Hyperdatenbanken zur Verwaltung von Informationsräumen

Hyperdatabases for Managing Information Spaces

Hans-Jörg Schek[1,2], Heiko Schuldt[2], Christoph Schuler[1], Can Türker[1], Roger Weber[1]
1 ETH Zürich (Schweiz)
2 UMIT Innsbruck (Österreich)

Zusammenfassung Mit den enormen Fortschritten in der Kommunikationstechnologie wächst die ehemals in isolierten Daten- und Dokumentenkollektionen verwaltete Information zu einem dicht verwobenen Verbund von Informationskomponenten – zu einem Informationsraum – zusammen. Um die damit verbundene Informationsflut in den Griff zu bekommen, ist eine neue Infrastruktur gefordert, die Koordinationsprozesse und zusammengesetzte Dienste unterstützt und deren korrekte Ausführung auch bei Systemfehlern und Nebenläufigkeit garantiert. Dieser Artikel skizziert das Konzept einer Hyperdatenbank, die eine verteilte und hochgradig skalierbare Infrastruktur eines beliebig großen, dynamischen Informationsraumes bereitstellt. Die Grundidee besteht darin, den kooperierenden Diensten eine zusätzliche Schicht hinzuzufügen, die wir Hyperdatenbankschicht nennen. Während die Vernetzung Bytes durch den Informationsraum transportiert, ist die Hyperdatenbankschicht – neben anderen Aufgaben – für die korrekte Ausführung von dienstübergreifenden Prozessen zuständig. ►►► **Summary** With the enormous advances in communication technology, the information that was originally managed in isolated data and document collections is glued to a web of information components, called information space. To cope with the implied information flood, a new infrastructure is required that supports coordination processes and composition of services and that guarantees their correct execution even in case of system failures and concurrency. This article sketches the concept of a hyperdatabase, which provides a highly scalable and distributed infrastructure of a large, dynamically changing information space. The basic idea is that each cooperating service is equipped with an additional layer, called hyperdatabase layer. While the network is responsible for transferring bytes through the information space, the hyperdatabase layer – besides other tasks – will correctly execute processes on top of existing services.

KEYWORDS H.2.7 [*Database Management*] Database Administration, Hyperdatenbank, Informationsraum, Prozessmanagement, Metadatenverwaltung, Publish & Subscribe, Koordinationsdienste, Informationsinfrastruktur

1 Von isolierten Datenquellen zu Informationsräumen

In der Vergangenheit, als die Kommunikationskosten im Vergleich zu den Rechenkosten noch dominierten, beherrschten monolithische Infrastrukturen die Informationsverarbeitung. Mit den enormen Fortschritten in der Kommunikationstechnologie setzt sich jedoch immer mehr der Trend hin zu einer verteilten, parallelen Informationsverwaltung durch. Information wird heutzutage nicht nur in vielen klassischen Datenbanken und Dokumentenkollektionen verwaltet, sondern tritt massiv zunehmend auch auf Webseiten, in XML-Dokumenten und hinter Web-Formularen verborgen auf. Die Fortschritte in der mobilen Kommunikation ermöglichen zudem eine immer stärkere Einbindung von eingebetteten Systemen und mobilen „Smart Objects“, welche Daten über verschiedene Sensoren erfassen und über wohldefinierte Schnittstellen anbieten. Wir befinden uns also, je länger desto mehr, in einem sehr dicht verwobenen Verbund von Informationsträgern, -anbietern und -verbrauchern; der Begriff „*Informationsraum*“ soll diese inhärente Verteilung von Information symbolisieren.

Verbunden mit dem technologischen Wandel setzt sich immer mehr eine *dienstorientierte* Betrachtung von Information gegenüber einer *daten-* und *dokumentenzentrierten* durch. Einerseits können, wie im

klassischen Fall, Daten und Dokumente über Dienstschnittstellen angefordert und verwaltet werden. Andererseits treten aber vermehrt neuere Formen von Diensten auf, die selber keine Daten halten, sondern Daten verarbeiten, analysieren, aggregieren oder auf der Basis von Daten verschiedener Quellen neue Schlüsse ziehen [13]. In einem Informationsraum spielt es allerdings keine Rolle, ob ein Dienst selbst Daten trägt oder ob er nur Daten verarbeitet. Jeder Dienst repräsentiert eine Dimension des Informationsraumes, der durch die Menge der Dienste aufgespannt wird. Information ist nun nicht nur ein Datum hinter irgendeiner Dienstschnittstelle, sondern vielmehr die Anreicherung verschiedener Ursprungsdaten unter Verwendung von berechnenden Diensten.

Ein Beispiel für einen Informationsraum liefert das ETHWorld-Projekt [6; 16], das zum Ziel hat, einen virtuellen Campus aufzubauen, in dem sich Studenten, Assistenten, Professoren, Forscher usw. treffen. Die Teilnehmer treten dabei einerseits in der Rolle des Informationsanbieters auf, andererseits aber auch in der Rolle des Informationssuchenden; als Vermittler treten Suchmaschinen auf. In ETHWorld werden nicht nur textuelle Informationsbedürfnisse unterstützt, sondern auch visuelle. Teilnehmer von ETHWorld können daher auch nach Abbildungen, Fotos, Skizzen und Zeichnungen suchen. In der klassischen Betrachtung fand die Informationssuche häufig außerhalb der Datenbanken statt. Dabei wurden die Daten periodisch in die Suchstrukturen überführt. In ETHWorld verfolgen wir einen integrierten Ansatz, bei dem die Datenhaltung, die Aufbereitung der Dokumente und die Suche als eine Einheit in dem Sinne gesehen werden, dass zwischen ihnen Abhängigkeiten bestehen. Bild 1 zeigt dies im Falle eines Einfügens eines neuen Molekülbildes. Die Bestandteile des Bildes werden analysiert (z. B. Formerkennung, Strukturerkennung) und in einem speziellen Suchdienst abgelegt. Mit anderen Worten: die Suchstrukturen werden inkrementell mit den Datenquellen abgeglichen. Das ETHWorld-Beispiel verdeutlicht, dass Informationsquellen mit rechenintensiven Diensten wie Textklassifikation, Termextraktion, Bildverarbeitung verbunden werden. Jederzeit können weitere Dienste hinzukommen.

