

Universität
Karlsruhe

Languages for (Semantic) Service Description

WSDL, OWL-S, WSMO, DSD

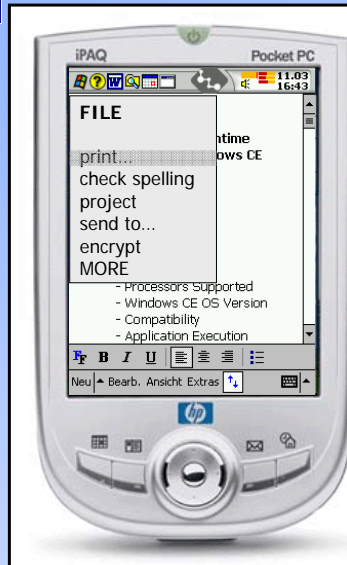
Birgitta König-Ries, Michael Klein
IPD, Universität Karlsruhe

Dagstuhl
October, 2004

1



Vision: "Really Mobile Office"



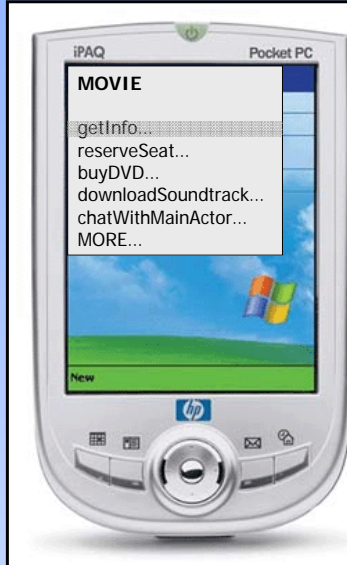
- Overcome limited computing capabilities
- Overcome missing attached devices
- Overcome missing internet connections

→ Do operations with your mobile device as though you sat at your desktop PC

2



Vision: Office for Real World

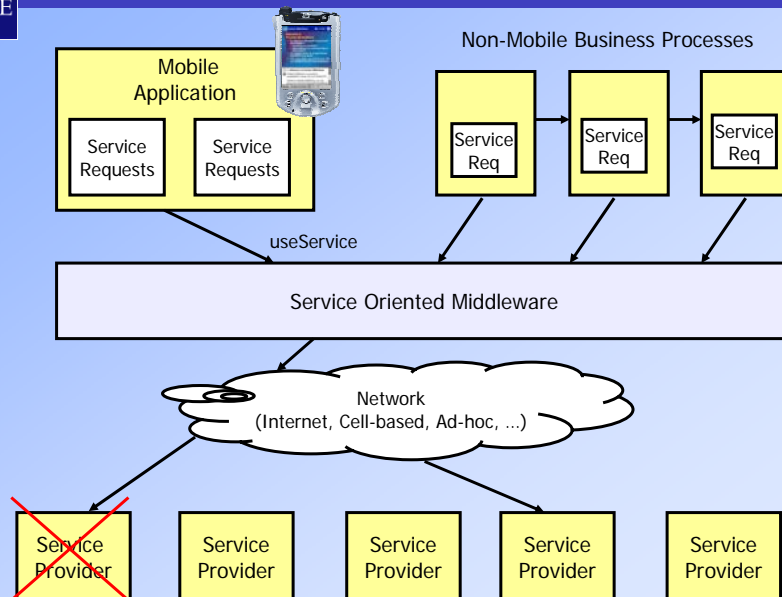


- Also affect the real world!

3



Service Oriented Architecture



4



Service Oriented Computing

- Network not as collection of documents but as **collection of functionality**
- Applications and business processes use functionality of other entities
- **loosely** coupled / reusable components
- especially interesting:
 - in **mobile networks** with limited capabilities
 - in the internet: large companies, many services

5



Characteristics of SOC

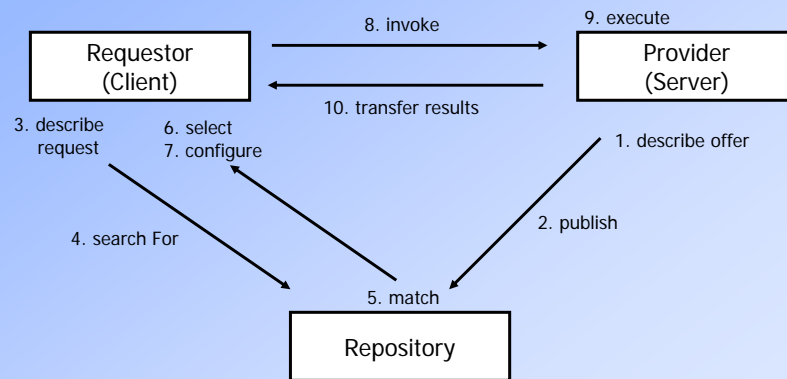
Characteristics

- Functionality is hidden behind an **interface**
- Offered Functionality is **publicly described**
- **Service** = public functionality that can be published, discovered, and executed across the network (client-server paradigm)

