Integrierte Dokumenten- und Ablaufmodellierung von E-Business-Prozessen

Kirsten Lenz, Andreas Oberweis

Lehrstuhl für Entwicklung betrieblicher Informationssysteme
J.W. Goethe-Universität Frankfurt/Main
Postfach 11 19 32
D-60054 Frankfurt/Main
{lenz|oberweis}@wiwi.uni-frankfurt.de

Abstract: Aufgrund der starken Zunahme internetbasierter E-Business-Aktivitäten werden Sprachen für die Modellierung von E-Business-Prozessen immer wichtiger. Ziel der Modellierung ist die formale Darstellung der organisationsübergreifenden Prozesse und aller prozeßrelevanten Objekte. Zur integrierten Modellierung von E-Business-Prozessen und den prozeßrelevanten Dokumenten wird daher eine neue Variante höherer Petri-Netze, die sogenannten XML-Netze, vorgeschlagen. XML-Netze sind eine formale, graphische Beschreibungssprache, die einerseits die Modellierung des XML-Dokumentenflusses und andererseits die Modellierung des Kontrollflusses des zugrundeliegenden Geschäftsprozesses ermöglicht. XML-Netze basieren auf dem XML-Schema-Modell, einer neuen graphischen Beschreibungssprache für XML-Schemata, und der an das XML-Schema-Modell angelehnten graphischen Dokumentenmanipulationssprache XManiLa. Operationen auf XML-Dokumenten werden durch sogenannte Filterschemata beschrieben.
XML-Netze können zunächst für den Entwurf der E-Business-Prozesse verwendet werden. Es können sowohl bestehende E-Business-Prozesse als auch Prozeßvarianten abgebildet und hinsichtlich struktureller Abweichungen untersucht werden. Aufgrund ihrer formalen Definition erlauben XML-Netze eine simulative Validierung der abgebildeten E-Business-Prozesse und können durch eine entsprechende Workflow-Engine direkt ausgeführt werden.

1 Einleitung

Aufgrund der starken Zunahme internetbasierter E-Business-Aktivitäten werden Sprachen für die Modellierung von E-Business-Prozessen immer wichtiger. Bei der Modellierung sollen die realen E-Business-Prozesse in einer für den Anwender verständlichen, aber –im Gegensatz zur natürlichsprachlichen Beschreibung– unmißverständlichen Notation, abgebildet werden. Ziel der Modellierung ist die formale Darstellung der organisationsübergreifenden Prozesse und aller prozeßrelevanten Objekte. Die formalen Prozeßbeschreibungen bilden die Basis für eine systematische Analyse, die Simulation und die Ablaufunterstützung der Prozesse. Bei der Ausführung der Prozesse müssen E-Business-Objekte auf elektronischem Wege zwischen den am Prozeß beteiligten Unternehmen ausgetauscht werden [Mer99]. Die prozeßrelevanten E-Business-Objekte können einfach strukturierte Objekte sein, bei denen es sich meist um Massendaten handelt, die unternehmensintern für die

Durchführung des Prozesses benötigt werden und die üblicherweise in relationalen Dankenbanken verwaltet werden, beispielsweise Kundennummern oder die Stückzahlen von gelagerten Produkten. Komplex strukturierte Objekte wie Bestellungen, Rechnungen oder auch Personalakten sind u.a. Dokumente, die zwischen den Unternehmen oder den Prozeßbeteiligten eines Unternehmens ausgetauscht und bei der Durchführung des Prozesses gelesen oder bearbeitet werden. Die entsprechenden elektronischen Dokumente können mit Hilfe der Auszeichnungssprache *Extensible Markup Language* (XML) [HaM01, Ray01] strukturiert werden. Der XML-Standard [W3C00] bildet dabei die Basistechnologie für verschiedene Sprachen, beispielsweise *XML Schema* zur Beschreibung von Dokumentenklassen [W3C01a].

Aus prozeßorientierter Sicht sind die Modellierung, die Analyse, die (Um-)Gestaltung und die automatisierte Ausführung der Prozesse von Bedeutung [Jab00, Obe96]. Petri-Netze [ReR98] sind eine Beschreibungssprache für Abläufe, welche die Vorteile einer graphischen Darstellung und einer formalen Semantik vereint. Für eine Menge konkreter Abläufe können Ablaufschemata durch Petri-Netze in einer für den Anwender verständlichen Form dargestellt werden [DeO96]. Dabei unterschieden sich die einzelnen höheren Petri-Netz-Typen [EHP02] in der Interpretation der ablaufrelevanten Daten. Prädikate/Transition-Netze [Gen86], bei denen die prozeßrelevanten Objekte als Relationen einer Datenbank interpretiert werden können, haben sich bereits bei dem Entwurf von Informationssystemen bewährt. Sie sind jedoch für die Beschreibung von Prozessen basierend auf XML-Dokumenten wenig geeignet.

