

32. Web Services, Workflows and Service-Oriented Architectures

Lecturer: Dr. Sebastian Götz

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und
Multimediatechnik
<http://st.inf.tu-dresden.de>
12. Mai 2017

- 1) Web Services as a specific form of service-oriented architectures
- 2) WSDL
- 3) BPEL
- 4) BPMN
- 5) Evaluation
- 6) Appendix
 - 1) OWL-S

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://www.martinfowler.com/workflowpatterns.com/documentation/documents/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.bpmn.org/> BPMN home page at OMG

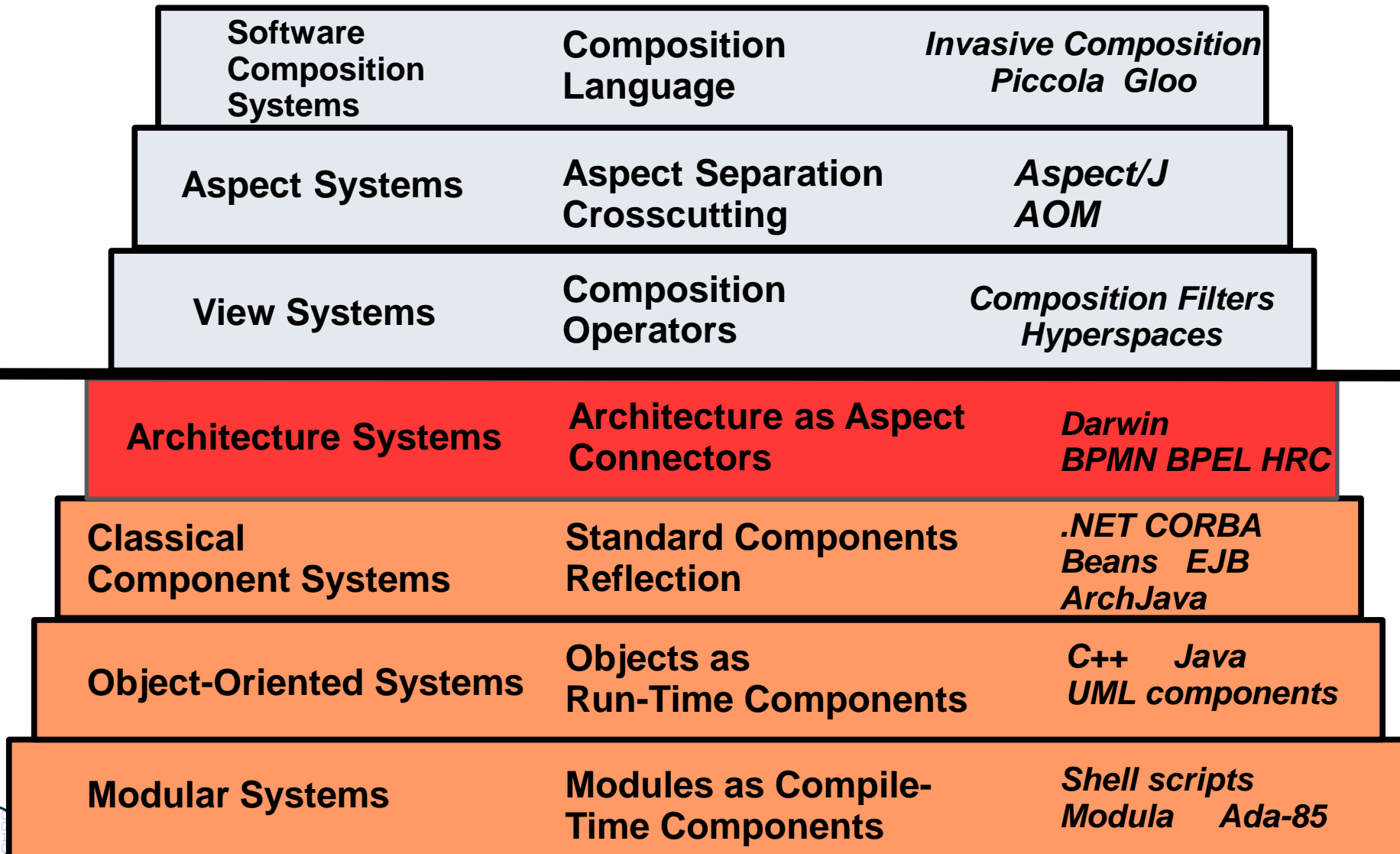


Other Literature

- ▶ Matthias Weske. Business Process Management – Concepts, Languages, Architectures. Springer. 2007
- ▶ YAWL <http://www.yawlfoundation.org/>
- ▶ H. P. Alesso, C. F. Smith. Developing Semantic Web Services. A K Peters Ltd, Natick, Massachusetts, 2004.
- ▶ BPMN 2.0 language specification
 - ▶ <http://www.omg.org/spec/BPMN/2.0/>
- 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. ICSSOC 2007, LNCS 4652.

The Ladder of Composition Systems

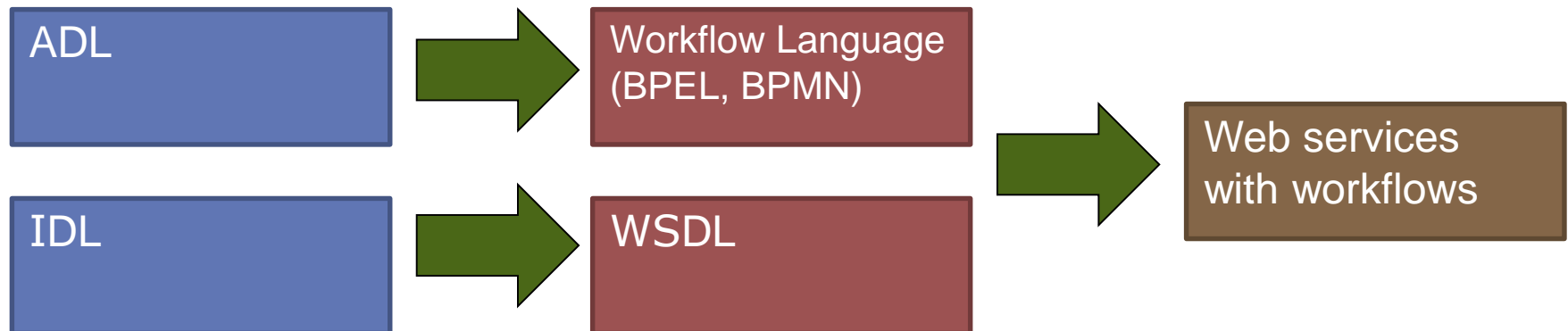
Component-Based Software Engineering (CBSE)



