Prototype presentation and demonstration

Vassil Momtchev
Ontotext
26/04/2007

Outline

- Phase 1 implementation goals and specifications
- First prototype features
- Structural overview of the prototype
- First prototype internals in action
- Live demonstration (aligned to EPS scenario)

Phase 1 Implementation Goals

- Develop proof-of-concept prototype for individual work packages
- Verify Triple Space conceptual model and architecture
- Ensure interoperability between the individual components and the integration middleware
- Provide test bed for the use case development
- Problems not addressed in the first prototype:
  - Security
  - Distribution

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

First Prototype Specifications

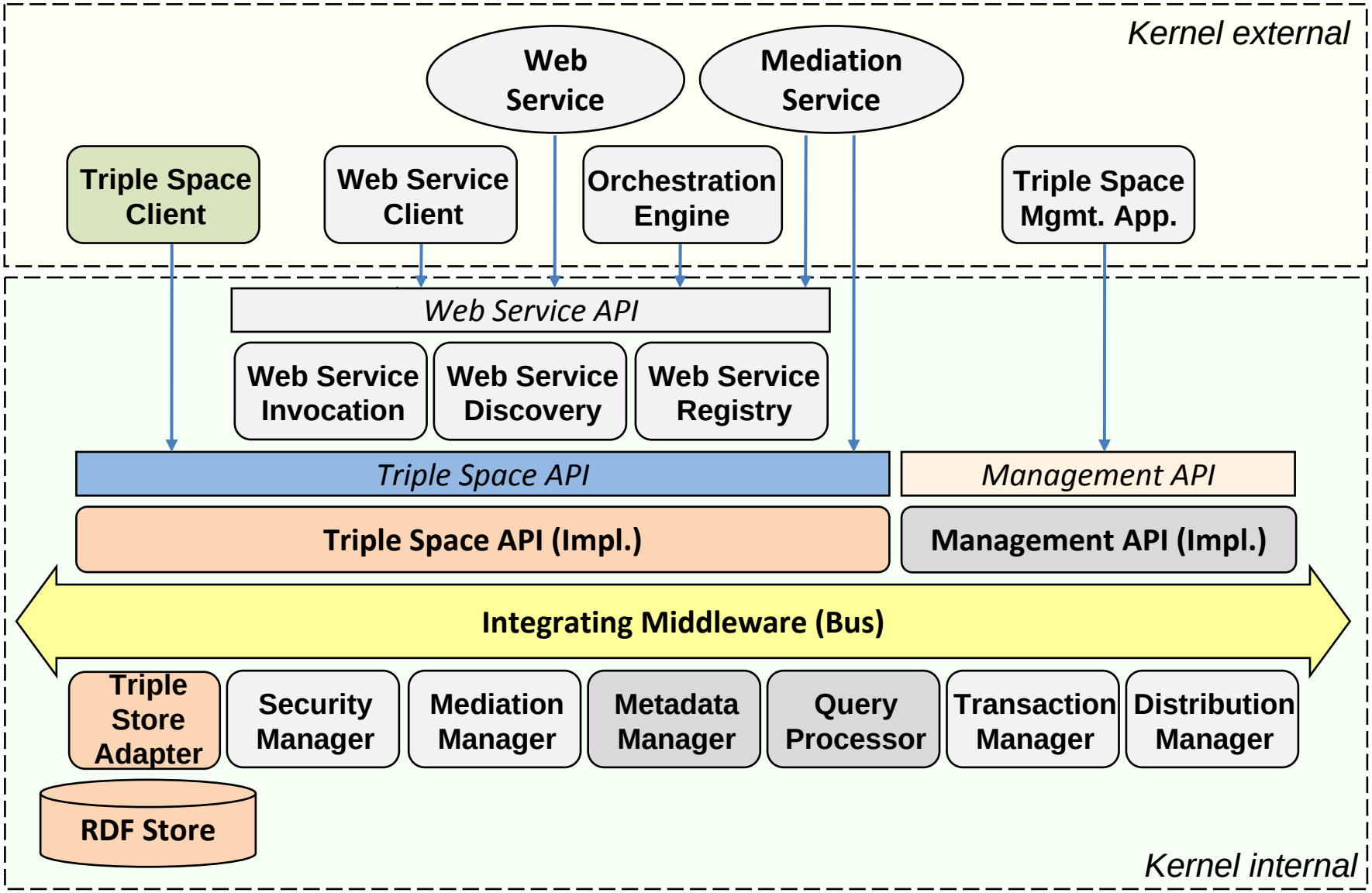
- The Triple Space Prototype implementation is designed according:
  - Specifications produced in tasks:
    - T1.2 Specification of storage model and architecture based on RDF stores
    - T2.1 Specification of representation of RDF triple semantics in tuples
    - T3.1 Specification of a semantic Linda model
  - The “Triple Space Reference Architecture” document defined by WP6

