Vorbereitungsseminar
Bachelorprojekt ASG
SS 2005

Workflow/Web Service Composition

Torsten Hahmann

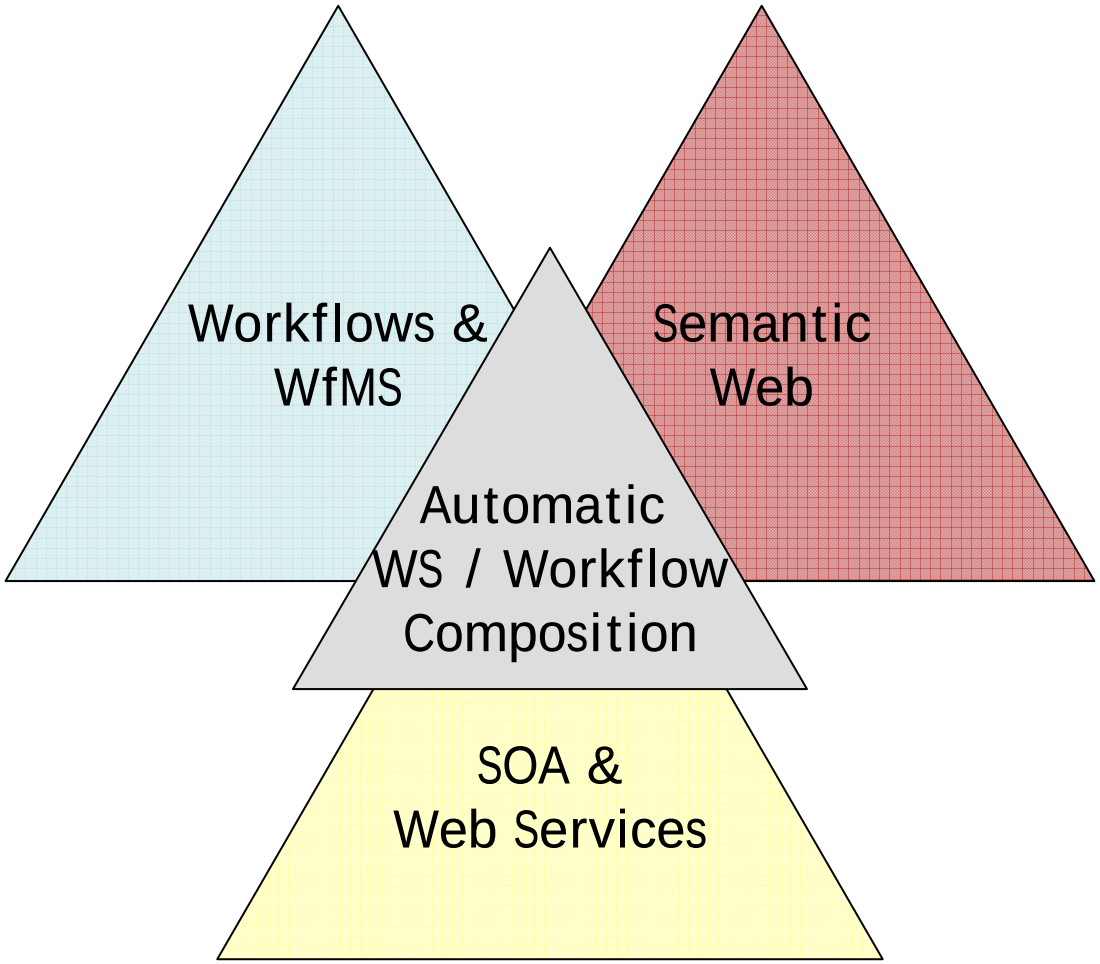
Agenda

- Introduction
  - Workflow & Web Service Composition
  - Workflow Management Systems
- Automatic Web Service Composition
  - Process definition language: BPEL
  - Semantics (Semantic Web Services)
  - Process composer (Planner)
  - Workflow Engine
- Summary

Agenda

- Introduction
  - Workflow & Web Service Composition
  - Workflow Management Systems
- Automatic Web Service Composition
  - Process definition language: BPEL
  - Semantics & Semantic Web Services
  - Process composer (Planner)
  - Workflow Engine
- Summary

Overview


Workflows &
WfMS
Semantic
Web
Automatic
WS / Workflow
Composition
SOA &
Web Services

Workflow – definition

- Workflow: *Process that can be automated by invoking applications or external services and/or assigning manual tasks*

- Workflow Composition: *arranging activities to form a business process*
  - if invocation is limited to Web Services calls:

Workflow Composition
=
Web Service Composition

Web Service Composition

- Web Service Composition in a broad sense:

  „the automatic selection, composition, and interoperation of Web services to perform some task, given a high-level description of an objective“ [OWL-S: Semantic Markup for Languages]

  - Web Service discovery
  - Composition (in a narrow sense) of new Services
  - Execution of new, composite Services

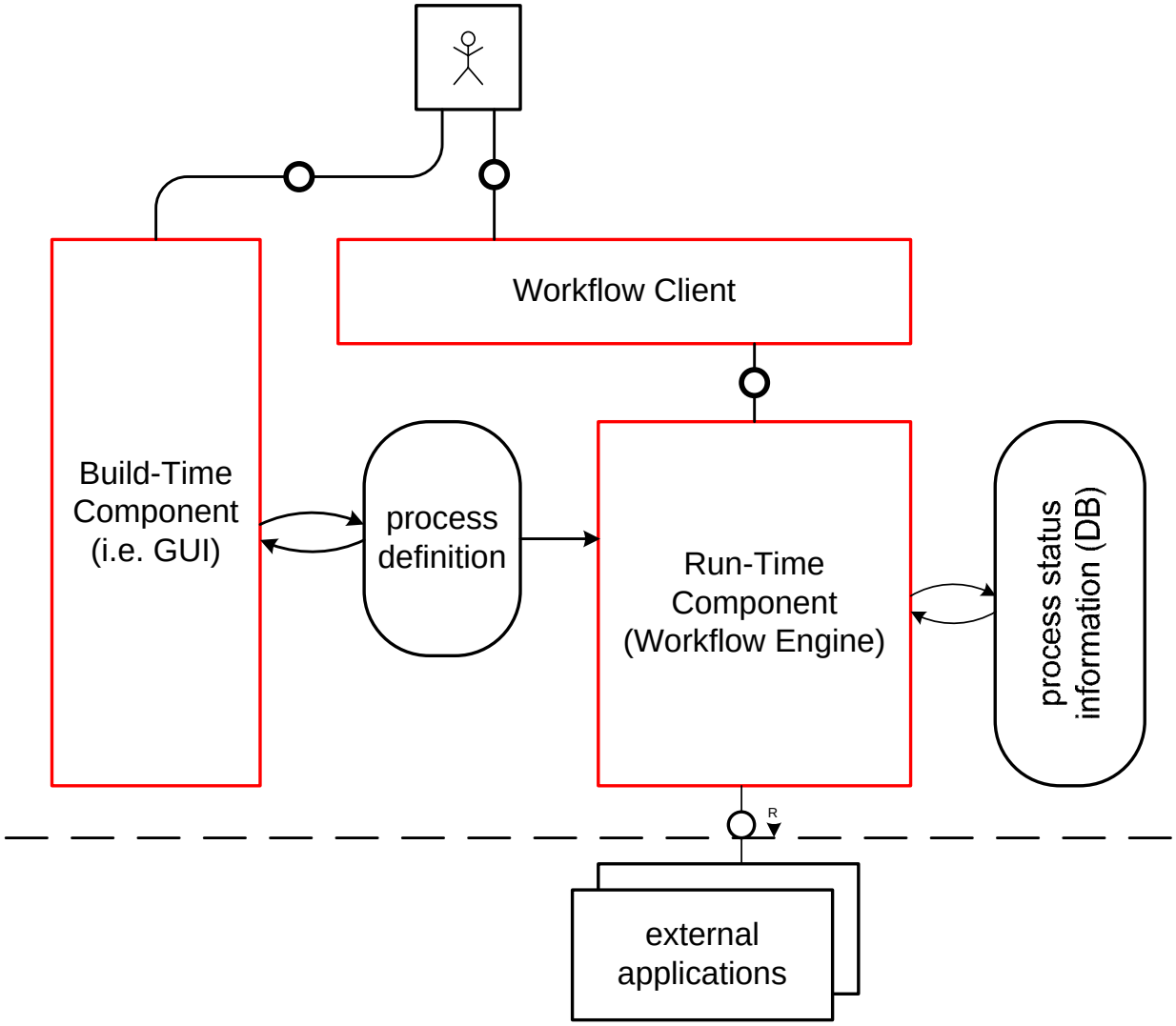
Web Service Composition (contd.)

- two different approaches:

