Can Türker, Klaus Haller, Heiko Schuldt, Hans-Jörg Schek

Mobilität in Informationsräumen

In diesem Beitrag zeigen wir auf, wie unsere Vision einer Hyperdatenbankinfrastruktur für die dezentrale "peer-to-peer" Ausführung von Anwendungen in Form von vordefinierten Prozessen erweitert werden kann, wenn wir zum einen mobile Dienstanbieter unterstützen und wenn wir zum anderen Ad-hoc-Prozesse zulassen. Es handelt sich nicht um einen Erfahrungs- oder Praxisbericht, sondern um die Beschreibung zweier laufender Forschungsrichtungen an der ETH Zürich, von denen wir glauben, dass sie beide wichtig sind und dass mehr Forschung dazu nötig ist. Das Fernziel ist die Zusammenführung dieser Richtungen in eine möglichst einfache Infrastruktur, in der vordefinierte und Ad-hoc-Prozesse mobile und stationäre Dienste kombinieren und zuverlässig verarbeiten.

1 Einleitung

Ein *Informationsraum* besteht aus Kollektionen von Daten, Dokumenten und Diensten. Der Zugriff auf die Daten bzw. Dokumente des Informationsraumes ist dem Nutzer eines Informationsraumes nur über die angebotenen Dienste möglich. Anwendungen innerhalb des Informationsraumes sind in Form von Prozessen realisiert, die mehrere Aufrufe von Diensten zusammenfassen.

Die Nutzer eines Informationsraumes haben hohe Anforderungen in Bezug auf die Verfügbarkeit der Dienste, durch die der Zugriff auf und die Verarbeitung von Information erfolgt. Gerade bei der Verwendung von mobilen Servern bzw. beim Zugriff auf Sensordaten kann jedoch der Fall eintreten, dass ein gewünschter Dienstanbieter temporär nicht verfügbar ist. Da neben einem temporären Abmelden auch eine Änderung der Verbindungsparameter erfolgen kann (ein Dienstanbieter meldet sich ab und nimmt später wieder von einem anderen Ort am Informationsraum teil), muss die Infrastruktur geeignete Mechanismen vorsehen, um mit diesem Problem geeignet und effizient umgehen zu können. Das bedeutet insbesondere, dass der aktuelle Aufenthaltsort solcher Geräte den Benutzern bzw. Clients gegenüber transparent behandelt werden muss.

Mobile Clients erlauben ein großes Maß an Flexibilität für die jeweiligen Benutzer. Information wird unabhängig vom Aufenthaltsort jederzeit zugreifbar und verwertbar. Dies hat auch Konsequenzen für die Anwendungen in Informationsräumen. Eine wesentliche Beobachtung ist, dass die Anwendungen sich nicht ausschließlich auf vordefinierte Prozessmuster beschränken, sondern vermehrt auch aus individuellen, benutzerspezifischen Kombinationen von Dienstaufrufen bestehen. Diese Individualität hat zur Folge, dass Prozesse in der Regel nur einmal (oder einige wenige Male) instanziert werden. Solche Prozesse bezeichnen wir als *Ad-hoc-Prozesse*.

Traditionelle Infrastrukturen, gerade zur Prozessverwaltung, sind darauf eingerichtet, standardisierte Abläufe mit hoher Wiederholfrequenz ausführen zu können. Dies hat zur Folge, dass zur Optimierung dieser Abläufe einige Meta-Informationen zu diesen verteilten Anwendungen im Netzwerk repliziert werden können. Die Entwicklungen hin zu mobilen Geräten und individuellen Client-Lösungen haben aber auch zur Folge, dass neben diesen etablierten Anwendungen und Abläufen vermehrt die oben genannten Ad-hoc-Anwendungen vom System zu unterstützen sind. Gerade hier besteht jetzt nicht mehr die Möglichkeit, den Informationsraum auf solche Prozesse hin zu konfigurieren.

Ferner muss auch hier die Mobilität der Nutzer, die diese Ad-hoc-Prozesse starten, unterstützt werden. Daher darf die Ausführung von Ad-hoc-Prozessen nicht komplett auf einem Client-Rechner bzw. einem (mobilen) Gerät des Nutzers erfolgen, zum Beispiel indem von dort aus die einzelnen Dienste per Remote Procedure Call (RPC) aufgerufen werden. Vielmehr muss nach dem Start des Prozesses der Nutzer offline gehen können ohne dadurch die Prozessausführung zu unterbrechen. Des weiteren soll die Möglichkeit bestehen, solche Ad-hoc-Prozesse von jeder beliebigen Komponente des Informationsraumes aus zu starten. Insbesondere soll aufgrund der Autonomie der Nutzer kein globaler Prozesskoordinator eingeführt werden. Dies hat zur Folge, dass alle Ad-hoc-Prozesse selbstbeschreibend sein müssen, indem sie sowohl ihre Prozessdefinition als auch ihren Zustand kapseln. Während der Prozessausführung werden Definition und Zustand zu den entsprechenden Dienstanbietern, gemäss der Spezifikation in der Prozessdefinition, transferiert. Dort erfolgt lokal ein Dienstaufruf, der Prozesszustand wird aktualisiert und der Ad-hoc-Prozess wandert weiter zum nächsten Dienstanbieter. Die Lösung des Problems der notwendigen Entkopplung des Ad-hoc-Prozesses von seinem Nutzer und des Vermeidens bzw. Fehlens eines globalen Koordinators liegt also in der Mobilität der Anwendungen im Informationsraum.

Wie die vordefinierten Prozesse können Ad-hoc-Prozesse weiterhin konkurrierend gemeinsame Dienste nutzen. Dies erfordert erweiterte Mechanismen, die trotz Fehlens eines globalen Koordinators eine korrekte Fehlerbehandlung von Ad-hoc-Anwendungen ermöglichen und die eine Synchronisationsfunktionalität bereitstellen, mit der der nebenläufige Zugriff auf gemeinsame Ressourcen kontrolliert werden kann.

Das nachfolgende Kapitel 2 gibt einen kurzen Überblick über die Vision und Realisierung einer Hyperdatenbank für vordefinierte Prozesse. Kapitel 3 geht auf die Konsequenzen der Mobilität von Dienstanbietern ein und skizziert, wie das explizite Verwalten von Verbindungen und temporären Entkopplungen in dieser Infrastruktur möglich ist. In Kapitel 4 wird die Realisierung von Ad-hoc-Anwendungen in der Hyperdatenbankinfrastruktur betrachtet. Kapitel 5 fasst beide Aspekte der Mobilität für die Verwaltung und Verarbeitung von Information in einem Informationsraum zusammen.

