September 2003

Universal Component Trading - Trading in heterogenen Komponentenumgebungen

Stefan Eicker
Universität Duisburg-Essen, eicker@wi-inf.uni-essen.de

Holger Schwichtenberg
Universität Duisburg-Essen

In: Uhr, Wolfgang, Esswein, Werner & Schoop, Eric (Hg.) 2003. *Wirtschaftsinformatik 2003: Medien - Märkte - Mobilität*, 2 Bde. Heidelberg: Physica-Verlag

ISBN: 3-7908-0111-9 (Band 1)
ISBN: 3-7908-0116-X (Band 2)

Universal Component Trading – Trading in heterogenen Komponentenumgebungen

Stefan Eicker, Holger Schwichtenberg
Universität Duisburg-Essen

Zusammenfassung: Auf den unteren Ebenen des ISO/OSI-Protokoll-Stacks existieren zahlreiche Ansätze zur Lokalisierung von Diensten; die Ansätze finden auch verbreitet Anwendung in modernen Netzen. Trading, d.h. eine Lokalisierung von Diensten auf der Anwendungsebene, besitzt dagegen bisher kaum praktische Relevanz. Dies ist nicht zuletzt darauf zurückzuführen, dass der entsprechende ODP-Trading-Standard und alle darauf aufbauenden Standards und Implementierungen gravierende Schwächen aufweisen; zu nennen sind u.a. die fehlende Unterstützung von Softwarekomponenten, die fehlende semantische Beschreibung von Diensten sowie Defizite bezüglich der Architektur des Traders. Im vorliegenden Beitrag wird mit dem Universal Component Trading (UComT) ein Konzept zur Vermittlung von Softwarekomponenten vorgestellt, die aus verschiedenen Komponentenmodellen stammen können. Im Gegensatz zum ODP-Trading-Standard unterstützt das UComT-Konzept auch unterschiedliche Handelseinheiten, unterschiedliche Beschreibungs- und Suchsprachen sowie unterschiedliche Zugriffsprotokolle innerhalb eines Traders. Dies wurde insbesondere durch eine modulare Trader-Architektur und durch Einsatz der Standard-Beschreibungssprache XML erreicht.

Schlüsselworte: Dynamik von Netzwerken, Integration heterogener Systeme, Components, Trading, XML

1 Einleitung

Mit zunehmender Bedeutung wieder verwendbarer Komponenten wird auch die Suche nach geeigneten Komponenten sowie die geregelte Suche und Inanspruchnahme ihrer Methoden als „Dienste“ immer wichtiger. Die Vermittlung von Diensten in verteilten Systemen wird allgemein als *Trading* bezeichnet: Ein Trader speichert Informationen über Dienstanbieter und beantwortet Anfragen von Dienstinteressenten, die dadurch zu Dienstnutzern werden. Somit sind drei Parteien an dem Trading-Vorgang beteiligt: Dienstanbieter, Trader und Dienstnutzer. Der Dienstanbieter wird u.a. auch Exporter, der Dienstnutzer Importer und der Trader Broker oder Service Location Service genannt.

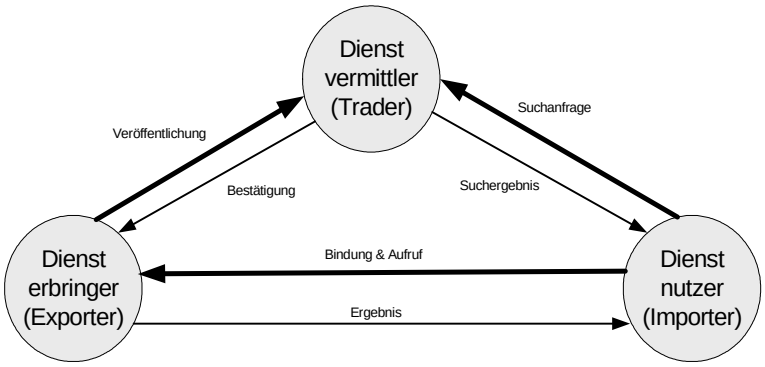

Abbildung 1: Grundmodell des Trading

Die Begriffe Importer, Exporter und Trader stellen grundsätzlich Rollen eines Objekts dar, die sich nicht gegenseitig ausschließen: Ein Exporter kann auch Dienste importieren und ein Importer gleichzeitig Exporter sein, ein Trader kann als Importer und/oder Exporter auftreten.

Auf den unteren Ebenen des ISO/OSI-Stacks haben sich einige der zahlreichen Ansätze zur Lokalisierung von Diensten in der Praxis durchsetzen können; Beispiele sind das Universal Plug and Play (UPnP) der Firma Microsoft und das bei Bluetooth verwendete Service Discovery Protocol (SDP). Trading für anwendungsorientierte Dienste ist dagegen kaum verbreitet, und dies, obwohl bereits in den 90er Jahren im Rahmen des Open Distributed Processing (ODP; ISO/IEC 13235 bzw. ITU X.950) ein entsprechender Standard entwickelt wurde. In der Literatur herrscht weitgehend Einigkeit darüber, dass angesichts der Vielzahl sowohl möglicher Dienste als auch Diensterbringer und analog zu anderen Märkten wirklich offene verteilte Anwendungssysteme in offenen Systemumgebungen nur auf der Basis eines effektiven Trading entstehen können [vgl. z.B. Mue+95, S. 476].

Im Folgenden wird zunächst ein Überblick über den State of the Art im Bereich des Tradings gegeben. Auf Basis einer kritischen Betrachtung der bestehenden Ansätze wird dann das Universal Component Trading (UComT) als neuer Ansatz vorgestellt, der versucht, die modernen Anforderungen an das Trading zu erfüllen.

2 State of the Art des Tradings

Mitte der 90er Jahre gehörte das Trading zu einem Schwerpunkt der Forschungsaktivitäten im Bereich der verteilten Systeme. Davon zeugen in dieser Zeit entstandene Ansätze wie der ANSAWare-Trader [ANSA98] im Rahmen der Advanced Network System Architecture (ANSA), der ODP-Trader-Standard [ISO98] im Reference Model of Open Distributed Processing (RM-ODP) [ISO94]

und der CORBA-Dienst COS-Trading [OMG00]. Zahlreiche Implementierungen der Trader entstanden, und kontinuierlich wurden Erweiterungsvorschläge publiziert. Dann geriet das Trading jedoch weitgehend in Vergessenheit; seitdem dokumentieren nur noch vereinzelt Veröffentlichungen die wenigen übrig gebliebenen Forschungsaktivitäten.

2.1 ODP-Trading

Das bekannteste Trading-Konzept wurde im Rahmen des Reference Model of Open Distributed Processing (RM-ODP) entwickelt. Trading wird im ODP-Standard von 1995 bereits als Teil der Architektur genannt, wurde aber erst 1998 in einem eigenen Standard (ISO/IEC 13235 bzw. ITU X950) umgesetzt, der als ODP-Trading sowie auch als ISO-Trading bezeichnet wird.

Das ODP-Trading stellt eine plattform- und implementierungsunabhängige Spezifikation dar (vgl. dazu und zum Folgenden [Bear97, S. 5ff.]). Es unterscheidet Dienste und Diensttypen: Ein Dienst ist eine Instanz eines Diensttyps (Service Type) mit einem eindeutigen Service Offer Identifier. Diensttypen können in einer Hierarchie angeordnet werden, die eine Untertyp-Beziehung beschreibt. Ein Dienst kann Diensteigenschaften (Service Properties) besitzen, wobei jede Diensteigenschaft zu einem Diensteigenschaftstyp (Service Property Type) gehört. Neben den Diensteigenschaften sind so genannte Dienstangebotseigenschaften (Service Offer Properties) definiert; dabei handelt es sich um Eigenschaften, die sich nicht auf den Dienst, sondern auf das Dienstangebot beziehen. Ein Beispiel wäre der Gültigkeitszeitraum eines Dienstangebots.

