Smart Spaces

Overview:

The Smart-M3 Platform: Multi-device, Multi-vendor, Multi-domain

Outline

§1. Architectural Overview
§2. Notion of Application (multi-agent)
§3. Interfaces
(agents <-> smart space)
§4. Service Formalism

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

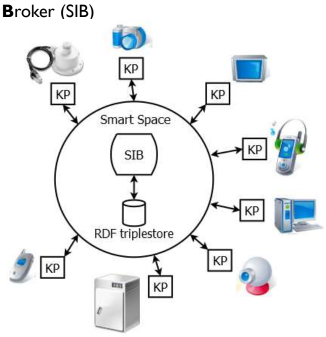
§1. Architectural Overview

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

Smart-M3 space

- **M**ultidevice, **M**ultidomain, **M**ultivendor
- Infrastructure: **S**emantic **I**nformation **B**roker (SIB) maintains smart space content in RDF triples
- Application: **K**nowledge **P**rocessors (KPs, agents) running on IoT devices
- Interaction: Blackboard and Pub/Sub
  - join, leave
  - insert, update, remove
  - (un)subscribe
- Smart space: *KPs share ad-hoc knowledge and reason over it to construct services*



Key principles

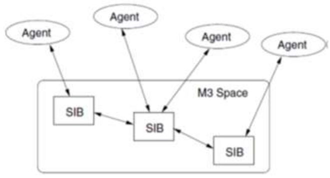
- *Giant global graph* of semantic web vs. *dynamic and local* semantic web
- Interoperability via information sharing
  - Sharing local semantic information e.g., about the immediate environment of a device
  - Accessing locally relevant parts of the giant global graph
  - Cross-domain interoperability 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

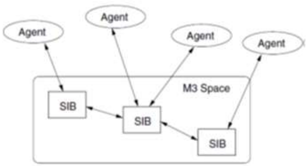
Basic Terms



- **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

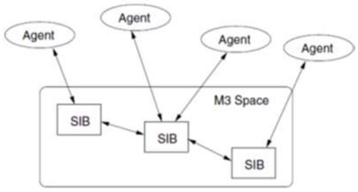
- Information stored in one or more SIBs
  - One SIB is the basic case
- Each SIB maintains an RDF store
- The global SIB network satisfies the distributed deductive closure
  - Any KP sees the same knowledge regardless the SIB it connects to



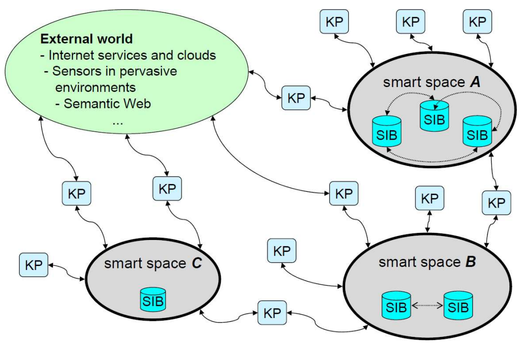
Global RDF graph of shared knowledge

Global RDF graph

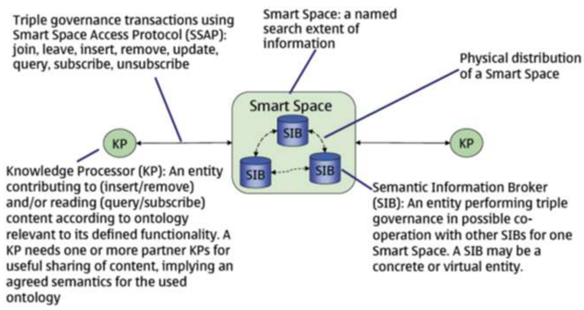
- Many subgraphs
- Many ontologies
- The use of any ontology is not maintained
- Information consistency is not guaranteed



Global view on Smart-M3 spaces

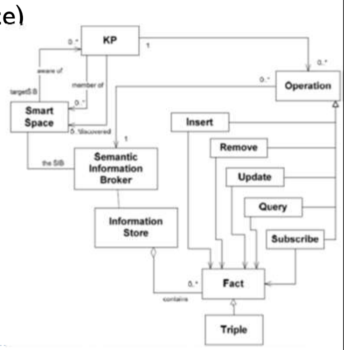


Smart-M3 Infrastructure 1



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
  - Intel, Samsung, Apple, ...