6



General Process: the „Service Triangle“



7



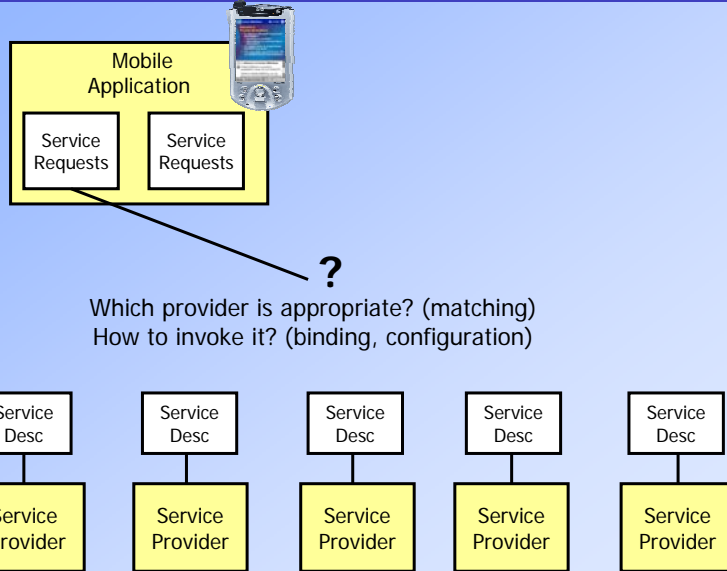
Main Advantages of SOC

- **Dynamic Service Binding**
- „Functionality binding on the fly“
 - robust
 - context-aware
 - network independent
- Most important task: **automation**
 - → Complete Process must be automatical
- Important:
 - **Service Description Language** (and its processing)
 - Determines degree of automatition

8



Service Description Language



9



Content of the Presentation

1. Service Oriented Computing
2. Syntactic Service Description Languages
 1. WSDL
3. Semantic Service Description Languages
 1. OWL-S
 2. WSMO
 3. DSD

10



WSDL

Web Service Description Language

11



What is WSDL?

- WSDL = Web Service Description Language
- Based on **XML** (syntax, primitive types from XSD)
- Is standardized by **W3C**, at the moment version 2.0

- Besides WSDL, other languages for service usage:
 - **UDDI** (registry)
 - **SOAP** (message exchange protocol)

12



WSDL: Basic concepts

In WSDL:

- **service** = collection of ports that offer operations
- **operation** = function that processes an input message and returns an output message
- **message** = has (more or less) complex fields

13



A Piece of Example WSDL Code

```
<types>
  <schema targetNamespace="http://example.com/stockquote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="TradePriceRequest">
      <complexType>
        <all>
          <element name="tickerSymbol" type="string"/>
        </all>
      </complexType>
    </element>
    <element name="TradePrice">
      <complexType>
        <all>
          <element name="price" type="float"/>
        </all>
      </complexType>
    </element>
  </schema>
</types>

<message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
</message>
<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>
```

14



Matching

- Often only: **manual binding** after **keyword-based search**
- Help through **simple matchers**:
 - Check whether input message of request „provides more“ information that input message of offer
 - Check whether output message of request „wants less“ information that output message of offer
- „wants less“
 - each type in the request has to appear in the offer
 - types of the offer have to be a subtype of the desired types in the request
- Many, many variations...
- Also: much work on composition

15



Advantages of WSDL

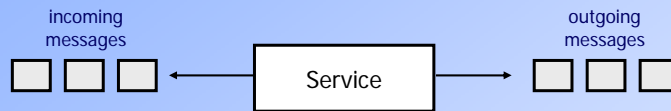
- Easy to understand
- Widely accepted
- Integrated in important programming languages
 - Microsoft .NET
 - Java: Java Web Service Development Pack
 - XML Processing, SOAP Binding, XML Registry...
- Many tools
 - Axis (open-source SOAP server)
 - IBM's WebSphere
 - ...

16



Problems with WSDL

Message oriented service description



- Also in: IDLs (CORBA, DCOM, EJB), ebXML, ...
- **Problems**
 - No advantage to pure interface description
 - **Semantics not clear**
Functionality has to be derived from flow of information
 - **Strict message compatibility**
If messages between provider and requestor do not match, the service cannot be used

17



Top 7 of Important Things about WSDL

- 1) Standardized by W3C
- 2) Based on XML
- 3) Used together with UDDI and SOAP
- 4) Describes Services by the types of their incoming and outgoing information ("message based")
- 5) Simple matching algorithms → manual discovery
- 6) Widely accepted, many tools
- 7) Many problems due to lack of semantics

18



Content of the Presentation

1. Service Oriented Computing
2. Syntactic Service Description Languages
 1. WSDL
- 3. Semantic Service Description Languages
 1. OWL-S
 2. WSMO
 3. DSD

19



OWL-S

Web Ontology Language - Services

20



Who is OWL-S?

- Maintained by a (rather closed) **coalition**
 - Burstein (BBN)
 - Sycara, Paolucci (CMU)
 - Hobbs (ISI)
 - Lassila (Nokia)
 - Martin (SRI)
 - McIlraith (Stanford)
 - Payne (U of Southampton)
 - Parsia (U of Maryland)
 - McDermott (Yale University)
- Formerly: **DAML-S**
- Founded by **DARPA** since August 2000

21



What is OWL-S?

- OWL-S: OWL-based web service ontology
- **Goal:** Develop tools and technology to enable automation of services on the semantic web.

