On Specifying and Providing Data Concerns for Adaptive Service Composition and Execution

Hong-Linh Truong

Distributed Systems Group, Vienna University of Technology

truong@infosys.tuwien.ac.at
http://www.infosys.tuwien.ac.at/Staff/truong

Joint work with: Schahram Dustdar, Marco Comerio, Atif Manzoor, Andrea Maurino, Michael Mrissa, Flavio De Paoli
Overview

- 9:30-11:30
  - Motivation
  - Data Concerns
  - Data Concerns Evaluation and Publishing

- 13:00-14:30
  - Service Contract Compatibility Evaluation
  - Service Information Overloading
  - Conclusion and Future Work
Motivation
Service Composition and Execution

- The ultimate goal: provide *relevant* results in an *acceptable* quality

- Solutions
  - Adaptive/context-aware service composition and execution

- But several challenging issues
Examples of problems

- Irrelevant service information in service composition
  - Too much service information returned to the developer or (semi-)automatic algorithms
- Irrelevant information between service composition and execution
  - Temporal distance between the composition time and execution time
    - The expected quality of service is invalid, triggering quality-aware service adaptation
- Irrelevant information in service usage
  - Results are returned without a clear usage and ownership causing data compliance problems
Example 1: Too much service information
Example 2: Mashup

- Composition of Yahoo! Boss News Search, Google News Search, and Flickr
- recent news and high-qualified images, but free-of-charge, related to "Haiti earthquake"
If the composer is aware of context and quality parameters

- Possible mappings of context and quality requirements

but it is a tedious task and hard to be automated and we are not sure we have a correct mapping.
Example 3: access to data-as-a-service (DaaS) in the cloud

- Retrieve big datasets from RESTful services for further extraction, transform or data composition activities

http://www.undata-api.org/
**Example 3: quality problem**

- **Example:** study the population growth and literacy rate from 1990-2009 for all countries in the world
- **Without QoD:** get datasets and perform mashup
- **With QoD support:**
  - Population annual growth rate (percent):
    - dataelementcompleteness= 0.8654708520179372, datasetcompleteness=0.7356502242152466;
  - Adult literacy rate (percent):
    - dataelementcompleteness=0.5874439461883408, datasetcompleteness=0.04349775784753363

→ **Should we retrieve the data and perform data composition?**
Example 4: Sensor-as-a-Service in Smart Environments

- Smart environments with several low level sensors:
  - Recognize human activities: idle, relaxing, cleaning up,
  - Provide context information for adaptive service discovery and execution
  - E.g., FP7 SM4All, FP7 EU OPPORTUNITY
- Virtual Sensor-as-a-Service provides human activities
Context, quality, and relevance dependencies in data service composition

- Only supporting **QoS and context awareness at the service level as a whole** is not enough
Our big picture of problems

- Context and quality information models
  - Unstructured description of context, QoS and quality of data (QoD)
  - Different specifications and terminologies
  - Mismatching semantics of information about services and data

- Context and quality information access APIs
  - No/Limited description of data and service usage
  - No API for retrieving quality and context information
  - No quality and context information associated with requested data

- Context and quality evaluation techniques
  - Missing evaluation of compatibility of context and quality of multiple services
  - Large/irrelevant data quantity

Require a „holistic integration“ of information models, APIs and evaluation techniques to support adaptive composition and execution
Current research focuses

- Context and quality information models
  - Often used only a fraction of QoS or context, several specifications that cannot be easily linked

- Access APIs
  - Mainly static publishing, few metrics at runtime but typically at the service as a whole level

- Adaptive and context-aware algorithms
  - Mainly for adapting individual services in a composition
  - Either consumer-service flow or composite service-service flow

The role of data concerns? Context and quality associated with data resources? Dependency chain: consumer-service-service-resource?
Our suggested roadmap

- Developing a meta-model and domain-dependent semantic representations for quality and context information specifications
  - Reconciliation of context/quality terms
  - Linked data
- Developing context and quality information that can be accessed via open APIs
  - On-the-fly content/quality access
- Developing techniques for context and quality evaluation
  - context and quality compatibility evaluation and composition
Data Concern Specification
Background on DaaS

- Web services technologies, the SaaS model and the cloud computing model foster the concept of data/information as a service (DaaS)
- No precise definition but DaaSs
  - Provide data capabilities rather than provide computation on data or data based on computation
- Providing DaaS is an increasing trend
  - In both business and e-science environments
    - Bio data, weather data, company balance sheets, etc., via Web services
  - Academic research and industrial relevant research topics
Background - our view on DaaS

