15. Web Services and Service-Oriented Architectures

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
http://st.inf.tu-dresden.de
Version 13-1.1, May 31, 2013

1) Web Services as a specific form of service-oriented architectures
2) SOAP
3) WSDL
4) BPEL
5) BPMN
6) Trust and security
7) Evaluation

Obligatory Reading

- ISC, Chapter 2.4
- Lohmann, Niels, Verbeek, Eric, Dijkman, Remco. Petri Net Transformations for Business Processes – A Survey. In : Transactions on Petri Nets and Other Models of Concurrency II, Editor: Jensen, Kurt, van der Aalst, Wil, Lecture Notes in Computer Science 5460, 2009, Springer Berlin / Heidelberg
  - http://www.springerlink.com/content/n7464131r6751453/
- W.M.P. Van der Aalst. Don't go with the flow: Web services composition standards exposed. IEEE Intelligent Systems, Jan/Feb 2003. http://tmitwww.tm.tue.nl/research/patterns/download/ieeewebflow.pdf
- P. Wohed, W.M.P. Van der Aalst, M. Dumas, A. ter Hofstede. Analysis of Web Service Composition Languages: The Case of BPEL.
- http://www.bpmnforum.com/FAQ.htm FAQ of BPMN

Literature

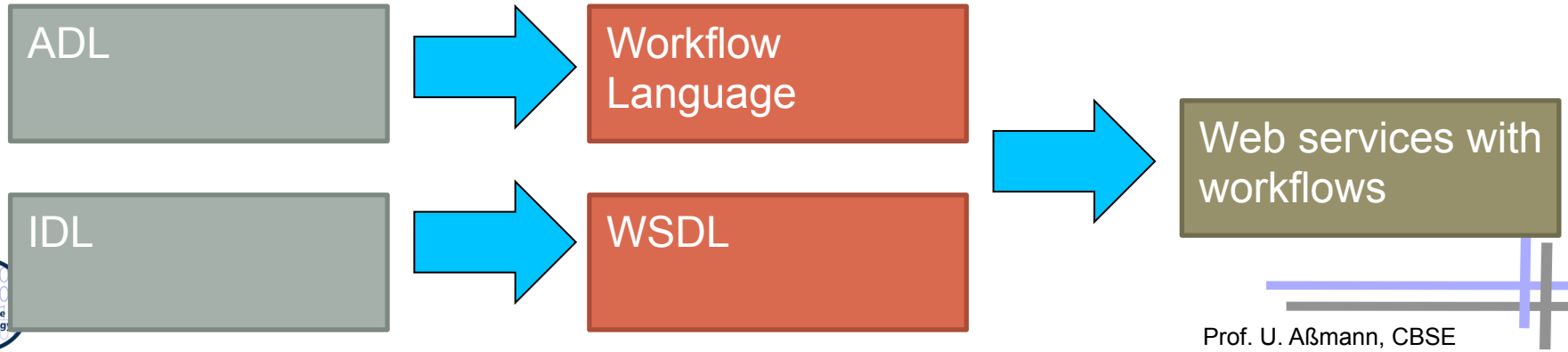
- Matthias Weske. Business Process Management – Concepts, Languages, Architectures. Springer. 2007
- YAWL http://sourceforge.net/projects/yawl/
- H. P. Alesso, C. F. Smith. Developing Semantic Web Services. A K Peters Ltd, Natick, Massachusetts, 2004.
- http://www.bpmb.de/index.php/BPMNPoster
- Liste der BPMN Werkzeug-Hersteller http://www.bpmn.org/BPMN_Supporters.htm
- Scheer, A.-W. ARIS - Business Process Frameworks. Springer, Berlin, 1998, ISBN 3-540-64439-3
- Michael C. Jaeger. Modelling of Service Compositions: Relations to Business Process and Workflow Modelling. ICSOC 2007, LNCS 4652.

The Ladder of Composition Systems

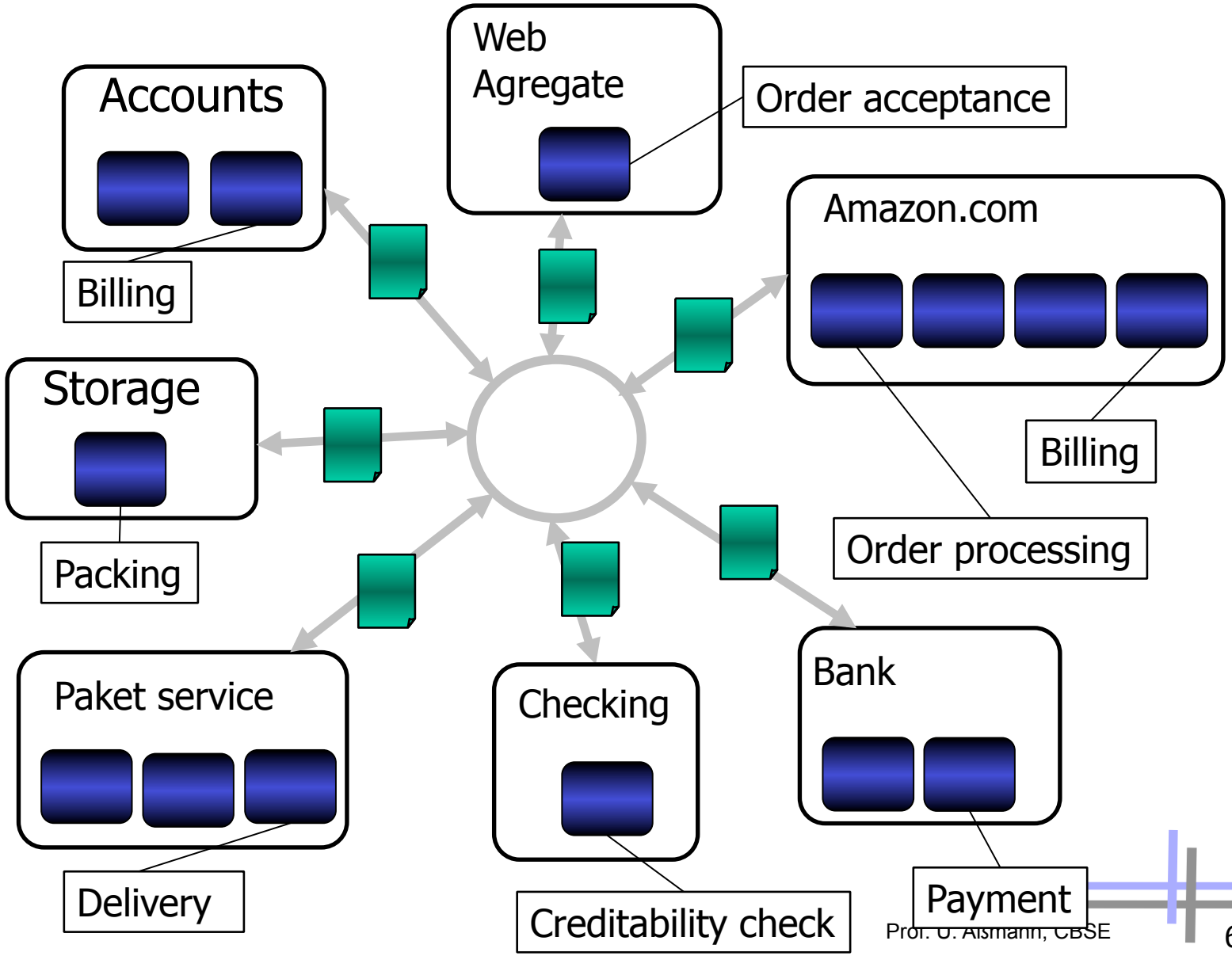
Software Composition Systems	Composition Language	*Invasive Composition* *Piccola Gloo*
Aspect Systems	Aspect Separation Crosscutting	*Aspect/J* *AOM*
View Systems	Composition Operators	*Composition Filters* *Hyperspaces*
Architecture Systems	Architecture as Aspect Connectors	*Darwin* *BPMN HRC*
Classical Component Systems	Standard Components Reflection	*.NET CORBA* *Beans EJB* *ArchJava*
Object-Oriented Systems	Objects as Run-Time Components	*C++ Java* *UML components*
Modular Systems	Modules as Compile-Time Components	*Shell scripts* *Modula Ada-85*

15.1 Web Services and Architecture Systems

- Architecture systems may have different forms of architectural languages:
  - Topology-based (Unicon, ACME, Darwin)
  - Coordination schemes (CoSy)
  - Imperative scripts (Darwin)
- Web Service Systems and Languages (WSS) are a form of architectural system
  - They separate programming-in-the-small from programming-in-the-large (2-level programming)
    - Components encapsulate the service knowledge
    - The architectural level (orchestration, aggregation, composition) treats the big picture
- However, WSS have an imperative architectural language
  - They are based on XML standards (SOAP, WSDL, BPEL)