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

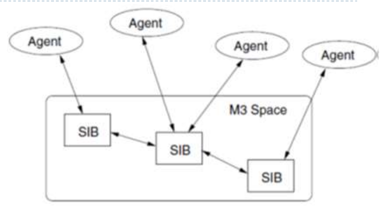
Smart Space Access Protocol (SSAP): 3

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

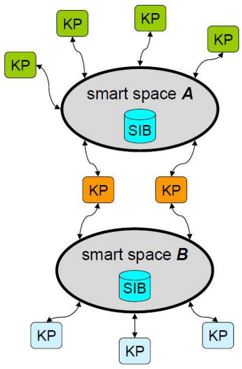
- Traditional application:
  - monolithic
  - single screen
  - strong coupling



- 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

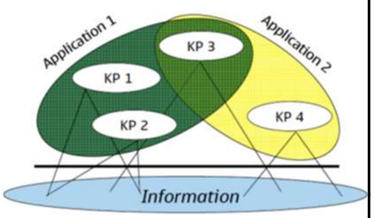
- KPs (e.g., sensors) are information providers
- KPs (e.g., clients) are information consumers
  - read the information
- KPs (e.g., reasoners) process further the information internally
  - publish the result (new knowledge)

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



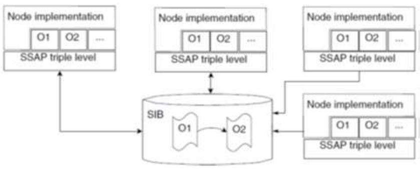
Examples of Applications

- See Ch.5

§3. Interfaces

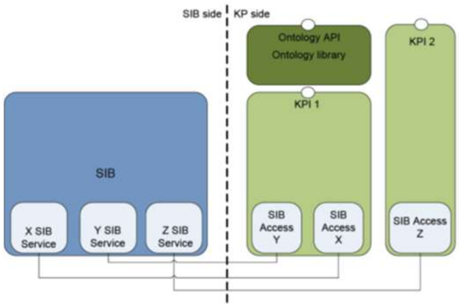
KP <-> SIB communication: KP Interface or **KPI**

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)

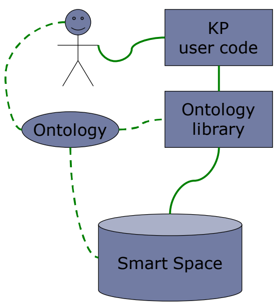
Название KPI	Автор	Репозиторий
M3-Python KPI	Исследовательский центр Nokia (Хельсинки, Финляндия)	http://sourceforge.net/projects/smart-m3/
KPI_Low	Научно-технический Центр Финляндии VTT (Оулу, Финляндия)	http://sourceforge.net/projects/kpilow/
C_KPI вариант развития KPI_Low	Петрозаводский государственный университет (Петрозаводск, Россия).	http://sourceforge.net/projects/smartslog/
Smart-M3 Java KPI library	Болонский университет (Болонья, Италия) и Научно-технический Центр Финляндии VTT (Оулу, Финляндия)	http://sourceforge.net/projects/smartm3-javakpi/
Smart-M3 PHP KPI library	Болонский университет (Болонья, Италия)	http://sourceforge.net/projects/sm3-php-kpi-lib/

KP Interface (KPI)

Название KPI	Автор	Репозиторий
C# KPI for Smart-M3	Болонский университет (Болонья, Италия)	http://sourceforge.net/projects/m3-csharp-kpi/
WP C# KPI for Windows Phone (модификация C# KPI)	Петрозаводский государственный университет (Петрозаводск, Россия)	http://sourceforge.net/projects/smartslog/
SmartSlog (Smart Space onтoLOGy)	Петрозаводский государственный университет (Петрозаводск, Россия)	http://sourceforge.net/projects/smartslog/
SOFIA Application Development Kit	Научно-технический Центр Финляндии VTT (Оулу, Финляндия)	http://code.google.com/p/sofia-application-development-kit/
SMOOL (Smart Tool to create Smart Spaces)	Технологическая корпорация Tecnalia (Испания)	https://code.google.com/p/smool/

