INTEGRATION ISSUES OF AN ONTOLOGY BASED CONTEXT MODELLING APPROACH

Thomas Strang
German Aerospace Center (DLR), Institute for Communications and Navigation
D-82234 Wessling/Oberpfaffenhofen, Germany
E-mail: thomas.strang@dlr.de

Claudia Linnhoff-Popien, Korbinian Frank
Ludwig-Maximilians-University (LMU), Institute for Computer Science
D-80538 Munich, Germany
Email: {linnhoff|frank}@informatik.uni-muenchen.de

ABSTRACT

In this paper we analyse the applicability of our ontology based context modelling approach, considering a range of use cases. After wrapping up the model and the Context Ontology Language (CoOL) derived from it, we introduce some interesting applications of the language, based on a scenario showing the challenges in context aware service interactions. We focus on two submodels of our model for context aware service interactions, namely *Context Bindings* and *Context Obligations*, and demonstrate how to integrate them into different existing service architectures.

KEYWORDS

Context Awareness, Context Binding, Context Obligation, Interoperability, Integration, Ubiquitous Computing

1. INTRODUCTION

Determination of interoperability as prerequisite for collaboration is a big issue during any service interaction in distributed systems. Particularly in ubiquitous or even pervasive computing environments [Satyanarayanan M., 2001], the most advanced type of distributed systems, significant improvements of service interactions can be achieved by enabling context awareness. This has been driving a need for some formalism to be able to determine interoperability on the context level [Strang T. and Linnhoff-Popien C., 2003]. For that purpose we developed a Context Ontology Language (CoOL), which may be used to describe contextual facts and contextual interrelationships in a precise and traceable manner and thus may be engaged to determine contextual interoperability. Precision and expressiveness of our language is achieved by using *ontologies* [Uschold M. and Grüninger M., 1996] [Barners-Lee et al, 2001], which are particularly useful to *make implicit knowledge explicit* and thus utilizable by computers. By projecting our contextual base model called *ASC model* to language elements, special applicability to describe contextual facts and interrelationships is given in our language.

In this paper we analyse the applicability of our model, considering a range of use cases. This paper is organized as follows: To outline the complexity of context aware service interaction, and to provide a reference example for the succeeding sections, we introduce a scenario in section 2, as well as a wrap up of our base model in section 3. Section 4 deals with some exemplary simple and complex instances of our base model. The applicability of our language is shown thereafter in section 5, considering a Web Service architecture as an example. We describe how CoOL may be integrated in this architecture to establish context awareness during service discovery and execution, and what are the necessary extensions to the existing protocols and architecture components. We will show in section 6 how it can be seen as a context extension for an ontology of services called DAML-S [Ankolekar A. et al, 2001], before we summarize our paper with a conclusion in section 7.

2. SCENARIO

Imagine a tourist being in a foreign city, having taken some photos with his/her camera-enabled smart mobile and wants to take home some prints of these photos to show them her/his family. Instead of searching and using any online print service (*PhotoShopService*) via some traditional browser technology, the tourist wants to use an optimal online print service w.r.t. the context of the user, any service provider candidate and the environment. Contextual parameters which may influence the decision for the specific selected provider may be for instance the distance between the current geographic position of the user and the next pickup point of that provider, the opening hours of the pickup point, price of production and delivery, production time, current load of the provider etc. Most of these parameters may not be explicitly known, neither to the user nor to the service provider candidate. For instance the tourist may not be able to specify where s/he exactly is in the foreign city. But s/he may be able to specify, who s/he is, and use this information to find out where s/he is through some context provider offering this kind of information. After selecting an appropriate service provider from the list of candidates, the tourist may want to see a map of his current surroundings (*MapService*) and get a routing advice to the next pickup point of the selected online print service, which may be for instance a vending machine at the train station. Again, if the user is unable to specify her/his current position, s/he may request that kind of information from a context provider, and use this information as an input value for a route advisory service.

3. ASC MODEL AND THE CONTEXT ONTOLOGY LANGUAGE

In this section we will give a short wrap up of our model which facilitates the understanding of the concepts used in the succeeding sections. A comprehensive introduction to the model and the language derived from it is given in [Strang T. et al, 2003].

