Smart Spaces

Chapter 4:

The Smart-M3 Platform

Outline

§1. Architectural Overview

§2. Notion of Application (multi-agent)

§3. Interfaces
(agents <–> smart space)

Research scope

- SOFIA project
  Smart Objects For Intelligent Applications
- DIEM project
  Devices and Interoperability Ecosystem
- EIT ICT Labs
  one of Knowledge and Innovation Communities (KICs) selected by the European Institute of Innovation & Technology to accelerate innovation in Europe
- FRUCT
  Open Innovations Association

http://sourceforge.net/projects/smart-m3/
BSD open source license

§1. Architectural Overview

Smart applications needs
a smart space infrastructure

Challenges from practice

- Digital convergence and interoperability
  - Many ways for communication with the external world
  - Domain specific interoperability standards, e.g., UPnP (in home entertainment)
  - Limited set of use cases
  - Lengthy and uncertain standardization process
- Ubiquitous computing – devices everywhere
  - Ideally, interoperability with whatever devices that are in the locality at any given time

Key principles

- *Giant global graph* of semantic web vs. *dynamic and local* semantic web
- Interoperability via information sharing
  - Sharing local semantic information
    e.g., about the immediate environment of a device
  - Accessing locally relevant parts of the giant global graph
  - Cross-domain interoperability due to ontology compositions
  - Standardizing an ontology allows an indefinite set of use cases to be implemented

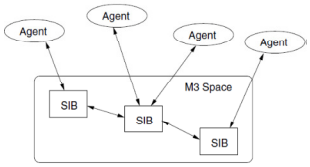
Release: two parts

Smart-M3 releases at
http://sourceforge.net/projects/smart-m3/

1. Infrastructure (**SIB side**, shared knowledge)
   Deployed implementation of smart spaces for applications
2. SDK (**KP side**, interfaces to shared knowledge)
   Development tools for various platforms and network access protocols
   - Most of them are hosted at separate repositories

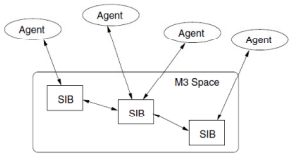
Basic Terms

- **SIB**: semantic information broker
- **KP** (M3 agent): knowledge processor
- **SSAP** (Smart Space Access Protocol)
- **M3 Space** is a named search extend of information
- **KPI:** KP-SIB interface



SIB network

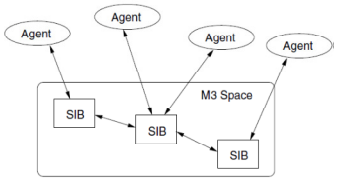
- Information stored in one or more SIBs
  - One SIB is the basic case
- Each SIB maintains an RDF store
- The global SIB network satisfies the distributed deductive closure
  - Any KP sees the same knowledge regardless the SIB it connects to



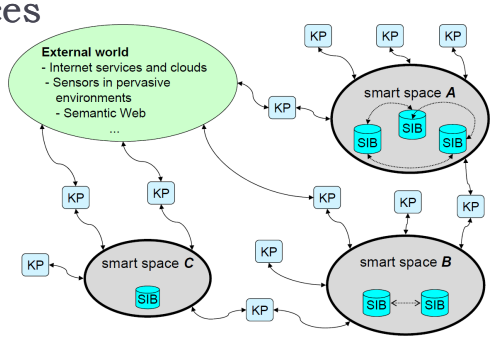
Global RDF graph of shared knowledge

Global RDF graph

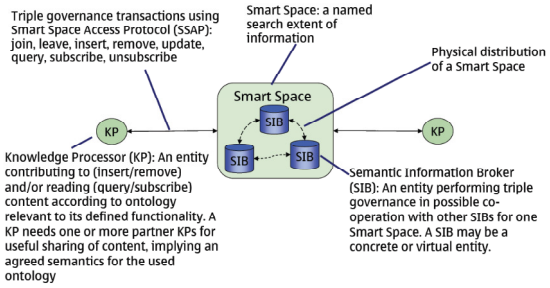
- Many subgraphs
- Many ontologies
- The use of any ontology is not maintained
- Information consistency is not guaranteed



Global view on Smart-M3 spaces

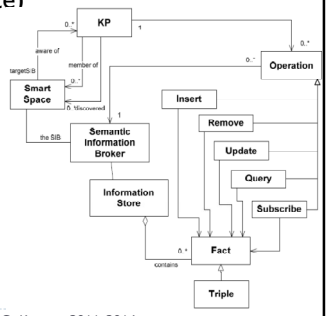


Smart-M3 Infrastructure 1



Smart-M3 Infrastructure 2

- M3 space (smart space)
- SIB
- M3 agent (KP, node)
- M3 store (knowledge store, RDF triples)
- SSAP operation



Smart-M3 Infrastructure 3

1. M: Multi-domain
2. M: Multi-device
3. M: Multi-vendor

- Many kinds of devices can interact with each other
  - mobile phone, television set, laptop, ...
- Device may be composed of parts that are considered as individual partners for interaction with another device
  - PC keyboard for typing input to a mobile phone
- Free in choosing the manufacturer
  - Nokia, Samsung, ...

