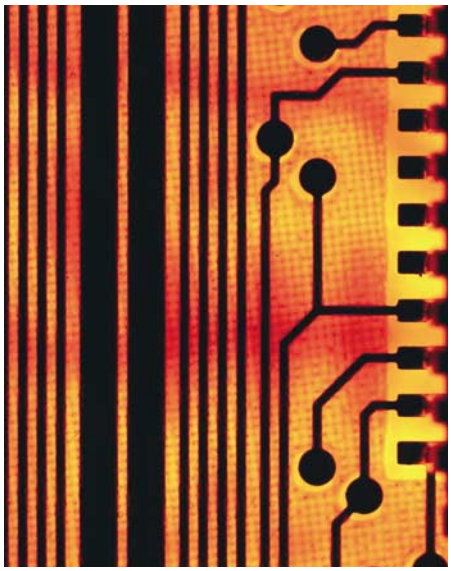
UDDI Revisited

Mit der Version 3 wird UDDI erwachsen



Obwohl UDDI meistens in einem Atemzug mit SOAP und WSDL genannt wird, fristet der Standard zur Verwaltung von Informationen über Web Services außerhalb von PowerPoint-Präsentationen meist noch ein Schattendasein. Mit der vor der Verabschiedung stehenden Version 3 wird UDDI um diverse Merkmale ergänzt, die es für den Einsatz abseits der ursprünglichen Science Fiction-Vision qualifizieren. Grund genug, den Standard noch einmal detailliert zu betrachten.

von Stefan Tilkov

UDDI steht für Universal Description, Discovery and Integration [1] und wird vom UDDI Technical Committee innerhalb von OASIS [2], einem weltweiten Industrie-Konsortium zur Definition von eBusiness-Standards standardisiert. Ziel ist es, Informationen über Dienstanbieter und die von ihnen zur Verfügung gestellte Web Services so zu verwalten, dass Web Services sowohl zur Entwicklungs- als auch zur Laufzeit einfach gefunden und genutzt werden können.

Vision und Realität

Gestartet wurde UDDI mit der Vision einer weltweiten, replizierten Registratur, ähnlich dem Domain Name System (DNS), das für die Verwaltung von Internet-Domain-Namen verwendet wird. In den Präsentationen aus UDDI-Gründertagen finden sich Beispiele wie das eines kleinen Trödelladens aus Neuseeland, der seine Dienste im zentralen UDDI-Verzeichnis registriert und dessen Produkte dadurch – vollautomatisch und ohne Benutzerintervention – in die Kataloge großer Online-Anbieter aufgenommen werden. So funktioniert das Geschäftsleben in der Praxis jedoch nicht – zumindest noch nicht. Dem ersten Auftrag, der ersten Bestellung geht ein Auswahlprozess voraus, der nicht so einfach automatisiert werden kann, wie es sich die Verfechter dieser Idee gewünscht hätten. In der Regel werden Geschäftsbeziehungen zu Partnerschaften; gute Erfahrungen aus der Vergangenheit werden ebenso wie persönliche Beziehungen als wichtiges Kriterium bei der Bewertung herangezogen. Die Vorstellung, eine mehrere Hunderttausend Euro schwere Bestellung vom System automatisch an einen bis dato unbekannten Anbieter zu vergeben, zeugt von einer Naivität, durch die viele potenzielle Anwender davon abgehalten wurden, UDDI ernst zu nehmen.

Die normative Kraft des Faktischen – in diesem Fall die Tatsache, dass Unternehmen die ursprüngliche Vision in der Praxis schlicht ignoriert haben – hat dazu geführt, dass UDDI in der Version 3 um zahlreiche Merkmale ergänzt wurde, die dessen Einsatz innerhalb von Unternehmen und zwischen ohnehin miteinander kooperierenden Partnern unterstützen. Das zentrale, von IBM, Microsoft und anderen betriebene UDDI-Verzeichnis [3] existiert auch weiterhin als UBR (Universal Business Registry); daneben können jedoch sowohl separate öffentliche Verzeichnisse als auch unternehmensinterne betrieben werden. UDDI v3 bietet dabei die Möglichkeit, Dienste und andere Entitäten aus einer Registry in eine andere zu transferieren. Bevor ich jedoch auf diese und andere wesentliche Neuerungen der neuen Version eingehe, möchte ich zunächst eine Einführung in die grundlegenden UDDI-Prinzipien voranstellen.

