

Smart-M3 hands on training

Hannu Laine

<hannu.e.laine@nokia.com>

Jukka Honkola

<jukka.honkola@innorange.fi>



Training Contents

- Overview of M3 concept
- Overview of different communities
- Hello world example and demo
- Q & A
- Hands-on session
 - I will participate in the conference also on Thursday, feel free to ask questions also after this training



Smart-M3 basics

A Blackboard like system to store and share information between processes

Two key components

- Semantic Information Broker (SIB): stores the information

- Knowledge processors (KPs): accesses and modifies the information

Information is stored in RDF

- RDF triples constituting a graph

Operations to modify and query the information

- Insert: insert a graph to smart space

- Remove: remove a graph from smart space

- Update: update (remove and insert) a graph in a smart space

- Query: query for information in a smart space

- Subscribe: set up a persistent query

Open source (BSD license) implementation in sourceforge

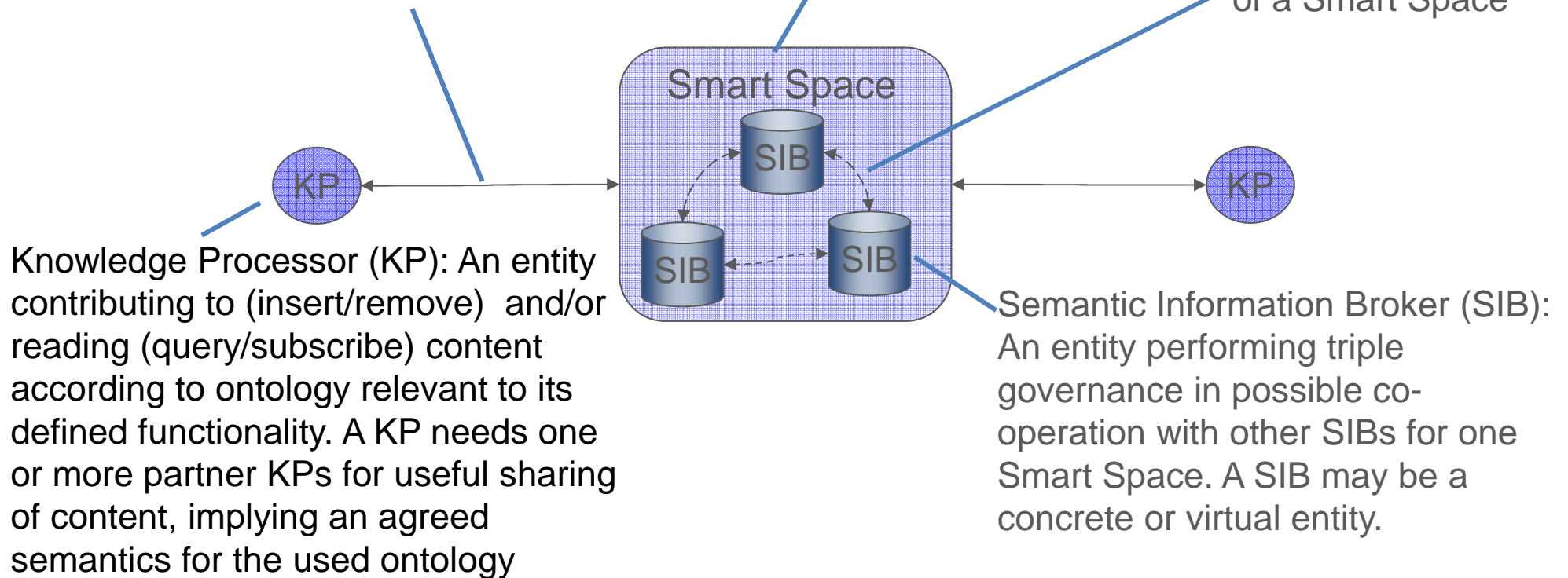


Overview

Triple governance transactions using Smart Space Access Protocol (SSAP): join, leave, insert, remove, update, query, subscribe, unsubscribe

Smart Space: a named search extent of information

Physical distribution of a Smart Space



The Big Picture

Several communities have been established to use & improve the M3 concept:

- <http://www.fruct.org/smart>
- <http://www.open-m3.org/>
- <http://www.sofia-community.eu/>
- <http://sourceforge.net/projects/smart-m3/>
 - Several related projects also in sourceforge

There is obviously some overlap...



