INF5120
Modellbasert systemutvikling

- Interoperability
- UML Profile for EDOC
- Java 2 Enterprise Edition (J2EE)
- Service-Oriented Architectures
- Web Services

Forelesning 14.04.2005
Brian Elvesæter

Interoperability
Diskusjonsoppgave

- Sett dere sammen to-og-to og diskuter begrepet "interoperabilitet"

Interoperability

- **Interoperability (Def)** is "the ability of two or more systems or components to exchange information and to use the information that has been exchanged"
  - IEEE Standard Computer Dictionary
  - Carnegie Mellon, Software Technology Roadmap, 2004

- Enterprise systems and applications **need to be interoperable** to achieve seamless operational and business interaction, and create networked organizations
  - European Group for Research on Interoperability, 2002

Interoperability, key to increase competitiveness of enterprises

- The cost of non-interoperability are estimated to 40% of enterprises IT budget.

(Source: the Yankee Group 2001)
Integration and Networked Enterprises

Vertical integration

People and Operations
Software Logic, Middleware, and Protocols
Communication Networks

Horizontal integration

Sensors and Actuators
Mobile Operational Facility (e.g., COP)
Remote Operational Terminal (PC, PDA...)

ICT Integration and Networked Enterprises

- People and Operations
- Software Logic, Middleware, and Protocols
- Communication Networks

SINTEF

Integration and Networked Enterprises

Vertical integration

People and Operations
Software Logic, Middleware, and Protocols
Communication Networks

Horizontal integration

Sensors and Actuators
Mobile Operational Facility (e.g., COP)
Remote Operational Terminal (PC, PDA...)

ICT Integration and Networked Enterprises

- People and Operations
- Software Logic, Middleware, and Protocols
- Communication Networks

SINTEF

Integration and Networked Enterprises

Vertical integration

People and Operations
Software Logic, Middleware, and Protocols
Communication Networks

Horizontal integration

Sensors and Actuators
Mobile Operational Facility (e.g., COP)
Remote Operational Terminal (PC, PDA...)

ICT Integration and Networked Enterprises

- People and Operations
- Software Logic, Middleware, and Protocols
- Communication Networks

SINTEF

Integration and Networked Enterprises

Vertical integration

People and Operations
Software Logic, Middleware, and Protocols
Communication Networks

Horizontal integration

Sensors and Actuators
Mobile Operational Facility (e.g., COP)
Remote Operational Terminal (PC, PDA...)

ICT Integration and Networked Enterprises

- People and Operations
- Software Logic, Middleware, and Protocols
- Communication Networks

SINTEF

Integration and Networked Enterprises

Vertical integration

People and Operations
Software Logic, Middleware, and Protocols
Communication Networks

Horizontal integration

Sensors and Actuators
Mobile Operational Facility (e.g., COP)
Remote Operational Terminal (PC, PDA...)

ICT Integration and Networked Enterprises

- People and Operations
- Software Logic, Middleware, and Protocols
- Communication Networks

SINTEF
To achieve meaningful interoperability between enterprises, interoperability must be achieved on all layers:

- Business layer: business environment and business processes
- Knowledge layer: organisational roles, skills and competencies of employees and knowledge assets
- ICT layer: Applications, data and communication components
- Semantic dimension: support mutual understanding on all layers
The Waves of Client/Server Technology

First Wave
- File Servers
- Database Servers
- Groupware
- TP Monitors

Second Wave
- Distributed Objects
- OMG CORBA
- COM/OLE
- Web/Internett
- Java

Third Wave
- Server-side components
- J2EE/EJB
- COM+
- Corba
- Comp Agents, P2P
- XML

Fourth Wave
- MDA, Web Services, Net Service-oriented Architecture
- SOAP, XML
- WSDL/WSFL
- Service-oriented Architecture
- SOAP, XML
- WSDL/WSFL
- Service-oriented Architecture
- SOAP, XML
- WSDL/WSFL
- Service-oriented Architecture