32.1 Web Services as Architecture Systems

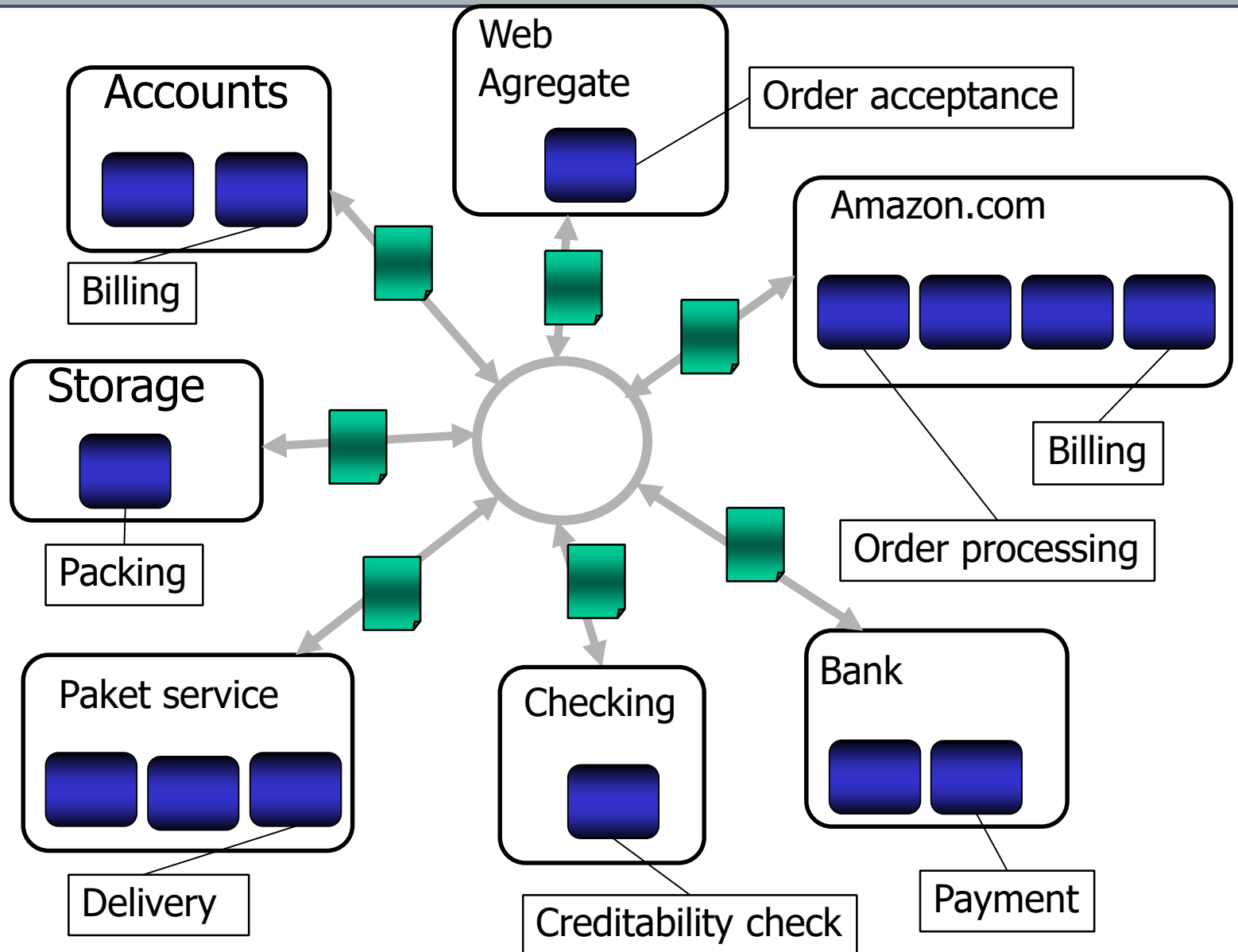
Web Services and Architecture Systems

- ▶ 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)



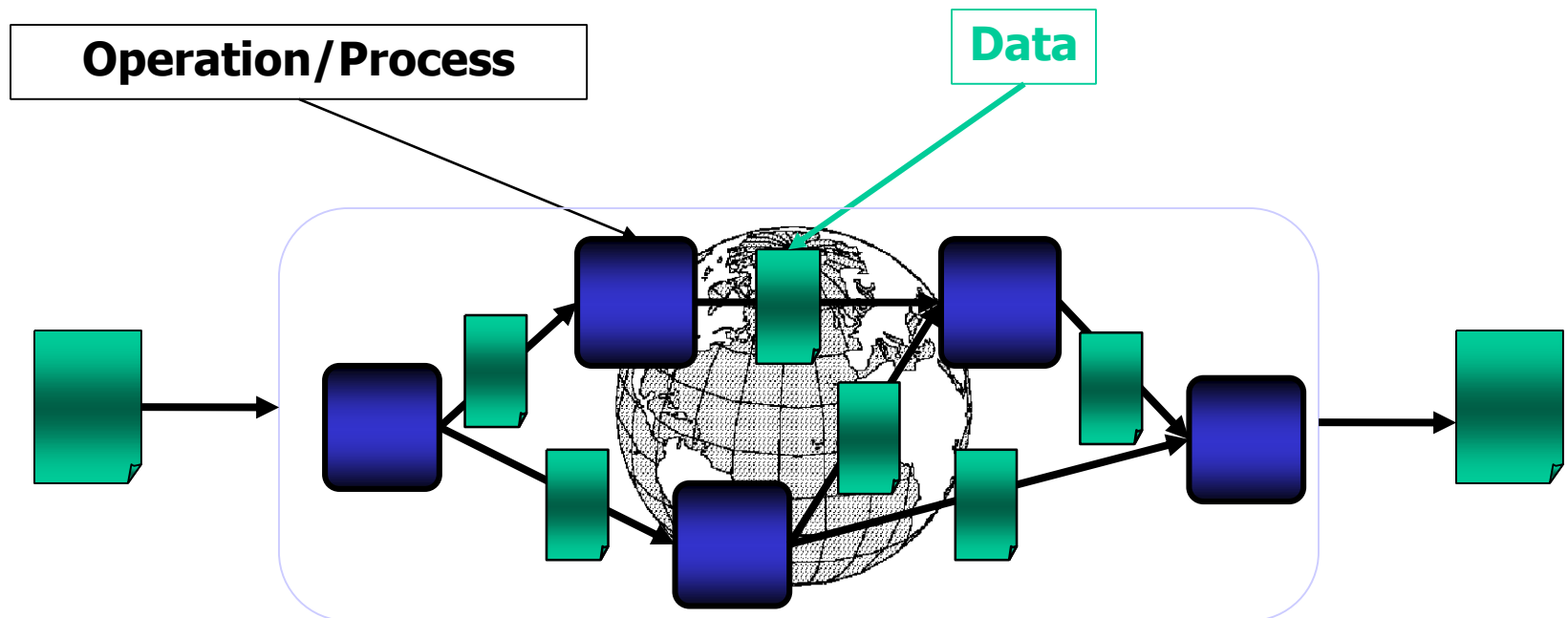
Web Services are Black-Box Components

Component-Based Software Engineering (CBSE)



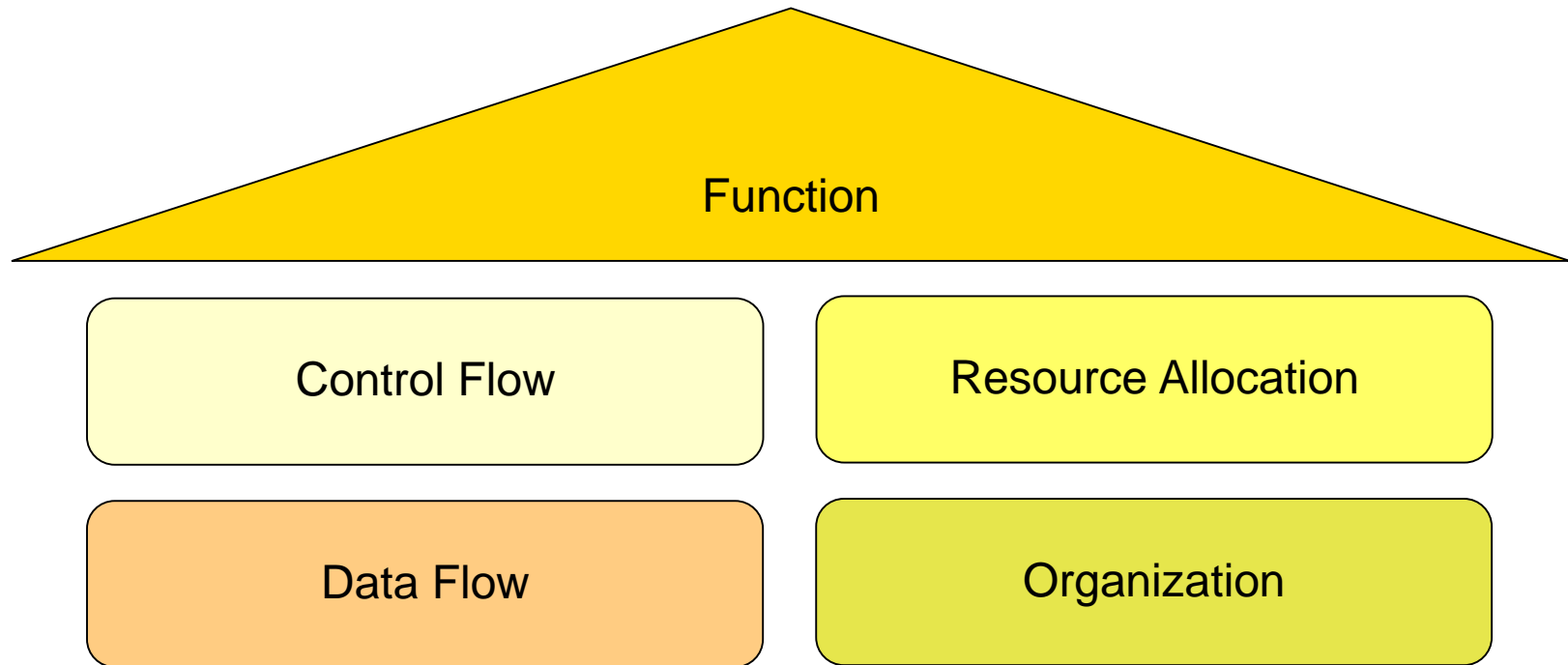
Web Service Architectures are Described by Workflows

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



Workflows Languages Have Aspects

- Standard workflow modeling discerns about 5 aspects
 - ex. ARIS house [Scheer's company IDS, now Software AG]



[Scheer]

Which Types of Operational Specifications Exist for Workflows?