Zur integrierten Modellierung von E-Business-Prozessen und den prozeßrelevanten Dokumenten wird daher eine neue Variante höherer Petri-Netze, die sogenannten *XML-Netze* [LeO02], vorgeschlagen. Die Definition der XML-Netze erfordert eine formale graphische Beschreibungssprache für die prozeßrelevanten Dokumente, mit deren Hilfe die Struktur der Dokumente und Wertebereiche für deren Inhalt durch ein Schema vorgegeben werden können. Darüber hinaus werden die durchzuführenden Aktivitäten auf den entsprechenden XML-Dokumenten ebenfalls graphisch in einer ähnlichen Notation dargestellt. Um die verteilte Ausführung der Prozesse zu unterstützen, können die Ablaufmodelle in einzelne Prozeßteile, die Prozeßfragmente, zerlegt werden.

XML-Netze können mit unterschiedlichen Zielsetzungen zur Unterstützung von E-Business-Prozessen verwendet werden: Sie können zunächst für den Entwurf der Prozesse eingesetzt werden. Darüber hinaus kann das Verhalten der XML-Netze simuliert werden. Durch Analyse und Simulation der XML-Netze kann auch die Verbesserung der E-Business-Prozesse unterstützt werden. XML-Netze können sowohl bestehende E-Business-Prozesse als auch Prozeßvarianten abbilden und hinsichtlich struktureller Abweichungen untersucht werden.

Der Beitrag ist folgendermaßen gegliedert: Zunächst wird das XML-Schema-Modell, eine Beschreibungssprache für Schemata für XML-Dokumente, vorgestellt. Danach wird die auf dem XML-Schema-Modell basierende Dokumentenmanipulationssprache XManiLa erläutert. Dazu werden Filterschemata eingeführt, welche als eine Art "Schablone" für die Anfrage-relevanten XML-Dokumente dienen. In Kapitel 4 werden XML-Netze an einem Beispiel erläutert.

2 Konzeptuelle Modellierung von E-Business-Objekten

Für die konzeptuelle Modellierung der E-Business-Prozesse ist es zunächst wichtig, die prozeßrelevanten Objekte zu modellieren. Für die relevanten Daten werden Methoden der konzeptuellen Datenmodellierung wie das Entity/Relationship-Modell (E/R-Modell) [Che76] verwendet. Die Modellierung komplexer Objekte, beispielsweise von XML-Dokumenten, wird bislang nicht ausreichend durch graphische Methoden unterstützt.

2.1 Anforderungen an eine Beschreibungssprache für die Modellierung von XML-Schemata

Die XML Schema-Spezifikation [W3C01a] ist sehr komplex und besitzt keine graphische Notation. Die konzeptuelle Modellierung von XML-Schemata dient dazu, Sachverhalte der Realwelt auf einem hohen Abstraktionsniveau abzubilden und damit die Schemata möglichst unabhängig von der Implementierung zu entwerfen. Ziele bei der Entwicklung einer Beschreibungssprache für den konzeptuellen Entwurf von XML-Schemata sind,

- ein formale Definition für XML-Schemata, welche die Menge aller zulässigen Modellierungskonstrukte und deren Beziehungen vorgibt, aufzustellen,
- eine graphische Darstellung für alle Modellierungskonstrukte anzubieten, die den Bedürfnissen des Modellierers entsprechend angepaßt werden kann,
- den stufenweisen Entwurf der Schemata durch Schematransformationen zu unterstützen und
- dabei auf bekannte Modellierungskonzepte zurückzugreifen, um erfahrenen Modellieren eine leicht erlernbare Beschreibungssprache zu Verfügung zu stellen und somit die Akzeptanz der Beschreibungssprache zu erhöhen.

Als Grundlage der Definition des XML-Schema-Modells (XSM) wurden die Klassendiagramme der UML (siehe beispielsweise [BRJ99]) gewählt. Verwandte Ansätze zur Modellierung von XML-Schemata sind in [BCF99, CSF00, MLM01, MeH01] zu finden.

2.2 Das XML-Schema-Modell

XML-Schemata für eine Klasse von XML-Dokumenten mit ähnlicher Struktur können mit Hilfe des XML-Schema-Modells entworfen und durch Diagramme, die sogenannten XML-Schema-Diagramme (XSD), graphisch dargestellt werden. Nachfolgend werden die einzelnen Konstrukte des XML-Schema-Modells vorgestellt. Auf die formale Definition wird dabei jedoch aus Platzgründen verzichtet.
Alle Elemente mit gleichen Eigenschaften, d.h. mit ähnlicher Struktur, werden zu einem Elementtypen zusammengefaßt. Jedem Elementtypen werden ein (eindeutiger) Name, eine Menge von Attributen und ein Datentyp, der mögliche Inhalte der Elemente beschreibt, zugeordnet. Elementtypen, die keinen Inhalt besitzen, werden als leere Elementtypen bezeichnet. Den Attributen eines Elementtyps wird ebenfalls jeweils ein Datentyp zugeordnet, der den Wertebereich des Attributs bestimmt. Graphisch werden

Elementtypen durch UML-Klassen repräsentiert, in denen auch die Attribute angegeben werden können.
Assoziationen beschreiben die Beziehungen zwischen den Elementen. Gleiche Assoziationen, d.h. Assoziationen derselben Art zwischen Elementen derselben Elementtypen, werden zu Assoziationstypen klassifiziert. Assoziationstypen dienen dazu, die Bildung eines komplexen Elementtyps aus anderen (komplexen oder atomaren) Elementtypen zu beschreiben. Die Konstruktionsvorschrift gibt für den Assoziationstypen an, wie der Inhalt der Elemente des Supertyps aus den Elementen der Subtypen gebildet wird. Zulässige Konstruktionsvorschriften für den Inhalt eines Elements des Supertypen sind die Sequenz, die Auswahl und die Permutation. Graphisch werden Assoziationstypen durch UML-Assoziationen, den Aggregationen, dargestellt (Abbildung 1).