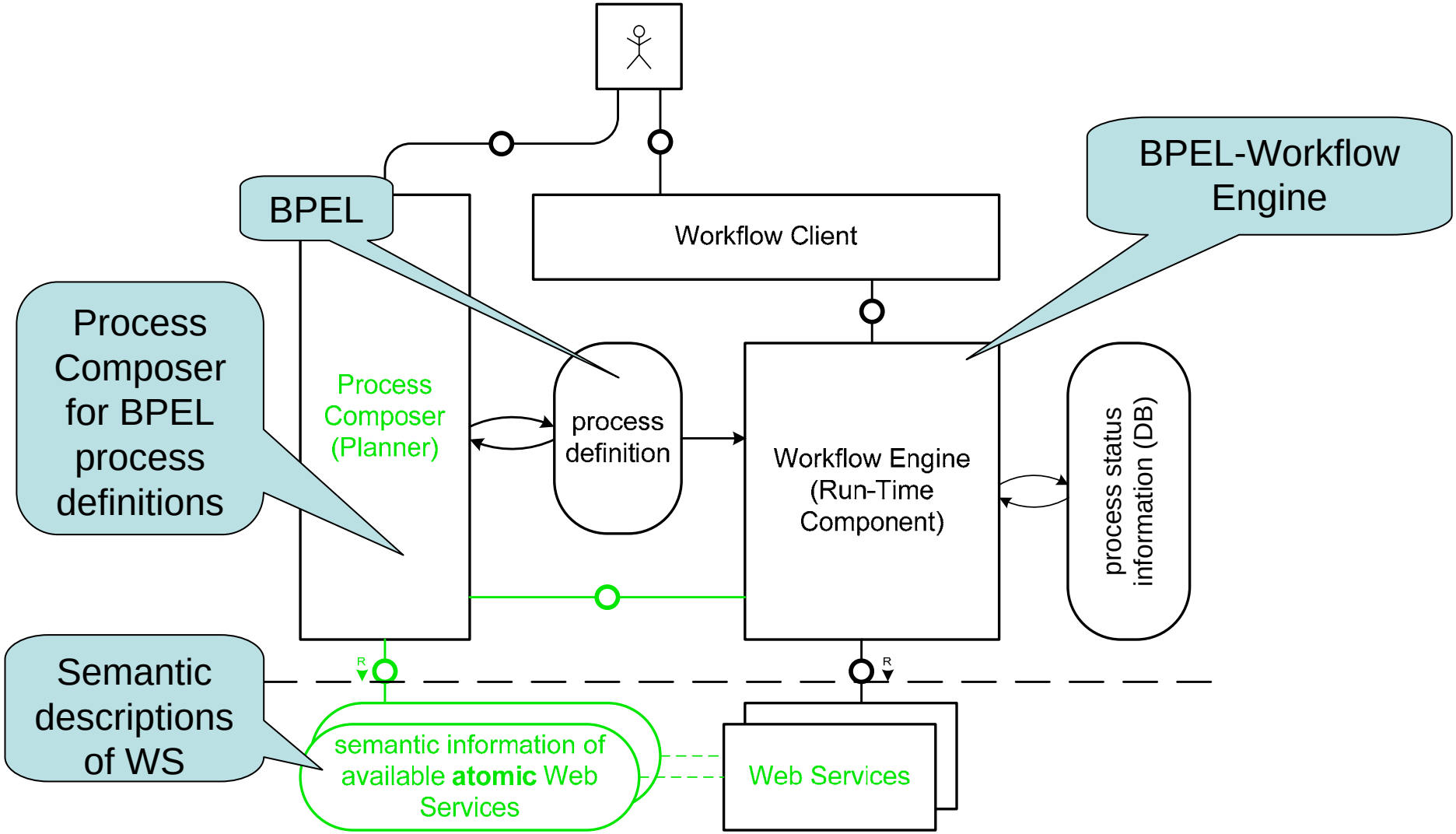
1. low-level process modeling and execution languages (like WS-BPEL)
   - directly executable in existing engines
   - manual definition of new (composed) processes that interact with existing ones
   - does not allow for automation

2. high-level unambigious description language for Web Services (like OWL-S)
   - allows reasoning about web services
   - automation of discovery and composition possible

Workflow Management System (WfMS)



WfMS for Automatic Composition



Parts allowing WS Composition

1. Language for process definitions: BPEL
   - must be supported by available workflow engines
2. Semantics to describe capabilities of WS and requested functionality
3. Process composer (Planner) which creates BPEL process definitions to fulfill a request
   - based on semantic descriptions of
     - request
     - available atomic Web Services
4. Workflow Engine that can work with BPEL process definitions

Agenda

- Introduction
  - Workflow & Web Service Composition
  - Workflow Management Systems
- Automatic Web Service Composition
  - Process definition language: BPEL
  - Semantics (Semantic Web Services)
  - Process composer (Planner)
  - Workflow Engine
- Summary

Process definition language: Why not WSDL?

- typical business interactions:
  - sequences of peer-to-peer messages (synchronous and asynchronous)
  - long-running, stateful

  ➔ protocol for message exchange needed

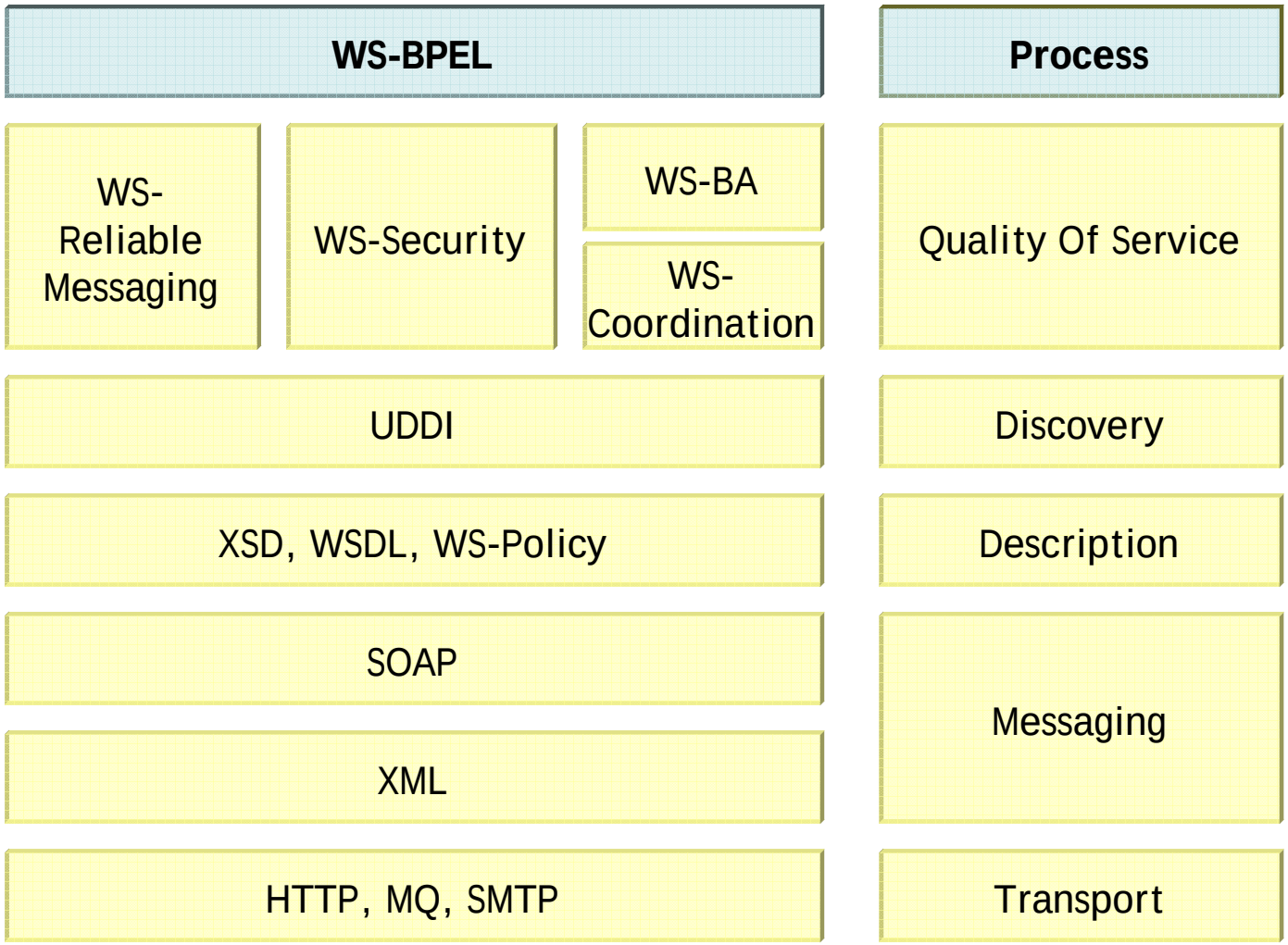
- WSDL
  - based on stateless interaction model
  - only synchronous (request/response) and uncorrelated asynchronous interactions

  ➔ specifies only method invocations (no order)

BPEL - background

- Process modeling language based on Web services
- BPEL4WS was originally developed by BEA, IBM, Microsoft and (later joined) SAP, Siebel
  - version 1.0 proposed in 2002
  - current version: 1.1, 2.0 as draft
  - based on the specifications: WSDL, XML Schema, XPath, WS-Addressing (allows standardized addressing)
- Today OASIS is in charge of the standardization of BPEL → called WS-BPEL

BPEL in the Web Service Stack



Process definition language: BPEL

- conceptual separation of
  - abstract vs. executable process

  ➔ internal, executable process can be altered without changing the abstract process
- supporting two-level programming model
  - „programming in the large“ ➔ abstract process vs.
  - „programming in the small“ ➔ executable process
- common core of process descriptions concepts
  - BPEL specification focused on common core
  - extensions required for private, abstract processes

BPEL (contd.)

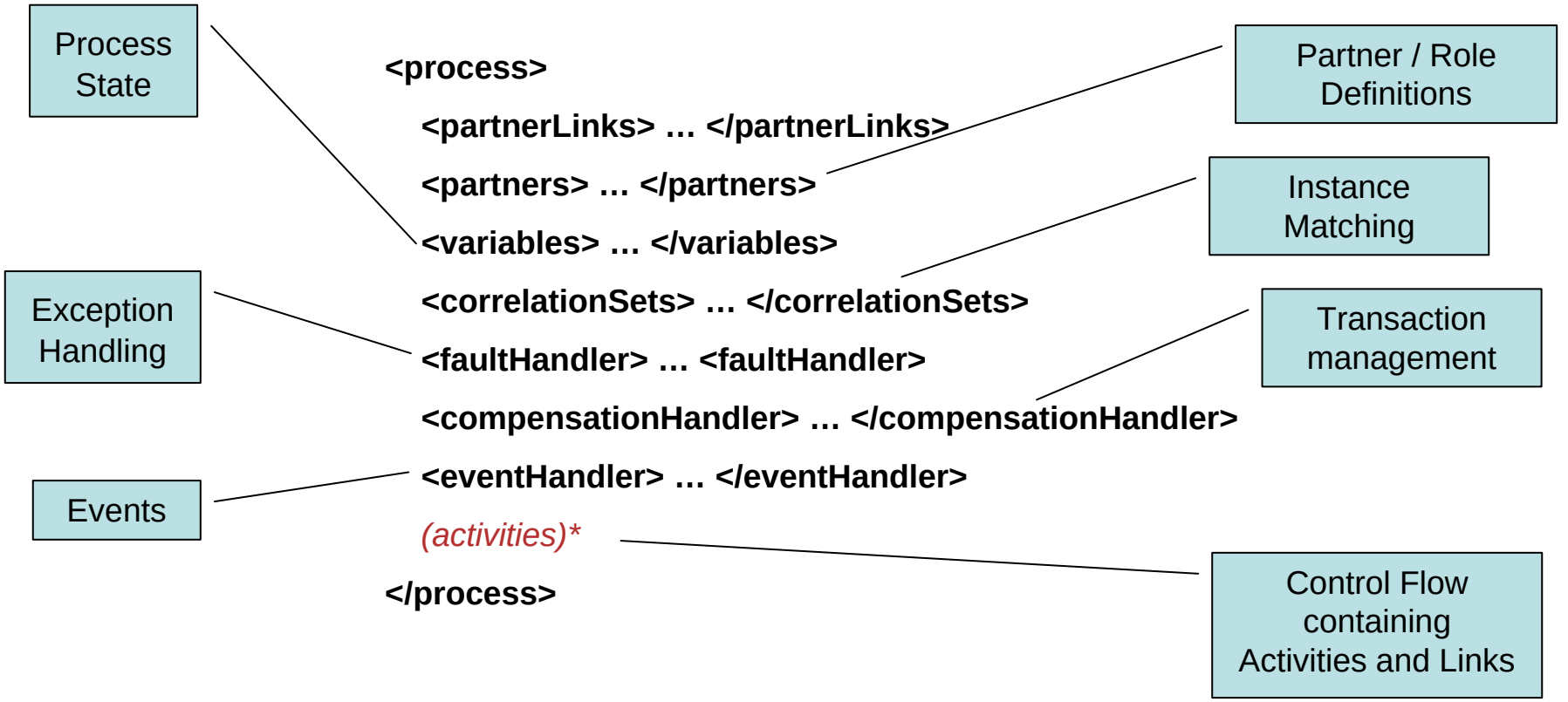
describes ...

