Semantic Web Technologies II
SS 2009

08.07.2009

KASWS – Karlsruhe Semantic Web Services

Dr. Sudhir Agarwal
Dr. Stephan Grimm
Dr. Peter Haase
PD Dr. Pascal Hitzler
Denny Vrandecic
Martin Junghans

Agenda

- Repetition
- Shortcomings of existing approaches
- KASWS – Karlsruhe Semantic Web Services

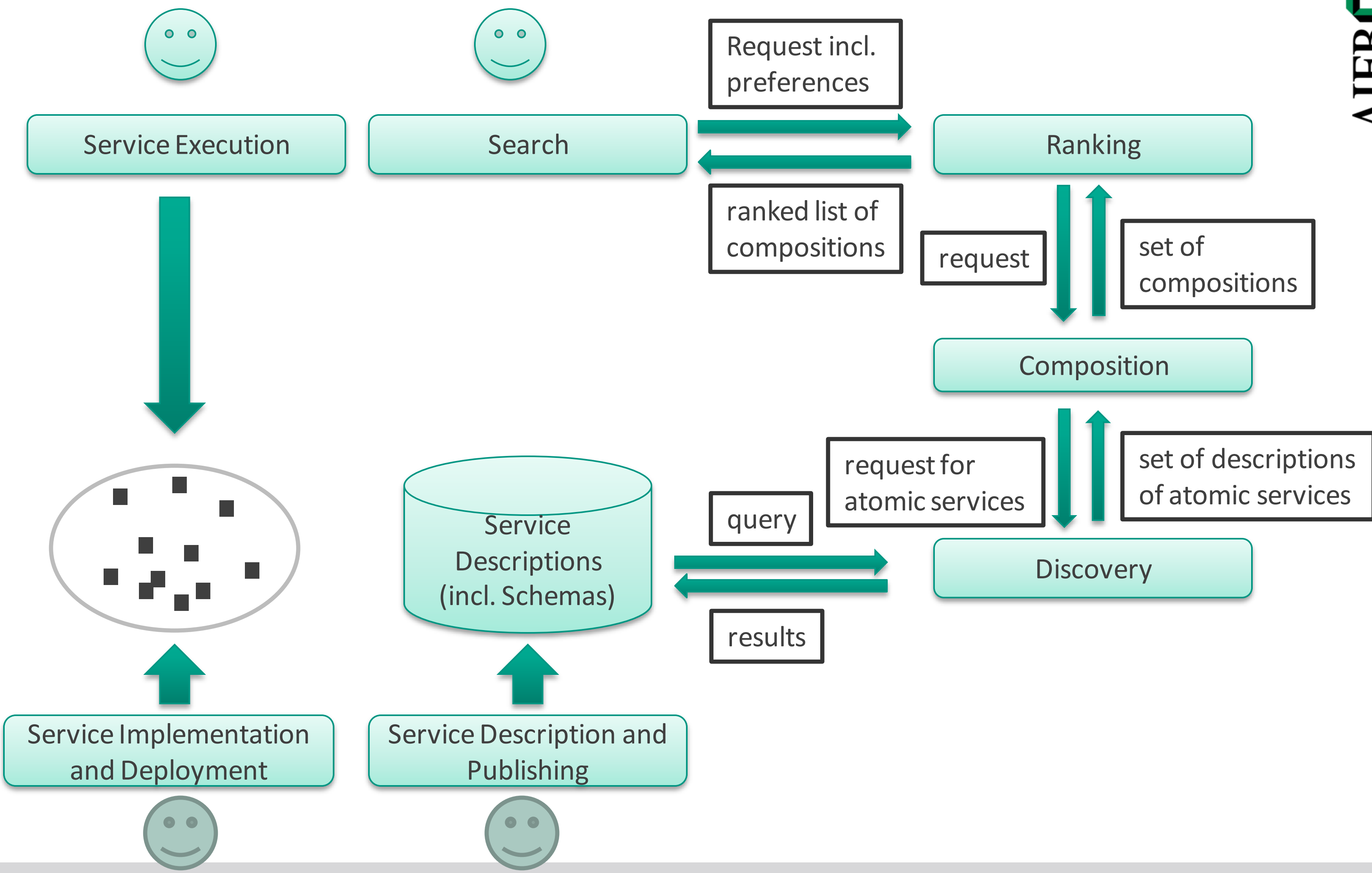
Agenda

- Repetition
- Shortcomings of existing approaches
- KASWS – Karlsruhe Semantic Web Services