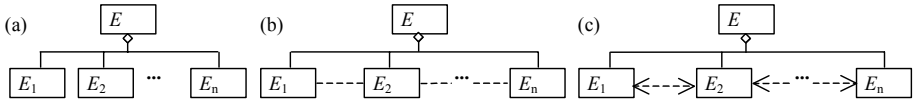

Abbildung 1: Graphische Darstellung von (a) Sequenztyp, (b) Auswahltyp und (c) Permutationstyptyp

Jedem Elementtypen, der Subtyp eines Assoziationstypen ist, wird die entsprechende Kardinalität bezüglich dieses Assoziationstyps als Paar einer Minimumskardinalität und einer Maximumskardinalität angegeben. Die Kardinalität beschreibt, aus wie vielen Elementen dieses Typs sich ein Element des Supertyps zusammensetzen kann[1].

Beispiel: Abbildung 2 zeigt den Sequenztypen mit dem Supertypen *Kunde* für Kundendokumente einer Buchhandlung. Der Inhalt aller Elemente dieses Typs setzt sich folgendermaßen zusammen: Zuerst gibt es genau ein Element vom Typ *Kundennummer*, dann ein oder zwei Elemente vom Typ *Adresse*. Danach können beliebig viele Elemente vom Typ *BestelltesBuch* folgen, es muß aber kein solches Element vorkommen.

□

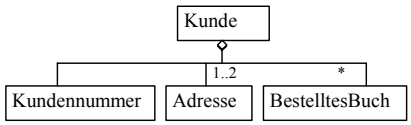

Abbildung 2: Sequenztyp mit drei Subtypen unterschiedlicher Kardinalität

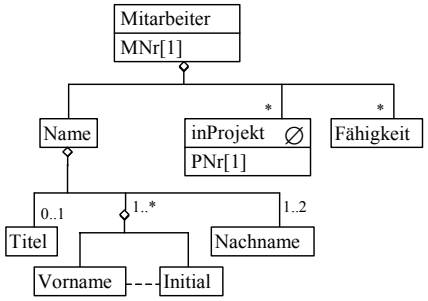

Abbildung 3: Elementares XML-Schema für Mitarbeiterdokumente

[1] In der Mathematik ist die Kardinalität als die Anzahl der Elemente einer Menge definiert. Hier wird der Begriff "Kardinalität" benutzt, um die Anzahl von aufeinanderfolgenden Elementen desselben Typs zu beschreiben, nicht also einer Menge von Elementen im streng mathematischen Sinn.

Durch den Einsatz von abstrakten Elementtypen können komplexe Elementtypen mittels Schachtelung von Assoziationstypen deklariert werden. Abstrakte Elementtypen klassifizieren Gruppierungen von Elementen und können in einer verkürzten Darstellung des XML-Schema-Diagramms weggelassen werden (beispielsweise im Elementtyp *Name* des XML-Schemas aus Abbildung 3).
Einen Spezialfall der XML-Schemata stellen die elementaren XML-Schemata dar, die sich aus Elementtypen und darauf definierten Assoziationstypen mit paarweise verschiedenen Supertypen zusammensetzen. Insbesondere existiert jeweils genau ein top-level-Elementtyp[2]. Abbildung 3 zeigt ein elementares XML-Schema, das XML-Dokumente für Mitarbeiter beschreibt.

3 Anfragen an XML-Dokumente

Für eine Beschreibungssprache für dokumentenbasierte Geschäftsprozesse ist eine möglichst einheitliche Notation zur Definition von Schemata und die Spezifikation der Anfrage vorteilhaft, um den Entwurf von Abläufen mit Petri-Netzen so einfach wie möglich zu gestalten. Der für Geschäftsprozesse sinnvolle integrierte Ablauf- und Dokumenten- bzw. Datenentwurf erfordert neben der graphischen Notation zur Ablaufmodellierung und zum Entwurf der XML-Schemata auch eine graphische Notation zur Formulierung der Anfragen und der Dokumentenmanipulation.
Die Dokumentenmanipulationssprache XManiLa dient zur Selektion und Manipulation von XML-Dokumenten und ist eine Erweiterung des XML-Schema-Modells. Andere Anfragesprachen sind meistens textbasiert [AQH97, BMN02, DFF99, RLS98, W3C01b]. Bei XML-GL [CDF01], einer graphischen XML-Anfragesprache, liegt der Fokus auf der Erzeugung der Ergebnisdokumente für die jeweilige Anfrage.
Die Grundlage für XManiLa bilden die sogenannten Filterschemata, die in ihrer Notation an XML-Schemata angelehnt sind. Verglichen mit den rein strukturbeschreibenden XML-Schemata ermöglichen Filterschemata zusätzlich die Zuweisung von Variablen oder Konstanten zu Elementtypen oder Attributen, die Auszeichnung eines Elements, das verändert werden soll (je nach Kontext, in dem das Filterschema eingesetzt wird, eingefügt oder gelöscht) und den Einsatz eines Elementplatzhalters an den Stellen, an denen der Inhalt der Dokumente für die Anfrage nicht relevant ist, weil er weder für die Auswahl eines Dokuments benötigt wird, noch Änderungen unterworfen ist.

