Service Interoperabilität auf Kontextebene

Thomas Strang[1], Claudia Linnhoff-Popien[2]

[1]German Aerospace Center (DLR)
Institut für Kommunikation und Navigation
Münchener Str. 20
D-82230 Weßling/Oberpfaffenhofen, Germany
Email: thomas.strang@dlr.de

[2]Ludwig-Maximilians-Universität München (LMU)
Institut für Informatik
Oettingenstr. 67
D-80538 München, Germany
Email: linnhoff@informatik.uni-muenchen.de

Abstract: In Verteilten System ist Interoperabilität eine Grundvoraussetzung für jede Interaktion zwischen verteilten Komponenten. Dieses Paper analysiert eine Vielzahl verschiedener Ansätze auf unterschiedlichen Ebenen, mit denen versucht wird, Interoperabilität zu erreichen, mit einem Fokus auf der Service Interoperabilität. Unsere Untersuchungen motivieren die Einführung der Kontextebene als neue Ebene der Service Interoperabilität. Es wird gezeigt, wie der Kontext im Verhältnis zur Interoperabilität steht, und warum es sinnvoll ist, kontextuelle Service Interoperabilität auf einer eigenen Ebene zu behandeln. Insbesondere die Ubiquitous Computing Systeme profitieren von einem von uns vorgeschlagenen Verfahren zur dynamischen Kontext Assoziation, das sowohl für die Server- wie auch die Client-Seite den Zugriff auf verteilte, sich schnell ändernde Kontext Informationen ermöglicht. Dies wird u.a. durch eine neue, auf XML basierende Context Ontology Language (CoOL) erreicht, mit der ein "gemeinsames Verständnis" der Beziehungen und Abhängigkeiten zwischen Diensten und Kontext Modellen beschrieben werden kann.

1 Einleitung

Moderne IT-Infrastrukturen sind heutzutage in der Regel nach dem Prinzip der Verteilten Systeme entworfen, bei dem eine Menge voneinander unabhängiger Computer miteinander vernetzt sind, und die dabei als kohärentes Gesamtsystem sowohl den Anwendern wie auch verschiedenen Applikationen zur Verfügung stehen [Tan02]. Eine der wichtigsten Eigenschaften Verteilter Systeme ist die Kooperation verschiedener Komponenten zwecks Erreichung gemeinsamer Ziele [LP96]. Dabei ist ein besonderes Merkmal, daß die Unterschiede der beteiligten Komponenten bezüglich der Hardware, des Netzwerks oder der Software sowie die Art der Kommunikation zwischen den Komponenten durch das System angeglichen und damit vor dem Benutzer versteckt werden. Dadurch können Anwender und Applikationen mit einem Verteilten System in konsistenter und einheitlicher Weise interagieren.

Jede Kooperation erfordert von den beteiligten Partnern ein "gemeinsames Verständnis" über Form und Nutzen der ausgetauschten Daten und erbrachten Dienste. Dabei wird die sogenannte *Interoperabilität* zwischen zwei oder mehr Komponenten durch die An-

wendung einer gemeinsamen Spezifikation gewährleistet. Nach [Weg96] ist Interoperabilität die Fähigkeit von zwei oder mehr Softwarekomponenten, trotz unterschiedlicher Programmiersprachen, Interfaces oder Ausführungsplattformen, miteinander kooperieren zu können. Dies ist allerdings nur eine sehr allgemeine Definition der Interoperabilität, wie wir in Abschnitt 2 zeigen werden. Innerhalb der letzten 20 Jahre wurde die Interoperabilität auf vielen verschiedenen Ebenen betrachtet und analysiert, aber die vorgeschlagenen Ansätze zur Lösung der dabei auftretenden Probleme sind auch heute noch unvollständig. Insbesondere der Kontext einer Applikation oder eines Dienstes wird nur unzureichend in den Interoperabilitätsbetrachtungen berücksichtigt. Dies begründet unsere Motivation zur Einführung des *context levels* in Abschnitt 3. In diesem Abschnitt führen wir den Leser in die von uns verwendete Terminologie zur Beschreibung von Kontexten ein, wie Kontext Informationen erfaßt und verarbeitet werden, und zeigen, warum insbesondere Ubiquitous Computing Umgebungen von der Berücksichtigung des Kontextes profitieren. Die Kontextuelle Service Interoperabilität wird in Abschnitt 4 von zwei unterschiedlichen Perspektiven - Kompatibilität und Ersetzbarkeit - analysiert. Es wird motiviert, warum eine Context Ontology Language zur Spezifikation eines gemeinsamen Verständnisses der Beziehungen zwischen Diensten und Kontext Modellen hilfreich wäre, bevor wir unsere Arbeit in Abschnitt 5 zusammenfassen und einen Ausblick auf weitere Anwendungsfelder geben.

2 Interoperatbilität in Verteilten Systemen

Interoperabilität ist ein Ziel, das bereits frühzeitig bei der Entwicklung und Erweiterung von Verteilten Systemen angestrebt wird, und erfordert in der Regel eine Middleware- oder Plattform-Komponente zur Laufzeit. Sie erleichtert die Entwicklung in Bezug auf die Wiederverwendbarkeit von Komponenten, die gewöhnlich an unterschiedlichen Orten von verschiedenen Leuten zu unterschiedlichen Zeiten entwickelt werden.