Repetition

- Web services
  - Principle
  - Standards
- Semantic Web services
  - Motivation
  - Need for a formalism to describe Web services
  - Discovery, ranking, composition

A Simple Scenario



Agenda

- Repetition
- Shortcomings of existing approaches
- KASWS – Karlsruhe Semantic Web Services

Shortcomings of Existing Approaches

- Existing approaches
  - OWL-S (Semantic Markup for Web Services)
  - WSMO
  - WSMO-Lite
  - Only briefly discussed in this lecture

OWL-S

- Description of semantic Web services
- Ontology on top of Web Ontology Language (OWL)
- Types of knowledge about services
  - Service profile
    – Service functionality
  - Process model
    – Interaction with services
    – Input, output, pre-conditions, results
  - Service grounding
    – Connection to WSDL service description

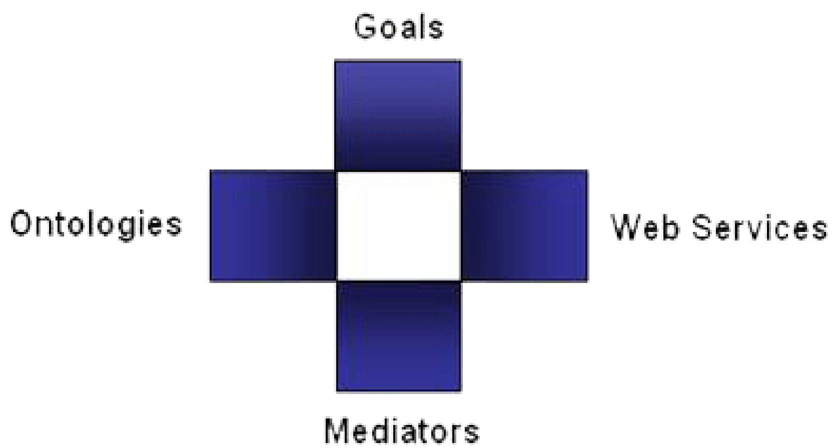
Shortcomings of OWL-S

- No concrete formalism for pre-conditions, results
- OWL, OWL-S have description logics (DL) semantics
  - DL reasoners are not capable to capture dynamics
  - DL cannot handle changing knowledge bases contemporarily
- Web services are dynamic in nature (processes)
  - Variables are essential for process modeling
  - E.g., a variable (value can be changed) cannot be modeled by DLs

Top-level Elements Defined by WSMO

Objectives that a client may have when consulting a Web Service

Provide the formally specified terminology of the information used by all other components



Semantic description of Web Services:

- **Capability** (*functional*)
- **Non-functional properties**
- **Interfaces** (*usage*)

Connectors between components with mediation facilities for handling heterogeneities

(http://www.wsmo.org)

Drawbacks of WSMO

- As OWL-S, WSMO uses ontologies to model processes
  - DL semantics
  - Can‘t reason about variables and state changes
- No practical support for non-functional properties
- Limited to classical Web services only
  - No support for Web sites
- Multiple environments not supported
  - A process triggered by invoking a Web service may run partially outside the Web