Our *Aspect-Scale-Context (ASC)* model is named after the core concepts of the model, which are *aspect*, *scale* and *context information*. Think of an *aspect* primarily as set of related *scales* of discrete or continous values. A scale is a set of objects defining the range of valid *context information*. In other words, a valid context information with respect to an aspect is one of the elements of the aspect's scales. For instance the aspect "GeographicCoordinateAspect" may have two scales, "WGS84Scale" and "GaussKruegerScale", and a valid context information may be an object instance created with `new GaussKruegerCoordinate ("367032", "533074")` in an object oriented programming language like Java. Scales based on primitive datatypes like scalars instead of objects are captured by corresponding wrapper classes. Thus a valid context information of the aspect "SpatialDistanceAspect" with the given scales "MeterScale" and "KilometerScale" may be an object instance created with `new Integer(10)`.

Each scale is *constructedBy* one class of context information. All scales within one aspect are constrained by the ASC model in a way, that there must exist a mapping function called *IntraOperation* from one scale to at least one other of the already existing scales of the same aspect. Like that, it is possible to access every scale from every other scale of the same aspect by a series of IntraOperations. In other words, a new scale of an aspect may be virtually constructed by providing an IntraOperation from an existing scale. This allows to build multiple related scales by providing different IntraOperations representing different scaling factors ("nautical miles", "km" or "m" for a "SpatialDistanceAspect" aspect). Scales which require access to scales of one or more other aspects can be defined using *InterOperations*. An example for such a scale would be "KilometerPerHourScale" of a "SpeedAspect" aspect. This scale can be defined using an InterOperation with two *Parameter*, delta_s and delta_t, where the parameter delta_s is from an aspect "SpatialDistanceAspect" and delta_t is from an aspect "DurationAspect".

Due to the fact that a scale is an unordered *set* of context information instance objects, there may be no relative sort order between the context information inherently given. Therefore we introduced the *MetricOperation* which may be used to compare two context information instance objects of the same scale in an implementation-defined manner to see if they match or what their relative sort order is by returning either the first or the second parameter. Thus the return value indicates the ordering of the two objects.

Information about the signature of any InterOperation, IntraOperation or MetricOperation is available in the signature specification pointed to with the property *identifiedBy*, e.g. an operation within a WSDL file or an AtomicProcess within a DAML-S grounding.

Each context information has an associated scale defining the range of valid instances of that type of context information. Context information characterizing the content of another context information is a meta information and thus a context information of higher order and expresses the quality of the lower order context information. Our Context Ontology Language includes already a set of standard quality aspects like a *minimumError*, a *meanError* and a *timestamp*, but any other kind of context information characterizing the quality of another context information may be assigned to the context information of interest using the *hasQuality* property of the ASC model.

Our Context Ontology Language is mainly a projection of our ASC model into a XML based markup language, supplemented by a set of instance documents (*ontology artifacts*, see next section) useful for referring when describing contextual facts. The language is defined using either DAML+OIL or OWL, which are both ontology languages based on XML. Thus the correctness of statements defined in CoOL may be validated using any appropriate DAML+OIL or OWL validator. In our system, after validation of any CoOL document in the expressed manner, these documents are converted to F-Logic documents, loosing some of the expressiveness of DAML+OIL respectively OWL, but enabling a much more effective use towards a backend component of the context provider domain: the OntoBroker reasoner [Decker S. et al, 1999]. F-Logic, for instance, is much more appropriate for specifying *relevance conditions* (what are the conditions for considering an entity and/or specific context information to be relevant).

4. ASC ARTIFACTS

We defined a catalog of basic aspects and scales (*ontology artifacts*) for testing and evaluation purposes using the concepts introduced with the ASC model. Among them are the following:

AbsoluteTimeAspect allows to specify a point in time on a *UTCScale* or any related timezone scale, which are all double-linked to the UTCScale by corresponding IntraOperations.

DurationAspect maps a time period to the number of milliseconds since a specific, but variable point in time. Thus its default *DurationScale* maps to the natural numbers. There exists a *TimeRepresentationScale* which maps each value from the DurationScale to a format better readable for humans by the use of a *getRepresentationFromDuration* IntraOperation.

TimePeriodAspect is an aspect of two scales, *AbsAbsTimePeriodScale* and *AbsRelTimePeriodScale*. Both scales are constructed by an InterOperation *getAbsAbsPeriod* respectively *getAbsRelPeriod* having two Parameter. The first parameter is a value from the *UTCScale* of the *AbsoluteTimeAspect*. The second parameter of getAbsAbsPeriod is also from *UTCScale*, whereas the second parameter of getAbsRelPeriod is a value from the default *DurationScale* of the *DurationAspect*.

