A portable implementation of Semantic Information Broker in OSGi technology

Alfredo D’Elia
ARCES
DISI
adelia@arcес.unibo.it
www.unibo.it

Innovation House – Otaniemi
12th November 2013
Overview

• Reference scenario
• Motivations
• Features
• Internal architecture
• Performance evaluation
• Conclusion
Reference scenario

**PHYSICAL WORLD**

- Sensors
- People and smart devices
- Appliances

**SEMANTIC AGENTS**

- Context: Raw sensor data, profiles, location, requests, geometries, preferences, goals...
- Services: health, privacy, integrated localization, personalized behaviour, proactive suggestions, social solutions, ...

**INFORMATION WORLD**

- XML
- XML Schema
- RDF Model & Syntax
- Ontology
- Rules / Query
- Logic
- Proof
- Trust
- Encryption
Reference scenario

**Physical World**

- Sensors
- People and smart devices
- Appliances

**Semantic Agents**

Knowledge processor (KP):
- Interacts in R/W mode with the information world
- Adapts the legacy technology to the software architecture to share all relevant information
- Multiplatform: libraries (KPI) and tools for the most common languages and platforms augment developers productivity

**Information World**

Semantic Information Brokers
- Information as a *semantic* graph made of RDF triples
- Notification mechanism to promptly react to context changes
Motivations

• Meet the requirements of evolving scenarios

• Portability

• Compatibility

• Flexibility

• Meet the technological expectations of dynamic communities (Semantic Web, Context awareness, etc.)
Combining M3 and OSGi

- **Smart M3:**
  - General purpose
  - Information interoperability support
  - Semantic software agents

- **OSGi**
  - Java
  - Portable
  - Modular
  - Maintainable
  - Reliable

- The OSGi SIB provides the advantages of both smart-M3 and OSGi in a single core component for semantic context aware applications.

- The OSGi SIB is more appropriate than a monolithic implementation as long as the scenario and the requirements are in rapid evolution.
Features

• All basic SIB primitives

• Completely compatible with the current version of SSAP and existing libraries

• SPARQL query and update

• SPARQL subscription

• Persistent SPARQL UPDATE (similar to a rule) \textit{new}

• DL reasoning at SROIQ level (using Pellet) \textit{experimental prototype}
Internal architecture

- **SUB**
  - **TOOLS**
    - CheckIRU
    - reduceDesc
    - remSub
    - remSubs
    - addSub
    - getNewId

- **JOIN**
  - **TOOLS**
    - IsJoined
    - Join
    - Leave

- **SIB**
  - **TOOLS**
    - JOIN
    - SUB
    - SSAP
    - TCP
  - ProcessExtendedSSAP

- **MH**
  - **TOOLS**
    - TH
    - SIB
    - SSAP
  - AddMessage

- **TH**
  - **TOOLS**
    - Service
    - Uses
    - Requires

- **SSAP**
  - **TOOLS**
    - CreateResponse
    - ParseSSAP

- **TCP**
  - **TOOLS**
    - Callback
  - **TOOLS**
    - MH
    - TH
Gate bundle: one instance for each communication protocol or technology. Adding support for e.g. Bluetooth does not affect the rest of the SIB at all.
Internal architecture

SSAP bundle: Extensions and modifications of the SSAP protocol can be supported by modifications to this bundle and nothing else in the SIB.
Internal architecture

- DL reasoning (e.g. Pellet)
- Rule framework (e.g. Jess)
- Persistent Storage (e.g. TDB, SDB)
- New API (e.g. Virtuoso, OWL API)

Store bundle: Core of the SIB, new features can be experimented by opportune modifying or configuring this bundle
Internal architecture

New bundles:
- Access control
- Encryption
- Distribution

Join

IsJoined
Join
Leave

SUB

TOOLS

CheckIRU
reduceDesc
remSub
remSubs
addSub
getNewId

TCP

JOIN

SUB

SSAP

GetToken
ReleaseToken

CreateResponse
ParseSSAP

AddMessage

TOOLS

SIB

ProcessExtendedSSAP

MH

TH

Callback

Service
Uses
Requires
Performance evaluation

• Performance of primitives supported by the internal RDF store match in optimal conditions the performance of the store itself (i.e. Jena Framework for RDF graphs)

• Interesting topic is to study SIB specific functionalities (i.e. not originally supported by the internal store) and how they affect the other primitives
Performance with respect to subscription patterns

Insert time for 10 triples

- OSGI
- M3 B
- PIGLET
Conclusion

- New SIB implementation running on most popular operating systems
- New features implemented and tested in prototype versions
- Performance comparable to the reference implementation which is more optimized, but based on a monolithic approach
- Simple adaptation to possible future requirements among which:
  - new SSAP version
  - new primitives
  - security