1. abstract process (public)
   - public message exchange protocol revealed
     - external behaviour of services must be exposed
     - defines a business protocol rule
   - grounded on the private process
   - data handling on an abstract level (handles only protocol-relevant data)
2. executable processes (private)
   - executable internal processes
     - used internally during run-time
   - not exposed (information hiding)

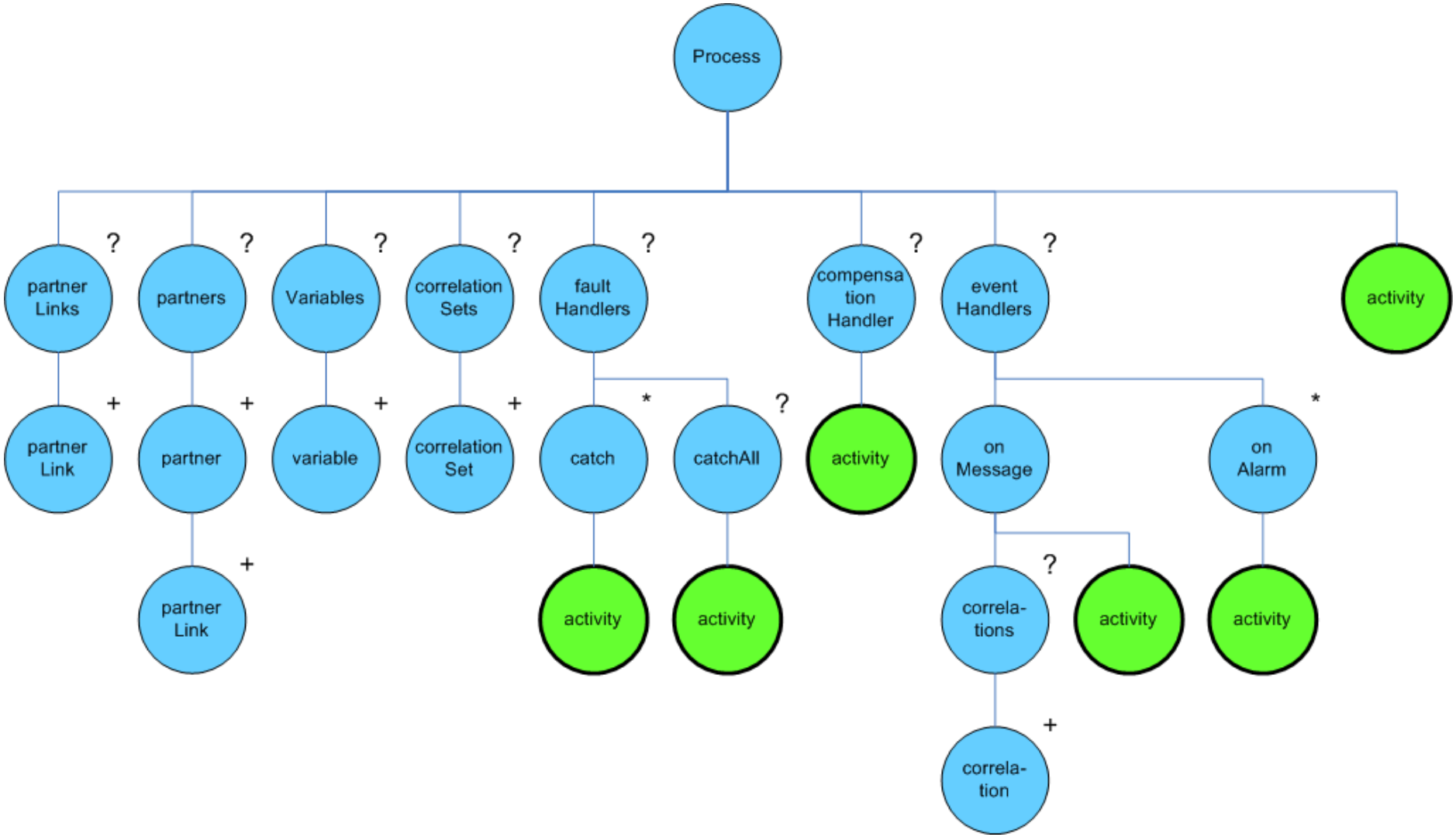
external WS provide abstract description ➔ possible Input for composition

Output of composition planner creates executable process

BPEL syntax in XML

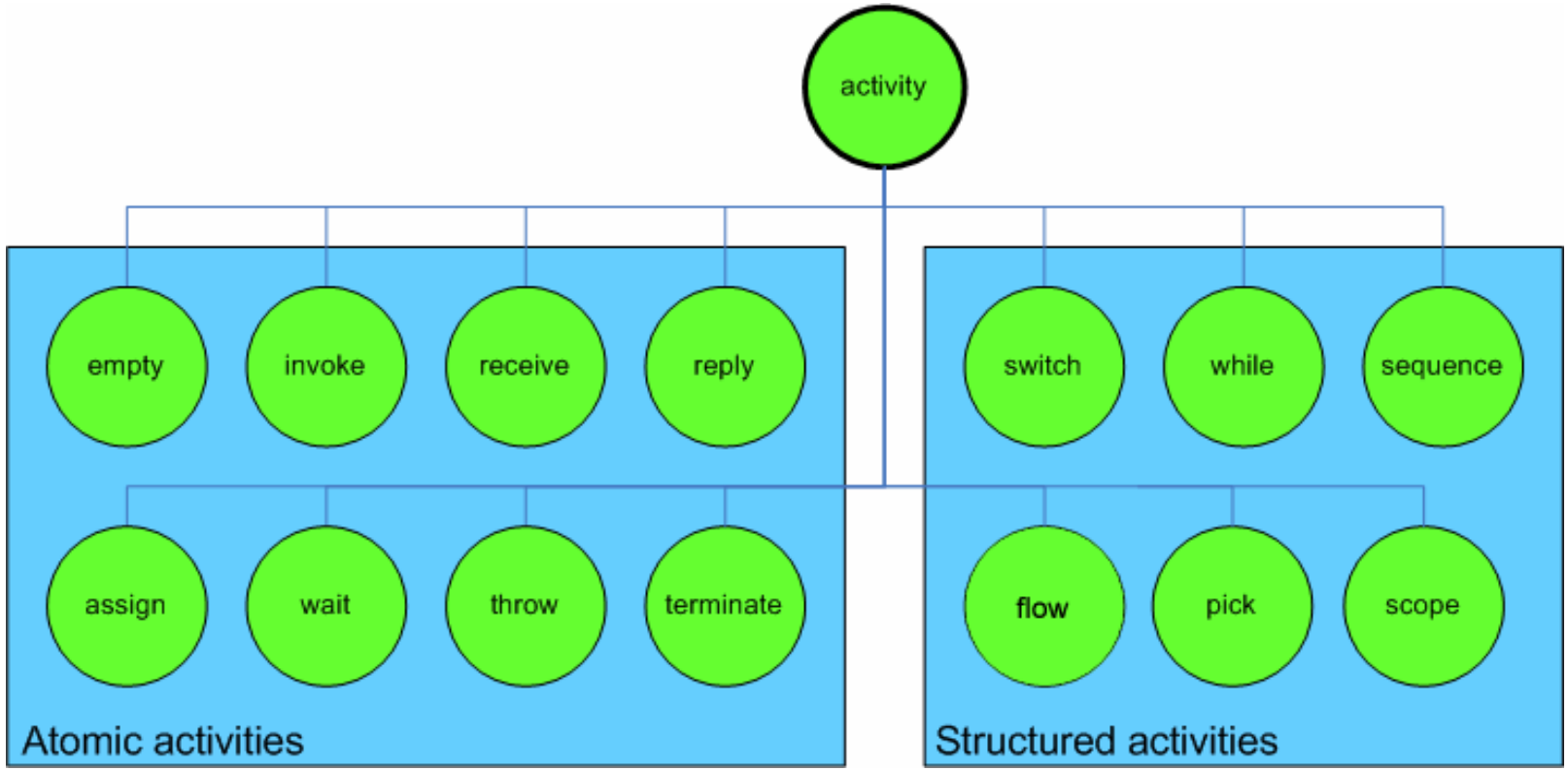


BPEL – Language constructs



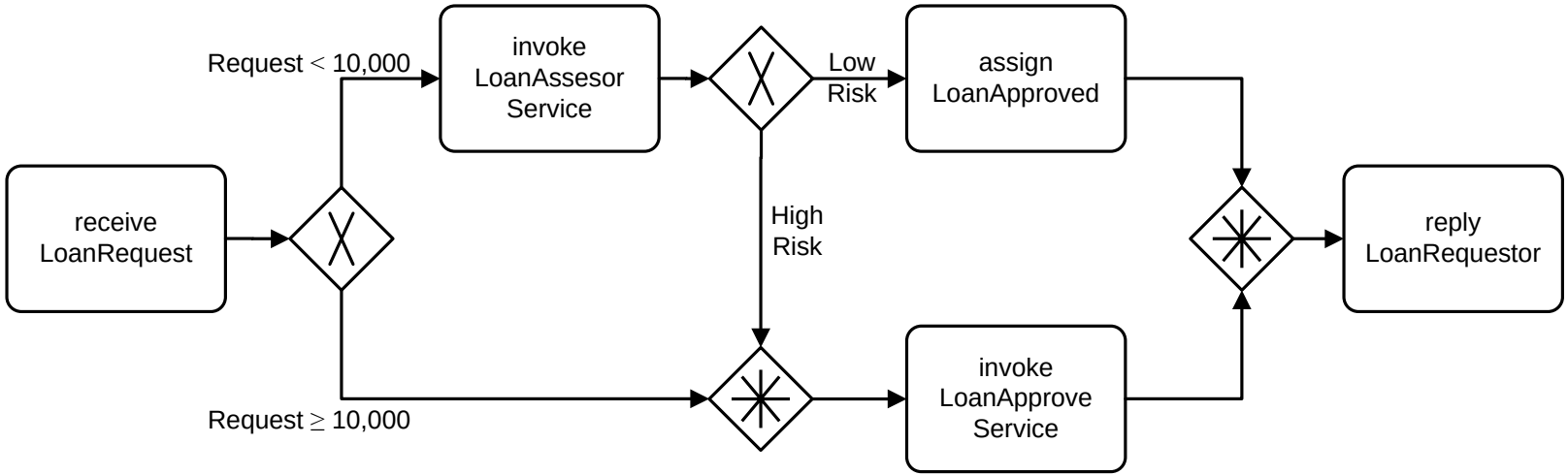
BPEL – Language constructs (contd.)