- Read-only DaaS versus CRUD DaaS
  - Read-only DaaS:
    - Service consumer can only read the data
  - CRUD DaaS
    - Service consumer can read/write, using their own data format

- Service APIs versus Data
  - Service APIs are used to CRUD data
  - They are not the same wrt concerns
  - The data provider is not the same as the DaaS service provider
### Products

Your business processes are easily enhanced with accurate and cost-effective Web services from Strikelron. These services are easy to integrate into your business processes, applications, websites, and more, providing seamless functionality that improves productivity, reduces maintenance and management, and enables efficiency. Discover innovative Web services from Strikelron and access powerful, live, accurate, and actionable data—when and where you need it.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email Verification</td>
<td>Know before you send. Just provide any email address and you'll receive an indicator response regarding its validity. Which means you'll be able to easily identify bad email addresses and non-existent domains.</td>
</tr>
<tr>
<td></td>
<td>Buy Now</td>
</tr>
<tr>
<td>US Address Verification</td>
<td>Put the United States Postal Service to work for you. By using the USPS to verify, correct and enhance any address in the U.S., corrects addresses, adds ZIP + 4 data, provides delivery point verification, gives congressional districts, carrier routes and more.</td>
</tr>
<tr>
<td></td>
<td>Buy Now</td>
</tr>
<tr>
<td>AddressDoctor Global Address Verification</td>
<td>Think of this as address verification on steroids. Verifies and corrects addresses in over 240 countries. Plus, it provides additional formatting options, too, like specifying country of origin and preferred language.</td>
</tr>
<tr>
<td></td>
<td>Buy Now</td>
</tr>
<tr>
<td>Canada Address Verification</td>
<td>Are your potential Canadian customers for real? Canada Address Verification uses the Canada Post national databases to correct and verify addresses from any Canadian location.</td>
</tr>
<tr>
<td></td>
<td>Buy Now</td>
</tr>
</tbody>
</table>
Issues

- DaaS concerns include QoS, DQ, service licensing, data licensing, data governance, etc.
- There is a lack of techniques for the publishing, discovery, selection and evaluation of data concerns
- There is a lack of techniques for integrating concerns for DaaSs
  - Data concerns and Service APIs concerns
## The Importance of Concerns in Data Consumer’s View

<table>
<thead>
<tr>
<th>Concerns</th>
<th>Read-only DaaS</th>
<th>CRUD Daas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Quality</td>
<td>Important factor for the selection of DaaS. For example, the accuracy and completeness of the data, whether the data is up-to-date</td>
<td>Expected some support to control the quality of the data in case the data is offered to other consumers</td>
</tr>
<tr>
<td>Data source</td>
<td>Important factor for the trustworthiness of the DaaS.</td>
<td></td>
</tr>
<tr>
<td>Data &amp; Service Usage</td>
<td>Important factor, in particular, price, data and service APIs licensing, law enforcement, and IPRs</td>
<td>Important factor, in particular, price, service APIs licensing, and law enforcement</td>
</tr>
<tr>
<td>Data Governance</td>
<td></td>
<td>Important factor, for example, the security and privacy compliance, data distribution, and auditing</td>
</tr>
<tr>
<td>QoS</td>
<td>Important factor, in particular availability and response time</td>
<td>Important factor, in particular, availability, response time, dependability, and security</td>
</tr>
<tr>
<td>Service Context</td>
<td>Useful factor, such as classification and service type (REST, SOAP), location</td>
<td>Important factor, e.g. location (for regulation compliance) and versioning</td>
</tr>
</tbody>
</table>
Conceptual Model for DaaS
Concerns and Contracts

Check http://www.infosys.tuwien.ac.at/prototyp/SOD1/dataconcerns/
Capability Concerns

- **Data Quality capabilities**
  - Based on well-established research on data quality
    - Timeliness, up-to-date, free-of-error, cleaning, consistency, completeness, domain-specific metrics, etc.
  - We mainly support the specification of DQ metrics for the whole DaaS but possible to extend to the service operation level

- **Data Security/Privacy capabilities**
  - Data protection within DaaS, e.g. encryption, sensitive data filtering, and data privacy
  - Many terms are based on the W3C P3P
Capability Concerns (cont.)