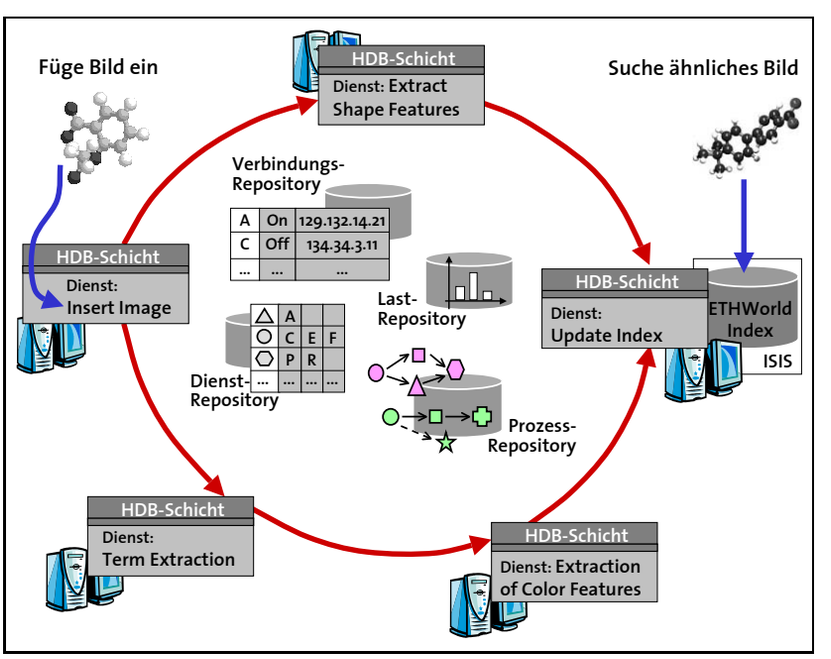

Bild 1 ETHWorld-Anwendungsbeispiel.

Abhängigkeiten, wie sie im obigen Beispiel zwischen Datenquelle und Suchindex vorkommen, treten in Informationsräumen in großem Maße auf. Unsere Arbeit widmet sich jedoch nicht der Modellierung oder der automatischen Herleitung solcher Abhängigkeiten, sondern geht der Frage nach, wie in einem so stark verteilten Informationssystem diese Abhängigkeiten eingehalten werden können. Zu diesem Zweck führen wir den Begriff des Koordinationsprozesses ein: wie die SQL-Anweisungen eines Triggers die Einhaltung von Abhängigkeiten über mehrere Tabellen hinweg garantieren können, soll ein Koordinationsprozess gekoppelt an ein Ereignis (Änderung der Quelldaten) die Integrität innerhalb des Informationsraumes garantieren. Die Transaktionskontrolle eines Datenbanksystems garantiert, dass die Änderung der Ursprungsdaten auch das Nachführen der Abhängigkeiten einschließt. Analog muss die Infrastruktur für einen Informationsraum dafür sorgen, dass eine beobachtete Änderung auch zwangsläufig zum korrekten Ausführen eines daran gekoppelten Prozesses führt. Wegen der Autonomie der Teilsysteme kann dies allerdings nicht mehr innerhalb einer Transaktionsklammer geschehen. Stattdessen muss die Infrastruktur die Ausführung eines einmal gestarteten Prozesses stets zu einem wohldefinierten Ende führen, unabhängig davon, ob andere Prozesse laufen oder ob Systeme abstürzen, sich abmelden oder später erst wieder erreichbar sind (verallgemeinerte Atomarität und Isolation) [18]. Wir nennen eine solche Infrastruktur eine *Hyperdatenbank*, also eine Datenbank über Datenbanken, da sie ähnliche Aufgaben erfüllen muss wie eine klassische Datenbank, aber anstelle von Tabellen und Tupeln mit Diensten, Aufrufen und Parametern operiert.

Die Koordinationsprozesse müssen zum einen wohldefiniert und zum anderen mit den auslösenden Ereignissen verbunden sein. Letzte-

res impliziert, dass die Dienstaufrufe automatisch überwacht werden können. Solche Dienste bezeichnen wir als *kooperierend*. Nicht-kooperierende Dienste können nicht als auslösende Ereignisse in einen Koordinationsprozess eingebunden werden. Eine grundlegende Anforderung an die konsistente Verwaltung eines Informationsraumes ist daher, dass alle Dienste, die einen Seiteneffekt innerhalb des Informationsraumes haben, kooperierend sein müssen. Neben den Koordinationsprozessen existieren weitere Klassen von Prozessen, die mehrere Dienste kombinieren, integrieren und selbst wieder einen Dienst anbieten. Ein Beispiel hierfür wäre die Suche über mehrere Suchmaschinen hinweg, die elektronische Einschreibung (welche einen komplexen administrativen Workflow realisiert) oder das Erstellen von Berichten über verschiedene Informationsquellen (z. B. Projektbericht mit Personendaten und Veröffentlichungen). Grundsätzlich unterscheiden sich diese Prozesse von den Koordinationsprozessen nur durch ihre Art der Instanziierung. Einmal gestartet muss die Infrastruktur für beide Arten von Prozessen die gleichen Ausführungsbedingungen erfüllen.

Das nachfolgende Kapitel skizziert den Ansatz einer Hyperdatenbank, welche die zukünftige Infrastruktur zur Verwaltung eines Informationsraumes anbietet. Kapitel 3 berichtet von der Implementierung eines Hyperdatenbank-Prototypsystems. Ein kurzer Ausblick auf zukünftige Arbeiten beschließt diesen Artikel.

2 HDB = Netzwerk + Datenbanken

In unserer Betrachtung stellt eine Hyperdatenbank (HDB) die zukünftige Infrastruktur für die Verwaltung von Informationsräumen dar. Sie soll bestehende Dienste verknüpfen, Daten aggregieren, kombinieren und anreichern und so neue, höherwertige Dienste anbieten können. Zu diesem Zweck benötigen wir zunächst eine Prozessspezifikationsumgebung, wofür wir grafisch orientierte Werkzeuge verwenden. Auf diese gehen wir im Folgenden nicht weiter ein. Das Ergebnis ist in jedem Fall ein Prozess, der von der HDB wiederum als Dienst angeboten wird und welcher Bestandteil von weiteren Prozessen sein kann.

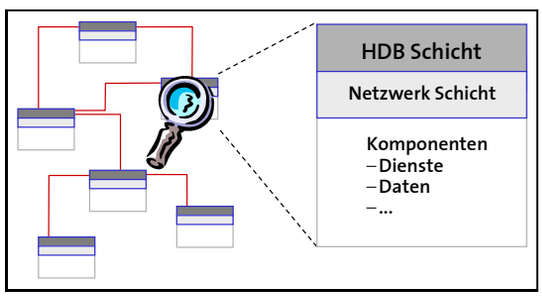

Bild 2 Hyperdatenbankschicht ergänzt Netzwerk.

