Service-oriented middleware

Contents

- Associated terminology
  - Service, SOA, SOC, service science, web services
- Service oriented architecture and facilities
  - Layers, WS facilities: SOAP, WSDL, UDDI, BPEL
- Impact of SOC

Service

- A service is a provider/client interaction that creates and captures value
- A business serivice is a coherent compositon of functionalities, provided by an autonomous service provider, through published interfaces in an open network environment; it can be independently administered through business rules and policies
- viewpoints: users and business vs. computing

Service domain in business

- Service sector is important in post-manufacturing countries
  - 70 – 80 % of economy
- Service science
  - emerging discipline in business schools adressing marketing, customer relation, operations research, business, economics
  - requirements towards ICT

Service computing

- Service oriented architecture SOA
- Service oriented computing SOC
- role
  - facilitating the communication, storage and processing of informaiton
  - benefits: cost, user-base

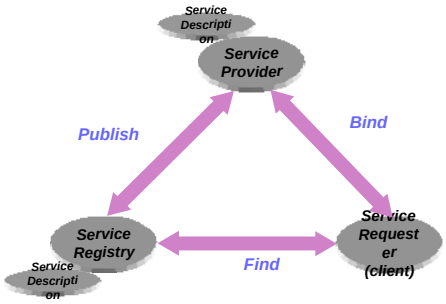
SOA

- a paradigm for organising and utilising distributed capabilities that may be under the control of different ownership domains.
- provides univorm means to offer, discover, interact with and use capabilties to produce desired effects cosistent with measureable preconditions and expectations
- OASIS SOA RM

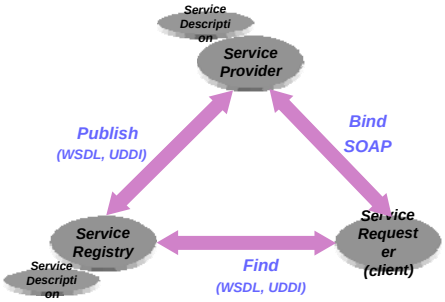
Web Services

- A web service is a software system designed to support interoperable machine-to-machine interactionover a netowrk.
- Web services are an example implementation of SOA

Service-oriented Architecture



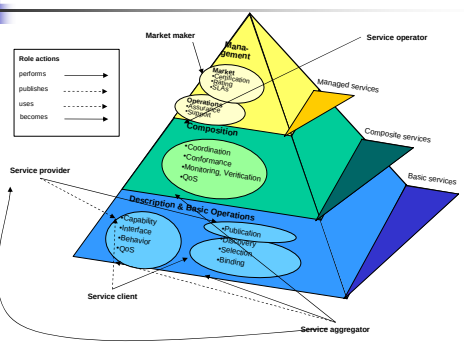
Service-oriented Architecture



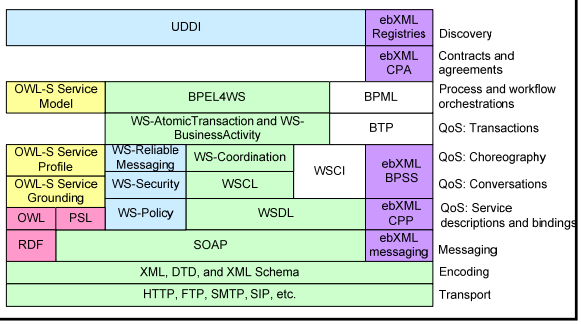
SOA principles

- Service encapsulation and abstraction; statelessness?
- Service autonomy
- Service reusability
- Service discoverability
- Service composability
- Loose coupling and service contract

Extended Service-oriented Architecture



Standards for Web Services



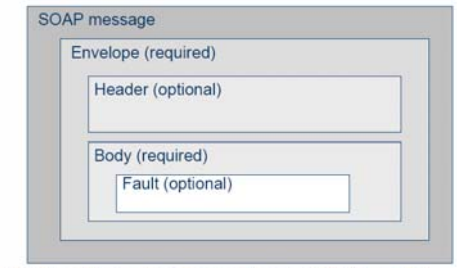
SOC and Web Service stack

Process	BPEL4WS, WSCI, WS-CDL
Discovery	UDDI
Description	WSDL
Messaging	SOAP (XML-RPC)
Transport	HTTP

