Ein Ansatz zur dynamischen Anpassung von Webservices

Burkhard Freitag, Tobias Geis

Universität Passau
{freitag,geis}@im.uni-passau.de

Abstract: Um Webservices zur Verarbeitung heterogener Daten im Web dynamisch an die aktuellen Erfordernisse anpassen zu können, wird ein flexibles und effizientes Variantenmanagement auf der Grundlage semantischer Information benötigt. Die in diesem Beitrag vorgeschlagene Modellierung und das zugehörige Multiversionsprotokoll erlauben es, die Verarbeitungsprozesse ohne Zeitverlust dynamisch zu verändern und zu optimieren.

1 Motivation

Webservices verarbeiten heterogene Daten, die auch als Ströme wie z.B. bei der Überwachung von Newsgroups vorliegen können. Die Verarbeitung dieser Daten hängt von Faktoren wie Art, Inhalt und Format der Daten ab, kann aber auch das Profil des Nutzers, der wiederum ein Webservice sein kann, oder auch andere Metainformationen berücksichtigen. Derartige Anpassungen sind letztlich auf der Ebene der Prozesse, die den Webservices zugrunde liegen, zu leisten. Dazu kann sowohl die Auswahl der „richtigen“ Ausprägung eines Teilprozesses als auch die ständige Aktualisierung von Teilprozessen wünschenswert bzw. erforderlich sein. Angesichts solcher - möglicherweise ständiger - Änderungen von Teilprozessen im laufenden Betrieb ist die Konsistenzerhaltung eine wichtige Aufgabe.

Es ist nahe liegend, Webservices als Workflows zu spezifizieren und Workflow Management Systeme zu ihrer Verwaltung und Ausführung einzusetzen. Die Forderung nach dynamischer Anpassbarkeit wird damit auf das Workflow Management übertragen. Bei langlaufenden Workflows, die z.B. Foren oder Newsticker überwachen, muss es beispielsweise möglich sein, die Algorithmen zur Informationsgewinnung jederzeit zu verbessern. Änderungen des Formats, des Inhalts, der Benutzerpräferenzen etc. dürfen nicht zu Laufzeitfehlern oder inadäquaten Resultaten führen. Im Folgenden wird ein Ansatz für ein Workflow Management System vorgestellt, das diesen Anforderungen genügt, indem es Informationen über semantische Eigenschaften der von ihm gesteuerten Prozesse ausnutzt.

Nach der Einführung von Prozessvarianten und -versionen wird eine transaktionale Repräsentation von Workflows und Anpassungsoperationen vorgestellt. Das anschließend präsentierte Multiversionsprotokoll steuert die korrekte dynamische Auswahl von Varianten und Versionen. Der Beitrag endet mit einer kurzen Diskussion und Zusammenfassung.

2 Varianten und Versionen von Prozessen

Es sollen zwei verschiedene Arten von Anpassungen betrachtet werden: Wir sprechen von einer neuen *Version* eines (Teil-)Prozesses, wenn eine neue (Teil-)Prozessdefinition die alte ersetzen soll. Im Gegensatz dazu verstehen wir unter *Varianten* parallel existierende Definitionen des gleichen Teil-Prozesses, die z.B. abhängig von den Eigenschaften im Benutzerprofil oder der Art der Eingangsdaten, eingesetzt werden. Die Auswahl der jeweils „richtigen“ Variante wird dabei von einem Inferenzmechanismus übernommen, der auf Informationen über den Einsatzkontext einerseits und auf Metadaten oder auf einer beschreibungslogische Charakterisierung der Prozess-Varianten andererseits aufsetzt. Aus Platzgründen wird in diesem Beitrag nicht näher auf die Ermittlung der richtigen Varianten eingegangen.