Das Datenmodell von UDDI

Die Grundlage von UDDI bildet zunächst ein vergleichsweise einfaches Datenmodell zur Abbildung von Dienstanbietern, abstrakten Diensten und deren konkreter Implementierung (siehe Abb. 1):

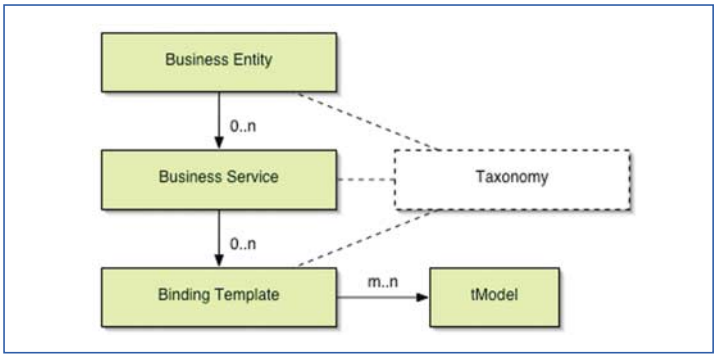


Abb. 1: UDDI-Kernentitäten

© Software & Support Verlag

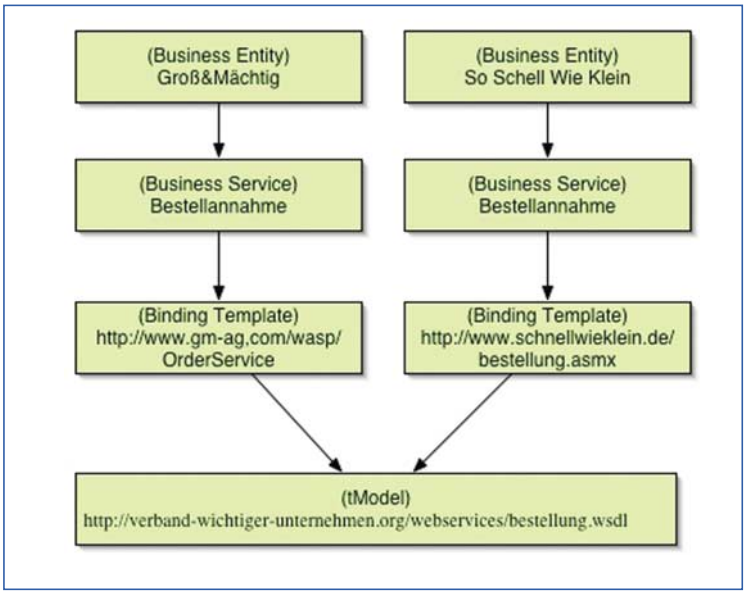


Abb. 2: Beispielentitäten

- *Business Entity*: Dahinter kann sich eine beliebige Organisation – ein Unternehmen, eine Unternehmenseinheit, eine Institution usw. – verbergen, die Dienste in Form von Web Services zur Verfügung stellt. Einer Business Entity können mehrere Business Services zugeordnet sein. In der Abbildung nicht dargestellt sind Publisher Assertions, die Beziehungen zwischen zwei Business Entities kapseln (aus der Sicht desjenigen, der die Aussage veröffentlicht).
- *Business Service*: Dies ist eine logische Gruppierung von Diensten, die einem bestimmten Zweck dienen. Jeder Business Service ist genau einer Business Entity zugeordnet, das Modell ist also streng hierarchisch. Der Business Service ist eine abstrakte Beschreibung, unabhängig von einer konkreten technischen Schnittstellenform. Es kann eine beliebige Anzahl solcher Schnittstellen geben, z.B. wenn Dienste sowohl über SOAP, eine interaktive Benutzerschnittstelle in Form einer Webanwendung, eMail oder das gute alte Telefon zur Verfügung gestellt werden.
- *Binding Template*: An dieser Stelle wird es nun technisch: Ein Binding Template bildet genau eine technische Schnittstelle ab, die an einem bestimmten Endpunkt – z.B. einer SOAP- oder eMail-Adresse – existiert. Ein Binding Template ist wiederum genau einem Business Service zugeordnet.

Bis zu dieser Stelle ist das Modell sehr einfach – ein Unternehmen bietet verschiedene Dienste an, diese wiederum können technisch unterschiedlich abgebildet werden. Was passiert nun, wenn eine technische Schnittstelle von mehreren Unternehmen angeboten wird? An dieser Stelle kommt zum ersten Mal die UDDI-Allzweckwaffe ins Spiel – das *tModel*.