Fifth Wave
- P2P
- Grid
- Agents, FIPA
- Multi Media, QoS

Abstract Technology Model

Interaction/Pres services

User Interface
Document model
Web interaction

XML

Workflow service

System/Use Mngt

Server Components

Concurrency service

Transaction service

Persistence service

Data services & Legacy systems

Integration service

Shared Business Services

User services (application/process)

Event - publish & subscribe

Security service

Trading service

Naming service

Streaming

Message

Deferred Synch request

Synchron. request

Base Source: Client/Server Survival Guide, 1994
Robert Orfali, Dan Harkey
Enterprise, Conceptual & ICT Platform levels

Enterprise Model (EM, CIM)

Platform independent Model (PIM)

PSM execution platform (web services, bpmf, bpai, .net, J2EE)

Platform Specific Model

AKM ii - EM execution platform

Execution platform (includes code and model-driven services from the level below)

Processes executed by platform

Ad-hoc, unique processes (partial automation)

Repetitive, adaptive routine processes

Standard, repetitive, static processes

Programming and code generation.

PSM services that are coded become part of the PIM execution platform, implemented PIM services become part of the AKM ii.

Repository

Enterprise, Conceptual & ICT Platform levels

Enterprise model

Platform independent Model (PIM)

AKM ii - EM execution platform

Enterprise model interoperability

Ontology-based semantic interoperability?

Web services (UDDI, SOAP)

Platform Specific Model

PIM execution platform

Network protocols

Interoperability Challenges

Enterprise model

Platform independent Model (PIM)

AKM ii - EM execution platform

Enterprise model

Enterprise model interoperability

Ontology-based semantic interoperability?

Web services (UDDI, SOAP)

Platform Specific Model

PIM execution platform

Network protocols

Interoperability objective

Integrate enterprise models across companies and EM tools

Exchange information despite semantic and syntactical incompatibility

Enable enterprises to invoke services (and processes packaged as services) from each other, and include remote services in local processes
ATHENA Research Challenges in spanning the Domains Modelling, Ontology and Architecture

Scientific & technology objectives*

- to define a technologically neutral reference model that provides a stable, generic foundation for specific technical innovations;
- to define interoperability requirements for applications, data and communications and provide solutions that meet these requirements;
- to provide methods which enterprises can use to manage organisational roles, skills, competencies, and knowledge assets for its own operation and for collaboration with other enterprises;
- to provide semantic mediation solutions which enable and support the above;
- to provide components of interoperability infrastructures

* objectives have been set for the 5 year programme
UML Profile for Enterprise Distributed Object Computing Specification


"A UML Profile is a predefined set of Stereotypes, Tagged Values, Constraints and notation icons that collectively specialize and tailor the UML for a specific domain or process (e.g., Unified Process profile). A profile does not extend UML by adding any new basic concepts. Instead, it provides conventions for applying and specializing standard UML to a particular environment or domain."
Enterprise Collaboration Architecture (ECA)

- ECA is a model-driven architecture approach for specifying Enterprise Distributed Object Computing (EDOC) systems.
  - recursive collaboration
  - different levels of granularity, for both business and systems modeling
  - loosely and tightly connected systems
  - synchronous and asynchronous communication
  - container managed and message-based architectures.
**ECA Models**

- **Component Collaboration Architecture (CCA)** details how to model, at varying and mixed levels of granularity, the structure and behaviour of the components that comprise a system.
- **Entities model** describes how to model entity objects that are representations of concepts in the application problem domain and define them as composable components.
- **Events model** describes a set of model elements that may be used on their own, or in combination with the other EDOC elements, to model event driven systems.
- **Business Process model** specializes the CCA, and describes a set of model elements that may be used on their own, or in combination with the other EDOC elements, to model system behaviour in the context of the business it supports.
UML for EDOC