Current State

- Smart-M3 only openly available implementation, Sofia ADK release in 1/2012
 - ADK is an eclipse plugin for developing M3 applications
 - ADK will be released in Sofia community
- Smart-M3 implementation needs work
 - Replacement for Piglet RDF store
 - Packaging of different components
 - Improving documentation
 - General improvements of different components



Applications / Demos

- Smart conferencing system done in FRUCT
- Several advanced demos done in Sofia
 - Building maintenance, car, smart lighting, ...
- Several demos done in DIEM
 - Greenhouse: <http://youtu.be/2JtErnMt368>
 - Dinetender: <http://youtu.be/Bi1TEcl-nAE>
 - Meeting Application: <http://youtu.be/yFuiC75pZZA>
 - Flowerstick: <http://youtu.be/QBxqGpDYmw8>
- No commercial deployments at the moment



ssls

- ssls is a "smart-space shell" that offers a shell-like interface to contents of a SIB
- Downloadable from <http://sourceforge.net/projects/ssls/>
- Makes debugging M3 applications easier
- Integration with smodels reasoning engine
 - <http://dl.acm.org/citation.cfm?id=1885832>
- Written with python, thus easily portable



Outline

Hello World

Smart-M3 Python KP API

Exercise Ontology

Exercise 1:

- Discover available SmartSpaces
- Join/Leave

Exercise 2:

- Subscription

Exercise 3:

- Insert/remove information

Smart-M3 “Hello World” exercise

Consists of three Knowledge Processors

Creator: inserts creatable things into the Space

Observer: observes which things exist and based on that information updates information whether the world exists or not

Greeter: monitors whether the world exists and says “Hello World”

Exercise: Develop Observer KP application with using Python API

Creator and Greeter KPs exist



Smart-M3 Python KP API

m3_kp.py

Works on python 2.5 and 2.6, tested on linux, Maemo

Pure python library that can be imported by python programs

URI(Node), bNode(Node), Literal(Node) classes used to represent RDF nodes

Triple class used to represent RDF triples

KP class used to represent knowledge processors

methods for discovering SIBs, join/leave and instantiating/destroying transaction classes

Transaction classes: Insert, Remove, Update, Query, Subscribe

Instances of these used to perform operations in SIB

Exceptions for error handling

Older version (Node.py) is deprecated

The old library contains many inconsistencies, retained for backwards compatibility



Python KP API Transaction methods

Insert

`send([Triple] insert_graph)`

Remove

`remove([Triple] remove_graph)`

Triple list may contain triples with wildcards

Update

`update([Triple] insert_graph, [Triple] remove_graph)`

Query

`[Triple] ← triple_query([Triple])`

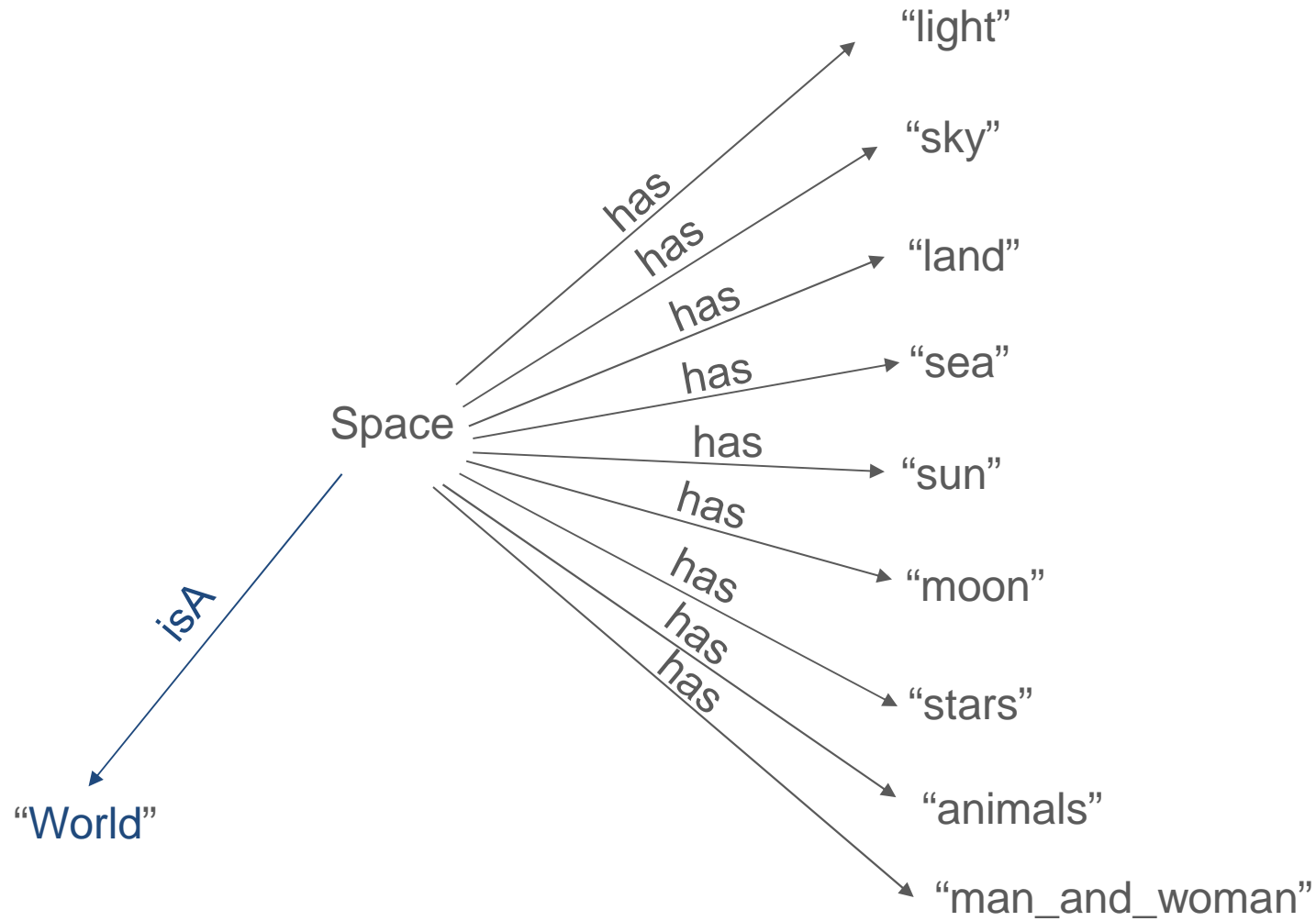
`result ← wql_*_query(wql_expression) (see documentation)`