2 Verwaltung von Informationsräumen

In unserer Vision stellt eine *Hyperdatenbank* (HDB) die Infrastruktur für die Verwaltung eines Informationsraumes dar. Im Folgenden skizzieren wir kurz die wesentlichen Aspekte der HDB-Infrastruktur, die zum Verständnis dieses Artikels notwendig sind. Für eine detaillierte Beschreibung der HDB-Infrastruktur verweisen wir auf [SSS03, SSS02].

In einem Informationsraum lassen sich zwei Arten von Anwendungen identifizieren: Koordinations- und Benutzerprozesse. *Koordinationsprozesse* dienen der Wahrung der Abhängigkeiten zwischen den Diensten und vor allem zwischen den Daten und Dokumenten des Informationsraumes. Sie verbergen die vorliegenden Abhängigkeiten vor den Dienstbenutzern. Im Falle eines Verzeichnisdienstes, das auf mehreren Informationsquellen basiert, führt beispielsweise ein Koordinationsprozess jede Änderung der Originaldaten bzw. -dokumente möglichst schnell im Verzeichnis nach, um eine effektive Suche im Informationsraum zu ermöglichen.

Die zweite Art von Anwendungen innerhalb eines Informationsraumes, *Benutzerprozesse*, realisiert die Komposition neuer Dienste durch Zusammenfassung existierender Dienste zu Prozessen. Ziel ist es hier, neue Dienste innerhalb des Informationsraumes bereitzustellen und diese Dienste mit denselben Prinzipien wie die Basisdienste zugreifbar zu machen. Beiden Prozessarten liegt das Modell der *transaktionalen Prozesse* [SAB02, SAS99] zugrunde. Prozesse bestehen aus einzelnen Aktivitäten, wobei jede Aktivität einen Dienstaufruf darstellt. Im Prozessmodell sind neben regulären Dienstaufrufen auch *alternative* und *kompensierende* Dienstaufrufe erlaubt. Erstere ermöglichen die Angabe von alternativen Prozessausführungsstrategien und damit eine flexible Fehlerbehandlung, Letztere sorgen im Fehlerfall auch ohne geeignete Alternative für eine garantierte, korrekte Terminierung der Prozessausführung.

Für die Abarbeitung von Prozessen stellt die HDB eine Reihe von HDB-Basisfunktionalität zur Verfügung, etwa das Auffinden, Auswählen und Aufrufen von Diensten. Die Auswahl eines möglichst optimalen Dienstanbieters (Peer) erfolgt dynamisch zur Laufzeit unter Berücksichtigung der Dienstbeschreibung und einer Menge von Auswahlkriterien, wie zum Beispiel Kosten, Auslastungsgrad oder erwartete Antwortzeit. Hierzu verwaltet die HDB Information über die Dienstanbieter und deren Dienste sowie über die Koordinationsprozesse zur Wahrung der vorliegenden Abhängigkeiten zwischen diesen Diensten. Die HDB-Infrastruktur überwacht die Dienste des Informationsraumes und initiiert bei Bedarf automatisch die notwendigen Koordinationsprozesse.

Abbildung 1 zeigt einen Ausschnitt der Architektur einer konkreten Implementierung einer HDB-Infrastruktur, wie sie im OSIRIS-Projekt (**O**pen **S**ervice **I**nfrastructure for **R**eliable and **I**ntegrated process **S**upport) [SSS01] der Datenbankgruppe der ETH Zürich erfolgt.

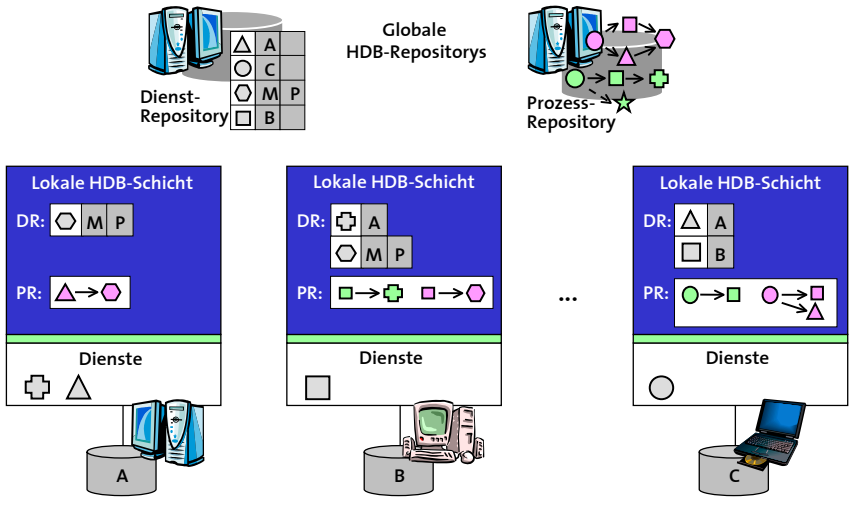

Abb. 1: Ausschnitt aus der OSIRIS-HDB-Infrastruktur

Hauptbestandteil der HDB-Infrastruktur ist eine *lokale HDB-Schicht*, die, ähnlich wie eine Netzwerkschicht, HDB-Basisfunktionalität lokal für die Peers bereitstellt. Sie installiert auf jedem Peer eine Prozessausführungsumgebung, die unter Beachtung der lokal verfügbaren Dienste und der globalen *vordefinierten* Prozesse peer-spezifisch konfiguriert wird.

Die global relevanten Metadaten über die Systemkonfiguration und die Dienste und Prozesse des Informationsraumes werden in globalen *HDB-Repositorys* verwaltet. Für das Verständnis der weiteren Ausführungen sind die beiden folgenden Repositorys wichtig: Das *Dienst-Repository (DR)* enthält die Informationen über die registrierten Dienste und ihre Anbieter, während das *Prozess-Repository (PR)* die Beschreibungen der Koordinations- und Benutzerprozesse verwaltet.

Um ein hohes Maß an Skalierbarkeit zu erzielen, ist die Architektur einer Hyperdatenbank möglichst dezentral gehalten. Das Wissen über die Dienste und Prozesse wird geschickt in den lokalen HDB-Schichten repliziert, so dass die Prozesse ohne zentrale Synchronisation ausgeführt werden können. Jede lokale HDB-Schicht erhält dazu den Ausschnitt der globalen Metadaten, der für die vollständig lokale Peer-to-Peer-Ausführung der Prozesse notwendig ist. Die Infrastruktur garantiert zum einen die Konsistenz der Replikate mit Hilfe eines Publish/Subscribe-Verfahren [SSS03] und zum anderen die sichere Prozessfortschaltung vom aktuellen zum nächsten Peer, der für die Ausführung des nächsten Prozessschrittes (Dienstes) ausgewählt wurde.