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

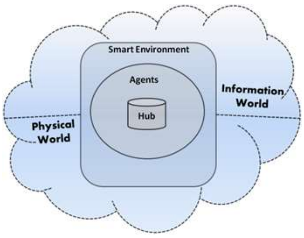
- USERS: Freedom of choice
  I want to select my device freely from any vendor knowing that it works with all devices I already have.
  – M3 = multi vendor
- DEVICE MANUFACTURERS: Seamless operation with all devices
  I want to create innovative products that consumers want to buy because they work seamlessly 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)
- 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).
- E. Ovaska, T. S. Cinotti, and A. Toninelli. The design principles and practices of interoperable smart spaces. in Advanced Design Approaches to Emerging Software Systems: Principles, Methodology and Tools. IGI Global, 2012, pp. 18–47.
- D. Korzun, S. Balandin, A. Gurtov. Deployment of Smart Spaces in Internet of Things: Overview of the design challenges. ruSMART 2013
- А.М.Кашевник, Д.Ж.Корзун, С.И.Баландин. Разработка интеллектуальных систем на базе платформы SMART-M3. Уч.пос. Изд-во ПетрГУ, 2013.
  http://elibrary.karelia.ru/book.shtml?levelID=031&id=18104&cType=1

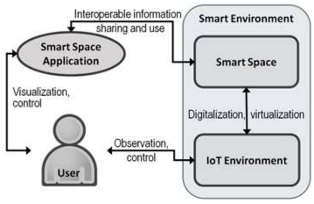
§4. Service Formalism

- Dmitry Korzun. Service Formalism and Architectural Abstractions for Smart Space Applications. Proc. Central & Eastern European Software Engineering Conference in Russia (CEE-SEC(R)), 2014.



Smart Space Application (SSA)

- Distributed system of agents hosted in IoT environment
- Smart properties of SSA:
  1. Understanding the situation where the application is used and by whom
  2. Interpreting the semantics of shared information
  3. Tolerating uncertainty at development and run time



SSA Services

- SSA acquires knowledge about the environment and its users to provide them with **services** using the best-suit resources from all kinds of participants
- Agent is a Knowledge Processor (KP) over shared content *I*
- Service development: in terms of scenarios with knowledge reasoning acts
- Control flow: initiated from the user side and completed at a point where the user perceives the service

Semantics in Smart Spaces

- Smart space aims at encompassing all information pieces the application needs for its service operation
- Semantics is a relationship (or mapping) established between such information pieces
- Informational content I; let a and b in I
  - Relation between concepts: $a \longrightarrow b$ since a in A and b in B
  - Relation between facts: $a \longrightarrow b$ is kept directly in I
  - Local relation: $a \longrightarrow b$ due to agent's decision (kept by the agent itself)

Content Representation

- Content I is a large collection of disparate information pieces (knowledge fragments)
- Corpus-based representation: I is structured (semantic relations) dynamically, in ad-hoc manner
- Ontology-driven approach: I consists of information objects and semantic relations among them
- Representation of I is an ontology graph (semantic network, OWL based, reducible to RDF for machine processing):
  - terms are structured by classes and terminological relations/restrictions
  - individuals (instances) of terms have data properties and relations between individuals (object properties) represent assertions
- Smart space provides search query interfaces to reason knowledge over I and its instant structure

Dynamic, localized space based Semantic Webs

P2P-like Network of Knowledge

- Object in I = peer (P2P node)
- A self-contained piece of information stored in the smart space
- OWL view: an individual (i.e., someone shares an instance of domain term) having data properties and linked with other objects by object properties
- P2P view: an autonomic entity with own data, linkage (semantic relations) and participation (join/leave) decision making
- P2P network G_I is formed on top of I

On the P2P level we abstracted from which KP inserts/updates/removes the object, so focusing on dynamics of information stored in the smart space

Model Properties (1/2)