Die Abarbeitung eines Prozesses benötigt eine Infrastruktur, welche das Auffinden und Ausführen von Diensten übernimmt. In der aktuellen .NET-Philosophie von Microsoft übernimmt beispielsweise der Programmierer diese Aufgabe [1]. Zur Programmerstellungszeit wählt er einen Dienst aus einem UDDI-Verzeichnis aus und fügt dessen Aufruf explizit in den Programmcode ein. Dieser Mechanismus ist aber zu wenig flexibel. Vielmehr wollen wir zur Laufzeit entscheiden können, welchen Dienstanbieter wir verwenden (z. B. aufgrund von Kriterien wie Kosten, Last oder erwarteter Antwortzeit), ohne dies jedes Mal selbst programmieren zu müssen. Mit anderen Worten, die Infrastruktur muss in der Lage sein, aufgrund einer Dienstbeschreibung und einer Menge von Auswahlkriterien zur Laufzeit einen (optimalen) Anbieter zu finden. Eine Hyperdatenbank erweitert deshalb die bekannten Netzwerkschichten um zusätzliche Schichten. Während die TCP/IP-Schicht für das korrekte Ausliefern von Bytes zuständig ist, übernimmt die darüber liegende Hyperdatenbankschicht das Auffinden (*service discovery*), das Auswählen (*load balancing*) und das Aufrufen (*routing*, *binding*) von Diensten. Anstelle eines starren Aufrufschemas wie in .NET erlaubt eine Hyperdatenbank einen dynamischen und optimierten Einsatz von Basisdiensten.

Natürlich stellt sich die Frage, wer für die Ausführung von Prozessen und die Einhaltung von Garantien verantwortlich ist. Dem Nutzer sollen die im Hintergrund ausgeführten Koordinationsprozesse verborgen bleiben. Dies gilt ebenfalls für die Ausführung von Prozessen, die systemgesteuert durch geschicktes Routing der Dienste optimiert werden. Auch diese Aufgaben übernimmt unsere Hyperdatenbankschicht. Dies ist daher – in Analogie zum .NET-Framework – eine Art Middleware.

Allerdings genügt diese Funktionalität noch bei Weitem nicht. Ein Benutzer möchte darüber hinaus gewisse Garantien für die Abarbeitung eines Prozesses haben, selbst im Falle von Fehlern und parallel ablaufenden Prozessen. Dies gilt im verstärkten Maße auch für Koordinationsprozesse, welche zum Beispiel einen Index aufgrund von Änderungen in den Basisinformationsquellen nachführen. Zur Lösung dieser Probleme verwenden wir Mechanismen, wie sie traditionell in Datenbanken eingesetzt werden, allerdings angewandt auf einer höheren Abstraktionsebene. Als Verallgemeinerung der traditionellen Atomarität in einer „Alles-oder-Nichts"-Semantik garantiert eine Hyperdatenbank, dass ein Prozess stets – im Fehlerfall mittels Kompensationsoperationen – in einem wohldefinierten Zustand endet. Ebenso beinhaltet sie eine Concurrency-Control-Komponente, die aber nicht auf der Basis von Daten (Tupel, Relation etc.), sondern auf

der semantischen Ebene der Dienste (unter Berücksichtigung ihrer Aufrufparameter) operiert.

Die Fehlerbehandlung ist ein weiterer wichtiger Aspekt einer Hyperdatenbank. Falls ein ausgewählter Dienstanbieter nicht mehr verfügbar ist, muss nach einem neuen Anbieter gesucht werden. Falls kein Anbieter verfügbar ist oder der Aufruf des Dienstes fehlschlägt, muss eine Fehlerbehandlung auf der semantischen Ebene möglich sein. Mit anderen Worten muss ein Prozess auch Alternativen kennen, die im Falle eines Fehlers ausgeführt werden. Falls keine Alternativen vorhanden sind, soll durch geeignete Kompensationsoperationen, die Teil der Prozessbeschreibung sind, eine geordnete Rücksetzung des Prozesses möglich sein. Ebenso muss bei einem Rechnerausfall garantiert werden, dass die Prozesse, die von diesem Rechner zur Zeit bearbeitet wurden, entweder auf einem anderen Rechner weitergeführt oder auf diesem Rechner fortgesetzt werden, sobald der Rechner wieder verfügbar ist. Auf keinen Fall sollen Benutzer- und Koordinationsprozesse von Hand neu gestartet werden müssen.

Eine Hyperdatenbank sollte möglichst dezentral organisiert sein. Prozesse sollen nicht auf einem einzigen Rechner ablaufen, sondern auf jedem Teilnehmer (*Peer*) des Informationsraumes. Damit kann man erreichen, dass die Ausführung der Prozesse näher bei den zugrunde liegenden Diensten und Informationen stattfindet. Eine zentrale Prozessabarbeitung würde zudem die Skalierbarkeit des Ansatzes in Frage stellen. In unserem Ansatz sehen wir deshalb jeden Peer als potentiellen Kandidaten für die Abarbeitung eines Prozesses. In einer optimierten Version würden diese aber nur jene Teile eines Prozesses abarbeiten, in denen lokale Dienste beansprucht werden. Falls in der Abarbeitung des Prozesses ein lokal nicht verfügbarer Dienst benötigt wird, so wird der Prozess auf den Rechner eines entsprechenden Anbieters dieses Dienstes migriert. Eine solche *Peer-to-Peer*-Abarbeitung ermöglicht eine nahezu beliebige Skalierung des Informationsraumes, vorausgesetzt, dass die Anzahl der Konflikte zwischen parallel ablaufenden Prozessen nicht zu groß ist. Für die Realisierung einer verteilten Concurrency Control bedarf es einer Replikation von Metadaten, die zur Synchronisation notwendig sind. Durch eine geschickte Replikation des Serialisierungsgraphen kann die Korrektheit (Isolation) von Prozessausführungen vollständig lokal überprüft werden. Details über die konkrete Umsetzung einer verteilten Concurrency Control ohne zentrale Koordination können der Arbeit [21] entnommen werden. Letztlich benötigen wir globale Meta-Informationen und Meta-Dienste, die Informationen über die Dienste und Rechner des Informationsraumes sammeln, verwalten und zur Verfügung stellen. Zum Beispiel muss die Hyperdatenbank wissen, welche Rechner welche Dienste anbieten, wie teuer diese Dienste sind und wie stark die Rechner ausgelastet sind. Ebenso benötigen wir Dienste, welche unter anderem Prozessbeschreibungen speichern, Prozesse instanziieren, Synchronisation, Recovery und Concurrency Control übernehmen und laufende Prozesse überwachen.