- ▶ **Data-flow graphs** (data flow diagrams, DFD) focus on data flowing through operations
 - Activity diagrams: data flows through actions
 - See courses Softwaretechnologie II
- ▶ **Control-flow graphs** (CFG) focus on control dependencies
 - Nodes are control-flow operations that start other operations on a state
 - The standard representation for imperative programs
- ▶ **State systems** focus on transitions between states
 - Finite State Machines (FSM): events trigger state transitions
 - Statecharts: Hierarchical FSM
- ▶ **Mixed approaches**
 - **Colored Petri nets:** tokens mark control and data-flow, see course Softwaretechnologie II
 - **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

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, to Statecharts, or to data-flow diagrams
 - YAWL (van der Aalst, Eindhoven)
 - 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

What is a Business Process?

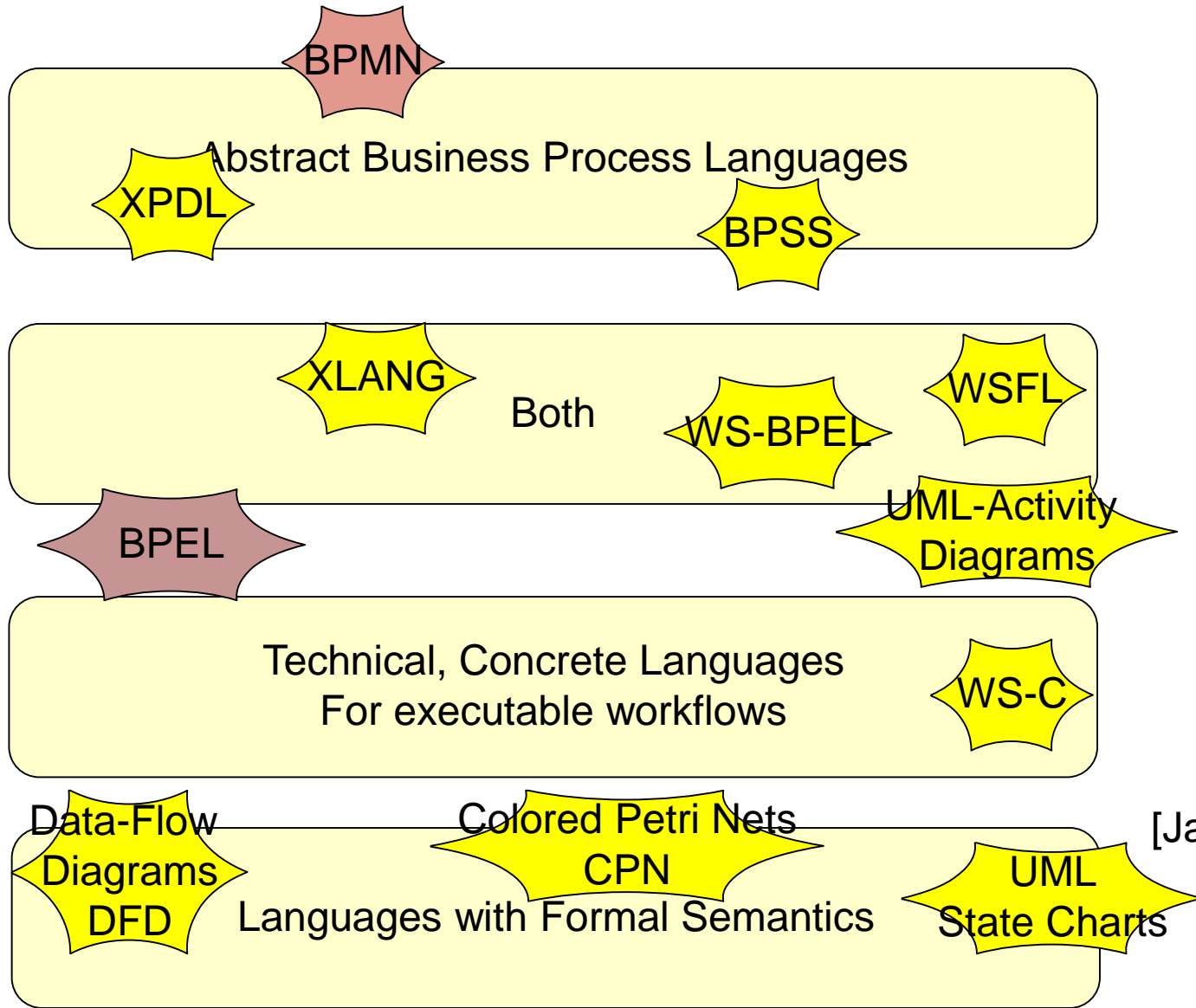
Business Processes are Abstract Workflows

Component-Based Software Engineering (CBSE)

- **Business processes** are *partial* or *abstract* workflows describing processes in enterprises
 - 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*).



Languages Serve Different Abstraction Levels



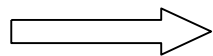
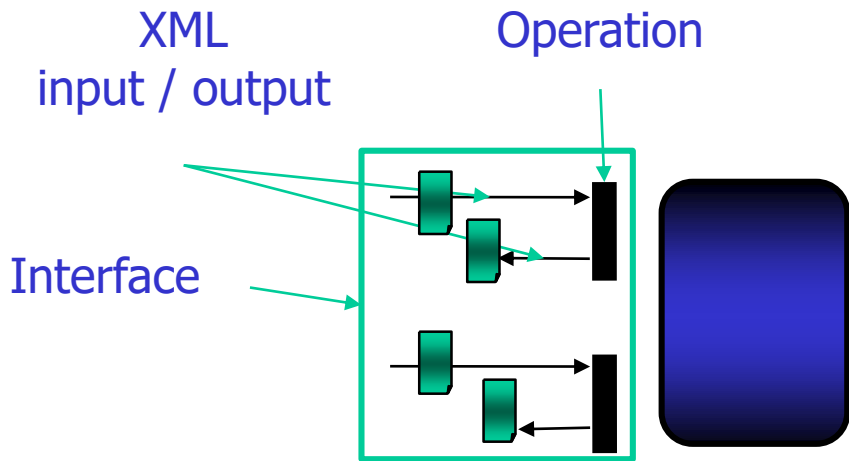
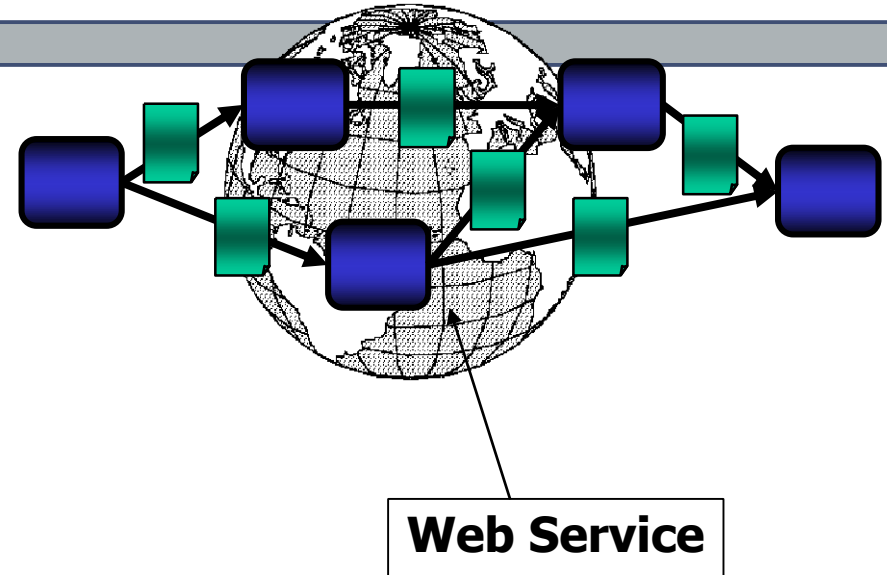
What are Workflow Engines?