Main improvements over WSDL

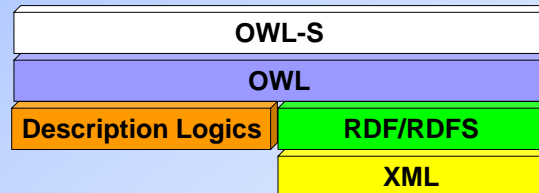
- Separates semantics from concrete description (approach from the **semantic web**)
- Separate concerns of the description (approach from **aspect orientation**)
- Does not only describe message flow (approach from **agent community**)

22



OWL-S and Ontologies

- OWL-S includes ideas from the **Semantic Web**
 - Explicit semantics
 - semantics not interwoven with the description
 - made explicit in additional **ontology**
 - Therefore: Usage of a **Ontology Description Language**
 - → constructs with a well-defined semantics
 - → reasoning over concepts is possible
 - → equality, subsumption ...
 - In OWL-S: OWL (Web Ontology Language)



23



Ontology in OWL

Parts:

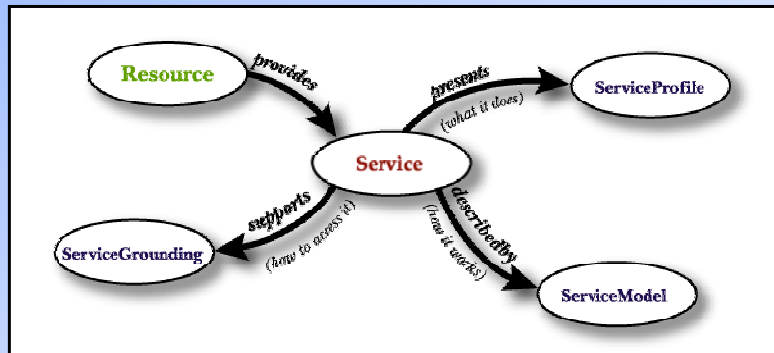
- Concepts, properties, individuals (from DL)
- Concrete domain and values (from XSD)
- XML-based syntax (among others)
- well defined, model-theoretic semantics
- Compatible with rules from Semantic Web Rule Language (SWRL)

24



OWL-S

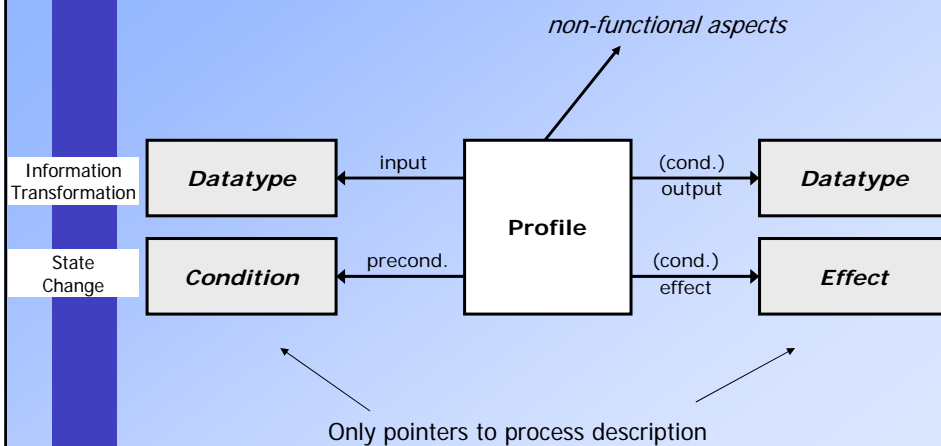
- Domain independent, **upper ontology** for services
- **Separation of concerns**
 - What does the service do?
 - How do I have to interact?
 - How do I have to invoke it technically?
- Do not mix what and how as in WSDL



25



Profile



26



Process

Goal

- Describes the **procedure** that is necessary to interact with the service (from client's point of view)
- Shows details and processing of IOPEs

Atomic Process

- Execute in a single step (from client's point of view)
- Can be invoked directly

Composite Process

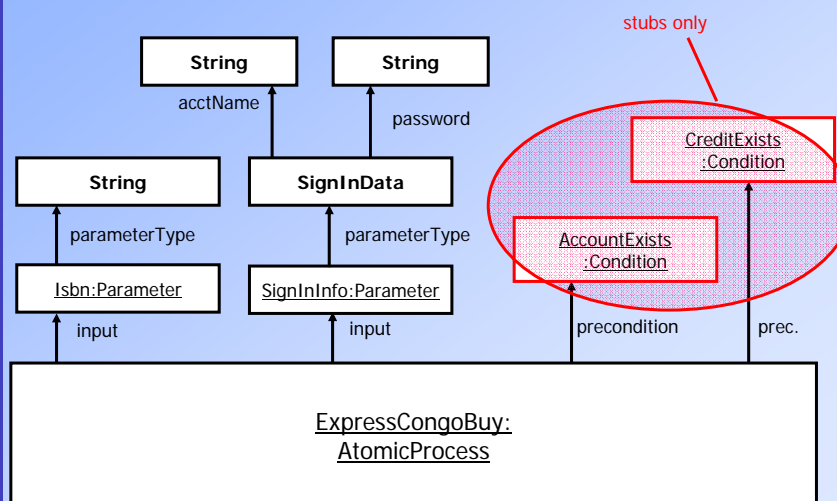
- Has a state
- Several messages have to be exchanged between client and server

27



OWL-S Example (1)