- Types of activities



BPEL – example

loanapproval.bpel



BPEL - expressivity

Basic patterns	<sequence>, <flow>/<link>, <switch>
Advanced branching/synchronization	
Multiple Choice	<link>
Synchronizing Merge	<link>
Multiple Merge	
Discriminator	
Structural patterns	
Abritrary cycles	
Implicit termination	by default
State-based patterns	
Deferred Choice	<pick>
Interleaved parallel routing	(<flow>), serializable scope
Milestone	
Cancellation patterns	<terminate>

Semantic Web Services (SWS)

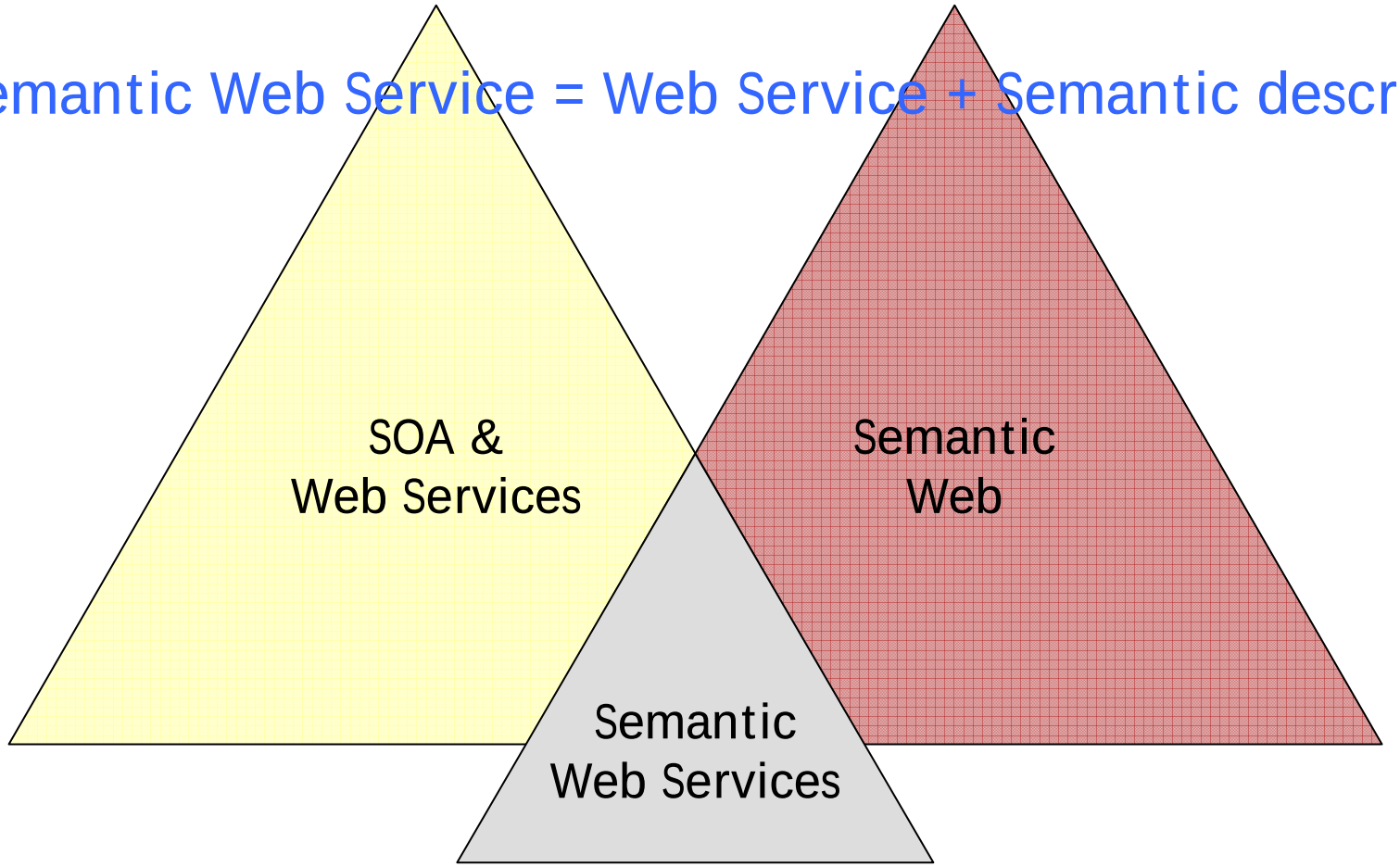
„Semantic Web concepts are used to define intelligent web service, i.e., services supporting automatic discovery, composition, invocation and interoperation.“

„These efforts try to improve current web service technology around SOAP, WSDL and UDDI, which provides very limited support for real automation of services.

[R. Lara et al.: „Semantic Web Services: description requirements and current technologies“]

Semantic Web Services (contd.)

Semantic Web Service = Web Service + Semantic description



Challenges to tackle with SWS

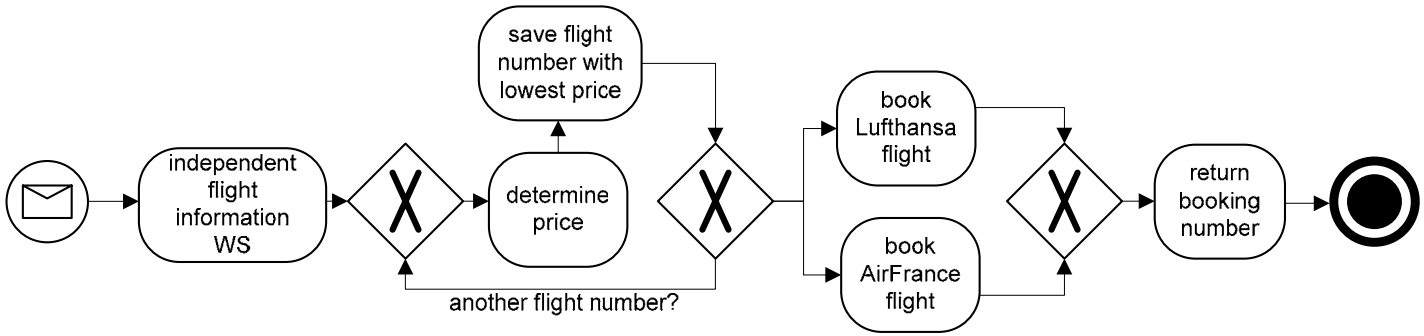
- Automatic discovery of services
  - Semantic match between declaritive description of services sought and services offered
- Automatic composition of services
  - Allow the composition of services to provide functionality that available services cannot provide individually
- Both tasks need a declarative language to describe semantics of available and sought services (goal)
  - „**Capability**“ of a Service

[R. Lara: „Semantic Web Services: description requirements and current technologies“]

Static Composition – Scenario 1

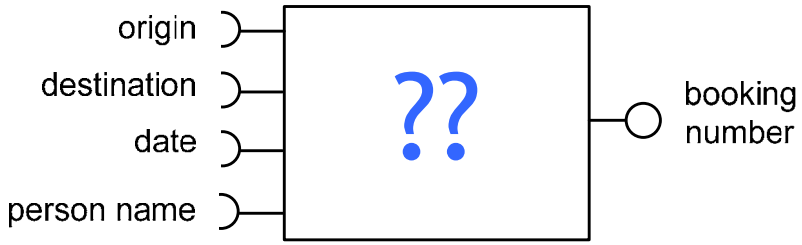
- User wants to book a flight (as cheap as possible)
  - Given origin, destination, date and person name
- Travel agency has defined business process (as workflow) for „flight booking“
  - Composition is explicitely modeled

    ➔ No composition during run-time required
  - Classical discovery of available Services necessary
    - fixed input & output for every activity, process is pre-defined