- **CCA**
  - ProcessComponent Structural specification (Component/Protocol)
  - Choreography, Composition, Document Model
  - Model Management, CCA Notation and UML 1.4 Notation
- **Entities profile**
  - Entity component, Data Manager - Composite data
  - Identity, Entity Data, Key, Entity role (ad hoc extension), Data Probe
  - Events (Publication/Subscription flow ports), Information viewpoint, Composition viewpoint
- **Events profile**
- **Business Processes profile**
  - BusinessProcess, CompoundTask, Activity, ProcessPortConnector,
    - ProcessFlowPort, DataFlow, InputGroup, OutputGroup, ProcessMultiPort
  - ProcessRole, Performer, Artifact, ResponsibleParty
- **Relationships profile**
- **Patterns profile**

ECA and viewpoints
ECA and RM-ODP viewpoints

- **Enterprise viewpoint**
  - (CCA, Process, Entity, Relationship, Event)

- **Information viewpoint**
  - (Entity, Relationship)

- **Computational viewpoint**
  - (CCA, Event)

- **Engineering viewpoint**
  - (DCP - Distributed Component Profile/Messaging)

- **Technology viewpoint**
  - (UML for J2EE/EJB/JMS, CORBA 3/CCM, COM, SOAP, ebXML, ...)

(Patterns - applied to all viewpoints)

---

The Marketplace Example

Mechanics Are Us
Buyer
GetItThere Freight
Shipper

Order
Conformation
Shipped
Shipped
Delivered

Physical Delivery

Process
Complete

Acme Industries
Seller

Status

ICT
CCA example: Community Process

Activity diagram shows protocol
Deriving provided and required Component Interfaces

Additional communication refinement can be done: (Conversation, client/server, notification, streaming, ...) + other DCP refinement

