  – use SDRule-L and DL to formalize the semantics in Onto-DIY
  – But, it is recommended to use any kinds of conceptual modeling means, as long as they meet the request and technically/conceptually sound.
Future Work

• Create an automatic mapping between the semantics in the three semantic divisions
• User-centric service creation
  – Context-aware service creation – tracing end-users’ behaviors and help them to organize and filter information in order to provide personalized service
  – Service creation with trust – assisting end-users to control privacy policies to protect their sensitive data in a nonintrusive manner
  – Social service co-creation – allowing users to share their existing “Do-It-Yourself” solutions with others in order to “Do-It-Together”
  – Social service venue sharing – bringing a business concern to Onto-DIY in order to promote software innovation and encourage users to create their solutions and provide to the market
Questions?

• Thank you!