WSMO-Lite

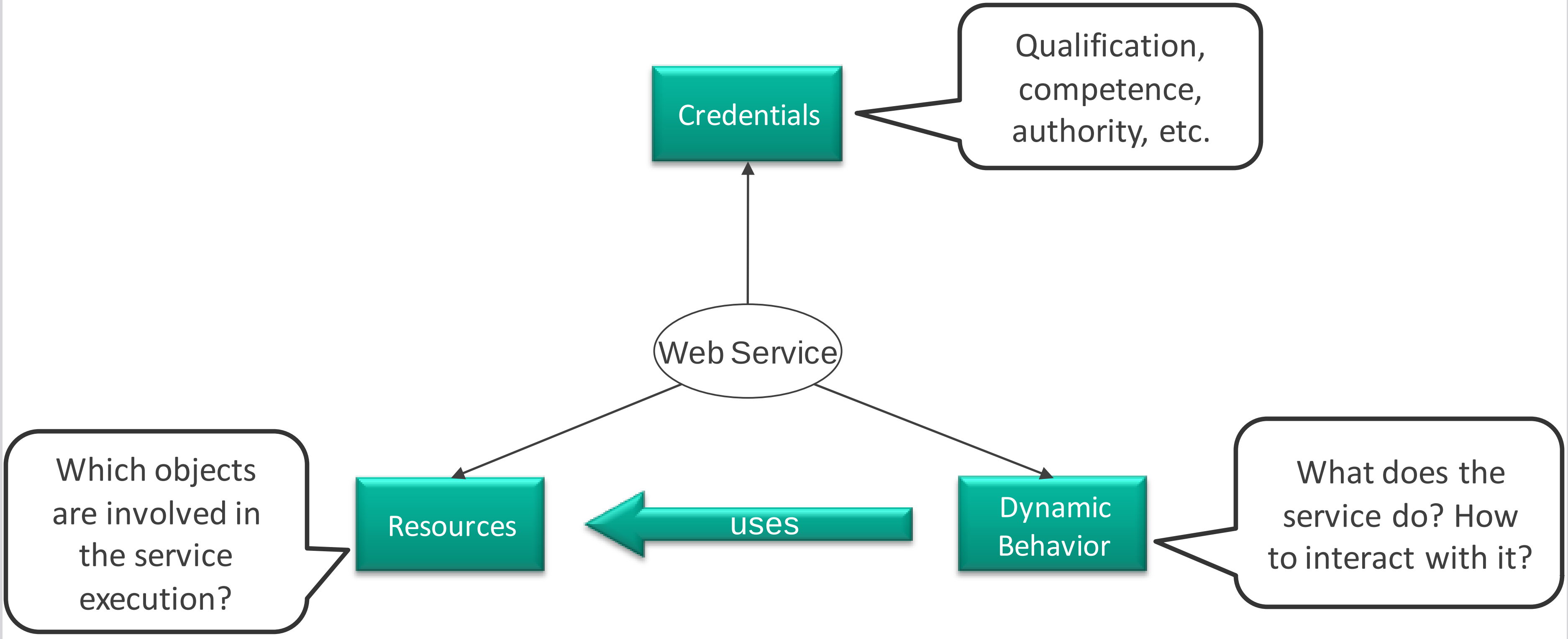
- WSMO-Lite
  - Service ontology
  - Defines service model that can be used with SAWSDL
    - restricts values of the `<modelReference>` attribute for various SAWSDL elements
    - Also applicable to hRESTS + MicroWSMO
- Shortcomings
  - Similar to previously mentioned ontology-based approaches
  - No reasoning support for dynamics