GeographicPlaceAspect covers geographic position information. This aspect has the two scales *WGS84Scale* and *GaussKruegerScale*, which are double-linked via the IntraOperations *getWGS84fromGK* and *getGKfromWGS84*.

SymbolicPlaceAspect covers symbolic position information as string-based description ("Building 122, Room 217").

EventAspect has a single scale containing an unordered set of eventIds.

PriceAspect may be used to characterize an entity w.r.t. a price in different currencies, each defined in its own scale. Each currency scale is double-linked to the default *EuroCurrencyScale* by IntraOperations *getEURfromXXX* and *getXXXfromEUR*.

SpatialDistanceAspect allows to specify a value equal to or greater than zero based on the scales *NauticalMilesScale*, *KilometerScale* or *MeterScale*, each of them linked via the matching IntraOperation. The KilometerScale has an additional InterOperation *getDistanceBetweenGaussKrueger* for calculating a certain distance between two Gauss-Krueger-Coordinates.

AirlineClassAspect consisting of two enumeration-based scales called *ComfortClassScale* and *BookingClass-Scale*. The mightiness of these two scales differ, because there are many more booking classes than comfort classes (first, business and economy), which is covered by the corresponding IntraOperation.

WeatherAspect as base for complex descriptions of weather conditions, e.g. regarding humidity, temperature, wind, clouds etc.

SpeedAspect may be used to specify indications of speed based on the three scales *KnotsScale* (e.g. for horizontal speed in aeronautics), *FeetPerMinuteScale* (e.g. for vertical speed in aeronautics) or *KilometerPerHourScale* (e.g. for horizontal terrestrial traffic). Each scale is (in our example) constructed by an InterOperation with a parameter from a geographic distance aspect and a parameter from the *DurationAspect*, which calculates the speed in a delta_s devided by delta_t manner.

This list is neither representative nor complete. In fact, one of the main concerns of designing CoOL has been to be able to create and extend the list of individual aspects, scales and types of context information.

5. APPLIED ASC: BINDINGS AND OBLIGATIONS

Our ASC model may be applied at diverse places in service interaction architectures to describe contextual facts and relationships. In this section we will focus on two of them. The first one is what we called the *ContextBinding*, which may be used to establish a virtual link from some input or output parameter of a service operation to a specific aspect, enabling automatic determination of valid or even optimal parameters. The second one are the *ContextObligations*, which are the obligations of a service w.r.t. to the context of its usage (e.g. the geographic scope "delivery area" covered by the service with respect to a well defined aspect "RegionAspect").

5.1 Context Binding in Web Services

Remember the scenario introduced in section 2. If the user is unable to specify his current position, he may request the context information w.r.t. the aspect of interest from a context provider and use this information as an input value for a PhotoShopService or MapService himself. Applied to a Web Service environment using SOAP [Box D. et al, 2000] and WSDL [Christensen E. et al, 2001] that means to invoke the context provider in the role of a service provider and proceed with the data in a user proprietary way, e.g. using the received context information as input for a MapService invocation, see figure 1.

Figure 1. Direct Service Invocation

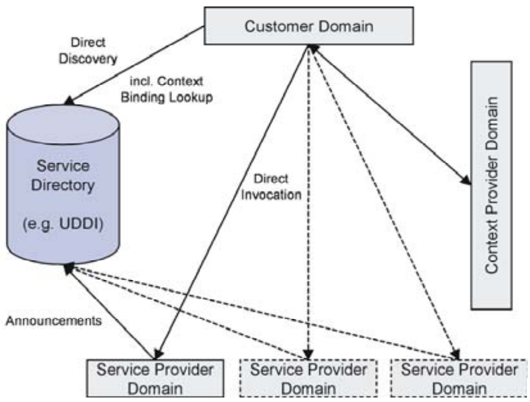
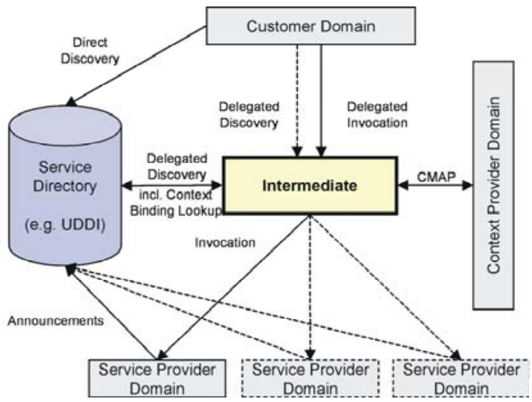


Figure 2. Delegated Service Invocation



Alternatively he could employ a middleware component of a context aware service platform, delegating the task of searching for any matching or even optimal value needed in a specific service interaction, encapsulating the context handling at the middleware component.