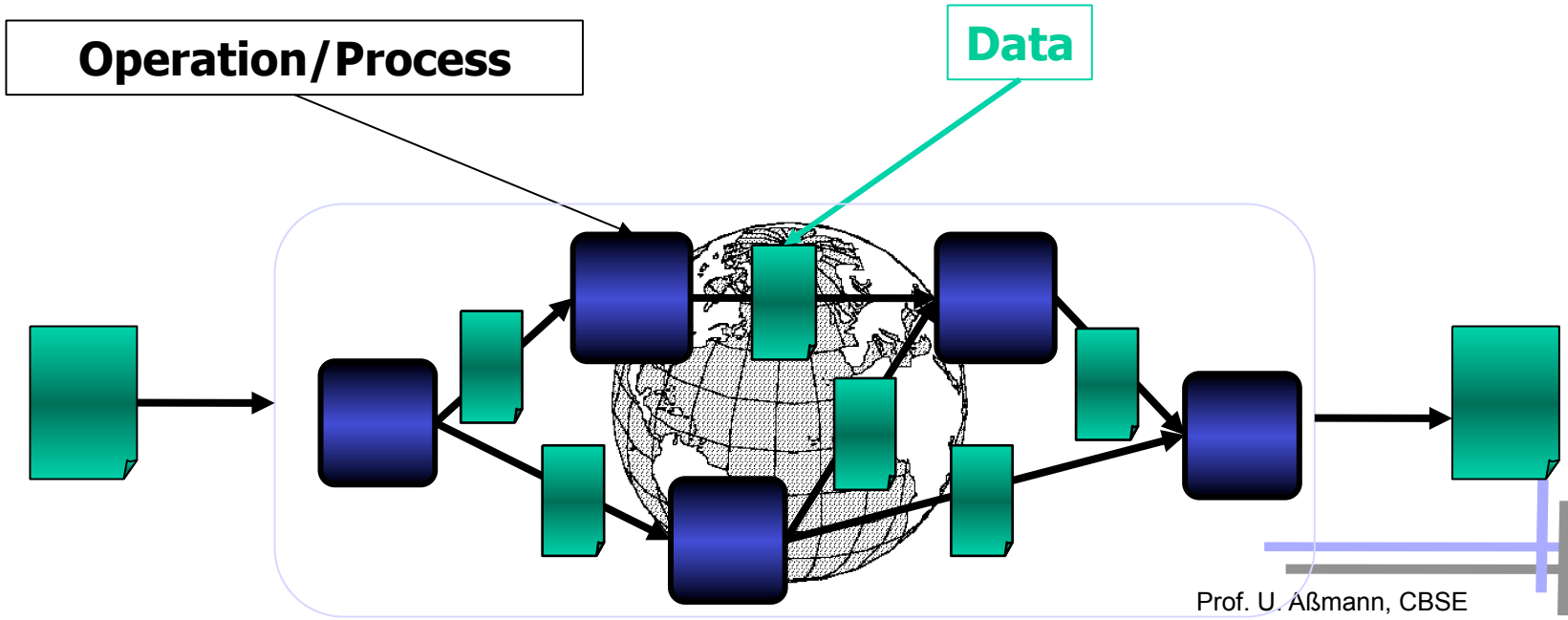


Reuse of Web Services as Black-Box Components



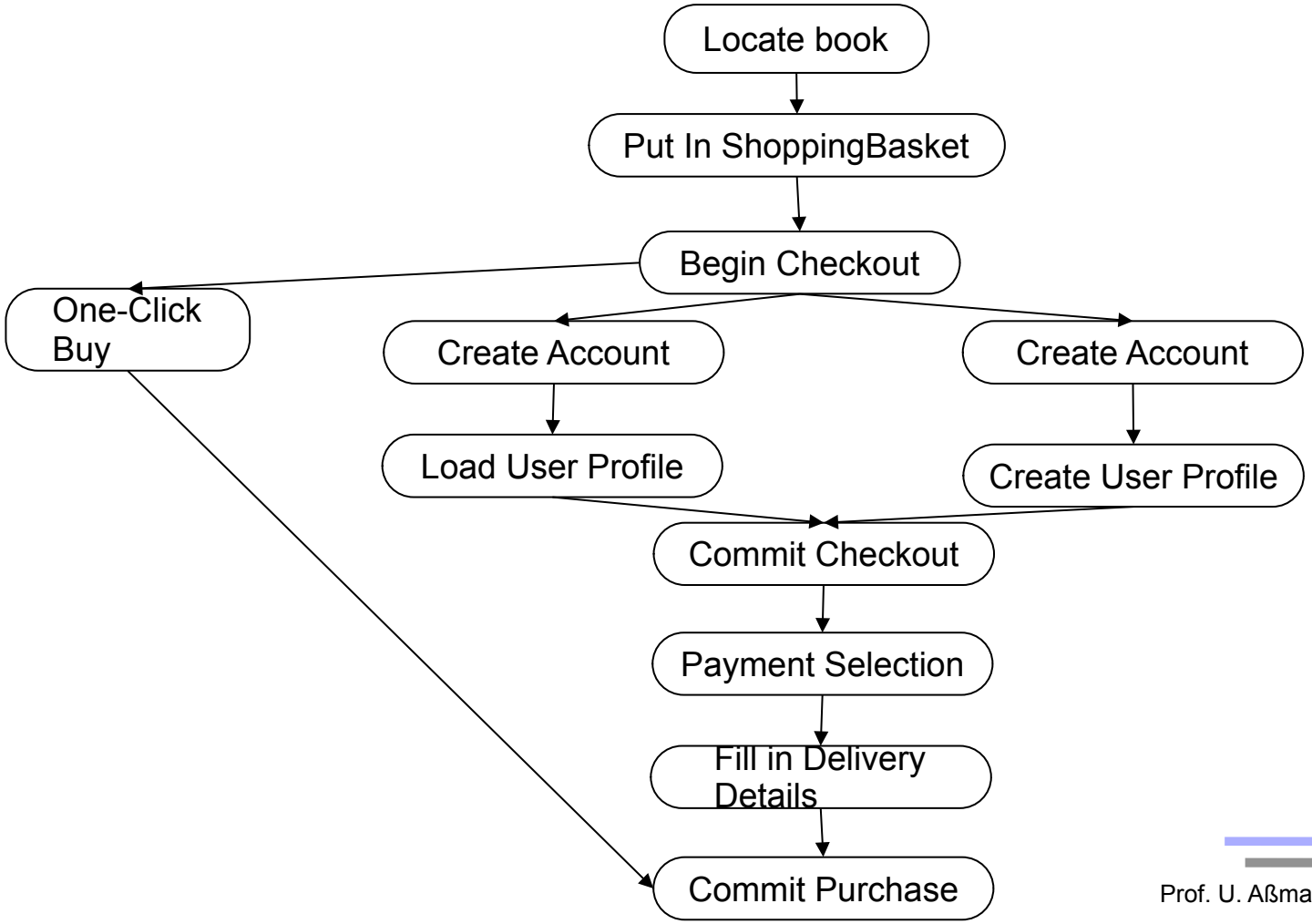
Web Service Architectures are Described by Workflows

- Workflow specifications combine control and data flow
- Web service architectures are the first step to service-oriented architectures (SOA), based on traders
  - Services will be offered, searched and discovered, downloaded, executed
- *Enterprise services* transfers web services to business systems
- *Customer services* serve the end-user of the web



Ex. Buying a Book from Amazon

- Workflows can be specified graphically
  - E.g., with UML activity diagram [Alesso/Smith]



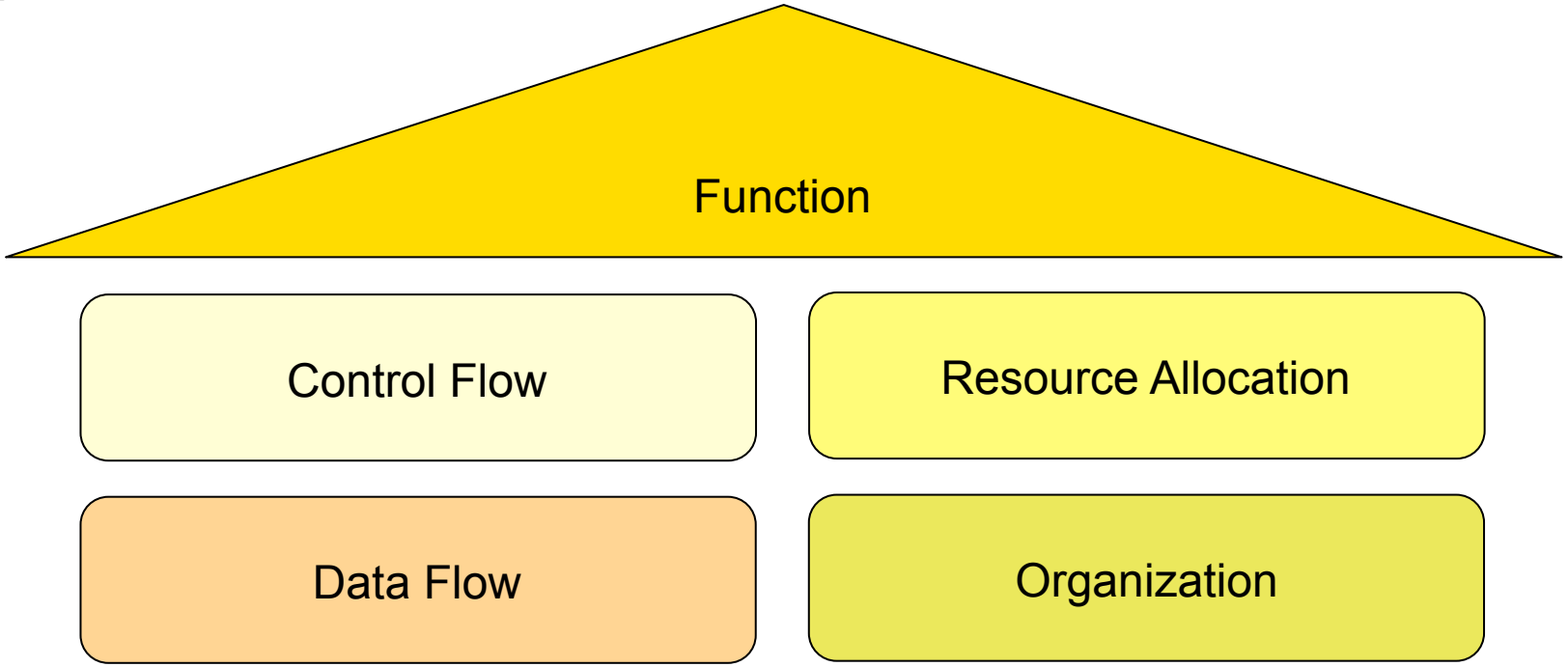
Which Types of Operational Specifications Exist?

- Data-flow graphs (data flow diagrams, DFD)
  - Data flows through operations
  - Activity diagrams: data flows through actions
  - See courses Softwaretechnologie II, Software-Entwicklungswerkzeuge
- Control-flow graphs (CFG)
  - Nodes are control-flow operations that start other operations on a state
  - The standard representation for imperative programs
- State systems
  - Finite State Machines (FSM): events trigger state transitions
  - Statecharts: Hierarchical FSM
  - Colored Petri nets: tokens mark control and data-flow, see course Softwaretechnologie II
- Mixed approaches
  - Cyclic data-flow graphs (also called static-single assignment graphs, SSA)
    - Cycles are marked by phi-nodes that contain control-flow guards
  - **Workflow languages**: mix control and data-flow
    - Provide specific split and join operators for control and data flow

Workflows Have Aspects

Standard workflow modeling discerns about 5 aspects

ex. ARIS house [Scheer's company IDS, now Software AG]

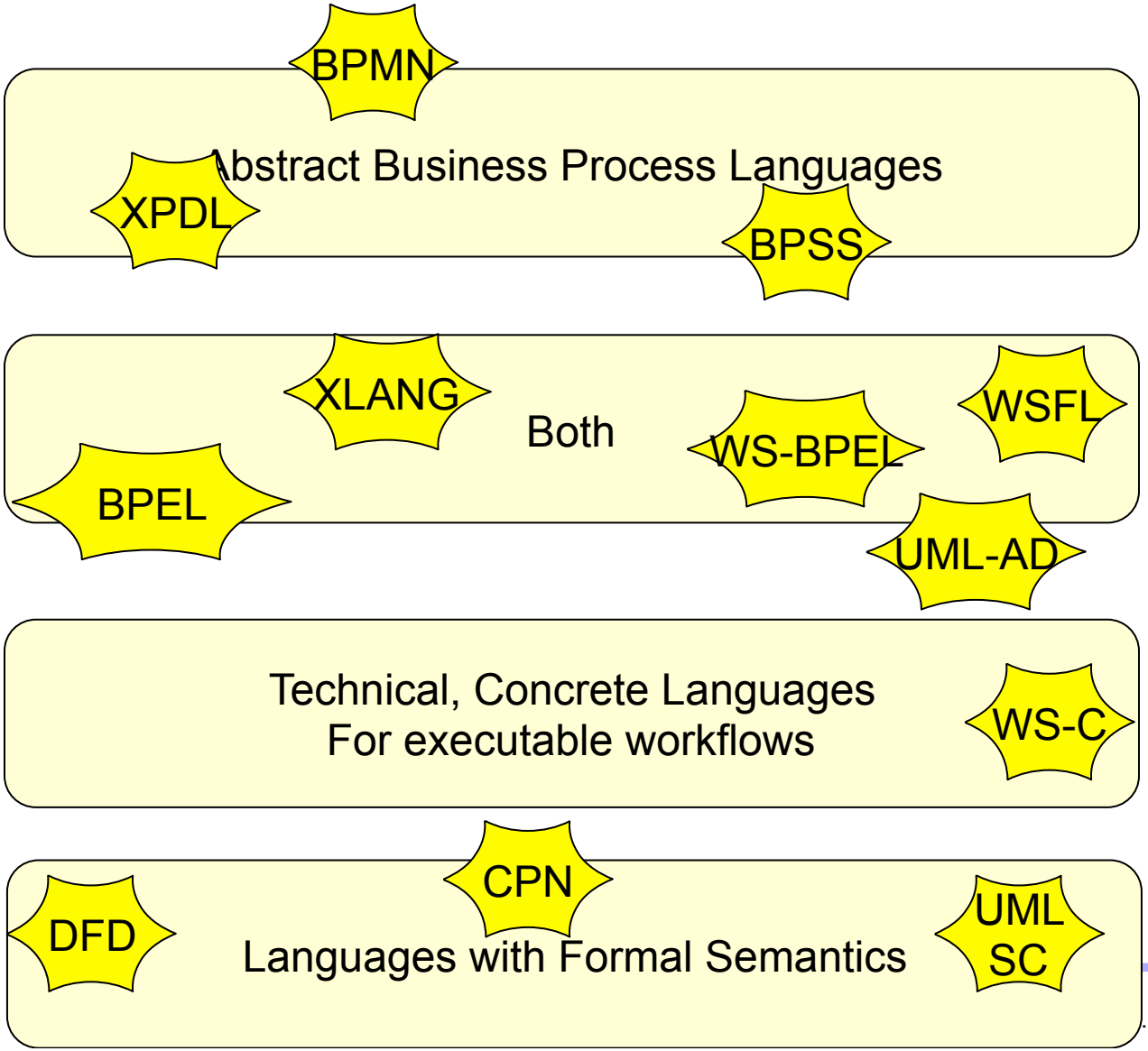


[Scheer]

Workflow Languages

- A workflow language specifies control and data flow over a set of operations
  - The workflow is executable with an interpreter, the *workflow engine*
  - A single operation need not be executed automatically, but can be performed by humans (… for people)
  - The workflow runs in parallel
- Workflows are usually compiled to Colored Petri Nets or to Statecharts
  - YAWL (van der Aalst, Einhoven)
  - Workflow Nets
- Industrial Examples:
  - Lotus Domino (IBM)
  - Business Process Execution Language (BPEL)
  - ARIS system for SAP, based on EPC (event process chains)
  - Business Process Modeling Notation (BPMN), also in use at SAP