- Auditing capabilities
  - Logging, reporting (e.g., daily, weekly, and monthly), and warning
  - Support system maintenance, SLA monitoring, billing, and taxation

- Data lifecycle
  - Backup/recovery, distribution (e.g., a service is in Europe but data is stored in US), and disposition
  - Support system maintenance but also regulation on data
Capability Concerns (cont.)

- Data and service license
  - Usage permission: for data (distribution, transfer, personal use, etc.) and for service APIs (adaptation, composition, derivation, etc.)
    - We utilize some terms from ODRL/ODRL-S
  - Copyrights
  - Liability: e.g., who is responsible for the loss due to a network disruption?
  - Law enforcement (e.g., US or European court)
  - Domain specific IRPs
Data Source Concerns

- A DaaS may utilize data from many sources.
- Similar DaaSs may utilize data from the same source.
- Data source properties
  - Name: e.g. ddfFlus or DataFlux or Mr A
  - Size
  - Timespan: the duration of collected data, e.g., more than 4 years in the eBay Data License
  - Update Frequency: how often the data is updated
  - Etc.
Service Context Concerns

- Location:
  - Selecting a DaaS in Amazon US Zone or European Zone?
- Service Type: REST or SOAP?
  - E.g., mobile client daas
- Level of Service
- Service Classification
  - Based on UNSPSC Code Classification Services
- Data Classification
- Service/data versioning
XML Diagram for the DaaS Capability Specification
Domain-specific aspect: Data Concerns for Context Information in Smart Environments

- **Quality of context (QoC)**

<table>
<thead>
<tr>
<th>Classification</th>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Characteristics</td>
<td>Accuracy</td>
<td>Extent to which data is correct</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>Degree of exactness with which measurements agree</td>
</tr>
<tr>
<td></td>
<td>Granularity</td>
<td>Degree of detail with which measurements are described</td>
</tr>
<tr>
<td></td>
<td>Time Period</td>
<td>Time interval between two measurements</td>
</tr>
<tr>
<td></td>
<td>Sensor State</td>
<td>Physical state of sensor</td>
</tr>
<tr>
<td></td>
<td>Sensor Range / Span</td>
<td>Maximum distance for which measurements are valid</td>
</tr>
<tr>
<td>Measurement Context</td>
<td>Measurement Time</td>
<td>Time of collection of context in context collection</td>
</tr>
<tr>
<td></td>
<td>Sensor Location</td>
<td>Location of sensor when context information is collected</td>
</tr>
<tr>
<td></td>
<td>Information Entity</td>
<td>Location of the real world entity about which context is collected</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>At the time of collection of context</td>
</tr>
<tr>
<td></td>
<td>Available Attributes</td>
<td>Number of attributes that have a value for that context object</td>
</tr>
<tr>
<td>Specifications and Consumer Requirements</td>
<td>Validity Time</td>
<td>Maximum length of time for which a specific type of context information is stable</td>
</tr>
<tr>
<td></td>
<td>Required Attributes</td>
<td>Number of attributes that are required to have a value for that type of context information</td>
</tr>
<tr>
<td></td>
<td>Critical Value</td>
<td>Level of importance of context information of a specific type</td>
</tr>
<tr>
<td></td>
<td>Access Level</td>
<td>Information about the rights of context consumer to access certain type of information</td>
</tr>
<tr>
<td>QoC Metric</td>
<td>Reliability</td>
<td>Indicates the extent to which context can be considered credible</td>
</tr>
<tr>
<td></td>
<td>Timeliness</td>
<td>Indicates validity of context to use considering its freshness</td>
</tr>
<tr>
<td></td>
<td>Completeness</td>
<td>All aspects of phenomenon in the environment have been shown</td>
</tr>
<tr>
<td></td>
<td>Significance</td>
<td>Critical value of context information for a specific application</td>
</tr>
<tr>
<td></td>
<td>Usability</td>
<td>Indicates suitability of for use for an intended purpose</td>
</tr>
<tr>
<td></td>
<td>Access Right</td>
<td>Indicate the extent to which owner of context allows the context consumer to access context</td>
</tr>
<tr>
<td></td>
<td>Representation</td>
<td>Extent to which context representation format is consistent to consumer requirements</td>
</tr>
</tbody>
</table>

Table 1 Brief description of concepts in QoC processing model
A service contract includes a set of generic, data-specific and service-specific conditions established based on concerns.
Populating DaaS Concerns

