Dynamic Device and Service Discovery Extension for WS-BPEL

Hendrik Bohn, Frank Golatowski, Dirk Timmermann
University of Rostock, Germany
Institute of Applied Microelectronics and Computer Engineering

5th International Conference on Service Systems and Service Management ICSSSM'08

June 30 – July 2, 2008
Melbourne, Australia

0. Overview

- Motivation for process management for devices
- Basics
  - Web Service Business Process Execution Language (WS-BPEL)
  - Devices Profile for Web Services (DPWS)
  - Challenges in extending WS-BPEL with DPWS functionality
- Discovery concept
  - Discovery interactions, announcements and proxies
- Extensions for discovery
  - Integration options: Extension activities, existing WS-BPEL activities
- Evaluation of prototypes
  - Prototyping and comparison
- Conclusions and future work

1. Motivation – The innovation is the Integration!

- A lot of heterogeneous devices from different manufacturers need to be connected and interact with each other
- **Connection:** SOA approach forms a solid foundation but
- **Interaction:** Composition and automation of services/components will reveal the full potential

1. Motivation – Problem!

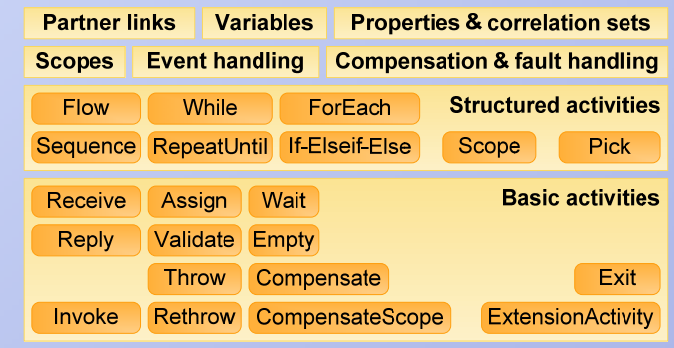
- Connectivity between heterogeneous devices is provided by several SOA approaches

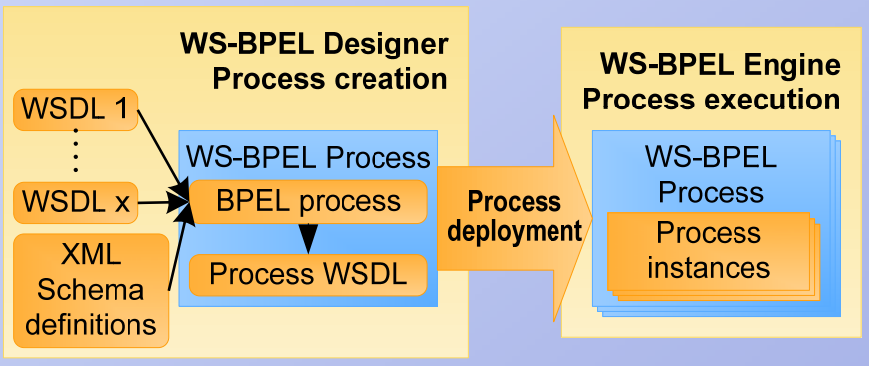
- Composition standards **only** specified for Web services being provided by software components (e.g. WS-BPEL)

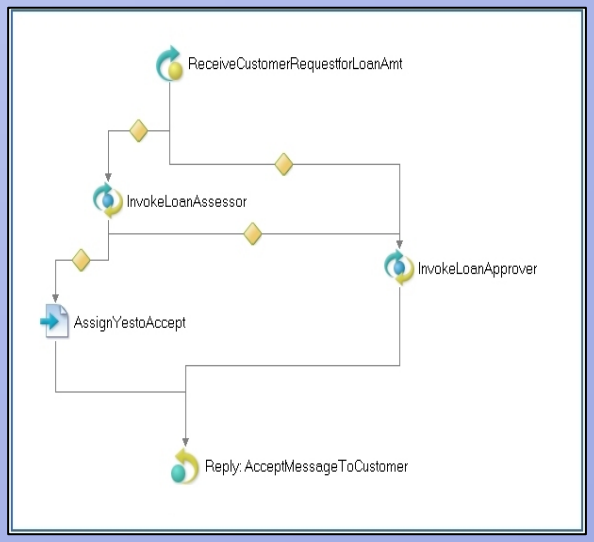
Conclusion:
WS-BPEL for the Devices Profile for Web Services
→ Extending WS-BPEL
→ Implementing a compliant WS-BPEL engine