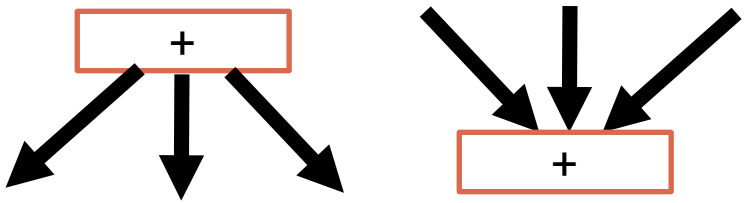
Web Service Languages Serve Different Abstraction Levels



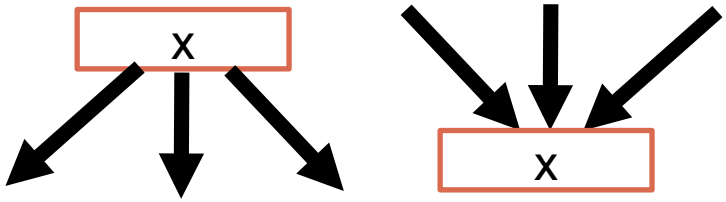
[Jaeger 2007]

Typical Control-Flow Operators in Workflow Languages (Gateways)

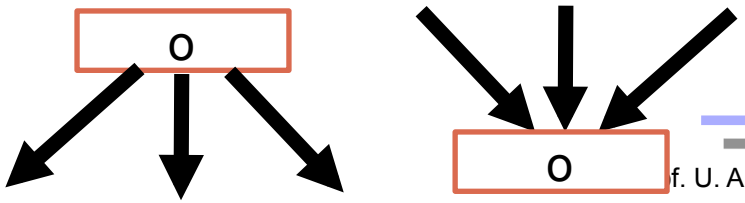
- Control-flow operators in Workflow languages are more complex than simple transitions in Petri Nets, which support only AND-split and -join
- AND-split: all
- AND-join: all of n



- XOR-split: 1 of n
- XOR-join: 1 of n



- OR-split: m of n
- OR-join: m of n

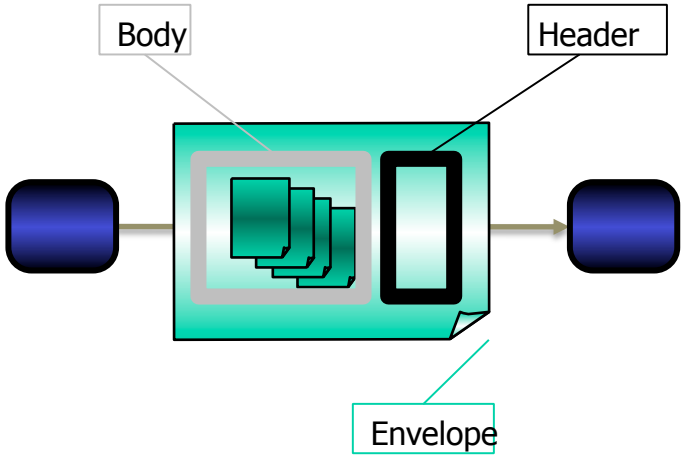


What are Workflow Engines?

- Workflow engines are interpreters of workflows
  - They maintain the parallelism in a workflow and synchronize all processes
- Usually, they also support for interactive applications
  - Undo
  - Transactions with rollback and commit
  - Compensation (in case of error)
- They are, for web services and component systems, *composition engines* that execute a composition program, the workflow

15.3 SOAP, An XML-based Interaction Protocol

- Simple Object Access Protocol (SOAP) defines the message format
- Message contains target address and an envelope
  - with name space, encoding attributes and
  - Header (fixed format) contains
    - Authentication (Sender, Receiver),
    - Transactions,
    - Error handling information,
    - Routing information …
  - Body contains user data (free format)
- Transport is transparent, predefined channels:
  - HTTP (with back channel, de facto standard)
  - SMTP, TCP (with back channel)



Example: SOAP Header

Message Header
HTTP

```
POST /TreatmentAdmin HTTP/1.1
HOST: www.hospital-admin.com
Content-Type: text/xml
Charset="utf-8"
Content-Length: nnnn
SOAPaction: http://localhost/TreatmentAdmin

<SOAP-ENV:Envelop
   xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelop/
   SOAP-ENV:Encoding="http://…/encoding">
   <SOAP-ENV:Header>
        <a:Authentication
                xmlns:a=http://localhost/TreatmentAdmin … >
            …
        </a:Authentication>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body> … </SOAP-ENV:Body>
</SOAP-ENV:Envelop>
```

SOAP Envelop

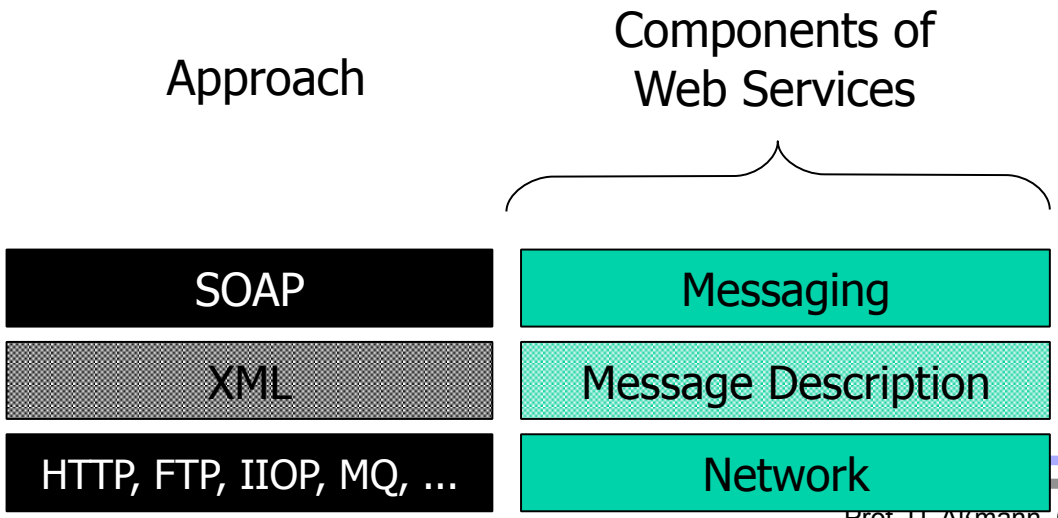
Example: SOAP Body

```
<SOAP-ENV:Body>

<m:AddTreatment xmlns:a=http://localhost/TreatmentAdmin>

<treatment>
  <patient insurer="1577500"nr='0503760072'/>
  <doctor  city   ="HD"     nr='4321'/>

  <service>
    <mkey>1234-A</mkey>
    <date>2001-01-30</date>
    <diagnosis>No complications.
    </diagnosis>
  </service>

</treatment>

</SOAP-ENV:Body>
```

SOAP Interaction Protocol

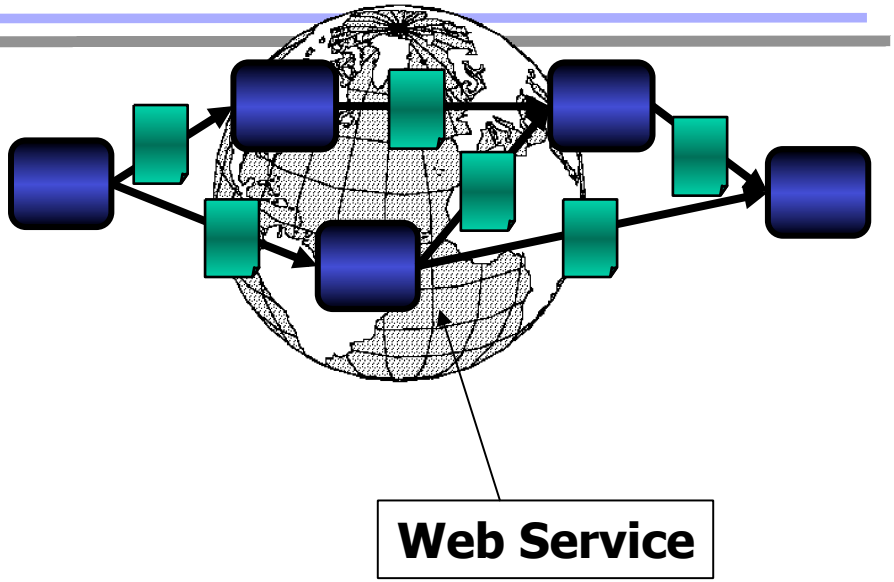
+ W3C Recommendation (standard)
+ Implements RPC

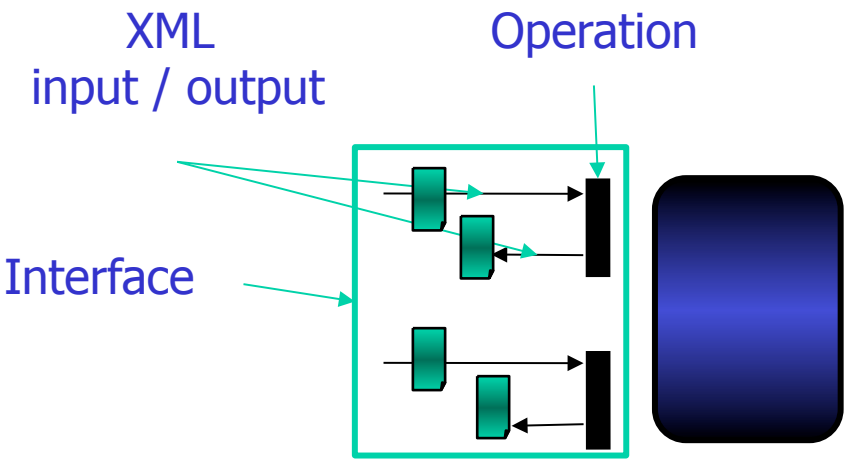
- Untyped user data, types to encode in the message
- Interpretation of SOAP messages required
- High overhead / low performance



15.4 WSDL and The Interface Concept of Web Services

Service Interface

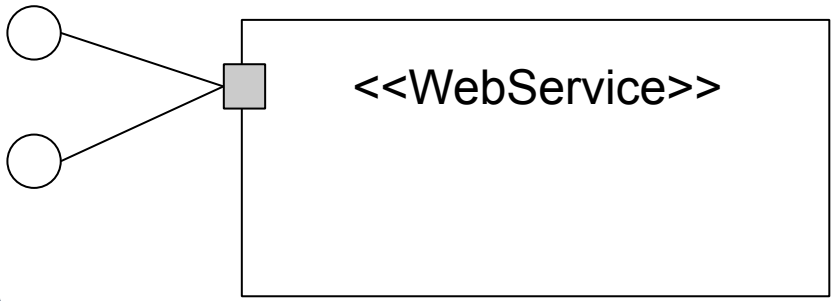




Web Services Description Language (WSDL) defines a service interface

WSDL Components and Their Interfaces

- A WSDL *Interface* consists of a set of ports
  - Functions with types of parameter and results in XML Schema
  - Event ports
  - Plays a similar role as ports of a UML component
- Advantages
  - WSDL abstracts from underlying protocol (http, SOAP, mime, IIOP)
  - Component model can be mapped to CORBA, EJB, DCOM, .NET
  - WSDL unifies call and event ports
  - WSDL abstracts from the underlying component model, introducing the component model as a *secret*



WDSL Specification Structure

- Types
  - In XML schema or another typing language
- Messages
  - The data that is communicated
- Operation
  - An interface of the service, with input and output, fault parameters
- Port type
  - A named set of operations (as in UML)
- Binding
  - A mapping of the port to underlying component models, e.g., http, soap, or mime
- Service
  - A set of related ports