Agenda

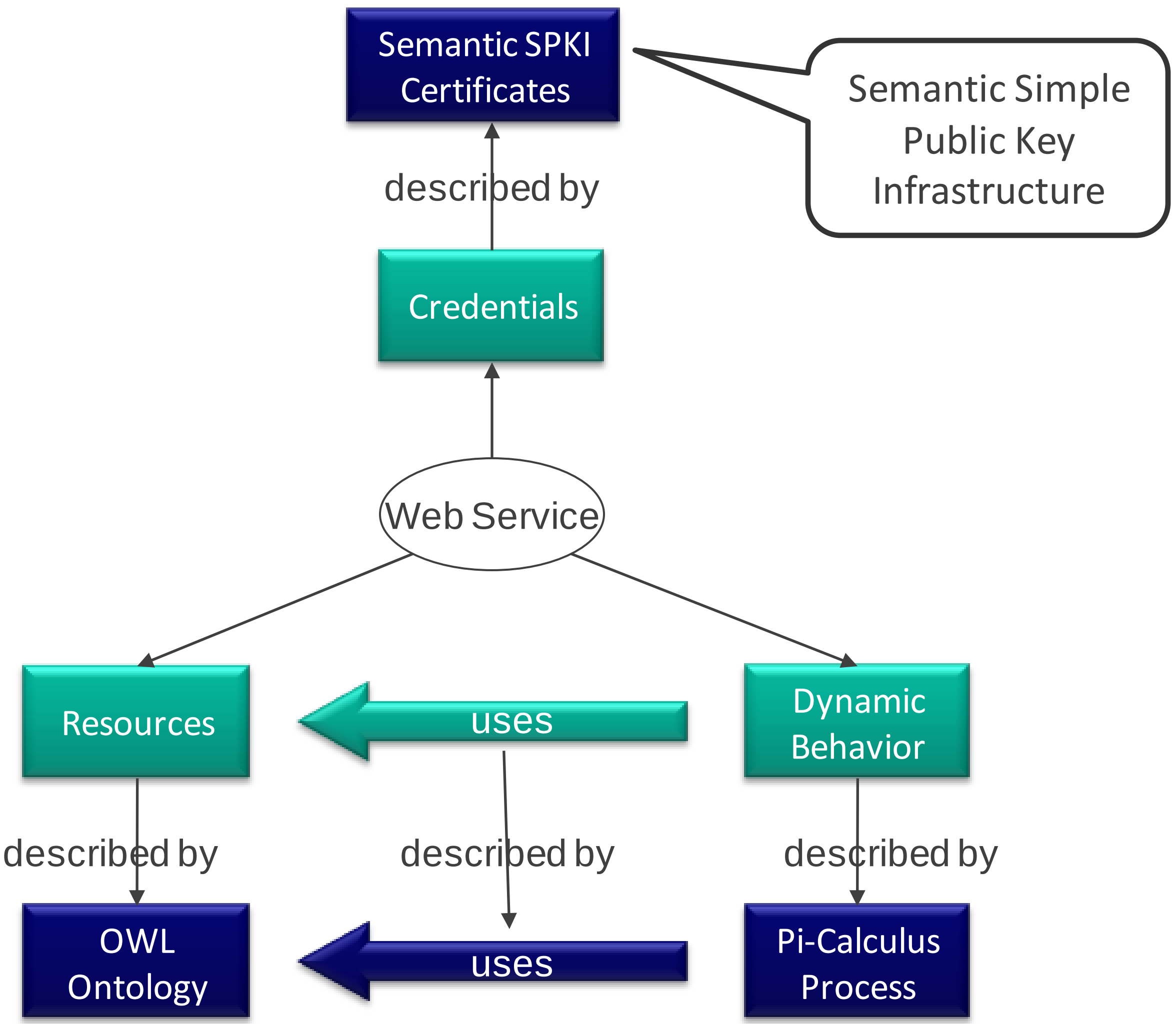
- Repetition
- Shortcomings of existing approaches
- KASWS – Karlsruhe Semantic Web Services

KASWS Conceptual Model

- Extended notion of Web service
  - WSDL and RESTful Web services, Web sites, Web pages



Formalisms


Semantic SPKI Certificates
Semantic Simple Public Key Infrastructure
described by
Credentials
Web Service
Resources
uses
Dynamic Behavior
described by
described by
described by
OWL Ontology
uses
Pi-Calculus Process

OWL Recap

- Domains modeled in terms of concepts and relations
- Concepts can form sub-class hierarchies
- Attributes of concepts link them with literal values
- Relations define connections between concepts
- Individuals instantiate the concepts