3 Mobilität von Dienstanbietern

Bisher sind wir implizit davon ausgegangen, dass alle registrierten Dienstanbieter ständig (über die gleiche Adresse) vernetzt sind. Mit dieser Annahme sind die

bereitgestellten Dienste stets verfügbar, abgesehen von den klassischen Fehlerfällen, in denen ein Dienstanbieter "abgestürzt" ist. Solche Fehler werden in OSIRIS bereits behandelt.

Die obige Annahme trifft in mobilen Umgebungen nicht mehr zu. Etliche Dienstanbieter neigen aus diversen Gründen dazu, mobil zu sein. Sie können sich jederzeit vom Informationsraum entkoppeln und bei Bedarf dynamisch wieder dazuschalten. Dabei kann es durchaus vorkommen, dass sich ein Dienstanbieter über neue Verbindungsparameter (zum Beispiel eine neue Adresse) in den Informationsraum einklinkt.

Im medizinischen Umfeld etwa können Patienten mit Sensoren ausgestattet sein. Diese ermitteln periodisch oder ereignisgesteuert relevante Patientendaten, beispielweise die Pulsfrequenz oder Blutzuckerwerte. Anschließend werden entsprechende Prozesse gestartet, die zum Beispiel die Messdaten in die Krankenakte des Patienten schreiben, Algorithmen zur Erkennung bestimmter Krankheitsmuster initiieren, in "Notfällen" den Patienten, den zuständigen Arzt oder sonstige Personen (Verwandte etc.) auf mögliche Unregelmäßigkeiten und Symptome hinweisen.

Allerdings sind Patienten, auch wenn sie im Krankenhaus sind, nicht immer am gleichen Ort. Folglich sind die Sensoren, die sie überwachen, mobil. Aus Sicht der HDB-Infrastruktur treten diese Sensoren als Dienstanbieter auf. Sie stellen die Patientendaten zum Beispiel dem überwachenden medizinischen Personal bzw. klinischen Anwendungen bei Bedarf zur Verfügung.

Wenn eine neuerliche Messung ansteht, kann der Patient offline sein oder sich gerade an einem Ort aufhalten, der sich von dem Ort der letzten Messung unterscheidet. Die Messung bzw. die anschließende Verarbeitung der Patientendaten soll dabei nicht durch die Mobilität des Patienten beeinträchtigt werden. Ebenso darf das temporäre Abschalten der Sensoren nicht immer als Fehlerfall gewertet werden, sondern muss als legaler Zustand wahrgenommen werden, da das temporäre Entkoppeltsein bewusst erfolgen kann. Außerdem sollten kurzzeitige (unbeabsichtigte) Verbindungsunterbrechungen wie sie zum Beispiel bei Funkverbindungen häufig vorkommen nicht sofort als Fehlerfall behandelt werden. Derartige Verbindungsunterbrechungen lassen sich relativ einfach und transparent mit den aus verteilten Datenbanken bekannten Time-Out-Verfahren [ÖV99] behandeln. Hier kommt es im Wesentlichen auf die geschickte Wahl der Time-Out-Periode an. Diese Periode kann sogar individuell gestaltet sein, etwa den Eigenschaften der jeweiligen mobilen Geräten entsprechend angepasst werden. Im Folgenden werden wir daher diese Art von Verbindungsunterbrechungen nicht weiter betrachten.

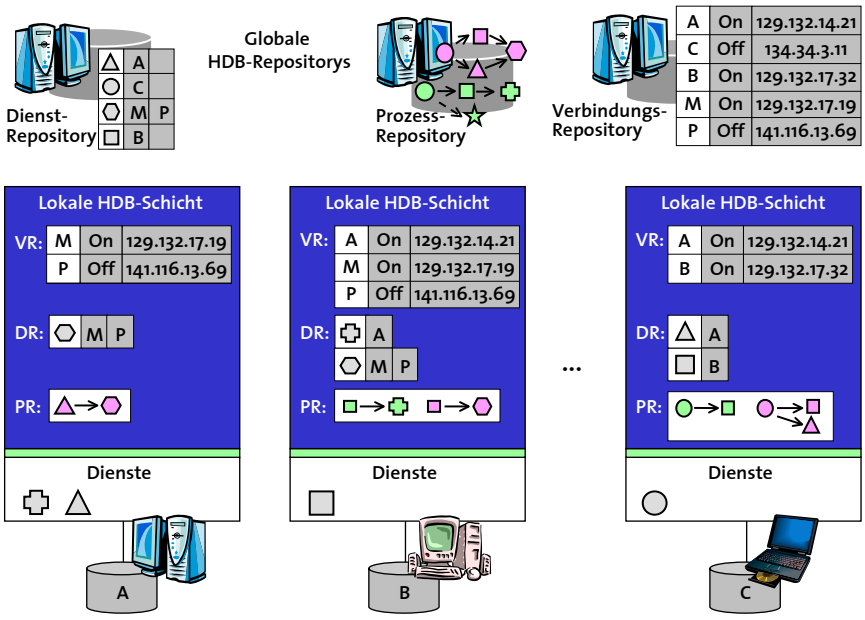

Abb. 2: Um Verbindungsmanagement erweiterte HDB-Architektur

3.1 Verbindungsmanagement

Um mit der Mobilität der Dienstanbieter transparent umgehen zu können, müssen in der Hyperdatenbank Verbindungsinformationen verwaltet werden. Ein naiver Weg ist das Entfernen aller zu einem entkoppelten Dienstanbieter gehörenden Metadaten, wenn der Dienstanbieter offline geht. Mit anderen Worten werden alle Dienste eines Dienstanbieters entfernt, sobald die HDB-Infrastruktur erkennt, dass der Dienstanbieter nicht mehr verfügbar ist. Da es dann keine Informationen über Dienstanbieter bzw. deren Dienste im Informationsraum gibt, wird ausgeschlossen, dass diese Dienste im Kontext von Prozessen aufgerufen werden können.

Dieser Ansatz ist jedoch dann unvorteilhaft, wenn sich Dienstanbieter *oft kurzzeitig* entkoppeln, zum Beispiel häufig zwischen dem Online- und Offline-Zustand aufgrund schlechter Verbindungen wechseln. Bei jeder neuen Verbindungsaufnahme müssen die Daten über den Dienstanbieter wieder komplett eingebracht und konsistent auf allen betroffenen lokalen HDB-Schichten repliziert werden. Dies ist zumeist mit einem großen Aufwand verbunden.