Smart Space Access Protocol (SSAP): 1

- **Join**: Join a KP to a named space
- **Leave**: Leave a named space.
  After leaving, no more operations may be performed until a join operation
- **Insert**: Atomically insert a graph in the space
- **Remove**: Atomically remove a graph from the space
- **Update**: Atomically update a graph in the SIB.
  Update is a combination of remove followed by insert, executed atomically
  - A graph to remove, a graph to insert

Smart Space Access Protocol (SSAP): 2

- **Query**: Query for information in the space using any supported query language (SPARQL)
- **Subscribe**: Set up a persistent query in the space;
  a change to the query results is reported to the subscriber
- **Unsubscribe**: Cancel an existing subscription

Guarantees

- Operations are done in the same order as they were performed by the KP
- For a received operation, the SIB will process no operation received later before processing the earlier operations

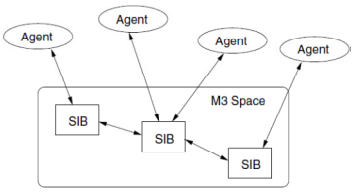
Smart Space Access Protocol (SSAP): 3

Not implemented yet

- Logic rules over RDF triple store
  - deriving new knowledge (views, concepts) from the RDF graph, like in Prolog
  - resource allocation and access
  - Synchronization and conflict resolution
- Access control mechanism based on the information content
  - Knowledge privacy
  - Tagging information with ownership and access rights
  - KP provides credentials when joining a particular named M3 space
- Test-and-set type of primitives for basic synchronization
- SIB network and a protocol of distributed deductive closure

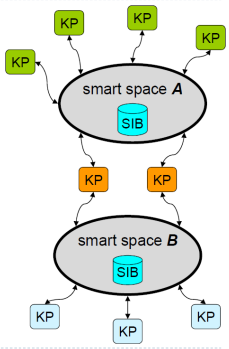
§2. Notion of Application

- Traditional application:
  - monolithic
  - single screen
  - strong coupling
- M3 application:
  - Scenario to meet user's goal
  - Scenario emerges from observable actions



Ad-hoc KP assembly with loose coupling

- Actions are from participating KPs
- Observations are from
  1. the M3 space
  2. the use of services



Transient Scenarios

- The scenario is changing as
  - KPs join and leave the M3 space
  - Services become available or unavailable
- The loose coupling between the KPs
  - KPs communicate by modifying and querying the M3 space content
  - KPs may also communicate with each other by other available means (non-Smart-M3)

KP ontology

- Each KP understands its own, non-exclusive set of information
  - RDF graph
  - KP ontology allows analyzing this graph
- Overlapping is essential for interoperability
  - KPs can see each others actions

KP mash up

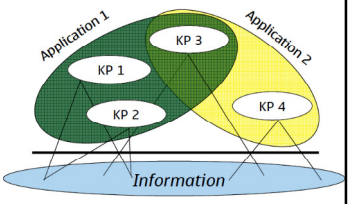
- KPs (e.g., sensors) are information providers
- KPs (e.g., clients) are information consumers
  - read the information
- KPs (e.g., reasoners) process further the information internally
  - publish the result (new knowledge)

Open problems:

- KPs compete for the same resources
  - synchronization
- KPs use different ontologies
  - Compositions, Overlaying

Combining Smart Apps

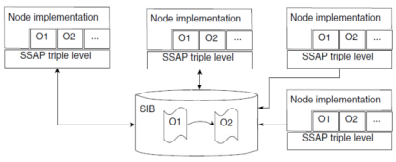
- Overlapping spaces due to overlapping ontologies
- KP intermediary
- Example
  - Smart conference
  - +
  - Smart blogging



§3. Interfaces

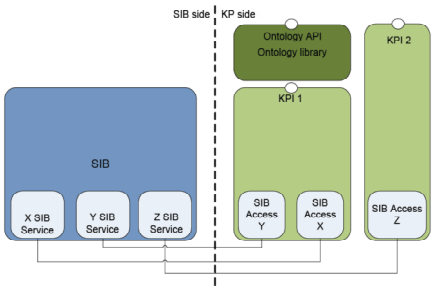
KP <-> SIB communication: KP Interface or **KPI**