- Briefly: OWL allows modeling of objects of a domain
- OWL not suitable for modeling processes
  - Can not capture the semantics of dynamics
  - E.g., changing states and order of activities

Introduction to Pi-Calculus

- One of the most widely known process algebras
  - Mainly developed by Robin Milner (Turing Award 1991)
  - Extension of Calculus of Communicating Systems (CCS)

- Different variants of Pi-Calculus for different purposes
  → We will use our own variant (KASWS)

Introduction to Pi-Calculus

- Pi-Calculus roughly offers
  - Communication activities
    - – Communication takes place by exchanging messages over some communication channel
    - – Communication channels can be treated as data → enables modeling of flexible processes whose configurations can change at run time
  - Control constructs
    - – Activities can be performed
      - – sequentially,
      - – in parallel, or
      - – by choice (deterministic or non-deterministic)

KASWS - Basic Activities

- Input
  - Communication activity
  - Receives some values from the user
  - $c[v_1,\ldots,v_n]$ receives n values along channel c and binds them to variables $v_1,\ldots,v_n$
- Local operation
  - Deterministic method
  - E.g., a database query executed or update performed by the Web server
  - $y_1, \ldots, y_m = l\,(x_1, \ldots, x_n)$ executes method l with arguments $x_1, \ldots, x_n$ locally and returns $y_1, \ldots, y_m$

KASWS - Basic Activities

- Output
  - Communication activity
  - Outputs values to the user
  - c<x_1, …, x_n> emits values x_1, …, x_n along channel c

- Activities can be connected to build processes
- c, l, v, x, y all are instances of the domain ontology

KASWS - Sequence

- Activities can be connected in sequence with a.P
  - with a an activity and P a process expression
  - E.g. the process $c[v_1,\ldots,v_4].\ x_1=l_1(v_1, v_2).\ x_2=l_2(v_3,v_4).c\langle x_1, x_2\rangle.P$
    - receives four values along channel c and binds them to $v_1,\ldots,v_4$ then
    - uses v_1 and v_2 to calculate x_1 with local operation l_1 then
    - uses v_3 and v_4 to calculate x_2 with local operation l_2 then
    - sends x_1 and x_2 along channel c then
    - behaves like process P

Conditional Branching

- A conditional process $\omega?P_1:P_2$ is a process that behaves like P_1 if ω is true, otherwise like P_2

- Example process $y=l(x_1, x_2).y?P_1:P_2$
  - Suppose l returns the truth value of "LessThan(x_1,100) and OldMan(x_2)"
  - "LessThan" and "OldMan" are predicates and concepts from the domain ontology
  - y, x_1 and x_2 are instances of the domain ontology
  - Then the process behaves like P_1 if x_1 is at most 100 and x_2 is an old man, otherwise like P_2

Modeling traditional Web Services

- A traditional Web service can be seen as a process
  input(argVal).c=local(argVal).c?
  result=computeResult.output[result]:output[error]
- Meaning:
  - Web service performs an input activity, thereby binds the arguments values provided by the client to variables,
  - then it may check whether the input values fulfill some constraints
  - if the conditions are fulfilled,
    then it computes the result and outputs the result
    else it outputs an error message

Modeling of Web Pages

- Click on a link is equivalent to invoking a Web page
  - A URL may have arguments
- Form submission method is either GET or POST
  - GET is equivalent to click on a link (directly invocable) with input values encoded in the URL such as http://www.google.de/search?hl=de&q=KASWS&btnG=Suche
  - POST does not encode parameters in URL (not directly invocable) but sends them in body of HTTP request
- How to describe a URL with arguments?
- How to describe invocation of a URL?