Als Beispiel stelle man sich einen Webservice vor, der Unterrichtsmaterial an den jeweiligen Benutzer anpasst. Als dynamische Komponenten lassen sich erstens die Benutzerprofile, denen z.B. neue Kompetenzen hinzugefügt werden können, und zweitens die Ausbildungsprozesse, die ständig aktualisiert und umstrukturiert werden, identifizieren. Die Benutzerprofile werden in standardisierter Form (IMS LIP, PAPI) – idealerweise von einem anderen Webservice – zur Verfügung gestellt. Varianten machen es möglich, personalisierte Ausbildungsprozesse abhängig vom Wissensstand des Benutzers automatisch abzuleiten und auf neu erworbene Kompetenzen zu reagieren. Das in Abschnitt 4 skizzierte Multiversionsprotokoll sorgt dafür, dass jeweils die adäquate Version der möglichen Varianten konsistent ausgewählt wird (vgl. Abbildung 1). Dies geschieht, indem es Änderungen und Umstrukturierungen in noch nicht absolvierten Teilausbildungsabschnitten zulässt, und Änderungen an bereits absolvierten Abschnitten, die z.B. wiederholt werden, um den Lernstoff zu vertiefen oder aufzufrischen, nur dann propagiert, wenn die neue Version zu der absolvierten Version „semantisch äquivalent“ ist. Letzteres ist in praktischen Anwendungen beispielsweise dann der Fall, wenn Teile aus didaktischen Gründen umstrukturiert oder Fehler korrigiert wurden. Ein herkömmliches Multiversionsprotokoll würde hier aufgrund der Tatsache, dass „zu spät“ geschrieben wurde, den Zugriff auf die neueren Versionen verhindern.

3 Transaktionale Repräsentation

In dem hier beschriebenen Ansatz wird das Workflow Management System ULTRAflow [FF01, FRF02] zur Definition des einem Webservice unterliegenden Prozesses eingesetzt. Intern wird die DATALOG-ähnliche, regelbasierte Workflow-Spezifikationssprache ULTRA [FWF00, FF01] eingesetzt. Die Bedienoberfläche stellt aber zusätzlich einen graphischen Formalismus zur Verfügung, der auch das Arbeiten mit Workflow-Patterns unterstützt. Dank der hierarchischen Struktur von ULTRA-Regeln können Workflows in einfacher Weise durch Wiederverwendung und Komposition von Sub-Workflows definiert werden.

Die Definitionen von Aktionen und Teil-Workflows werden in ULTRAflow als Datenob-

jekte zusammen mit ihren Metadaten in einem Repository gehalten. Wenn eine Workflow-Instanz gebildet werden und zur Ausführung kommen soll, wird die zugehörige Definition gelesen und in ausführbare Schritte umgesetzt. Auf Teil-Workflows wird dabei eine Late-Binding Strategie angewendet, so dass die tatsächlich zur Ausführung kommende Definition dynamisch an den entsprechenden „Aufruf“ des Teil-Workflows gebunden wird. Diese Repräsentation von Workflow-Definitionen bildet in dem hier vorgestellten Ansatz die Grundlage für die dynamische Anpassbarkeit von Workflows und damit Webservices.

Workflow-Instanzen werden in ULTRAflow transaktional ausgeführt und aktualisiert. Dabei erlauben wir allgemein die konkurrente Ausführung einer Workflow-Instanz und eine Änderung der ihr zugrunde liegenden Workflow-Definition. Das Ausführungsmodell unterscheidet zwischen *administrativen Transaktionen*, die etwaige Änderungen der Workflow-Definition durchführen, und *operativen Transaktionen*, die die Erzeugung der Aktionsfolge einer Workflow-Instanz kontrollieren. Aus Sicht des Transaktionsmanagements sind die operativen Transaktionen reine Leser (der Workflow-Definition). In [FRF02] wird ein *Multiversionsprotokoll* zur Koordination der Lesezugriffe durch operative Transaktionen und der Schreibzugriffe durch administrative Transaktionen beschrieben, das die Änderung von Teil-Workflows im laufenden Betrieb ermöglicht. Dieses Protokoll gewährleistet Korrektheit und Konsistenz und verhindert so das Auftreten der in der Literatur als Dynamic Change Bug [vdA01] bezeichneten Fehler.