The role of stakeholders in the most trivial view

Data provider
DaaS Provider

DaaS Concerns

Consumer

Third-party service

evaluate, specify, publish and manage

specify, select, monitor, evaluate

monitor and evaluate
- StrikerIron Web services
- Xignite Web services
Service Classification

- **ServiceObjects**
  - Web Services

- **WebservicesX**
  - Web services

- **XWebService**
  - Web services

Department of Telematics, NTNU, 23 Aug, 2010, Trondheim
Concerns in HTML descriptions

- 29 services from 7 providers, most are SOAP-based
Concerns of DaaSs for Scientific Data

From the DaaS description point of view

<table>
<thead>
<tr>
<th>Service Registries</th>
<th>DQ</th>
<th>QoS</th>
<th>Business</th>
<th>Licensing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Ownership</td>
</tr>
<tr>
<td>GBIF</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Usage permission</td>
</tr>
<tr>
<td>EBI Web Services</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>EMBRACE Service Registry</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BioCatalogue</td>
<td>No</td>
<td>No</td>
<td>unstructured</td>
<td>unstructured</td>
</tr>
</tbody>
</table>

From the DaaS description point of view
Data Concern Evaluation
Data concern-aware service engineering process
Data Concern Evaluation

- Evaluation scope
  - Data resources
  - Service Operation
  - Service as a whole

- Evaluation modes
  - Off-line and on-the-fly

- Integration models
  - Push versus pull
  - Pass-by-value versus pass-by-reference
Possible Data Concern Evaluation

Pull, pass-by-references

Pull, pass-by-values
Active data sources, sensors

Push, pass-by-values

Domain-specific: smart environment
Data Concern Publishing

- Off-line publishing of data concerns
  - suitable for static data concerns
  - the publishing of data concerns of a data resource is separated from the service operation which provides the access to the data resource
- On-the-fly publishing of data concerns by associating concerns with retrieved data resources
  - The resulting data resources (e.g., via queries) are annotated with data concerns evaluated by data concerns evaluation tools.
  - suitable for providing dynamic data concerns
- On-the-fly publishing of data concerns through queries
  - the use of different service operation parameters to query data concerns of data resources
  - suitable for validating data concerns before accessing data resources
QoD Framework

- A proof-of-concept implementation of data concern-aware service engineering process
QoD Framework: Publishing Concerns

- Common data concern publication specification
- A tool for providing data concerns according to the specification
QoD Framework: Publishing Concerns

- Using parameter convention
  - Based on metric names in the data concern specification
- Specifying requests by using utilizing query parameters the form of metricName=value

GET/resource?accuracy="0.5"&location="Europe"
QoD Framework: Data Concern Annotation

```java
@GET
@Produces("application/xml")
public String getXml(@PathParam("id") String id, @QueryParam("QoD") String QoD) throws Exception {
    // ...
    DataObject dataobject = getDataResourceByID(id);
    // ...
    // return the requested data resource only
    if (QoD == null) {
        return toXML(dataobject, "http://www.undata-api.org", "results");
    } // evaluate and return only the quality of data of the requested data resource
    if (QoD.isEmpty()) {
        QoDUNDataEvaluator qodEval = new QoDUNDataEvaluator(dataobject);
        qodEval.setType(true);
        qodEval.evaluate();
        return qodEval.getMetricsInXML();
    } // evaluate and return the requested data resource and its quality of data
    if (QoD.equals("annotation")) {
        QoDUNDataEvaluator qodEval = new QoDUNDataEvaluator(dataobject);
        qodEval.setType(true);
        qodEval.evaluate();
        DataObject resultObject = dataobject;
        // ...
        resultObject.getSequence().add("qod", qodEval.getDataConcern().getDataConcern());
        return toXML(resultObject, "http://www.infosys.tuwien.ac.at/SOD1/undata-api", "results");
    } // ...
}
```
Data privacy for DaaS: Privacy Concern Model

- Data privacy concerns are annotated with WSDL and MicroWSMO
Data Privacy for DaaS: Example for Twitter Data

http://infochimps.org/datasets/twitter-haiti-earthquake-data

Department of Telematics, NTNU, 23 Aug, 2010, Trondheim
Applications of Concerns
How can we utilize data concerns?