Drill down to sub-components & activities with events
Entities & Relationships

```
<<CompositeData>>
OrderCD
<<Entiry>>
+ idNumber : Integer
+ custId : Integer
+ companyName : String
+ address : AddressCD
+ items [1..n] : OrderItemCD

<<CompositeData>>
OrderConfirmationCD
<<Entiry>>
+ name : String
+ order : Order
+ total : Float
+ idString : String

<<CompositeData>>
AddressCD
<<Entiry>>
+ addressLine1 : String
+ addressLine2 : String
+ city : String
+ state : String
+ postalCode : String
+ country : String

<<CompositeData>>
OrderDeniedCD
<<Entiry>>
+ order : OrderCD

<<CompositeData>>
OrderItemCD
<<Entiry>>
+ productId : String
+ quantity : Integer

<<Entity>>
Address
<<Reference>>
+ addressLine1 : String
+ addressLine2 : String
+ city : String
+ state : String
+ postalCode : String
+ country : String

<<Entity>>
Customer
<<Reference>>
+ custId : Integer
+ companyName : String

<<Entity>>
Order
<<Reference>>
+ idNumber : Integer
+ customer : Customer

<<Entity>>
Product
<<Reference>>
+ productId : String

<<Assembly>>
Order
<<Entity>>
+ idNumber : Integer
+ items [1..n] : OrderItemCD

<<Reference>>
Order
<<Entity>>
+ orderConfirmed() : OrderConfirmationCD
+ orderDenied() : OrderDeniedCD

<<Reference>>
Order
<<Entity>>
+ order : Order

CPP: Collaboration Protocol Profile
CPA: Collaboration Protocol Agreement
BT: Business Transaction
BD: Business Document

Mapping to ebXML (Messaging/SOAP)

CPP: Collaboration Protocol Profile
CPA: Collaboration Protocol Agreement
BT: Business Transaction
BD: Business Document

Ref: www.ebxml.org
Mapping to DCP - Distributed Component Profile

Java 2 Enterprise Edition (J2EE)
Java 2 Enterprise Edition (J2EE) ble utviklet for å adressere problemene rundt utvikling, deployering og håndtering av flerlags systemløsninger.

En applikasjonstjener tilbyr kjøretidsinfrastrukturen og tjenestene som er nødvendige for å deployere applikasjoner og komponenter i en distribuert omgivelse.
J2EE API spesifikasjoner

- **Enterprise JavaBeans (EJB):** Komponentarkitektur for å bygge gjenbrukbare tjenerkomponenter.
- **JDBC:** Java-grensesnitt mot relasjonsdatabaser
- **Java Remote Method Invocation over the Internet-ORB Protocol (RMI-IIOP):** Fjernmetodeinvokering mellom virtuelle Java maskiner. IIOP protokollen tillater integrasjon mot andre programmeringsspråk.
- **Java Message Service (JMS):** Asynkron kommunikasjon ved hjelp av meldinger.
- **Java Interface Definition Language (Java IDL):** En Java CORBA ORB som implementerer et subsett av CORBA spesifikasjonen.
- **Connectors:** Aksessintegrasjon mot andre informasjonssystemer som CICS, Tuxedo, SAP R/3 og PeopleSoft.

---

J2EE API spesifikasjoner (forts.)

- **JavaServer Pages (JSP):** Dynamisk generering av websider.
- **Java Servlets:** Servlets er komponenter som deployeres på en webtljener.
- **Java Transaction API (JTA):** Transaksjonstjeneste.
- **eXtensible Markup Language (XML):** En Java XML API som gir et grensesnitt mot en XML parser.
- **JavaMail:** Elektronisk mail.
- **JavaBeans Activation Framework (JAF):** Automatisk aktivering av Java komponenter for å håndtere ulike objekter.
Java 2 Enterprise Edition (J2EE)

Enterprise JavaBean (EJB)
Goals of EJB

- Standard component architecture for distributed business applications written in Java
- Easy to write applications. Hide details of such areas as transaction and state management, multi-processing, resource pooling etc.
- Maintain the “Write-once, run anywhere” philosophy of Java
- Compatible with other programming languages than Java
- Compatible with CORBA

EJB 2.0

- Integration of Enterprise JavaBeans with the Java Message Service™ (JMS).
- Improved Support for Persistence for Entity Beans.
- Support for Relationships among Enterprise JavaBeans.
- (Support for Inheritance and Subclassing of Enterprise JavaBeans. - not in 2.0)
- Query Syntax for Entity Bean Finder Methods
- Support for Additional Methods in the Home Interface
- Local Interfaces
- Mechanisms for Container Extensions
- EJB Server Network Interoperability Protocol
EJB Container Framework

- The server-side container provides the following to the enterprise Beans:
  - Component packaging and deployment (JAR, manifest, deployment descriptors)
  - Declarative transaction management (your component do not need to do any explicit transaction management)
  - Bean activation and passivation (bean is notified when it is loaded into memory and when it is deactivated from memory)
  - Bean state management
  - Container metadata
  - Security

What the container provides

- The diagram illustrates the view that a session container provides to its clients
The runtime objects

- At runtime there is three objects involved for each enterprise bean
- The home and remote objects are generated by the container, and forwards messages to the "real" bean

EJB - Session beans

- Characteristics
  - Executes on behalf of the client