Subscribe

As in query, callback `subs_callback(added, removed)` when results change



Hello world ontology



Exercise 1: Discovery and Join/Leave

SmartSpace discovery

Import m3_kp

Create a KP instance

Discovery by KP.discover() method

- Manual (default) method asks for smart space name and IP address
- mDNS based method exists, but implementation is outdated (does not work with SIB)

discover() returns a list of smart space handles, these can be used in join and other transactions

Joining to a discovered SmartSpace

KP.join(ss_handle) will attempt to join to a smart space

Currently no access control exists, thus join() should always succeed

Leaving a SmartSpace

KP.leave()



Exercise 1 (cont.)

Get a copy of all exercise files

Skeleton file `SS_HelloWorld_aggregator_1.py`

Edit with favorite editor

Running

Open three terminal windows

- `export PIGLET_HOME='pwd'; rm $PIGLET_HOME/X; sibd`
 - 1) Runs the SIB
- `sib-tcp`
 - 1) Runs TCP transport process for the SIB
- `python XXX.py`



Exercise 2: Create subscription

Subscription is a persistent query

Subscribe to information what items Space has.

Create a triple with python value 'None' as a wildcard URI for the unknown field

- `t = Triple(None, None, None) ~ (*, *, *)`
- The triple can be used as the query expression in `Subscribe.subscribe_rdf()`

Create instance of Subscription class

`s = KP.CreateSubscribeTransaction()` (use KP instance instead of KP)

Initiate the subscription

`s.subscribe_rdf(t, callback)`

callback should be a class with method `handle(self, added, removed)`

returns a baseline result, callback will be a diff compared to last result



Exercise 2 (cont.)

Skeleton file `SS_HelloWorld_aggregator_answer_1.py`

In addition to Exercise 1 solution contains

- Subscription initialization
- Printing the baseline result
- printing additions / removals and the new result set

Test your subscription with `hello_world/trunk/src/play_creator`

Command line application to create/delete items that the world constitutes of

Run in a terminal window (`sibd`, `sib-tcp`, `whiteboardd` must be running)

Press enter to get list of SmartSpaces and select one of them

The application shows what items exist in the Space

- Use `c1 c2 ...` to create items
- Use `d3 d4 ...` to remove items



Exercise 3: Insert/Remove information

Monitor the subscription results and check if all items that constitute the world exist

Create a triple (<Space>, <isA>, “World”)

Based on the information from subscription either insert or remove the triple from the smart space

Testing

Use setup from Exercise 2

In addition, run `SS>HelloWorld_world_observer.py` in a terminal window

Use `play_creator` to modify the SmartSpace content

Example answer in `SS>HelloWorld_aggregator_answer_3.py`