Das ODP-Trading definiert für einen Trader verschiedene Schnittstellen jeweils mit einem Satz von Funktionen. Ein Trader muss nicht alle Schnittstellen anbieten; der Standard definiert vielmehr abhängig von den unterstützten Schnittstellen verschiedene Klassen von Tradern, u.a. Query Trader und Stand Alone Trader. Die Zusammenarbeit von Tradern sieht der Standard über das so genannte Interworking vor.

2.2 Implementierungen des ODP-Trading

Der im Rahmen des britischen Forschungsprojekts Advanced Network System Architecture (ANSA) entwickelte *ANSAWare-Trader* hatte maßgeblichen Einfluss auf den ODP-Standardisierungsprozess und ist deshalb als Vorläufer des ODP-Traders einzustufen [Desc93, S. 2]. Dienste werden beim ANSA-Trading - wie beim ODP-Trading - durch Diensttypen und Diensteigenschaften in Form von Attribut-Wert-Paaren beschrieben. Eine Unterscheidung in Diensteigenschaften und Dienstangebotseigenschaften wird nicht vorgenommen; dynamische Diensteigenschaften werden nicht unterstützt.

Go-Between ist ein Trader, der in den Jahren 1990 und 1991 an der University of Queensland (Australien) ebenfalls parallel zum ISO-Standard entwickelt wurde [Raym91]. Gleiches gilt für den *DRYAD-Trader* der Universität Helsinki [Kutv94]. Eine Implementierung einer frühen Version des ODP-Trading lieferte das Projekt "Management Environment for Large Open Distributed Systems" (MELODY) der Universität Stuttgart [KoBu95]; der *MELODY-Trader* handelt mit DCE- und CORBA-Objekten. Implementierungen für DCE werden in [BeBe94] und [Mue+95] präsentiert. Auch die Realisierung des ODP-Trading auf Basis des Verzeichnisdienstes X.500 wurde von verschiedenen Forschern eruiert (vgl. [PoMe93], [War+94], [WaBe95], [RiHo95]).

Auf Basis der Kritik am monolithischen Aufbau des ODP-Traders wurde eine objektorientierte Implementierung des ODP-Trading auf Basis von CORBA unter Nutzung bestehender CORBA-Dienste (insbesondere dem Property Service) vorgeschlagen [Bea+97, S. 17f.]. Die Spezifikation orientiert sich an dem Spezifikationsstil der CORBA Object Services (COS), ist aber nicht zu verwechseln mit dem CORBA Trading Object Service (COS-Trading), dem Trading-Dienst der Verteilungsplattform CORBA [OMG00, S. 1]. Dieser basiert auf dem ODP-Trading, realisiert aber nicht alle vorgesehenen Funktionen des ODP-Trading sowie auch z.T. andere Operationen in den Schnittstellen des Application Programming Interface (API).

Verfügbar sind sowohl kommerzielle als auch wissenschaftliche Implementierungen und Weiterentwicklungen des COS-Trading. Kommerzielle Systeme bietet vor allem die Firma IONA an; dazu zählen der im Rahmen der CORBA-Implementierung Orbix 2000 ausgelieferte COS-Trader, der Trader des Produkts ORBacus sowie CORBazar als eine partielle Implementierung des COS-Trading in Java für OrbixWeb 3.0 [Pete98]. Frei verfügbarer COS-Trader sind die entsprechende Java-Implementierung im JacORB [Bros02], die Trading Object Service Implementation (TOI) der GMD (heute: Fraunhofer Institute for Open Communication Systems) sowie der TAO Trading Service [Wido02].

2.3 Kritik am ODP-Trading und Erweiterungsvorschläge

Die wesentliche Kritik am ODP-Trading besteht aus heutiger Sicht v.a.D. darin, dass es auf den Handel mit Objekten festgelegt ist und nicht den Handel mit Komponenten als größere Einheiten unterstützt. Zudem ist das ODP-Trading auf homogene Objektmodelle beschränkt. Dem Einsatz in modernen Umgebungen sind schließlich auch dadurch erhebliche Grenzen gesetzt, dass die Dienstbeschreibung durch Diensttypen und Attribut-Wert-Paare zu einfach ist und zu wenig Semantik über die Funktionalität des Dienstes enthält (vgl. auch [Pude95, S. 3] und [TeNi99, S. 187]).

In der Literatur werden weitere Kritikpunkte genannt:

- Die Komposition mehrerer Dienste zur Befriedigung einer Suchanfrage im ODP-Trading ist nicht vorgesehen [Vasu98, S. 2].
- Der ODP-Trader übernimmt keine Garantie für das Funktionieren eines vermittelten Dienstes [JaMu96, S. 3].
- Die monolithische Architektur des Traders, die für die Berücksichtigung der Unterstützung unterschiedlicher Plattformen gewählt wurde, ist zu inflexibel für sich ändernde/weiterentwickelnde Anforderungen [Bea+97, S. 16f.].
- Realisierungen sind hochkomplex: Auf Seiten des Traders bezieht sich diese Kritik insbesondere auf die Komplexität der Schnittstellen ([Bea+97, S. 16], auf Seiten des Clients dagegen auf die Komplexität der Suchanfragen [Mar+01, S. 4f.]. Auch das Interworking wird als zu schwergewichtig bewertet [VaBa98, S. 3].
- Es fehlt eine Typprüfung zur Entwicklungszeit; die wichtige Fragestellung, wie die Programmierer zur Übersetzungszeit ihrer Applikationen Kenntnis über Datentypen erlangen, bleibt dadurch unbeantwortet [Pude95, S. 1].

3 Anforderungen an ein Universal Component Trading und grundlegende Konzepte

Die wesentliche Kritik am ODP-Trading-Ansatz, die fehlende Berücksichtigung einerseits heterogener Umgebungen und andererseits von Komponentenmodellen, fordert die Entwicklung eines entsprechend erweiterten Ansatzes (vgl. auch [Iri+01a, S. 31] und [Iri+01, S. 124]). Dazu sind neben den Konzepten des ODP-Tradings auch die in der Literatur vorgeschlagenen Verbesserungen daraufhin zu untersuchen, inwieweit sie für den erweiterten Ansatz von Bedeutung sind.

Die Adaption der Trading-Idee auf Komponentenmodelle wurde bereits in einigen wenigen Arbeiten diskutiert ([TeNi99], [KeBe01], [Iri+01a], [Mar+01]); in den Arbeiten werden allerdings Trader für ein homogenenes Komponentenmodell, vorzugsweise das CORBA Component Model, spezifiziert bzw. implementiert. Von den verfügbaren Tradern unterstützt nur der MELODY-Trader zwei Modelle (DCE und CORBA). Die Kopplung von Softwarekomponenten aus unterschiedlichen Komponentenmodellen ist bisher nur über so genannte Bridges [Gut+99, S. 29ff.] oder auf Basis eines übergreifenden Zugriffsprotokolls wie dem Simple Object Access Protocol (SOAP) [Gud+02] möglich. Wir haben deshalb den Ansatz *Universal Component Trading (UComT)* für die Anwendung des Trading-Konzepts auf objektorientierte Komponentenmodelle entwickelt; seine Trading-Architektur ist in der Lage, mit beliebigen Komponentenmodellen sowohl als Dienstanbieter als auch als Dienstnutzer zusammenzuarbeiten. Im Folgenden zeigen wir Anforderungen an einen UComT-Trader und entsprechende Konzepte auf.