SOAP

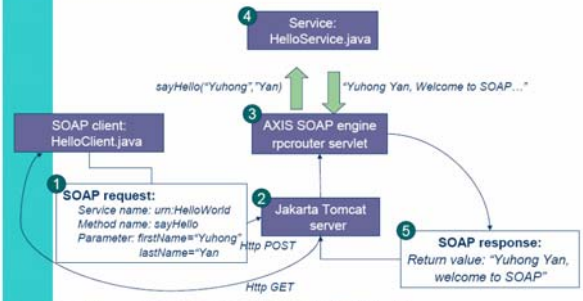
- Simple Object Access Protocol
- a lightweight protocol for exchange of information in a decentralised, distributed environment
- inter-application communication
- platform-independence

SOAP example



From Etban Cerami, "Web Services Essentials", P53, figure 3-2

Apache SOAP architecture



From Etban Cerami, "Web Services Essentials", p69,. Fig 4-3

HelloWorldService.java

```
package samples.HelloWorld;

public class HelloWorldService
{
  public String sayHello(String firstName, String lastName) throws Exception
  {
     String aString = firstName + " " + lastName + ", welcome to SOAP Web
     Service World!";
    return aString;
  }

  public String addString(String symbol, String dataType) throws Exception
  {
     String aString = symbol + dataType;
     return aString;
  }
}
```


TestClient

```
public class TestClient
{
  public static void main(String args[])
  {
    try
    {
      Options opts = new Options( args );
      args = opts.getRemainingArgs();

      Service  service = new Service();
      Call     call    = (Call) service.createCall();
```

Parse arguments

Prepare to call remote service

TestClient.java (2)

```
call.setTargetEndpointAddress( new java.net.URL(opts.getURL()) );
if( args[0].equals("1") )
    call.setOperationName( new QName("urn:HelloWorld", "sayHello"));
else
    call.setOperationName( new QName("urn:HelloWorld", "addString"));

call.addParameter( "p1", XMLType.XSD_STRING,
        ParameterMode.IN );
call.addParameter( "p2", XMLType.XSD_STRING,
        ParameterMode.IN );
```

Set the URL of the remote service

Set the URL of the remote service

Service Name / *Service Method*

Add Paras for the remote methods

Paras Name / *IN/OUT/INOUT* / *XML data type*

TestClient.java (3)

```
call.setReturnType( XMLType.XSD_STRING );

call.setUsername( opts.getUser() );
call.setPassword( opts.getPassword() );

String res = (String) call.invoke( new Object[] { args[1], args[2] } );
System.out.println( "Return is: " + res );
  }
  catch( Exception e )
  {
   e.printStackTrace();
  }
 }
}
```

Define the return value

Security options

Invoke Service

Pass two paras

Request SOAP message

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:sayHello
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="urn:HelloWorld">
   <p1 xsi:type="xsd:string">Yuhong</p1>
   <p2 xsi:type="xsd:string">Yan</p2>
  </ns1:sayHello>
 </soapenv:Body>
</soapenv:Envelop
```

Remote method name

Parameter name

Name of the web service

Value of the parameter

Parameter type. need to match WSDL.

Response SOAP message

```
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:sayHelloResponse
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="urn:HelloWorld">
   <ns1:sayHelloReturn xsi:type="soapenc:string"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
      Yuhong,Yan, welcome to SOAP Web Service World!
   </ns1:sayHelloReturn>
  </ns1:sayHelloResponse>
 </soapenv:Body>
</soapenv:Envelope>
```


SOAP Encoding

- Scalar Type: map to simple types in XML Schema

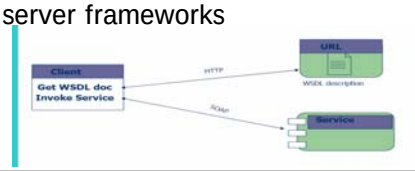
```
<return xsi:type="xsd:double" > 54.99 </return>
```


WSDL: Web Services Description Language

- Describes a programmatic interface to a Web service, including
  - Definitions of data types
  - Input and output message formats
  - The operations provided by the service
  - Network addresses
  - Protocol bindings

WSDL

- a contract between requestor and provider
- description of SOAP services
- automated tools to generate client and server frameworks



Structure of WSDL

<definitions>:Root WSDL Element

Service Implementation Definition	<service>:Where is the service located?
	<port>: Where is the service located?
Service Interface Definition	<message>:What message will be transmitted?
	<portType>:What operations (functions) will be supported?
	<binding>:How will the messages be transmitted on the wire? What SOAP-specific details are there?
	<types>:What data types will be transmitted?

From Etban Cerami, "Web Services Essentials", P121 fig 6-1

An example of WSDL: HelloWorld