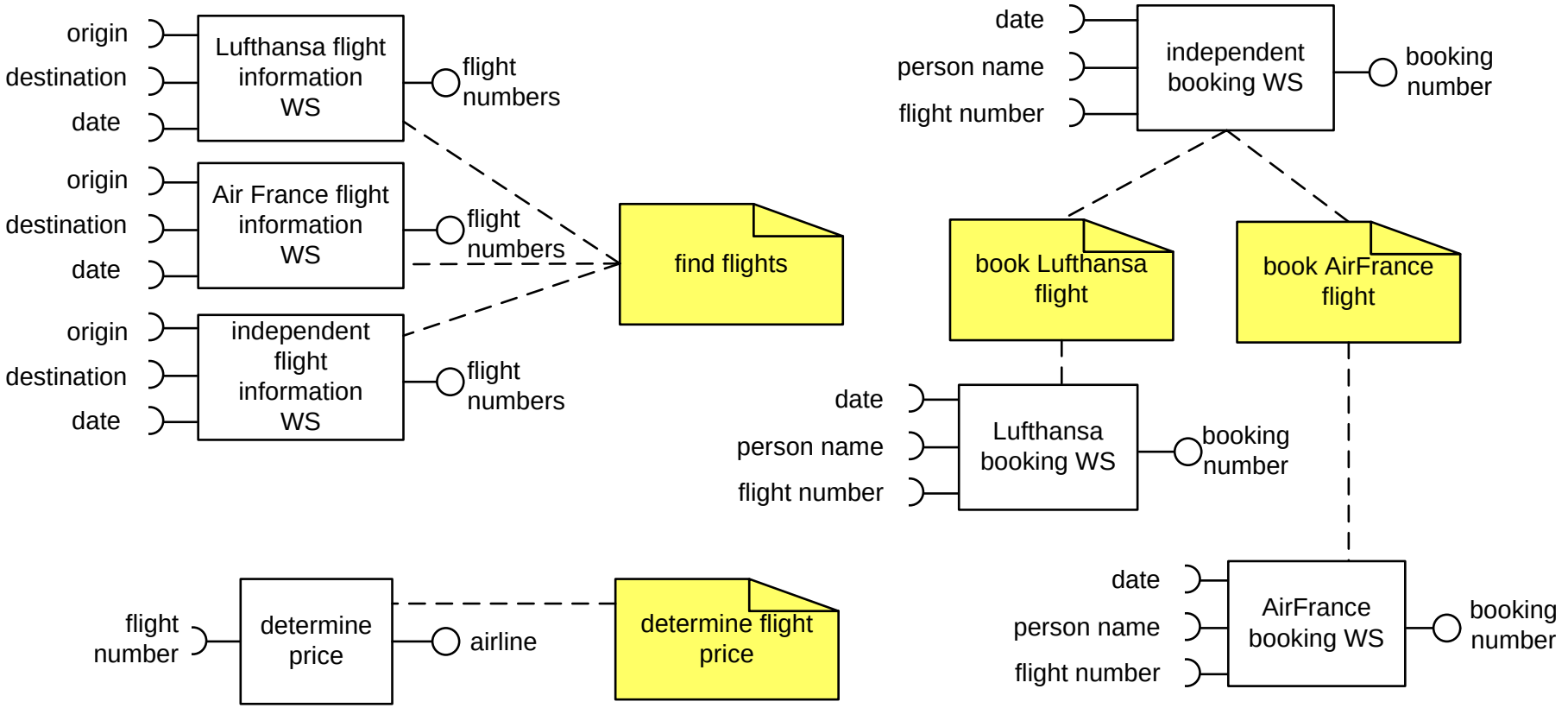


Dynamic Composition – Scenario 2

- Such service returning a flight-ticket and seat reservation is not available – alternative:
  - Service for flight information
  - Service for price determination
  - Service for flight booking
  - ➔ Composition of services necessary:
    - Process not pre-defined
    - Discovery in an extended sense:
      only initial input and final required output is fixed



Discovery & Composition - example



Capability requirements

- Must include:

1) Pre-conditions and Post-conditions (effects)
   - Expresses generic functionality, so that more specific concepts in the ontology can be found
     - E.g. „payment information“ as pre-condition instead of „credit card information“ or even data-types (as in WSDL)

1) Textual description

1) Service references (necessary for refinement and invocation of Services)

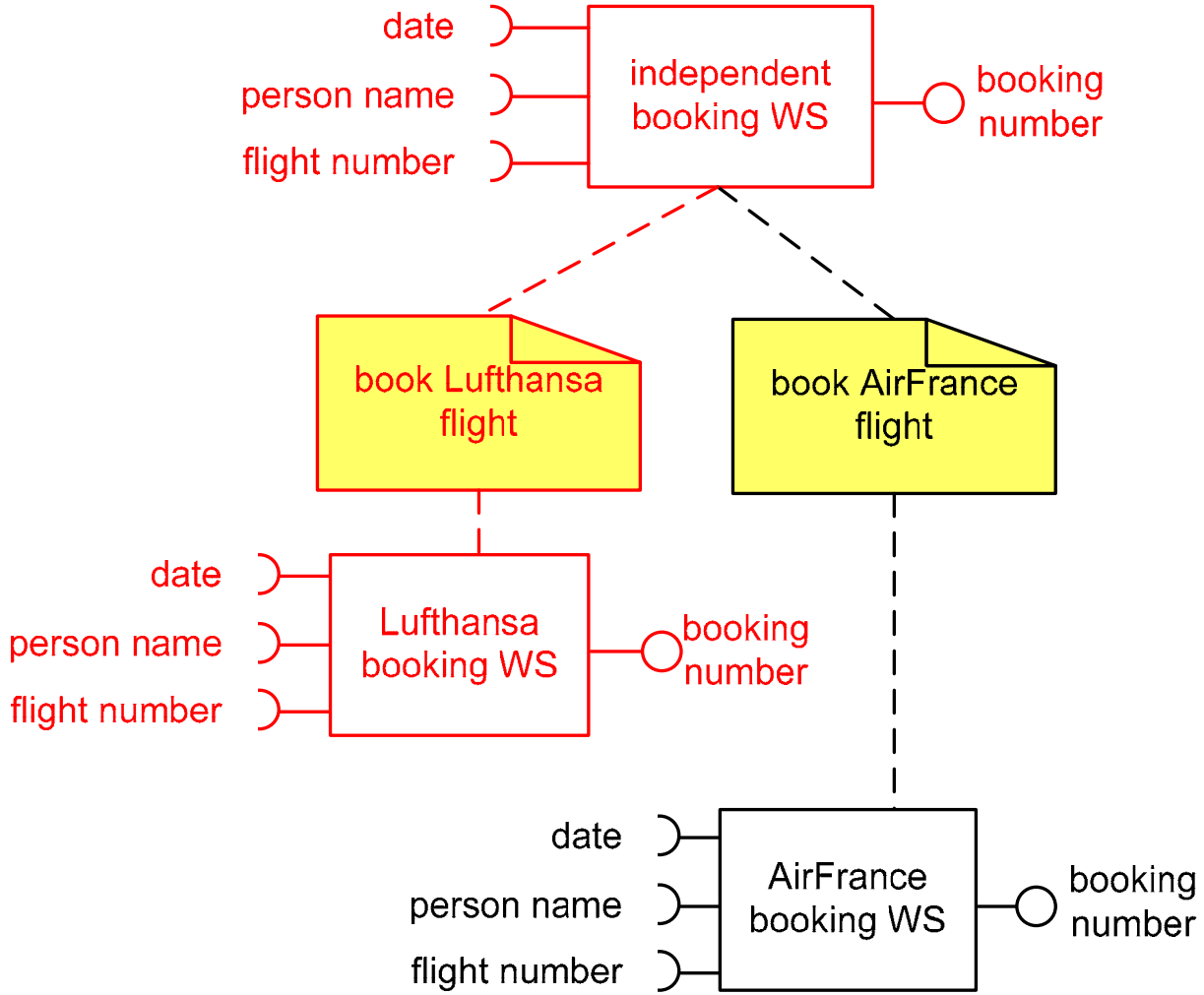
1) Identifier (to allow references, also necessary for each pre-condition and each post-condition)

Capabilities

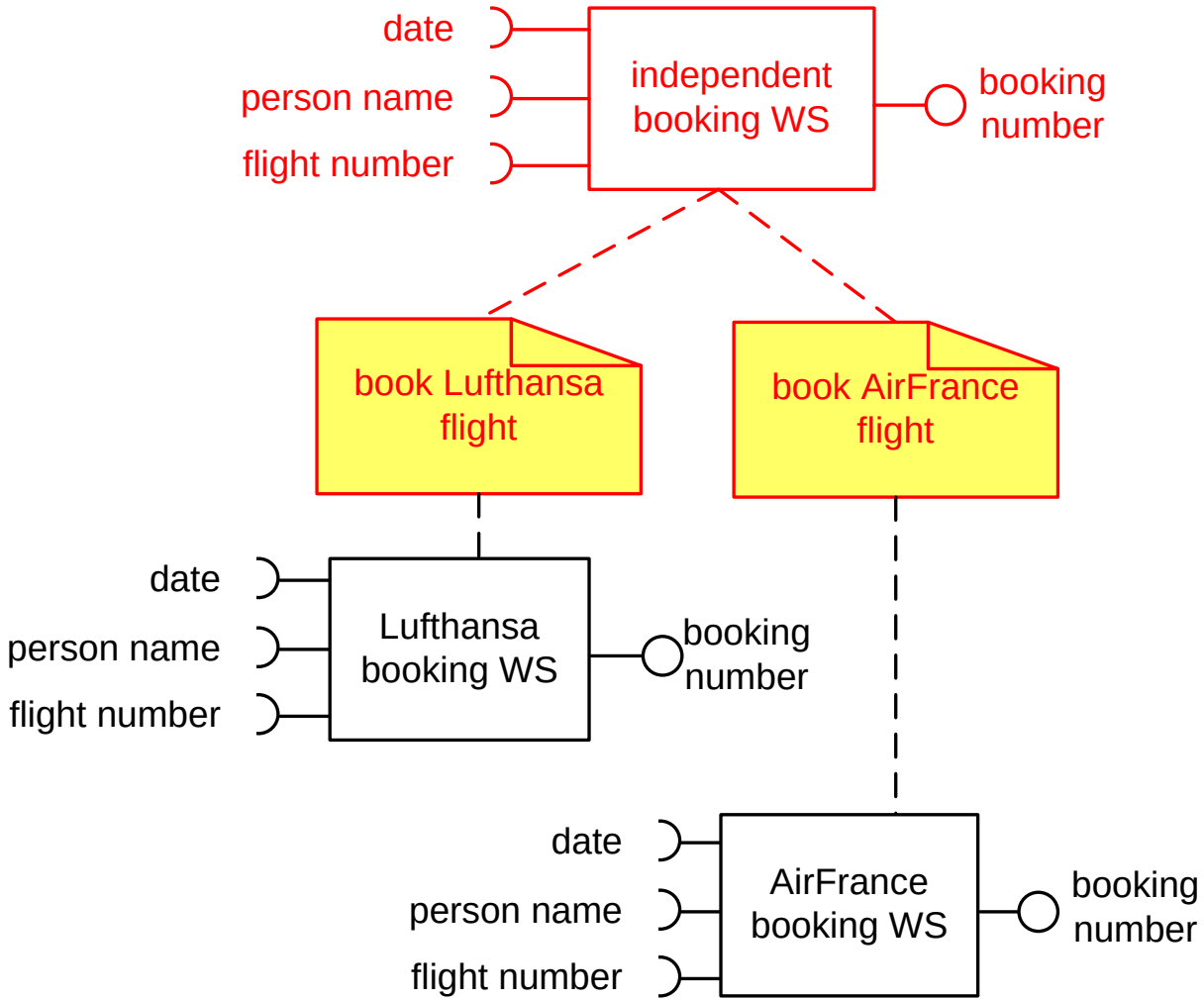
Allow…

- Generic process definition at run time
- Dynamic discovery of services
- Dynamic composition of services
  - Expressing functionality in terms of pre- and post-conditions
- n-to-m mappings between Services and Capabilities
  - Each Service can offer different capabilities
  - Each Capability can be provided by different Services