Eine weitere Einschränkung ist, dass sich die Dienstanbieter nach dem Eintreffen von Aufträgen nicht explizit abmelden und die Resultate später in den Informationsraum einbringen können. Ein Beispiel für das Offline-Arbeiten ist die Annotation von Krankheitsbildern, die ein Arzt auf seinen Laptop transferiert und die er ohne Netzwerkverbindung, zum Beispiel im Krankenwagen, bearbeitet.

Aus den oben genannten Gründen bietet sich im Umfeld mobiler, dynamischer Dienstanbieter ein *explizites* Verbindungsmanagement an. Hierzu wird die Menge der globalen HDB-Repositorys um ein Verbindungsrepository erweitert, das alle verbindungsrelevanten Informationen verwaltet (siehe Abbildung 2). Das Verbindungsrepository merkt sich für jeden Dienstanbieter die aktuell

bekannten Verbindungsparameter. Entkoppelt sich ein Dienstanbieter vom Informationsraum, so wird diese Zustandsänderung im Verbindungsrepository vermerkt und an alle lokalen HDB-Schichten weitergeleitet, für die diese Änderung relevant ist. Die aktive Mitteilung einer Zustandsänderung ist natürlich nur bei einer bewussten Verbindungstrennung möglich.

Jeder Dienstanbieter teilt dazu seine Zustandsänderungen den korrespondieren Verbindungsrepositorys mit. Für die Verteilung dieser Änderungen an die lokalen HDB-Schichten wird wiederum das Publish/Subscribe-Prinzip verwendet. In diesem Fall abonnieren sich die lokalen HDB-Schichten auf Zustandsänderungen der Dienstanbieter, deren Dienst sie aufgrund der existierenden Prozessbeschreibungen möglicherweise in Anspruch nehmen könnten.

3.2 Proxymanagement

Um vollständig von Entkopplungen abstrahieren zu können, müssen die lokalen HDB-Schichten mit Proxy-Funktionalität ausgestattet werden. Das heißt, die HDB-Infrastruktur muss für jeden Dienstanbieter einen ständig vernetzten *Stellvertreter* (Proxy) anbieten, der die ankommenden Mitteilungen empfängt und verwaltet, solange der Dienstanbieter entkoppelt ist. Wenn sich ein Dienstanbieter wieder an den Informationsraum ankoppelt, ist er durch einen Abgleich mit seinem Proxy immer auf dem aktuellen Stand. Dies betrifft zum einen während der Abwesenheit eingegangene Dienstaufrufe und zum anderen die aktuell verfügbaren Dienste im Gesamtsystem. Zum Beispiel wird er keinen Dienst von einem Anbieter aufrufen, der zum Zeitpunkt der Entkopplung noch vernetzt war, nun aber entkoppelt ist.

Mit Hilfe des Proxy-Konzepts können Aufenthaltsortänderungen von Dienstanbietern transparent erfasst werden. Koppelt sich ein Dienstanbieter von einer anderen Adresse wieder in den Informationsraum an, so wird dies mit Hilfe des lokalen Verbindungsrepositorys erkannt und der Proxy-Kontext wird entsprechend wiederhergestellt.

4 Ad-hoc-Prozesse als mobile Anwendungen

OSIRIS ist dafür optimiert, standardisierte Abläufe mit hoher Wiederholfrequenz auszuführen. *Ad-hoc-Prozesse* zeichnen sich hingegen dadurch aus, dass sie ein spezielles, in dieser Form meist einmaliges Benutzerbedürfnis erfüllen und daher nur einmal bzw. einige wenige Male ausgeführt werden.

Beispiele für Ad-hoc-Prozesse finden sich auch in Krankenhausinformationssystemen. Zwar kann man standardisierte Prozessmuster für die Behandlung von Patienten definieren, allerdings gibt es viele individuelle, zum Teil unvorhersehbare Rahmenparameter, die den genauen Ablauf dieser Therapie-Prozesse bereits im voraus beeinflussen und die eine individuelle Konfiguration erfordern. Daher ist es unter solchen Randbedingungen sinnvoll, Ad-hoc-Prozesse zu unterstützen.

Ad-hoc-Prozesse sind nicht vordefiniert. Entsprechend sind die lokalen HDB-Schichten des Informationsraumes nicht für die Ausführung solcher Prozesse vorkonfiguriert. In unserer Realisierung stellen Ad-hoc-Prozesse gekapselte Einheiten bestehend aus der Prozessdefinition und dem Prozesszustand dar. Die Ausführung von Ad-hoc-Prozessen erfordert daher, dass die Prozessdefinition und der Prozesszustand mit jedem Dienstaufruf möglicherweise von Peer zu Peer weitergereicht werden. In diesem Sinne sind Ad-hoc-Prozesse inhärent mobil. Sie wandern mit der Prozessausführung zusammen durch den Informationsraum und erlauben dem Benutzer sich temporär abzumelden, während sie weiterhin aktiv ihre Aktivitäten gemäß der Prozessdefinition abarbeiten.

Wie für Koordinations- und Benutzerprozesse, so gilt für Ad-hoc-Prozesse, dass die verlässliche Ausführung von zentraler Bedeutung ist. In einem mobilen Umfeld kommt noch hinzu, dass die Umgebung nicht immer (technisch) zuverlässig ist. Dies erfordert, dass Prozesse dynamisch temporär nicht verfügbare Dienste ersetzen und sich so dynamisch dem System anpassen können müssen.