```
<wsdl:definitions
targetNamespace="http://localhost:8080/axis/services/urn:Hello
World" xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/axis/services/urn:HelloWorld"
xmlns:intf="http://localhost:8080/axis/services/urn:HelloWorld"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://www.w3.org/1999/XMLSchema"
xmlns:tns2="http://www.w3.org/2003/05/soap-encoding"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```


<Service> : where the service is located

```
<wsdl:service name="HelloWorldServiceService">
 <wsdl:port
  binding="impl:urn:HelloWorldSoapBinding"
  name="urn:HelloWorld">
 <wsdlsoap:address
  location="http://localhost:8080/axis/services/urn:H
  elloWorld" />
 </wsdl:port>
</wsdl:service>
```

The service location: the endpoint of the

<Binding>: bind messages to operations

```
<wsdl:binding name="urn:HelloWorldSoapBinding"
 type="impl:HelloWorldService">
<wsdlsoap:binding style="rpc"
 transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="sayHello">
<wsdlsoap:operation soapAction="" />
<wsdl:input name="sayHelloRequest">
<wsdlsoap:body
 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 namespace="http://HelloWorld.samples" use="encoded" />
</wsdl:input>
<wsdl:output name="sayHelloResponse">
<wsdlsoap:body
 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 namespace="http://localhost:8080/axis/services/urn:HelloWorld"
 use="encoded" />
</wsdl:output>
</wsdl:operation>
```

Use SOAP HTTP protocol

<portType> : operations supported

```
<wsdl:portType name="HelloWorldService">
 <wsdl:operation name="sayHello"
  parameterOrder="in0 in1">
  <wsdl:input message="impl:sayHelloRequest"
   name="sayHelloRequest" />
  <wsdl:output
   message="impl:sayHelloResponse"
   name="sayHelloResponse" />
 </wsdl:operation>
</wsdl:portType>
```

Input Message

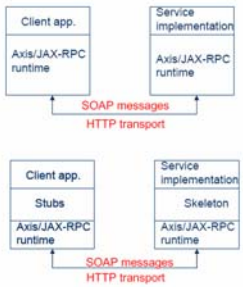
Output Message

<message>: define content of message

```
<wsdl:message name="sayHelloRequest">
 <wsdl:part name="in0" type="tns2:string" />
 <wsdl:part name="in1" type="tns2:string" />
 </wsdl:message>
<wsdl:message name="sayHelloResponse">
 <wsdl:part name="sayHelloReturn"
  type="tns2:string" />
 </wsdl:message>
```


Three Ways for Client to Invoke a Service

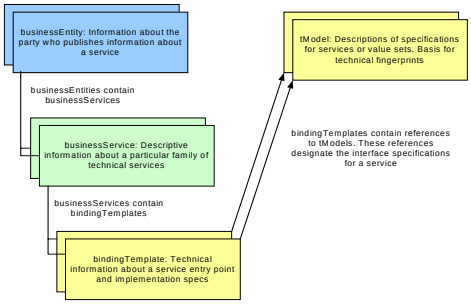
- Dynamic invocation interface (DII) (bottom right).
  - No stubs/ties; WSDL not necessary.
  - More coding
  - Flexible.
- Using static stubs (top right).
  - Tool creates stubs and ties c.f. RMI; client operates on proxy (stub).
  - Less code than DII.
- Dynamic proxy.
  - No stubs, client creates proxy class at runtime from WSDL.
  - More portable than static stubs.