- **Service composition and execution**
  - Adaptive, context-aware service selection and execution algorithms
    - Can be extended to cover data concerns
  - Runtime data concerns access
    - Data quality-aware adaptation of services
  - Data and service contract compatibility
  - Filtering irrelevant service information

- **Application domains**
  - DaaSs in the Cloud, data composition, context-aware computing
Service Contract Compatibility
SECO2: Motivation and Background

- Besides a WSDL document stating the offered functionalities, a Web Service can be characterized by a *service contract*.
- A service contract
  - establishes the understanding between a service consumer and a service provider;
  - specifies conditions on non-functional parameters (NFPs)/concerns, such as:
    - *Quality of Service* (e.g., response time);
    - *Business terms* (e.g., service price);
    - *Context terms* (e.g., service coverage);
    - *License terms* (e.g., limitation of liability).
- No/several standard languages for service contract descriptions
  - Several proposals (e.g., WSLA, WSOL, ODRL-S, WS-Policy)
Motivation and Background (cont.)

- The heterogeneity of languages specifying contracts
- The compatibility among services in a composition
- The compatibility between a (composite) service and a consumer’s specific-conditions

Yahoo! MVS covers the Europe and accepts a maximum of 100 requests in a day

Purchase Processing Service (PPS)
Merchant Validation Service (MVS)
Payment Verification Service (PS)
Shipping Evaluation Service (SES)
Purchase Validation Service (PVS)
Motivation and Background (cont.)

Past research...

- has neglected contracts of composite services when performing service composition
  - by considering mainly functional parameters
  - by assuming that contracts are described by a single language.
- has not focused on tools and algorithms dealing with contract compatibility evaluation when combining different services from different providers.
  - mainly contract negotiation between consumer and service in a point-to-point manner.
Motivation and Background (cont.)

- Some works address QoS-based compatibility for control flows of service compositions.
- Currently, no techniques to check contract compatibility for data (i.e., the input/output of services), whose contract terms are not always the same to that of the service operations.
  - An example is Google Maps: a *free-for-charge service* but the *copyrighted data* (i.e., the maps)
  - There is still a big debate on data licensing but you can sell your data, e.g., see http://infochimps.org/
- QoS, Business, License and Context terms differently influence data/control flows of the service composition.

<table>
<thead>
<tr>
<th></th>
<th>control flow</th>
<th>data flow</th>
<th>independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality of Service (QoS)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service Context</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>License</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Data and control flows in contract compatibility evaluation
The SeCO² Framework

- SeCO² deals with service contract compatibility by considering:
  - two aspects – service APIs and provided data concerns;
  - a rich set of contract properties (e.g., QoS, Data quality, Business, License and Context terms);
  - several service contract specification languages (e.g., WSLA, WSOL, ODRL-S) together.

- SeCO² supports:
  - semantic service contract descriptions (namely, SeCO policies);
  - service contract compatibility evaluation and recommendation;
  - compatibility based on both data and control flows of the service composition;
  - an extensible reference ontology (namely, SeCO reference ontology) and a Contract term knowledge-base;
  - a rich set of mapping and compatibility evaluation rules.
Currently we deal with modeling and mapping service contracts and contract compatibility evaluation among services in a composition.
Modeling and Mapping Service Contracts

- **Problem**: Heterogeneity in service contract specifications.

- Three types of languages for the specification of service contract properties:
  - **Type A** (e.g., ODRL-S): includes languages allowing the specification of predefined properties.
  - **Type B** (e.g., WSLA): includes languages allowing the specification of user-defined properties.
  - **Type C** (e.g., WSOL): includes languages allowing the specification of properties defined in user ontologies.

- Ontology alignment tools cannot be used to fully automate the mapping between different specifications.
Solution: SeCO\textsubscript{2} makes service contracts comparable through the wrapping to specifications (i.e., SeCO Policies) built on a common meta-model

- without loss of information;
- by means of the SeCO Reference Ontology and predefined mapping rules;
- supporting the use of lexical databases (e.g., WordNet) and ontology alignment tools (e.g., H-match).
SeCO Reference Ontology and SeCO Policies

- **SeCO Reference Ontology and SeCO Policies**
  - built on the Policy Centered Meta-model (PCM) [DePaoli08].

- **SeCO Reference Ontology**
  - built applying general modeling rules to profile models;
  - defines expressive descriptions of contract properties.