Dieses globale Meta-Wissen muss nicht zwangsläufig in allen Peers der Hyperdatenbank in seiner Gesamtheit verfügbar sein. Dies würde zu hohen Replikationskosten führen. Stattdessen erhalten die Peers lediglich Replikate der globalen Metadaten, die notwendig sind, um die für sie relevanten Teile von Prozessen abzuarbeiten. Ferner müssen einige dieser Replikate nicht unbedingt synchron aktualisiert werden. So muss zum Beispiel ein Peer nicht alle Anbieter eines Dienstes kennen; es genügt, wenn er eine genügend große Auswahl hat, um ein (sub-)optimales Routing vornehmen zu können.

Zusammengefasst stellt eine Hyperdatenbank ähnliche Schnittstellen und Dienste zur Verfügung wie eine klassische Datenbank, allerdings auf einer semantisch höheren Ebene. Anstelle direkt auf den Daten mittels SQL zu operieren, verwendet die Hyperdatenbank Dienstaufrufe zur Anfrage und Modifikation von Informationen. Ein Prozess in der Hyperdatenbank entspricht einer Transaktion in einer Datenbank. Die Kontrolle der Nebenläufigkeit und Atomarität von Prozessen entspricht der Concurrency Control und Recovery in einer Datenbank. Dienste zur Merkmalsextraktion und die Organisation der Merkmale in Suchdiensten entsprechen der internen Indexkomponente einer Datenbank. Die geschickte Ausführung von (Such-)Prozessen entspricht der Anfrageoptimierung in Datenbanken.

Alternative Ansätze. Ein Informationsraum kann auf verschiedene Arten umgesetzt werden. Im Folgenden skizzieren wir alternative Ansätze, welche ähnliche Zielsetzungen haben wie unser Hyperdatenbankansatz.

Prozesse und Workflows [12] spielen eine zentrale Rolle im Informationsraum. Die traditionelle Vorgehensweise bei der Implementierung eines verteilten Workflow-Systems beruht auf der Dezentralisierung der Dienste sowie der zugrunde liegenden Daten und auf einem oder mehreren zentralen Koordinationsdiensten (*Workflow-* bzw. *Process-Engine*). Derartige Architekturen haben den Nachteil, dass solche Koordinationsdienste zu einem Engpass werden können, wenn eine hinreichend große Menge von parallelen Prozessen zu bearbeiten ist. Da die proportionale Zunahme der Rechenleistung im Vergleich zu der Zunahme der Netzwerkgeschwindigkeit auch in absehbarer Zukunft geringer ausfallen wird, ist die maximale Anzahl von parallelen Prozessen und damit die Skalierbarkeit des Systems primär durch die Rechenleistung des zentralen Koordinationsdienstes beschränkt.

Grid-Infrastrukturen [7–9] unterstützen die explizite Verwaltung

verteilter Daten und Rechenressourcen, um berechnungsintensive Prozesse effizient durchführen zu können. Zunehmend werden auch Dienste in das Grid hineingenommen; man spricht in diesem Zusammenhang häufig von einem *Service Grid*. Die Grid-Architektur basiert ähnlich wie der Hyperdatenbankansatz auf einer Menge von Verzeichnissen, welche globale Metadaten über die Dienste und Daten sammeln. Aufrufe werden dann durch diese Verzeichnisse an die entsprechenden Knoten weitergeleitet. Das Problem bei diesem Ansatz ist die statische Verwaltung der Metadaten.

Peer-to-Peer-Infrastrukturen [4; 14] sehen keine zentralen Komponenten vor. Die Kommunikation zwischen den einzelnen Peers (Komponenten, Dienstanbietern) erfolgt vollständig horizontal, das heißt direkt. Peer-to-Peer-Systeme zeichnen sich durch ihre hochgradige Skalierbarkeit und den hohen Grad an lokaler Autonomie der Peers aus. Im Gegensatz zu den traditionellen Einsatzgebieten von Peer-to-Peer-Systemen, etwa dem File-Sharing wie in Gnutella [10], benötigt ein Hyperdatenbanksystem eine konsistente und vollständige Sicht auf den Informationsraum, da dort der Aspekt der globalen Koordination im Vordergrund steht. Traditionelle Peers hingegen tolerieren Inkonsistenzen, da sie ohnehin keine vollständige Sicht auf die globalen Metadaten haben.

Föderierte Datenbanken [3] bleiben auf der Datenebene und erzielen durch Schemaintegration und durch globale und lokale Transaktionen eine einheitliche Datenbanksicht auf die Daten in den Komponenten-Datenbanken. Im Hyperdatenbankansatz dagegen abstrahieren wir von den Details der Daten und streben keine Schemaintegration im Großen an. Benötigte Datentransformationen werden über Dienste erledigt. Entscheidend kommt aber die gleichberechtigte Behandlung von Berechnungsdiensten hinzu, wie wir es bereits früher im Zusammenhang mit externen Objekten in föderierten Datenbanken gefordert haben [20].

Aus unserer Sicht greift jede dieser Architekturen das Problem der Verwaltung eines Informationsraumes von einer anderen Seite her an, zum Teil mit unterschiedlichen Schwerpunkten. Die Hyperdatenbank, als Abgrenzung zu den oben aufgeführten Ansätzen, bietet eine allgemeinere Infrastruktur, die auf die Probleme der Verwaltung eines Informationsraumes zugeschnitten ist und dabei Aspekte der vorgestellten Ansätze kombiniert. Im folgenden Abschnitt beschreiben wir mit OSIRIS einen Prototyp eines Hyperdatenbanksystems, das in der Datenbankgruppe der ETH Zürich entwickelt wird und das den zuvor aufgelisteten Anforderungen genügt.

3 Der OSIRIS-Prototyp

OSIRIS (*Open Service Infrastructure for Reliable and Integrated process Support*) [19] besteht, wie in Bild 3 dargestellt, im Wesentlichen aus der HDB-Schicht bei jedem Dienstanbieter (Komponente), der einen oder mehrere Dienste im Informationsraum anbietet. Zusätzlich zu den lokalen HDB-Schichten gibt es eine Menge von globalen Diensten oder Repositories. Diese verwalten die globalen Metadaten über die Systemkonfiguration und die Dienste und Prozesse des Informationsraumes.

In den folgenden Abschnitten skizzieren wir die Implementierung der wesentlichen Komponenten von OSIRIS.