3.1 Trading von heterogenen Komponenten

In der Praxis haben sich vier Komponentenmodelle etabliert, das Component Object Model (COM) der Firma Microsoft, JavaBeans/Enterprise Java Beans (EJB) der Firma Sun, das CORBA Component Model (CCM) der Object Management Group sowie die Common Language Infrastructure (CLI), die von der EMCA und der ISO standardisiert wurde. Diese Modelle müssen wegen ihrer Bedeutung einbezogen werden.

Bezüglich des Tradings mit Komponenten aus verschiedenen Komponentenmodellen sind drei Herausforderungen zu berücksichtigen: Erstens unterstützen die Komponentenmodelle unterschiedliche Baupläne für Softwarekomponenten, d.h., die Meta-Modelle der Komponenten unterscheiden sich. Zweitens verwenden die Komponentenmodelle unterschiedliche Beschreibungen der durch die Komponenten angebotenen Operationen. Drittens unterscheiden sich auch die Protokolle für den Zugriff auf Komponenten.

Aus den drei Herausforderungen ergeben sich unmittelbar sechs elementare Eigenschaften für das UComT und für UComT-Trader:

1. Trader unterstützen unterschiedliche Baupläne von Komponenten.
2. Vergleichbare Elemente in verschiedenen Bauplänen können in Beziehung gesetzt werden.
3. Trader unterstützen unterschiedliche Arten von Dienstbeschreibungen.
4. Trader unterstützen unterschiedliche Zugriffsprotokolle auf Komponenten.
5. Trader verwalten eine Liste erreichbarer Bridges zur Kopplung heterogener Komponentenmodelle.
6. Die Kommunikation zwischen Trader und Trader-Client wird auf Basis eines standardisierten Zugriffsprotokolls realisiert.

3.2 Handelseinheiten

Kovács zeigt in [Kova96, S. 5] den Bedarf nach dem Trading kombinierter Dienste (Complex Services) auf, die aus mehreren Einzeldiensten bestehen. Im Rahmen des Semantically Enhanced Component Trading wurde auch bereits das Konzept eines Dienstes als eine Einheit von mehreren funktional zusammenhängenden Komponenten vorgestellt [TeNi99, S. 187]. In Rahmen des UComT-Ansatzes gilt es, die genannten Ansätze zu präzisieren und zu erweitern; daraus ergibt sich die grundlegende Anforderung an das Universal Component Trading, dass nicht nur das Handeln mit ganzen Komponenten, sondern auch mit Teilen von Komponenten und mit zusammengesetzten Komponenten zu ermöglichen ist. UComT defi-

niert deshalb als mögliche Handelseinheit eines Traders Komponenten, einzelne Bausteine von Komponenten und Zusammensetzungen von Komponenten.

3.3 Dienstbeschreibung und Dienstangebotsdetails

Für die Beschreibung von in Tradern zu verwaltenden Diensten sind zahlreiche Alternativen bekannt: ANSA-Trading, ODP-Trading und COS-Trading sehen eine Beschreibung auf Basis von Diensttypen und Diensteigenschaften in Form von Name-Wert-Paaren vor. Die in vielen Verteilungsplattformen und Komponentenmodellen (z.B. ODP, COM, CORBA) verwendete Interface Definition Language (IDL) ist zwar syntaktisch ausdrucksstärker, bietet aber ebenfalls keine Möglichkeit, die Semantik von Schnittstellen und Operationen zu spezifizieren. Zu den Ansätzen mit einer gewissen semantischen Ausdrucksstärke zählen die Kind Description Language [Kini99], die wissensbasierte Dienstvermittlung auf Basis von Konzeptgraphen [Pud+95a, Pude95], Coloured Petri Nets [Mue+95, S. 484], Ontologien ([TeNi99, S. 194] und [TeNi00, S. 10f.]) und Agenten-Sprachen wie das Knowledge Interchange Format (KIF) und die Knowledge Query Manipulation Language (KQML) [TeNi99, S. 192].

Alle bisherigen Erfahrungen in Bezug auf Beschreibungssprachen zeigen, dass sich die Beschränkung/Festlegung auf eine einzige Sprache im Allgemeinen nicht durchsetzen lässt. Ein UComT-Trader muss deshalb die Dienstbeschreibung durch „beliebige“ Sprachen unterstützen und dies auch zur gleichen Zeit. Um die Suche nach Diensten nicht durch die Verschiedenartigkeit der Dienstbeschreibung einzuschränken, sind zwei Lösungswege denkbar: Einerseits können Konverter zwischen verschiedenen Dienstbeschreibungstypen im Trader vermitteln; dazu kann auf dem in der Literatur vorgeschlagenen Type Identifier Transformation Protocol [Vog+95] aufgesetzt werden. Andererseits besteht – im Gegensatz zu allen bisherigen Trader-Konzepten – die Möglichkeit, den UComT-Trader in die Lage zu versetzen, Dienstangebote mit Dienstbeschreibungen in verschiedenen Dienstbeschreibungssprachen zu verwalten. Darüber hinaus ist mit Blick auf die riesige Menge potenzieller Dienstangebote eine Klassifizierung der Angebote vorzusehen. Eine mögliche Klassifizierung stellt insbesondere die Klassifizierung nach dem Dienstanbieter dar.

Entsprechend dem Konzept der Dienstangebotseigenschaften im ODP-Trading muss auch das UComT Zusatzinformationen unterstützen, die nicht Teil des Dienstes sind, sondern vom Exporter im Rahmen der Komponentenregistrierung an den Trader übermittelt werden. Es bietet sich an, diesbezüglich das erfolgreiche Konzept des ODP-Trading zu übernehmen, und Dienstangebotsdetails durch Attribut-Wert-Paare abzubilden. Im Gegensatz zu den Dienstangebotseigenschaften des ODP-Trading, die nur einfache Datentypen erlauben, sollen UComT-Dienstangebotsdetails für eine semantisch reichere Darstellung allerdings auch mehrwertige Attribute und komplexe Datentypen vorsehen.

Dienstangebotsdetails können sich auf technische Eigenschaften (z.B. Komponentenmodell, Liste der unterstützten Zugriffsprotokolle, Versionsnummer, Plattformvoraussetzungen) oder auf nicht-technische Informationen (z.B. Produktname, Urheber, Kosten, Dokumentation) beziehen. Welche Dienstangebotsdetails über alle denkbaren Dienste hinweg relevant sind, kann auf keinen Fall vorab festgelegt werden. Ein UComT-Trader muss deshalb – insbesondere auch zur Laufzeit – um die Verarbeitung zusätzlicher Dienstangebotsdetails erweitert werden können. Da eine solche Erweiterung abhängig von den von einem Exporter oder einem Importer spezifizierten Details ist, bietet es sich an vorzusehen, dass Exporter dem Trader bei der Komponentenregistrierung und Importer im Rahmen der Dienstsuche entsprechende Erweiterungsmodule übermitteln.

3.4 Dynamische Eigenschaften

Als dynamisch – in der Literatur auch als indirekt [WaBe95, S. 1 und 2]) – werden im ODP-Trading-Standard Eigenschaften von Diensten bezeichnet, die nicht beim Trader gespeichert, sondern von diesem erst zum Zeitpunkt einer Suchanfrage beim Exporter erfragt werden. Ein einfaches Beispiel ist die aktuelle Größe der Warteschlange eines Druckdienstes. Alle anderen Eigenschaften werden beim ODP-Trading als statisch bezeichnet. „Statisch“ bedeutet nicht, dass sich die Werte einer Eigenschaft nicht ändern können; eine solche Wertänderung kann vielmehr über eine entsprechende Aktualisierungsnachricht durch den Exporter des Dienstes initiiert werden.