Aktivitäten können einzelne Dienstaufrufe, aber auch Aufrufe einer Menge von semantisch äquivalenten Diensten darstellen. Wir abstrahieren hier von Ontologie-Problemen und gehen davon aus, dass jeder Dienst eine entsprechende Beschreibung hat, mit der sich die semantische Äquivalenz herausfinden lässt. Ein Beispiel für semantisch äquivalente Dienste ist die Suche nach einem nahegelegenen Krankenhaus mit freien Kapazitäten. Hierzu werden jeweils entsprechende Dienste in den Informations-

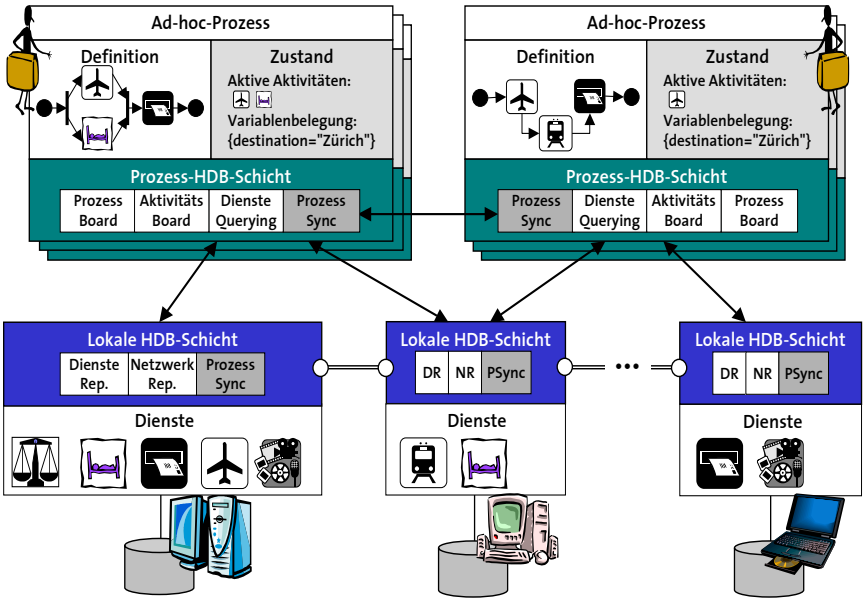

Abb. 3: HDB-Infrastruktur für Ad-hoc-Prozesse

systemen von Krankenhäusern parallel aufgerufen. Diese Dienste ermitteln, ob es freie Kapazitäten gibt, so dass nachdem die Ergebnisse dieser Aufrufe feststehen, über alle parallelen Aufrufe hinweg das nächstgelegenste verfügbare Krankenhaus bestimmt werden kann. Diese Form der Parallelität nennen wir Dienstparallelität, um sie von der Prozessparallelität abzugrenzen, bei der zum Beispiel gleichzeitig ein OP-Raum und ein Ärzteteam reserviert wird.

4.1 Ausführung von Ad-hoc-Prozessen

In OSIRIS werden die lokalen HDB-Schichten entsprechend vorkonfiguriert und mit Informationen über die möglichen Prozesse im System ausgestattet. Dies ist vor allem für feststehende Koordinationsprozesse, wie sie zum Beispiel zur Konsistenzsicherung eingesetzt werden, sinnvoll. Da Ad-hoc-Prozesse aber keinem vorgegebenen Muster folgen, ist es nicht effizient, Meta-Information für die Prozessausführung im System zu verteilen.

Vielmehr ist jeder Ad-hoc-Prozess proaktiv für die Ausführung seiner Aktivitäten verantwortlich. Das schließt vor allem auch die Fehlerbehandlung einzelner Prozesse sowie die Synchronisation paralleler Prozesse mit ein. In unserer Realisierung erfolgt die Fehlerbehandlung und Synchronisation gemäß der Theorie der transaktionalen Prozesse [SAB02]. Dies spiegelt sich auch in der HDB-Infrastruktur für Ad-hoc-Prozesse wieder (siehe Abbildung 3). Neben der Prozessdefinition und dem aktuellen Zustand der Prozessinstanz enthält ein Ad-hoc-Prozess auch die HDB-Funktionalität, die für die Fehlerbehandlung und Synchronisation von parallelen Prozessen benötigt wird.

Ein Ad-hoc-Prozess wird von einem Nutzer des Informationsraumes auf einem Peer des Informationsraumes gestartet. Hierzu muss der Nutzer der lokalen HDB-Schicht der Komponente eine Spezifikation des Prozesses übergeben. Die Ausführung des Prozesses erfolgt dann in engem Zusammenspiel der lokalen HDB-Schichten der einzelnen Komponenten des Informationsraumes und der in den Ad-hoc-Prozess eingebetteten HDB-Funktionalität.

Zunächst wird der Prozessspezifikation die erste Aktivität des Prozesses entnommen. Es müssen nun Anbieter gefunden werden, die diesen Dienst bereitstellen. Falls die benötigte Information in der lokalen HDB-Schicht vorhanden ist, dann kann diese Information direkt ausgelesen und das Migrationsziel des Ad-hoc-Prozesses bestimmt werden. Dies ist zum Beispiel der Fall, wenn die Konfiguration dieser lokalen Schicht für einen Koordinationsprozess genau die gesuchte Information bereits enthält, Falls die Information nicht lokal verfügbar ist, dann muss im Informationsraum nach Anbietern für diesen Dienst gesucht werden. Hierzu wird die Dienste-Querying-Komponente des Ad-hoc-Prozesses verwendet. Diese Komponente arbeitet mit dem Netzwerk-Repository der lokalen HDB-Schicht zusammen. Das Netzwerk-Repository enthält Verweise auf Verzeichnisdienste im Informationsraum, die selbst verteilt sein können. Der Ad-hoc-Prozess ermittelt die möglichen Dienstanbieter mit Hilfe dieser Verzeichnisse. Für die Anfrage im Netzwerk können weitere Parameter als Auswahlkriterium verwendet werden, zum Beispiel die Kosten der Dienstaufrufe, Last- und Verbindungsinformation etc. Für detailliertere Informationen über die dynamische Auswahl von Dienstanbietern verweisen wir auf [HS01].

Je nach Art der Aktivität migriert der Ad-hoc-Prozess zu einem oder mehreren Dienstanbietern. Letzteres geschieht mit Hilfe von Prozess-Klonen. Dabei sind zwei Arten von Parallelität zu berücksichtigen:

1. *Prozessparallelität*: Ein Prozess kann mehrere parallele Abschnitte enthalten, also Folgen von Diensten, die gleichzeitig aufgerufen werden.
2. *Dienstparallelität*: Für die Ausführung einer Aktivität kann es mehrere semantisch äquivalente Dienste geben.

Sowohl Prozessparallelität als auch Dienstparallelität lassen sich realisieren, indem die Ad-hoc-Prozessinstanz mitsamt der mitgeführten HDB-Funktionalität dupliziert (“geklont”) wird. Das erlaubt das eigenständige Weiterbewegen dieser Klone, um so Dienste parallel aufzurufen. Am Ende einer Aktivität, hinter der sich mehrere Dienstaufrufe verstecken, sowie am Ende paralleler Pfade in der Prozessbeschreibung muss eine Synchronisation innerhalb des Prozesses erfolgen. Dies wird von der HDB-Funktionalität, die in die Ad-hoc-Prozesse eingebettet ist, unterstützt.