Bereits zu Beginn der Entwicklung weg von zentralistischen Ansätzen hin zu Verteilten Systemen bestehend aus vernetzten leistungsfähigeren Einzelrechnern Mitte der achtziger Jahre wurde mit dem Vorschlag des *Remote Procedure Calls (RPC)* der erste Schritt auf der Interoperabilitäts-Evolutionsleiter erklommen. Der RPC ermöglicht einen rechnerübergreifenden Funktionsaufruf. Durch Anwendung von normierten Verfahren (z.B. zur Codierung der Aufrufparameter) konnte die Interoperabilität zwischen Hardwarekomponenten unterschiedlicher Hersteller (*Plattform Interoperabilität*) sichergestellt werden.

Ende der achtziger Jahre wurde mit CORBA's *Interface Definition Language (IDL)* die nächste Ebene auf der Interoperabilitäts-Evolutionsleiter erreicht. Durch Verknüpfung verteilter Technologien als Mittel der Abstraktion von Kommunikationsschichten und der Objektorientierung als Konzept der Kapselung und Wiederverwendbarkeit konnte erreicht werden, daß Anwendungen, die in unterschiedlichen Programmiersprachen geschrieben wurden, auf unterschiedlichen Betriebssytemen laufen und verschiedene Netzwerkprotokolle nutzen, zueinander interoperabel sind (*Programmiersprachen Interoperabilität*).

In den neunziger Jahren wurden die Untersuchungen zum Thema Interoperabilität weiter

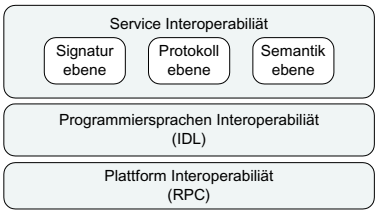

Abbildung 1: Klassische Ebenen der Interoperabilität

intensiviert und vor allem auf der Ebene der Anwendungen und Dienste verfeinert. In der Literatur wird bei der *Service Interoperabilität* üblicherweise auf die drei Ebenen *signature level*, *protocol level* und *semantic level* verwiesen [Hei95, MSW96, VHT] (siehe auch Abbildung 1).

- Auf Signaturebene erfolgt eine exakte Beschreibung der Syntax der nutzbaren Dienste. Dies umfaßt in der Regel den Namen der Operationen sowie Typ und Reihenfolge aller Parameter eines Dienst-Interfaces. Populäre Sprachen zur Interface Spezifikation sind CORBA's *Interface Definition Language (IDL)* oder die *Web Service Definition Language (WSDL)* [CCMW01] bei Web Services. Die Standardisierung der Signaturebene wird heute als am weitesten fortgeschritten angesehen.

- Interoperabilität wird auf Protokollebene durch Festlegung der relativen Ordnung, in der die Methoden eines Dienstes aufgerufen werden, bzw. in der ein Dienst seinerseits Funktionen anderer Dienste aufruft, sowie der Blockierungsbedingungen angestrebt. Während die ersten Ideen zur Interoperabilität auf Protokollebene bereits 1997 von Yellin und Strom in [YS97] veröffentlicht wurden, erfahren entsprechende Ansätze zur Zeit unter dem Begriff *Web Service Choreography Interface (WSCI)* [AAF+02] eine Renaissance. Ebenfalls auf der Protokollebene bestimmen Zugriffsregeln, unter welchen Bedingungen auf einen Dienst zugegriffen werden darf [Mil00]. Eine ausführliche Analyse der Protokollebene wird u.a. in [VHT00] vorgenommen.

- Auf der Semantikebene wird versucht, dem Problem des unterschiedlichen Verständnisses entgegenzutreten. Informationen über die Semantik einer Komponente werden nämlich durch die Beschreibung ihrer Interfaces nicht erfaßt. In [Hei95] wird dazu das Beispiel angeführt, daß nach einer Studie die Wahrscheinlichkeit, daß zwei Datenbankdesigner den gleichen Namen für identische Datenelemente verwenden, nur bei etwa 7% bis 18% liegt. Oftmals ist es so, daß Entwickler und Nutzer einer bestimmten Komponente unterschiedliche Ansichten von deren Einsatzmöglichkeiten und dem Funktionsumfang der Komponente haben. Dies erfordert eine Ontologie [UG96], die versucht, das gemeinsame Verständnis durch ausschließliche Verwendung einer festgelegten Terminologie zur Beschreibung der semantischen Bedeutung zu erreichen. Hier sind auch heute noch zahlreiche Fragestellungen ungelöst. XML-basierte Sprachen wie DAML+OIL [HM01] oder DAML-S [ABH+01, PKPS02], die im Rahmen des Semantic Web (*www.semanticweb.org*) entworfen wurden, sind möglicherweise geeignet, einen Teil der Anforderungen in dieser Ebene zu erfüllen.

Im nächsten Abschnitt wird vorgestellt, in wie weit Spezifikationen auf einer vierten Ebene, dem *context level*, nützlich und anwendbar sind, um die Interoperabilität von Komponenten (insbesondere Diensten) zu beschreiben.

3 Kontextuelle Ebene

Es zeigt sich, daß eine Betrachtung der Service Interoperabilität auf den drei Ebenen *signature*, *protocol* und *semantic* nicht ausreicht. Bereits S. Heiler hat in [Hei95] angemerkt, daß "interessante semantische Informationen kontextabhängig sind". Diese werden jedoch von der semantischen Ebene nur unzureichend erfaßt [VHT00]. Denn auch wenn auf semantischer Ebene das (interne) Verhalten (*behaviour*) eines Dienstes z.B. durch *pre/post conditions* [LW94, ABH+01] oder *abstract type frameworks* [KMK96] beschrieben werden kann, bleiben Bezüge auf die (externe) Situation bisher bei Interoperabilitätsbetrachtungen unberücksichtigt. Daher möchten wir unsere Überlegungen in einer vierten Ebene der Service Interoperabilität, dem von uns vorgeschlagenen *context level*, bündeln (siehe auch Abbildung 2).