  - Can be transaction-aware
  - Updates shared data in an underlying database
  - Does not represent directly shared data in the database
  - Is relatively short-lived
  - Is destroyed if the EJB server crashes - in such cases the client must re-establish a new session object to continue
EJB - Entity beans

- Characteristics
  - Represents data in the database
  - Is transactional
  - Allows shared access for multiple users
  - Can be long-lived (lives as long as the data in the database)
  - Survives crashes of the EJB server. A crash is transparent to the client

- Entity beans are persistent, allow shared access, and have primary keys.

Enterprise Bean’s home interface

- The home interface allows the client to:
  - create new EJB objects
  - remove existing EJB objects
  - get the metadata for the enterprise bean
  - find existing EJB objects (only for entity beans)

- The Home interface can be located using the Java Naming and Directory Interface (JNDI)
Enterprise Bean’s remote interface

- The remote interface defines the “business” methods of the enterprise bean.
- The methods defined in this interface must follow the rules for Java RMI. This means that their arguments and return values must be of valid types for Java RMI, and their throws clause must include the java.rmi.RemoteException.
- For each method defined in the remote interface, there must be a matching method in the enterprise Bean’s class. The matching method must have:
  - The same name.
  - The same number and types of its arguments, and the same return type.
  - All the exceptions defined in the throws clause of the matching method of the enterprise Bean class must be defined in the throws clause of the method of the remote interface.

EJB - Session and Entity Beans
Java 2 Enterprise Edition (J2EE)

UML Profile for EJB

UML for EJB
- from Java Community Process

- JSR - JAVATM SPECIFICATION REQUEST 26
- This document describes a standard mapping between the Enterprise JavaBeansTM architecture and the Unified Modeling Language. The mapping lets software designers use UML to describe software systems based on the EJB architecture.

- Supported by tools from Summer 2001
Virtual metamodel (VMM)
Physical Model

Example
Service-Oriented Architecture
What is a Service-Oriented Architecture (SOA)?

- “A set of components which can be invoked, and whose interface descriptions can be published, discovered and invoked over a network.” (W3C)
  - [http://www.w3.org/]

- “The policies, practices, frameworks that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer. Services can be invoked, published and discovered, and are abstracted away from the implementation using a single, standards-based form of interface.” (CBDI)
  - [http://www.cbdiforum.com/index.php3]

SOA Characteristics

- Well-defined interfaces
- Services usually represent a business function or domain.
  - Business services (business functionality)
  - Infrastructure services (“middleware services”)
- Modular design
  - Compositions and granularity
- Services are loosely coupled
  - Dynamic discovery and binding
- Services are standardized ("platform independent")
  - Using Internet/Web protocols and standards as the common "glue" provide "syntactical interoperability"
Service-Orientation

- Any service-oriented environment is expected to support several basic activities:
  1. Service creation
  2. Service description
  3. Service publishing to Intranet or Internet repositories for potential users to locate
  4. Service discovery by potential users
  5. Service invocation, binding
  6. Service unpublishing in case it is no longer available or needed, or in case it has to be updated to satisfy new requirements.

Service Model

Service provider: a service provider is the party that provides software applications for specific needs as services. Service providers publish, unpublish and update their services so that they are available on the Internet.

Service requester: a requester is the party that has a need that can be fulfilled by a service available on the Internet. A requester finds the required services via a service broker and binds to services via the service provider.

Service broker: provides a searchable repository of service descriptions where service providers publish their services and service requesters find services and obtain binding information for these services. Examples of service brokers are UDDI (Universal Description, Discovery, Integration).
Benefits of SOA

- The service concept applies equally well to the business as it does to software applications.
- Service orientation offers a level of flexibility far exceeding that of Component Based Development (CBD).
  - A component is built or bought once and integrated into an organisation’s application architecture.
  - A service is invoked dynamically when required, allowing providers to continuously improve their service and users to select the best available service at any one time.