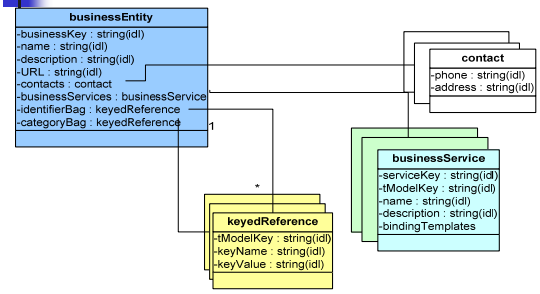
UDDI

- Universal Description, Discovery and Integration
- White pages
  - locations of the business, contacts, uniq ids
  - <businessEntity>
- Yellow pages
  - categorised by taxonomy etc.
  - <businessService>
- Green pages
  - technical information: location, category, spec of service
  - <businessService> and <bindingTemplate>

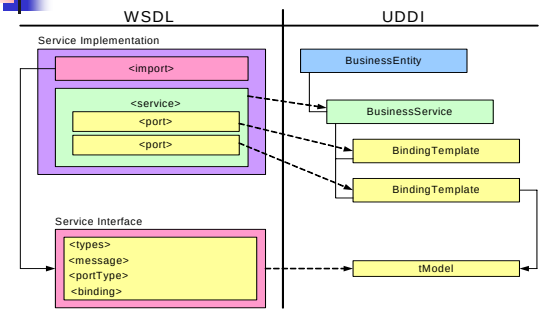
Core Data Structures for UDDI



Yellow, Green, and White Pages in UDDI



WSDL ←→ UDDI Correspondence



Process description

- Orchestration
  - BPEL4WS
- Choreography
  - WSCI
- How to model
  - control flow
  - data flow
- Workflow
  - hybrid: human/software
  - collab working environment
  - process descriptions similar to orchestration languages
  - WS process engine can be similarto workflow engine

Process Abstractions

Orchestration: views a process as a partial order of actions under the control of a central conductor; akin to a workflow

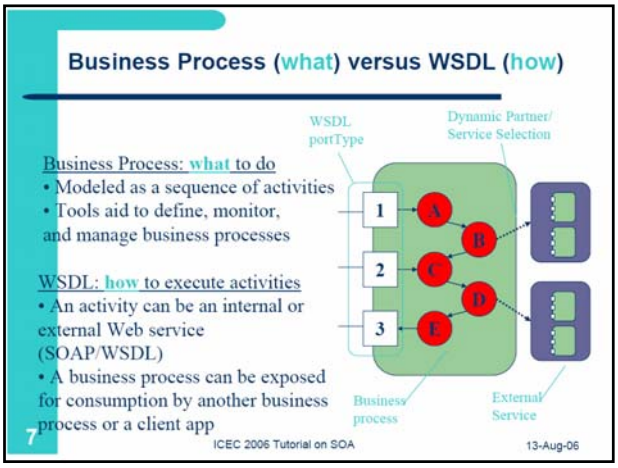
Choreography: views a process as an exchange of messages among participants; akin to a conversation as described by WSCL and WSCI

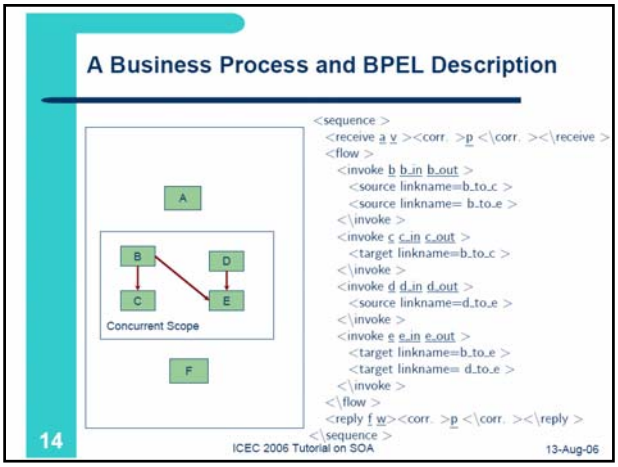
Collaboration: views a process as a joint set of activities among business partners