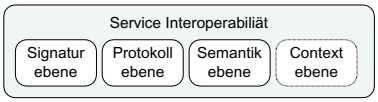

Abbildung 2: Erweiterte Ebene der Service Interoperabilität

Da die Begiffe Kontext und Situation in aktuellen Veröffentlichungen teilweise sehr unterschiedlich interpretiert werden, ist es notwendig, an dieser Stelle kurz die von uns verwendete Terminologie zu definieren:

- Eine *Kontext Information* ist eine Information, die dazu benutzt werden kann, den Zustand einer Entität bzgl. eines Aspekts zu charakterisieren.
- Eine *Entität* ist eine Person, ein Ort oder allgemein ein Objekt.
- Ein *Aspekt* ist eine Klassifikation, Symbol- oder Wertemenge, deren Teilmengen eine Obermenge aller möglichen Zustände darstellen.
- Ein *Kontext* ist die Menge aller Kontext Informationen, die die für eine Aufgabenstellung relevanten Entitäten in den relevanten Aspekten charakterisieren.
- Eine *Entität ist relevant* bzgl. einer Aufgabe (Task, Goal, Perspective, Interaktion, Dienstnutzung), wenn ihr Zustand mindestens bzgl. eines relevanten Apektes charakterisiert wird.
- Ein *Aspekt ist relevant*, wenn während der Erfüllung einer Aufgabe auf den Zustand bzgl. dieses Aspekts zugegriffen, bzw. der Zustand bzgl. dieses Aspekts den Ablauf der Aufgabe beeinflußt.
- Eine *Situation* ist die Menge aller bekannten Kontext Informationen.

Unsere Definition der Terminologie ist in den ersten Punkten sehr nahe derjenigen von Dey [Dey01]. Verglichen mit Dey unterscheiden wir bei der *Relevanz* zwischen einer relevanten Entität und einem relevanten Aspekt. Die oben genannten Definitionen unterscheiden sich von den meisten anderen Definitionen des Kontexts (z.B. [Dey01, SL01, Sch95]) durch die Einführung des *Aspektes*. Man kann sich einen Aspekt primär als eine Achse von diskreten oder kontinuierlichen Werten vorstellen, auf der eine konrete *Kontext Information* eine Instanz dieses Aspekts ist, deren Inhalt den Zustand einer Entität "weich" durch ein

oder mehrere Elemente der Achse charakterisiert. Basiert zum Beispiel ein Aspekt auf kontinuierlichen Wertebereich wie einem Intervall aus den reellen Zahlen, dann könnte eine gültige Kontext Information bezüglich dieses Aspekts eine Wahrscheinlichkeitsdichtefunktion sein [AKR+01]. Für einzelne *Entitäten* kann die Relation zwischen unserer Definition des *Kontext* und mehreren *Aspekten* auf Couderc's Metapher einer "Art Cursor in einem multi-dimensionalen Informationsraums" [BC02, Cou01] abgebildet werden.

Abbildung 3 zeigt die beispielhafte Verwendung der oben eingeführten Terminologie anhand einer spezifischen Kontext Information (geografische Position), die eine bestimmte Entität (Mobiltelefon) bezüglich eines spezifischen Aspekts (Gauss-Krüger Koordinate) charakterisiert, ausgedrückt als XML Instanzdokument.

```
<instance xmlns="http://demo.heywow.com/schema/cool"
          xmlns:a="http://demo.heywow.com/schema/aspects"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <contextInformation>
    <entity system="urn:phonenumber">+49-179-1234567</entity>
    <characterizedBy>
      <aspect name="GaussKruegerCoordinate">
        <observedState xsi:type="a:o2GaussKruegerType">
          367032533074</observedState>
        <units>10m</units>
      </aspect>
      <certaintyOfObserver>90</certaintyOfObserver>
    </characterizedBy>
  </contextInformation>
</instance>
```

Abbildung 3: Beispiel eines Kontext Information Instanzdokuments

Nach der oben eingeführten Definition ist die Situation eines Dienstes die Menge aller bekannten Kontext Informationen, die zu einem Zeitpunkt vor oder während der Dienstnutzung zur Verfügung stehen. Dabei ist es unerheblich, ob diese Kontext Informationen durch den Dienst selbst bzw. dessen Provider, oder über andere Elemente der Infrastruktur bereitgestellt werden. Dies ist ein wichtiges Unterscheidungsmerkmal zu vielen Ansätzen der semantischen Ebene, wo insbesondere unter Verwendung von pre/post conditions lediglich auf den *internal state* eines Objekts zurückgegriffen werden kann [VHT00]. Wir arbeiten daher z.Zt. an einem Verfahren, das es erlaubt, zur Laufzeit Kontext Informationen mit beliebigen Objekten dynamisch zu assoziieren.