Die sich verändernden Rahmenbedingungen eines Dienstes stellen für einen Importer vielfach wichtige Entscheidungsparameter für die Inanspruchnahme des Dienstes dar. Das UComT muss deshalb entsprechende Aktualisierungsprozesse in möglichst großer Vielfalt vorsehen; neben den vom ODP-Trading vorgesehenen Prozessen zählen dazu auch die periodische Anfrage in von seinem Administrator festgelegten Intervallen seitens des Traders sowie die Übermittlung einer Aktualisierungsnachricht an den Trader durch einen Dritten.

3.5 Suchanfragen und Dienstauswahl

Das ODP-Trading schreibt nicht eine bestimmte Suchsprache vor, sondern erlaubt verschiedene Suchsprachen; allerdings wird für einen Trader eine bestimmte Suchsprache vorgegeben sowie für das Interworking von Tradern die Standard Contraint Language (SCL) [OMG00, S. 9ff. und S. B-1ff.]. Analog zu der Frage der Dienstbeschreibungssprachen kann das UComT auch bezüglich der Suchsprachen keine Einschränkungen machen, sondern muss vielmehr – wie in der Literatur gefordert [Bea+97, S. 21] – alle in diesem Bereich relevanten Sprachen unterstützen. Zu solchen Sprachen zählen die Standard Constraint Language (SCL)

und die Matching Constraint Language (MCL) der OMG, SQL und die XML Query Language (XQL).

Bezüglich der Suchanfragen kann auf dem Konzept des Semantically Enhanced Traders [TeNi00, S. 12] aufgesetzt werden; es unterscheidet insbesondere eine unterschiedliche Suchgenauigkeit einerseits in einer Laufzeit- und andererseits in einer Entwicklungszeit-Schnittstelle. Zu unterstützen sind zudem Suchmodi, die sich hinsichtlich der Präzision der Suche, der Typsicherheit der Ergebnisse und der Beteiligung föderierter Trader unterscheiden.

Die Dienstauswahl beinhaltet die Anwendung einer gegebenen Suchanfrage auf die Menge der im Trader verfügbaren Dienstangebote. Dabei wird die Übereinstimmung der Suchanfrage mit den Dienstangeboten geprüft und auf dieser Basis eine Auswahl getroffen. Dieser Vorgang wird in der Literatur als „Matching“ (z.B. [Kova96, S. 5]) bezeichnet.

Beim ODP-Trading-Standard wird eine feste Vorgabe bezüglich des Matching-Algorithmus vorgegeben. Diese Vorgehensweise widerspricht jedoch der von einem UComT-Trader zu fordernden Flexibilität. Denn unterschiedliche Matching-Algorithmen sind beim UComT insbesondere notwendig, da verschiedene Arten von Dienstbeschreibungen im Dienstangebot des Traders sowie verschiedene Sprachen zur Formulierung von Suchanfragen existieren können. Der UComT-Trader muss deshalb über mehrere alternative Matching-Algorithmen verfügen.

Zu unterscheiden ist das Trading zur Entwicklungszeit (Design/Development Time Trading) und das Trading zur Laufzeit (Runtime Time Trading) [Pud+95a, S. 62]. Beim Design Time Trading ist immer ein Mensch beteiligt, der über eine entsprechende Benutzerschnittstelle Einfluss auf den Trading-Prozess nehmen können muss. Auch beim Runtime Trading ist die Möglichkeit einer Benutzerinteraktion vorzusehen; dies kann insbesondere dann sinnvoll sein, wenn auf eine Suchanfrage hin entweder kein Suchergebnis oder aber mehrere identische Ergebnisse gefunden werden.

3.6 Verfügbarkeitsprüfung

Beim ODP-Trading wird – wie von Jacob und Mudge kritisiert [JaMu96, S. 3] – nicht überprüft, ob ein registrierter Dienst, der einem Importer als Suchergebnis präsentiert wird, funktionsbereit ist. Sicherstellen kann ein Trader diese Funktionsbereitschaft letztlich nicht, da sich in der Zeit zwischen einer entsprechenden Überprüfung durch den Trader und der Inanspruchnahme eines Dienstes durch den Importer die Sachlage ändern kann. Jedoch ergibt sich über alle Versuche von Dienstinanspruchnahmen hinweg eine deutliche Effizienzverbesserung, wenn der Trader periodisch alle von ihm angebotenen Dienste überprüft oder auf Anforderung – d.h. optional – bei einer Suchanfrage eine Überprüfung der gefundenen Dienste vornimmt.

Bezüglich der „Tiefe“ der Überprüfung kann in beiden Fällen zwischen einer seichten Prüfung, einer mittleren Prüfung und einer tiefen Prüfung unterscheiden werden: Bei der seichten Prüfung testet der Trader nur, ob das Rechnersystem, auf dem die Komponente läuft, erreichbar ist. Bei einer mittleren Prüfung versucht der Trader, eine Referenz auf ein Objekt in der Komponente zu erhalten. Bei einer tiefen Prüfung ruft der Trader zum Test eine Methode eines Objekts in einer instanziierten Komponente auf.

3.7 Daueraufträge

Neben der sporadischen Inanspruchnahme eines Dienstes durch einen Importer sind mannigfaltige Szenarien realistisch, in denen ein Importer kontinuierlich einen Dienst in Anspruch nimmt. Bezogen auf das System „Importer-Trader“ wäre es in solchen Szenarien ineffektiv, wenn der Importer kontinuierlich identische Suchanfragen stellen müsste. Deshalb sollte der Importer dem UComT-Trader auch einen Dauerauftrag erteilen und damit sein kontinuierliches Interesse an einem bestimmten Dienst anmelden können. Ein solcher Dauerauftrag sollte dazu führen, dass der Trader in regelmäßigen (definierbaren) Intervallen von sich aus die Suchanfrage selbstständig durchführt und den Importer darüber informiert, wenn sich bei dem Ergebnis der Suchanfrage Änderungen ergeben.

3.8 Dienstaufruf

Für die Nutzung eines ausgewählten Dienstes sind vier Szenarien denkbar:

1. Der Importer erhält vom Trader eine Referenz auf die Komponente (bestehend aus Servernamen und eindeutiger Komponentenidentifizierung sowie der Liste der möglichen Aufrufprotokolle). Der Importer ruft die Komponente selbst ohne Beteiligung des Traders auf. Diese Möglichkeit wird beim ODP-Trading genutzt.
2. Der Importer überträgt dem Trader die Aufgabe, die Komponente aufzurufen (z.B. weil der Importer nicht über das passenden Zugriffsverfahren verfügt). Der Trader ruft dann die Komponente mit den Parametern des Importers auf, und liefert dem Importer das Ergebnis zurück. In diesem Fall fungiert der Trader als Proxy für den Aufruf der Komponente.
3. Die Komponente wird bei der Registrierung zum Trader übertragen und auf Anfrage eines Importers vom/im Trader ausgeführt.
4. Der Importer erhält vom Trader eine zuvor von dem Exporter an den Trader übermittelte Komponente und kann diese lokal bei sich ausführen. Dies entspricht dem Konzept vom mobilen Code (siehe z.B. [Lee96]).

Ein universelles Trading muss alle vorgestellten Szenarien unterstützen; damit der Trader als Proxy für den Zugriff auf Komponenten genutzt werden kann.