- *Technical Model* (tModel): Das tModel findet in UDDI für unterschiedliche Zwecke Verwendung. Der wichtigste ist die Spezifikation einer technischen Schnittstelle – z.B. eines konkreten Typs von Web Services mit klar definierten Operationen und Datenstrukturen. Ein Binding Template kann auf mehrere tModels verweisen. Diese haben jedoch keine Assoziation zu genau einem Binding Template, sondern können von mehreren Binding Templates verwendet werden. Allgemein werden tModels benutzt, um ein wiederverwendbares, technisches Konzept abzubilden; dabei kann es sich auch um eine Taxonomie oder ein Protokoll wie SOAP oder HTTP handeln. Doch dazu später mehr.

Zunächst möchte ich an einem Beispiel den Zusammenhang der vier Kernentitäten veranschaulichen: Nehmen wir an, ein Unternehmensverband definiert einen Standard-Web Service zur Durchführung von Bestellungen. Dazu wird eine Schnittstellenbeschreibung erstellt, üblicherweise in Form eine WSDL-Datei. Im UDDI-Verzeichnis wird dazu ein tModel erstellt, das die WSDL-Datei referenziert. (Dies geschieht typischerweise über einen Web-URI, z.B. *http://verband-wichtiger-unternehmen.org/webservices/bestellung.wsdl*. Spätestens an dieser Stelle sollte auch ein weiterer Punkt klar sein: UDDI ist eine Registry, kein Repository; d.h. es werden Verweise auf Informationen darin registriert, nicht jedoch die Informationen selbst aufgenommen.)

Informationen über die Unternehmen, die den Web Service (und vielleicht auch noch eine Reihe weiterer) unterstützen, sind im Verzeichnis als Business Entity abgebildet. Im Beispiel könnte es sich um die Anbieter Groß&Mächtig AG und So-SchnellWieKlein GmbH handeln, die beide Bestellungen annehmen – in UDDI-Terminologie: einen Business Service *Bestellannahme* anbieten. Demnach werden im Verzeichnis zwei Business Services registriert.

Ebenfalls für jedes der beiden Unternehmen wird nun ein Binding Template erstellt, das dem Business Service *Bestellannahme* zugeordnet ist, eine Verbindung zu dem für den Web Service registrierten tModel enthält und diesen wiederum an einen konkreten Endpunkt bindet – z.B. *http://www.gm-ag.com/wasp/OrderService* und *http:// schnellwieklein.de/bestellung.asmx*. Eine grafische Darstellung dieses Beispiels finden Sie in Abbildung 2.

Zugriffsschicht

Der Zugriff auf die Informationen im UDDI-Verzeichnis erfolgt über – wie sollte es anders sein – eine SOAP-Schnittstelle. Die UDDI-Spezifikation definiert dazu verschiedene Gruppen von APIs, so genannte API Sets. Die wichtigsten davon sind das Inquiry API Set für die Suche nach und die Abfrage von Informationen sowie das Publication API Set für die Veröffentlichung (Registrierung) von Informationen im Verzeichnis. Neben der WSDL-Beschreibung dieser Schnittstellen existieren außer der Open Source-Bibliothek UDDI4J [4] noch die herstellerabhängigen Zugriffs-Bibliotheken der Anbieter von UDDI-Verzeichnissen, die zum Teil eine effizientere Zugriffsmöglichkeit bieten. Jenseits der programmatischen Zugriffsmöglichkei-

ten bieten UDDI-Implementierungen immer auch eine Benutzerschnittstelle, in aller Regel in Form einer Webanwendung, die ebenfalls die Verwaltung von UDDI-Informationen erlaubt.

Metadaten, Kategorien, Taxonomien

Das UDDI-Kernmodell ist sehr einfach – genau genommen zu einfach, um damit allein alle Informationen abbilden zu können, die für ein Unternehmen oder einen Dienst möglicherweise sinnvoll sein könnten. Anstatt nun ein hochkomplexes Modell zu entwerfen und mit allen Beteiligten abzustimmen, um es dann am Ende doch nicht allen recht machen zu können, haben sich die Autoren der UDDI-Spezifikation einen anderen Weg ausgesucht: Die Verwendung von Metadaten.