WSDL Reuses Data Types of XSD
Here: Type Definitions <schema> <element> <complextype>

```
<wsdl:types>
   <XMLSchema:schema … [target name space definitions]>
      <XMLSchema:element name="addTreatment">
               <XMLSchema:complextype>
                 <XMLSchema:sequence>
                   <s:element minOccurs="1" maxOccurs="1" name="parameter"
                                nillable="true" type="a:treatment"/>
                 </XMLSchema:sequence>
               </XMLSchema:complextype>
      </XMLSchema:element>
      <XMLSchema:element name="addTreatmentResponse">
               <XMLSchema:complextype>
                 <XMLSchema:sequence>
                   <s:element minOccurs="1" maxOccurs="1" name="result"
                                nillable="true" type="XMLSchema:bool"/>
                 </XMLSchema:sequence>
               </XMLSchema:complextype>
      </XMLSchema:element>
      <XMLSchema:complextype name='treatment'> …
               </XMLSchema:complextype>
   </XMLSchema:schema>
<wsdl:types>
```

Port Types

- Event or message based:
  - Notification: data-out port
  - One-way: data-in port
- Call-based
  - Request-Response: procedure port
  - Solicit-Response: send, then receive (caller port)

Example: WSDL Ports, Typed by Message Types

```
<wsdl:definitions [name space definitions]>
     <wsdl:types> … </wsdl:types>
     <wsdl:message name="addTreatmentSOAPIn">
             <part name="parameters" element="addTreatment"/>
   </wsdl:message>
   <wsdl:message name="addTreatmentSOAPOut">
     <part name="parameters" element="addTreatmentResponse"/>
  </wsdl:message>

  <wsdl:porttype name="TreatmentAdminSOAP">
    <wsdl:operation name="addTreatment">
      <wsdl:input  message="addTreatmentSoapIn
      <wsdl:output message="addTreatmentSoapOut"/>
    </wsdl:operation>
  </wsdl:porttype>

  <binding [binding to SOAP / HTTP Protocols] …
</wsdl:definitions>
```

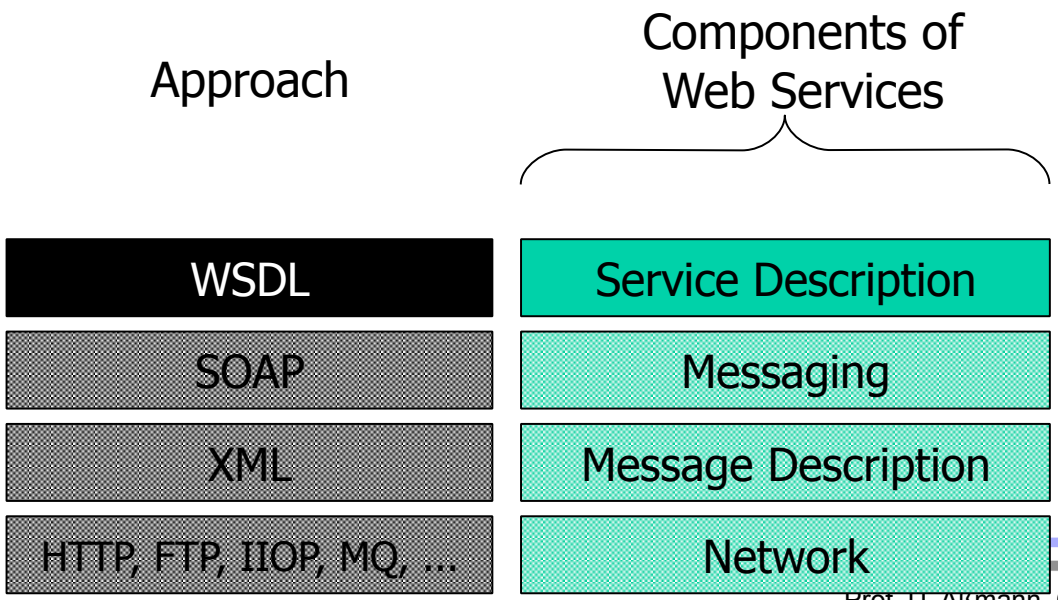
Actual interface

Example: Binding WSDL to SOAP

```
<wsdl:binding name="livetoken" type="Token">
     <soap:binding style="document"
                   transport="http://schemas.xmlsoap.org/html">
     <operation name="GetLastPrice">
            <soap:operation
               soapAction="http://www.stocktrade.com/GetPrice">
            <input> <soap:body use="literal"> </input>
            <output> <soap:body use="literal"> </output>
     </operation>
   <wsdl:binding>
```

WSDL Service Interface

- WSDL is a Interface Definition Language (IDL)
  - Part of BPEL (see later)
  - No inheritance on WSDL
  - No standard mapping to data in programming languages
  - No web service as parameters/results
- W3C Recommendation (standard)



Trading on the Web: Offer and Find Services

- Standardized publishing, advertisement …
- Extended name server, describing interface and properties
- XML Descriptor
  - White Page: Address
  - Yellow Page: Semantics (based on standard taxonomy)
  - Green Page: Technical specification of service
- Logically central, physically distributed data base

⇨ Universal Description, Discovery and Integration (UDDI) defines service properties for service trading

UDDI

White Page

Registered (and other) names
Service Description
Contact person (name, e-mails, …)
Telephone/fax number
Web site
…

Yellow Pages

Service category
Type of industry
Type of products/services
Geographic localization
…

Green Pages

Offered service
Documentation, description
Principles cooperation realization
…

WSDL

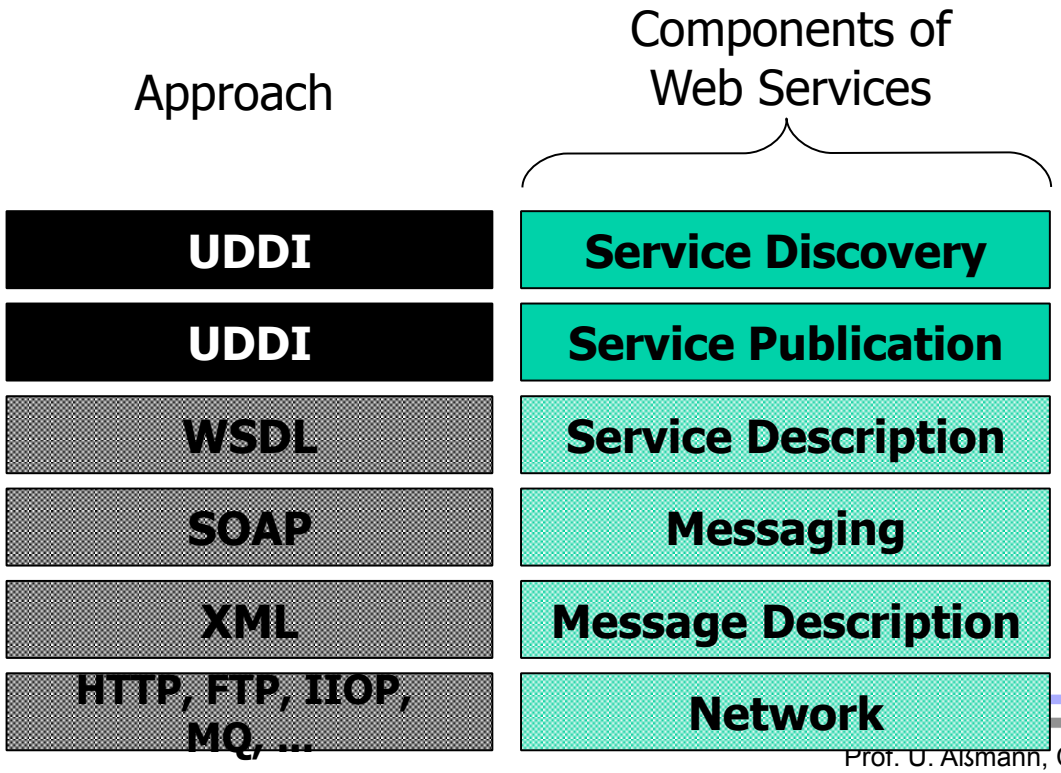
WSDL

UDDI: Offer and Find Services

-Required and approached

- No search strategies

- No no trader or market place



15.5 Business Process Execution and Web Service Workflows with BPEL

BPEL, a web service composition language

What is a Business Process?

"A collection of related, structured activities--a chain of events--that produce a specific service or product for a particular customer or customers."
www.gao.gov/policy/itguide/glossary.htm

"A business process is a recipe for achieving a commercial result. Each business process has inputs, method and outputs. The inputs are a pre-requisite that must be in place before the method can be put into practice. When the method is applied to the inputs then certain outputs will be created."
en.wikipedia.org/wiki/Business_process

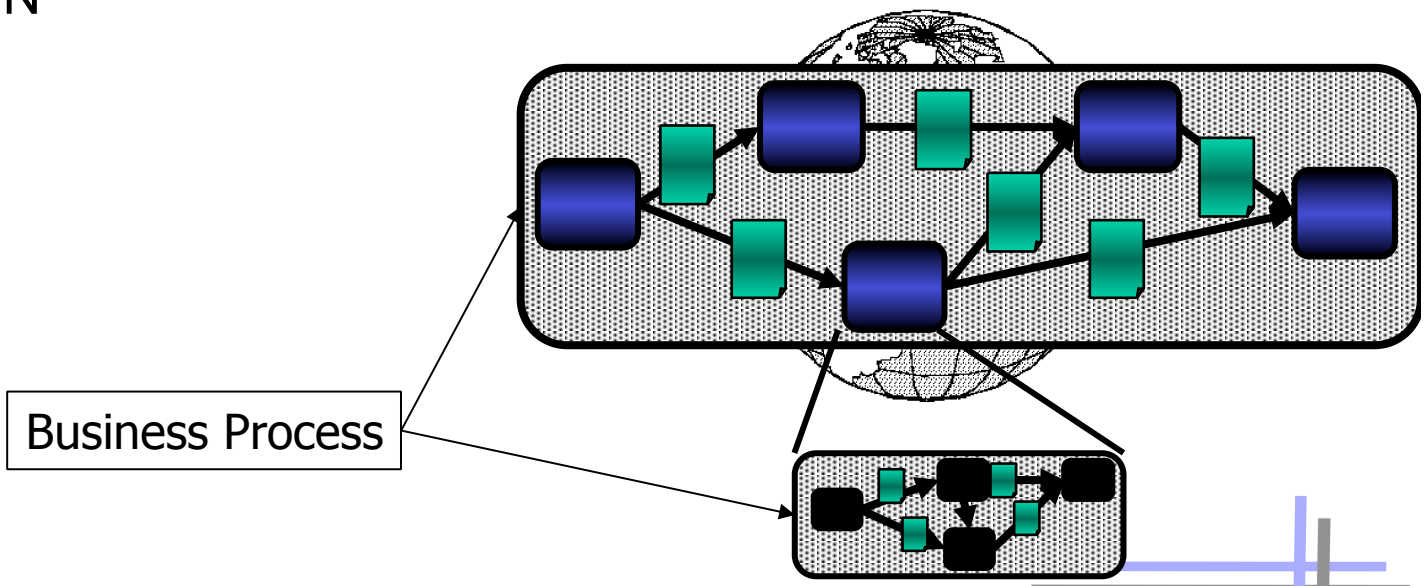
A business process is described on the modeling level, can be abstract, underspecified and need not be executable

A business process can be refined iteratively to become executable.