- **SeCO Policies**
  - represent service contracts defined as clusters of contract property instances.
Mapping Service Contracts

- A proper technique for each type of language
  - Specifications in **Type A** are wrapped applying fixed mapping rules.
  - Specifications in **Type B** and **Type C** can require interactions with service providers to handle the absence of knowledge (i.e., mapping rules).
  - The definition of new mapping rules is supported by lexical databases and ontology alignment tools.

```xml
<?xml version="1.0" encoding="UTF-8"?>
...
<Expression>
  <And>
    <Expression>
      <Predicate xsi:type(equal)
        <SLAParameter>ServiceUsage</SLAParameter>
        <Value>adaptation</Value>
    </Predicate>
  </Expression>
  <Expression>
    <Expression>
      <Predicate xsi:type(equal)
        <SLAParameter>PrePayment</SLAParameter>
        <Value>9.99</Value>
    </Predicate>
  </Expression>
</And>
</Expression>
```

```xml
<mapping>
  <ServiceUsage>
    <pcm:Permissions>
      ...
    </pcm:Permissions>
  </ServiceUsage>
</mapping>
```

```
instance policy1 memberOf pcm#Policy
  pcm#hasNp hasValue {permission1, prePayPayment1}

instance permissions1 memberOf Permissions
  pcm#hasExpression hasValue permissionsExpression1

instance permissionsExpression1 memberOf PermissionsExpression
  pcm#hasOperator hasValue pcm#all
  pcm#hasParameters hasValue adaptation

instance prePayPayment1 memberOf PrePayPayment
  pcm#hasExpression hasValue prePayPaymentExpression1

instance prePayPaymentExpression1 memberOf PaymentExpression
  pcm#hasOperator hasValue pcm#equal
  pcm#hasParameter hasValue 9.99
  pcm#hasUnit hasValue euro

SeCO Policy spec.
```
Evaluating Service Contract Compatibility: activities and flows

Service Contract Mapping

Service Contract Compatibility Evaluation (at the service level as a whole)
Evaluating Service Contract Compatibility

- **Problem:** evaluation of contract compatibility in a service composition.

- **Input:**
  - service composition description in terms of data and control flows;
  - contracts of the services involved in the composition.

- **Output:**
  - compatible/incompatible service contract properties.

- The compatibility is checked considering
  - semantic relations among values associated with qualitative contract properties;
  - constraint operators used to define quantitative contract properties;
  - data and control flows of the service composition.
## Compatibility Evaluation Rules

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Data Flow</th>
<th>Control Flow</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Location</strong></td>
<td>Service Context</td>
<td></td>
<td></td>
<td>Partnership</td>
</tr>
<tr>
<td><strong>Pricing</strong></td>
<td>Business</td>
<td>X</td>
<td></td>
<td>Compatible value list</td>
</tr>
<tr>
<td><strong>Payment (for data usage)</strong></td>
<td>Business</td>
<td>X</td>
<td></td>
<td>Binary, Ternary</td>
</tr>
<tr>
<td><strong>Payment (for service usage)</strong></td>
<td>Business</td>
<td></td>
<td>X</td>
<td>Binary, Ternary</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>QoS</td>
<td></td>
<td>X</td>
<td>Binary, Ternary</td>
</tr>
<tr>
<td><strong>Permissions</strong></td>
<td>License</td>
<td></td>
<td>X</td>
<td>Subsumption</td>
</tr>
<tr>
<td><strong>Data Ownership</strong></td>
<td>License</td>
<td></td>
<td>X</td>
<td>Compatible value list</td>
</tr>
</tbody>
</table>
Evaluating Service Contract Compatibility