2. Basics – Web Services Business Process Execution Language

- WS-BPEL allows the structured programming of processes (condition statements, loops, parallelism)
- Process is an executable Web service with an own WSDL being able to invoke other Web services, callback functionality
- Event, fault and compensation handling as well as data manipulation
- Data manipulation, callback functionality

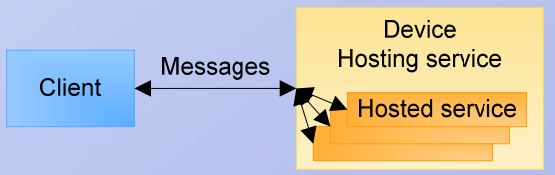


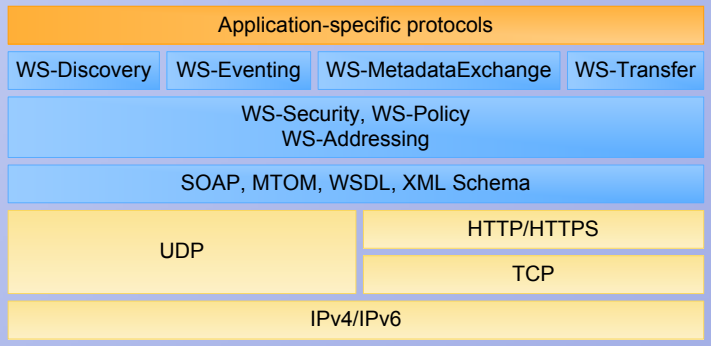




2. Basics – Devices Profile for Web Services (DPWS)

- Addressing mobile devices (Changing addresses as well as availability)
- Services reside on devices (Close binding to device and location), which are stateful and might have limited resources
- Web service messaging (SOAP) and description (WSDL), Data formats and types (XML Schema)
- Protocol-independent addressing (WS-Addressing)
- Interaction requirements and service capabilities (WS-Policy)
- Secure interactions (WS-Security), Publish/subscribe mechanism (WS-Eventing)
- Dynamic device and service discovery (WS-Discovery, WS-MetadataExchange, WS-Transfer)





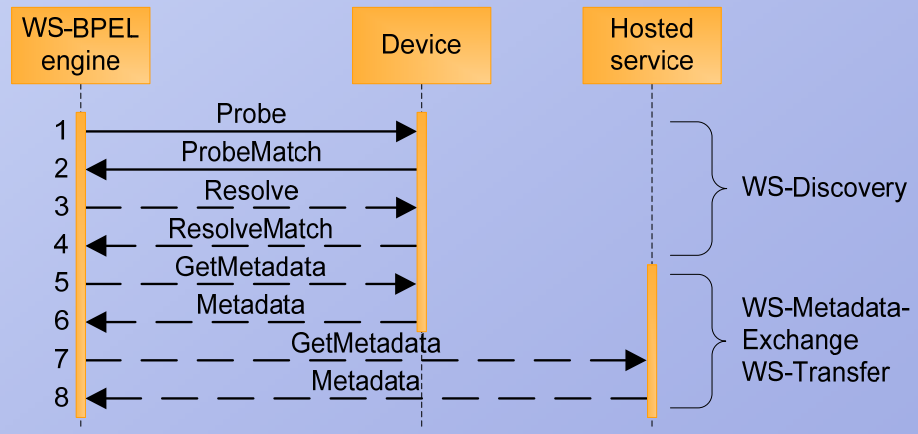
2. Challenges in extending WS-BPEL for DPWS

- Integration of and support for devices
- Dynamic discovery of devices and their hosted services (WS-Discovery)
- Publish/subscribe mechanism (WS-Eventing)
- Protocol-independent addressing of devices and services as well as messages (WS-Addressing)
- Evaluation of policies and security requirements (WS-Policy)

Focus of this paper:
Dynamic device and service discovery for WS-BPEL

3. Discovery concept – Discovery interactions

- Searching for a device of a certain type and scope using multicast UDP (Probe)
- Matching devices answer using unicast UDP including their endpoint references (ProbeMatch)
- Requesting transport address of desired device (Resolve)
- Requesting device metadata which includes endpoint references of hosted services (GetMetadata)
- Requesting service metadata (GetMetadata)



