Schemaevolution in Workflow-Management-Systemen

Stefanie Rinderle · Peter Dadam

Ein Schema (Muster, Entwurf) dient der modellhaften Beschreibung realer Sachverhalte. Mit Hilfe solcher Modelle kann der Mensch in seiner täglichen Arbeit unterstützt werden. So gibt es z. B. *Datenbankschemata* zur Beschreibung von Daten und deren Beziehung untereinander und *Workflow-Schemata* zur Modellierung von Arbeitsprozessen (engl. workflow).

Ein Workflow-Schema legt unter anderem fest, welche Aktivitäten in welcher Reihenfolge zur Ausführung kommen. Dadurch ist für Workflow-Instanzen, die ausgehend von diesem Workflow-Schema erzeugt und ausgeführt werden, klar, ob sie parallele oder alternative Ausführungspfade aufweisen und welche externen Programme ggf. mit einer bestimmten Aktivität verknüpft sind. Durch das Workflow-Schema sind also bereits zur Modellierungszeit alle zur Laufzeit möglichen Ausführungsvarianten bestimmt.

Sowohl Datenbank- als auch Workflow-Schemata beschreiben in der Regel Sachverhalte aus der realen Welt. Wenn sich diese Sachverhalte ändern oder neue Aspekte hinzukommen, müssen unter Umständen auch die Schemata angepasst werden. Bei der Evolution von Datenbank-Schemata geht es dabei fast ausschließlich darum, die Datentypen, Datenstrukturen und Integritätsbedingungen des „alten" Datenbank-Schemas (möglichst) semantikerhaltend auf die entsprechenden Datentypen, Datenstrukturen und Integritätsbedingungen des „neuen" Datenbank-Schemas abzubilden.

Dagegen bedeutet im einfachsten Fall (!) eine Workflow-Schema-Änderung, dass es für eine gewisse Zeit Workflow-Instanzen geben wird, die noch nach dem alten Schema, und solche, die (weil nach der Schemaänderung gestartet) nach dem neuen Schema abgewickelt werden. In vielen Fällen reicht dieses einfache Koexistenzmodell jedoch nicht aus, sei es, weil sich z. B. gesetzliche Rahmenbedingungen geändert haben oder weil das alte Workflow-Schema gravierende Mängel aufweist.

Soll eine Schemaänderung also auch auf die bereits laufenden Instanzen angewendet werden, spricht man von Workflow-Schemaevolution [1, 6]. Um zu verstehen, was bei einer Workflow-Schemaevolution zu leisten ist, stellt man sich eine Workflow-Instanz am besten als ein in Ausführung befindliches Programm vor, das aus einer Folge von Prozeduren besteht. Diese werden entweder nacheinander gerufen (sequenzielle Ausführung) oder treten in einer If-then-else-Bedingung auf (alternative Verzweigung) oder können auch parallel zur Ausführung kommen. Bei einer Workflow-Schemaevolution wird nun versucht, in einem solchen Programm ein oder mehrere neue Prozeduren einzufügen, zu löschen, zu verschieben, Verzweigungsbedingungen zu ändern, parallele Pfade hinzuzufügen oder zu löschen; und zwar so, dass eine zuvor korrekte Parameterversorgung aller Prozeduren unter allen Ausführungsalternativen auch nach der Änderung gewährleistet ist. Hierbei ist zu beachten, dass diese Instanzen alle unterschiedlich weit in ihrer Ausführung sind und dass die Anwendbarkeit einer Schemaevolution auf eine bestimmte Workflow-Instanz davon abhängt, wie weit deren Ausführung bereits fortgeschritten ist [6, 7]. So macht z. B. die Einfügung des Schrittes „Sendung kontrollieren" in Abb. 1 gemäß des neuen Workflow-Schemas nur für solche Workflow-Instanzen Sinn, bei denen der Schritt „Ware versenden" noch nicht ausgeführt worden ist.

Den oben beschriebenen Sachverhalt (Propagation von Schemaänderungen

Stefanie Rinderle, Peter Dadam
Abt. Datenbanken und Informationssysteme, Universität Ulm
E-Mail: {rinderle, dadam}@informatik.uni-ulm.de

* Vorschläge an Prof. Dr. Frank Puppe <puppe@informatik.uni-wuerzburg.de> oder Dieter Steinbauer <dieter.steinbauer@schufa.de>
Alle „Aktuellen Schlagwörter" seit 1988 finden Sie unter: www.ai-wuerzburg.de/as