Algorithm 1 Compatibility Evaluation

1: for all \( s_i \in S \) do
2:     for all \( s_j \in S(j \neq i) \) do
3:         \( \Omega(s_i, s_j) = \phi \) where \( \Omega(s_i, s_j) \) is a set of triples \([p_w, p_z, \lambda(p_w, p_z)]\)
4:     for all \( p_w \in P(s_i) \) do
5:         for all \( p_z \in P(s_j) \) do
6:             \( \lambda(p_w, p_z) = \phi \), where \( \lambda(p_w, p_z) \) is a set of triples \([pr_i, pr_j, result]\)
7:             \( \Upsilon(p_w, p_z) = \phi \), where \( \Upsilon(p_w, p_z) \) is a set of comparable properties \([pr_1, pr_2]\)
8:             \( \Upsilon(p_w, p_z) = \text{Matching}(p_w, p_z) \)
9:         for all \([pr_1, pr_2] \in \Upsilon(p_w, p_z)\) do
10:            \( \text{rule} = \text{Extract}(pr_1, name) \)
11:            if \( \text{rule} = \text{′} CF - inf ′ \) then
12:                \( \lambda(p_w, p_z) = \lambda(p_w, p_z) \cup \text{EvalRuleF}(\text{rule}, pr_1, pr_2, cf_j \in CF(s_i)) \)
13:            else
14:                if \( \text{rule} = \text{′} DF - inf ′ \) then
15:                    \( \lambda(p_w, p_z) = \lambda(p_w, p_z) \cup \text{EvalRuleF}(\text{rule}, pr_1, pr_2, df_j \in DF(s_i)) \)
16:                else
17:                    \( \lambda(p_w, p_z) = \lambda(p_w, p_z) \cup \text{EvalRule}(\text{rule}, pr_1, pr_2) \)
18:                end if
19:            end if
20:         end for
21:     end for
22: \( \Omega(s_i, s_j) = \Omega(s_i, s_j) \cup [p_w, p_z, \lambda(p_w, p_z)] \)
23: end for
24: end for

For all SeCO Policy couples

Identify comparable SeCO properties

Extract the evaluation rule

Evaluate according to flow influences
Illustrating Example

<table>
<thead>
<tr>
<th></th>
<th>Data Ownership</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Service</td>
<td>Personal-use</td>
<td>100 tr/min</td>
</tr>
<tr>
<td>Yahoo! MVS</td>
<td>Copyrighted</td>
<td>100 tr/min</td>
</tr>
<tr>
<td>XWeb PPS</td>
<td>Free-distribution</td>
<td>100 tr/min</td>
</tr>
<tr>
<td>Aivea SES</td>
<td>Free-distribution</td>
<td>100 tr/min</td>
</tr>
<tr>
<td>WebX PS</td>
<td>Free-distribution</td>
<td>500 tr/min</td>
</tr>
<tr>
<td>DOTS PVS</td>
<td>Free-distribution</td>
<td>500 tr/min</td>
</tr>
</tbody>
</table>
Illustrating Example

- **Data Ownership**:  
  - a *License* term stating how the data are protected;  
  - influences the *data flow* of the service composition;  
  - assumes values characterized by relations of compatibility/incompatibility  
    - *copyrighted* is compatible with *personal-use*  
    - *copyrighted* is incompatible with *free-distribution*

- **Scalability**:  
  - a *QoS* term indicating the maximum number of transactions accepted per minute.  
  - influences the *control flow* of the service composition;  
  - assumes numeric values.
Illustrating Example

- **Data Ownership is evaluated exploiting the axiom:**

  \[
  \text{axiom dataOwnershipCompatibility}\\
  \text{definedBy}\\
  \text{compatible ( ?X , ?Y ) :-}\\
  \quad ( \text{?X memberOf seco#DataOwnValue}) \text{ and}\\
  \quad ( \text{?Y memberOf seco#DataOwnValue}) \text{ and}\\
  \quad \text{seco#compatible( ?X, ?Y)}
  \]

- **Scalability is evaluated applying the algorithm**

  Given pr1,pr2

  if(((pr1,pr2).equals("seq"))||((pr1,pr2).equals("par"))){
      if(pr2.value<pr1.value)
          result = "INCOMPATIBLE";
      else
          result = "COMPATIBLE";
  }
Illustrating Example
Service Information Overloading
Problems

- Too much service information pushed to the composition developer and (semi-)automatic service composition algorithms

- Our solutions
  - Use quality of data metrics to characterize service information
  - Filter service information based on consumers' requests.
# Metrics

Interpretability specifies the availability of documentation and metadata for correct interpretation of service information

\[
\text{Interpretability} = \frac{\sum (\text{score}(\text{category}_i) \times w_i)}{\sum w_i}
\]

<table>
<thead>
<tr>
<th>Category</th>
<th>Service information</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>schema</td>
<td>conceptual service and data schemas</td>
<td>WSDL, SAWSDL, pre/post conditions, data models</td>
</tr>
<tr>
<td>documentation</td>
<td>documents</td>
<td>APIs explanation, best practices</td>
</tr>
<tr>
<td>NFP</td>
<td>non-functional properties</td>
<td>categorization, location, QoS information</td>
</tr>
<tr>
<td>contract</td>
<td>service contracts and contract templates</td>
<td>service level agreements based on NFPs</td>
</tr>
<tr>
<td>provenance</td>
<td>Provenance information</td>
<td>versioning of schemas, NFPs, contracts</td>
</tr>
</tbody>
</table>
Metrics