- ▶ **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



32.2 WSDL for the Definition of Interfaces of Web Services

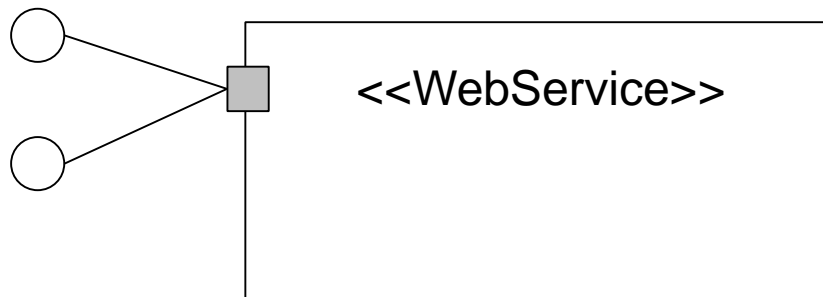
Service Interface



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
 - WSDL unifies call and 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 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 (not in IDL)
- ▶ **Operation**
 - An interface of the service, with input and output, fault parameters
- ▶ **Port type**
 - A named set of operations (as in UML components or IDL)
- ▶ **Binding**
 - A mapping of the port to underlying component models, e.g., http, soap, or mime
- ▶ **Service**
 - A set of related ports (as in UML components)

WSDL Reuses Data Types of XSD

Here: Type Definitions <schema> <element> <complextype>

Component-Based Software Engineering (CBSE)

```
<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:boolean"/>
```

```
        </XMLSchema:sequence>
```

```
      </XMLSchema:complextype>
```

```
    </XMLSchema:element>
```

```
  </XMLSchema:schema>
```

```
</wsdl:types>
```



Different Kinds of Port Types

▶ **Event- or message-based ports**

- Notification: data-out port
- One-way: data-in port

▶ **Call ports:**

- Request-Response: procedure port (callee port)
- Solicit-Response: send, then receive (caller port)

```
<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>
```



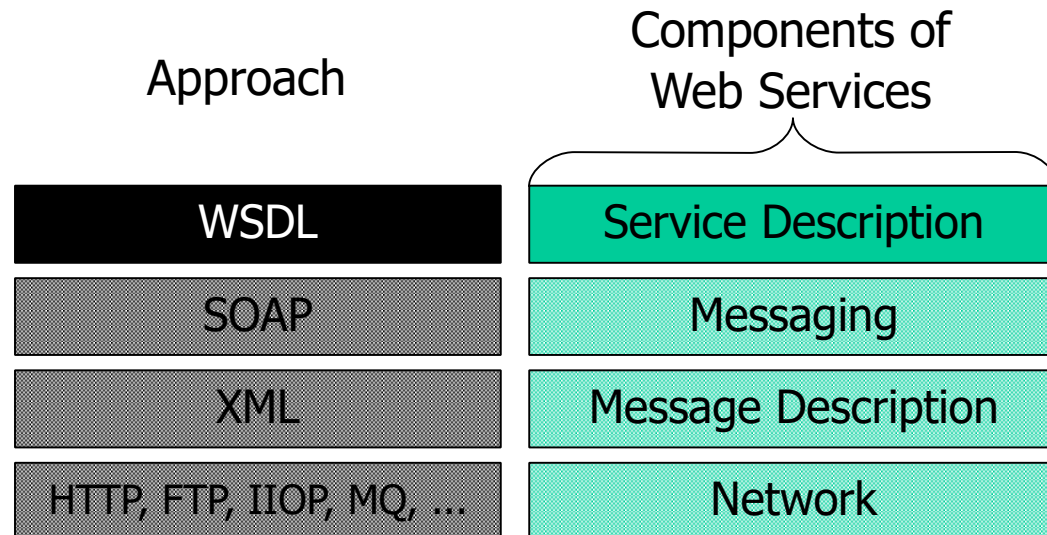
Example: Binding WSDL to SOAP

```
<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 an Interface Definition Language (IDL)
 - Part of BPEL (see later)
 - ▶ W3C Recommendation (standard)

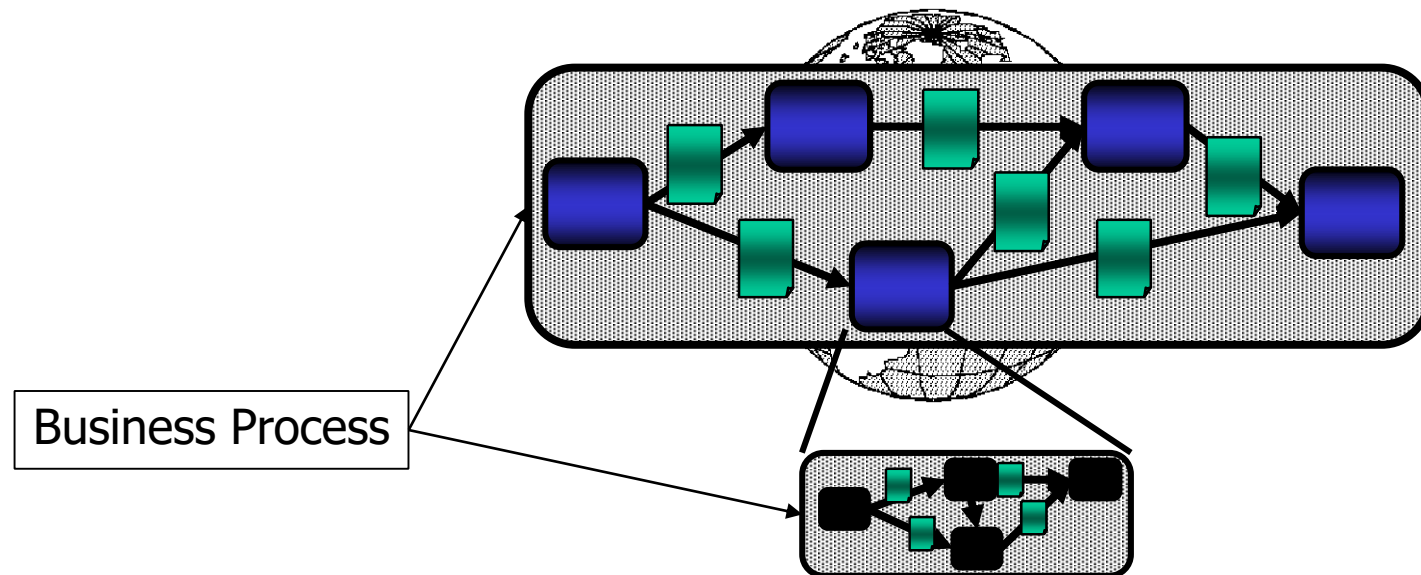


32.3 Business Process Execution and Web Service Workflows with BPEL

- BPEL, a web service composition language

Business Process Definition

- ▶ BPEL is the major language for execution of business processes today
 - ▶ Business process on the web
 - ▶ IBM & Microsoft
- ▶ There are many languages proposed today:
 - ▶ 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 at their ports
 - ▶ BPEL uses WSDL for service interface descriptions, as IDL
 - ▶ BPEL adds connection types (*partner link types*)



BPEL Made Simple

- ▶ BPEL is an 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 XML syntax
 - ▶ 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** specifies compensation actions for inconsistencies or damages a fault has provoked
 - ▶ Optimistic transactions with compensations
- ▶ **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>
```