Service Identifier and Service Invocation

- Service identifier is a named process
  - $A(x_1,\ldots,x_n) =_{def} P$
    with P a process expression and $x_1,\ldots,x_n$ the arguments
  - URL with arguments is described as a service identifier $A(x_1,\ldots,x_n)$ with
    - A equals to base URL
      - E.g., “http://www.google.de/search”
    - $x_1,\ldots,x_n$ the arguments after “?” in the URL.
      - E.g., “hl”, “q” and “btnG”

Service Identifier and Service Invocation

- Service invocation $@A\{v_1,\ldots, v_n\}$ denotes invocation of the service identifier A with values $v_1,\ldots, v_n$ for its arguments $x_1,\ldots, x_n$
  - Click on a link is described as invoking a service identifier with values $v_1, \ldots, v_n$
    - E.g., “de”, “KASWS” and “suche”

Data Flow among Services

- In Web, different services run independently and concurrently to each other
- In general, there is no central control
- Data is transferred by exchanging messages

- How to describe data flow among Web services?

Composition Operator & Data Flow

- A composition process $P_1 || P_2$ is a process that behaves like P_1 and P_2 running concurrently
- A message can be exchanged between two concurrently running processes when one process performs an output activity and the other an input activity at the same channel

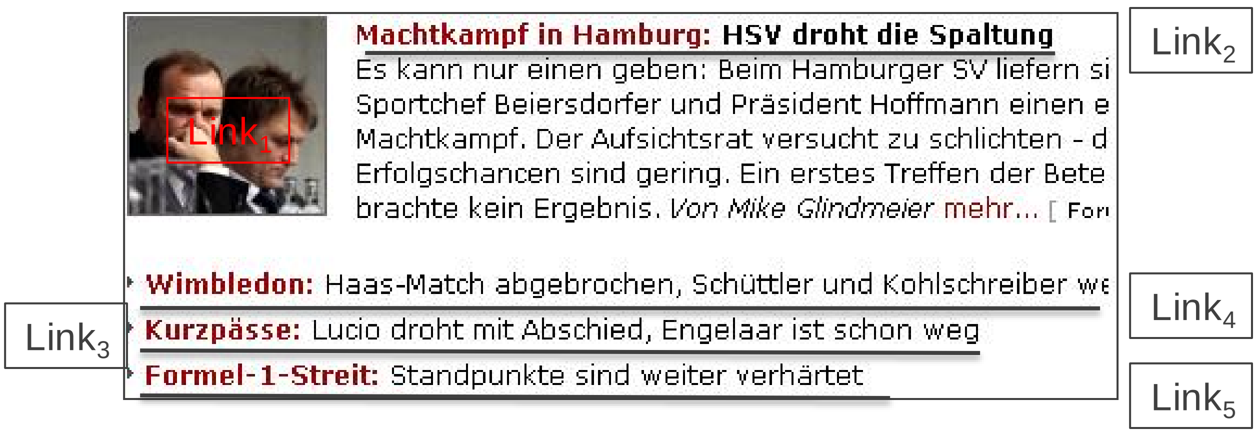
Composition Operator & Data Flow

- Suppose P_1 = c[BookOrder].Q_1
  and P_2 = c<NewOrder>.Q_2

- Consider the process P_1 || P_2
  - Process P_1 sends a message with `BookOrder` as content to the channel c. It then behaves like Q_1
  - Process P_2 receives a message at channel c, and binds the received value to `NewOrder`. It then behaves like Q_2

User Selection

- In addition to information, a Web page presents possibilities for further navigation



  - Links and „action“ of forms are the only choices a user has for moving to a next page

- How to model the alternatives a user may chose from?