Zu bemerken ist, dass hier nur Transaktionen betrachtet werden, die entweder die Erzeugung der Aktionsfolge einer Workflow-Instanz aus einer im Repository gespeicherten Definition oder die Änderung einer solchen Definition selbst betreffen. Ein transaktionaler Zugriff von Workflow-Instanzen auf gemeinsame Daten kann ebenfalls erreicht werden, wird hier aber nicht weiter berücksichtigt.

4 Multiversionsprotokoll

Eine Weiterentwicklung des erwähnten Multiversionsprotokolls setzt semantische Informationen ein, um die „Äquivalenz“ von Teilworkflow-Definitionen zu ermitteln. Hierdurch soll u.a. ermöglicht werden, dass ein Teilprozess S auch dann durch eine semantisch äquivalente, aber optimierte Version ersetzt werden kann, wenn er kontinuierlich wiederholt wird. In einem konventionellen Multiversionsprotokoll ist das nicht möglich, weil die Definition von S bereits vor dem ersten Durchlauf gelesen wurde und die Aktualisierung damit „zu spät“ kommt. Ein solcher Teilprozess kann z.B. das Durchsuchen einer Newsgroup nach bestimmten Inhalten sein.

Um die formale Korrektheit beweisen zu können, musste der klassische Multiversions-View-Serialisierbarkeitsbegriff [WV02, p.192], der die Gleichheit der *Reads-From* Relation mit einem Monoversions-Schedule auf derselben Transaktionsmenge fordert, erweitert werden. Wir schwächen diese Forderung ab und verlangen nun die Äquivalenz der *Reads-From* Relationen. Dieser neue Serialisierbarkeitsbegriff wird als *Semantische-Multiversions-Viewserialisierbarkeit* bezeichnet. Äquivalenz kann dabei z.B. über Zusicherungen vorgegeben oder durch Inferenz aus Ontologien abgeleitet werden.

Administrative Transaktionen (s. Abschnitt 3) legen bei jedem Schreiben einer Workflow-Definition eine neue Version an. Es wird zwischen den zu der bis dahin aktuellen Version semantisch äquivalenten und semantisch nicht äquivalenten Definitionen unterschieden. Äquivalente Definitionen werden in der Äquivalenzklasse der bis dahin aktuellen Version gespeichert, für die übrigen werden neue Äquivalenzklassen angelegt.

Operative Transaktionen versuchen den ihnen zugeordneten Timestamp bei jedem Datenzugriff zu aktualisieren. Der Timestamp darf über alle neuen Versionen geschoben werden, die nur Definitionen betreffen, die mit den bereits gelesenen äquivalent sind. Lesen darf das Protokoll jeweils die aktuellste Version einer Definition, die mit der letzten Version vor dem Timestamp semantisch äquivalent ist. Die operative Transaktion liefert alle möglichen Varianten als Resultat. Liegen verschiedene Varianten einer Teil-Prozess-Definition vor, wird die „richtige“ Variante aufgrund der Metadaten über die bereits ausgewählten Varianten und semantischen Informationen aus anderen Quellen, wie z.B. dem Benutzerprofil, ermittelt.

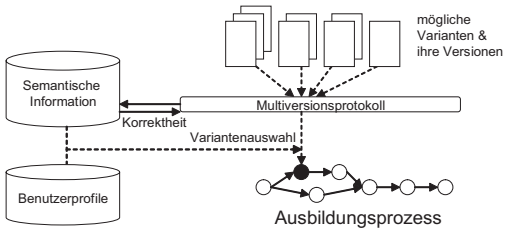

Abbildung 1: Einsatz des Protokolls

Das Multiversionsprotokoll von ULTRAflow ist auch in verteilten Systemen implementierbar. Es schließt ein gegenseitiges Blockieren von Lesern und Updatern aus, da Queries Updater erst nach ihrem Commit sehen. Updater regeln das konkurrente Schreiben von Versionen mit einem Zwei-Phasen-Sperrprotokoll. Operative Workflows werden niemals aufgrund einer Änderung unterbrochen. Weiterhin profitieren Workflows von Definitionsänderungen während ihrer Laufzeit. Es ist auch möglich, gewisse Workflows vor Updates zu schützten. „Junge“ Workflows profitieren wahrscheinlicher von Änderungen als „alte“.