Connector

<!-- 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>
```



BPEL Tools

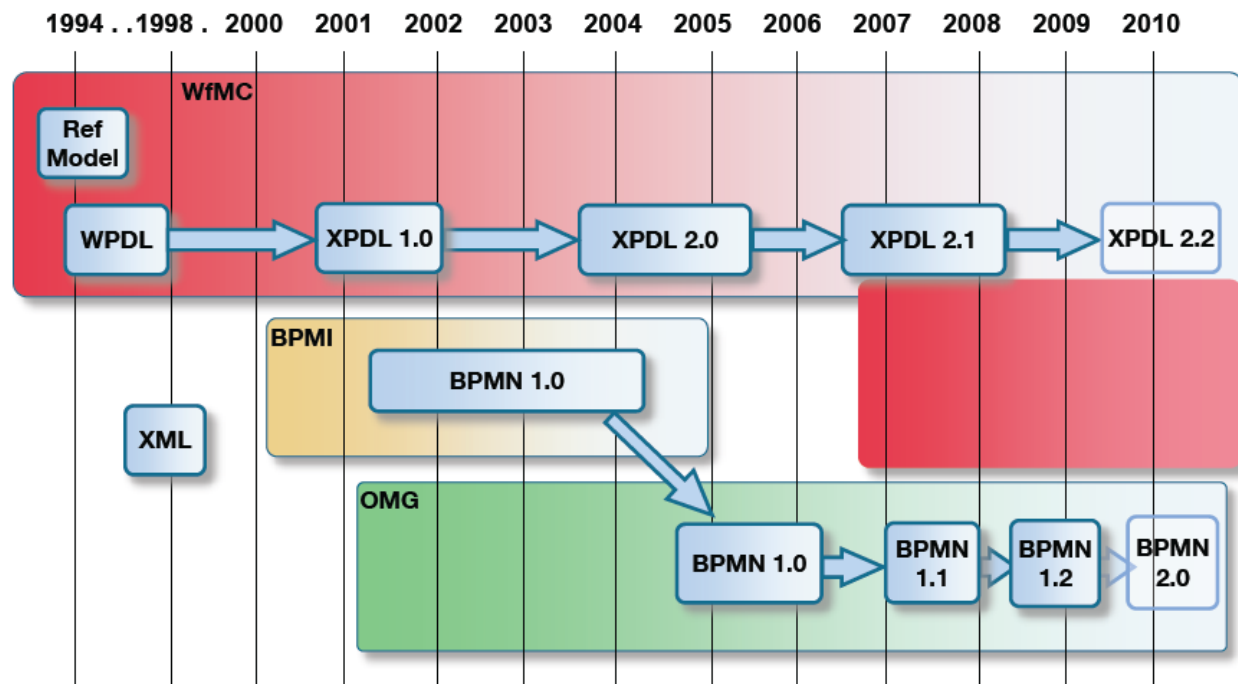
- ▶ Eclipse BPEL project
 - ▶ <http://www.eclipse.org/bpel/>
- ▶ Orchestra tool
 - ▶ <http://orchestra.ow2.org/xwiki/bin/view/Main/WebHome>
- ▶ People work on the translation of Colored Petri Nets and UML activity diagrams from and to BPEL
 - CPN have good formal features (see ST-2)
 - Can be used for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

32.4 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

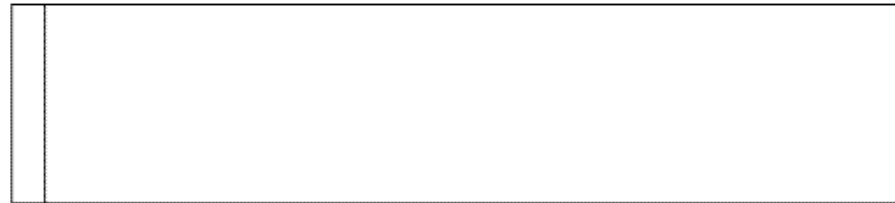


Association

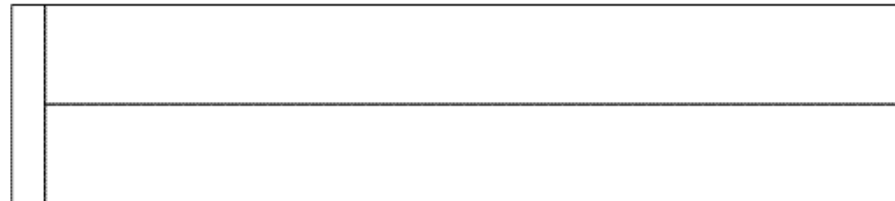


Swimlanes

Pool



Lanes (within a Pool)



Artifacts

Data Object



Name
[State]

Text Annotation

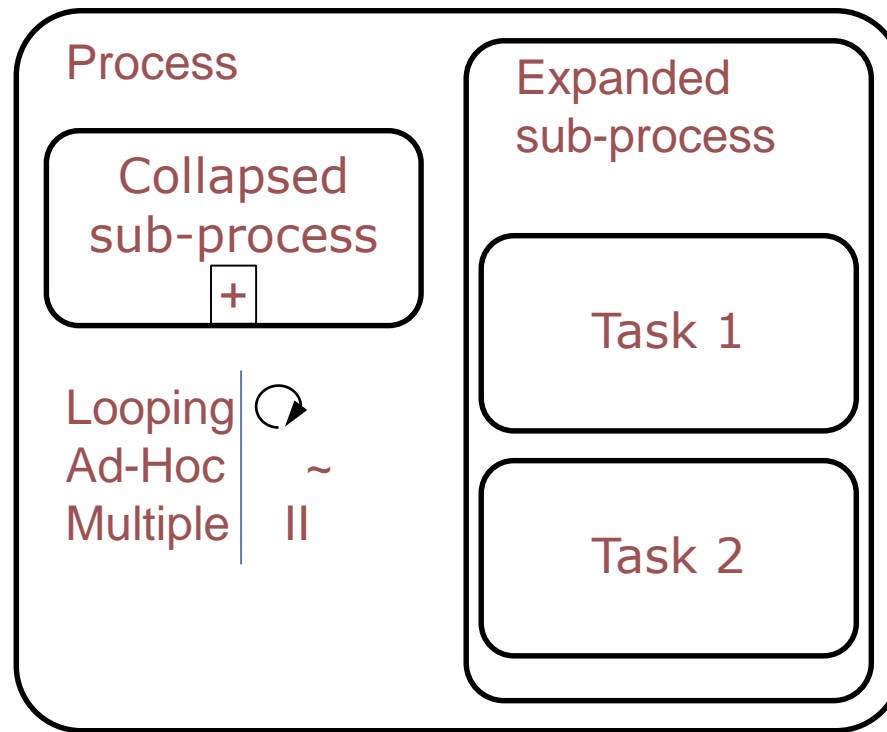
Text Annotation Allows a Modeler to provide additional Information

Group



Activities and Processes

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