Workflow: a narrower concept than a process, which emphasizes control flows and data flows from a central perspective

BPEL4WS

- Business process execution language for Web Service
  - execution order of the activities
  - triggering conditions
  - partners for external activities
  - composition of WSs
  - binding to WSDL (activities / operations)





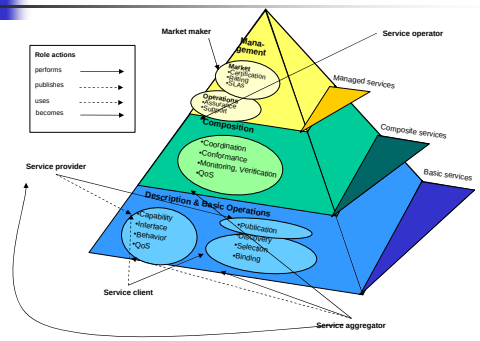
WSCI

- Web Service Choreography Interface
  - flow of messages
  - observable behaviours of WS, no internal
- WSDL comparison
  - does not describe execution order of operations
  - does not describe correlations

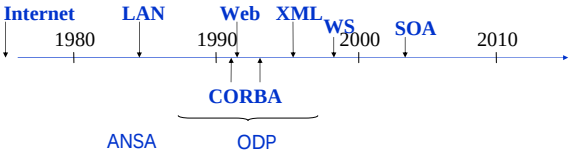
What is the difference?

- BPEL activities
  - atomic: receive, reply, invoke, assign, throw, terminate, wait, empty
  - structured: sequence, flow, switch, while, pick
  - fault handlers
  - link for activity dependencies
  - data flow by variables
- WSCI activities
  - atomic: action (mapping to WSDL operation), delay, empty, fault, call, spawn, join
  - complex: process, all, choice, foreach, sequence, switch, until, while
  - variables to map SOAP messages similar to BPEL

Impact of SOC



Timeline



Connectivity Drives the Emergence and Convergence of Technologies

Paradigm problems

- Structured programming, object-oriented, component-oriented paradigms
- Programming paradigms direct the application achitectures
- Difficulty to reach across organisational boundaries
- Syncrony required by models

Open Environments: Characteristics

- Cross enterprise boundaries
- Comprise autonomous resources that
  - Involve loosely structured addition and removal
  - Range from weak to subtle consistency requirements
  - Involve updates only under local control
  - Frequently involve nonstandard data
- Have intricate interdependencies

Autonomy

Independence of business partners (users)

- Political reasons
  - Ownership of resources
  - Control, especially of access privileges
  - Payments
- Technical reasons
  - Opacity of systems with respect to key features, e.g., precommit

Heterogeneity

Independence of component designers and system architects

- Political reasons
  - Ownership of resources
- Technical reasons
  - Conceptual problems in integration
  - Fragility of integration
  - Difficult to guarantee behavior of integrated systems

Dynamism

- Independence of system administrators
- Needed because the parties change
  - Architecture and implementation
  - Behavior
  - Interactions
- Make configurations dynamic to improve service quality and maintain flexibility

Locality

- Global information (data, schemas, constraints) causes
  - Inconsistencies
  - Anomalies
  - Difficulties in maintenance
- Global information is essential for coherence
  - Locations of services or agents
  - Applicable business rules
- Relaxation of constraints works often
  - Obtain other global knowledge only when needed
  - Correct rather than prevent violations of constraints: often feasible
  - When, where, and how of corrections must be specified, but it is easier to make it local

Applications of Composable Services

- Portals
  - Organized by topic or affinity
  - Best when personalized
- E-commerce
- Legacy system integration
- Virtual enterprises
- Grid computing

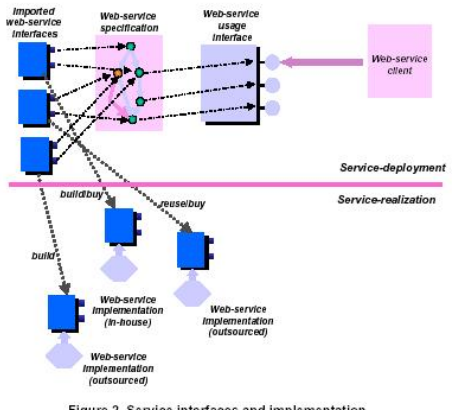
Shift To A Service-Oriented Architecture