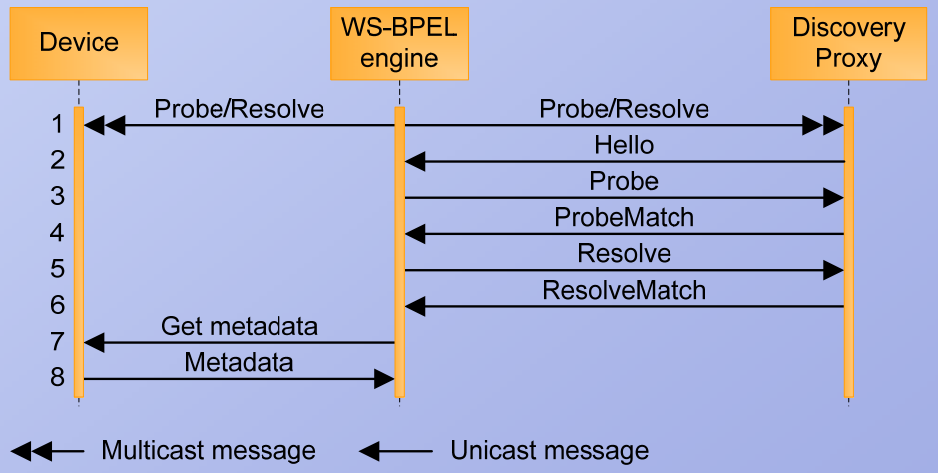
3. Discovery concept – Discovery announcements and proxies

Announcements

- Devices announce their endpoint references when entering and leaving a network (Hello/Bye)
- May include types, scopes and transport addresses

Discovery proxy

- Used for network-spanning discovery
- Suppresses multicast discovery



4. Extension for discovery – Integration options

Using extension activities

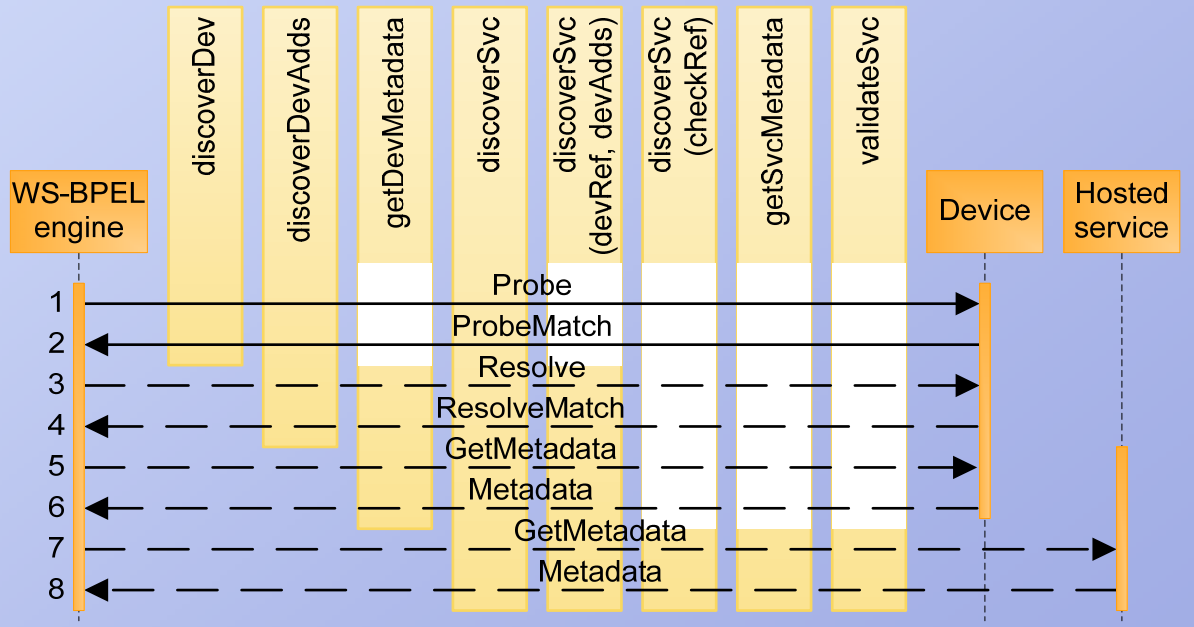
- Discovery process is handled by the WS-BPEL engine
- Extension activities are used to initiate discovery and receive the discovery results

Using existing WS-BPEL activities

- Discovery process is implemented by the process designer
- Invocation activities are used to model the discovery interactions