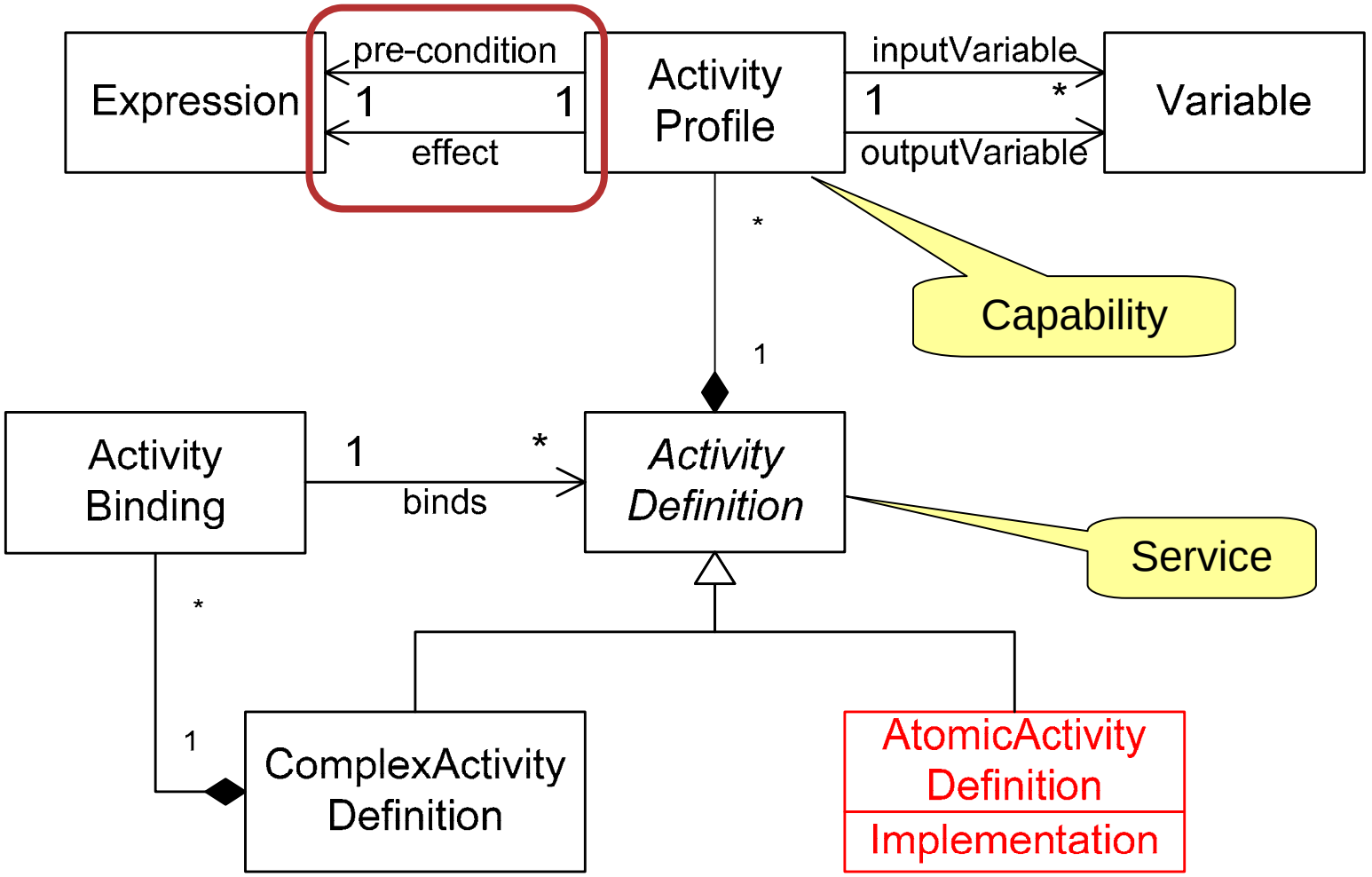
Services vs. Capabilities - example



Services vs. Capabilities (contd.)



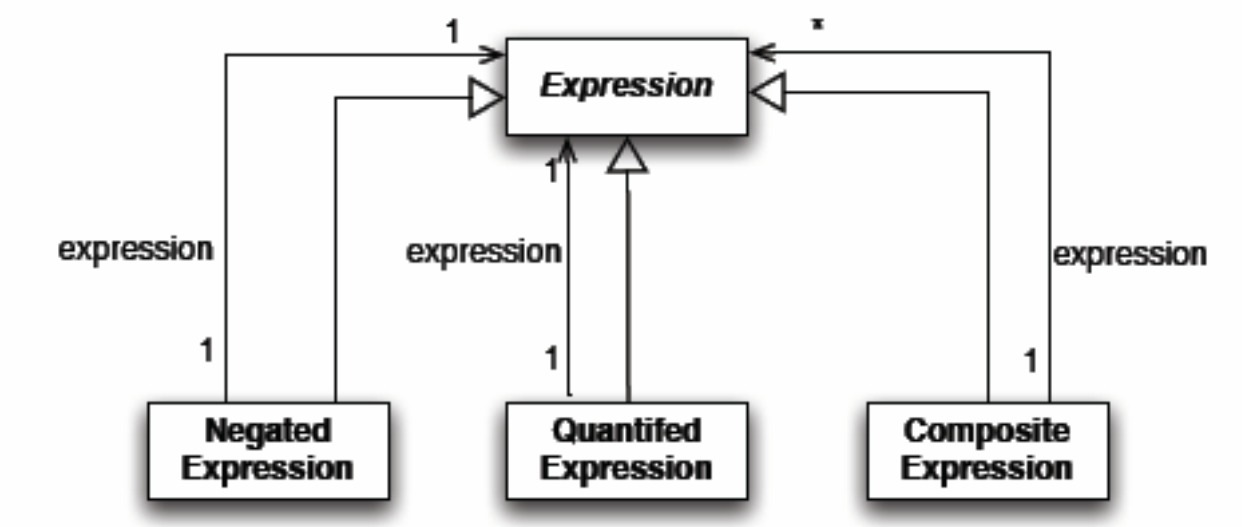
Planner: Meta-model



[H. Meyer: „Entwicklung und Realisierung einer Planungskomponente für die Komposition von Diensten“]

Semantic expressions

- Constructs allowed:
  - Negation: ¬
  - Quantifiers: ∃, ∀
  - Composition (conjuction and disjunction)



[H. Meyer: „Entwicklung und Realisierung einer Planungskomponente für die Komposition von Diensten“]

Semantic expressions - examples

- logical (comparable to predicate logic)

  $P1(A) \wedge \neg P2(B)$

  $(P1(A) \vee P1(B)) \wedge P3(A,C)$

  - objects as parameters allowed

- numerical

  $P1(A) + P1(B) \leq 100$

  - constraints regarding execution time, CPU time, memory, cost or other limitations expressible in numerical format (QoS)

Process composer (Planner)

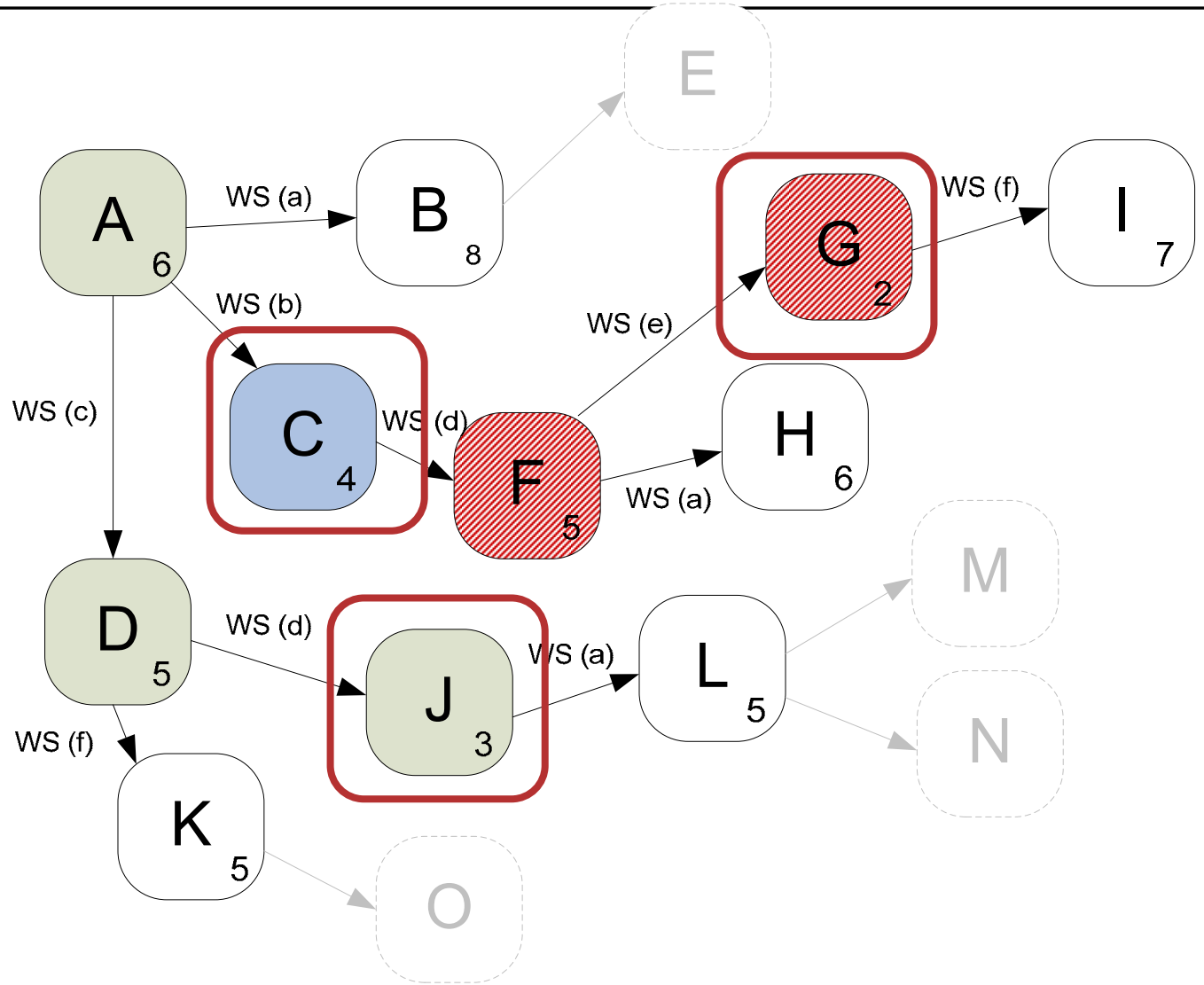
- uses a planning algorithm to determine a execution plan which fulfils a given goal
  - plan is developped by comparing pre-conditions and effects of available Web Services
    - uses a heuristic to determine the “distance” to the goal
  - plan is a partial order of a set of activities
    - activities = elements of composition (services)
    - links between activities describe dependencies regarding the execution order
    - activities not linked together are in general parallel executable

```
<flow> … </flow>
```