3.1 XML-Schemata und Filterschemata

Filterschemata stellen Schablonen für Dokumente und deren Elemente dar. Sie setzen sich aus Elementfiltern entsprechend den Elementtypen eines XML-Schemas zusammen. Für die Attribute von Elementtypen können Attributfilter eingesetzt werden. Attributfilter und Elementfilter können entweder Variablen oder konstante Werte enthalten. Die Konstanten stammen aus dem Wertebereich des entsprechenden Attri-

[2] Ein top-level-Elementtyp ist ein Elementtyp, der nicht Subtyp eines Assoziationstypen ist.

buts oder des Elementtyp-Inhalts. Sie können daher für Attribute oder atomare Elementtypen als einfache Werte, beispielsweise als Zeichenkette oder Zahl, angegeben werden. Die Elementfilter können mit Hilfe von Sequenz, Auswahl oder Permutation zu komplexen Elementfiltern zusammengesetzt werden. Ein Filterschema ist vergleichbar aufgebaut wie ein elementares XML-Schema. Die graphische Darstellung eines Filterschemas wird auch als Filterdiagramm bezeichnet.
Ein Filterschema ist zu einem bestimmten XML-Schema zulässig, wenn alle durch das Filterdiagramm beschriebenen XML-Dokumente gültig bezüglich dieses XML-Schemas sind. In Filterdiagrammen zulässiger Filterschemata können Elementtypnamen und Attributnamen weggelassen werden, wenn sie durch die Zuordnung des entsprechenden XML-Schema-Diagramms eindeutig sind. Abbildung 4 (a) zeigt ein zu dem Mitarbeiter-Schema aus Abbildung 3 zulässiges Filterdiagramm, welches das Mitarbeiterdokument aus Abbildung 4 (b) beschreibt.
Elementfilter können nicht nur als Schablone für Elemente eines Typs verwendet werden, sondern auch, um Elemente zu manipulieren, d.h. sie zu löschen oder zu erzeugen. In der graphischen Repräsentation wird der sogenannte Manipulationsfilter von den anderen Elementfiltern, auch Vergleichsfilter genannt, durch einen schwarzen Balken an der linken Kante des Rechtecks unterschieden.

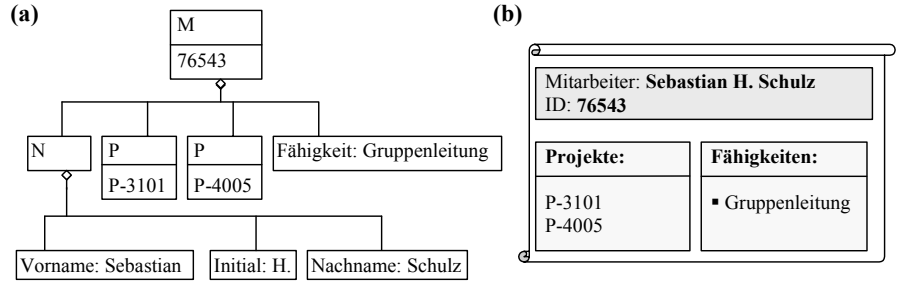

Abbildung 4: Zulässiges Filterschema zum XML-Schema aus Abbildung 3

3.2 Anfragen mit XManiLa

Filterschemata alleine stellen noch keine Dokumentenanfragen oder -manipulationen dar. Die Anfragesprache XManiLa basiert auf der Kombination von XML-Schemata, Filterschemata und Konzepten der Petri-Netze. Eine Anfrage mit XManiLa besteht allgemein aus einer Stelle, einer Transition und einer Kante zwischen Stelle und Transition. Ein elementares XML-Schema dient zur Typung der Stelle. Alle für die Anfrage relevanten Dokumente müssen bezüglich dieses Schemas gültig sein. Als Kantenbeschriftung muß ein bezüglich des XML-Schemas zulässiges Filterschema gewählt werden. Die Richtung der Kante spezifiziert, ob es sich um eine Einfügeoperation oder eine Löschoperation handelt (siehe Abbildung 5).
Nachfolgend werden Beispiele für das Erzeugen eines neuen Mitarbeiterdokuments, das Löschen eines Elements aus einem bestehenden Mitarbeiterdokument und das Lesen von XML-Dokumenten zum XML-Schema aus Abbildung 3 vorgestellt.