Die Prozessparallelität wird durch das Klonen von Ad-hoc-Prozessen erreicht. Parallele Abschnitte des Prozesses werden von unabhängigen Klonen ausgeführt. Die Klone können je nach Ausführung auf verschiedenenen Peers des Informationsraumes arbeiten. Zur Synchronisation der Klone besitzt jeder Klon ein *Prozessboard*, in dem er die Prozessvariablen verwaltet. Das Prozessboard wird beim Zusammenführen der Klone entsprechend konsolidiert. Diese Zusammenführung wird durch die Prozessdefinition festgelegt.

Im Falle der Dienstparallelität werden ebenfalls mehrere Klone erzeugt, die unabhängig voneinander verschiedene Peers des Informationsraumes ansteuern und lokal dort Dienste aufrufen. Aktivitäten, hinter denen sich einfache Dienstaufrufe verbergen, können nach der Migration des Klons zum entsprechenden Dienstanbieter direkt ausgeführt werden. Nach Abarbeitung der Aktivität ist eine Zusammenführung der Klone notwendig. Dieser Zusammenschluss wird durch die in den Ad-hoc-Prozessen eingebettete HDB-Funktionalität realisiert. Jeder Klon verwaltet dazu lokale Variablen im *Aktivitätsboard*, das zum Beispiel die Ergebnisse der abgearbeiteten Dienstaufrufe speichert. Die Zusammenführung der Klone ist in diesem Fall parametrisiert. Die Werte in den lokalen Variablen entscheiden darüber, welcher Klon den kompletten Prozess fortführt. Bei Suche nach einem passenden Krankenhaus, zum Beispiel, würde derjenige Klon ausgewählt werden, bei dem der Ergebniswert “verfügbar” lautet und die Variable “Distanz” den geringsten Wert im Vergleich zu allen anderen Klonen aufweist, die ebenfalls verfügbare Krankenhäuser gefunden haben. Nach der Konsolidierung der Klone erfolgt die Ausführung der nächsten Aktivität gemäß des oben beschriebenen Verfahrens.

4.3 Synchronisation von Ad-hoc-Prozessen

Ziel der Synchronisation von Ad-hoc-Prozessen ist, die Ausführung verschiedener Prozesse voneinander zu isolieren, damit jeder Prozess während seiner ganzen Laufzeit über eine konsistente Sicht auf den Informationsraum verfügt. Als formale Basis für die Isolation der nebenläufigen Prozessausführungen liegt unseren Arbeiten die Unified Theory of Concurrency Control and Recovery [VHB98] zugrunde. Diese vereinheitlichte Theorie überwindet die strikte Trennung bei der Betrachtung von Concurrency Control und Recovery, welche zu unnötigen Restriktionen führt, wenn beide Probleme gleichzeitig gelöst werden müssen. Grundidee der vereinheitlichten Theorie ist das explizite Betrachten von Aktivitäten, die für die Fehlerbehandlung nötig sind (zum Beispiel der Aufruf von Kompensationsdiensten zu einer regulären Aktivität), was schließlich dazu führt, dass jeder Prozess mit einem Commit endet (nach der Betrachtung der Rücksetzaktivitäten wird ein Abort durch ein Commit ersetzt). Damit wird es möglich, gleichzeitig über die Korrektheit der "normalen" Ausführung und die der Fehlerbehandlung in einem gemeinsamen Framework zu schließen.

Bei der Auswahl bzw. dem Entwurf eines entsprechenden Protokolls für Ad-hoc-Prozesse der HDB liegen folgende Annahmen zugrunde:

1. Es handelt sich um langlebige Prozesse, zum Beispiel transaktionale Workflows in E-Commerce-Anwendungen wie Online-Versteigerungen oder elektronische Handelsplätze.
2. Die Konfliktwahrscheinlichkeit zwischen verschiedenen Dienstaufrufen ist gering.
3. Die Anwender gehören zu Communitys, welche Anwendergruppen mit ähnlichen Anforderungen repräsentieren. Folglich nutzen die Anwender einer Community ähnliche Dienste. Daher treten Konflikte meist innerhalb dieser Communitys, aber nur selten zwischen Anwendern verschiedener Communitys auf.
4. Daten und Dokumente verschiedener Peers sind unabhängig voneinander. Folglich kann es nur zum Zugriff auf gemeinsame Ressourcen (und damit zu Konflikten) zwischen lokalen Dienstaufrufen kommen, nicht aber zwischen Dienstaufrufen auf verschiedenen Peers.

Entsprechend unserer Philosophie, die Prozessausführung vollständig zu dezentralisieren, wird die Isolation der Prozessausführung durch ein verteiltes Protokoll sichergestellt. Hierzu eignen sich Sperrverfahren aufgrund der Annahme (1) nicht. Aus Annahme (2) folgt, dass optimistische Verfahren vorteilhaft einsetzbar sind. Verfahren, die erst zum Commit-Zeitpunkt die Korrektheit einer Ausführung überprüfen, haben jedoch den Nachteil, dass Fehler erst sehr spät erkannt werden, wie zum Beispiel bei der rückwärtsgerichteten optimistischen Concurrency Control BOCC [KR81]. Wir folgen daher einem hybriden Ansatz: Dienstaufrufe werden direkt, das heißt optmistisch ausgeführt, während gleichzeitig kontinuierlich Meta-Information über die korrekte Ausführung geführt wird. Dies geschieht, indem ein Serialisierungsgraph auf Zyklen überprüft wird.

Diese Überprüfung hat aber dezentral zu erfolgen. Hierbei gehen wir aufgrund der Annahmen (2) und (3) davon aus, dass der globale Serialisierungsgraph in kleine, unverbundende Partitionen (*Regionen*) zerfällt. Falls wir nun sicherstellen können, dass alle Serialisierungsgraphen der Regionen zyklenfrei sind, so trifft dies auch auf den globalen Graphen zu. Um dies zu erreichen, statten wir jeden Prozess mit einer Kopie des Serialisierungsgraphen seiner Region aus, womit die Prozesse in die Lage versetzt werden, Zyklenfreiheit selbstständig zu garantieren. Hierzu ist jeder Ad-hoc-Prozess bzw. dessen eingebettete HDB-Funktionalität auf die Zusammenarbeit mit der lokalen HDB-Schicht angewiesen. Dieser fällt die Aufgabe zu, die lokalen Konflikte zu erkennen. Hierzu speichert sie, welche Dienstaufrufe bisher lokal stattgefunden haben. Verfügt sie weiterhin über eine Konfliktmatrix, so kann sie somit die lokalen Konflikte - und andere können gemäß Annahme (4) nicht auftreten - ermitteln. Als Antwort auf einen Dienstaufruf kann dann die Prozess-Sync-Komponente der lokalen HDB-Schicht die neu verursachten Konflikte an den Ad-hoc-Prozess, genauer an dessen Prozess-Sync-Komponente zurückgeben, welche damit ihren Serialisierungsgraphen auf den neuesten Stand bringt. Anschließend werden die Änderungen dieses Graphens bzw. der komplette aktualisierte Graph an die anderen Prozesse der gleichen Region weiterpropagiert. Für eine ausführliche Diskussion dieses Protokolls verweisen wir auf [HSS03].