- Completeness specifies the ratio of missing values of provided NFP information, $\text{NFP}_p$ to the expected minimum set of NFPs, $\text{NFP}_{\min}$

  \[
  \text{Completeness} = 1 - \frac{\|\text{NFP}_p \cap \text{NFP}_{\min}\|}{\text{NFP}_{\min}}
  \]

- Timeliness specifies how current a non-functional property is.

  \[
  \text{Timeliness} = 1 - \left(\frac{\text{Age}}{\text{ExpectedLifetime}}, 1\right)
  \]
Filtering mechanism

- Two types of filtering
  - Interpretability and NFPs.

- NFP-based filtering:
  - Step 1: Extract and establish NFPmin and ExpectedLifetime from the developer’s requirement;
  - Step 2: Evaluate QoD metrics, e.g., Completeness and Timeliness;
  - Step 3: Establish filtering thresholds based on QoD metrics;
  - Step 4: Eliminate services whose information does not meet conditions setup in Step 3;
  - Step 5: Refine the filtering by repeating Step 3.
Example 1: Weather Services

- Using Seekda! 50 weather services
  - service interface, information about documentation, availability, user rating, etc.

- Data preparation
  - score(schema) = 1 as their schemas are basically a WSDL file.
  - Seekda!'s {none, partially, good} = score(documentation) = {0, 0.5, 1}.
  - We assumed NFPmin = {availability, reliability, responsetime} whereas seekda! provides only availability and response time.
  - Provenance information and service contract are missing.
Filtering Data
Example 2: Service Contracts

- Having 100 identical services, each has 5 service contracts
  - 500 WSML service contracts in the PoliMaR framework (http://polimar.sourceforge.net/)
- Looking for a shipping service able to satisfy specified conditions on
  - payment method, payment deadline, insurance, base price and hours to delivery.
  - No older than 1 year before 19 June 2010
- Using following parameters:
  - $\text{NFPmin} = \{\text{payment method, payment deadline, insurance, base price, hours to delivery}\}$
  - $\text{ExpectedLifetime} = 1\text{year}$.
Filtering Evaluation

- Performance evaluation with threshold: Completeness $\geq 0.6$ and Timeliness $> 0.2$.

<table>
<thead>
<tr>
<th>Exp.</th>
<th>Filter 1</th>
<th>Filter 2</th>
<th>Filtered Contracts</th>
<th>Filtering Time</th>
<th>Ranking Time</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>no</td>
<td>no</td>
<td>500</td>
<td>0 sec</td>
<td>37.5 sec</td>
<td>37.5 sec</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>309</td>
<td>2.7 sec</td>
<td>19.9 sec</td>
<td>22.6 sec</td>
</tr>
<tr>
<td>3</td>
<td>no</td>
<td>yes</td>
<td>395</td>
<td>2.2 sec</td>
<td>25.9 sec</td>
<td>28.1 sec</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>246</td>
<td>3.5 sec</td>
<td>14.2 sec</td>
<td>17.7 sec</td>
</tr>
</tbody>
</table>

Table 4. Results of applying Completeness (Filter 1) and Timeliness (Filter 2) filtering activities.
Filtering Evaluation

- With thresholds = \{0, 0.2, 0.4, 0.6, 0.8, 1\} which are equivalent to \{not required, optional, preferred, strong preferred, required, strict required\}
Further Readings


Conclusions

- We present
  - Data concern specification, evaluation and publishing
  - Some applications of data concerns in service composition and execution
  - Data concerns are important for Web mashup, DaaS in the cloud, as well as Sensor-as-a-service

- Several examples are based on quality of data
  - But in principle it can also be applied to data privacy and other concerns

- But there are still several fragment research results that need to be integrated
Future Work

- Open issues
  - Adaptive algorithms based on data concerns are open
  - Evaluation of data concerns is challenging, especially for domain specific data concerns
  - The dependency among data concerns and service concerns
- Integration and experiment
- Data contract

Which topics we can collaborate? And how?
Thanks for your attention!

Hong-Linh Truong
Distributed Systems Group
Vienna University of Technology
Austria

truong@infosys.tuwien.ac.at
http://www.infosys.tuwien.ac.at