- Function oriented
- Build to last
- Prolonged development cycles
- Application silos
- Tightly coupled
- Object oriented
- Known implementation

- Coordination oriented
- Build to change
- Incrementally built and deployed
- Enterprise solutions
- Loosely coupled
- Message oriented
- Abstraction

Gained by … services

- services are self-describing, open components
  - support low-cost composition
- service providers
  - organisations that procure service implementations
  - provide service descriptions
  - provide runtime environment
- service descriptors advertise
  - service capabilities
  - interface (signature) + behaviour (workflow)
  - QoS, security, availability etc
- technology neutral, support location transparency



… service composition layer

- loosely coupled services
- Coordination
  - control execution of component services
  - manage dataflow between them
- Conformance
  - interoperability testing, enforcement of business rules
- Monitoring, verification
  - subscribing to events or results of component services or composed service
- QoS
  - aggregation of QoS contracts from component service QoS

… service management layer

for service operator

- management of service platform
- deployment of services/applications
- assesment of application performance

for open service market

- directories, domain specific business protocols
- view of products and services, business terminology, detailed business process descriptions
- negotiation of SLA (service level agreement)

Web services standards

- Bring together well-known ideas
- Provide necessary functionality for interoperation
- Are complicated in their details
- Meant for tool vendors rather than programmers
- Increasingly hidden by tools

Description

The description should be unambiguous, formal representations of

- A service's functionality
- A service's nonfunctional attributes
- A user's needs and preferences

Engagement

- Architecture: P2P, messaging
- Transactions: replications, recovery
- Coordination
- Workflows and processes

Collaboration

- Reasoning
- Consistency maintenance
- Negotiation
- Organizational modeling
- Protocols, interaction patterns
- Contracts, monitoring, and compliance

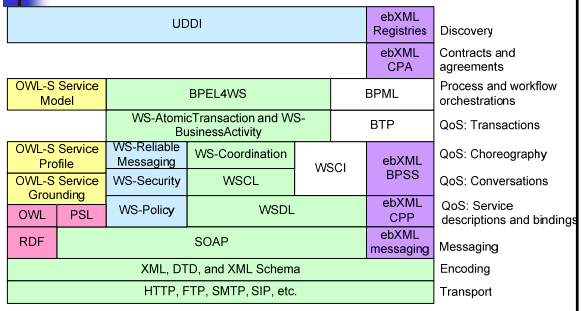
Discovery and Selection

- Semantic matchmaking
- Team matchmaking
- Economic selection
- Reputation and recommendation
- Distributed architectures
- Accommodating quality of service
- Trust

Engineering

- Methodologies
  - Ontologies
  - Process models
- Service Management
  - Administrative
  - Scalability
- Security

Standards for Web Services



Conclusion: Impacts of the emergence of SOA

Business changes

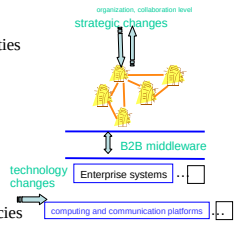
→ New business network models, opportunities
→ IT investment style changes
→ More open service markets
→ Regulatory involvement

Software development cycle and tools

→ Continuous process of composition and management
→ configurability by business rules and policies

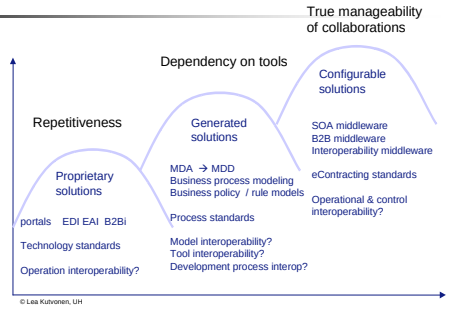
Architectural changes for IT

→ Enterprise system architectures
→ Middleware stacks



25.09.07

Conclusion: Trends



25.09.07