This alternative leads to the introduction of an *Intermediate SOAP Node* [Box D. et al, 2000] providing a *Context Management Access Point (CMAP)* interface [Strang T. et al, 2003], see figure 2. Through the CMAP interface any in a service interaction involved party from customer domain, service provider domain or even a third party domain may specify *relevance conditions*, which are filters that can be used to identify one or more relevant entities out of the set of all known entities in the context provider domain. Moreover, the interface can be used to specify the aspects of interest relevant for a service interaction, as well as some quality limits constraining the quality of the context information.

Our approach fits very well to the Web Service architecture and its protocols. For instance, interoperability on the signature level is usually established by providing a common service signature description in a WSDL file. WSDL itself is defined in XML schema in an extensible way. We provide a *Context Binding* XML extension scheme, enabling the binding of any input or output parameter of a given WSDL service description to a specific aspect by adding a few additional attributes, see figure 3. These attributes may be used during service discovery to detect a binding to an aspect, enabling the user or the intermediate to react to the requirement of providing appropriate information for each parameter marked in that way during service execution.

Figure 3. Excerpt from extended signature description: SampleMapService.wsdl

```
…
<wsdl:message name="showMapRequest"> <!-- input msg -->

      <wsdl:part   <!-- a parameter of the input msg -->

        <!-- standard WSDL attributes for wsdl:part  -->
          name="currentPositionParam"
          type="xsd:string"

        <!-- add. attribs expressing context binding -->
          xmlns:cb="http://context-aware.org/schema/wsdl-cb"
          cb:aspect="urn:asc-a#GeometricPlace_Aspect"
          cb:scale="urn:asc-a#GaussKrueger_Scale" />

</wsdl:message>
…
```

Even if a service user is unable to provide information matching the required aspect as marked in a WSDL context binding, it may be possible by the context provider to deliver that information, if the information necessary to identify the context information is known to the context provider. For instance the context provider may have some contract with the mobile network operator, so that the context provider may request the current position of the user from the mobile network operator with an appropriate entity identifier, which is usually the mobile phone number. This entity identifier may be no existential parameter of the original service signature, which are encoded in the SOAP body of Web Service calls usually. Thus we added this kind of in-band meta information as SOAP header element, with `<cb:ContextBinding ..>` as an anchor, see figure 4 for an example.

Figure 4. Context Binding Header in a SOAP Envelope

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">

   <SOAP-ENV:Header>
     <cb:ContextBinding xmlns:ps="urn:PhotoShop"
       xmlns:cb="http://context-aware.org/schema/soap-hdr-cb"
       SOAP-ENV:mustUnderstand="0" cb:part="ps:pickupPoint" >
      <Entity identifiedBy="urn:PhoneNumber#+4917998765" />
     </cb:ContextBinding>
   </SOAP-ENV:Header>

   <SOAP-ENV:Body>
     <ps:selectPickupPoint xmlns:ps="urn:PhotoShop">
       <orderId xsi:type="xsd:int">4711</oid>
       <pickupPoint xsi:type="ps:PickupAdr" ... />
     </ps:selectPickupPoint>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

The attribute `SOAP-ENV:mustUnderstand` may be used to signal if the intermediate MUST (`SOAP-ENV:mustUnderstand="1"`) set the part of the message, or MAY (`SOAP-ENV:mustUnderstand="0"`) set it. The first one may be useful, if the user is not able to provide a contextually bound parameter at all, the latter one, if the intermediate should be enabled to replace a given one by a context-based better one. Note: Context bindings are not specific to WSDL documents, even if we defined a context binding extension schema to WSDL. For instance, we will show in section 6 how context bindings are linked to another kind of service description. But the `SOAP-ENV:mustUnderstand` attribute is defined as part of SOAP, not within WSDL, so the use in the sense described above is healthy in any SOAP based service interaction, regardless of the service signature description technique in use.