4.4 Verwandte Arbeiten

Ad-hoc-Anwendungen in Form von Transaktionen oder Prozessen werden in mehreren verwandten Arbeiten durch mobile Agenten realisiert [CGN96, AP98]. Allerdings liegt der Fokus dieser Arbeiten auf einzelnen Aspekten wie Fehlertoleranz oder Transaktionsmigration. Der Isolationsaspekt bleibt unberücksichtigt. [SAE01] diskutiert teilweise auch den Aspekt der Isolation, allerdings unter anderen Voraussetzungen als wir. Ebenso gibt es einige Arbeiten, die sich mit Mobilität und transaktionalen Garantien beschäftigen. Hier seien [DHB97, WC99, KK00] stellvertretend genannt. Der Hyperdatenbank-basierte Ansatz für Ad-hoc-Anwendungen zeichnet sich gegenüber diesen Ansätzen dadurch aus, dass sämtliche Probleme, die durch Parallelität oder Fehlertoleranz entstehen, gesamthaft adressiert werden.

5 Zusammenfassung und Ausblick

Mobilität und Individualität von Dienstanbietern und -nutzern stellen besondere Anforderungen an Infrastrukturen moderner Informationssysteme. In diesem Artikel haben wir die Unterstützung dieser beiden Aspekte in Hyperdatenbank-Infrastrukturen skizziert. Da die vorgeschlagenen Erweiterungen orthogonal zueinander sind, können sie wie in Abbildung 4 dargestellt in einer gemeinsamen Infrastruktur integriert werden, in der dann sowohl vordefinierte als auch Ad-hoc-Prozesse transparent im mobilen Umfeld ausgeführt werden können.

Die lokalen HDB-Schichten ermög-

lichen die Ausführung von Prozessen unter garantierter, korrekter Terminierung gemäß dem Modell der transaktionalen Prozesse.

Ad-hoc-Prozesse stellen individuelle Anwendungen einzelner Nutzer dar. In einigen Fällen enstehen Ad-hoc-Prozesse dynamisch, indem vordefinierte, standardisierte Prozessmuster nach der Instanzierung individuell angepasst werden. Solche dynamischen Änderungen sind vor allem im Kontext von medizinischen Informationssystemen von großer Bedeutung [RD98, MR99]. Der hier verfolgte Ansatz sieht die Individualisierung von Anwendungen bereits zum Definitionszeitpunkt vor.

Bei Ad-hoc-Prozessen besitzen die lokalen HDB-Schichten, im Gegensatz zur Ausführung vordefinierter Prozesse, keine Prozessbeschreibungen. Mit dem Aufruf eines Dienstes erhält die korrespondierende lokale HDB-Schicht die Kontrolle über den Prozess. Nach der Abarbeitung des Dienstes protokolliert die HDB-Schicht die Ausführung des Dienstes, um eine spätere Kompensation zu unterstützen, und gibt die Kontrolle an die HDB-Schicht eines ausgewählten Dienstes gemäß Prozessbeschreibung weiter.

Bei Ad-hoc-Prozessen hingegen besitzen die lokalen HDB-Schichten keine Prozessbeschreibungen. Die Prozessbeschreibungen kommen mit den Prozessinstanzen. Die HDB-Schichten bieten lediglich eine Ausführungsumgebung für die Prozesse, die unter anderem auch das Migrieren der Prozessinstanz samt Beschreibung ermöglicht. Ad-hoc-Prozesse sind nicht nur auf das mobile Umfeld beschränkt. Die Mobilität von Dienstanbietern und -nutzern verstärkt jedoch den Wunsch nach Ad-hoc-Prozessen, die individualisierte Lösungen für aktuelle, spezielle Anforderungen eines Nutzers liefern.

Die Erweiterung der HDB-Infrastruktur um explizites Verbindungsmanagement erfolgt im Rahmen des OSIRIS-Projekts. Nach der vollständigen Implementierung möchten wir unseren Prototyp nutzen, um zu untersuchen, unter welchen Konstellationen sich das Replizieren der Verbindungsinformation gegenüber einer eher konservativen Strategie mit zentralen Verbindungsinformationen auszahlt. Dazu ist allerdings eine ausführliche Evaluation notwendig.

Im Rahmen des Projekts AMOR (**A**gents **MO**bility and t**R**ansactions) entsteht ein Protoyp für die Ausführung von Ad-hoc-Prozessen. In diesem Projekt haben wir uns bisher mit dem Identifizieren von Diensten in Peer-to-Peer-Netzwerken beschäftigt [HS01]. Darüber hinaus sind Arbeiten zur Prozessausführung mit Isolationsgarantie abgeschlossen [HSS03]. Momentan realisieren wir die Aspekte der Prozessparallelität und der atomaren Ausführung in unzuverlässigen Umgebungen. Die Implementierung der Ad-hoc-Prozesse erfolgt dabei in Form von mobilen Agenten. Unser langfristiges Ziel ist es, beide Prototypen in eine integrierte Plattform für die gemeinsame Ausführung von mobilen vordefinierten und Ad-hoc-Prozessen zu überführen.