UDDI erlaubt es, die Kernentitäten – Business Entities, Business Services, Binding Templates und tModels – nach unterschiedlichsten Systemen zu identifizieren und zu kategorisieren. So vergibt das Unternehmen D&B einen international anerkannten Identifikationscode für Unternehmen und deren Geschäftsbereiche, die 9-stellige D-U-N-S-Nummer; ein weiteres Beispiel für eine Identifikation wäre eine Umsatzsteuer-Identifikationsnummer (VAT-ID). Neben dieser (hoffentlich) eindeutigen Identifikation kann ein Unternehmen oder auch ein Dienst unterschiedlichsten Kategorien zugeordnet werden. Ein Beispiel dafür ist das United Nations Standard Products and Services Code System (UNSPSC). Es ist offensichtlich, dass es eine schlechte Idee wäre, in der Business Entity ein Attribut *D-U-N-S* oder *UNSPSC* aufzunehmen – schließlich gibt es neben vielen internationalen und nationalen Normen auch unternehmsinterne Identifikations- und Kategorisierungssysteme.

Aus diesem Grund ermöglicht es UDDI, vordefinierte oder benutzerdefinierte Taxonomien, also Identifikations- oder Kategorisierungssysteme, zu registrieren und zu referenzieren – und zwar aus allen Kernentitäten heraus. Vereinfacht erklärt können mit den Entitäten beliebige Werte assoziiert werden, die wiederum mit der Taxonomie – also mit dem Klassifizierungssystem – verbunden werden, in dem sie Sinn machen. Mit einer 9-stelligen Nummer allein kann man nicht viel anfangen, mit der Information, das es sich bei einer 9-stelligen Nummer um den D-U-N-S-Code handelt, sehr wohl. Taxonomien selbst wiederum werden als tModels abgebildet, können also ebenfalls kategorisiert werden. Es gibt eine Reihe vordefinierter Taxonomien als Teil des Standards.

Es ist natürlich möglich, Entitäten nach mehr als einem System zu identifizieren und zu kategorisieren. Damit wird das im Grunde einfache UDDI-Datenmodell mächtig genug, um selbst komplexe Informationen abzubilden und – womit wir bei der Antwort auf die Frage nach dem Nutzen wären – diese für die Suche und vor allem das Finden von registrierten Diensten einzusetzen. Für Unternehmen ergibt sich dadurch die Möglichkeit, die für die internen Bedürfnisse optimale Kategorisierungs- und Identifikationsstruktur als Kombination öffentlich verfügbarer und selbstdefinierter Systematiken in einem UDDI-Verzeichnis abzubilden.

Neues in Version 3

Die aktuelle Version 3 der UDDI-Spezifikation – die zum Zeitpunkt des Erscheinens dieses Artikels verabschiedet sein sollte – bringt verschiedene Verbesserungen und neue Funktionen gegenüber der Vorgängerversion mit sich. Zum größten Teil handelt es sich dabei um Merkmale, die schon vorher bereits in einigen Implementierungen umgesetzt waren und daher praxiserprobt sind.

Die wesentlichen Verbesserungen an bereits bestehenden Konzepten sind die Unterstützung für benutzerdefinierte Schlüssel, die Möglichkeit, auch Binding Templates zu kategorisieren, Verbesserungen der Zugriffs-APIs sowie die Möglichkeit, bestehende Taxonomien mittels Vererbung zu erweitern.

Bis inklusive Version 2 mussten Schlüssel für UDDI-Entitäten von der Implementierung automatisch beim Publizieren erzeugt werden. Version 3 erlaubt nun, diese Schlüssel vorzubelegen. Gleichzeitig wurde ein neues, nunmehr auch von Menschen lesbares Format für die Schlüssel eingeführt, das URI-basiert ist (also z.B. *uddi:uddi.org:v3_publication* anstelle von *uuid:64C756D1-3374-4E00-AE83-EE12E38FAE63*).

UDDI und WSDL

Für die Beschreibung von Web Services-Schnittstellen kommt üblicherweise WSDL zum Einsatz. Im Gegensatz zu analogen Standards aus der Welt der verteilten Objektsysteme wie z.B. CORBA IDL enthält eine WSDL-Beschreibung nicht nur Informationen über die verfügbaren Operationen und Datentypen, sondern auch über die Lokation, an der ein Dienst aufgerufen werden kann. Dies ist nur ein Beispiel für die Überschneidungen, die es zwischen WSDL und UDDI gibt. Beschäftigt man sich zum ersten Mal mit dieser Thematik, ist man (zu Recht) irritiert – es entsteht der Eindruck, dass UDDI und WSDL nicht so recht miteinander harmonieren.