An executable business process is called a *workflow (executable business process).*

Business Process Definition

- How to define a business process on the web
- There are many languages proposed today:
  - WSFL, WSCL, WSCI, XLANG, WSEL, UML, WSUI, WSXL, BPML, BPMN …
- IBM & Microsoft: BPEL, BPEL4WS
- OASIS: WS BPEL
- W3C: OWL-S, SML (Service Modeling Language)
- SAP: BPMN



Ingredients of BPEL

- BPEL is an *executable language for workflows*, executable business processes
  - An architectural language for web services
  - Based on workflow languages
  - Mixing control and data flow operators
- BPEL is a composition language
  - Composing web services, using their ports
  - Relying on messages (events) and calls
- BPEL uses WSDL for service interface descriptions, as IDL
- BPEL adds connections (*partner link types*)

BPEL Made Simple

- BPEL is a activity-diagram like language,
  - with parallelism and transactions
  - with different kind of join and split operators
  - with ports and connections
  - BPEL can be edited graphically, and has an XML abstract syntax
- To create a web service, becomes a similar activity as editing an UML activity diagram or Petri Net
- BPEL uses WSDL definitions to define types, message types, and port types
  - WSDL definitions can be without binding
    - Bindings can be added when the BPEL process is deployed
    - That increases reuse of the process
  - This achieves *component model transparency* (independence of the underlying component model)
  - *Partner link types* (connector types) describing typed connections

BPEL Specification Structure

- **Process definition**: Header with namespace declarations
- **Variables**: global variables of the process
- **PartnerLink declarations**: interface declaration
  - with whom is the process connected?
- **Partners**: actual partners of the communication
- **Correlation sets**: Which instance of a process is talking to which other instance?
- **Fault handler**: What happens in the case of an exception?
- **Compensation handler**: compensation actions
- **Event handler**: what happens in case of a certain event?
- A (structured) **main** operation
  - e.g., sequence or flow

A Simple Pizza Order

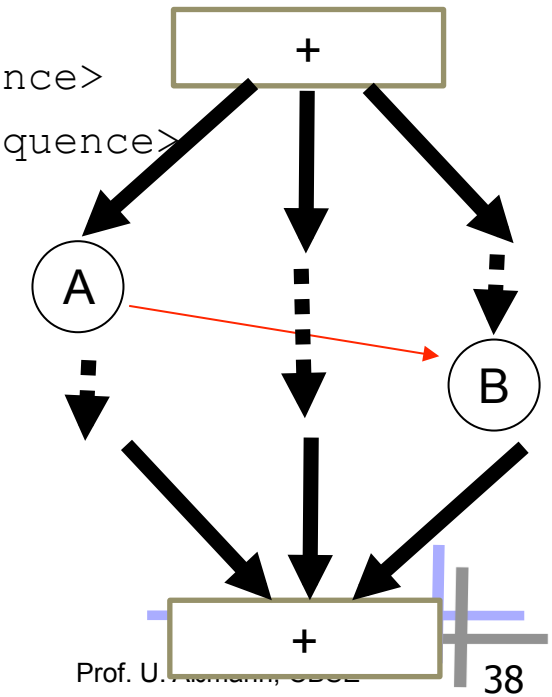
```
<!-- Process definition -->
<process name="OrderPizza" suppressJoinFailure="yes"
xmlns="http://schema.xmlsoap.org/ws/2003/03/business-process"
    pns="http://www.pizza.org/schema">
<partnerLinks>
  <partnerLink name="PizzaService" partnerLinkType="pns:OrderChannel"
    myRole="PizzaOrderer">
</partnerLinks>
<!-- Global Variables -->
<variables>
 <variable name="input" messageType="PizzaOrder"/>
 <variable name="output" messageType="PizzaDelivery"/>
</variables>
<faultHandlers> ... </faultHandlers>
<sequence name="body">
 <invoke name="order" partnerLink="PizzaService" portType="PizzaOrder"
    operation="body" variable="output">
 <receive name="acknowledgement" partnerLink="PizzaService" portType="Pizza"
    operation="body" variable="input">
</sequence>
</process>
```

Connector

Flow Operations are Workflow Graphs

- The <flow> operation is structured as a workflow graphs
  - The names of messages, ports, partner links help to span up the graph
  - <flow> executes its sequences in parallel
  - <links> can synchronize parallel tasks

```
<flow>
<links><link> name="A"</link>
           <link>name="B"</link>
</links>
<sequence>..<invoke> <target name="A">..</sequence>
<sequence>....<target name ="B">..........</sequence>
</flow>
```



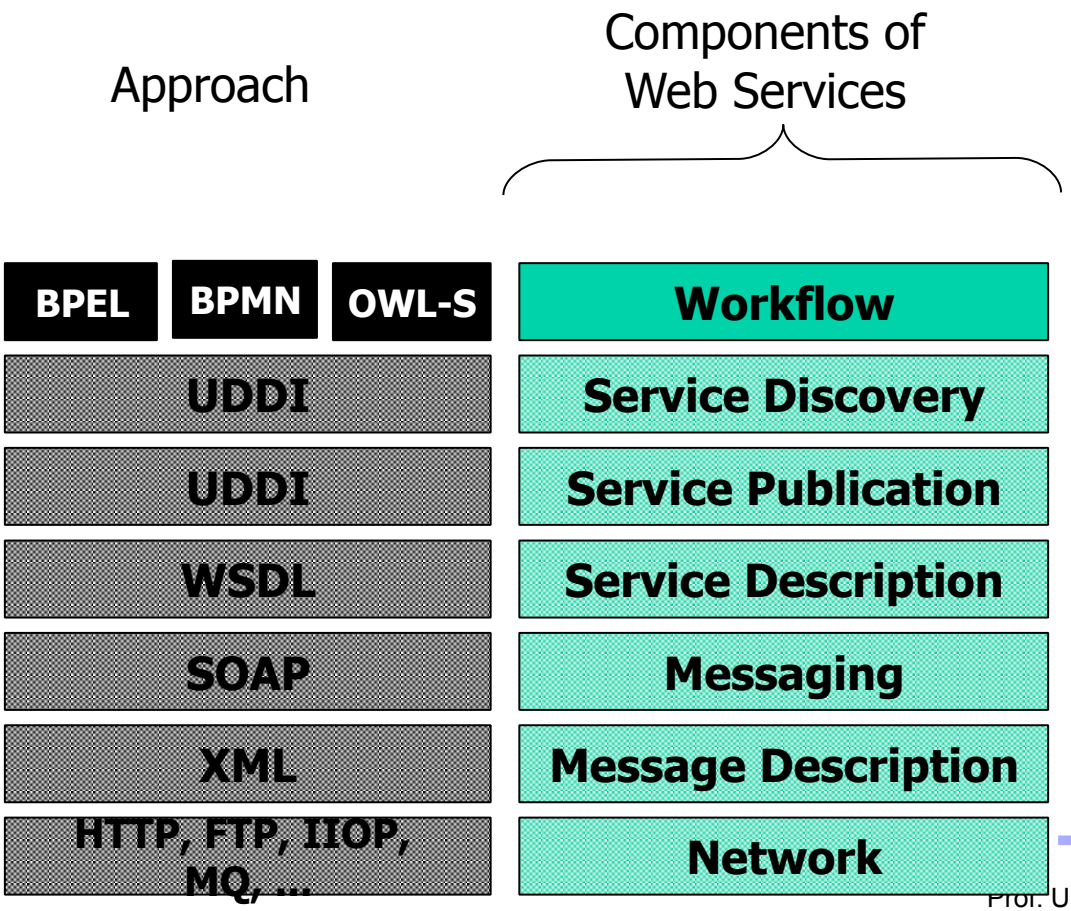
Other Operations in BPEL

- Structured control-flow operators
  - sequence
  - switch
  - while
  - flow
  - pick (XOR join)
  - terminate
- Compensate
  - Error compensation
- scope
- assign

BPEL Tools

- http://en.wikipedia.org/wiki/BPEL
- Eclipse BPEL project
  - http://www.eclipse.org/bpel/
- Orchestra tool
  - http://orchestra.ow2.org/xwiki/bin/view/Main/WebHome
- People work on the translation of Colored Petri Nets and UML activity diagrams from and to BPEL
  - CPN have good formal features (see ST-2)
  - Can be used for deadlock checking, resource control, etc.
  - YAWL is such a nice language, see the work of [van der Aalst]

Business Process

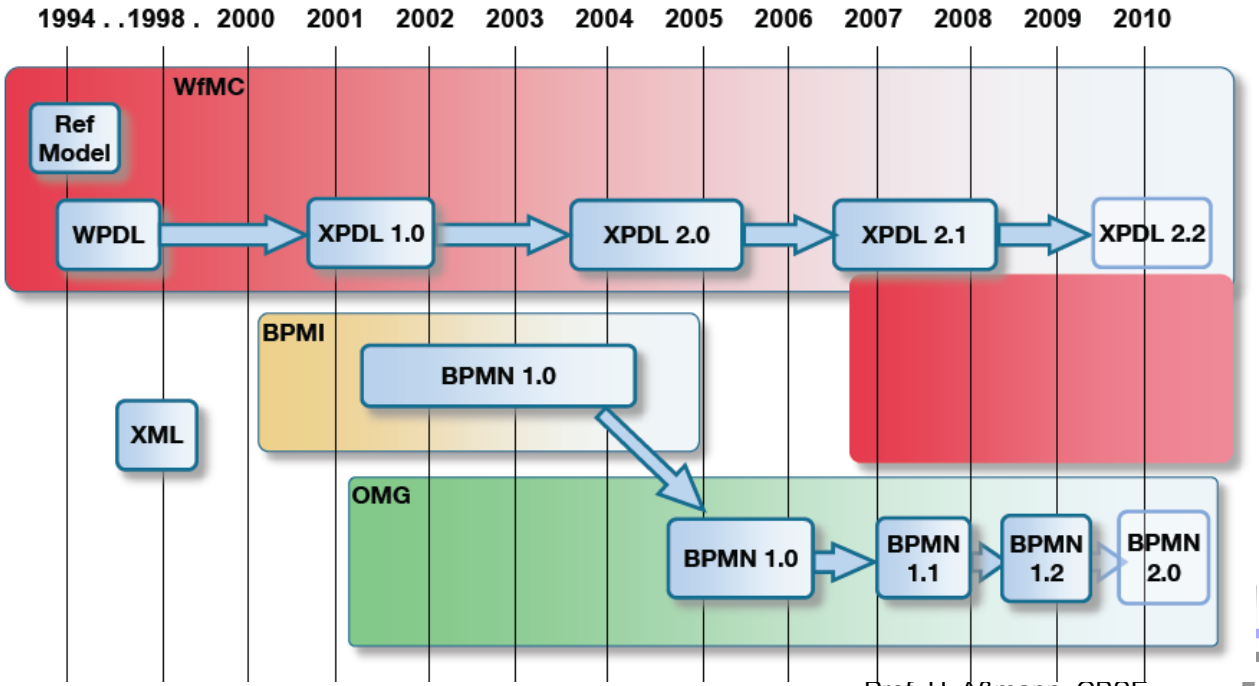


15.6 Business Process Modeling Notation (BPMN)

Another composition language

History

- The Business Process Modelling Notation (BPMN)
- Graphical notation for conceptual business processes
- Covers control, data, authorization, exception
- Standardized by OMG



Core Elements

Core Set of BPMN Elements

Flow Objects

Events

Activities

Gateways

Connecting Object

Sequence Flow

Message Flow

Association

Swimlanes

Pool

Lanes (within a Pool)

Artifacts

Data Object

Name
[State]