Extract from Congo Book Buy Example (OWL-S 1.0), own graphical notation

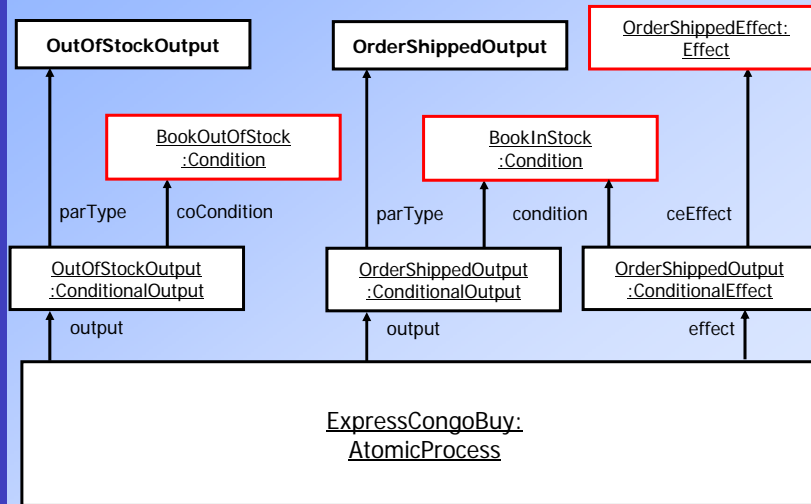


28



OWL-S Example (2)

Extract from Congo Book Buy Example (OWL-S 1.0), own graphical notation



29



Problems with OWL-S descriptions

- **No connection among IOPEs**
 - How are inputs transformed into outputs?
 - How do inputs influence effects?
- **Conditions and effects not describable**
 - Semantic Web Rule Language (SWRL) shall fill gap
 - but: problematic when referencing the real world
 - First ideas in OWL-S 1.1 not promising
- **Process centered**
 - main information in the dynamic process description
 - no matcher for this at the moment
 - current matcher only use infos from Profile and atomic processes

30



Further Problems: Requests

- No specialized **request description language**
- Requests = Descriptions of the perfect offer
- Matchmaker calculates similarity
- → **Preferences** of the requestor not expressible

31



Matchmaking

- Several similar approaches
- Ideas similar to WSDL:
 - Check whether the IOPEs of the offer „match“ the ones from the request
- „match“ differs in approaches
 - a) match = similar flat types
 - b) match = similar graphs
 - c) match based on reasoning services from DL (subsumes)
- All approaches suffer from the problems of OWL-S
- → **Rarely usable in real examples**

32



Top 5 of Important Things about OWL-S

- 1) Maintained by a coalition
- 2) Is ontology based (OWL)
- 3) Separates 3 aspects of description
- 4) Describes precondition and effect
- 5) Many problems with semantics
 - Unconnected IOPEs
 - conditions/effects not expressible
 - no specialized request language

33



WSMO

Web Service Modelling Ontology

Some of the WSMO slides are taken from
WSMO-Tutorial at ECOWS/NetObjectDays, Erfurt 2004
with kind permission of Christoph Bussler

34



Who is WSMO?

- **WSMO Working Group**
- **Chairs:**
 - Christoph Bussler (DERI)
 - Dieter Fensel (DERI)
- **It is open to:**
 - All members of SEKT, DIP, Knowledge Web, SWWS, and DERI
 - Experts in the field

35



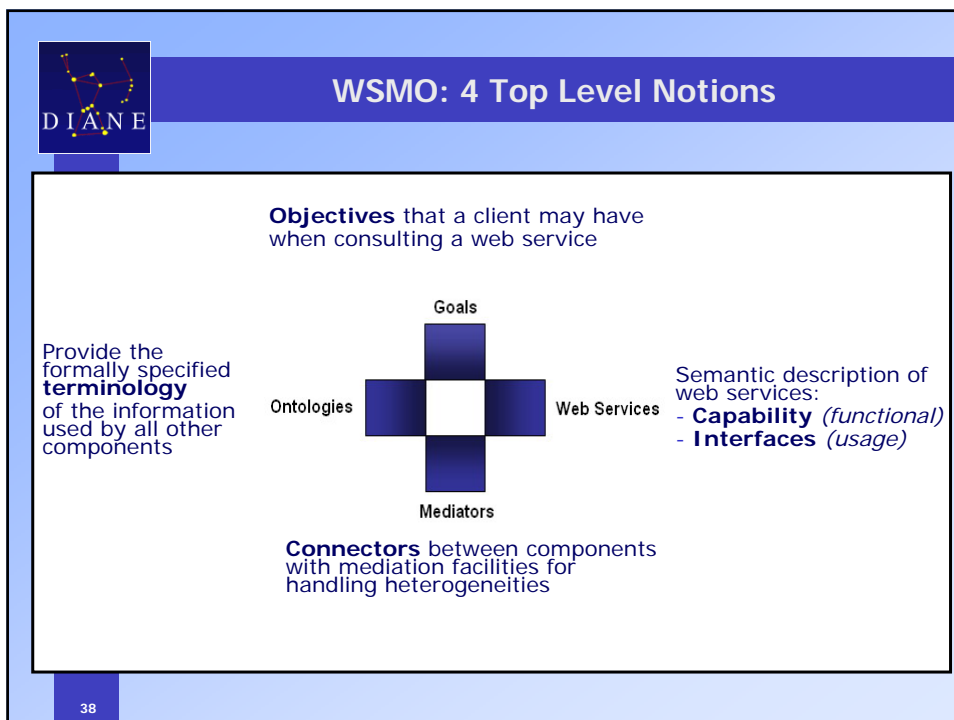
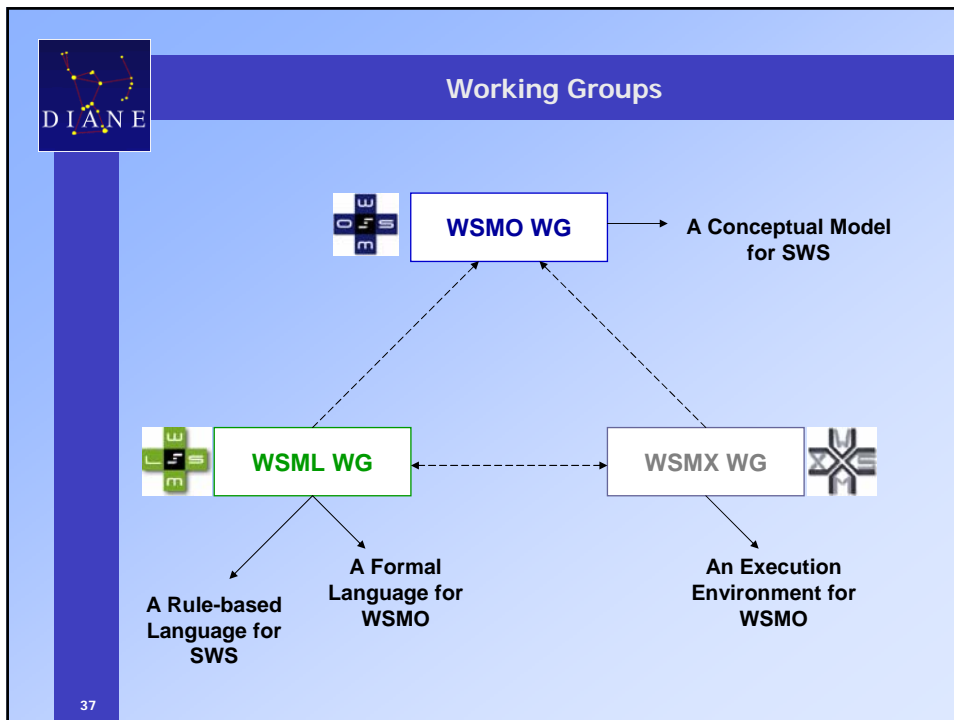
What is WSMO?