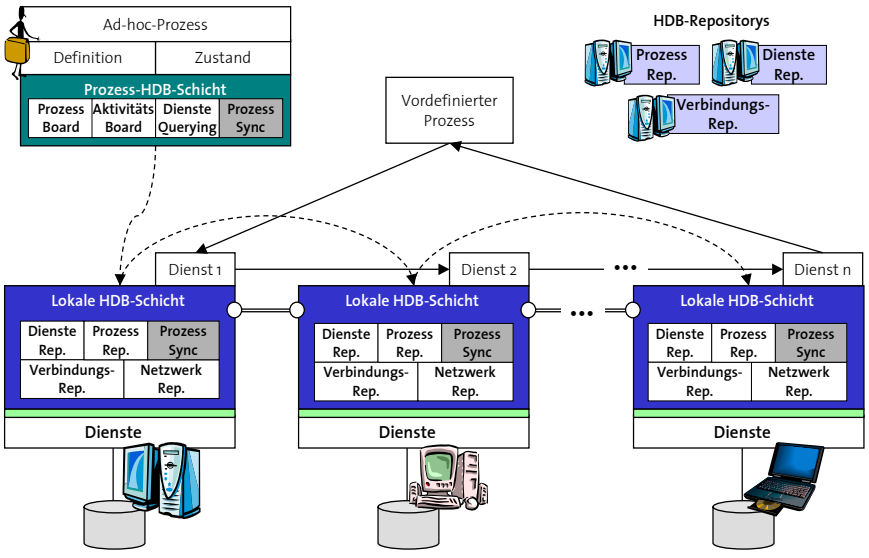

Abb. 4: HDB-Infrastruktur für mobile vordefinierte und Ad-hoc-Prozesse

Literatur

[CGN96] T. Cai, P. Gloor, S. Nog: DartFlow: A Workflow Management System on the Web using Transportable Agents. Technical Report TR-96-283, Dartmouth College, 1996.

[DHB97] M. H. Dunham, A. Helal, and S. Balakrishnan. A Mobile Transaction Model That Captures Both the Data and Movement Behavior. Mobile Networks and Applications (MONET), 2(2):149-162, 1997.

[AP98] R. Popescu-Zeletin, F. M. Assis Silva: An Approach for Providing Mobile Agent Fault Tolerance. In K. Rothermel and F. Hohl, editors, Mobile Agents, Proc. Second Int. Workshop, MA'98, Stuttgart, Germany, September 1998, Seiten 14-25, Lecture Notes in Computer Science 1477, Springer-Verlag, 1999.

[HS01] K. Haller, H. Schuldt: Using Predicates for Specifying Targets of Migration and Messages in a Peer-to-Peer Mobile Agent Environment. In Proc. of the 5th Int. Conf. on Mobile Agents, Atlanta, GA, Seiten 152-168, Lecture Notes in Computer Science 2240, Springer-Verlag, 2001.

[HSS03] K. Haller, H. Schuldt, H.-J. Schek: Transactional Peer-to-Peer Information Processing: The AMOR Approach. In Proc. of the 4th Int. Conf. on Mobile Data Management, Melbourne, Australia, 2003. Erscheint.

[KK00] K.-I. Ku, Y.-S Kim: Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems. In RIDE 2000, Proc. of the 10th Int. Workshop on Research Issues in Data Engineering: Middleware for Mobile Business Applications and E-Commerce, February 27-28, 2000, San Diego, CA, USA, Seiten 39-46. IEEE Computer Society Press, 2000.

[KR81] H. Kung, J. Robinson: On Optimistic Methods for Concurrency Control. ACM Transactions on Database Systems, 6(2): 213-226, 1981.

[MR99] R. Müller, E. Rahm: Rule-Based Dynamic Modification of Workflows in a Medical Domain. In Proc. der GI-Tagung "Datenbanksysteme in Büro, Technik und Wissenschaft, Freiburg, März 1999", BTW'99, Seiten 429-448, Springer-Verlag, 1999.

[ÖV99] M. T. Özsu, P. Valduriez: Principles of Distributed Database Systems. Prentice Hall, 2. Auflage, 1999.

[RD98] M. Reichert, P. Dadam: ADEPT$_{flex}$ - Supporting Dynamic Changes of Workflows without Losing Control. In Journal of Intelli-

gent Information Systems, 10(2):93-129, 1998.

[SSS01] C. Schuler, H. Schuldt, H.-J. Schek: Supporting Reliable Transactional Business Processes by Publish/Subscribe Techniques. In Proc. of the 2nd International Workshop on Technologies for E-Services (TES'01), Rome, Italy, September 2001.

[SSS03] H.-J. Schek, H. Schuldt, C. Schuler, C. Türker, R. Weber: Hyperdatenbanken zur Verwaltung von Informationsräumen. *Erscheint in:* it-Information Technology, 2003.

[SSS02] H.-J. Schek, H. Schuldt, C. Schuler, R. Weber: Infrastructure for Information Spaces. In Advances in Databases and Information Systems, Proc. of the 6th East-European Symposium, ADBIS'2002, Bratislava, Slovakia, September 2002, Seiten 23-36, Lecture Notes in Computer Science 2435, Springer-Verlag, Berlin, 2002.

[SAB02] H. Schuldt, G. Alonso, C. Beeri, H.-J. Schek: Atomicity and Isolation in Transactional Processes. ACM Transactions on Database Systems, 27(1):63-116, 2002.

[SAS99] H. Schuldt, G. Alonso, H.-J. Schek: Concurrency Control and Recovery in Transactional Process Management. In Proc. of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS'99, May 31-June 2, 1999, Philadelphia, Pennsylvania, Seiten 316-326, ACM Press, 1999.

[SAE01] R. Sher, Y. Aridor, O. Etzion: Mobile Transactional Agents. In Proc. of the 21st Int. Conf. on Distributed Computing Systems (ICDCS 2001), April 16-19, 2001, Phoenix, Arizona, USA, Seiten 73-80. IEEE Computer Society Press, 2001.

[VHB98] R. Vingralek, H. Hasse-Ye, Y. Breitbart, H-J. Schek: Unifying Concurrency Control and Recovery of Transactions with Semantically Rich Operations. Theoretical Computer Science, 190(2):363-396, 1998.

[WC99] G. D. Walborn, P. K. Chrysanthis: Transaction Processing in PRO-MOTION. In Proc. of the 1999 ACM Symposium on Applied Computing, February 28 - March 2, 1999, San Antonio, Texas, USA, Seiten 389-398. ACM Press, 1999.

Dr.-Ing. Can Türker studierte Informatik an der Technischen Universität 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 Datenbankgruppe von Prof. Dr. H.-J. Schek.

Dipl.-Technoinform. Klaus Haller studierte von 1995 bis zu seinem Diplom im September 2000 an der Universität Kaiserslautern. Seit Oktober 2000 ist wissenschaftlicher Mitarbeiter in der Gruppe von Prof. Dr. H.-J. Schek.

Dr.sc.techn. Heiko Schuldt 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. Seit Januar 2001 ist er dort als Oberassistent tätig.

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, 1978) 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 in 2001 zum ACM Fellow ernannt.

ETH Zürich
Institut für Informationssysteme
Datenbankgruppe
ETH Zentrum
CH-8092 Zürich

{tuerker, haller, schek, schek}@inf.ethz.ch