3.9 Zugangsbeschränkungen

Beim ODP-Trading sind Zugangsbeschränkungen zwar nicht explizit definiert, aber im Rahmen von implementierungsabhängigen Richtlinien in einen Trader integrierbar. In modernen Umgebungen ist ein Zugriffsschutz unerlässlich, weshalb der UComT-Trader Zugangsbeschränkungen explizit unterstützen muss. Zu unterscheiden sind Zugangsbeschränkungen zum Trader und Zugangsbeschränkungen zu Diensten: Eine Zugangsbeschränkung zum Trader ist nicht nur hinsichtlich der administrativen Funktionen des Traders, sondern auch bezüglich des Dienstimports und Dienstexports sinnvoll; denn andernfalls besteht die Gefahr, dass von Dienstanbietern die Registrierung konkurrierender Dienste aufgehoben wird oder Angreifer einen Dienst registrieren, der ihre Attacken unterstützt.

Erforderlich ist auch eine Zugangssteuerung zu den vom Trader angebotenen Diensten. Die in den Komponentenmodellen definierte Infrastruktur erlaubt in der Regel eine solche Zugangssteuerung für Komponenten; ein Beispiel ist die Code Access Security in der CLI [ECMA02]. Die Zugangssteuerung der Komponentenmodelle sollte auch im Trader abbildbar sein, um zu vermeiden, dass Clients als Suchergebnisse Komponenten erhalten, die sie aufgrund mangelnder Zugangsberechtigungen nicht nutzen können. Dies gilt allerdings in erster Linie für das Runtime-Trading; beim Design-Time-Trading können Suchergebnisse mit Komponenten, für die der Importer keine ausreichenden Rechte besitzt, tolerierbar bzw. sogar wünschenswert sein.

Bei den Zugangsbeschränkungen zu den Handelseinheiten sind die Sichtbarkeit/Auffindbarkeit und die Nutzung zu unterscheiden: „Sichtbarkeit“ beinhaltet, dass Handelseinheiten steuern können, welchen Importern der Trader sie im Suchergebnis anzeigen soll. Wir sprechen von der Sichtbarkeitsberechtigung bzw. dem Sichtbarkeitsschutz; über sie/ihn ist ein besonders hoher Sicherheitsgrad für sensible Bereiche erzielbar, da Dienste und die Form ihres Aufrufs potenziellen Angreifern verborgen bleibt.

Die Aufrufberechtigung beinhaltet das Recht, einen Dienst aufzurufen und damit zu nutzen. Importer können das (berechtigte) Interesse haben, auch über Dienste informiert zu werden, für die sie derzeit keine Zugriffsrechte besitzen. Sie können dann auf Basis der zur Verfügung gestellten Dienstbeschreibung erwägen, eine Zugangsberechtigung zu einem Dienst zu beantragen bzw. dem Nutzer des Importer-Systems eine entsprechende Autorisierung zu empfehlen. Entsprechend stellt die Aufrufberechtigung ein sinnvolles Selektionskriterium für die von Importern formulierten Suchanfragen dar.

3.10 Abrechnung der Dienstleistungen

Bei der Abrechnung der Leistungen des Traders sind verschiedene Handelszenarien zu unterscheiden. Denkbar ist, dass sowohl Importer als auch Exporter den Betreiber des Traders für seine Leistungen bezahlen, oder, dass der Importer den Exporter bezahlt und der Betreiber des Traders für eine Vermittlungstätigkeit eine Provision von diesen Einnahmen erhält. Zur Leistungsabrechnung benötigt ein Trader eine Schnittstelle sowohl zum Exporter als auch zum Importer. Beispiele für Exportern u.U. in Rechnung zu stellende Leistungen sind die Registrierung eines Dienstes, das (laufende) Anbieten des Dienstes, die Bereitstellung von Speicher für die Speicherung der Komponente im/auf dem Trader, die Bereitstellung des Stellvertreteraufrufs, die Übermittlung der Komponente in der Ergebnismenge einer Suchanfrage und die Übertragung der Komponente an einen Client. Dem Importer müssen insbesondere die Durchführung von Suchanfragen – eventuell in Abhängigkeit von der Anzahl der Handelsobjekte in der Ergebnismenge – und Stellvertreteraufrufe eines Handelsobjekts berechnet werden können.

Neben seinen eigenen Leistungen sollte der Trader einem Importer auch die Inanspruchnahme von Diensten in Rechnung stellen können. Dazu muss allerdings ein Exporter die Inanspruchnahme jeweils melden, sofern nicht der Trader als Stellvertreter selbst den Dienst genutzt hat.

3.11 Erweiterbarkeit

Bei der bisherigen Diskussion der Anforderungen an einen UComT-Trader ist deutlich geworden, dass der Trader nicht auf bestimmte Charakteristika z.B. bezüglich der unterstützten Komponentenmodelle, Beschreibungssprachen, Selektionsverfahren und Zugriffsprotokolle eingeschränkt werden kann. Eine entsprechende Erweiterbarkeit erfordert eine Modularisierung des Traders einschließlich der Möglichkeit, Module zur Laufzeit hinzuzufügen.

Für die bereits angesprochenen Erweiterungen über Importer und Exporter sowie für die selbständige Erweiterung des Traders bietet sich die Nutzung von URLs an. Diese werden dem Trader dann vom Importer bzw. Exporter übermittelt bzw. im Trader als Orte hinterlegt, an denen für ihn Erweiterungen hinterlegt worden sind.

3.12 Weitere Anforderungen

Es bestehen weitere Anforderungen an den UComT-Trader, die hier aus Platzgründen nicht diskutiert werden können. Dazu zählen insbesondere:

- die Ermittlung eines Suchergebnisses in einem iterativen Suchprozess, wie ihn Puder, Gudermann und Markwitz vorschlagen ([Pud+95a] und [Pude95]).

- die Protokollierung der Aktivitäten des Traders insbesondere zur Fehleranalyse; wegen des entsprechenden Datenvolumens muss der Umfang der Protokollierung durch den Administrator gesteuert werden können.
- die Optimierung der Leistung des Traders durch die Berücksichtigung der Erfahrungen, die Importer mit Diensten machen; dazu müssen entsprechende Bewertungen an den Trader gesendet, dort gespeichert und bei der Bearbeitung von Suchanfragen berücksichtigt werden können.
- Die Bereitstellung adäquater Benutzerschnittstellen durch den Trader; bei ihrer Gestaltung kann auf den von Goodchild erarbeiteten Design-Prinzipien aufgebaut werden [Good95].

4 Architekturmodelle für das Universal Component Trading

4.1 Grundarchitektur

Die Grundarchitektur des Universal Component Trading umfasst zwei Softwarekomponenten (vgl. Abbildung 2): Der UComT-Trader erbringt den Trading-Dienst und entspricht damit dem Trader Server Agent (TSA) ([Kova96, S. 8], [PoMe93, S. 315]). UComT-Clients existieren auf den exportierenden bzw. importierenden Systemen, um die Kommunikation des Exporters bzw. Importers mit dem UComT-Trader zu ermöglichen (entsprechend dem Trader User Agent (TUA) in den genannten Quellen). Zu unterscheiden sind ein Standard-Client, der sowohl die Funktion eines Importers als auch die eines Exporters umfasst, und ein spezieller Admin-Client zur Administration des Traders.

Sowohl der Standard-Client als auch der Admin-Client bieten jeweils eine Benutzerschnittstelle (User Interface – UI) für menschliche Benutzer und eine Programmierschnittstelle (Application Programming Interface – API) zur Nutzung entsprechender Funktionen durch andere Softwaresysteme. Die Benutzerschnittstelle im Standard-Client bietet insbesondere die Möglichkeit, Dienste zu registrieren (Export-Funktion) und Dienste zu suchen (Import-Funktion). Gemäß den diskutierten Anforderungen ist für die Kommunikation zwischen den UComT-Clients und dem UComT-Trader ein Zugriffsprotokoll zu wählen, das auf allen Plattformen verfügbar ist. Diese Anforderungen erfüllen zurzeit nur die XML-Webservices, die den entfernten Prozeduraufruf auf Basis des Simple Object Access Protocols (SOAP) [Gud+02] realisieren.