- WSMO = Web Service Modelling Ontology
- WSMO is an **ontology** and a **conceptual model** for the description of semantic web services

Mission:

- Strengthening **European research** and industry in Semantic Web Services
- Working towards **international standardization** together with US-based DAML program
- Strengthening world-wide research and standardization in **Semantic Web Services** field

36





Ontologies

- **Ontologies = “data model” of WSMO**
- **WSMO Ontology Language: WSML**
 - Compact syntax
 - Well-defined semantics
 - Based on f-logic (similar to object orientation)
- **WSMO Ontology Design**
 - Modularization: re-using ontologies
 - De-Coupling: heterogeneity handled by Mediators

39



Goals

De-coupling of request and service

- Requester formulates objective without regard to services for resolution
- ‘Intelligent’ mechanisms detect suitable services for solving the goal
- Allows re-use of services for different purposes

Goal specification through:

▪ **Post-Conditions**

Describe the state of the information space that is desired.

- The result expected from execution a web service
- Expressed as an axiom (based on ontology)

▪ **Effects**

Describe the state of the world that is desired.

- Expected changes in the world that shall hold after a service execution
- Expressed as an axiom (based on ontology)

40



Service Description

Parts:

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
- **Capability**
 - Pre-Conditions
 - Assumptions
 - Post-Conditions
 - Effects
- **Interfaces**
 - Choreography (between client and service)
 - Orchestration (between service and subservices)

41

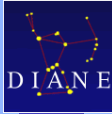


Example WSMO Service Description (1)

Precondition (expected input: buyer information and the desired trip with constraints)

```
precondition
definedBy
?Buyer memberOf po:buyer and
?Trip memberOf tc:trainTrip
[
  tc:start hasValue ?Start,
  tc:end hasValue ?End,
  tc:departure hasValue ?Departure
] and
(?Start.locatedIn = austria or ?Start.locatedIn = germany) and
(?End.locatedIn = austria or ?End.locatedIn = germany) and
?Departure > currentDate().
```

42



Example WSMO Service Description (2)

Assumption (a valid, i.e. not expired credit card)

```
assumption
definedBy
?CreditCard memberOf po:creditCard and
(currentDate.date.year < ?CreditCard.expirydate.expyear or
(currentDate.date.monthOfYear =<=?CreditCard.expirydate.expmonth
and currentDate.date.year = ?CreditCard.expirydate.expyear)).
```

43



Example WSMO Service Description (3)

Postcondition (returns a train trip with constraints)

```
postcondition
definedBy
?outputItinerary memberOf tc:itinerary
[
  ?trip memberOf tc:trainTrip
  [
    tc:start hasValue ?Start,
    tc:end hasValue ?End,
    tc:departure hasValue ?Departure
  ] and
  ?passenger hasValue _# memberOf loc:person and
  (?Start.locatedIn = austria or ?Start.locatedIn = germany) and
  (?End.locatedIn = austria or ?End.locatedIn = germany)
  and ?Departure > currentDate()
]
```

44



Example WSMO Service Description (4)

Effect (a trade (= contract of purchase) is created)

```
effect
  definedBy
    ?someTrade memberOf po:trade
    [
      ?po:items hasValues aTicket
      [
        itinerary hasValue outputitinerary
      ] and
      ?po:payment hasValue AcceptedPayment memberOf po:creditCard
    ].
```