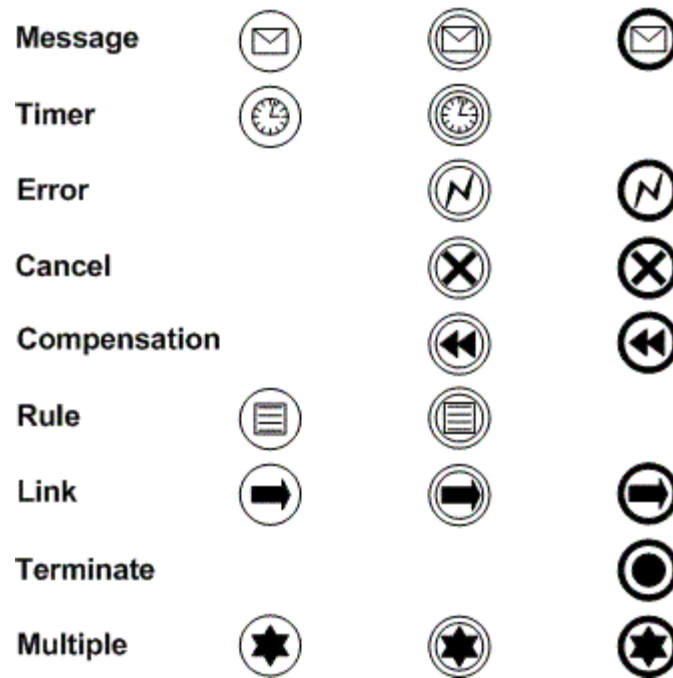
Events and Activities

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

Events



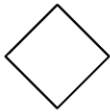





Event Types



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

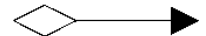
Normal

sequence flow



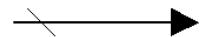
Conditional

sequence flow

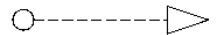


Default

sequence flow



Message flow

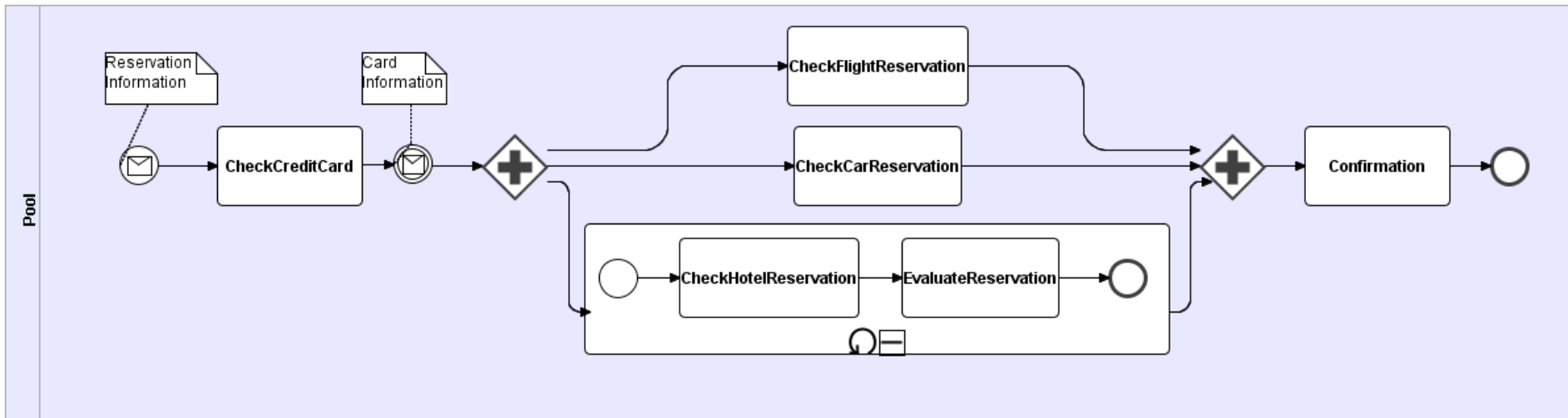


Association



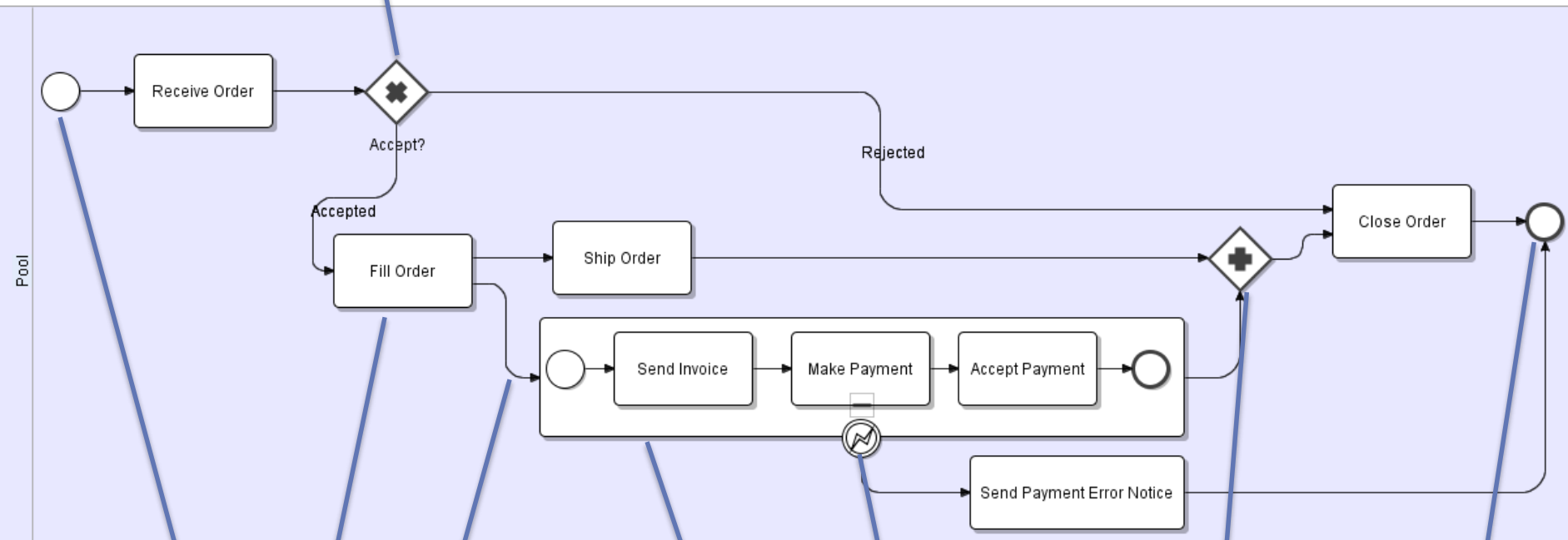
Example: Travel Process Control Flow

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



Example

XOR split Gateway



Pool

Start Event

Task

Sequence Flow

SubProcess

Error Intermediate Event

AND join Gateway

End Event



Why BPMN?

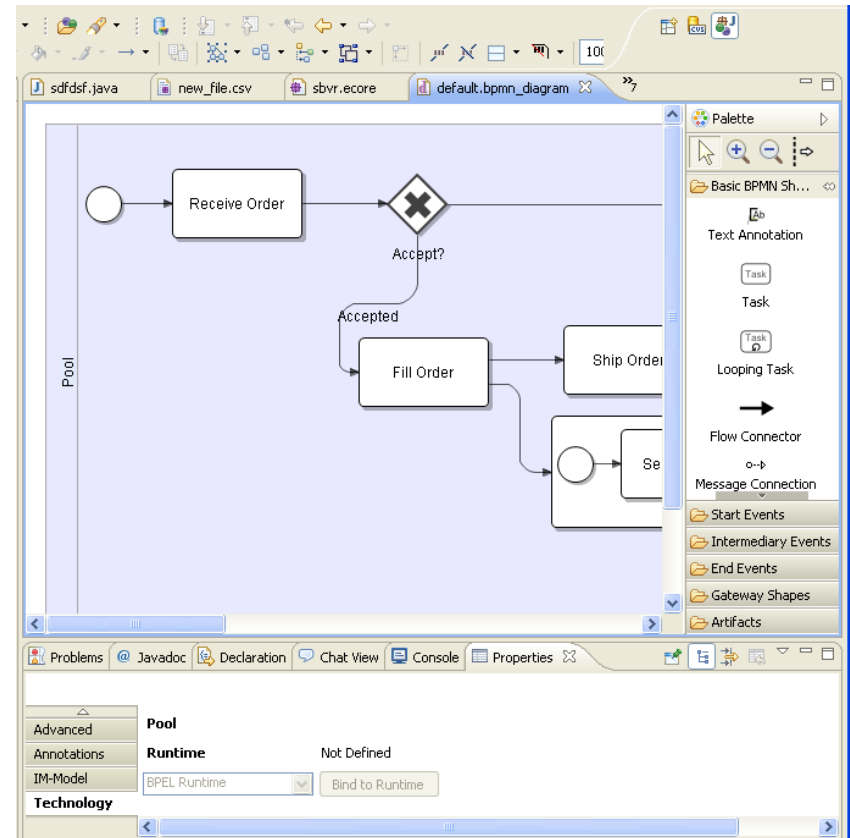
- BPMN 2.x
 - Modeling language for business processes: no execution semantics, only a partial mapping to Business Process Execution Language (BPEL)