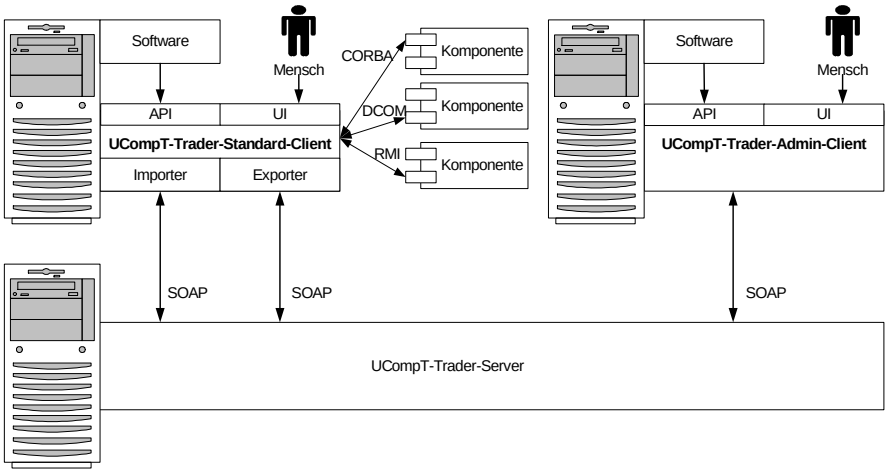

Abbildung 2: UComT-Grundarchitektur

4.2 Aufbau eines UComT-Trader-Servers

Die Architektur des UComT-Trader-Servers ist in Abbildung 3 dargestellt; ihr liegt in Erweiterung des Vorschlags von Bearman et al., Trader objektorientiert zu implementieren [Bea+97], ein komponentenbasiertes Design zugrunde. Um die unterschiedlichen Anforderungen insbesondere hinsichtlich der (dynamischen) Erweiterbarkeit erfüllen zu können, ist zwingend eine modulare Architektur erforderlich. Im Zentrum der Architektur steht der Trader-Kern, der die Basisfunktionen enthält und dazu folgende Softwarekomponenten umfasst:

- Ein Exporter-Manager dient zur Kommunikation mit den Exportern; er nimmt von Exportern Befehle zum Registrieren, zur Änderung der Registrierung, zum Löschen und zum Ändern dynamischer Attribute entgegen.
- Der Service-Locator ist ein Dienst, der selbstständig ein Netzwerk nach geeigneten Komponenten absucht.
- Der Service-Validator überprüft periodisch und bei Bedarf die Verfügbarkeit registrierter Komponenten.
- Aufgabe des Importer-Managers ist die Kommunikation mit den Importern, insbesondere die Entgegennahme und die Beantwortung von Suchanfragen.
- Der Matching-Manager leitet die Suchanfrage an verschiedene Description-Module weiter und nutzt dabei ggf. Konverter- und Linking-Module.

- Der Access-Manager nimmt Anfragen für Proxy-Aufrufe des Clients entgegen und prüft, ob ein passendes Access-Modul vorhanden ist; die Anfrage leitet er über das passende Access-Modul an eine Komponente weiter.

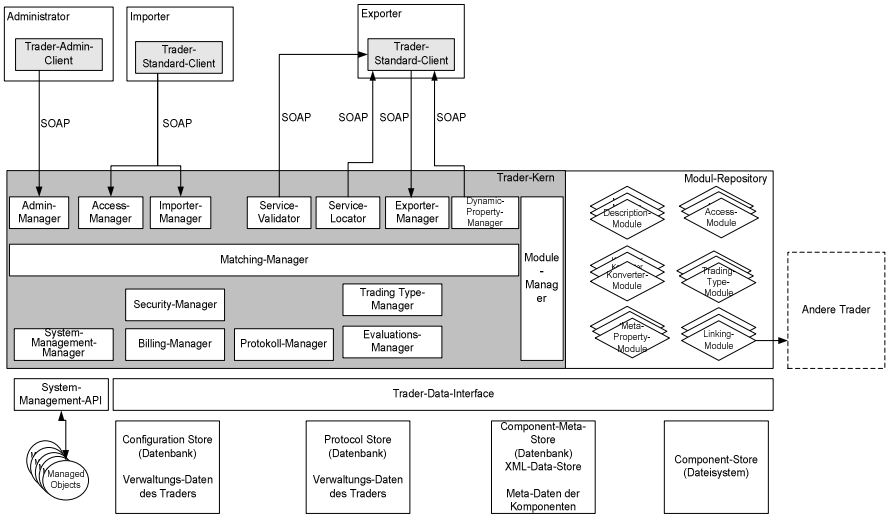

Abbildung 3: Architektur des UComT-Trader-Servers

- Der *Dynamic-Property-Manager* ermittelt dynamische Eigenschaften von einer Komponente, sofern diese als Ad-hoc-Eigenschaft oder als periodisch durch den Trader aktualisierte Eigenschaften gekennzeichnet sind.
- Der *Evaluation-Manager* dient dazu, Bewertungen der angebotenen Komponenten zu erstellen. Er kann Daten von einem Importer oder vom System-Management-Manager entgegennehmen bzw. anfordern.
- Der *System-Management-Manager* dient zum Zugriff auf Systemmanagement-Schnittstellen wie SNMP und WBEM, mit denen Dienste beobachtet werden können.
- Der *Billing-Manager* rechnet Dienstleistungen mit Importern und Exportern auf Basis der hinterlegten Abrechnungskonfiguration und der gespeicherten Abrechnungsdaten ab.
- Der *Administration-Manager* stellt eine Schnittstelle für den UComT-Admin-Client zur Verwaltung des Traders zur Verfügung.
- Aufgabe des *Protocol-Managers* ist die Protokollierung der Aktivitäten des Traders gemäß den Vorgaben seines Administrators. Er speichert Daten im Protocol Store bzw. löscht sie dort nach Ablauf vorgegebener Verfallszeiten.
- Der Security-Manager setzt die Einhaltung der Zugangsbeschränkungen zum Trader und zu den angebotenen Komponenten durch.

Neben dem Trader-Kern sieht die Architektur Erweiterungsmodule in Form von Softwarekomponenten vor, die an definierten Stellen in der Prozesskette des Traders eingreifen. An welchen Stellen ein Modul in die Prozesskette eingreift, wird allgemein vom Trader – in Abhängigkeit vom Modultyp – und im Speziellen durch das Modul festgelegt.

Zur Verwaltung installierter Erweiterungsmodule enthält der Trader-Kern als weiteres Modul einen *Module-Manager*. Er erlaubt die Installation, Konfiguration und Deinstallation von Modulen, und bildet die Schnittstelle zwischen Modulen und Komponenten des Trader-Kerns. Die Schnittstelle ermöglicht den Modulen insbesondere, über die Programmierschnittstellen Einfluss auf den Trader-Prozess zu nehmen. Erweiterungsmodule können entweder bei Bedarf durch den Kern des Traders aktiv aufgerufen werden, oder sich als Konsument von Ereignissen des Trader-Kerns registrieren; dadurch kann ein Aufruf von Funktionen in den Erweiterungsmodulen ohne explizite Kenntnisse über Aufbau von Kern und Modulen erfolgen.