Dabei sind *Context Observer* für die Erfassung der Sensordaten und die Darstellung als Kontext Information verantwortlich. Über *Shadow Objects* werden diese Kontext Informationen dann mit den eigentlichen Objekten assoziiert (siehe auch Abbildung 4). Jedes *Shadow Object* ist ein Stellvertreter für ein Instanzobjekt einer Entität, auf die per Netzwerk zugegriffen werden kann, oder auch nicht. Ein Beispiel: Ist das Objekt ein Thermometer, dann repräsentiert das assoziierte *Shadow Object* eine Kontext Information bezüglich eines Temperatur-Aspekts, obwohl das Thermometer möglicherweise selbst nicht über ein elektronisches Interface verfügt (einfaches Quecksilber-Thermometer). Es liegt in der Verantwortung des *Context Observers*, wie die Kontext Information erfaßt wird, z.B. durch periodische Ablesung durch einen Menschen (was allerdings offensichtlich ein Extremfall wäre). Dieses Beispiel zeigt, daß eine Instanz eines *Context Observers* eng gekoppelt zu einer Instanz einer Kontext Information ist. Diese charakterisiert ihrerseits den Zustand ei-

ner spezifischen Entität (Thermometer), stellvertretend repräsentiert durch ein assoziiertes *Shadow Object*, bezüglich eines spezifischen Aspekts (Temperatur).

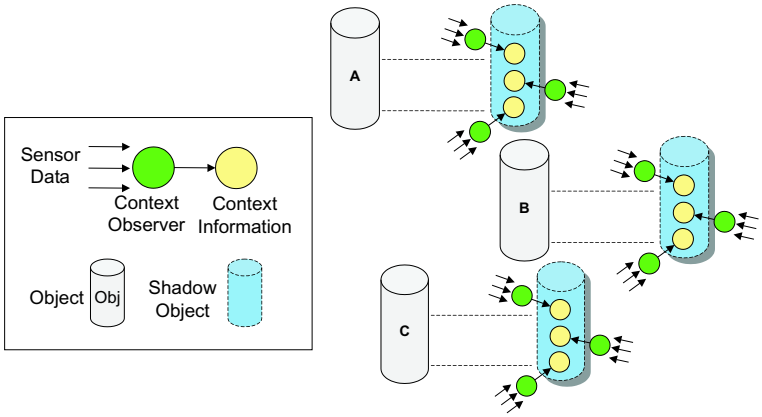

Abbildung 4: Assoziation von Kontext Informationen über Shadow Objekte

Bei Entitäten, auf deren Objektinstanzen direkt über das Netzwerk zugegriffen werden kann (z.B. Mobiltelefone), können die Attribute des Objekts (z.B. Mobiltelefon: Gespräch aktiv?) natürlich direkt über das Objekt Interface und den entsprechenden Dienstendpunkt ausgelesen werden. Darüber hinaus erlaubt die Integration des Verfahrens der *Shadow Objects* in die Middleware (z.B. als *intermediate SOAP node* [GHMF01] mit einer *acting role* analog einem Router) den Zugriff auf Kontext Informationen wie auf zusätzliche Attribute des Objekts, ohne das Objekt selbst adaptieren zu müssen [SLP03].

Die Trennung zwischen Kontext Informationen und ihrer Nutzung im Dienstumfeld im Sinne des *Observer Patterns* [GHJV02] erleichtert die Integration der Ubiquitous Computing Systeme, die seit Beginn des neuen Jahrtausends populär wurden. Das Ubiquitous Computing Paradigma steht im wesentlichen für drei neue Facetten der Nutzung von Verteilten Systemen:

1. Intelligente Kleinstgeräte (Smart Devices)
2. Spontane Vernetzung (Ad-hoc Networking)
3. Kontextadaptive Dienste (Context Awareness)

Dabei ist die kontextuelle Ebene nicht nur für den letzten Punkt von Bedeutung. Auch die spontane Vernetzung profitiert von der Spezifikation, welche Aspekte der jeweils aktuellen Netzwerksituation relevant und somit im Sinne der Ad-hoc Konfiguration zu berücksichtigen sind [AK02]. Smart Devices, insbesondere mobile vernetzte Kleinstgeräte wie Mobiltelefone, PDAs oder einfache Sensor-Module (z.B. Thermometer, Geräuschsensoren etc.) können einerseits wichtige Kontext Informationen über relevante Entitäten (z.B. Personen, Räume etc.) liefern. Andererseits profitieren gerade kleine Geräte mit ihren beschränkten Ein- und Ausgabemöglichkeiten sowie der limitierten Rechenleistung von Informationen, die aus dem Umfeld der Nutzung (wie z.B. die aktuelle Position) gewonnen werden, da

diese Daten z.B. nicht vom Benutzer eingegeben werden müssen. Die kontextuelle Ebene verwendet Kontext Informationen, die in Ubiquitous Computing Umgebungen durch Sensoren erfaßt werden und den Zustand relevanter Entitäten charakterisieren.

Insbesondere aus der Mobilität der Geräte in Ubiquitous Computing Systemen wird eine interessante Kontext Information gewonnen: Die örtliche Position in Bezug auf ein Referenzsystem (z.B. WGS84). Durch Verknüpfung mit symbolischen Informationen ("Gebäude A, Raum 008") und relationalen Informationen ("in der Nähe von") wird aus einer Position eine *Location*. Dienste, die auf den aktuellen Aufenthaltsort als Kontext Information zugreifen, werden unter dem Oberbegriff *Location Based Services (LBS)* zusammengefaßt. Allerdings ist der Ort bei weitem nicht die einzige wichtige Kontext Information [SBG99]. Eine weitere sehr wichtige Art von Kontext Information ist die Zeit. So kann ein Dienst sehr unterschiedliche Ergebnisse liefern, wenn er vor, während oder nach einem bestimmten Zeitpunkt aktiviert wird. Darüber hinaus kann jede von einem Sensor erfaßte Größe als Kontext Information dienen (Position, Zeit, Temperatur, Lichtstärke, Momentangeschwindigkeit etc.). Bei den sogenannten primären Kontext Informationen wird dabei auf die Rohdaten des jeweiligen Referenzsystems (WGS84-Koordinate, Aktuelle Weltzeit, Temperatur in Grad Celsius, Helligkeit in Lux, Geschwindigkeit in km/h etc.) zugegriffen. Im Gegensatz dazu durchlaufen die Rohdaten bei sekundären Kontext Informationen weitere Verarbeitungsstufen. Aus dieser Überlegung leitet sich unmittelbar ab, daß der Wertebereich eines Aspekts einer primären Kontext Information immer das Referenzsystem selbst ist. Bei sekundären Kontext Informationen ist im Gegensatz dazu die Definition einer Klassifikation, Symbol- oder Wertemenge als Aspekt zwingend erforderlich, um die Menge der möglichen Zustände der sekundären Kontext Information geschlossen angeben zu können. Ein Beispiel: Die Menge