Features

- Semantic Linda model:
  - Support for the core Triple Space API operations (i.e., out/rd/in/subscribe)
  - Logical organization of semantic tuples in spaces
  - Template matching in the space
- Storage system
  - Efficient semantic matching of RDF data and the associated meta-data
  - Allow for multiple storage implementations and reasoning scenarios and strategies
  - High scalability and efficiency

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

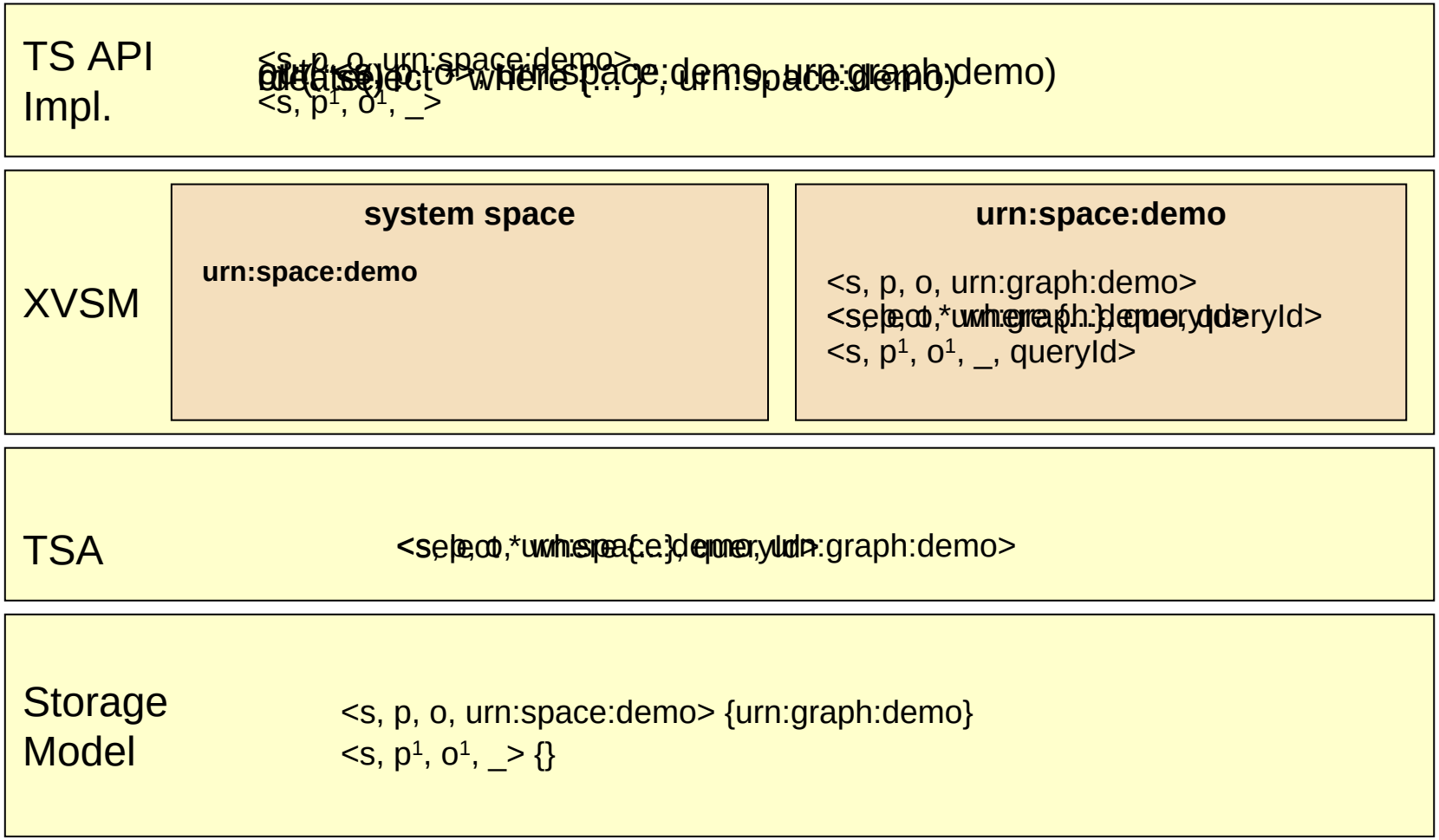
High-level First Prototype Structure



High-level Behavioural View

- Coordination Interfaces:
  - Triple Store Adapter
    - Event: Semantic data inserted in XVSM
      - Action: Insert the data in TripCom Store
      - Source: TS API implementation
    - Event: Semantic data taken from the XVSM
      - Action: Remove data from the TripCom Store
      - Source: TS API implementation
    - Event: Evaluate query request (template matching)
      - Action: Process query and writes back the result in the space
      - Source: TS API implementation
  - TS API Implementation
    - Event: Query result inserted in XVSM
      - Action: Check if the query is requested by this implementation and takes the result
      - Source: Triple Store Adapter