45



Mediation

- **Heterogeneity...**
 - Mismatches on structural/semantic/conceptual level
 - Occur between different components that shall interoperate
 - Especially in distributed/open/mobile environments
- **Concept of Mediation:**
 - **Mediators** as components that resolve mismatches
 - Mediation cannot be fully automated (integration decision)
- **Types of WSMO Mediators:**
 - OO Mediators*: import ontologies & resolving heterogeneities
 - GG Mediators*: connect goals & resolve mismatches
 - WG Mediators*: link web service and goal & resolve mismatches
 - WW Mediators*: connect several web services for collaboration

46



Matchmaking

- Matchmaker by Michael Kifer
- Transforms offers / request description in **Prolog rules**
- Checks with reasoning support of Prolog
 - if an offer fulfills the desired postcondition/effect of the request
 - if an offer's precondition and assumption can be fulfilled by the requestor
- Not many details...
- Does not integrate mediators now

47



Characteristics of WSMO

WSMO captures semantics much better than OWL-S

But still drawbacks:

- Specifying **preferences** in request difficult
- No clean separation between class types (id-based vs. value-based)
- WSML incomplete
- Matching incomplete

48



Top 8 of Important Things about WSMO

- 1) WSMO: European counterpart to DAML-S
- 2) WSMO = abstract concepts,
WSML = concrete, f-logic based language
WSMX = environment for running WSMO services
- 3) Explicit notion of a Goal (i.e. distinction between service offer and request)
- 4) Distinction between change of information and change of world
- 5) Mediators for explicitly handling heterogeneity
- 6) Connection between APPEs through formulas
- 7) Captures semantics rather good, some problems with class types and user preferences
- 8) Matching based on Prolog, unfinished

49



DSD

DIANE Service Description

50



What is DSD?

- DSD = DIANE Service Description
- Developed within **DIANE project**, member of **SPP 1140** on basis software for self organizing networks
- Main goal: **automatic service usage** (especially in mobile environments)
- DSD consists of
 - Specialized **ontology definition language**: DIANE Elements
 - **Generic ontologies** for service descriptions
 - **Guidelines** for correct usage („DSD Cookbook“)
- Additionally:
 - Complete **middleware** around DSD consisting of several modules:
 - Binder, Matcher, Caller on client side
 - Container Manager on server side

51



Main Concepts of DSD

- Pure **state** orientation
- **Layering** of ontologies
- Descriptions as configurable **instances**
- Full integration of **preferences** into requests
- Own **ontology definition language** with elements especially for service descriptions
- Full decoupling of interfaces through **request grounding**
- Different **representation styles**

52



Concept 1: Pure State Orientation

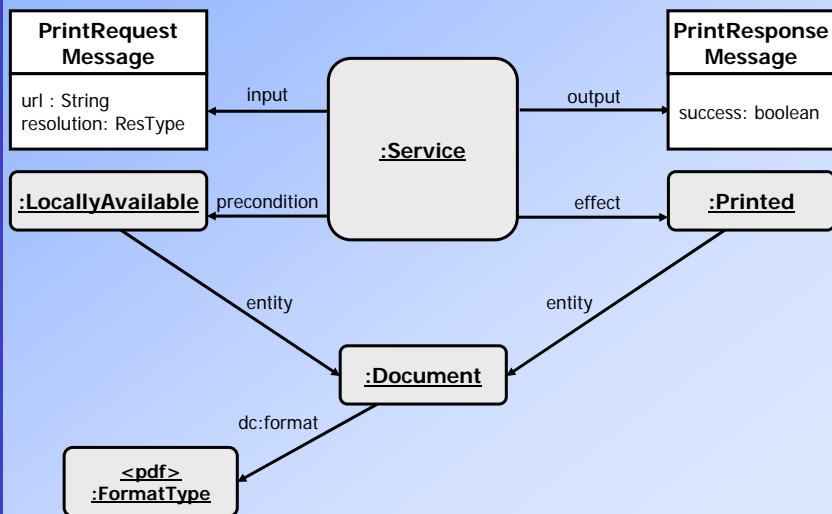
Idea: Purely state oriented service descriptions

- Completely omit explicit description of information flow
- Only use precondition and effect
- Describe services as instances and include messages' configurations possibilities as variables