$$\mathcal{W}_{TempCelsius} = \{t | t \in \mathcal{R}, t > -273\}$$

definiert den Wertebereich (Menge der möglichen Zustände) einer primären Kontext Information, die den Zustand eines Temperatursensors darstellt. Dagegen definiert die Menge

$$\mathcal{W}_{TempFeeling} = \{warm, cold\}$$

den Wertebereich einer sekundären Kontext Information, die den Zustand bezüglich eines anderen Temperatur-Aspekts charakterisiert. Allerdings obliegt es dem für die Kontext Information zuständigen *Context Observer*, zu entscheiden, welchen Zustand die Kontext Information repräsentiert, beispielsweise anhand einer Schwellwertentscheidung. Ein Kontext Sensor kann als Funktion f verstanden werden, der Daten aus n Sensorquellen fusioniert und einer bestimmten Instanz einer Kontext Information einen gültigen Zustand aus dem Wertebereich $\mathcal{W}$ eines Aspekts zuweist:

$$f(sensor_i | 1 \leq i \leq n) : \mathcal{D} \rightarrow (\mathcal{W}, p) \;\; mit \; p \in \mathcal{R}, 0 < p \leq 1$$

Dabei gibt *p* das Maß der Sicherheit an, mit der ein *Context Observer* von sich behauptet, den korrekten Zustand bestimmen zu können.

Entscheidend ist, daß innerhalb der Definition eines Aspekts festgelegt werden muß, unter welchen Bedingungen welcher Zustand eingenommen wird, nicht jedoch, woher die dazu notwendigen Informationen stammen (d.h. auf welche Sensoren dabei zurückgegriffen wird, Definitionsbereich $\mathcal{D}$), noch wie die Funktion f arbeitet.

Eine Unterklasse von Kontext Informationen stellen die sogenannten *Highlevel QoS* dar. Obwohl die Namensgebung u.U. etwas irreführend sein mag, werden diese und ihre Integrationsmöglichkeiten in Dienstumgebungen in zahlreichen Publikationen untersucht (u.a. in [Wel02, MN02, BV02]). Dabei handelt es sich allgemein um nicht-funktionale Parameter, die die Performanz, Zuverlässigkeit, Verfügbarkeit oder Sicherheit eines Dienstes charakterisieren. Auf diesen Parametern aufbauend kann z.B. innerhalb eines *Web Service Level Agreements (WSLA)* [LDF+02] festgelegt werden, welche Aktion im Fall des Unter- oder Überschreitens eines im WSLA Dokument definierten Grenzwertes von wem ausgelöst werden soll. Die dabei verwendeten Sprachelemente sind zum Teil sehr gut geeignet, um Aspekte und Relevanz-Bedingungen zu definieren. Allerdings ist das Gesamtkonzept von WSLA vor allem sowohl aufgrund des statischen Aufbaus eines solchen "Vertrages" im Form eines einzigen WSLA Dokuments, als auch aufgrund der beschränkten Gültigkeit zwischen genau zwei Vertragspartnern, für uns ungeeignet. Hier bedarf es einer flexibleren Möglichkeit der spontanen Komposition und Überprüfung der momentanen Situation, u.a. ausgehend von Kontext Informationen, deren Existenz zum Zeitpunkt der Erstellung eines WSLA Dokuments noch nicht bekannt ist.

Die nicht-funktionalen Parameter wie o.g. Qualitätsmerkmale werden in der Literatur nicht eindeutig einer der drei "klassischen" Service Interoberabilitätsebenen zugeordnet. Mani und Nagarajan [MN02] sehen diese beispielsweise als Erweiterung einer Interface Spezifikation, und ordnen sie somit der Signaturebene zu. Weller [Wel02] schlägt vor, auf Qualitäts-, Payment- und Deadlock-Informationen parallel zur Dienstausführung zuzugreifen, und siedelt diese dabei auf der Protokollebene an. Meistens werden Informationen, die nicht unmittelbar die Funktion eines Dienstes beschreiben, jedoch, wenn überhaupt, auf der semantischen Ebene behandelt. Ankolekar et. al. [ABH+01] haben z.B. einige wenige als *functional attributes* bezeichnete Charakteristiken wie *geographicRadius* oder *degreeOfQuality* definiert, mit denen nach ihrer Aussage das nicht-funktionale Verhalten beziehungsweise die Bedingungen zur Ausführung von Diensten beschrieben werden können. Aus unserer Sicht macht es jedoch viel mehr Sinn, die Interoperabilität von Diensten bezüglich kontextueller Abhängigkeiten und Möglichkeiten auf einer eigenen Ebene, dem *context level*, zu betrachten. Dies ermöglicht die thematische Fokussierung und erleichtert die Separierung von Diensten, die zwar bezüglich der drei anderen Ebenen interoperabel sind, auf kontextueller Ebene jedoch nicht. Als Beispiel für diesen Fall können die Premiumdienste [Lie02, Sch00] des europäischen Satellitennavigationssystems Galileo dienen. Obwohl die Software eines Empfängers auf allen drei klassichen Ebenen interoperabel zum Galileo-System ist, ist die Nutzung der Premiumdienste (höhere Genauigkeit der Daten etc.) nur im "kommerziellen" Kontext möglich (d.h. gegen Bezahlung), im "freien" Kontext jedoch nicht. Ein anderes Beispiel für diesen Fall sind zwei elektronische Fahrplanauskunfsdienste (EFA) für München und Berlin. Obwohl die EFA Berlin möglicherweise auf allen drei klassichen Ebenen interoperabel zur EFA München ist, gilt dies nicht für die kontextuelle Ebene, wenn z.B. ein ortsbezogener Kontext von