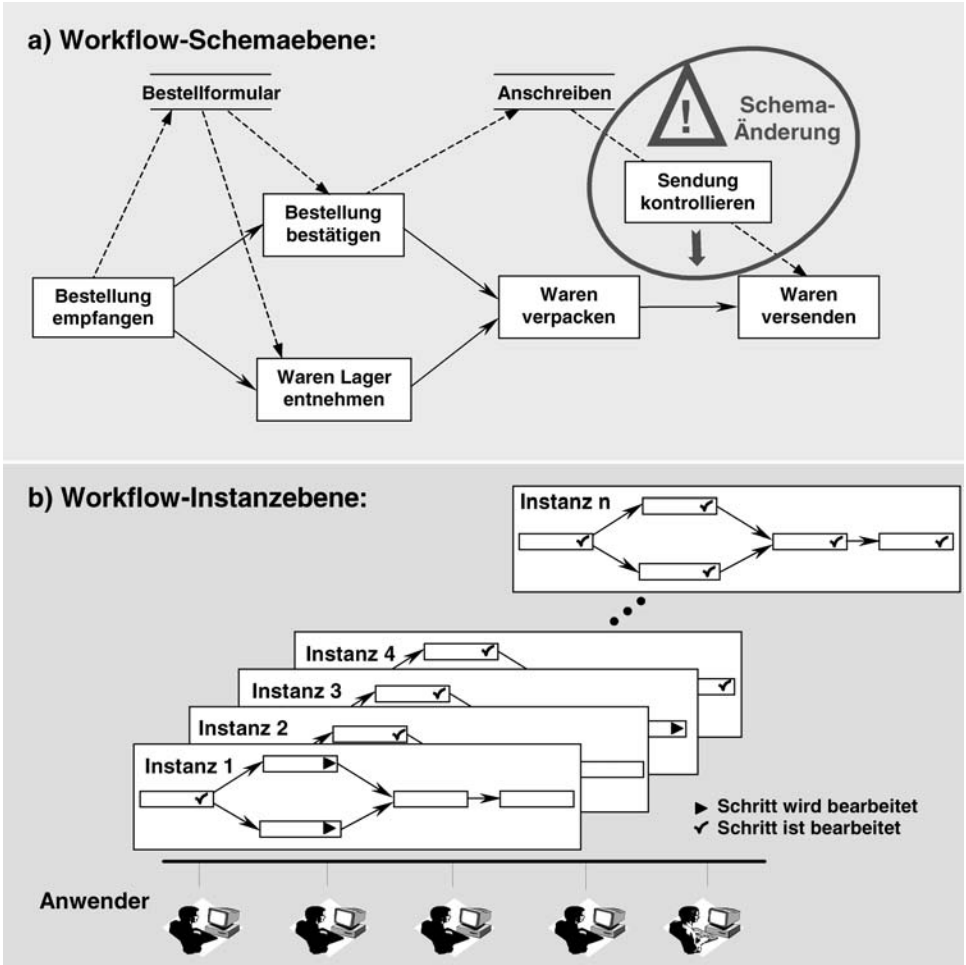

Abb. 1 Bestellprozess

auf laufende Instanzen) bezeichnet man in der Literatur auch als *dynamischen Fall* [3]. Grundlegende Herausforderungen sind hier, zu jedem Zeitpunkt Korrektheit und Konsistenz zu gewährleisten und die Migrationen – besonders bei einer großen Anzahl von Workflow-Instanzen – automatisch und effizient durchzuführen [6]. Betrachtet man dagegen nur die erforderlichen Strukturtransformationen ohne Berücksichtigung des Zustands, spricht man vom *statischen Fall*.

In der Forschung befasst man sich schon seit einiger Zeit mit Workflow-Systemen, die es erlauben, im Einzelfall ad hoc vom geplanten Ablauf abzuweichen [2, 4, 5]. Nur mit einer Funktionalität dieser Art werden Workflow-Management-Systeme wirklich breit einsetzbar werden. Die große Herausforderung besteht darin, eine Schemaevolution der oben beschriebenen Art zukünftig auch auf Instanzen anwendbar zu machen, deren „Instanz"-Schema durch Ad-hoc-Änderungen inzwischen vom Originalschema des Workflow-Typs abweicht. Die spannende Frage hier ist, welche Propagationen man noch zulassen soll und welche nicht mehr. Wenn eine Ad-hoc-Änderung die Schemaänderung beispielsweise bereits vorweggenommen hat, dann sollte die Schemaänderung natürlich nicht mehr auf diese Instanz angewendet werden. Betreffen Schemaänderung und Ad-hoc-Änderung dagegen unterschiedliche Bereiche einer Workflow-Instanz, sollte eine Propagation sehr wohl – wenn sonst nichts dagegen spricht – erfolgen können.

Zusammenfassend kann gesagt werden, dass Schemaevolution in Workflow-Management-Systemen aufgrund des dynamischen Charakters von Prozessen neuartige und interessante Fragestellungen aufwirft. Diese zu lösen, wird für zukünftige prozessorientierte Informationssysteme essenziell sein, damit die realisierten Anwendungen rasch und kostengünstig an neue Gegebenheiten angepasst werden können.

Literatur

1. van der Aalst, W.M.P., Basten, T.: Inheritance of workflows: an approach to tackling problems related to change. Theoretical Computer Science, 2002
2. van der Aalst, W.M.P., Jablonski, S.: Dealing with workflow change: Identification of issues and solutions. Int. J. Computer Syst., Science and Engineering 15(5), 267–276 (2000)
3. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Workflow evolution. Data and Knowledge Engineering 24(3), 211–238 (1998)
4. Müller, R., Rahm, E.: Dealing with logical failures for collaborating workflows. Proc. Int. 5th Conf. on Coop. Inf. Syst., Eilat, 2000, pp. 210–223
5. Reichert, M., Dadam, P.: $ADEPT_{flex}$ – Supporting dynamic changes of workflows without losing control. J. Intelligent Inf. Syst. 10(2), 93–129 (1998)
6. Rinderle, S., Reichert, M., Dadam, P.: Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata. Informatik – Forschung und Entwicklung 17(4), 177–197 (2002)
7. Sadiq S., Marjanovic O., Orlowska M.: Managing change and time in dynamic workflow processes. The Int. Journal of Coop. Inf. Syst. 9(1, 2) (2000)