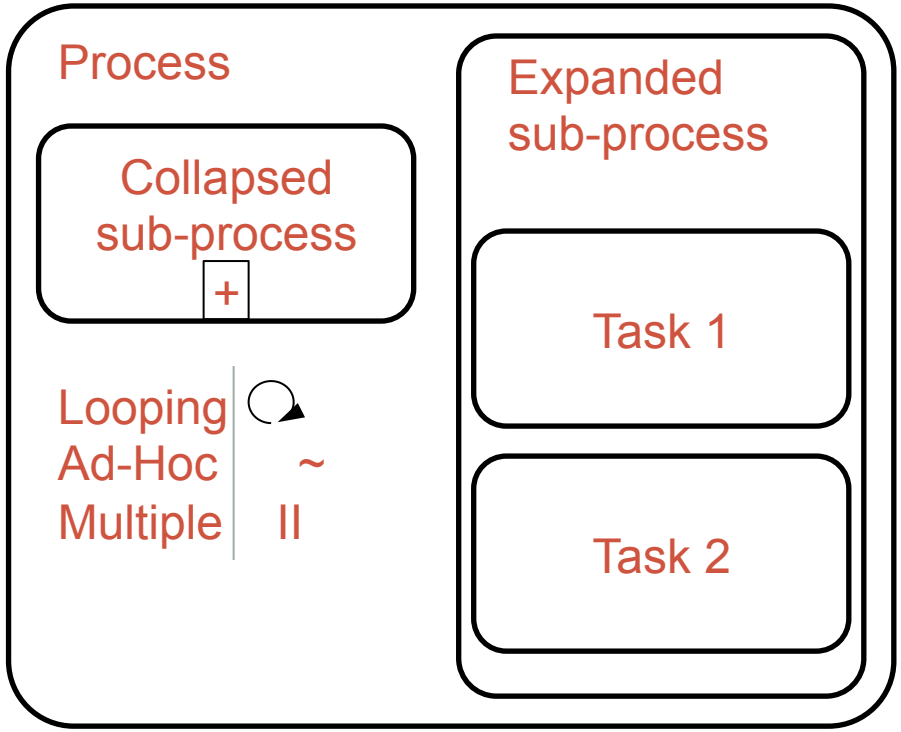
Text Annotation

Text Annotation Allows a Modeler to provide additional Information

Group

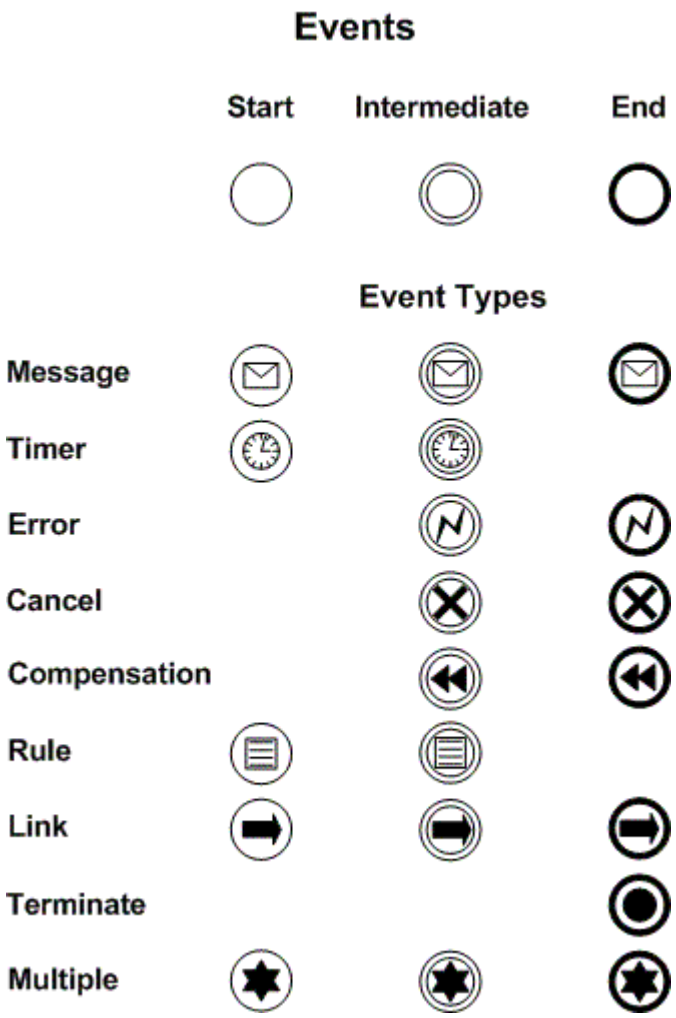
Activities and Processes

An **activity** is a generic type of work that a company performs. An activity can be atomic (task) or compound (process, sub-process).



Events and Activities

Events affect the flow of the process and usually have a cause (trigger) or an impact (result): 'Email received', 'Warehouse empty'



Gateways and Connections

A gateway is used to split or merge multiple process flows. It will determine branching, forking, merging and joining of paths.

Gateway control types

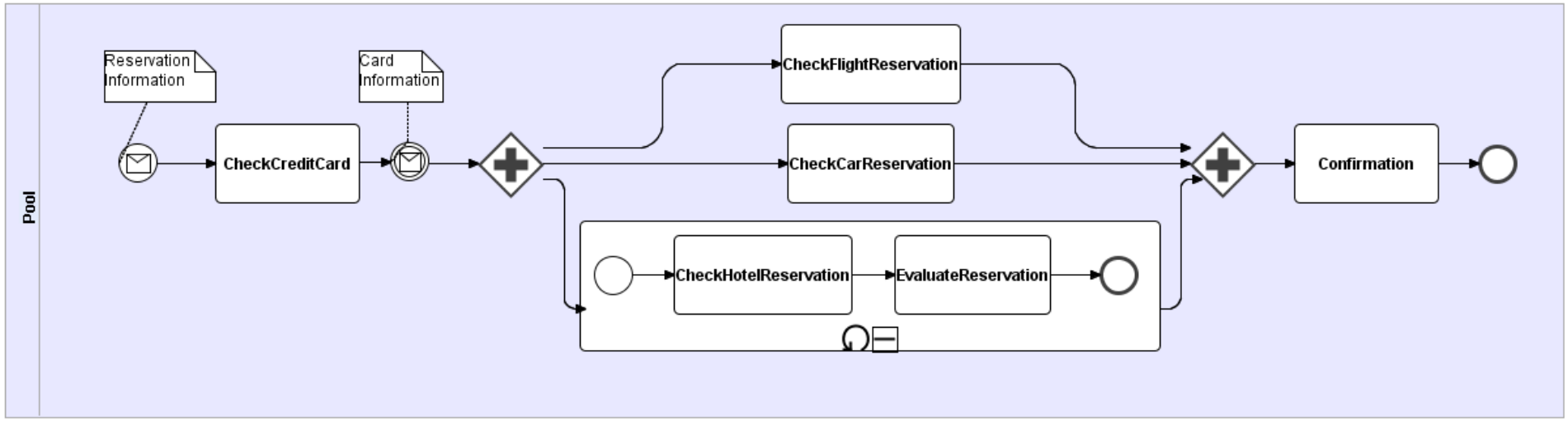
XOR (DATA)		Data based exclusive decision or merging. Both symbols have equal meaning. See also Conditional flow.
XOR (EVENT)		Event based exclusive decision only.
OR		Data based inclusive decision or merging.
COM-PLEX		Complex condition (a combination of basic conditions)
AND		Parallel forking and joining (synchronization).

Graphical connectors

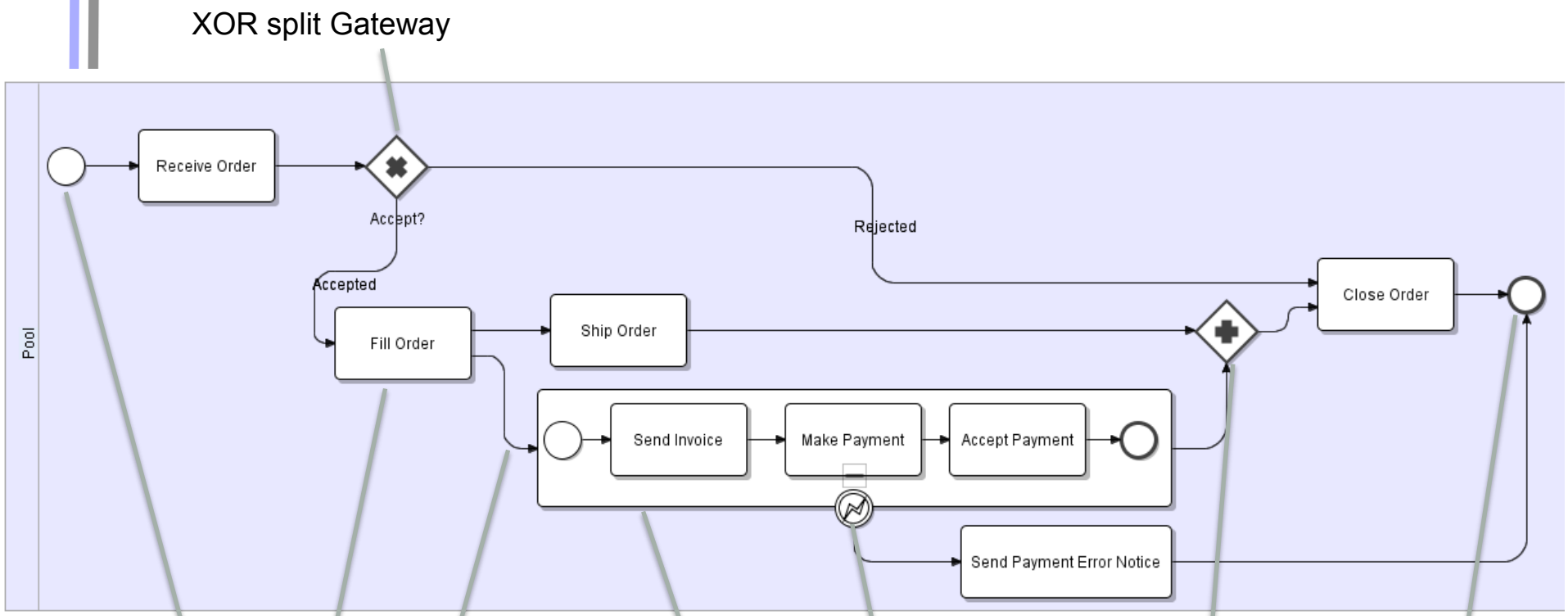
Normal sequence flow	
Conditional sequence flow	
Default sequence flow	
Message flow	
Association	

Example: Travel Process Control Flow

More refinement leads to business process specifications (with control and data flow)



Example


XOR split Gateway
Pool
Receive Order
Accept?
Rejected
Accepted
Fill Order
Ship Order
Close Order
Send Invoice
Make Payment
Accept Payment
Send Payment Error Notice
Task
SubProcess
Start Event
Sequence Flow
Error Intermediate Event
AND join Gateway
End Event

Why BPMN?

BPMN v1.x

- Modeling language, no execution semantics
- BPMN includes a partial mapping to Business Process Execution Language (BPEL)
- Changes in the BPMN model do not update in the BPEL code

BPMN v2.x

- Execution semantics
- Explicit service mapping
- First engines are available (jBPM for jBoss)

BPMN geared towards business analysts:

- BPMN constructs are simplified
- UML notation too bloated
- BPMN is on the platform-independent level, BPEL nearer the platform-specific level