3.1 Prozessmanagement

Die Prozessausführung wird in einem „Peer-to-Peer-Stil“ durchgeführt. Die Metadaten über die Systemkonfiguration und die Dienste des Informationsraumes werden so repliziert, dass die lokalen HDB-Schichten nur die für sie relevanten Metadaten erhalten. Eine derartige Replikation ermöglicht im Gegensatz zu traditionellen Workflow-Ansätzen eine vollständig lokale Ausführung von globalen Prozessen. Jede lokale Hyperdatenbankschicht kann selbstständig den nächsten Prozessschritt ausführen, sobald sie den geforderten Dienstaufruf abgearbeitet hat.

Voraussetzung hierfür ist allerdings eine vollständige Spezifikation der globalen Prozesse. Dies ist gewöhnlich die Aufgabe des Administrators des Informationsraumes. OSIRIS unterstützt ihn dabei mit dem OSIRIS-spezifischen grafischen

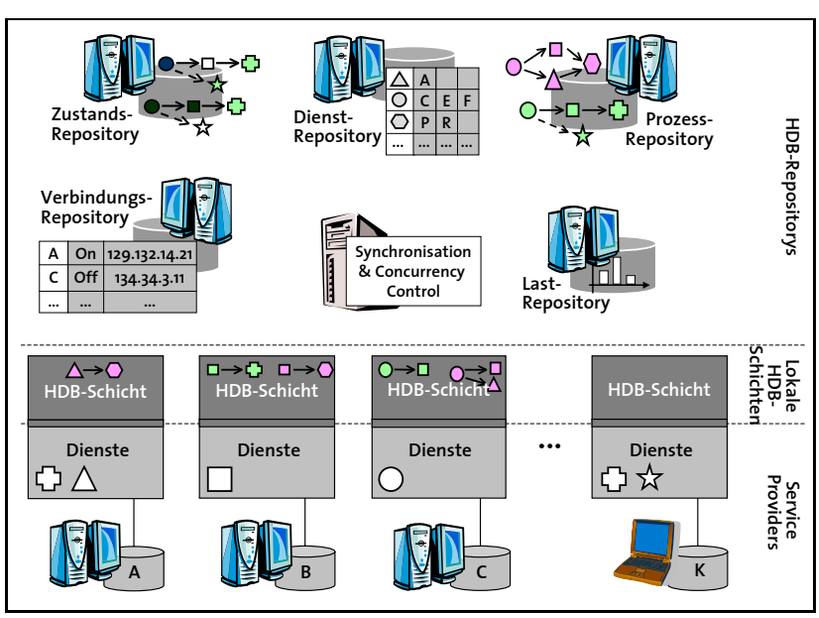

Bild 3 Die Architektur von OSIRIS.

Prozessmodellierungstool O'Grape (*OSIRIS Graphical Process Editor*). Die Prozessspezifikation wird in einem semistrukturierten Format im globalen *Prozess-Repository* (PR) abgelegt. Die Information, welche Dienste von welchen Dienstanbietern bereitgestellt werden, ist im *Dienst-Repository* (DR) abgespeichert. Jeder Dienstanbieter muss die Dienste, die er bereitstellen möchte, im DR registrieren lassen. Die Repositories PR und DR dienen später der partiellen, kontextbasierten Replikation der Metadaten, nicht jedoch der direkten Prozessausführung.

Nachdem ein Dienstanbieter p einen Dienst s_i registriert hat, erhält er die Ausschnitte der Prozessbeschreibungen, in denen s_i beteiligt ist. Dieser Ausschnitt enthält insbesondere Paare von Diensten (s_i, s_k), die Auskunft darüber geben, welcher Dienst s_k nach der Ausführung eines Dienstes s_i aufzurufen ist. Zu jedem Dienst s_k erhält der Dienstanbieter p eine Liste von Dienstanbietern, die diesen Dienst bereitstellen. Die Auswahl eines Dienstanbieters zur Laufzeit ermöglicht eine dynamische Prozessausführung, die unter anderem eine Lastbalancierung verwirklichen kann. Damit lassen sich bestimmte Kriterien wie zum Beispiel eine möglichst optimale und faire Auslastung aller Dienstanbieter innerhalb des Informationsraumes realisieren (siehe Abschnitt 3.3).

Die Prozessausführung selbst findet auf der Ebene der lokalen Hyperdatenbankschichten statt. Ein Prozess wird durch den Aufruf des ersten Dienstes initiiert. Nach dessen Ausführung durch den Dienstanbieter ruft die korrespondierende lokale Hyperdatenbankschicht den nächsten Dienst (gemäß Prozessbeschreibung) auf und gibt damit die Kontrolle über den Prozess an die jeweilige lokale Hyperdatenbankschicht des Anbieters des nächsten Dienstes weiter. Dieser Mechanismus wird bis zur erfolgreichen Terminierung des Prozesses schrittweise fortgesetzt.

3.2 Metadatenverwaltung

Die Kommunikation zwischen den einzelnen Schichten in OSIRIS beruht einzig und allein auf einem generischen *Publish/Subscribe*-Prinzip [5]. Aus konzeptioneller Sicht abonnieren sich Dienstanbieter für den Aufruf der Dienste, die sie anbieten. Jede lokale Hyperdatenbankschicht generiert (*publish*) ein Ereignis, nachdem sie einen Dienst s_i ausgeführt hat. Dieses Ereignis triggert lokal den Aufruf des in der Prozessbeschreibung nachfolgenden Dienstes s_k und gibt damit die Kontrolle an die lokale HDB-Schicht eines Dienstanbieters weiter, der sich zuvor für die Ausführung eines Dienstes s_k registriert (*subscribe*) hatte. Das Besondere an der OSIRIS-Umsetzung ist, dass das Matching zwischen den *Publications* und den *Subscriptions* nicht durch einen zentralen Publish/Subscribe-Broker durchgeführt wird, sondern vollständig lokal durch die jeweilige lokale Hyperdatenbankschicht.

Die Replikation der Metadaten ermöglicht die verteilte und dezentralisierte Implementierung der Publish/Subscribe-Funktionalität. Jede lokale Hyperdatenbankschicht wird mit einem Publish/Subscribe-Broker ausgestattet, sodass Ereignisse mithilfe der PR- und DR-Replikate lokal verarbeitet werden können. Die Replikate werden wiederum selbst mithilfe des Publish/Subscribe-Mechanismus konsistent gehalten.

Beim Registrieren eines Dienstes s_i werden implizit Subscriptions auf Änderungen von Metadaten, die diesen Dienst betreffen, generiert. Folgende Änderungen kommen hierfür in Frage:

(1) Spezifikation eines neuen Prozesses, der s_i in einem Schritt ausführt.
(2) Änderung der Beschreibung eines existierenden Prozesses, sodass der Dienst, der s_i folgt, entweder entfernt oder ersetzt wird.
(3) Registrierung eines neuen Anbieters, der einen Dienst anbietet, welcher in einem Prozess nach dem Dienst s_i aufgerufen werden muss.
(4) Entfernen von Diensten.