Non-deterministic Choice

- $P_1+P_2+\ldots+P_n$ denotes non-deterministic choice among processes $P_1,P_2,\ldots,P_n$
- E.g., the process buy+modify+cancel models following options a user may chose from
  - Buy those books in the shopping cart
  - Modify the order
  - Cancel the order
- The process $Link_1+Link_2+Link_3+Link_4+Link_5$ models the five links a user may chose from the example Web page

Summary of KASWS Syntax and Informal Semantics

- **Input**: $x<v_1, \ldots, v_n>.P$
  - Receive n objects along the channel x and bind them to variables $v_1, \ldots, v_n$
  - Then behave like P
- **Output**: $x[y_1, \ldots, y_n].P$
  - Send n objects $y_1, \ldots, y_n$ along channel x
  - Then behave like P
- **Local**: $y_1,\ldots,y_m = l(x_1,\ldots,x_n).P$
  - Perform local operation l with arguments $x_1,\ldots,x_n$ and outputs $y_1,\ldots,y_m$
  - Then behave like P
- **If-Then-Else**: x?P:Q
  - Check whether x is true
  - If yes, then behave like P, else like Q
- **Parallel**: P||Q
  - Perform P and Q in parallel
- **Non-Deterministic Choice**: P+Q
  - Perform either P or Q as selected by the user
- **Agent invocation**: $A(x_1,\ldots,x_n)$ with A an agent identifier (named process expression)
  - Invoke an agent identifier with parameters $x_1,\ldots,x_n$
  - Enables recursion, since the defining process expression of A can contain A
- **Null**: **0**
  - This process does nothing
  - used as termination symbol

Access Control

- Access control is important in almost every business process to ensure
  - Availability, e.g., a student must show his library card to be able to use the online library system; uncontrolled access may lead to Denial of Service attacks
  - Confidentiality, e.g., a student has access to information relevant to his library account only
  - Integrity, e.g., a student may not change or cause a change in the account of some other student
  - Legal Correctness, e.g., an online video store must check age before lending certain videos
  - ...

Access Control

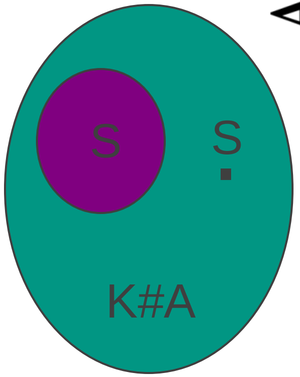
- Access control should not be identity based but credential based
  - Credentials are properties that can be proven, e.g., being a student, being an adult, etc.
  - In open systems, like the Web, identities of the actors are not known or are not important
    - Students of University of Berlin should be able to borrow books from the library of University of Karlsruhe, even if the latter does not have a database of all the students of the former
  - Need for modeling credentials and credential based access control policies (ACP)

- We use a semantically enhanced variant of Simple Public Key Infrastructure (SPKI) for modeling credentials and OWL for modeling ACPs

Modeling Credentials with Semantic SPKI

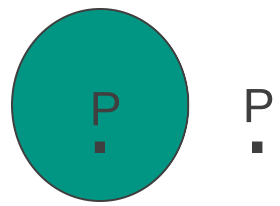
- Each actor (service) is identified by a public key

- An actor defines properties that it certifies
  - E.g., fast response time, high availability, etc.



- An actor issues name certificates (K, A, S)
  - K is the key of the issuing actor; A is the property name
  - K and A together (K#A) build a unique property name
  - S is the subject (name or key)
    - If S is a name: S is a subset of K#A
    - Else (S is a key): S in K#A
  - Certificate (K,A,S) has to be signed with the key K

Modeling Credential based Access Control Policies



- A trust policy P is either a key or a name
  - If P is a key, only actor with key K=P satisfies P
  - If P is a name, every actor with key K in P satisfies P

Integrating Access Control in Behaviour

- Access control checks can be integrated as a condition in a process expression
  - A special predicate CCD(P, C) checks whether the set of certificates C fulfills the policy P
  - CCD is used as condition in IfThenElse process expressions to integrate access control in the behaviour

Summary

- Limitations of existing SWS approaches
- KASWS as formalism for describing Web services, Web sites and Web pages
  - based on process algebra
  - supports credential based access control

- Further reading material
  - Book: Communicating and Mobile Systems: The Pi Calculus; Robin Milner
  - PhD Thesis: Formal Description of Web Services for Expressive Matchmaking; Sudhir Agarwal