Service Interoperabilität auf Kontextebene

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

[1] German Aerospace Center (DLR)
Institut für Kommunikation und Navigation
Oberpfaffenhofen, Germany
thomas.strang@dlr.de

[2] Ludwig-Maximilians-Universität (LMU)
Institut für Informatik
München, Germany
linnhoff@informatik.uni-muenchen.de

Abstract: In Verteilten System ist Interoperabilität eine Grundvoraussetzung für jede Interaktion zwischen verteilten Komponenten. Dieses Paper führt durch die unterschiedlichen Ebenen, auf 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]. Durch Anpassung der Unterschiede der beteiligten Komponenten bezüglich der Hardware, des Netzwerks oder der Software in der Middleware können Anwender und Applikationen mit einem Verteilten System in konsistenter und einheitlicher Weise interagieren.

Jede Kooperation erfordert von den beteiligten Partnern ein "gemeinsames Verständnis", der sogenannten *Interoperabilität*, über Form und Nutzen der ausgetauschten Daten und erbrachten Dienste durch Anwendung einer gemeinsamen Spezifikation. 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. Trotz einer Vielzahl von Untersuchungen zur Interoperabilität sind die vorgeschlagenen Ansätze zur Lösung der dabei auftretenden Probleme 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 (Abschnitt 4). 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.

Die erste Schritte auf der Interoperabilitäts-Evolutionsleiter wurden mit Einführung des *Remote Procedure Calls (RPC)* zur sogenannten *Plattform Interoperabilität*, sowie mit CORBA's *Interface Definition Language (IDL)* zur *Programmiersprachen Interoperabilität* erklommen.

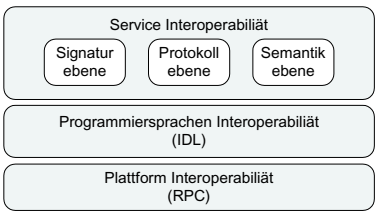

Abbildung 1: Klassische Ebenen der Interoperabilität

In den neunziger Jahren wurden die Untersuchungen zum Thema Interoperabilität weiter 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, VHT00b] (siehe 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)* 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 [VHT00a] vorgenommen.

- Auf der Semantikebene wird versucht, dem Problem des unterschiedlichen Verständnisses entgegenzutreten, da Informationen über die Semantik einer Komponente durch die Beschreibung ihrer Interfaces nicht erfaßt werden. 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 entworfen wurden, sind möglicherweise geeignet, einen Teil der Anforderungen in dieser Ebene zu erfüllen (siehe *www.semanticweb.org*).

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 [VHT00a]. 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 Abbildung 2).

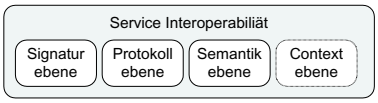

Abbildung 2: Erweiterte Ebene der Service Interoperabilität

Da die Begiffe Kontext und Situation in aktuellen Veröffentlichungen teilweise sehr un-

terschiedlich 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] 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.

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 [VHT00a]. Wir arbeiten daher z.Zt. an einem Verfahren, das es erlaubt, zur Laufzeit Kontext Informationen mit beliebigen Objekten dynamisch zu assoziieren.

```
<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

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

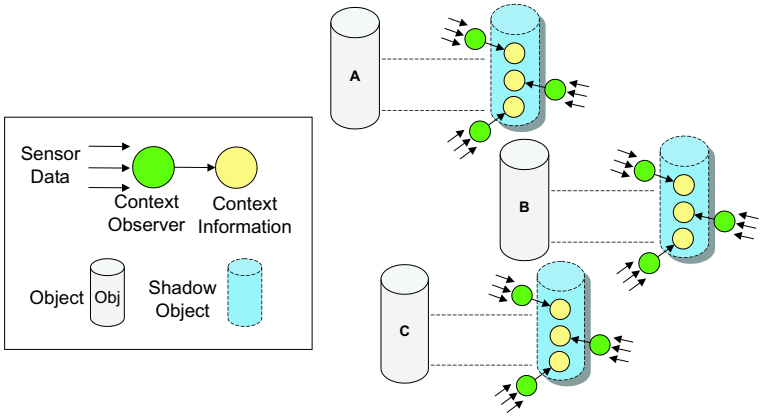

Abbildung 4: Assoziation von Kontext Informationen über Shadow Objekte

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].

4 Ubiquitous Computing und Kontextuelle Service Interoperabilität

Die Trennung zwischen Kontext Informationen und ihrer Nutzung im Dienstumfeld im Sinne des *Observer Patterns* [GHJV02] erleichtert die Integration der Ubiquitous Computing Systeme [Sat01], 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.

Die thematische Fokussierung der Untersuchung kontextueller Abhängigkeiten und Möglichkeiten von Diensten auf einer eigenen Ebene, dem *context level*, erleichtert die Separierung von Diensten, die zwar bezüglich der drei anderen Ebenen interoperabel sind, auf kontextueller Ebene jedoch nicht. Ein 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.

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 Bedeutung 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.

5 Zusammenfassung und Ausblick

Wir haben in den vorhergehenden Abschnitten gezeigt, daß durch die Einführung einer kontextuellen Ebene Dienste identifiziert werden können, 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

[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.

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

[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.

[LP96] Claudia Linnhoff-Popien. *Verteilte Systeme*. 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.

[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.

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

[Sat01] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, pages 10–17, August 2001.

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

[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.

[VHT00a] 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.

[VHT00b] 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, 2000.

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

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