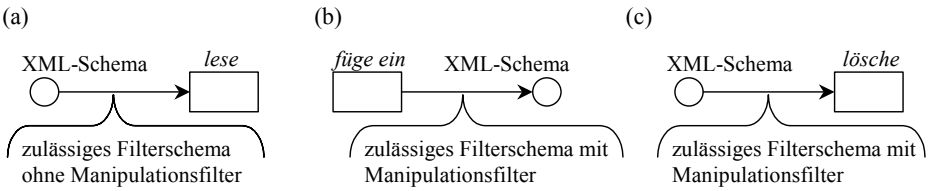

Abbildung 5: Allgemeine Anfragen mit XManiLa: (a) Lesen, (b) Einfügen und (c) Löschen

Beispiel: Erzeuge ein Mitarbeiterdokument für die neue Mitarbeiterin Henriette Meier mit der Nummer "63941". Das Dokument wird zur Menge der Mitarbeiterdokumente hinzugefügt. Die XManiLa-Anfrage ist in Abbildung 6 zu sehen.

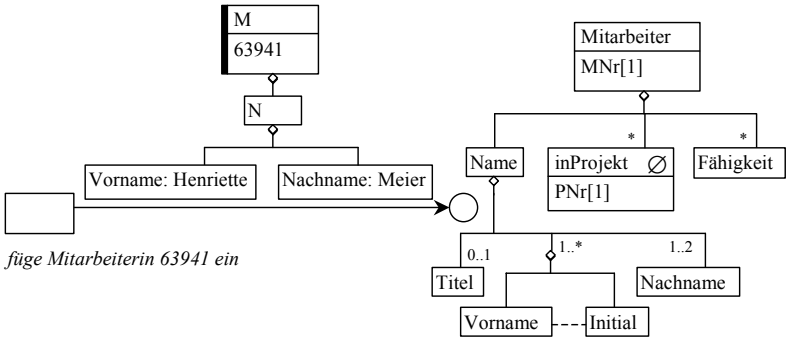

Abbildung 6: Erzeugen eines neuen Mitarbeiterdokuments

□

Beispiel: Mitarbeiter "76543" hat die Mitarbeit in Projekt "P-3101" beendet. Das Element `inProjekt` mit dieser Projektnummer soll aus seinem Dokument gelöscht werden, alle anderen Informationen bleiben erhalten.

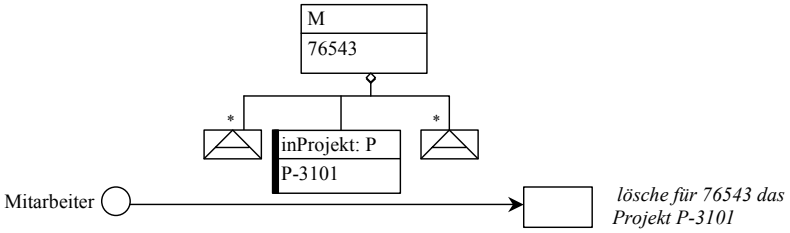

Abbildung 7: Löschen eines atomaren Elements aus einem Dokument

Das dazu notwendige Filterdiagramm zeigt Abbildung 7. Bei dieser XManiLa-Anfrage ist die Richtung der Kante entgegengesetzt zu der für Einfüge-Operationen, das Filterdiagramm spezifiziert daher das Entfernen von Dokumententeilen.

□

Beispiel: Gesucht wird ein Mitarbeiter, der entweder in Projekt P-3013 oder in Projekt P-4005 mitarbeitet. Für alle anderen Elementtypen als `inProjekt` werden im Filterdiagramm Elementplatzhalter eingesetzt, um zu beschreiben, daß vor und nach dem gesuchten Element beliebige andere Elemente stehen können. Die entsprechende XManiLa-Anfrage ist in Abbildung 8 zu sehen.