We defined several other elements which may be expressed as SOAP header tags to specify relevance conditions, optimisation criteria, select rules etc. Among them are

- `<cb:Entity identifiedBy=".." />` (see example in figure 4)
- `<cb:MinQuality identifiedBy=".." aspect=".." scale=".." metric=".." />`
- `<cb:MaxQuality identifiedBy=".." aspect=".." scale=".." metric=".." />`
- `<cb:NearestQuality identifiedBy=".." aspect=".." scale=".." metric=".." />`
- `<cb:RawFLogic .. />`

the most important ones. Particularly `<cb:RawFLogic .. />` may be used to specify complex raw F-Logic rules, which are forwarded to the context provider by the intermediate, and evaluated in the context provider's inference engine, delivering appropriate context information according to the given rule.

5.2 Context Obligations in Web Services

A context obligation is a guarantee of a service provider to maintain its state w.r.t. a specific aspect within the limits expressed by the obligation. As such, it is a scale for a context information characterizing a service. A parameter characterizing a specific service instance is known from carrier services as Quality-of-Service (QoS) parameter. This has been the reason for modelling QoS parameters as a specialization of context information in [Strang T. et al, 2003].

The limits defined within an obligation constrain the context information, which characterizes a service, to a certain subset of a scale of a specific aspect. To be able to constrain a scale with predicates like “LessOrEqual”, a MetricOperation must be provided, which defines the relative sort order on a scale. The context information may be further constrained by an arbitrary amount of context information of higher order, each based on a quality aspect. We use a range of obligations to characterize a service from a context perspective. Among them are for instance *GeographicScope*, *TimeScope* and *LegalScope*. See figure 5 for an exemplary context obligation.

Figure 5. Context Obligation for PhotoShop Pickup Places

```
<?xml version='1.0' encoding='UTF-8'?>
<co:ContextObligation xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:co="http://context-aware.org/schema/co">

  <Obligation name="GeographicScope">
    <Entity id="http://provider.com#MapService"/>
    <Predicate xsi:type="cb:LessOrEqual">
      <Aspect id="urn:SpatialDistanceAspect"/>
      <Scale id="urn:KilometerScale">
        <InterOperation id="getDistanceBetweenGaussKrueger">
          <Param id="GKCoord_1">367029 533256</Param>
        </InterOperation>
      </Scale
      <MetricOperation id="urn:OrderedRealNumber"/>
      <ContextInformation xsi:type="xsd:float">50.0</ContextInformation>
    </Predicate>
  </Obligation>

</co:ContextObligation>
```

Context obligations have some similarities with service level guarantees in WSLA [Ludwig H. et al, 2002]. But compared to WSLA, the expressivenes of context obligations is higher because they have a stronger binding to well defined ranges and domains by using aspects and scales. Furthermore they allow an arbitrary amount of meta information about the obligation, whereas WSLA restricts them to a *validity* in time only.

6. DAML-S CONTEXT EXTENSION BASED ON ASC

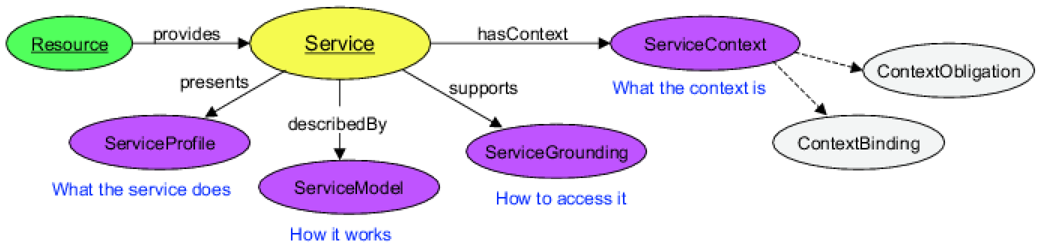
Context Bindings and Context Obligations are not specific to SOAP and WSDL. This section outlines, how they are applied to another family of service description and access protocols.

In the framework of the Semantic Web some serious effort has been done in designing technologies that allow to discover, invoke, compose and monitor web resources. Among them an *ontology of services* called *DAML-S* [Ankolekar A. et al, 2001] has been created, which can be used to create computer-interpretable descriptions of services from multiple perspectives. Within DAML-S there have been identified three essential types of knowledge about a service, characterized by the question it answers:

- What does the service require of the invoking components and provide for them (presented by a *ServiceProfile*),
- how does it work (described by a *ServiceModel*) and
- how is it used (supported by a *ServiceGrounding*).