1. KP can operate on the RDF triple level
   - Direct access with SSAP
   - Low-level programming
2. KP can understand the ontology behind the triples
   - RDF graph
   - Larger conceptual entities
   - Interpreting the information according to predefined ontologies



Logical architecture

- SIB: many communication mechanisms
  - TCP
  - NoTA: Network on Terminal Architecture
  - Bluetooth
- KP: selects appropriate mechanisms
- Ontology library: ontology concepts used in code



KP Development Tools

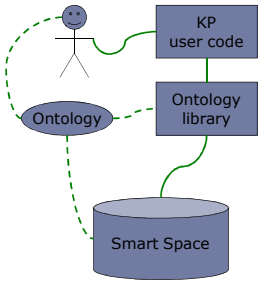
1. Low-level programming tools
   - Based on triples
   - Basic manipulations
   - RDF triple exchange
   - Mediator library for SSAP operations
2. High-level programming tools
   - Based on ontology entities
   - Advanced manipulations
   - Ontology library

KP Interface (KPI)

Library	Description
	Low-level KP programming: RDF triples
Whiteboard, Whiteboard-Qt + QML	Language: C/Glib, C/Dbus, C++/Qt. Network: TCP/IP, NoTA. BSD license. A part of the Smart-M3 distribution, http://sourceforge.net/projects/smart-m3/
KPI_Low	Language: ANSI C. Network: TCP/IP, NoTA. GPLv2. Primarily oriented to low-performance devices. VTT-Oulu Technical Research Centre (Finland), http://sourceforge.net/projects/kpilow/
Smart-M3 Java KPI library	Language: Java. Network: TCP/IP. University of Bologna (Italy) and VTT-Oulu Technical Research Centre (Finland), http://sourceforge.net/projects/smartm3-javakpi/
M3-Python KPI (m3_kp)	Language: Python. Network: TCP/IP. BSD license. A part of the Smart-M3 distribution, http://sourceforge.net/projects/smart-m3/
C# KPI library	Language: C#. Network: TCP/IP. University of Bologna (Italy).
	High-level KP programming: OWL ontology
Smart-M3 ontology to C-API generator	Language: Glib/C, Dbus/C. Network: TCP/IP, NoTA. BSD license. A part of the Smart-M3 distribution, http://sourceforge.net/projects/smart-m3/
Smart-M3 ontology to Python generator	Language: Python. Network: TCP/IP, NoTA. BSD license. A part of the Smart-M3 distribution, http://sourceforge.net/projects/smart-m3/
SmartSlog	Language: ANSI C, C#. Network: TCP/IP, NoTA. GPLv2. Petrozavodsk State University (Russia), http://sourceforge.net/projects/smartslog/

Ontology Library

- Simplifying KP code using high-level OWL terms
- Generator transforms OWL to ontology library
- KP code calls ontology library
- **SmartSlog**



Smart-M3 Value Offering

- USERS: Freedom of choice
  I want to select my device freely from any vendor knowing that it works with all devices I already have.
  – M3 = multi vendor
- DEVICE MANUFACTURERS: Seamless operation with all devices
  I want to create innovative products that consumers want to buy because they work seamlessly with other devices wherever he/she goes.
  – M3 = multi device
- SERVICES COMPANIES: Gaining competitive edge
  My company develops novel services using mash-up approach and we want seamless data portability to effortlessly create winning solutions for cross domain user experience.
  – M3 = multi domain
- APPLICATION DEVELOPERS: Focus on consumer 'wow'
  As an application developer I want to focus on creating consumer 'wow' instead of porting my code to all different platforms. I also want develop cross-domain mash-up services as easy as internet services are created today!
  – M3 = multi domain

Literature

- J.Honkola, H.Laine, R.Brown, O.Tyrkko. Smart-M3 Information Sharing Platform (ISCC 2010)
- F.Morandi, L.Roffia, A.D'Elia, F.Vergari, S.T.Cinotti. RedSib: a Smart-M3 Semantic Information Broker Implementation (FRUCT12, 2012)
- F.Wickström. Getting Started with Smart-M3 Using Python (TUCS Technical Reports 1071, 2013)
- D. Korzun, S. Balandin, V. Luukkala, P. Liuha, A. Gurtov. Overview of Smart-M3 Principles for Application Development (IS&IT 2011)
- D.Korzun, A.Lomov, P.Vanag, S.Balandin, J.Honkola. Generating Modest High-Level Ontology Libraries for Smart-M3 (UBICOMM 2010). Extended version in International Journal On Advances in Intelligent Systems (vol.4, nr3&4, 2011). Russian version in Теоретический и прикладной научно-технический журнал "Программная инженерия" (N5, 2012)