5 Diskussion

ULTRAflow bietet die Möglichkeit die Definition von Workflows modular aufzubauen und unterscheidet sich von Ansätzen, die auf speziellen Petrinetzklassen basieren wie ADEPT [RRD02] oder van der Aalsts Vorschlag [vdA01, vdAB02]. Einen modularen Aufbau findet man in der Regel nur bei objektorientierten Ansätzen. So baut z.B. MOKASSIN [Jo99] seine Spezifikationen feingranular auf, um Multiversions-fähig zu werden. TRAM [KG99] baut Workflowspezifikationen aus ’Complex Types’ und ’Activity Types’ auf. Bei der Einbeziehung semantischer Information geht ULTRAflow über andere Vorschläge (Flow Conditions bei TRAM [KG99], Event-Condition-Action in MOKASSIN [Jo99]) hinaus.

METEOR-S [CS03] verbindet auch Workflow Management Systeme mit Web Services. Dort wird semantische Information zur Web Service Discovery eingesetzt.

6 Zusammenfassung

Es wurde ein Ansatz für ein Workflowmanagement System vorgestellt, das zur effizienten und konsistenten Prozesssteuerung in Webservices eingesetzt werden kann. Durch die Einbeziehung semantischer Information kann im Rahmen der vordefinierten Varianten eine dynamische Anpassung der Abläufe erreicht werden.

Literatur

[CS03] Cardoso, J. und Sheth, A.: Semantic e-workflow composition. *Journal of Inteligent Information Systems*. 21:3:191–225. 2003.

[FF01] Fent, A. und Freitag, B.: ULTRAflow – a lightweight workflow management system. *Proceedings of the International Workshop on Functional and (Constraint) Logic Programming (WFLP2001), September 13–15, Kiel, Germany*. S. 375–378. 2001.

[FRF02] Fent, A., Reiter, H., und Freitag, B.: Design for change: Evolving workflow specifications in ULTRAflow. In: Pidduck, A. B., Mylopoulos, J., Woo, C. C., und Özsu, M. T. (Hrsg.), *Advanced Information Systems Engineering, 14th International Conference, CAiSE 2002, Toronto, Canada, May 27-31, 2002, Proceedings*. volume 2348 of *LNCS*. S. 516–534. Springer-Verlag. 2002.

[FWF00] Fent, A., Wichert, C.-A., und Freitag, B.: Logical update queries as open nested transactions. In: *Transactions and Database Dynamics*. volume 1773 of *LNCS*. S. 45 – 66. Springer-Verlag. Berlin. 2000.

[Jo99] Joeris, G.: Defining flexible workflow execution behaviors. *Workshop Proceedings - Workshop Informatik '99. Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications. Paderborn*. 1999.

[KG99] Kraldorfer, M. und Geppert, A.: Dynamic workflow schema evolution based on workflow type versioning and workflow migration. *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems CoopIS'99*. S. 104–114. 1999.

[RRD02] Rinderle, S., Reichert, M., und Dadam, P.: Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata. *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems CoopIS'99*. 2002. http://www.informatik.uni-ulm.de/dbis/.

[vdA01] van der Aalst, W. M. P.: Exterminating the dynamic change bug. a concrete approach to support workflow change. *Information System Frontiers*. 3(3):297–317. 2001.

[vdAB02] van der Aalst, W. M. P. und Basten, T.: Inheritance of workflows; an approach to tackling problems related to change. *Theoretical Computer Science*. 270:125–203. 2002.

[WV02] Weikum, G. und Vossen, G.: *Transactional Information Systems. Theory, Algorithms and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann. 2002.