Planning algorithms

- Planning as heuristic search
  - Hill-Climbing
    - fails at local maximum (distance is minimal; heuristic maximal)
  - Enforced-Hill-Climbing
    - fails in falsely chosen branches
  - Best-First-Search
    - does not always terminate (endless loops)
  - A*-Search
    - complete
    - in combination with appropriate heuristic: optimal paths

Planning algorithms



Planning algorithms (contd.)

- Hierarchical-Task-Network Planning (HTN)
  - Iterative replacement of high-level complex activities by „mini“-processes containing lower-level complex activities and/or atomic activities
    - goal: only atomic activities left ➔ plan finished
  - bottom-up composition of new high-level services not possible
- Planning as Model-Checking
  - problem viewed as state diagram
  - states are modeled by temporal dependencies
    - goal is also a state

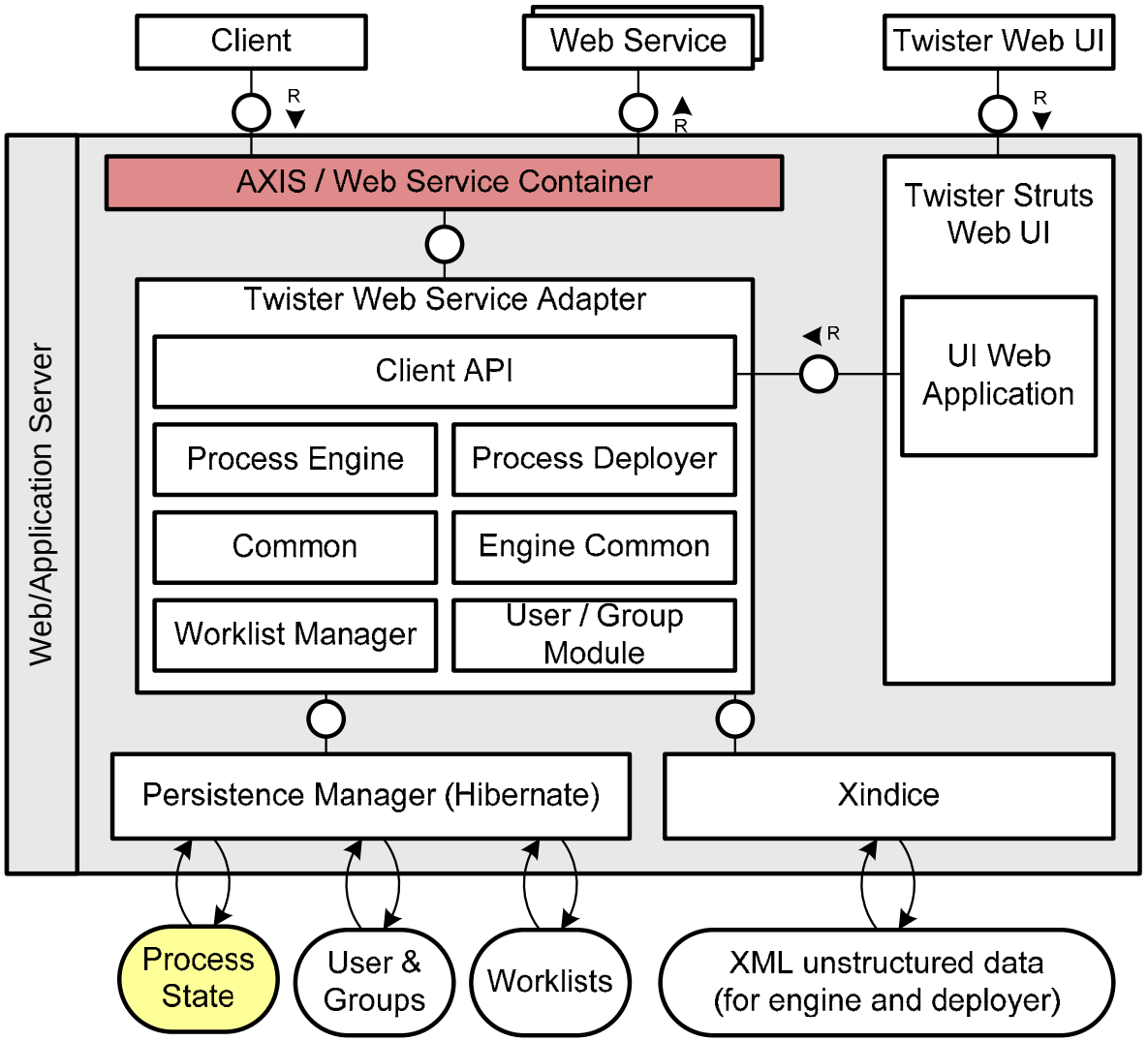
Workflow engines with BPEL support

several free & open-source workflow engines available

- http://java-source.net/open-source/workflow-engines
- http://www.manageability.org/blog/stuff/workflow_in_java
- mostly written in Java, many proprietary process definition formats

- Apache Twister (http://www.smartcomps.org/twister/)
  - version 0.3
- ActiveBPEL (www.activebpel.org)
  - version 1.1.2
- Codehouse Werkflow (werkflow.codehouse.org)
  - currently completely rewritten
  - new version coming soon (hopefully)

Twister - architecture



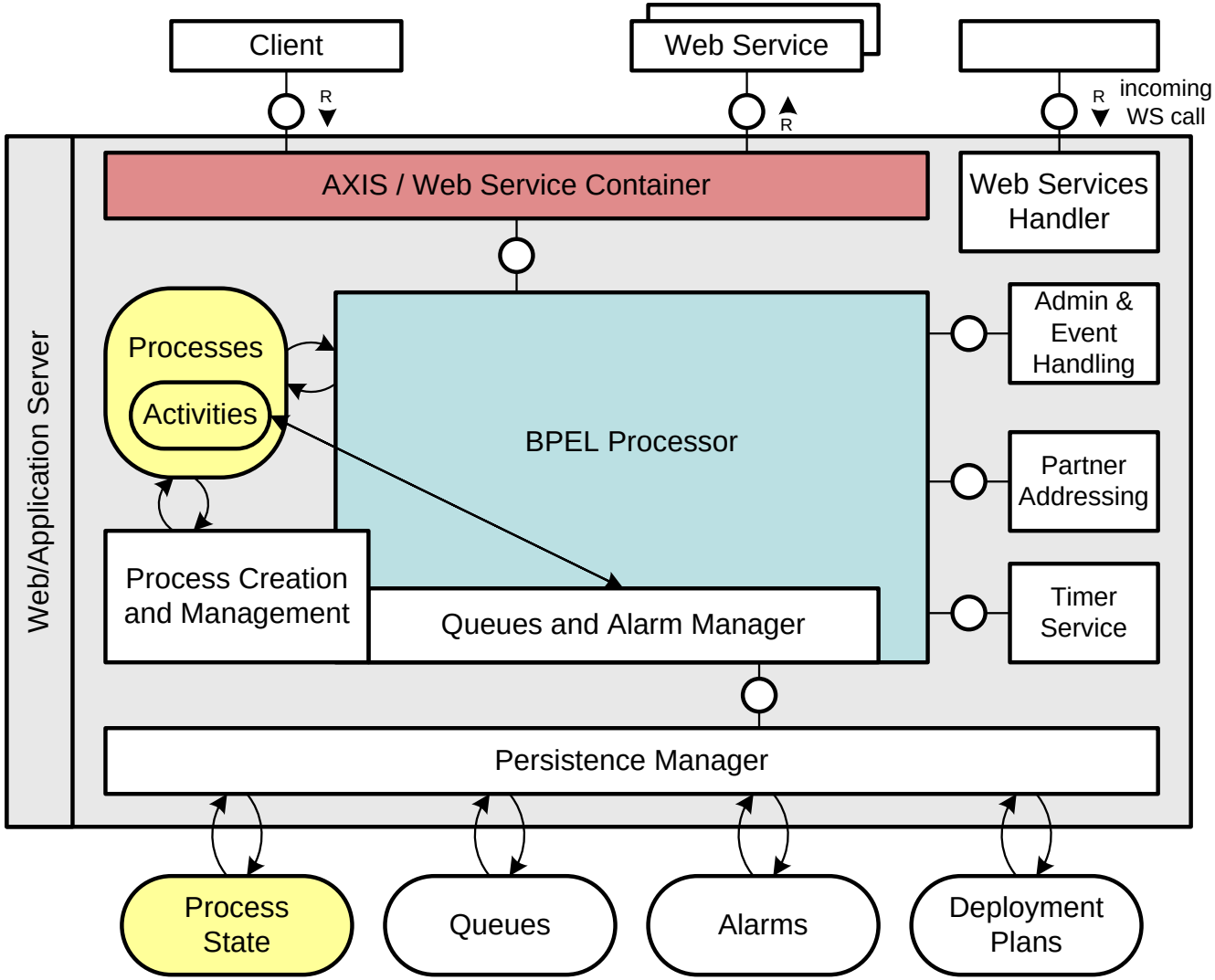
Twister – functional modules

- Process Engine
  - executes processes according to received messages & produces messages
- Process Deployer
  - parses and validates Web Service descriptions and process definitions
- User/Role/Group module
- Worklist Manager
- Engine Common
- Common