Give BPMN a try

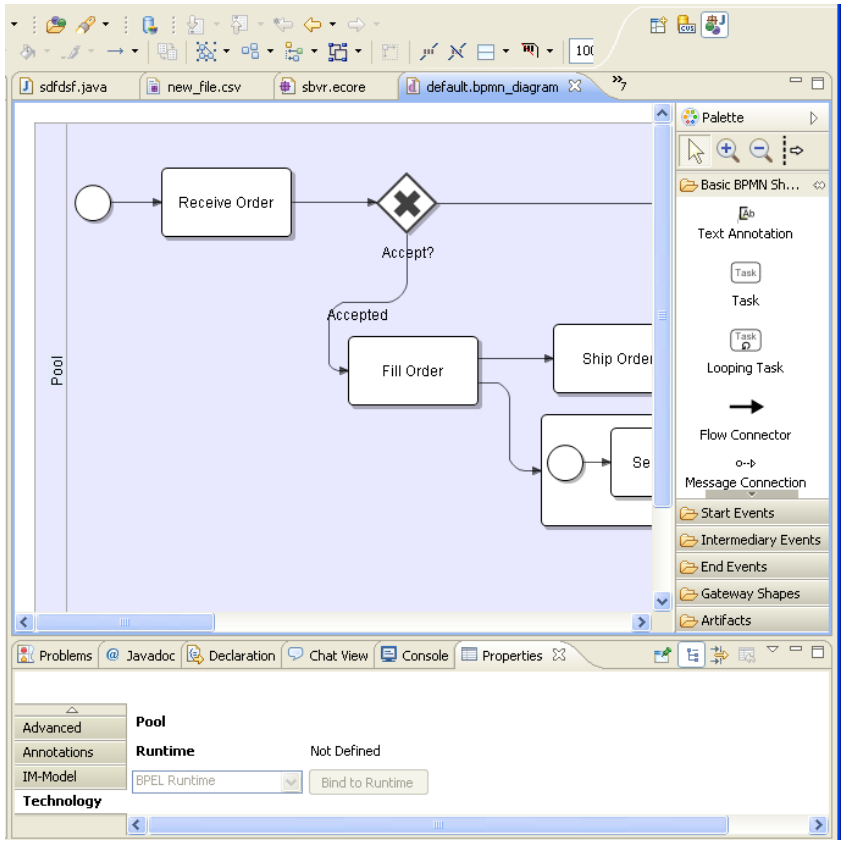
Free BPMN Editor
from Eclipse

Included in the SOA
Tools Project

http://www.eclipse.org/bpmn/

http://www.eclipse.org/
bpmn2-modeler/

SAP has decided to use BPMN
in their products



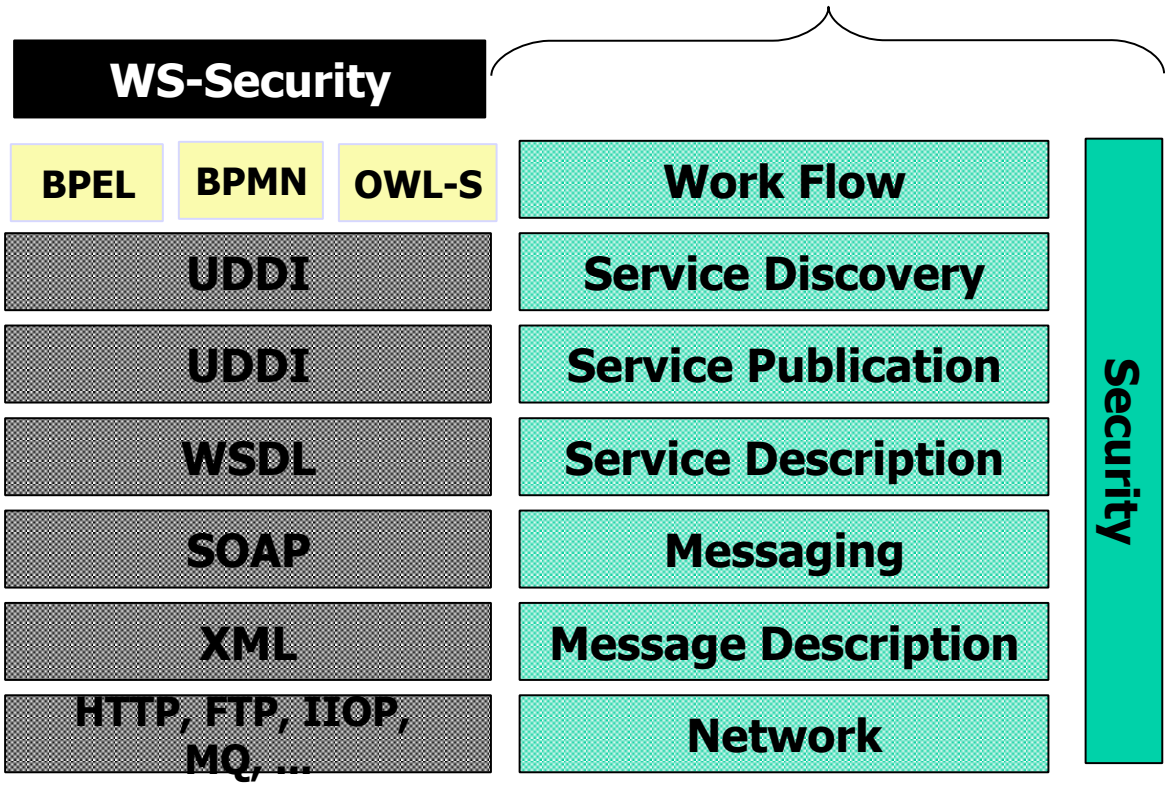
15.7 Trust and Security

Trust and Security

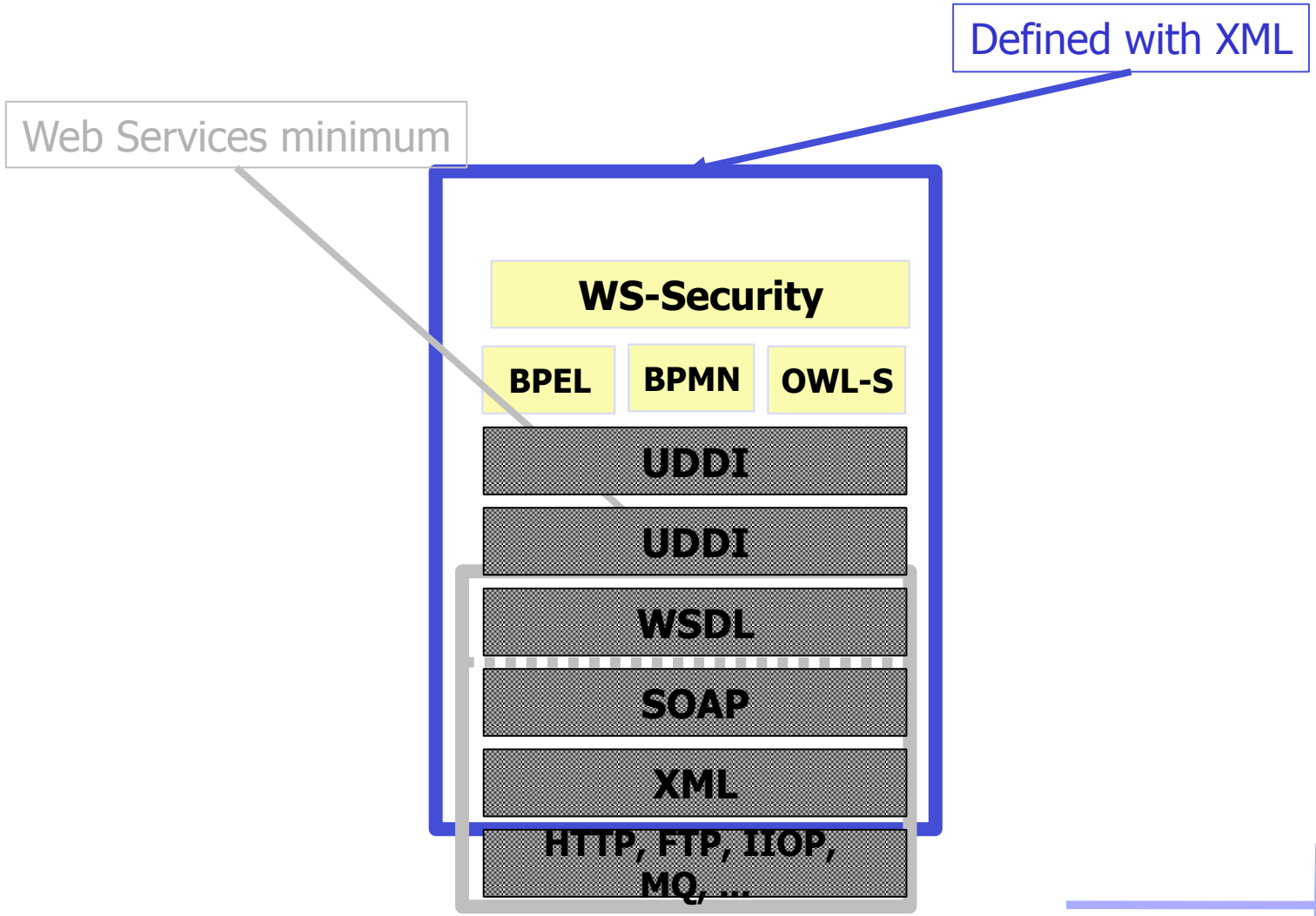
- Policies: when do I allow somebody to do something
- Integrity: intact, unchanged
- Confidentiality: cryptification policy
- Authentication: proof of identity
- Authorization: access to execute certain services
- Non-Repudiation: waranty on failior
- Legal Rights: copy rights, reselling rights, ...
- Privacy: handling personal data
- ...

Trust and Security

Components of
Web Services

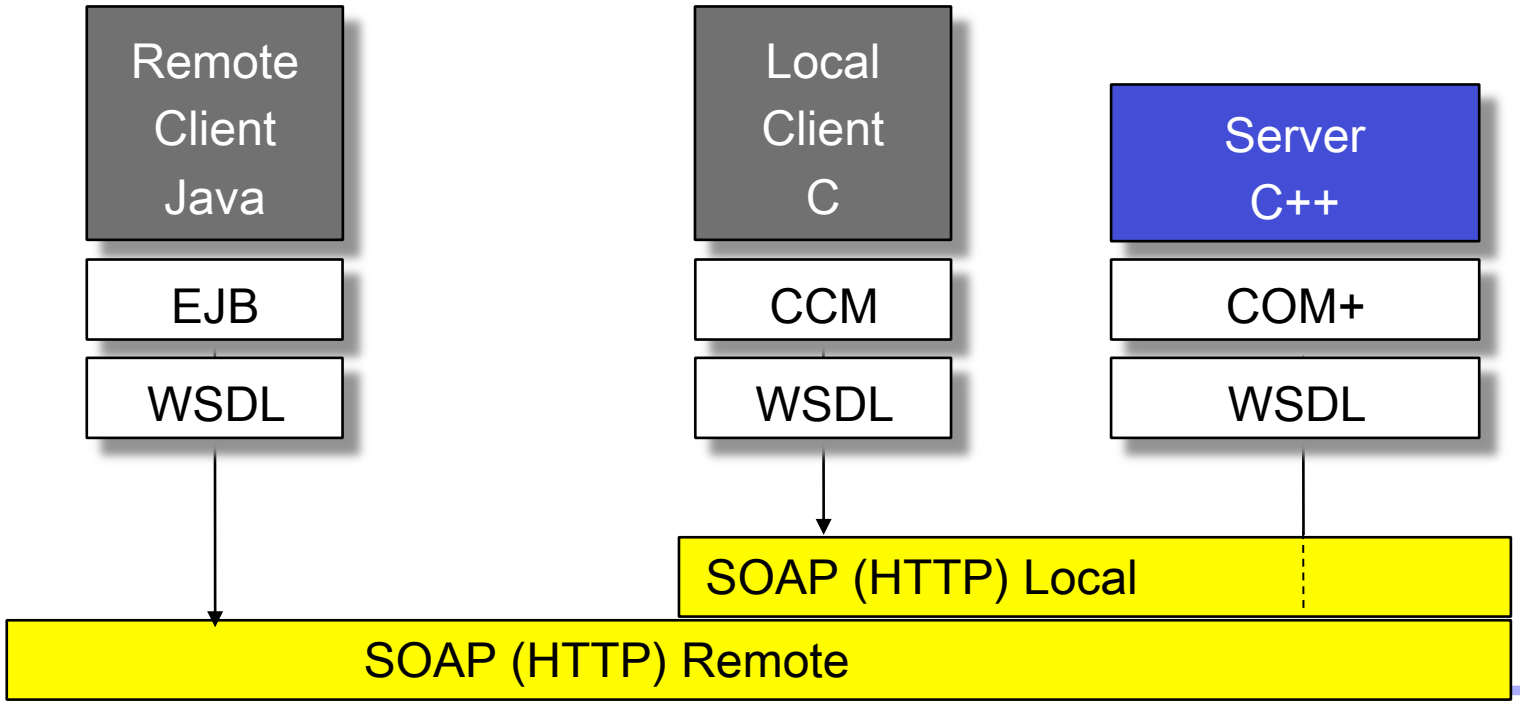


Technical Conclusion



Web Services – Component Model Transparency

- Language adaptation: XML Schema + WSDL
- Remote transparency: SOAP (+ HTTP)
- Component model transparency (EJB, COM+, CORBA, CCM, Beans, etc...)