Sechs Arten von Modulen sind zu unterscheiden:

- Ein Description-Modul implementiert die Unterstützung für eine spezielle Dienstbeschreibungstechnik (z.B. ODP-Dienstbeschreibungen, Konzeptgraphen); die Unterstützung umfasst auch die Speicherung der Dienstbeschreibungen sowie die Prüfung auf Übereinstimmung zweier Dienstbeschreibungen (Matching).
- Konverter-Module als optionale Module ermöglichen die Übersetzung von Suchanfragen in andere Dienstbeschreibungssprachen und damit die Erweiterung des Suchraums für eine Suchanfrage.
- Mit einem Linking-Modul können Verbindungen zu anderen Tradern aufgebaut werden, um das Interworking zwischen Tradern zu realisieren.
- Ein Access-Modul realisiert ein bestimmtes Zugriffsprotokoll, um Komponenten eines bestimmten Typs zur Verfügbarkeitsprüfung oder als Stellvertreter aufrufen zu können.
- Meta-Property-Module realisieren eine Implementierung für die bei der Komponentenregistrierung oder bei einer Suchanfrage übermittelte Meta-Daten.
- Ein Trading-Type-Modul ist für jeden Komponentenmodell-Baustein erforderlich, mit dem der Trader handeln soll.

4.3 Datenspeicher und Datenaustauschformat

Der UComT-Trader benötigt vier Datenspeicher; sie dienen zur Speicherung der Meta-Daten der beim Trader registrierten Komponenten (Component Meta Store),

zur Speicherung von Komponenten (Component Store), zur Speicherung erlaubter Handelseinheiten und deren „Enthaltensein-Beziehungen“ (Configuration Store) und schließlich zur Speicherung der vom Protocol Manager erfassten Aktivitäten des Traders (Protocol Store).

Zur Erfüllung der Anforderungen, dass beliebige Beschreibungssprachen eingesetzt und dass die an den Trader übermittelbaren Meta-Daten erweitert werden können, wird ein flexibles Datenaustauschformat benötigt. Die Verwendung von Application Programming Interfaces (APIs) mit definierten Funktionssignaturen bzw. definierten Parameter-Datentypen entspräche diesen Anforderungen nicht. Als Datenformat bietet sich vielmehr – wie in der Literatur bereits vorgeschlagen [SeNa01] – die Extensible Markup Language (XML) an, da mit ihr beliebige Datenstrukturen abgebildet werden können. Ein XML-Dokument ist außerdem beliebig erweiterbar, ohne dass die Integrität der bereits enthaltenen Daten gefährdet wird. Zudem ist XML ein Standard, der auf allen Plattformen verfügbar ist. Da ein XML-Dokument ein Textdokument ist, besteht schließlich auch eine hohe Durchlässigkeit in Netzwerken bzw. durch Sicherheitssysteme wie Firewalls.

Als Trader-Dokument bezeichnen wir ein XML-Dokument, das ein Trader von einem Importer oder einem Exporter erhält. Wichtige Trader-Dokumente sind insbesondere Registrierungsdokumente (zur Registrierung eines Dienstes, vgl. Abbildung 4) und Suchdokumente (zur Suche eines Dienstes).

5 Zusammenfassung und Ausblick

Für das anwendungsorientierte Trading existieren mit dem ODP-Trading und dem – auf dem ODP-Trading basierenden – COS-Trading zwei etablierte Standards mit zahlreichen Referenzimplementierungen. Das ODP-Trading besitzt jedoch grundlegende Schwächen im Bereich der Semantik der Dienstbeschreibung, der Flexibilität sowie der Trader-Architektur. Zudem ist der Standard auf das Trading mit homogenen Objekten ausgerichtet, und deshalb nicht für das Trading mit Softwarekomponenten aus verschiedenen Komponentenmodellen geeignet.

Mit dem Universal Component Trading (UComT) haben wir einen Ansatz für einen Trader vorgeschlagen, der mit Komponenten aus unterschiedlichen modernen Komponentenmodellen handeln kann. Durch die Nutzung der universellen Strukturierungssprache XML und auf Basis einer modularen Architektur ist ein UComT-Trader in der Lage, zur gleichen Zeit Komponentenspezifikationen unterschiedlicher Dienstbeschreibungssprachen sowie auch in unterschiedlichen Sprachen formulierte Suchanfragen zu verarbeiten.

Zurzeit entwickeln wir eine Referenzimplementierung der UComT-Trading-Architektur, anhand derer insbesondere überprüft werden soll, ob der flexible Ansatz des UComT ohne Nachteile für die Performance und für die Wartbarkeit des Tra-

ders umgesetzt werden kann. Als Implementierungsplattform für den Trader-Server nutzen wir mit der Common Language Infrastructure (CLI, ISO 23271/ECMA 334) das neueste der standardisierten und kommerziell verfügbaren Komponentenmodelle. Als Programmiersprache kommt die als „komponentenbasiert“ bezeichnete Sprache C# (ISO 23270/ECMA 334) zum Einsatz, für die auf zahlreichen Plattformen Implementierungen zur Verfügung stehen. Access-Module werden für COM, Java RMI, CORBA IIOP, SOAP und das CLI-Remoting entwickelt; UComT-Clients entstehen in den Sprachen C#, Java und Visual Basic.

```
<?xml version="1.0" encoding="utf-8" ?>
- <offer>
  - <technical>
      <name>mathebib.dll</name>
      <alias>Mathebib</alias>
      <alias>Uni Essen MatheBib</alias>
      <componentmodel name="CLI" version="1.0" />
      <plattform name="Microsoft Windows" version="98" />
      <plattform name="Microsoft Windows" version="NT4" />
      <plattform name="Microsoft Windows" version="2000" />
      <plattform name="Microsoft Windows" version="XP" />
      <plattform name="Microsoft Windows" version="2003" />
    - <descriptions>
      - <description language="ODP" descid="1" technicallevel="5" businesslevel="1">
          <ServiceType>...</ServiceType>
          <ServiceProperties>...</ServiceProperties>
        </description>
        <description language="natural" descid="1" technicallevel="1" businesslevel="5">Mathematische Komponente, die alle Grundrechnenarten (Addition,
          Subtraktion, Multiplikation, Devision) anbietet.</description>
        <description language="KDL" descid="1" technicallevel="3" businesslevel="3">...</description>
      </descriptions>
    - <access protocol="SOAP" version="1.2">
        <address>http://kom.wi-inf.uni-essen.de/UCT/mathlib.aspx</address>
      </access>
    - <access protocol="CLIRemoting" version="1.0">
        <address>132.252.52.20</address>
        <port>9999</port>
      </access>
    - <access protocol="DCOM" version="1.0">
        <address>132.252.52.20</address>
        <port>9998</port>
      </access>
    </technical>
  - <nontechnical>
      <product name="MatheBib" version="1.0" />
    - <billing>
        <costperuse currency="EUR">0.10</costperuse>
      </billing>
      <availability start="1.1.2003" end="1.10.2003" />
      <alternative>http://kom.wi-inf.uni-essen.de/UCT/mathlib2.offer.xml</alternative>
      <alternative>http://kom.wi-inf.uni-essen.de/UCT/mathlib3.offer.xml</alternative>
      <documentation language="german">http://http://www.kom.wi-inf.uni-essen.de/UCT/mathlib.htm</documentation>
    </nontechnical>
  - <trader>
      <extension elementname="/offer/nontechnical/billing/costperuse" type="VBNETScript">http://kom.wi-inf.uni-
        essen.de/UCTex/costperuse.vb</extension>
    </trader>
  </offer>
```

Abbildung 4: Beispiel für ein Registrierungsdokument (Auszug)

Literatur

[ANSA98] ANSA: ANSA Website. http://www.ansa.co.uk, 1998, Abruf am 2003-02-08.

[Bea+97] Bearman, M.; Duddy, K.; Raymond, K.; Vogel, A.: Trader Down Under: Upside Down and Inside Out. In: Theory and Practice of Object Systems TAPOS, Vol. 3(1), 1997.