Das UDDI-Konsortium trägt diesem Umstand durch eine Technical Note [5] Rechnung, die definiert, wie WSDL-Informationen auf das UDDI-Datenmodell abgebildet werden. Eine sehr gute Beschreibung dazu findet sich in [6]; die wesentlichen Aspekte werden durch Abbildung 3 verdeutlicht.

Die *portType*- und *binding*-Elemente aus der WSDL-Beschreibung werden auf *tModel*-Elemente in UDDI abgebildet und eindeutig als WSDL-Elemente kategorisiert (über durch den UDDI-Standard vordefinierte Kategorien). Die

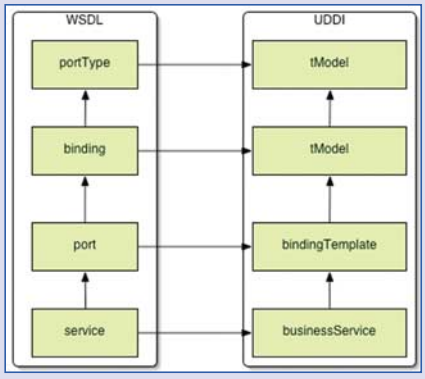


Abb. 3: Abbildung von WSDL-Informationen auf das UDDI-Datenmodell

Rolle des *port*-Elements aus WSDL – die Verknüpfung eines abstrakten Kommunikationsschemas mit einem konkreten Transportprotokoll – übernimmt in UDDI das Binding Template, ein WSDL-Service wird auf einen UDDI-Business Service abgebildet.

Einige UDDI-Implementierungen, z.B. Systinet WASP UDDI [7], bieten eine vereinfachte Programmierschnittstelle, um eine WSDL-Beschreibung mit einem einzigen Aufruf konform zur Technical Note im Verzeichnis zu registrieren.

© Software & Support Verlag

Die Zugriffs-APIs wurden in zweierlei Hinsicht verbessert. Zum einen gibt es eine Reihe von Standard-Anwendungsfällen, die nun direkt von der Registry in Form eines einzelnen Aufrufs unterstützt werden (und nicht mehr mehrere einzelne Aufrufe erfordern). Hieraus ergibt sich eine deutliche Performance-Verbesserung. Umfasst das Ergebnis einer Suchabfrage eine große Menge von Elementen, können diese nun in Blöcken definierbarer Größe abgefragt werden.

Über die Möglichkeit zur Abbildung von Vererbungsbeziehungen zwischen Taxonomien können Basistaxonomien elegant wiederverwendet werden.

Konzeptionelle Erweiterungen

Neben den Detailverbesserungen gibt es auch einige konzeptionelle Erweiterungen; die wichtigste davon ist ohne Zweifel die Unterstützung mehrerer, föderierter Verzeichnisse mit der ursprünglich einzigen UDDI-Registry (UBR) als Wurzel.

Die Spezifikation legt nun fest, wie UDDI-Instanzen untereinander Daten austauschen und synchron halten. Dies erlaubt ein mehrstufiges Konzept, in dem Unternehmen z.B. eine unternehmensinterne, eine mit Partnern gemeinsam genutzte und die zentrale Registry verwenden können, um Services unterschiedlichen Zielgruppen zur Verfügung zu stellen, ohne Redundanzen zu erzeugen.

Eine weitere wesentliche Erweiterung ist die Unterstützung eines Subskriptionsmechanismus. Dieser ermöglicht es, bei der UDDI-Instanz einen Web Service zu registrieren, der bei Änderungen an UDDI-Entitäten benachrichtigt wird.

Nutzungsmodelle

Grundsätzlich können die Informationen in der UDDI-Registry sowohl zur Entwicklungszeit eines Web Service-Consumers als auch zur Laufzeit verwendet werden.

Im ersten Fall dient das Verzeichnis als Unterstützung für Entwickler, um zunächst einmal herauszufinden, welche Dienste überhaupt angeboten werden, wie sie technisch aufzurufen sind und wo sie erreicht werden können. Web Services-Entwicklungsumgebungen mit einem integrierten UDDI-Browser unterstützen dabei z.B. die direkte Generierung einer clientseitigen Programmierschnittstelle in der passenden Programmiersprache. Gleichzeitig können Web Services typischerweise aus der Entwicklungs- oder Ablaufumgebung heraus in eine UDDI-Registry publiziert werden.