15.7 Evaluation of Web Services

as composition system

Component Model

- Mechanisms for secrets and transparency: very good
  - Location, language, component model transparency
  - Communication protocol transparency
  - Interface specification is flexible with WSDL
  - Different black-box component models can be hidden under WSDL specifications
- Generic BPEL Web Services are possible (without bound WSDL ports)
- BPMN Web Services can be stepwise refined from abstract to concrete

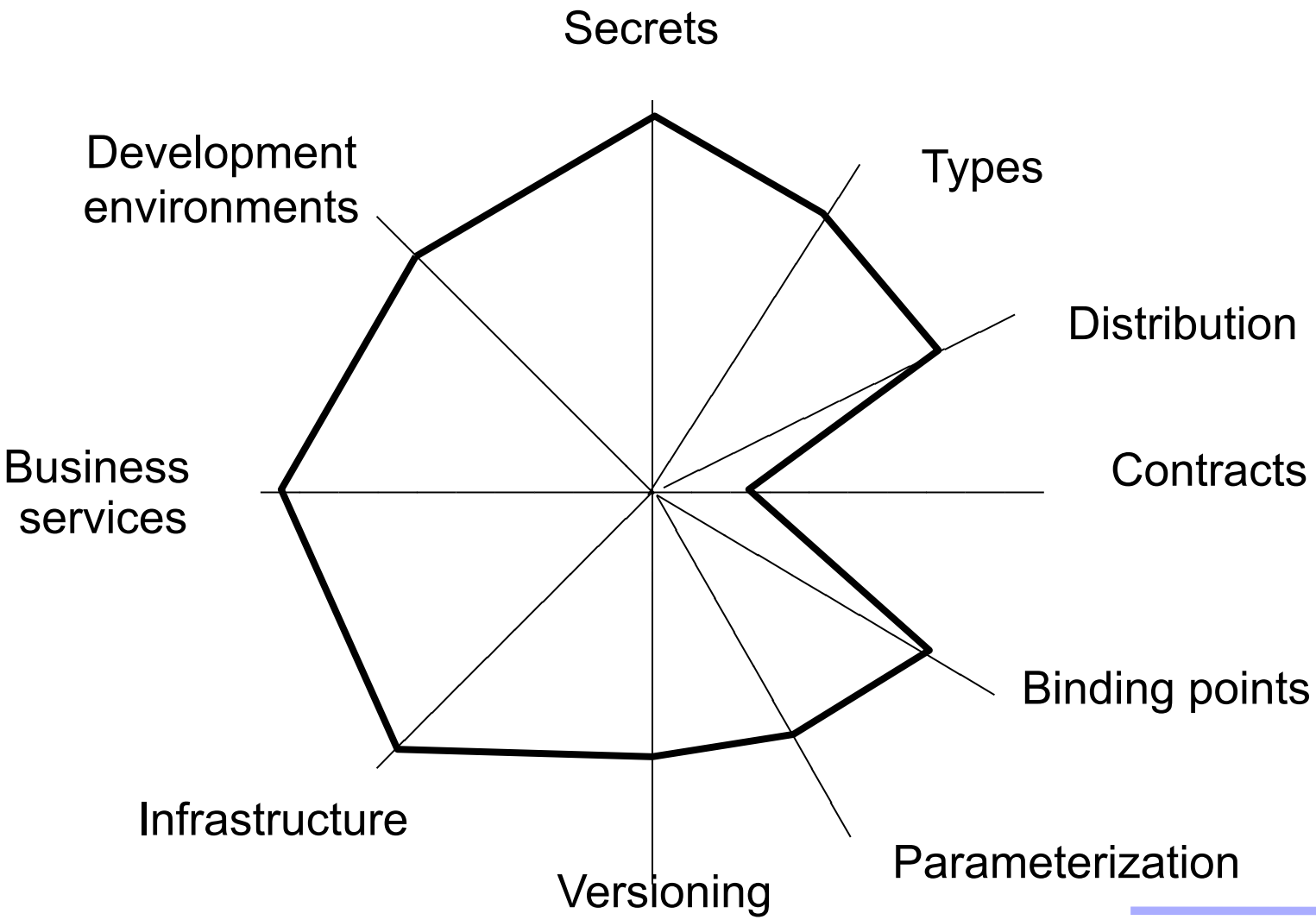
Composition Technique

- Mechanisms for connection
  - Protocol transparency allows for flexible connections
  - WSDL binding is flexible
- Mechanisms for aspect separation
  - Separate modeling from execution (abstract business processes from workflows)
- Scalability: Better
  - Changes of protocol possible
  - Changes of distribution easy
  - Changes of workflow easy

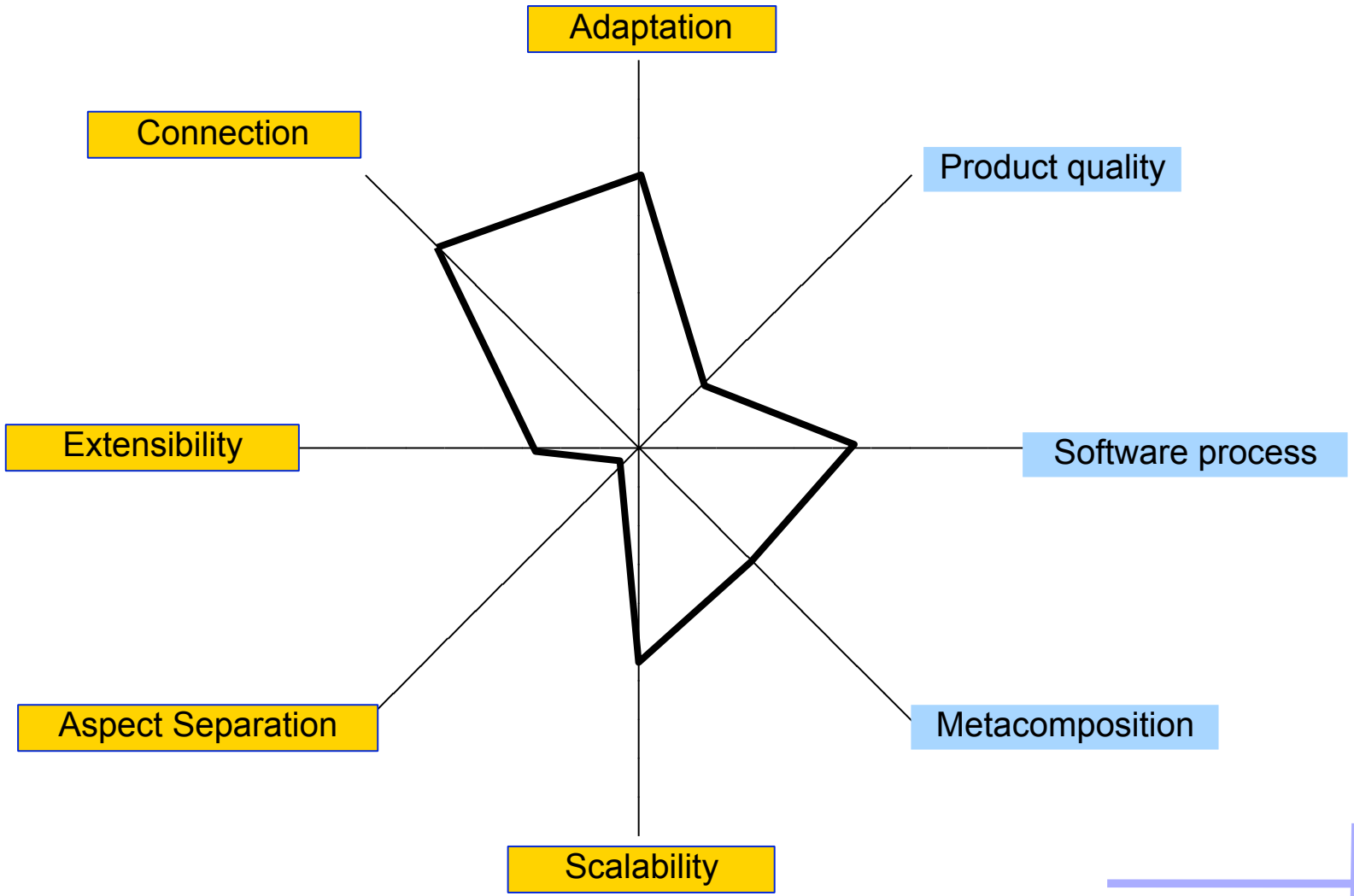
Composition Language

- BPEL, BPMN are flexible composition languages
  - Not yet full exchangeability of connector types
  - But graphic support for workflow specifications
  - Control- and data-flow operators (gateways)
  - Parallel execution semantics
  - Abstract (business processes) and executable level (workflows)
- Metacomposition fully supported
  - The generation of a BPEL or BPMN script is easy, because it is XML based
  - Environments generate workflow from other specifications
  - Generic workflow architectures will be possible

Web Services - Component Model



Web Services – Composition Technique and Language



Web Services as Composition Systems

Component Model

Contents: Completely hidden

Binding points: WSDL ports

Composition Technique

Adaptation: well supported

Automatic transactions, recovery

Several types of connectors

BPEL BPMN etc.

Composition Language

So Far: Blackbox Composition

- Standard middleware
  - CORBA, DCOM
- Implicit middleware
  - EJB
- Architecture systems
  - Web services

The Ladder of Component and Composition Systems

Aspect Systems Aspect Separation *Aspect/J*	View Systems Composition Operators *Composition Filters* *Hyperslices*	Software Composition Systems Composition Language *Invasive Composition* *Metaclass Composition* *Piccola*
Architecture Systems	Architecture as Aspect	*Darwin* *BPEL* *ACME* *BPMN*
Classical Component Systems	Standard Components	*.NET* *CORBA* *Beans* *EJB*
Object-Oriented Systems	Objects as Run-Time Components	*C++* *Java*
Modular Systems	Modules as Compile-Time Components	*Modula* *Ada-85*

The Second Part of the Course: Greybox Composition

- Generic programming
  - Generic program elements
  - Home-made connectors
- View-based programming
  - Formal foundations (lambda N, pi-calculus)
  - Record calculi, Scala
  - Hyperspace programming
- Aspect-oriented development
  - Aspect-oriented programming
  - Aspect-oriented design
- Invasive software composition
  - Slots and hooks
- Novel Forms of Composition
  - Uniform composition
  - Active document composition: Architectural styles for active documents and web systems
  - Roundtrip engineering and composition

Some Abbreviations

- ebXML: Electronic Business XML
- UDDI: Universal Description, Discovery and Integration
- OAG: Open Applications Group
- OASIS: Organization for the Advancement of Structured Information Standards
- SOAP: Simple Object Access Protocol
- HTTP: Hypertext Transfer Protocol
- tpaML: Trading Partner Agreement Markup Language
- UML: Unified Modeling Language
- UN/CEFACT: United Nations Centre for the Facilitation of Procedures and Practices in Administration, Commerce and Transport
- WSFL: Web Services Flow Language
- WSDL: Web Services Description Language
- WSIL: Web Services Inspection Language
- WSXL: Web Services Experience Language
- WSCL: Web Services Conversation Language
- WSUI: Web Services User Interface
- WSML: Web Services Meta Language
- WSCM: (Web Services Component Model) Numer omdöpt till WSIA
- WSIA: Web Services for Interactive Applications
- WSEL: Web Services Endpoint Language
- WSRP: Web Services for Remote Portals

Some URLs

- www.ebxml.org
- www.uddi.org
- www.oasis-open.org
- www.uncefact.org
- www.w3.org
- www.omg.org
- www.biztalk.org
- www.soapclient.com
- www.soapware.org
- www.xml.com
- www.xml.org
- www.webservices.org
- www.webservicesarchitect.com
- www.ws-i.org

The End

- Many slides inherited from
- Stig Berild's talk on the Nordic Conference on Web Services, Nov. 2002
- Prof. Welf Löwe, Web Service Competence Center (WSCC), Växjö Linnaeus University

15.9 OWL-S (Web Ontology Language for Services)

Additional material

- OWL-S definition at http://www.w3.org/Submission/OWL-S/
- http://daml.semanticweb.org/services/owl-s/1.0

OWL Web Ontology Language

- Classes and relationships
- Expressions to compute (derive) new classes and relationships *(derived model)*
  - Union, intersection of relations and classes
  - Cardinality restrictions
  - Existential quantifiers
- Roughly speaking, OWL corresponds to UML-class diagrams without methods + OCL + class expressions
- Instead of plain XML, OWL can be used to type data
  - Beyond trees and context-free structures, graphs, knowledge webs, semantic nets can be described (context-sensitive structures)

OWL-S

- Based on OWL, a language for specification of web services has been developed by the OWL-S coalition
- Specification has three parts:
  - *Service profile*: semantic service description, service offer, service functionality (*what does the service provide?)*
    - Based on domain ontologies in OWL, i.e., OWL-specified attributes
  - *Service model*: service realization, decomposition of a service (*how does the service work?)*
    - Service is also called a *process*
    - Here, OWL-S provides a process ontology
  - *Service grounding*: service mapping to underlying mechanisms (*how is the service mapped to a component model and transport protocol?)* Similar to WSDL grounding

OWL-S Processes

- **Atomic**
  - Cannot be decomposed
  - Can be called and executed
  - Can be mapped to WSDL process descriptions (*grounding*), and hence, to SOAP
- **Simple**
  - Cannot be decomposed
  - Can be executed, but not be called from outside
- **Composite**
  - Build from atomic and simple processes

Service Model (Process Model) of OWL-S

- Process Ontology
  - Describes a service (process) with an *IOPE* specification
    - Inputs
    - Outputs
    - Parameters
    - Effects
- Process control ontology (for composite processes)
  - Internal realization with state, activation, execution, completion (control-flow specification)

Creating an OWL-S specification

- Describe atomic processes
- Describe grounding of atomic processes
- Describe compositions
- Describe simple processes
- Describe profile of service

OWL-S Statements of a Composite Process

- Unordered (unspecified order)
- Sequence
- Split
- Split+Join (fork and join)
- Concurrent
- Choice
- If-then-else
- Repeat-until
- Repeat-while