In allen Fällen müssen die Änderungen den globalen Repositories gemeldet werden, wo dann ein Ereignis erzeugt wird, welches diese Änderungen an alle betroffenen lokalen Hyperdatenbankschichten propagiert. Diese Technik be-

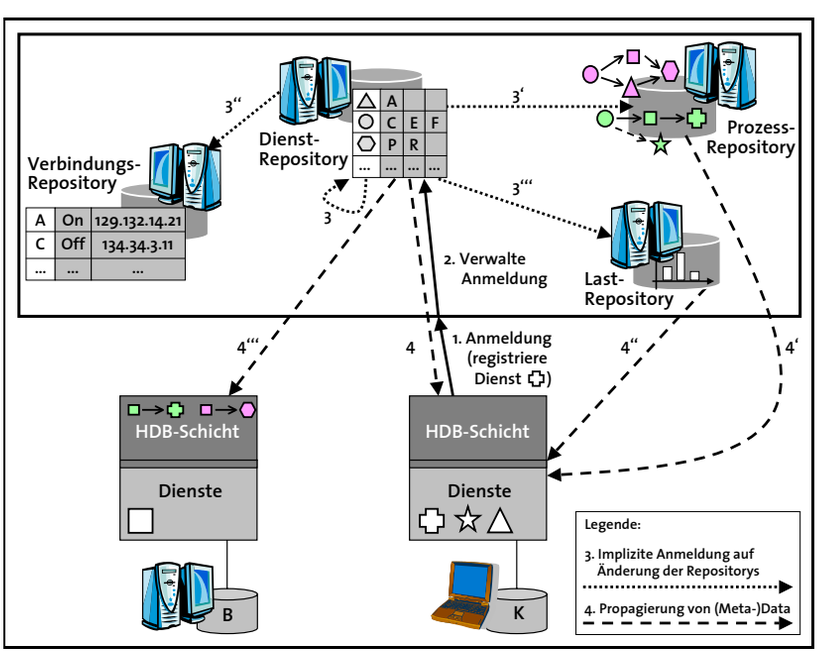

Bild 4 Replikation von Metadaten in OSIRIS.

zeichnen wir als DIPS (*Doubled Implicit Publish/Subscribe*). Bild 4 verdeutlicht die Replikation von Metadaten in OSIRIS mithilfe der DIPS-Technik.

Je nach Art der Metadaten spielt ihre Konsistenz eine mehr oder weniger wichtige Rolle. Die Konsistenz der Prozessausführung ist zum Beispiel weder durch die Approximationen der Auslastungsdaten noch durch Inkonsistenzen bei den Dienstanbieterdaten (wenn zum Beispiel die lokal replizierten Listen der Anbieter eines Dienstes nicht ganz vollständig sind) gefährdet.

3.3 Lastbalancierung

Nach dem Abarbeiten eines Dienstaufrufs muss die lokale Hyperdatenbankschicht einen Dienstanbieter auswählen, der den nächsten Dienst ausführen soll. Bei der Auswahl kann die Auslastung der jeweiligen Dienstanbieter herangezogen werden, um die globale Effizienz durch eine Lastbalancierung zu steigern. Eine Lastbalancierung erfordert das lokale Vorhandensein von Daten über die ungefähre (approximative) Auslastung der relevanten Dienstanbieter. Basierend auf diesen Daten wird jeweils der Anbieter mit der geringsten Last ausgewählt. Voraussetzung dafür ist, dass mehrere Dienstanbieter denselben bzw. einen semantisch äquivalenten Dienst bereitstellen.

Die Verteilung der Auslastungsdaten geschieht ebenfalls mit dem Publish/Subscribe-Prinzip. Die ursprüngliche, explizite Subscription generiert nicht nur implizite Subscriptions auf Änderungen der Repositories PR und DR, sondern auch auf Laständerungen der jeweiligen Dienstanbieter. Die Auslastungsdaten werden global im *Last-Repository* (LR) verwaltet (vgl. Bild 4). Im Gegensatz zu den Replikaten der Prozessbeschreibungen und der Dienstbeschreibungsdaten, die konsistent gehalten werden müssen, genügt es bei den Auslastungsdaten, dass die lokalen Replikate den tatsächlichen Auslastungsgrad approximieren. Mit anderen Worten werden in diesem Fall nicht alle geringfügigen Änderungen der lokalen Auslastung publiziert. Erst wenn eine vorgegebene Schranke über- bzw. unterschritten wurde, wird dies an das Last-Repository und in der Folge an die Subscribers weitergeleitet.

3.4 Fehlerbehandlung

OSIRIS unterstützt transaktionale Ausführungsgarantien, die auf den Ideen der transaktionalen Prozessverwaltung beruhen [17]. Der hier verwendete Begriff der Atomarität („garantierte Terminierung“) verallgemeinert das traditionelle „Alles-oder-Nichts-Prinzip“ von Datenbanktransaktionen. Es wird garantiert, dass alle transaktionalen Prozesse zu einem wohldefinierten Ende gelangen. Hierzu werden im Fehlerfall entweder Alternativ- oder Kompensationsdienste ausgeführt, die bei der Spezifikation der Koordinationsprozesse festgelegt werden.

Für die Umsetzung der Fehlerbehandlung in OSIRIS sind mehrere Fehlerarten zu unterscheiden. Tritt ein Fehler auf, wenn die HDB-Schicht einer Komponente einen Dienst aufrufen möchte, der gerade nicht verfügbar ist, dann kann einfach mithilfe der lokal in der HDB-Schicht des Aufrufers replizierten Metainformation ein anderer Anbieter (in diesem Fall mit der nächsthöheren Last bzw. einer größeren erwarteten Antwortzeit) ausgewählt werden. Gleichzeitig wird, um zu verhindern, dass der nicht verfügbare Dienst auch von anderen Komponenten vergeblich aufgerufen wird, eine Nachricht an das Dienst-Repository geschickt. Dies veranlasst das Austragen des nicht verfügbaren Dienstes und bewirkt mittels des Publish/Subscribe-Prinzips die Aktualisierung dieser Information in allen lokalen Replikaten. Auf diese Weise kann flexibel auf die Nicht-Verfügbarkeit mehrerer Dienste reagiert werden, indem bei Bedarf jeweils wiederholt auf den nächsten lokal bekannten Anbieter zurückgegriffen wird.