Oft vernachlässigt, aber der eigentlich interessantere Fall ist die Verwendung von UDDI zur Unterstützung eines dynamischen Lookups. Die Verwendung der konkreten Aufrufadresse (des Endpoints) eines Services in einem Services-Consumer erzeugt eine recht enge Kopplung, da z.B. ein Wechsel des Systems, auf dem der Dienst angeboten wird, nicht ohne Änderung des Consumers möglich ist. Über die UDDI-Registry kann hierzu eine Indirektion eingefügt werden – der Consumer kennt die Adresse der UDDI-Registry sowie den Namen oder den Schlüssel des Binding Templates für den Service, den er verwenden möchte; zur Laufzeit führt er (grundsätzlich, beim Start oder nur im Fehlerfall) eine Suche im Verzeichnis durch, um die Adresse des eigentlichen Dienstes zu erfragen bzw. zu aktualisieren.

Natürlich sind zur Laufzeit auch komplexere Nutzungsmöglichkeiten der UDDI-Informationen denkbar – hier sind der Fantasie im Prinzip keine Grenzen gesetzt (solange die Performanz des Systems nicht über Gebühr leidet). So könnte die dynamische Zuordnung eines Dienstes auf Basis geografischer Informationen erfolgen, ein gleichzeitiger Aufruf mehrerer Dienste erfolgen, die die gleiche Schnittstelle unterstützen, oder eine Transformation von Daten durch einen Dienst erfolgen, der dafür entsprechend registriert wurde.

Die Rolle von UDDI in einer SOA

In den letzten sechs bis zwölf Monaten nehmen wir ein stark erhöhtes Interesse an UDDI wahr, was aus unserer Sicht im Wesentlichen auf drei Faktoren zurückzuführen ist:

1. Viele Unternehmen haben im Bereich Web Services die Experimentierphase hinter sich gelassen. In unterschiedlichen Geschäftsbereichen und Abteilungen entstehen – nicht zuletzt wegen der geringen Einstiegshürde durch die Qualität der Entwicklungswerkzeuge – eine Vielzahl von Web Services. UDDI ist der nahe liegende Mechanismus, diese Services geordnet zu verwalten und das damit verbundene Wiederverwendungspotenzial zu erschließen.

2. Die Qualität der Implementierungen hat sich stark verbessert; insbesondere die Systeme, die bereits jetzt den Einsatz in einem unternehmensinternen Umfeld unterstützen – z.B. mit darauf ausgelegten Sicherheits- und Replikationsmechanismen – erlauben eine schnelle Inbetriebnahme einer Lösung, ohne massiv auf Eigenentwicklungen angewiesen zu sein.

3. Langsam, aber stetig fortschreitend setzt sich die Meinung durch, dass eine Service-orientierte Architektur (SOA) mehr sein muss als eine auf SOAP übertragene, RPC-orientierte CORBA-, DCOM- oder RMI-Architektur. Services im Sinne einer SOA haben eine andere Granularität als RPC- oder auch Komponentenschnittstellen; die Einführung einer SOA erfolgt daher auf einer anderen Ebene. In diesem Umfeld ist das Management der Dienste – neben der reinen Verwaltung von Informationen à la UDDI also auch die Überwachung, Absicherung usw. – eine Kernanforderung.

Fazit

Unternehmen, die Web Services zur Integration von Anwendungskomponenten verwenden – z.B. als Brücke zwischen einem .NET-Client und einem J2EE-Server – werden auf UDDI wahrscheinlich verzichten können. Auch für eine Punkt-zu-Punkt-Integration ist UDDI eher überflüssig.

Je mehr Services jedoch entstehen, je stärker diese auch außerhalb des ursprünglich vorhersehbaren Szenarios verwendet werden, kurz: je größer die Rolle ist, die der Gedanke der Service-Orientierung spielt, um so mehr Sinn macht es, sich mit UDDI zu beschäftigen. ●

Links & Literatur

[1] www.uddi.org/

[2] www.oasis-open.org/

[3] www.uddi.org/find.html

[4] www.uddi4j.org/

[5] www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v200-20031104.htm

[6] www.ibm.com/developerworks/webservices/library/ws-udmod1/

[7] www.systinet.com/products/wasp_uddi/

Martina Klassen, martina.klassen@innoq.com
innoQ Deutschland GmbH
Halskestr. 17
40880 Ratingen
Tel.: (0 21 02) 77162-100