- Services reduce complexity by encapsulation
- Services provide the ‘units of business’ that represent value propositions within a value chain or within business processes
- Services promote interoperability by minimizing the requirements for shared understanding
- Services enable interoperability of legacy applications

Web Services
What is a Web service?

- “Applications identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered as XML artefacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols.” (W3C)
  - [http://www.w3.org/]

- “Self contained, modular business applications that have open, Internet-oriented, standards-based interfaces.” (UDDI Consortium)
  - [http://www.uddi.org/]

Web services

- Web services are an accurate instantiation of the service-oriented model.
- They adhere to the set of roles and operations specified by the service oriented model.
- They have also managed to establish a standardized protocol stack.
### Overview of the Web service stack

<table>
<thead>
<tr>
<th>Composition/Choreography</th>
<th>Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPEL4WS, WSCI</td>
<td>WS−Coordination/Transaction</td>
</tr>
<tr>
<td>ebXML BPSS</td>
<td>OASIS BTP</td>
</tr>
<tr>
<td>Description</td>
<td>Advertisement/Discovery</td>
</tr>
<tr>
<td>WSDL, WS−Policy</td>
<td>UDDI, WS−Inspection</td>
</tr>
<tr>
<td>ebXML CPP/CPA</td>
<td>ebXML Registry</td>
</tr>
<tr>
<td>Messaging</td>
<td></td>
</tr>
<tr>
<td>SOAP, WS−Security, WS−ReliableMessaging, WS−Routing</td>
<td></td>
</tr>
<tr>
<td>ebXML Messaging Service</td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>Format and Encoding</td>
</tr>
<tr>
<td>HTTP, HTTPS, SMTP</td>
<td>Unicode, XML, XML Schema</td>
</tr>
</tbody>
</table>

---

### Web Service Description Language (WSDL)

- XML-based language for describing functional properties of Web services.
- A service consists of a collection of message exchange end points.
- An end point contains an abstract description of a service interface and implementation binding.
- The abstract description of a service contains:
  - (i) definitions of messages which are consumed and generated by the service
  - (ii) signatures of service operations.
- The implementation binding provides a means to map abstract operations to concrete service implementations.
  - It essentially contains information about the location of a binding and the communication protocol to use (e.g., SOAP over HTTP) for exchanging messages.
WSDL 1.1 Metamodel based on http://www.w3.org/TR/wsdl

WSDL Document
  0..1

WSDL Component
  0..1

Import
  NameSpace, Location

Service
  Name

Port
  Name

Port Type
  Name

Binding
  Name

Port Type
  Name

Definition
  Name, TargetNameSpace

Element
  Name, BaseType, MinOccurs, MaxOccurs

Include
  Location

Element
  Name

Message
  Name

Part
  Name, Type, Element

Include
  Location

Element
  Name

Definition
  Name, TargetNameSpace

0..1

Types

Schema
  TargetNameSpace

WSDL Document
  WSDL Component
  0..1

WSDL Component
  NameSpace, Location

Service
  Name

Port
  Name

Port Type
  Name

Binding
  Name

Port Type
  Name

Operation
  Name

0..1

Message

Part
  Name, Type, Element

Include
  Location

Element
  Name

Definition
  Name, TargetNameSpace

0..1

Types

Schema
  TargetNameSpace

WSDL Document
  WSDL Component
  0..1

WSDL Component
  NameSpace, Location

Service
  Name

Port
  Name

Port Type
  Name

Binding
  Name

Port Type
  Name

Operation
  Name

0..1

Message

Part
  Name, Type, Element

Include
  Location

Element
  Name

Definition
  Name, TargetNameSpace

0..1

Types

Schema
  TargetNameSpace
**Simple Object Access Protocol (SOAP)**

- XML-based protocol for structured message exchanges.
- It relies on existing transport protocols such as HTTP and MQSeries.
- A SOAP message contains two parts: the header and the body.
  - The header includes information such as intended purpose (e.g., service invocation, invocation results), sender's credentials, response type, and so on.
  - The body contains an XML representation of a service invocation request (i.e., name of operation to be invoked, values of input parameters) or response (i.e., results of service invocation).
- SOAP implementations exist for several programming languages including Java and C.

**XML - a Metameta Language**

- Metameta/How to define schema
  - XML
  - SGML
  - MathML
  - XSL
  - SMIL
  - OFX
  - HTML
- Meta/Schema
  - XML Doc
  - XSL Doc
  - HTML Doc
Universal Description Discovery and Integration (UDDI)

- Specification of an XML-based registry for Web services.
- Defines an interface for advertising and discovering Web services.
- The UDDI information model identifies three types of information
  - white pages, yellow pages, and green pages
- White pages contain general information such as business name (i.e., service provider's name) and contact information (e.g., provider's phone numbers).