First Prototype Internals in Action



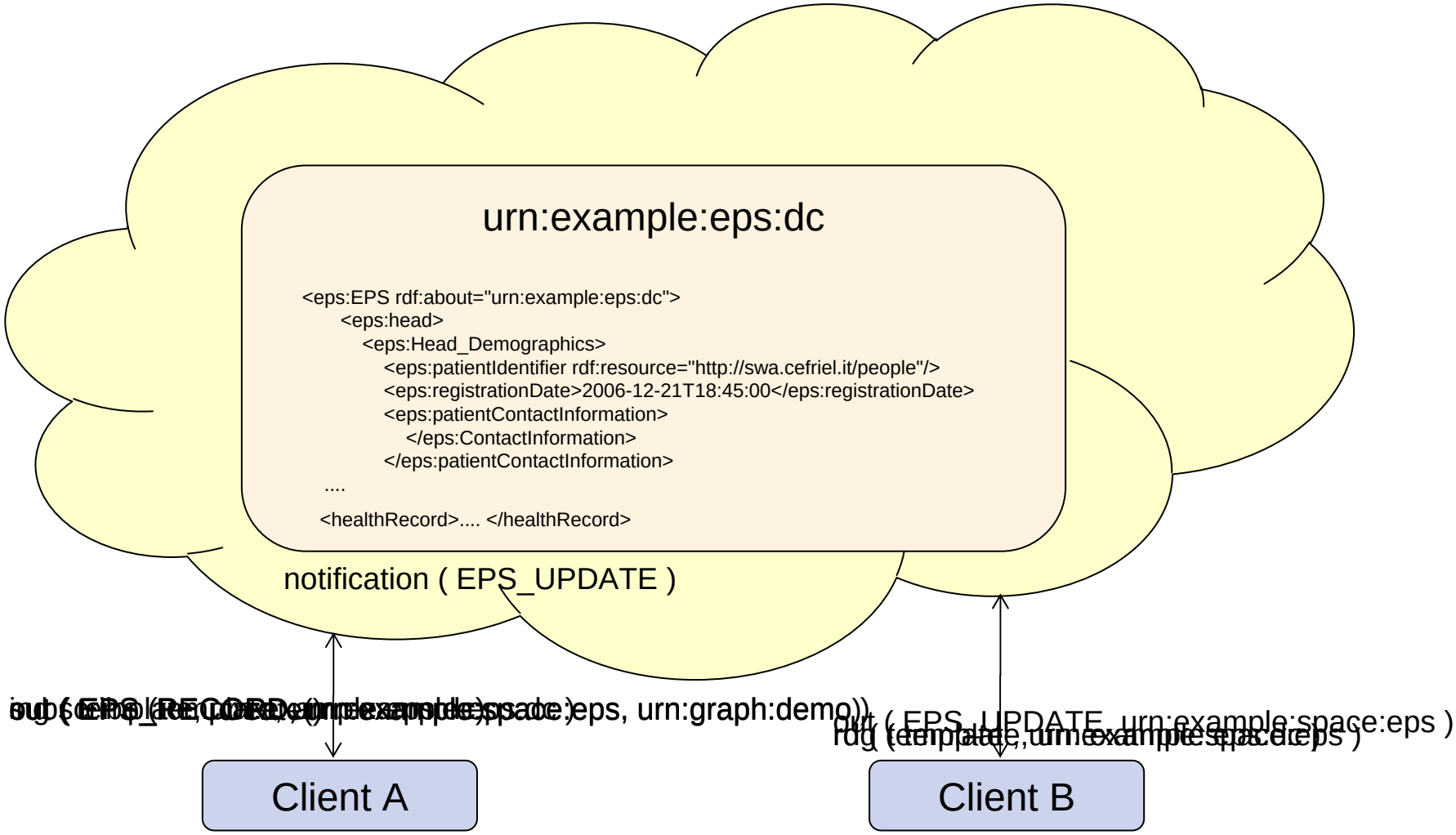
Demonstration Configuration

- XVSM System
  - Embedded mode
- ORDI Framework
  - ORDI-OWLIM Adapter
  - SwiftOWLIM 2.8.4 with owl-max ruleset (OWL-DLP+)
- log4j
  - Logging level set to DEBUG

```
public Iterator<? extends Statement> take(Resource subj, URI pred,
        Value obj, URI graph, int timeout) {
    try {
        log.debug("Data removed from XVSM container[" + uri + "] subj["
                + subj + "] pred[" + pred + "] obj [" + obj + "] ng [" + graph + "]");
        return ObjectToEntryConverter
                .deserializeDataStatements(manager.xvsm.take(cref,
                        getSelector(subj, pred, obj, graph)));
    } catch (Exception e) {
        throw new RuntimeException("XVSM search exception!", e);
    }
}
```


Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

Demonstration



Prototype Availability and Licensing

- http://sourceforge.net/projects/tripcom/
  - TSA API and implementation (LGPL)
  - Triple Store Adapter (LGPL)
- http://sourceforge.net/projects/ordi/
  - ORDI Model (LGPL)
  - OWLIM Adapter (LGPL)
  - YARS Adapter (LGPL)