München für die EFA-Abfrage in Berlin zugrunde gelegt wird.

Wichtig ist vor und während einer Dienstnutzung die Relevanz einer Kontext Information. Die Menge der relevanten Entitäten und damit die Kontext Informationen, die diese Entitäten charakterisieren, ändert sich ständig. Die Menge der spezifischen Aspekte, die die Relevanz einer Entität bestimmt, ist jedoch fix! Somit sind es die Aspekte, die auf der kontextuellen Ebene für jeden Dienst spezifiziert werden müssen. Anhand der spezifizierten Aspekte kann vor der Ausführung eines Dienstes ermittelt werden, ob es relevante Entitäten gibt, und über die so gefundenen assoziierten Kontext Informationen entschieden werden, ob die Ausführung des Dienstes (aus kontextueller Sicht) möglich und ratsam ist, oder nicht. Zur Laufzeit des Dienstes beeinflußt die Menge der relevanten Entitäten den Dienst in mehrfacher Hinsicht. Zum Beispiel bestimmt sie, ob die Bedingungen, die zur Auswahl des Dienstes und der daraus resultierenden Ausführung, noch gegeben sind. Falls nicht, wird die Ausführung des Dienstes vorübergehend oder permanent abgebrochen. Ein Beispiel hierfür ist die physische Entfernung von einem Point-of-Interest (POI): Mit zunehmender Distanz macht die Ausführung eines POI-spezifischen Dienstes immer weniger Sinn.

In Anlehnung an eine von Ötztürk und Aamodt eingeführte *Context Ontolgy* [OA97] bezeichnen wir diese Art von externer Kontext Informationen als *environment related context information*, da die kontextuelle "Umwelt" des Dienstes dessen Ausführung beeinflußt. Dem gegenüber stehen die aus Sicht eines Dienstes externen Kontext Informationen, die zur Erfüllung des Dienstes selbst zielführend sind. Diese bezeichnen wir als *target related context information*. Als Beispiel dient hier ein Routen-Dienst von Ort A nach Ort B, wo als Ort A der momentane Aufenhaltsort automatisch vorgegeben wird, der aus der aktuellen Situation ermittelt wird. Wir teilen allerdings nicht die Ansicht von Ötztürk und Aamodt, daß diese externen Kontext Informationen während der Dienstausführung als statisch anzusehen sind, wie das Beispiel der target related context information zeigt: Durch die Mobilität des Benutzers ändert sich der aktuelle Standort ständig. Als Beispiel für eine *internal context information* sei die Anfangszeit eines Meetings am Ort B genannt, die von einem Routen-Dienst bei der Berechnung eines Reiseplans intern zugrunde gelegt wird. Eine Übersicht dieser Klassifikation ist in Abbildung 5 dargestellt.

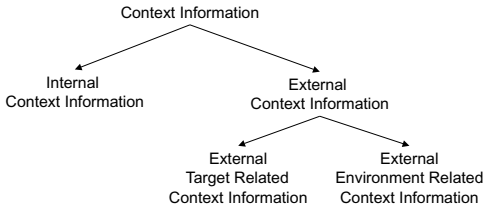

Abbildung 5: Klassifikation von Kontext Informationen

Unsere aktuelle Arbeit hat u.a. zum Ziel, eine *Context Ontology Language (CoOL)* basierend auf XML zu definieren, die es erlaubt, die relevanten Aspekte eines Dienstes auf der einen und eines Dienstnutzers auf der anderen Seite exakt zu spezifizieren. Durch die Möglichkeit, mit Hilfe einer CoOL Spezifikation nicht nur die Art eines Aspektes, sondern auch die möglichen Zustände bezüglich dieses Aspekts festzulegen, kann die Bedeu-

tung einer Kontext Information einerseits geschlossen definiert werden. Andererseits kann durch Anwendung einer anderen CoOL Spezifikation für die gleiche Entität eine hohe Flexibilität und Erweiterbarkeit erreicht werden. CoOL stellt das Bindeglied zwischen den Objekt-assoziierten Kontext Informationen und den Diensten an sich dar, und ermöglicht somit eine kontextadaptive Dienstnutzung, sowie darüber hinaus die Feststellung von kontextueller Service Interoperabilität.

4 Kontextuelle Service Interoperabilität

Eine der zentralen Fragen im Rahmen unserer aktuellen Arbeit ist, wann zwei Dienste bezüglich einer Situation interoperabel sind.

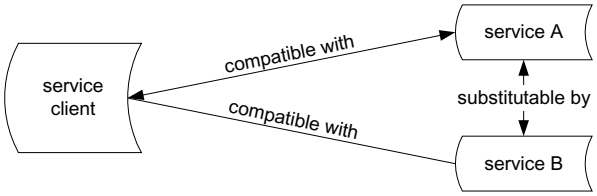

Abbildung 6: Compatibility vs. Substitutability

Vallecillo, Hernández und Troya identifizieren in [VHT00] zwei Eigenschaften, Kompatibilität (*compatibility*) und Ersetzbarkeit (*substitutability*), durch die sich Komponenten in den drei bekannten Kategorien der Dienstebene auszeichnen müssen, um interoperabel zu sein (siehe Abbildung 6).