 - Explicit service mapping to web services (as components)
 - Engines are available (jBPM for jBoss)
- BPMN geared towards business analysts:
 - BPMN constructs are simplified
 - UML notation too bloated
 - BPMN is on the platform-independent level, BPEL nearer the platform-specific level



Give BPMN a try

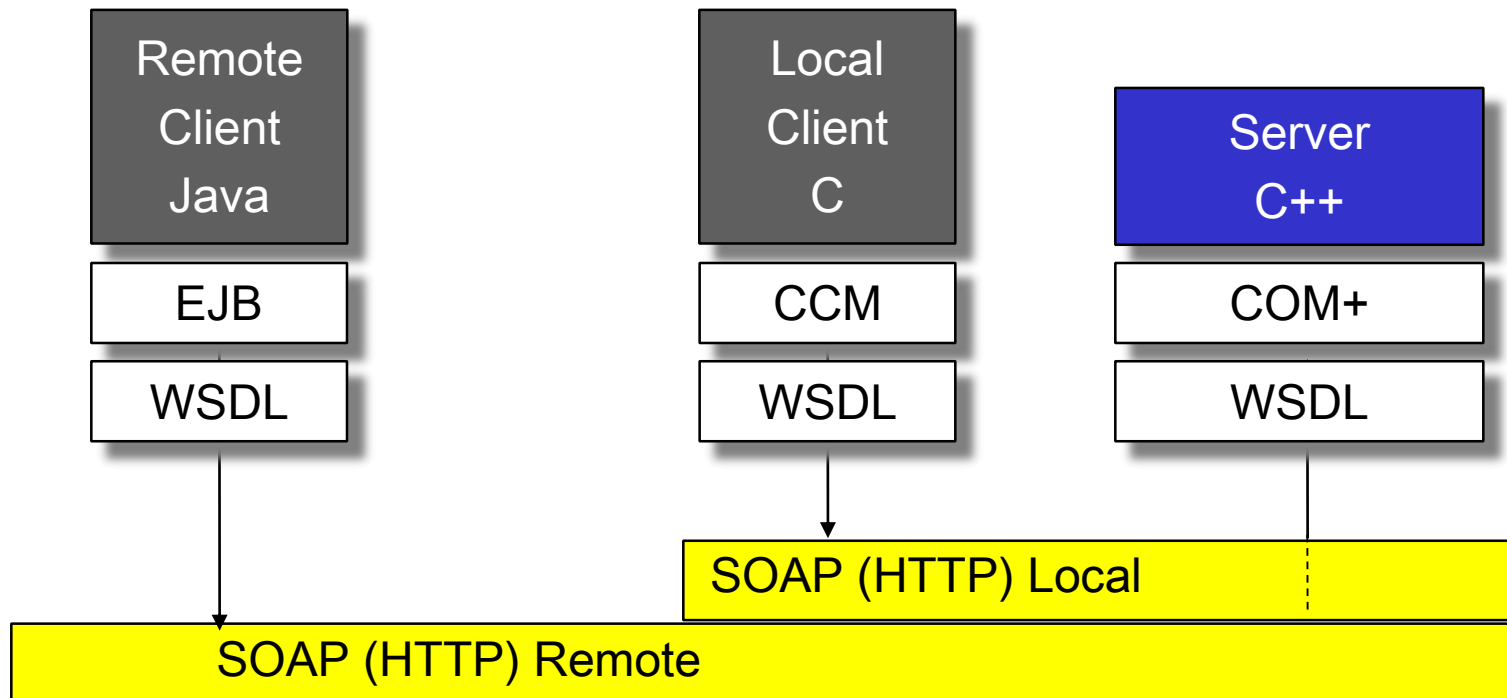
Component-Based Software Engineering (CBSE)

- Free BPMN Editor from Eclipse
- Included in the SOA Tools Project
- <http://www.eclipse.org/bpmn/>
- SAP has decided to use BPMN in their products



Web Services – Component Model Transparency

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



32.5 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 and USDL
 - ▶ 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



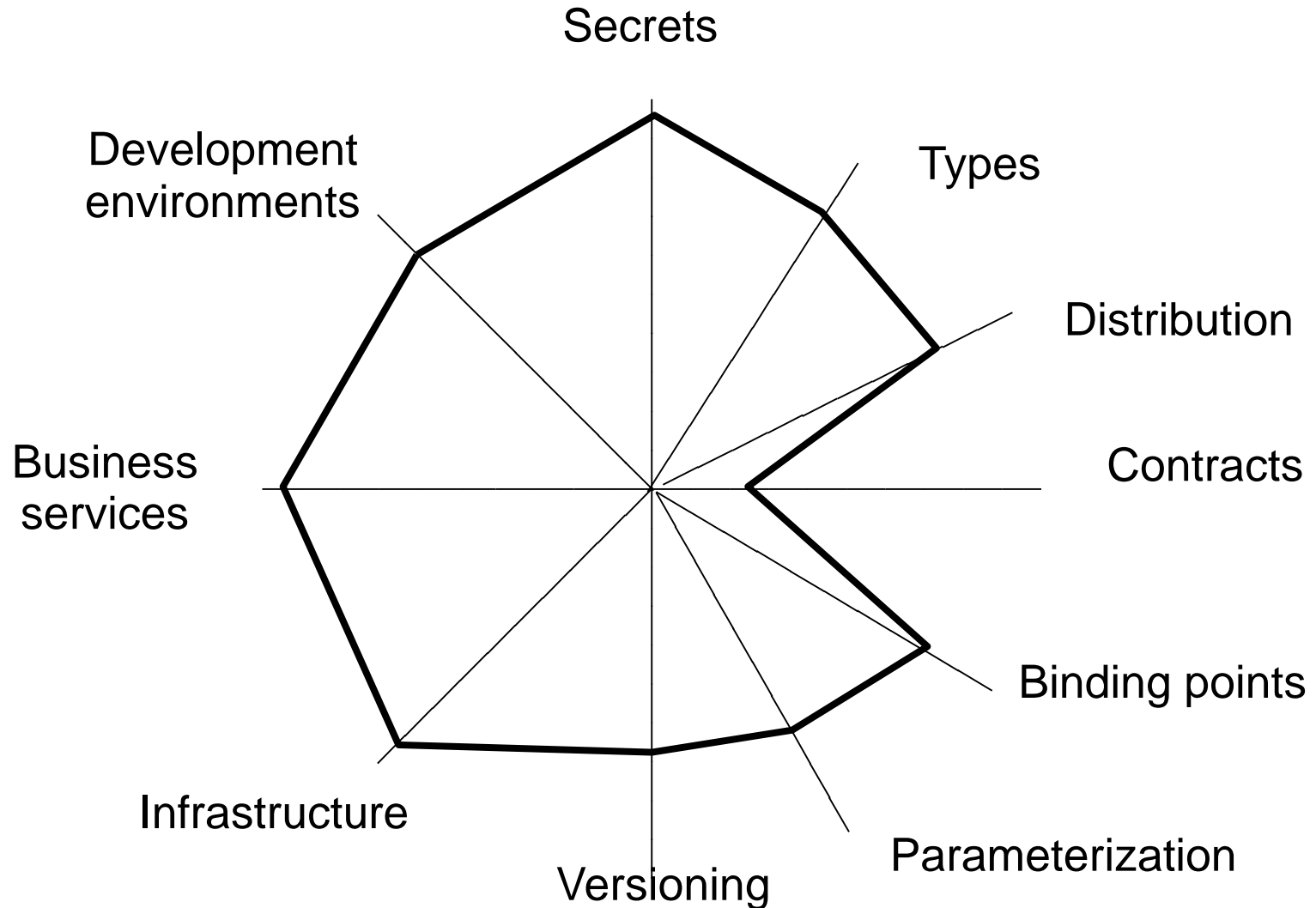
Composition Language

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



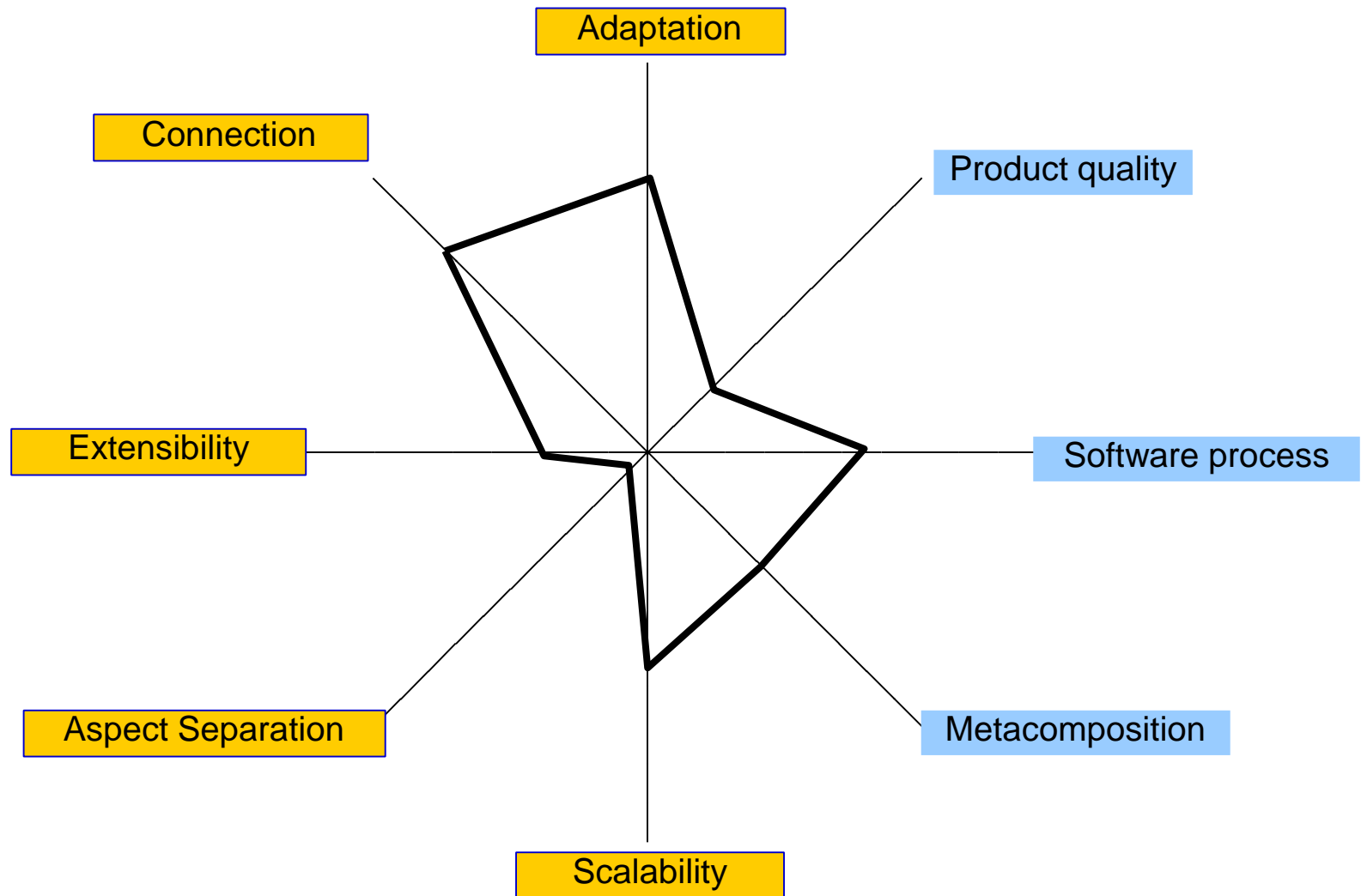
Web Services - Component Model

