Languages and Architectures for Pervasive Business Processes

Thomas Hildebrandt (hilde@itu.dk)
Programming, Logic and Semantics Group
IT University of Copenhagen

Road Map

- Pervasive Business Processes
- Formalisation of Business Processes
- A “truly” XML-based architecture

Traditional Business Processes

- long-lived processes: insurance claims, loan requests, ....
- handled by workflow management system as part of ERP system
- coordination, resource allocation, analysis & reengineering

Workflow Management System Reference Model

Pervasive Business Processes

- Shared communicating processes everywhere used for everything in any context
- Electronic medical journals, web-service composition, standardised work processes?
Challenges

- **Flexibility and Extensibility**: One language will not be perfect for every future application => process languages and engines should be extensible and adaptable.
- **Mobility**: Shared process execution on mobile disconnected devices, outsourcing & distribution of processes.

Architectures for Pervasive Processes?

- New business process languages (BPEL, WFDL, ...) based on XML.
- But languages are not extensible.
- the semantics is not based on XML.
- processes not exchanged between engines.
- process state stored in relational databases.

Reactive XML

- Persist run-time state as XML.
- Describe semantics (execution rules) by XML in general rewrite format (~ XSLT).
- General execution engine - no fixed rules!
- Easy to mix data and processes.
- Rewrite format based on formal model of bigraphical reactive systems (graph rewriting).
**Extensible BPM**

- Process Language Definition Tools
- Process Definition Tools
- Other Workflow Engines
- Workflow Client Applications
- Invoked Applications (e.g., Webservices)
- Administration and Analysis Tools

**Pervasive BPM**

- Process Language Definition Tools
- Traditional Workflow Engines
- Invoked Applications (e.g., Webservices)
- Workflow Engine & Client applications

**Shared Processes, Languages and Data in P2P XML Store**

**A simple Reactive XML example of context-dependent room allocation**

```xml
<?xml version="1.0" ?>
<r>
  <building name="itu">
    <floor name="itu3">
      <room name="3A07">
        <status bookedby="hniss" />
        <person name="hniss" />
      </room>
    </floor>
    <floor name="itu4">
      <room name="4A05">
        <status bookedby="none" />
      </room>
      <room name="4A09">
        <status bookedby="hilde" />
        <person name="hilde" />
      </room>
    </floor>
  </building>
  <positions>
    <pos name="hilde" where="4A09" />
    <pos name="hniss" where="4A09" />
  </positions>
</r>
```

**Centralized vs P2P**

- **Centralized:**
  Old well known technology, easy to update, expensive to distribute, centrally managed

- **P2P:**
  New technology, expensive to update, easy to distribute, no central management
Value-orientation

- never update data, aggressive sharing
- ideal for P2P infrastructures
- XMLstore: prototype implementation of value-oriented (P2P) XML database
  (PlanX - Henglein, Niss et al)

Perspectives of value-orientation

- ideal for P2P architectures
- complete historical data (statistics, decision making, backtracking, error correction, ...)
- effective sharing
- simple model