53



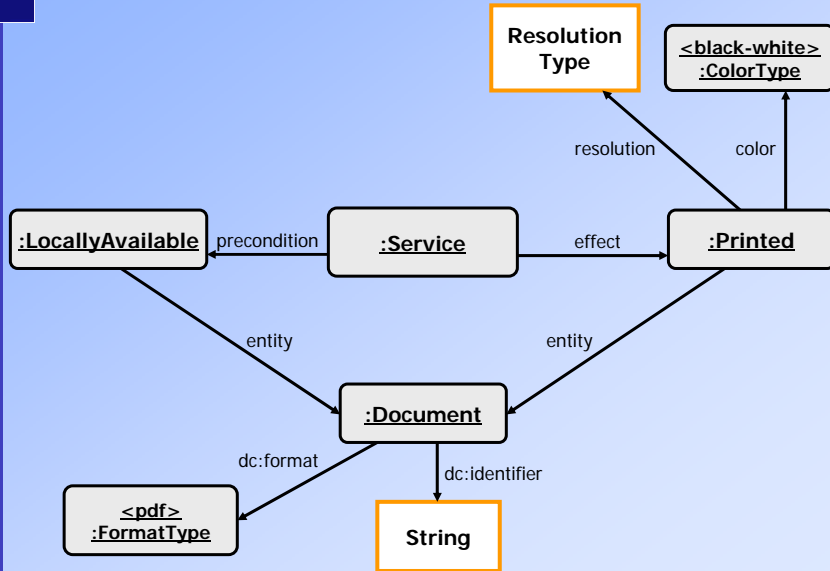
Example with Approach 1



54



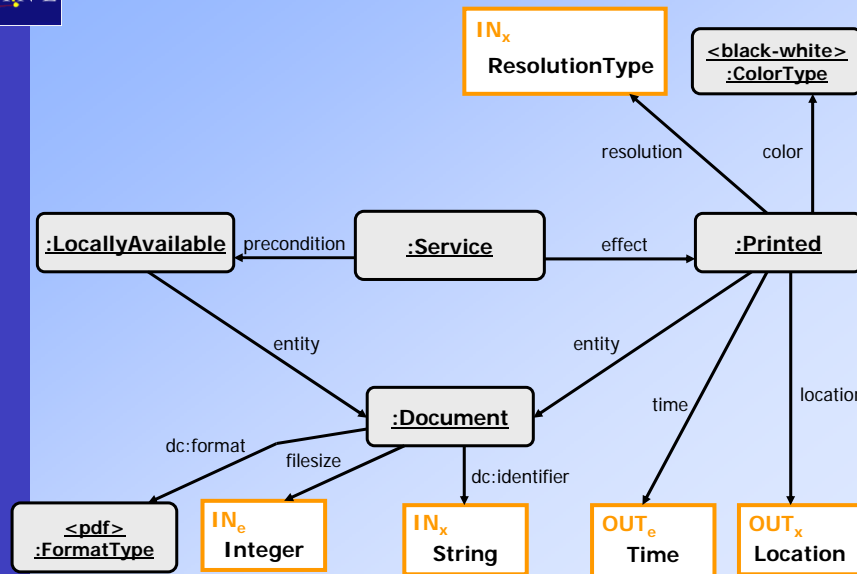
Example with Approach 2



55



Example with Approach 3



56



Advantages of Pure State Orientation in DSD

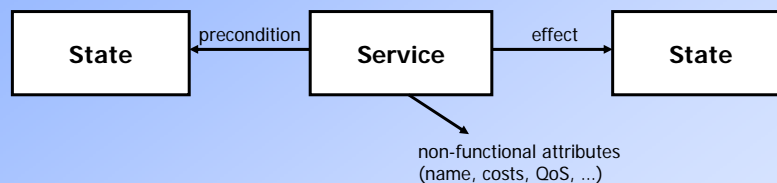
- **Interrelation** between precondition and effect is clearly visible
→ as they both point to a shared object
 - **Influence of parameters** on result is clearly visible
→ because of direct integration of variables into states
→ no separation between message and state
 - **Configuration process** is extended and clearly defined
→ as variables have categories (**IN_e**, **OUT_e**, ...)
→ allows pre-execution configuration
- **Semantics** of the service **need not be guessed**, but is completely included in the description

57



Basic Schema for Service Descriptions in DSD

Schema for state oriented service descriptions



Problem:

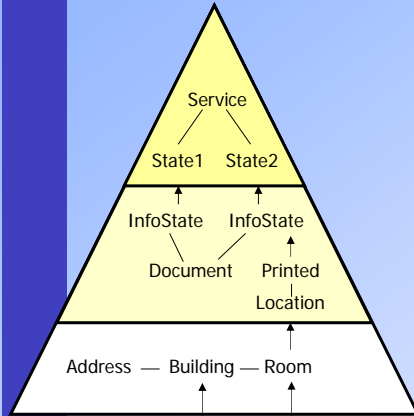
- Too generic!
- How to obtain structured service descriptions

58



Concept 2: Layering of Ontologies

S → class X, instance y, ...



Ontology Definition Language (=Metaschema)

- „DIANE Elements“

I. Upper Service Ontology (=Schema)

- Task: general structure of a service description
- common, small

II. Category Ontologies (=Schemata)

- few
- Limitation of states in precondition and effect
- For example: DocumentService

III. Domain Ontologies (= Schemata)

- describe single domains
- many, distributed
- Example: Location, Printing, Shoes

Instance Pool (= Instances)

- represent individuals of the real world
- many, distributed



Concept 3: Full Preference Integration

Basic matching algorithm: graph matching

Problem:

Matcher does not know the preference/tolerance of the requestor.

The matcher

- has to use general deviation **heuristics**
- or is very **conservative** and only allows exact matches
- **biased** matching

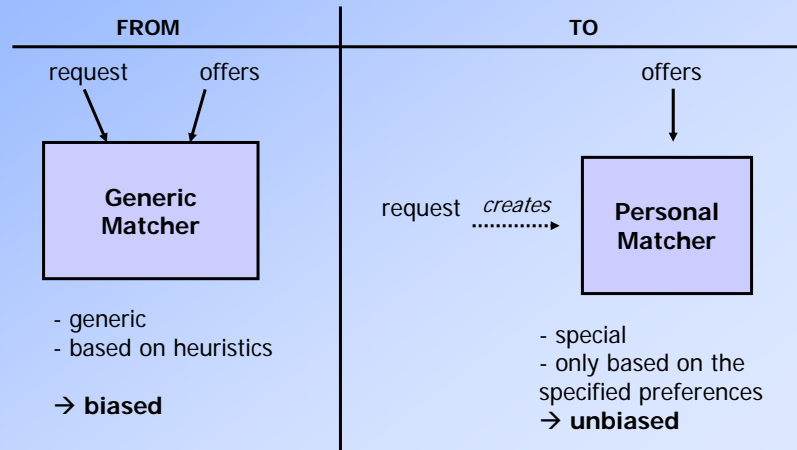
- often provides **many or no** matching offers
- Requestor has to **choose manually** or **blindly rely on it.**