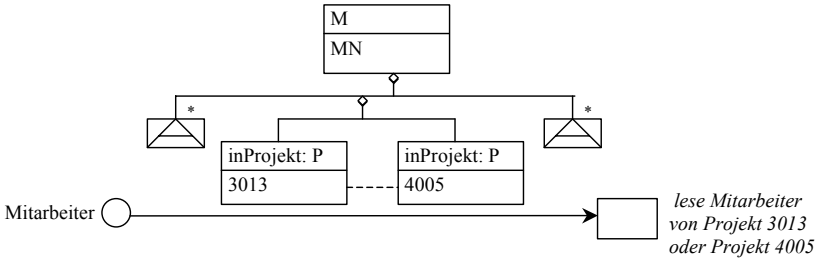

Abbildung 8: Suchen nach einem Element mit alternativem Inhalt

□

4 XML-Netze

Insbesondere Geschäftsprozesse im E-Business werden durch die bestehenden Ablauf-Beschreibungssprachen bislang unzureichend unterstützt. Ablauforientierte Beschreibungssprachen ermöglichen zwar die Modellierung des Ablaufs, aber nicht die integrierte Darstellung komplexer Objekte. Zur Nutzung dieser Beschreibungssprachen müssen komplexe Objekte entweder als unteilbares Ganzes behandelt oder auf relationale Daten heruntergebrochen werden. Im ersten Fall kann nicht mehr auf Teile des Dokuments zugegriffen werden, im zweiten Fall kann das Dokument nicht mehr als Ganzes identifiziert werden, was u.a. die Manipulation der Dokumente erheblich erschwert. Andere Beschreibungssprachen konzentrieren sich auf die Modellierung der komplexen Objekte und der Operationen auf diesen Objekten. Sie sind jedoch nicht für die Modellierung eines umfassenden Ablaufs und seiner Dynamik geeignet.
XML-Netze stellen einen neuen Typ höherer Petri-Netze dar. Sie setzen sich aus den üblichen Petri-Netz-Bestandteilen mit der nachfolgenden Interpretation zusammen: Jede Stelle in einem XML-Netz kann als Behälter für eine Menge von strukturierten Objekten zu einem gemeinsamen Schema, dem Stellentyp, interpretiert werden. Die Stellentypung erfolgt in einem XML-Netz im allgemeinen[3] durch ein elementares XML-Schema. Die Transitionen sind über gerichtete Kanten mit den Stellen verbunden. Sie repräsentieren Aktivitäten, die jeweils eine Klasse von Operationen auf den Dokumenten der adjazenten Stellen definieren. Jeder Transition kann eine Transitionsinschrift, bestehend aus einem logischen Ausdruck, hinzugefügt werden. Jeder Kante wird eine Kanteninschrift zugewiesen. Die Kanteninschrift präzisiert die Operation, die auf den Dokumenten der adjazenten Kante durchgeführt wird. Sie ordnet der Kante eine Menge von zu dem XML-Schema der adjazenten Stelle zulässigen Filterschemata zu. Wird die Menge der in den Filterschemata auftretenden Variablen belegt, dann repräsentieren die konstanten Filterschemata die entsprechenden Dokumente der Stelle vor bzw. nach Durchführung der Aktivität. Zu jedem Zeitpunkt kann für jede Transition entschieden werden, ob sie bezüglich einer Variablenbelegung aktiviert ist, d.h. ob

[3] XML-Netze können ohne Beschränkung der Allgemeinheit Teile von Prädikate/Transitionen-Netzen enthalten. Für Stellen dieser Netzteile geschieht die Stellentypung durch Relationenschemata. Die Stellen werden mit Relationen zu dem jeweiligen Schema markiert.

die Durchführung der Aktivität möglich ist. Durch die Schaltregel ist definiert, wie die entsprechenden Dokumente beim Schalten aktivierter Transitionen verändert werden.

Beispiel: Der folgende Ablauf beschreibt einen Teil des Lebenszyklusses eines Versicherungsvertrags in einem Versicherungsunternehmen:

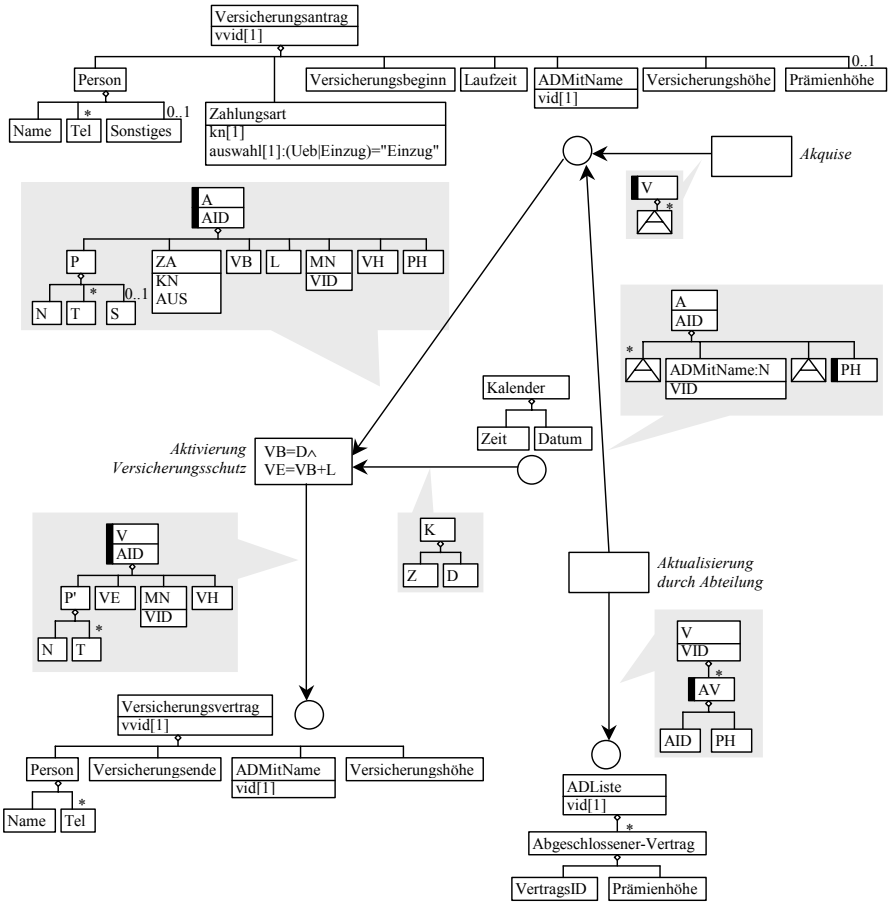

Abbildung 9: XML-Netz für den Beginn eines Versicherungsverhältnisses

Bei der Akquise durch einen Außendienstmitarbeiter wird ein Versicherungsantrag aufgesetzt. Versicherungsanträge enthalten die Personalien des Antragstellers, die gewünschte Zahlungsart (Überweisung oder Lastschriftverfahren), den Versicherungsbeginn, die vereinbarte Laufzeit und die Versicherungshöhe. Der Außendienstmitarbeiter gibt zusätzlich seinen Namen und seine Kennung an. Das Versicherungsunternehmen muß nach Eingang des Antrags die Prämienhöhe festlegen, die dem Antrag hinzugefügt wird. Gleichzeitig wird die Liste des Außendienstmitarbeiters aktualisiert.

Am Tag des vereinbarten Versicherungsbeginns tritt die Versicherung in Kraft, der Versicherungsschutz wird aktiviert. Das Datum, an dem die Versicherung ausläuft, bestimmt sich automatisch durch die vereinbarte Laufzeit. Für den Kunden wird ein Versicherungsvertrag ausgefertigt, der seinen Namen, seine Telefonnummern, das

Datum, an dem der Versicherungsschutz erlischt, Name und Kennung des verantwortlichen Außendienstmitarbeiters und die Versicherungshöhe enthält.
Die für den Ablauf relevanten XML-Schemata und das XML-Netz (ohne Markierung) zeigt Abbildung 9.

□

XML-Netze können mitsamt ihrer Markierung als XML-Dokumente in einer XML-Datenbank gespeichert und verwaltet werden. Für XML-Schemata und Filterschemata können aus den entsprechenden Diagrammen XML-Dokumente abgeleitet werden, welche die jeweiligen Schemata exakt beschreiben. Petri-Netze können (beispielsweise) mit Hilfe der Petri Net Markup Language (PNML) [WeK02] durch XML-Dokumente beschrieben werden, so daß eine integrierte Verwaltung der Prozeßbeschreibung in Form eines XML-Netzes und der zugehörigen Markierung möglich ist.

5 Zusammenfassung

Im vorliegenden Beitrag wurden XML-Netze vorgestellt, die zur integrierten Dokumenten- und Ablaufmodellierung von E-Business-Prozessen eingesetzt werden können. XML-Netze sind eine neue Variante höherer Petri-Netze, deren Stellen als Behälter für XML-Dokumente, die zu dem der Stelle entsprechenden XML-Schema gültig sind, interpretiert werden. Die Operationen auf den Dokumenten werden durch die Transitionen und Filterschemata der jeweils adjazenten Kanten beschrieben. Zunächst wurden XML-Schemata und ihre graphische Repräsentation, die XML-Schema-Diagramme, vorgestellt. Danach wurden Filterschemata mit einer an XML-Schema-Diagrammen orientierten graphischen Darstellung beschrieben, die zur Formulierung von Anfragen an XML-Dokumente und Dokumentenmanipulationen verwendet werden können. Abschließend wurden XML-Schemata und Filterschemata zur Vorstellung der XML-Netze verwendet.
Einen Ansatzpunkt für weitere Forschungsarbeit stellt u.a. die Analyse der XML-Netze dar. Hierbei muß beispielsweise untersucht werden, welche Analysemethoden anderer Petri-Netz-Typen auf XML-Netze übertragen werden können, so daß sie noch für komplexe XML-Netze anwendbar sind und in einer angemessenen Zeit ein Ergebnis liefern. Andererseits kann es sicherlich sinnvoll sein, eine Transformation der XML-Netze in Petri-Netze anderer Netztypen durchzuführen, um die für diese Typen bekannten Analysemethoden anwenden zu können. Der Entwurf von XML-Netzen soll außerdem durch ein Vorgehensmodell unterstützt werden. Das Vorgehensmodell beschreibt die Vorgehensweise beim Entwurf des XML-Netzes, beginnend mit dem Petri-Netz eines elementares Typs bis hin zum fertigen XML-Netz. Für XML-Netze sollen auch Methoden entwickelt werden, wie XML-Netze in andere Ablaufbeschreibungssprachen, beispielsweise Ereignisgesteuerte Prozeßketten, übersetzt werden können. Dies ist besonders dann sinnvoll, wenn Ablaufschemata, die als XML-Netze modelliert wurden, Anwendern vermittelt werden sollen, die nicht mit Petri-Netzen, sondern einer anderen Ablaufbeschreibungssprache vertraut sind.
Die Entwicklung eines XML-Netz-Editors und -Simulators bildet die Basis für die prototypmäßige Implementierung eines XML-Netz-basierten Workflow-Managementsystems. Bei der Implementierung sollen weitgehend bereits am Markt