Auf die kontextuelle Ebene übertragen könnte *Kompatibilität* in dem Sinne definiert werden, daß zwei interagierende Komponenten korrekt zusammenarbeiten, wenn beide Seiten ein identisches Verständnis der relevanten Aspekte haben. Dies beinhaltet die Art der Auswahl der relevanten Entitäten sowie die Menge der Zustände, die die Kontext Informationen in den relevanten Aspekten einnehmen können (einschließlich Verwendung der gleichen Einheiten). Dies kann durch den Bezug auf gemeinsame CoOL Spezifikationen erreicht werden.

Auf der gleichen Ebene kann *Ersetzbarkeit* eines Dienstes durch einen anderen bedeuten, daß die Menge der relevanten Aspekte bei beiden Diensten identisch ist. Wenn also die Zuständigkeit eines Dienstes für eine Aufgabenstellung bei einer bestimmten *environmental related context information* (siehe Abschnitt 3) gegeben ist, so muß ein auf kontextueller Ebene interoperabler Dienst bei dieser Kontext Information ebenfalls zuständig sein. Als Beispiel hierfür seien zwei Dienste A und B genannt, die zu einem gegebenen Ort eine Apotheke mit Notdienst angeben können. Der Dienst A kann dann und nur dann durch Dienst B im Sinne der Interoperabilität ersetzt werden, wenn B für den gleichen Ort, der bei A zu einem Ergebnis führte (Ausgabe einer Apotheke mit Notdienst in der Nähe des eingegebenen Ortes), ein zulässiges Ergebnis ausgibt (gleiche Ortszuständigkeit). Die beiden Ergebnisse müssen nicht notwendigerweise gleich sein. Ähnlich verhält es sich mit den *target related context information*: Ersetzbarkeit von A durch B bedeutet hier, daß B

im gleichen Kontext wie A zielführend ist.

In [Gro] wird ein zweistufiges Verfahren zur Zertifizierung der Interoperabilität vorgeschlagen, das zum Beispiel auch in [LDF+02] angewandt wird: Im ersten Schritt, der logischen Zertifizierung, werden die Spezifikationen der Komponenten auf Konsistenz, Vollständigkeit etc. geprüft. Im zweiten Schritt, der physischen Zertifizierung, wird überprüft, ob eine Implementierung den gegebenen Spezifikationen folgt. Während die logische Zertifizierung in der Regel eher durch menschlische Interaktion erfolgt, kann die physische Zertifizierung häufig sehr gut durch Simulation oder testweise Integration in bestehende Systeme erfolgen.

Eine ähnliche Vorgehensweise empfiehlt sich auch bei der Überprüfung auf Interoperabilität auf kontextueller Ebene. Wenn zunächst die Vollständigkeit, Korrektheit etc. der Definition der Aspekte und damit in Zusammenhang stehende Relevanz-Bedingungen durch logische Zertifizierung sichergestellt ist, kann die physische Zertifizierung leichter durch Automatismen festgestellt werden.

5 Zusammenfassung und Ausblick

Wir haben in den vorhergehenden Abschnitten gezeigt, daß zur Berücksichtigung des Kontexts bei Interoperabilitätsbetrachtungen die Einführung einer kontextuellen Ebene hilfreich ist. Hierdurch können Dienste identifiziert werden, die zwar bezüglich der drei klassichen Ebenen der Service Interoperabilität interoperabel sind, aber nicht bezüglich der Kontext Ebene (oder umgekehrt). Ebenso haben wir die Notwendigkeit einer Kontext Ontologie sowie einer zugehörigen Modellierungssprache motiviert. Eine der Zielsetzungen der skizzierten XML-basierten Context Ontology Language (CoOL) ist es, in der Middleware durch Kombination mit den Objekt-assoziierten Kontext Informationen ein Werkzeug zu haben, das es erlaubt, Szenarien der kontextadaptiven Dienstnutzung entscheidend zu verbessern. Dies ist insbesondere in modernen Ubiquitous Computing Umgebungen bei Verwendung mobiler Multi-Netzwerk-Endgeräte von Bedeutung. Durch Bündelung der Betrachtungen in der neuen kontextuellen Ebene der Service Interoperabilität kann in der Middleware mit diesem Werkzeug entschieden werden, unter welchen kontextuellen Voraussetzungen Dienst und Dienstnutzer interoperabel sind. Darauf aufbauend können z.B. Service Handover höherer Ebenen sowie Konzepte zur Teilautonomisierung realisiert werden.

Literatur

[AAF+02] Assaf Arkin, Sid Askary, Scott Fordin, Wolfgang Jekeli, Kohsuke Kawaguchi, David Orchard, Stefano Pogliani, Karsten Riemer, Susan Struble, Pal Takacsi-Nagy, Ivana Trickovic, and Sinisa Zimek. Web Service Choreography Interface (WSCI). http://www.w3.org/TR/wsci, 2002.

[ABH+01] Anupriya Ankolekar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin,

Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry Payne, Katia Sycara, and Honglei Zeng. DAML-S: Semantic Markup For Web Services. In *Proceedings of the International Semantic Web Workshop*, 2001.

[AK02] Michael Angermann and Jens Kammann. Cost Metrics For Decision Problems in Wireless Ad Hoc Networking. In *Proceedings IEEE CAS 2002*, Pasadena, USA, September 2002.