→ **Cannot or is not used for automatic service binding.**



Approach in DSD

- Tell the matcher **exactly** what the preferences are



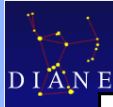
61



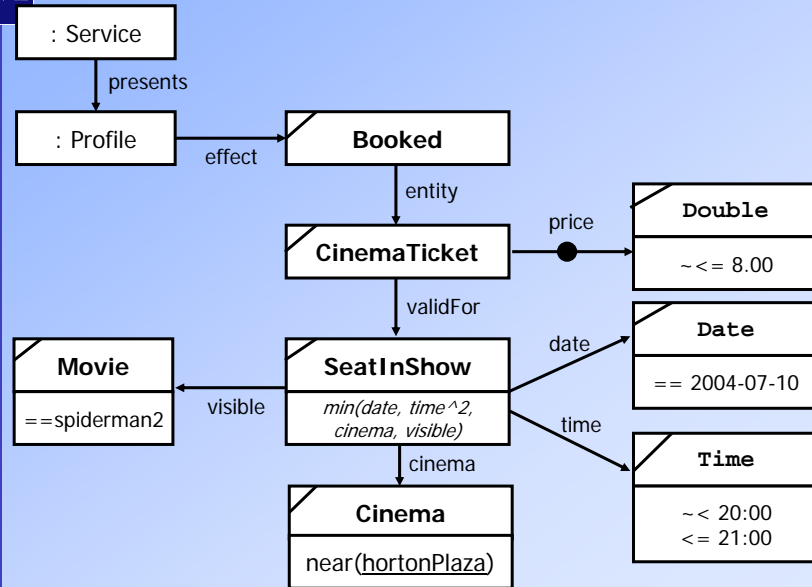
Preference-Containing Requests

- Request != single instance describing the **perfect** service
- But: **Set of suitable services**
- **degrees of membership** from (0,1] determines preference for this service
- → Leads to special request description elements (like fuzzy sets)
- → Matching is **test on set membership** → unbiased
- → But: Offer descriptions can contain **variables**. Matcher has to derive the assignment from the requestor's preferences

62



Example Request in DSD



63



Top 8 of Important Things about DSD

- 1) Developed within DIANE project in SPP 1140
- 2) Goal: Full automation of service usage (in mobile environments)
- 3) Purely state oriented
- 4) Descriptions as configurable instances
- 5) Preference-containing requests
- 6) Own ontology definition language with special elements
- 7) Layering of ontologies
- 8) Unbiased matching, also binds variables

64



SUMMARY

- The **service description language** is essential part of service oriented computing
- Especially in mobile environments: **automation**
- Presented: WSDL, OWL-S, WSMO, and DSD
- **Challenges:**
 - Automatic binding of stateful services (→ "holy grail")
 - Automatic service composition
 - Automatic creation of offer/request descriptions
 - Mediation of ontologies and processes
 - Distribution of ontologies in the network
 - Building powerful middlewares for mobile devices
 - ...

65



**Thank you
for your attention!**



DIANE Project
<http://www.ipd.uni-karlsruhe.de/DIANE/en>

66



DIANE

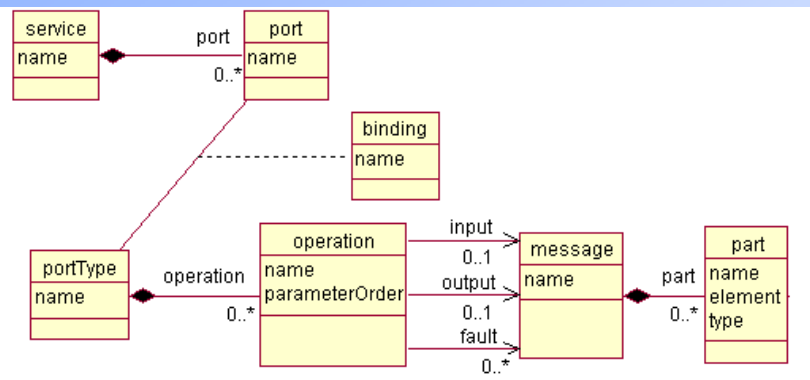
APPENDIX

67



DIANE

Basic Concepts of WSDL as UML



68



The Semantic Web

*“The semantic web is an **extension** of the current web in which information is given **well-defined** meaning, better enabling computers and people to work in cooperation”*

[Berners-Lee et al. 2001]

69



Further Problems of OWL-S inherited from OWL

Problems with A-Box reasoning

- Requests have to be schema-based
- Generic multiple-purpose requests instead of specialized single-purpose requests
- „I want to buy the book ‚Harry Potter‘“
→ „I want to have a service, where I can enter any book title and this book is sold to me“
- → Not natural way of expressing a request

OWL offers the „wrong“ modelling elements

- People think in object-oriented schemata and instances
- For Requests, conditions and rating possibilities would be necessary
- DL's quantifiers only needed in artificial examples

70



Representation Forms of DSD