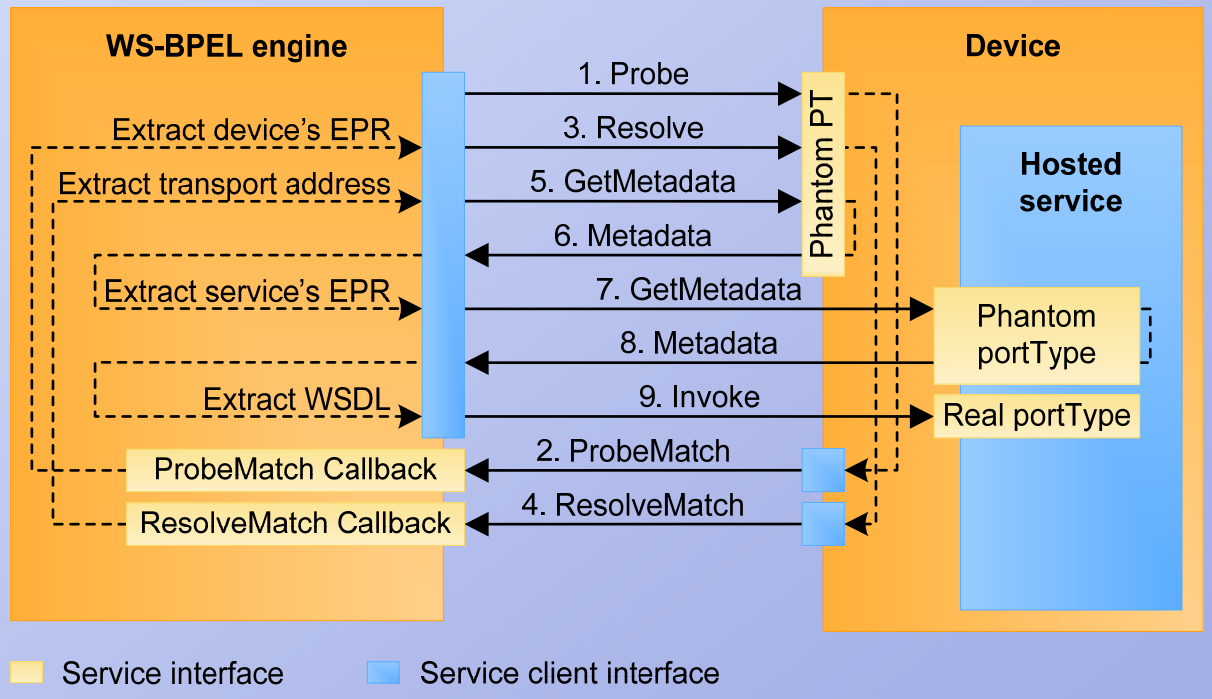
4. Extension for discovery – Using extension activities

- Several extension activities cover the discovery process and offer flexibility for the process designer
- Extension activities may be independent of the actual performance of the discovery process as
- WS-BPEL engine takes care of discovery announcements and provides an internal database for available devices



4. Extension for discovery – Using existing WS-BPEL activities

- Wild-card WSDL descriptions for devices and services
- WS-Addressing actions are used to identify WSDL discovery operations
- Discovery message content is accessed directly from the process



5. Evaluation of approaches – Prototyping

Prototype using extension activities

- Idea: Extending a DPWS-Stack with WS-BPEL functionality
- Java/Axis2-DPWS stack, BPEL transformation with XML parsing and XSLT
- Supporting simple Web service invocation including discovery extensions

Prototype using existing WS-BPEL activities

- Idea: Using existing WS-BPEL engine; model and integrate discovery functionality
- ActiveBPEL v5 (supports WS-Addressing)
- Simulated over HTTP as ActiveBPEL does not support SOAP-over-UDP and WS-BPEL does support neither notification nor solicit-response MEPs
- Several design patterns developed: Several answers to probe messages, fault handling in discovery, fault handling by discovery, applying other search criteria, handling device announcements, handling a discovery proxy

5. Evaluation of approaches – Comparison

Prototype using extension activities

- Discovery is performed by the WS-BPEL engine/transformer
- Process designer concentrates on Web service interactions
- Less influence on handling of faults & selection of alternatives

Prototype using existing WS-BPEL activities

- Discovery is modeled by the process designer
- All WS-BPEL concepts are available to be included in the discovery process
- Design patterns have been developed for recurring design problems
- Currently no WS-BPEL supporting DPWS discovery due to missing support for SOAP-over-UDP, multicast UDP and multicast suppression

Discovery extension using existing WS-BPEL activities will be followed up due to its flexibility.

6. Conclusions and future work

- Dynamic device and service discovery for WS-BPEL integrated
- Using existing WS-BPEL activities offers more flexibility to process designers and provides existing WS-BPEL concepts
- Current WS-BPEL engines do not support SOAP-over-UDP, multicast UDP, multicast suppression
- Client functionality for WS-MetadataExchange and the Get-Operation of WS-Transfer can be emulated

- Ongoing research on the publish/subscribe mechanism and security extensions for DPWS

Which questions are arising?

Thank you for your attention!

Hendrik Bohn
University of Rostock
Institute for Applied Microelectronics and Computer Engineering
hendrik.bohn@uni-rostock.de