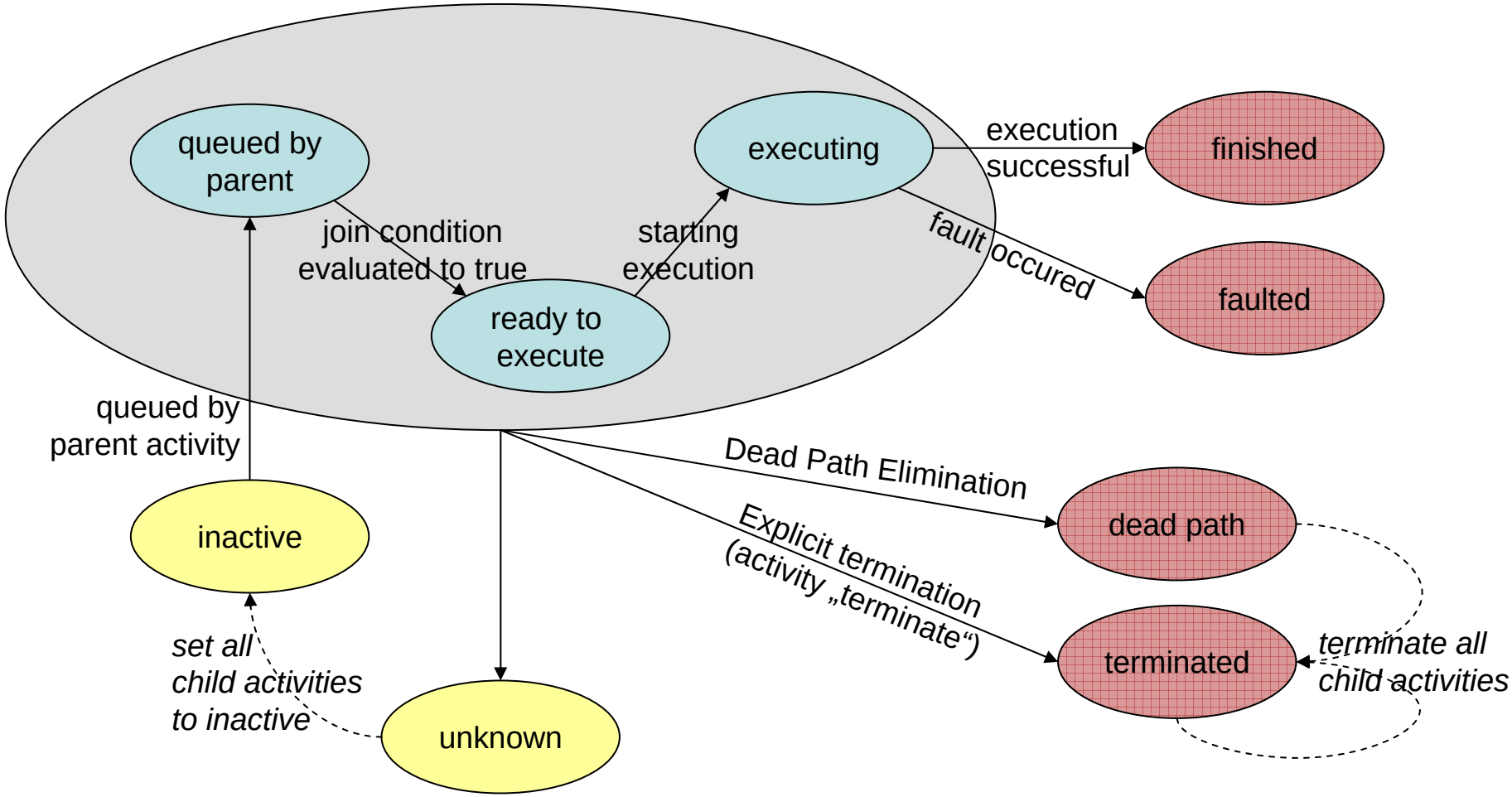
Twister – technological background

- implemented in Java
- Process Engine, User & Worklist Manager need to persist their internal state
  - use Hibernate as persistence layer
- Process Engine & Process Deployer need to persist XML unstructured data
  - XML database Apache Xindice
- XML APIs required (Dom4J, Xerces, MSV)
- servlet container required (i.e. Tomcat)

ActiveBPEL - architecture

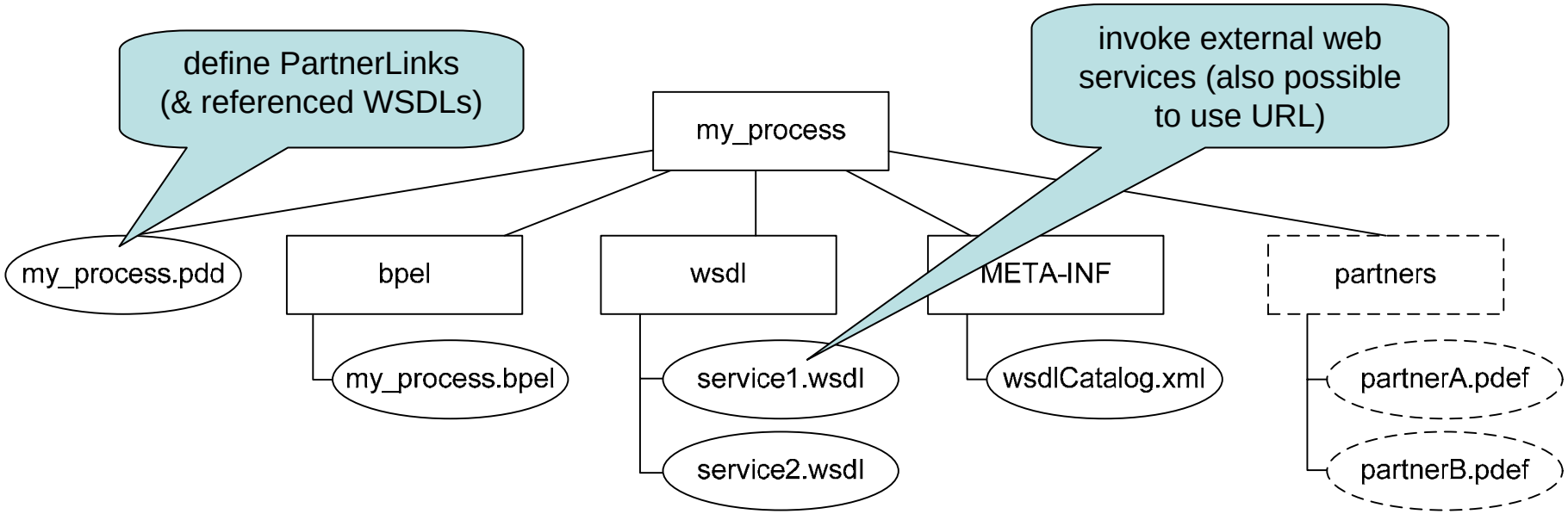


ActiveBPEL – process states



ActiveBPEL – deploying processes

- create deployment archive file
  - normal .jar archive
  - named .bpr
  - copy to subdirectory „bpr“ in Tomcat folder
  - following directory/file structure



Agenda

- Introduction
  - Workflow & Web Service Composition
  - Workflow Management Systems
- Automatic Web Service Composition
  - Process definition language: BPEL
  - Semantics (Semantic Web Services)
  - Process composer (Planner)
  - Workflow Engine
- Summary

Summary

- BPEL covers all necessary workflow patterns
  - output of composition: executable process
  - provides no adequate semantic to describe WS
- Planner provides meta-model for modeling capabilities
  - need to describe all Services in this language
  - need to transfer into BPEL process definition

  ➔ better way using standardized language ?
- ActiveBPEL is currently the best available free workflow engine that supports BPEL

Outlook

- Component to convert meta-model into executable BPEL processes must be developed
  - Harald Meyer currently working on
- Evaluation of standardized language for semantic descriptions
  - OWL-S (Ontology Web Language)
  - abstract BPEL processes with semantic extensions ?
- How can replanning be implemented with ActiveBPEL?
  - if plan fails, planner has to „replan“ starting from the last successful point

References: WfMS, BPEL

- T. Andrews et al.: BPEL4WS, version 1.1 (May 2003)
- D. König (IBM, OASIS): WS-BPEL. Slides (May 2005)
- P. Fornasier, P. Kowalski: bexee – BPEL execution engine. Slides (Jan. 2005)
- E. Owen (DERI): Web Service Composition – workflow patterns in BPEL (June 2004)
- C. Schittko: Web Service Orchestration with BPEL. Slides (Nov. 2003)
- M. Weske: WfMS. Lecture slides (2003/04)
- ActiveBPEL (2005) www.activebpel.org:
  - ActiveBPEL Engine Architecture
  - Installing and Configuring the ActiveBPEL Engine
  - Deploying BPEL Processes
  - ActiveBPEL Engine File Formats
- Twister (2004/05) www.smartcomps.org/twister/:
  - architecture guide
  - developer guide
  - installation guide
  - user guide
  - WS-BPEL guide

References: SWS & WS composition

- A. Ankolekar et al.: DAML-S: Web Service Description for the Semantic Web (2002)
- D. Fensel: The Web Service Modelling Framework WSMF (2001)
- R. Lara et al.: Semantic Web Services: description requirements and current technologies (2003)
- H. Meyer: Entwicklung und Realisierung einer Planungskomponente für die Komposition von Diensten – diploma thesis (2005)
- The OWL Services Coalition: OWL-S 1.0 (2003)
  - http://www.daml.org/services/owl-s/1.0/
- The OWL Services Coalition: OWL-S: Semantic Markup for Web Services (2004)
- M. Pistore et al.: Planning and Monitoring Web Service Composition (2004)
- M. Pistore et al.: WS-Gen: A Tool for the Automated Composition of Semantic Web Services (2004)
- M. Sabou et al.: An experience report on using DAML-S (2003)
- E. Sirin: Interactive Composition of Semantic Web Services (2005)
- E. Sirin et al.: Semi-automatic composition of Web Services using Semantic Descriptions (2003)
- P. Traverso, M. Pistore: Automated Composition of Semantic Web Services into Executable Processes (2004)
- L. Zeng et al.: Flexible Composition of Enterprise Web Services (2002)
- Masterprojekt Plaengine: WfMS für die integrierte Prozessplanung und –ausführung
  - www.plaengine.com
- Web Services Description Language (WSDL)
  - http://www.w3.org/2002/ws/desc/