Sind jedoch alle diese lokal bekannten Anbieter eines Dienstes nicht verfügbar, so muss die im Prozess festgelegte Fehlerbehandlung angewandt werden. Je nach Vorhandensein von alternativen Ausführungspfaden bedeutet dies, dass sofort auf eine solche alternative Ausführung umgeschaltet wird, dass einige bereits ausgeführte Dienste kompensiert werden müssen, um an eine Stelle zu gelangen, von der aus Alternativen möglich sind, oder dass sämtliche Dienste des Prozesses kompensiert werden müssen.

Schließlich ist zu verhindern, dass Prozesse durch Systemfehler der Komponente, die gerade für die Ausführung eines Dienstes dieses Prozesses zuständig ist, abbrechen und damit nicht fortgesetzt werden können. Zu diesem Zweck wird jeder Dienstaufruf in eine verteilte Transaktion zwischen Aufrufer und Aufgerufenem eingebettet. Dies garantiert, durch das atomare Commit, die korrekte Übertragung eines Aufrufs. Mit der Annahme eines Aufrufs übernimmt die lokale HDB-Schicht einer Komponente gleichzeitig die weitere Kontrolle der Prozessausführung. Hierzu wird, im Sinne der *Queued Transactions* [2], der Aufruf persistent in der lokalen HDB-Schicht gespeichert und erst innerhalb der verteilten Transaktion, die beim Aufruf des nächsten Dienstes gestartet wird, wieder ausgetragen. Daher kann eine fehlerhafte Komponente nach dem Neustart den Zustand vor dem Systemfehler ermitteln und falls nötig die Prozessausführung fortsetzen.

3.5 Umsetzung in ETHWorld

Eine konkrete Anwendung von OSIRIS erfolgt im Kontext der Koordination von Multimedia-Komponenten des bereits in der Einleitung eingeführten ETHWorld-Projektes. Ein Teilprojekt von ETHWorld beschäftigt sich mit der Ähnlichkeitssuche in multimedialen Dokumenten. Das ISIS-System (*Interactive SImilarity Search*) [11] bietet effektive Deskriptoren für die verschiedenen Arten von Dokumenten, effizi-

ente Suchmethoden für eine komplexe Ähnlichkeitssuche [22] und benutzerfreundliche Relevanzrückkopplungsmechanismen für die Anfragereformulierung.

Im Gegensatz zu gewöhnlichen Web-Suchmaschinen, welche die Daten nur periodisch alle paar Wochen aktualisieren, wird für die ISIS-Suchmaschine mittels des OSIRIS-Systems ein *Up-to-date*-Index verwaltet, der Änderungen des Informationsraumes unmittelbar (so schnell wie möglich) reflektiert. OSIRIS überwacht lokale Änderungen im Dokumentenbestand und triggert entsprechende Koordinationsprozesse von ISIS, um bei Änderungen den Suchindex zu aktualisieren. In Bild 1 hatten wir bereits den Koordinationsprozess gesehen, den der Aufruf des Dienstes *InsertImage* triggert. Hinter dem Koordinationsprozess verbergen sich die aufwändigen Schritte, die mit dem Einfügen eines Bildes verbunden sind, etwa die Extraktion verschiedener Merkmale wie Textur oder Farbverteilung.

4 Zusammenfassung und Ausblick

Das Konzept einer Hyperdatenbank ist unsere Antwort auf die Anforderungen einer rasant wachsenden und sich ändernden Informationslandschaft. Eine Hyperdatenbank sieht eine flexible Infrastruktur vor, die Datenbankfunktionalität auf einer höheren Ebene – in Informationsräumen – unterstützt. Zentral ist dabei die konsistente Koordination der Dienste des Informationsraumes. Eine Hyperdatenbank kombiniert Aspekte von Workflows (Prozessunterstützung mit transaktionalen Ausführungsgarantien), Grid Computing (Ressourcen-Management für berechnungsintensive Dienste), und Peer-to-Peer Computing (direkte Interaktion von Peers, das heißt Dienstanbietern), um eine skalierbare Infrastruktur bereitzustellen, die den ständig wachsenden Anforderungen gerecht wird.

OSIRIS ist ein erster Prototyp, der das Konzept einer Hyperdatenbank realisiert. Globale Prozesse werden vollständig dezentral gemäß dem Peer-to-Peer-Ansatz ausgeführt, ohne dabei auf Ausführungsgarantien zu verzichten. Die Kommunikation zwischen den einzelnen Peers erfolgt ausschließlich über einen einzigen generischen Mechanismus, dem Publish/Subscribe-Prinzip. Dieses Prinzip wird unter anderem für die Synchronisation der Dienste, die Verteilung der Metadaten und die Lastbalancierung genutzt.

Aktuelle Arbeiten beschäftigen sich mit der Erweiterung des OSIRIS-Prototyps in Richtung Ad-hoc-Prozesse und mobile Dienstanbieter [21]. Ad-hoc-Prozesse folgen keiner existierenden Prozessbeschreibung. Sie werden einmalig, dynamisch instanziiert. Basierend auf Techniken aus dem Bereich der mobilen Agenten sollen Ad-hoc-Prozesse unter transaktionalen Garantien ausgeführt werden können, ohne dabei auf eine zentrale Kontrollinstanz zurückgreifen zu müssen.

Mobile Dienstanbieter sind nicht ständig verfügbar. Sie können sich dynamisch bei Bedarf an das Netz ankoppeln und entsprechend wieder entkoppeln. Sie können im entkoppelten Zustand weiter arbeiten. Zwar kann das Ent- bzw. Ankoppeln von Dienstanbietern mit den skizzierten Mechanismen umgesetzt werden, hat aber den Nachteil, dass nicht zwischen einem beabsichtigten und einem unkontrollierten Abmelden unterschieden wird. Dieses Wissen kann helfen, die Prozessausführung besser zu steuern, zum Beispiel, wenn man weiß, dass ein Dienst bearbeitet wird, der Dienstanbieter aber offline ist. Deshalb ist ein Verbindungs-Repository notwendig, das die Verbindungsinformationen explizit verwaltet und sie sauber von den Informationen über die bereitgestellten Dienste trennt. Das Ent- und Ankoppeln kann dann als Koordinationsprozess mit dem bereits skizzierten DIPS-Mechanismus umgesetzt werden.

In Zukunft werden wir verstärkt in Kooperation mit der privaten Universität für medizinische Informatik und Technik Tirol (UMIT) in Innsbruck den Einsatz von Hyperdatenbanken im medizinischen Umfeld untersuchen.

Literatur

[1] Ariba, IBM, and Microsoft. UDDI Technical White Paper. *http://www.uddi.org*.

[2] P. Bernstein, E. Newcomer. Principles of Transaction Processing. Morgan Kaufmann Publishers, 1997.