[AKR+01] Michael Angermann, Jens Kammann, Patrick Robertson, Alexander Steingass, and Thomas Strang. Software Representation for Heterogeneous Location Data Sources within a Probabilistic Framework. In *Proceedings of International Symposium on Location Based Services for Cellular Users (Locellus 2001)*, pages 107–118, Munich, Germany, February 2001.

[BC02] M. Banatre and Paul Couderc. Unleashing context-aware application with Spatial Programming. In *Keynote on IST Mobile Venue 2002: Radio Resource Management and Mobile Location Workshop*, Athens, Greece, May 2002.

[BV02] Manuel F. Bertoa and Antonio Vallecillo. Quality Attributes for COTS Components. In *Proceedings of the 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002)*, Malaga, Spain, June 2002.

[CCMW01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl, 2001.

[Cou01] Paul Couderc. *Mobilité contextuelle dans les systèmes d'information*. PhD thesis, Université de Rennes-1, 2001.

[Dey01] Anind K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing*, 5(1), 2001.

[GHJV02] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley, 2002.

[GHMF01] Martin Gudgin, Marc Hadley, Jean-Jacques Moreau, and Henrik Frystyk. SOAP Messaging Framework. http://www.w3.org/TR/soap12-part1, October 2001.

[Gro] The Distributed Group. Online article: Interoperability. Available at http://www.lsi.us.es/∼tdg/doc/topics/interoperability.html.

[Hei95] Sandra Heiler. Semantic Interoperability. *ACM computing surveys*, (27):271–273, 1995.

[HM01] J. Hendler and D. L. McGuinnes. Darpa Agent Markup Language. *IEEE Intelligent Systems*, 15(6):72–73, 2001.

[KMK96] P. Kähkipuro, L. Marttinen, and L. Kutvonen. Reaching interoperability through ODP type framework. Technical Report C-1996-96, Department of Computer Science, University of Helsinki, June 1996.

[LDF+02] Heiko Ludwig, Asit Dan, Richard Franck, Alexander Keller, and Richard King. Web Service Level Agreement (WSLA). In *IBM WebService Toolkit Documentation*, 2002.

[Lie02] Volker Liebig. Galileo. *Ignition*, pages 2–7, September 2002.

[LP96] Claudia Linnhoff-Popien. *Distributed Systems*. Augustinus, 1996.

[LW94] B. Liskov and J.M. Wing. A behavioural notion of subtyping. *ACM Trans. Prog. Lang. Syst.*, (16):1811–1841, 1994.

[Mil00] Paul Miller. Interoperability: What is it and Why should I want it? http://www.ariadne.ac.uk/issue24/interoperability, 2000.

[MN02] Anbazhagan Mani and Arun Nagarajan. Online article: Understanding quality of service for Web services, 2002. Available at http://www-106.ibm.com/developerworks/library/ws-quality.html.

[MSW96] T. Murer, D. Scherer, and A. Wuertz. Improving Component Interoperability Information. In *Proceedings of Workshop on Component-Oriented Programming (WCOP'96) at 10th European Conference on Object-Oriented Programming (ECOOP'96)*, pages 150–158. dpunkt, July 1996.

[OA97] Pinar Ötztürk and Agnar Aamodt. Towards a Model of Context for Case-Based Diagnostic Problem Solving. In *Context-97; Proceedings of the interdisciplinary conference on modeling and using context*, pages 198–208, Rio de Janeiro, February 1997.

[PKPS02] M. Paolucci, T. Kawmura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *First Int. Semantic Web Conf.*, 2002.

[SBG99] Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. There is more to Context than Location. *Computers and Graphics*, 23(6):893–901, 1999.

[Sch95] William Noah Schilit. *A System Architecture for Context-Aware Mobile Computing*. PhD thesis, Columbia University, 1995.

[Sch00] Ron Schneiderman. Position - Location - Whose Technology? *Wireless Systems Design*, pages 14–20, March 2000.

[SL01] Albrecht Schmidt and Kristopf Van Laerhoven. How to Build Smart Appliances. *IEEE Personal Communications*, August 2001.

[SLP03] Michael Samulowitz and Claudia Linnhoff-Popien. Interaction patterns for dynamic personalisation of service endpoints in Ubiquitous Computing environments. In *Accepted for KIVS 2003*, Leipzig, Germany, 2003.

[Tan02] Andrew S. Tanenbaum. *Distributed Systems*. Prentice Hall, 2002.

[UG96] Mike Uschold and Michael Grüninger. Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[VHT] Antonio Vallecillo, Juan Hernandez, and Jose M. Troya. WOI'00: New Issues in Object Interoperability. In *LNCS 1964: ECOOP'2000 Workshop Reader*, pages 256–269. Springer.

[VHT00] Antonio Vallecillo, Juan Hernández, and José M. Troya. Component Interoperability. Technical Report ITI-2000-37, Departmento de Lenguajes y Ciencias de la Computación, University of Málaga, July 2000. Available at http://www.lcc.uma.es/∼av/Publicaciones/00/Interoperability.pdf.

[Weg96] Peter Wegner. Interoperability. *ACM computing surveys*, (28):285–287, 1996.

[Wel02] Scott Weller. Online article: Web services qualification, 2002. Available at http://www-106.ibm.com/developerworks/webservices/library/ws-qual/?dwzone=webservices.

[YS97] D. M. Yellin and R.E. Strom. Protocol specifications and component adaptors. *ACM Trans. Prog. Lang. Syst.*, (19):292–333, 1997.