SmartSlog Session Scheme for Smart-M3 Applications

Aleksandr A. Lomov

Petrozavodsk State University
Department of Computer Science

12th FRUCT Conference, November 5–9, Oulu, Finland
Table of Contents

1 Smart-M3 platform: infrastructure & SDK

2 SSAP session

3 SmartSlog session

4 Conclusion
Infrastructure: RDF-based space with pub/sub

- Semantic information brokers (SIBs) maintain space content in low-level RDF triples
- Application consists of several knowledge processors (KPs) running on various devices
- Smart space access protocol (SSAP) for SIB↔KP communication; it supports subscription to RDF triples
- Smart-M3: Multidomain, Multidevice, Multivendor
## KP development tools

### Low-level (RDF triple)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whiteboard, Whiteboard-Qt</td>
<td>C/Glib, C/Dbus, C++/Qt</td>
<td>(Smart-M3)</td>
</tr>
<tr>
<td>Smart-M3 Java KPI library</td>
<td>Java</td>
<td>(University of Bologna and VTT)</td>
</tr>
<tr>
<td>M3-Python KPI (m3_kp)</td>
<td>Python</td>
<td>(Smart-M3 distribution)</td>
</tr>
<tr>
<td>C# KPI library</td>
<td>C#</td>
<td>(University of Bologna)</td>
</tr>
<tr>
<td>KPI_low</td>
<td>ANSI C</td>
<td>(VTT-Oulu)</td>
</tr>
<tr>
<td>C_KP</td>
<td>ANSI C</td>
<td>(Petrozavodsk State University)</td>
</tr>
</tbody>
</table>

### High-level (OWL objects & properties)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smart-M3 ontology to C-API generator</td>
<td>C/Glib, C/Dbus</td>
<td>(Smart-M3)</td>
</tr>
<tr>
<td>Smart-M3 ontology to Python generator</td>
<td>Python</td>
<td>(Smart-M3)</td>
</tr>
<tr>
<td>SmartSlog</td>
<td>ANSI C, C#</td>
<td>(Petrozavodsk State University)</td>
</tr>
</tbody>
</table>
SmartSlog SDK

- Library generator for **Smart Space ontology**
- Mapping OWL to code (C, C#):
  - KP uses ontology library
  - ontology abstractions in API
  - modest code
  - low-level KPI is hided (KPI_low)
- High-level communication primitives
  - *session*
  - knowledge patterns
  - subscription

KP programmer can think in abstract ontology terms!
SSAP session

- Start with the Join operation
- Work with the smart space
- Terminate session with Leave operation
SSAP session drawbacks

- Each session is works independently
- Data do not share between sessions
- There is no scheme to work and manage several sessions
- There is no mapping smart spaces to a one local space
New session scheme

- SSAP sessions work in one session
- one store for all SSAP sessions
- local store maps to several smart spaces
New session scheme implemented in the SmartSlog

- Node represents a SSAP session
- Repository is a local store for ontology entities
- KPI represents different KP interfaces
- Information about changes (caching data)
- Information about subscriptions (manage subscriptions)
SmartSlog session

SmartSlog session starts

Home smart space

Outside smart space

SSAP-session

Join

Leave

KP

Node 1

Node 2

Join

Leave

KP

Node 2

Node 1

SmartSlog session ends
Session repository

- Storing data for several SSAP sessions (Nodes)
- Mapping several smart spaces to one local space
- Transferring data between smart spaces
KPI’s support

Session loads KPIs in run-time and initializes them. Nodes get the KPI from the session and work with it through KPI Wrapper (C# version).

KPI Wrapper assembly contains:

- Simple implementation of Triple and BlankNode classes
- Some interfaces and abstract classes to simplify implementation Wrapper for KPI
# ANSI C and C# versions comparison

<table>
<thead>
<tr>
<th>ANSI C 0.41alpha</th>
<th>C# 0.17alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>One local store for all sessions.</td>
<td>One local store for each session.</td>
</tr>
<tr>
<td>It needs to switch between several sessions manually by using a special function.</td>
<td>Appropriate objects represent sessions.</td>
</tr>
<tr>
<td>Session works with one SSAP-session.</td>
<td>Session contains several SSAP-sessions.</td>
</tr>
<tr>
<td>Different low-level KPIs can not be used in the run-time.</td>
<td>Session can work with different KPIs (using Wrapper) in the run-time.</td>
</tr>
</tbody>
</table>
Session examples

**ANSI C version**

```c
sslog_init_new_session_with_parameters("X", "127.0.0.1", 10010, "Home");
sslog_init_new_session_with_parameters("X", "194.85.173.9", 10012, "Bar");
sslog_switch_sib("Home"); individual_t *timo = sslog_new_individual(CLASS_MAN);
sslog_add_property(timo, PROP_FNAME, "Timo"); ...
sslog_insert_individual(timo);
sslog_switch_sib("Bar"); sslog_add_property(timo, PROP_DRINKS, "beer");
sslog_leave_session_all(); sslog_repo_clean_all();
```

**C# version**

```csharp
Node timoNode = new Node(nodeNameTimo, smartSpaceName, address, port);
Individual timo = timoNode.CreateIndividual(OntStructure.Man); node.Insert(timo);

ISession wifeSession = SessionManager.CreateSession();
Node wifeNode = new Node(wifeSession, ...);
Individual wife = wifeNode.CreateIndividual(OntStructure.Woman);
wifeNode.Insert(wife);
SessionManager.DestroySession(wifeSession);
```
Conclusion

- New session scheme
  - several SSAP session
  - repository - one local store

- Future directions
  - Subscriptions managing
  - Property caching

- SmartSlog developers wiki:
  http://oss.fruct.org/wiki/SmartSlog/

- Open source code:
  http://sourceforge.net/projects/smartslog/

Thank you!