[Bear97] Bearman, M.: Tutorial on ODP Trading Function. DSTC / Faculty of Information Sciences & Engineering, University of Canberra, 1997. http://www.dstc.edu.au/Products/CORBA/Trading_Service/Tutorial/trtute.html, 1997, Abruf am 2003-01-30.

[BeBe94] Beitz, A.; Bearman, M.: An ODP Trading Service for DCE, Proceedings of the First International Workshop on Services in Distributed and Networked Environments (SDNE), Prag 1994, Los Alamitos/USA 1994.

[Bros02] Brose, G.: JacORB, http://jacorb.inf.fu-berlin.de/features.html, 2003, Abruf am 2003-02-08.

[Desc93] Deschrevel, J.P.: The ANSA Model for Trading and Federation; ANSA Architecture Report APM1.005.01, 15. Juli 1993.

[ECMA02] ECMA: Common Language Infrastructure, Partition I: Concepts and Architecture. ECMA-Dokument ECMA-335 (TC39, Task Group 3), 2002.

[Good95] Goodchild, A.: An evaluation scheme for trader user interfaces. In: Raymond, K.; Armstrong, L. (Hrsg.): Open Distributed Processing – Experiences with distributed environments, London 1995.

[Gud+02] Gudgin, M.; Hadley, M.; Mendelsohn, N.; Moreau, J.-J.; Nielsen, H.F.: SOAP Version 1.2; W3C Candidate Recommendation, 19 December 2002.

[Gut+99] Guttman, E.; Perkins, C.; Veizades, C.; Day, M.: Service Location Protocol, Version 2, IETF RFC 2608.

[Iri+01] Iribarne, L.; Albes, C.; Castro, A.; Vallecillo, A.: A non-functional approach for COTS-components trading. In: Proceedings of the IV. Workshop on Requirements Engineering, Buenos Aires 2001.

[Iri+01a] Iribarne, L.; Troya, J. M.: Trading for COTS components in Open Environments. In: Proceedings of the 27th Euromicro Conference, Los Alamitos/USA 2001.

[ISO94] ISO: Basic Reference Model of Open Distributed Processing. ISO-Dokument Nr. ISO/IEC 10746 (ITU X.901-X.904), Juli 1994.

[ISO98] ISO: Trading Function Specification. ISO-Dokument Nr. ISO/IEC 13235-1 (ITU X.950), 1998.

[JaMu96] Jacob, B.; Mudge, T.: The Trading Function in Action. In: Proceedings of the Seventh ACM SIGOPS European Workshop, Connemara/Ireland 1996.

[KeBe01] Kebbal, D.; Bernard, G.: Component Search Service and Deployment of Distributed Applications. In: Proceedings of the Third International Symposium on Distributed Objects and Applications (DOA'01), Los Alamitos/USA 2001.

[Kini99] Kiniry, J.R.: Leading to a Kind Description Language: Thoughts on Component Specification, California Institute of Technology Technical Report CS-TR-99-04, Pasadena/USA 1999.

[KoBu95] Kovacs, E.; Burger, C.: Projektbeschreibung MELODY – Management Environment for Large Open Distributed sYstems, Fakultätsbericht 8/95 der Fakultät Informatik der Universität Stuttgart, Stuttgart 1995.

[Kova96] Kovacs, E.: Advanced Trading Services Through Mobile Agents. In: Proceedings of the Workshop on Trends in Distributed Systems (TreDS'96), Aachen 1996.

[Kutv94] Kutvonen, L.: Comparison of the DRYAD Trading System to ODP Trading Function Draft, Technical Report C-1994-51, University of Helsinki 1994.

[Lee96] Lee, O.K.: A Trading Service for Mobile Code. Vorschlag für W3C/OMG Workshop on Distributed Objects and Mobile Code 1996, Boston 2002.

[Mar+01] Marvie, R.; Merle, P.; Geib, J.-M.; Lebanc, S.: TORBA: Trading Contracts for CORBA. In: Proceedings of the 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'01), San Antonio/USA 2001.

[Mue+95] Mueller-Jones, K.; Merz, M.; Lamersdorf, W.: The TRADEr: Integrating Trading into DCE. In: Raymond, K.; Armstrong, L. (Hrsg.): Open Distributed Processing – Experiences with distributed environments, London 1995.

[OMG00] OMG: Trading Object Service Specification, Version 1.0, May 2000.

[Pete98] Peterbauer, K.: CORBazar – A Trading Object Service; 1998; http://www.infosys.tuwien.ac.at/Teaching/Finished/Praktika/CORBazar/, 1998, Abruf am 2003-02-08.

[PoMe93] Popien, C.; Meyer, B.: Federating ODP Traders: An X.500 Approach, Proceedings of the International Conference on Communications, Geneva/Switzerland 1993.

[Pude95] Puder, A.; Gudermann, F.; Markwitz, S.: Ein Mehrphasen-Protokoll für wissensbasierte Dienstvermittlung. In: Entwicklung und Management verteilter Anwendungssysteme (EMVA), Münster 1995.

[Pud+95a] Puder, A.; Markwitz, S.; Gudermann, F.: Service Trading Using Conceptual Structures. In: Proceedings of the 3rd International Conference on Conceptual Structures (ICCS`95), Berlin 1995.

[Raym91] Raymond, K.: Go-Between: A Prototype Trader. CEDIS Report, University of Queensland/Australien, 1991.

[RiHo95] Richman, A.; Hoang, D.: Accomplishing Distributed Traders Utilizing the X.500 Directory. In: 2nd IEEE Malaysia International Conference on Communications (MICC'95), Malaysia 1995.

[SeNa01] Senivongse, T.; Nanekrangsan, W.: An Extension to a CORBA Trader to Support XML Service Descriptions. In: New Developments in Distributed Applications and Interoperable Systems, Proceedings of the 3. International Working Conference on Distributed Applications and Interoperable Systems, Deventer 2001.

[TeNi00] Terzis, S.; Nixon, P.: Towards a Semantically Enhanced Component Trader Architecture, Technical Report TCD-CS-2000-23, Trinity College, Dublin 2000.

[TeNi99] Terzis, S.; Nixon, P.: Component Trading: The basis for a Component-Oriented Development Framework. In: Bosch, J.; Szyperski, S.; Weck, W. (Hrsg.): Proceedings of the 4th International Workshop on Component-Oriented Programming, Berlin 1999.

[VaBa98] Vasudevan,V.; Bannon, T.: WebTrader: Discovery and Programmed Access to Web-Based Services, Technical Report, Object Services and Consulting Inc. 1998.

[Vasu98] Vasudevan, V.: Augmenting OMG traders to handle Service Composition. Technical Report, Object Services and Consulting Inc. 1998.

[Vog+95] Vogel, A.; Bearman, M.; Beitz, A.: Enabling Interworking of Traders. In: Raymond, K.; Armstrong, L. (Hrsg.): Open Distributed Processing – Experiences with distributed environments, London 1995.

[WaBe95] Waugh, A.; Bearman, M.: Designing an ODP Trader Implementation using X.500, Proceedings of the International Conference on Open Distributed Processing 95, Brisbane/Australien 1995.

[War+94] Warren Pratten, A.; Hong, J.W.; Bennett, J.M.; Bauer, M.A.; Lutfiyya, H.: Design and Implementation of a Trader-Based Resource Management System, Proceedings of CASCON'94, Toronto/Canada 1994.

[Wido02] Widoff, S.B.: TAO Trading Service Documentation, November 2002. http://www.cs.wustl.edu/~schmidt/ACE_wrappers/TAO/docs/releasenotes/trader.html, 2002, Abruf am 2003-02-08.