[3] S. Conrad. Föderierte Datenbanksysteme: Konzepte der Datenintegration. Springer-Verlag, 1997.

[4] M.G. Curley. Peer-to-Peer Computing Enabled Collaboration. *Int'l Conf. on Computational Science, ICCS 2002*, pp. 646–654, 2002.

[5] P. Eugster, P. Felber, R. Guerraoui, A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Comput. Surveys*, 2003.

[6] ETHWorld – The Virtual Campus of ETH Zürich. *http://www.ethworld.ethz.ch*.

[7] I. Foster, C. Kesselmann, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int'l Journal of Supercomputer Applications*, 15(3):123–130, 2001.

[8] I. Foster, C. Kesselmann, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Global Grid Forum, 2002. *http://www.gridforum.org/ogsi-wg*.

[9] W. Gentzsch. Grid Computing: A New Technology for the Advanced Web. In *Advanced Environments, Tools, and Applications for Cluster Computing, NATO Advanced Research Workshop, IWCC 2001*, pp. 1–15, 2002.

[10] Gnutella RFC. *http://rfc-gnutella.sourceforge.net*

[11] ISIS – Interactive SImilarity Search. *http://www.isis.ethz.ch*.

[12] D. Hollingsworth. Workflow Management Coalition: The Workflow Reference Model. Document TC00-1003, 1995. *http://www.wfmc.org*.

[13] R. Koster, A. Black, J. Huang, J. Walpole, C. Pu. Infopipes for Composing Distributed Information Flows. *Int'l*

Workshop on Multimedia Middleware, 2001.

[14] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu. Peer-to-Peer Computing. TR, HPL-2002-57, HP Laboratories Palo Alto, March 2002.

[15] H.-J. Schek, K. Böhm, T. Grabs, U. Röhm, H. Schuldt, R. Weber. Hyperdatabases. *1st Int'l Conf. on Web Information Systems Engineering*, pp. 14–23, 2000.

[16] H.-J. Schek, H. Schuldt, R. Weber. Hyperdatabases – Infrastructure for the Information Space. *6th IFIP Working Conf. on Visual Database Systems*, pp. 1–15, 2002.

[17] H. Schuldt, G. Alonso, C. Beeri, H.-J. Schek. Atomicity and Isolation for Transactional Processes. *ACM Trans. on Database Systems*, 27(1):63–116, March 2002.

[18] H. Schuldt, G. Alonso, H.-J. Schek. Concurrency Control and Recovery in Transactional Process Management. *18th ACM Symp. on Principles of Database Systems, PODS'99*, pp. 316–326, 1999.

[19] C. Schuler, H. Schuldt, H.-J. Schek. Supporting Reliable Transactional Business Processes by Publish/Subscribe Techniques. *2nd Int'l Workshop on Technologies for E–Services, TES'01*, pp. 118–131, 2001.

[20] H.-J. Schek, G. Weikum. Erweiterbarkeit, Kooperation, Föderation von Datenbanksystemen. *GI-Fachtagung BTW'91*, S. 38–71, 1991.

[21] C. Türker, K. Haller, H. Schuldt, H.-J. Schek. Mobilität in Informationsräumen. In *Datenbank-Spektrum*, Heft 5, S. 16–23, 2003.

[22] R. Weber, H.-J. Schek, S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. *24th Int'l Conf. on Very Large Data Bases, VLDB'98*, pp. 194–205, 1998.

(v.l.n.r.)
oben: H. Schuldt, R. Weber, H.-J. Schek
unten: C. Schuler, C. Türker

Prof. Dr.-Ing. Hans-Jörg Schek ist seit 1988 Professor für Informatik an der ETH Zürich und seit Oktober 2002 auch o. Univ.-Professor an der Privaten Universität für Medizinische Informatik und Technik Tirol (UMIT). Er hat ein Diplom in Mathematik (Universität Stuttgart, 1968) und promovierte (1972) und habilitierte (1978) an der Universität Stuttgart im Ingenieurbereich. Die Stationen seiner beruflichen Laufbahn sind das Wissenschaftliche Zentrum der IBM in Heidelberg (1972–1983), C4-Professor an der TH Darmstadt (1983–1988) und Forschungsaufenthalte am IBM Almaden Forschungslabor (1987), am ICSI Berkeley (1995) und an der Universität Heidelberg, Medizinische Informatik (1995). Hans-Jörg Schek wurde 2001 zum ACM Fellow ernannt.
Adresse: siehe unten

Prof. Dr. sc.techn. Heiko Schuldt ist seit Juli 2003 Professor für Informatik an der Privaten Universität für Medizinische Informatik und Technik Tirol (UMIT) in Innsbruck und leitet dort die Abteilung Information & Software Engineering. Er studierte Informatik an der Universität Karlsruhe (TH), wo er im Juli 1996 sein Diplom erwarb. Danach war er wissenschaftlicher Mitarbeiter in der Gruppe von Prof. Dr. H.-J. Schek an der ETH Zürich, wo er im Dezember 2000 promovierte. Von Januar 2001 bis Juni 2003 war er dort als Oberassistent tätig.
Adresse: UMIT Innsbruck, Abteilung Information & Software Engineering, Innrain 98, 6020 Innsbruck, Österreich, E-Mail: heiko.schuldt@umit.at

Dipl. Inform.-Ing. Christoph Schuler studierte Informatik an der ETH Zürich, wo er 1998 sein Diplom erwarb. Seit September 1998 ist er wissenschaftlicher Mitarbeiter in der Gruppe von Prof. Dr. H.-J. Schek.
Adresse: siehe unten

Dr.-Ing. Can Türker studierte Informatik an der TU Darmstadt. Nach seinem Diplomabschluss im September 1994 war er wissenschaftlicher Mitarbeiter in der Datenbankgruppe von Prof. Dr. G. Saake an der Universität Magdeburg, wo er im Oktober 1999 promovierte. Seit November 1999 ist er Oberassistent in der Gruppe von Prof. Dr. H.-J. Schek.
Adresse: siehe unten

Dr. sc.techn. Roger Weber studierte Informatik an der ETH Zürich, wo er im September 1996 sein Diplom erwarb. Seitdem arbeitet er in der Gruppe von Prof. Dr. H.-J. Schek. Zunächst als wissenschaftlicher Mitarbeiter, nach seiner Promotion im Dezember 2000 als Oberassistent.
Adresse: ETH Zürich, Institut für Informationssysteme, ETH Zentrum, 8092 Zürich, Schweiz, E-Mail: {schek,schuler,tuerker,weber}@inf.ethz.ch