Some selected elements of the current version of DAML-S find a corresponding counterpart in our Context Ontology Language. For instance, the non-functional attribute *geographicRadius* of a DAML-S ServiceProfile may be expressed as a context information based on an aspect *scope*, which is one of the default aspects within CoOL, whereas the non-functional attribute *qualityRating* may be mapped to some quality aspect. DAML-S specifies currently only a few of these attributes which may be useful to describe the interoperability of a service on the context level. They cover only a few contextual aspects, and their specification is not very formal. To have a much more formal and thus computer-interpretable approach to describe the contextual requirements and impact of a service, we suppose to extend DAML-S with a fourth type of knowledge about a service, dealing with the contextual issues.

Figure 6. DAML-S with Context Extension



This new perspective (we call it *ServiceContext*, see figure 6) may serve as a more formal description of a service's contextual interoperability by providing a comprehensive but extensible model based on the ASC model. The question answered by this new perspective is

- which contextual parameters have an influence on a service during discovery and execution, and what is its general usage context (has a *ServiceContext*).

The obligations of a service w.r.t. the context of its usage can be expressed in a *ContextObligation* submodel, and a *ContextBinding* submodel may be used to establish a virtual link from some input or output parameter of an *AtomicProcess* of a *ServiceGrounding* to a specific aspect (see example in figure 7), enabling automatic determination of valid or even optimal parameters.

Figure 7. DAML-S Context Binding

```
<?xml version="1.0"?>
<!DOCTYPE CoOL [
<!ENTITY xsd  "http://www.w3.org/2000/10/XMLSchema#">
```

```
<!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<!ENTITY grnd "http://wwww.daml.org/services/daml-s/0.7/Grounding.daml#">
<!ENTITY CoOL "http://context-aware.org/schema/cool.owl#"> ]>

<rdf:RDF xmlns="&CoOL;" xmlns:xsd="&xsd;" xmlns:rdf="&rdf;"
         xmlns:grounding="&grnd;" xmlns:cool="&CoOL;">

  <cool:ServiceParameterBinding>
    <Operation>
      <grounding:damlsProcess rdf:resource="urn:PhotoShop.daml#setLocation" />
      <Parameter>
        <PartName rdf:datatype="&xsd;NCName">pickupPlace</PartName>
        <contentFromAspect rdf:resource="urn:place.cool#PostalPlaceAspect"/>
        <contentFromScale rdf:resource="urn:place.cool#PostalAddrScale"/>
      </Parameter>
    </Operation>
  </cool:ServiceParameterBinding>

</rdf:RDF>
```

Further work has to be done to complete the ServiceContext model and its submodels when the DAML-S specification itself is officially released and stable.

7. CONCLUSION

In the previous sections we showed how to use the concepts defined in our context model to specify contextual facts and interrelationships. We listed a catalog of artifacts based on our model which are useful to describe and qualify an exempary set of context information, particularly for characterizing services with respect to typical relevant aspects for service discovery and service usage.

We emphasised on the integration issues of our model and notably the submodels context binding and context obligation. It has been demonstrated, why context bindings are useful to link service parameters to well defined aspects and scales, and how context bindings can be realized for instance in Web Service architectures in two different ways. Further improvements for context aware service discovery and monitoring have been achieved by using context obligations to express a guarantee to maintain a service's state within well defined limits.

Our work presented in this paper will lead to a general context framework based on Web Services, which will provide a wide range of components enabling context awareness at any stage of a service interaction.

REFERENCES

Ankolekar A. et al, 2001. DAML-S: Semantic Markup for Web Services. *Proceedings of the International Semantic Web Workshop*.

Barners-Lee et al, 2001. The Semantic Web. *Scientific American*, Vol. 284, No. 5, pp. 34-43.

Box D. et al, 2000. Simple Object Access Protocol (SOAP), *http://www.w3.org/TR/SOAP/*.

Christensen E. et al, 2001. Web Services Description Language (WSDL). *http://www.w3.org/TR/wsdl*.

Decker S. et al, 1999. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. *Semantic Issues in Multimedia Systems*, pp. 351-369, Boston, USA.

Ludwig H. et al, 2002. Web Service Level Agreement (WSLA). *IBM WebService Toolkit Documentation*.

Satyanarayanan M., 2001. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, pp. 10-17.

Strang T. et al, 2003. A Context Ontology Language to enable Contextual Interoperability. Submitted to *4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS2003)*, Paris, France.

Strang T. and Linnhoff-Popien C., 2003. Service Interoperability on Context Level in Ubiquitous Computing Environments. *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w)*. L'Aquila, Italy.

Uschold M. and Grüninger M., 1996. Ontologies: Principles, methods, and applications. *Knowledge Engineering Review*, Vol. 11, No. 2, pp. 93-155.