- Yellow pages contain meta-data that can used to effectively locate businesses and services based on classification schemes.
- Green pages contain service access information including service descriptions and binding templates.
  - A binding template represents a service end point (i.e., a service access interface). It refers to an entity called the tModel.
Business Process Execution Language for Web Services (BPEL4WS)

- BPEL4WS – BPEL for short – is a language based on XML that allows for controlling the process flow of a set of collaborating Web services.
- It can be seen as a (business) extension to the Web services paradigm.
- Partner interaction is based on the notion of peer-to-peer interaction between Web services.
- BPEL introduces concepts to express the peer-to-peer conversational relationships between services.
- A partner link type models this relationship between two services with their role within the relationship and their port type.
- Partner links specify the services that a business process interacts with and is introduced as a WSDL extension element.

Security

- Authentication – Who is it?
- Authorisation – What can they do?
- Integrity – Ensure that information is intact.
- Signature – Create and verify electronic signatures
- Confidentiality – Make content unreadable by unauthorised parties.
- Privacy – Limit access and use of individually identifiable information
- Digital Rights Management – Limit use and sharing of content according to license agreements.
**WS-Security (by Microsoft/IBM)**

- **WS-License**
  - How to define a set of commonly used license types and to specify how they can be included within the WS-Security `<credentials>` tag
- **WS-Policy**
  - Specify security requirements, capabilities, constraints and policies on Web Services intermediaries and endpoints
- **WS-Trust**
  - Define security trust model allowing interoperation across security trust domains
- **WS-Privacy**
  - Define a model for web service clients and services to state privacy preferences and practices
  - The second phase is intended to include those specifications for meeting more advanced requirements, specifically:
- **WS-SecureConversation**
  - How to dynamically establish trust across trust domains using key exchange
- **WS-Federation**
  - How to manage identities and other information across heterogeneous federated systems
- **WS-Authorisation**
  - How to manage authorisation data and policies in a Web Services environment.

---

**Web services and the semantic Web**

- **OWL-S 1.0**
- **DAML-S 0.9**
- **RDF(S)**
- **XOL**
- **SHOE**
- **OML**
- **RDF Schema**
- **XML**
- **OWL Full**
- **OWL DL**
- **OWL Lite**
- **DAML-OIL**
- **DARPA DAML**
- **OIL**
- **RDF**
- **RDF(S)**
**Resource Description Framework (RDF)**

- Framework that enables describing and interchanging metadata with a few simple constructors.
- The most important feature of RDF is simplicity, so that provides a very well understood metadata structure for information modeling, based on three assumptions:
  - Resource: Anything that can have an URI such as a web site or a Web Service.
  - Property: A property of the thing that the statement describes.
  - Statement: A link between a resource and its property.
- Practically each statement is composed of three terms:
  - Subject: An URI that indicates a resource.
  - Predicate: The property of the subject.
  - Object: The value of the property.

**RDF example**

```
<rdf:Description rdf:about="http://thispaper">
  <s:Author>name of the author</Author>
</rdf:Description>
```
Web Ontology Language for Web Services (OWL-S)

- OWL-S is a W3C recommendation that provides a complete framework
  - based on the RDF syntax and the OWL ontology model for
  - describing Web Services in terms of what they can do and
  - how they can work
- OWL-S was born for solving challenges related to Web Services such as
  - semantic Web Services discovery
  - dynamic Web Services composition and
  - Web Services execution monitoring.

OWL-S model of service

The ServiceProfile has the similar functions of a registry, such as UDDI. It describes organizations that present the service, provides a list of the I/O parameters, preconditions and effects used by the service and describes additional information such as the quality and the accuracy.

The ServiceModel provides the fundamental information needed for the composition and interoperation of services. OWL-S allows three different kinds of processes: Atomic, Simple and Composite.

The ServiceGrounding specifies explicitly the input and output links between the atomic processes of the service, showing how the communications between these processes are to be realized as messages. However, the real implementation of the Service Grounding is a set of elements which links each atomic process to the corresponding WSDL file.