Component-Based Software Engineering (CBSE)



Web Services – Composition Technique and Language

Component-Based Software Engineering (CBSE)



Web Services as Composition Systems

Component-Based Software Engineering (CBSE)

Component Model

Contents: Completely hidden
Binding points: WSDL ports,
USDL quality specs

Composition Technique

Adaptation: well supported
Automatic transactions, recovery
Several types of connectors

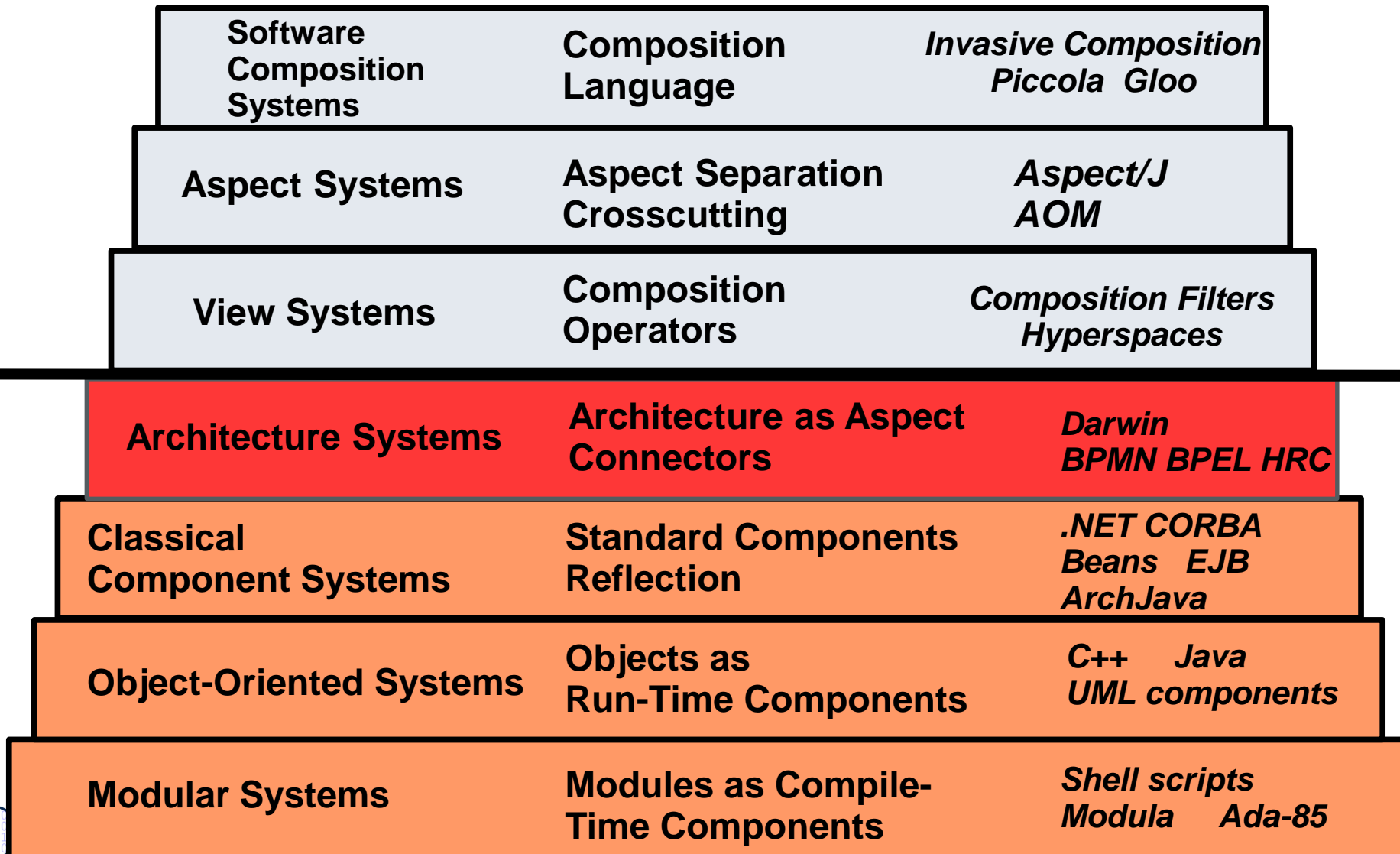
BPEL BPMN etc.

Composition Language



So Far: Blackbox Composition Systems

Component-Based Software Engineering (CBSE)



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

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



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



32.6 OWL-S (Web Ontology Language for Services)

Additional material

- ▶ OWL-S definition at <http://www.w3.org/Submission/OWL-S/>

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