- Virtualization
  - Object is a digital representation of a real thing or of an artificial entity
  - Agents (running processes) and domain entities (informational objects) are equal nodes
  - All system components become observed on "one stage" and manipulated by changing their digital representation
- Hierarchy
  - Hierarchical semantic relations
  - Concept relations, e.g., "Is-a"
  - Stable (long-term) relations by the problem domain, e.g., ontology classes hierarchy
- Composition
  - Granularity level: node clustering and aggregation
  - Short-term relations are possible: dynamic grouping

Model Properties (2/2)

- Emergent semantics
- Non-hierarchical semantic relations
- Node establishes links to other nodes (semantic neighbors)
- Subject to frequent changes
- Local relations (perceivable by some agents only)
- Data integration
- Virtual data integration system
- Some objects in I represent external data sources (and the means to access data or even reason knowledge over these data)
- Hub-like relations

Service as Knowledge Reasoning

Information Service	Control Service
1. event-based activation	1. event-based activation
2. information selection	2. information selection
3. target UI devices	3. formulation of control action
4. service delivery	4. service delivery

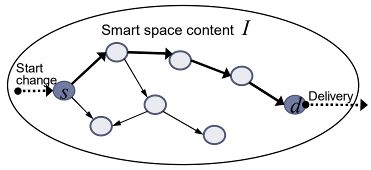
Step-wise process:

- Change of i in I forms an event observed by other participants
- When i_1 in I is changed it courses creating or updating i_2 in I, …
- The process can be branched, i.e., one change affects many objects

Architectural view: *A service is made by interaction of software agents, when each agent makes its contribution by changing objects in I*

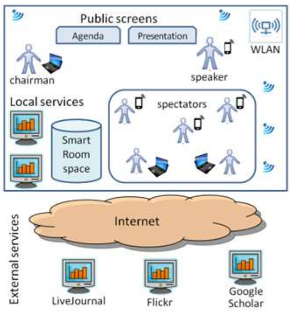
Service as P2P Path

- Injection of the change starts the service, analogous to a P2P node starting a lookup query
- The sequence of changes flows in G_I
- Note that parallel paths are possible
- Any point when an agent reads an object can be considered a final step of the service construction since the agent consumes an outcome

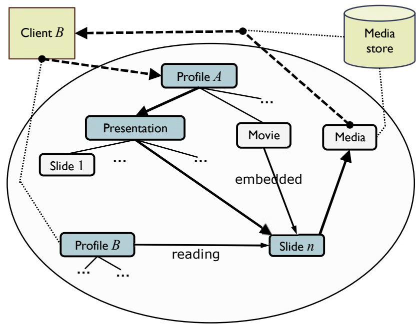


Example: SmartRoom System

- Holding collaborative activity (conferences, meetings, …)
- Deployed in room equipped with electronic devices to create a virtual workspace
  - Agenda: activity program
  - Presentation
  - Personal mobile devices
- Software agents construct and deliver services in a shared smart space
  - Local services
  - External services



Example Service: show me a slide



Architectural Abstractions

- Whole app.logic = sum of concurrent activities of KPs
- KP is responsible for links $i \to j$ of service $s \to * d$
- Event-driven programming, e.g., persistent semantic query
- Rule-based programming

Type	Description
P-C	Producer-Consumer pattern. KP_P publishes information into I. KP_C queries this information and reacts.
Pipe	$KP_0, KP_1, \ldots, KP_n$ form a kind of supply chain (linear) with source KP_0 and destination KP_n. The P-C abstraction is a particular case for $n = 1$.
Tree	Some KPs induce reaction of more than one other KPs. A kind of one-to-many synchronization with epidemic-style dissemination of changes in I.
Flow	Cyclic supply chains are possible. The KPs are organized in iterative processing flow when the same KP is activated multiple times.

Часть 4 проекта

Процессоры знаний

- Действия каждого агента (процессора знаний) в каждом сценарии построения сервисов.
- Диаграммы последовательности (для сценариев) или высокоуровневый алгоритм действий агента (как параллельный вычислительный процесс).
- Действия по доступу к интеллектуальному пространству.
- Действия (алгоритмы) по анализу данных.
- Действия по доставке сервиса пользователю