verfügbare Produkte eingesetzt werden: Für die Verwaltung der XML-Dokumente wird der Tamino XML Server (http://www.softwareag.com/tamino/), ein natives XML-Datenbanksystem der Software AG (http://www.softwareag.com/) eingesetzt. Als Editor für die XML-Schemata wird der XML-Editor der XML Spy Suite (http://www.xmlspy.de/) der Firma Altanova (http://www.xmlspy.com/company.html) verwendet. Der Entwurf der Filterschemata soll durch ein Werkzeug unterstützt werden, das die Ableitung zulässiger Filterschemata aus den jeweiligen Dokumentenschemata durch Funktionalitäten wie das Ausblenden von Elementtypen, die (automatische) Zuweisung von Variablen oder den Eintrag von Konstanten durch den Modellierer ermöglicht. Das Werkzeug zum Entwurf der Ablaufschemata, der XML-Netz-Simulator und die Workflow-Engine zur Ablaufsteuerung sollen auf der INCOME Suite (http://www.promatis.de/produkte/income_suite/index.htm) der Firma PROMATIS (http://www.promatis.de/) aufgesetzt werden.

Literatur

[AQH97] Abiteboul, S.; Quass, D.; McHugh, J.: *The Lorel Query Language for Semistructured Data*, in: International Journal on Digital Libraries, 1(1), April 1997, S. 68-88 (ftp://db.stanford.edu/pub/papers/querying.ps).

[BCF99] Booch, G.; Christerson, M.; Fuchs, M.; Koistinen, J: *UML for XML Schema Mapping Specification*, 12. August 1999 (http://www.rational.com/media/uml/resources/media/uml_xmlschema33.pdf).

[BMN02] Bex, G.J.; Maneth, S.; Neven, F.: *A Formal Model for an expressive fragment of XSLT*, in: Information Systems, 27, 2002, S. 21-39.

[BRJ99] Booch, G.; Rumbaugh, J.; Jacobson, I.: *The Unified Modeling Language User Guide*, Addison Wesley, München et al., 1999.

[CDF01] Comai, S.; Damiani, E.; Fraternali, P.: *Computing Graphical Queries over XML Data*, in: ACM Transactions on Information Systems, 19(4), Oktober 2001, S. 371-430.

[Che76] Chen, P.P.: *The Entity-Relationship Model: Toward a Unified View of Data*, in: ACM Transactions on Information Systems, 1(1), S. 166-192, 1976.

[CSF00] Conrad, R.; Scheffner, D.; Freytag, J.C.: *XML Conceptual Modeling Using UML*, in: Laender, A.H.F.; Liddle, S.W.; Storey, V.C. (Hrsg.): ER2000 Conference, Lecture Notes in Computer Science Vol. 1920, Springer-Verlag, Berlin – Heidelberg, 2000, S. 558-571.

[DeO96] Desel, J.; Oberweis, A.: *Petri-Netze in der Angewandten Informatik: Einführung, Grundlagen und Perspektiven*, in: Wirtschaftsinformatik, 38(4), 1996, S. 359-367.

[DFF99] Deutsch, A.; Fernandez, M.; Florescu, D.; Levy, A.; Maier, D.; Suciu, D.: *Querying XML Data*, in: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 12(3), 1999, S. 27-34.

[EHP02] Ehrig, H.; Hoffmann, K.; Padberg, J.; Baldan, P.; Heckel, R.: *High-Level Net Processes*, in: Brauer, W. et al. (Hrsg.): Formal and Natural Computing, Lecture Notes in Computer Science Vol. 2300, Springer-Verlag, Berlin – Heidelberg, 2002, S. 191-219.

[Gen86] Genrich, H.J.: *Predicate/Transition Nets*, in: Brauer, W.; Reisig, W.; Rozenberg, G. (Hrsg.): *Petri Nets : Central Models and Their Properties, Advances in Petri Nets, Part I*, Lecture Notes in Computer Science Vol. 254, Springer-Verlag, Berlin – Heidelberg, 1986, S. 207-247.

[HaM01] Harold, E.R., Means, W.S.: *XML in a Nutshell*, O'Reilly, Peking, 2001.

[Jab00] Jablonski, S.: *Workflow Management Between Formal Theory and Pragmatic Approaches*, in: van der Aalst, W.M.P.; Desel, J.; Oberweis, A.: Business Process Management — Models, Techniques, and Empirical Studies, Lecture Notes in Computer Science Vol. 1806, Springer-Verlag, Berlin – Heidelberg, 2000, S. 345-358.

[LeO02] Lenz, K.; Oberweis, A.: *Interorganizational Business Process Management with XML Nets*, erscheint in: Ehrig, H.; Reisig, W.; Rozenberg, G.; Weber, H. (Hrsg.): Advances in Petri Nets, Lecture Notes in Computer Science, Springer-Verlag, Berlin – Heidelberg, 2002.

[MeH01] Mello, R.S.; Heuser, C.A.: *A Rule-Based Conversion of a DTD to a Conceptual Schema*, in: Kunii, H.S.; Jajodia, S.; Sølvberg, A. (Hrsg.): ER2001 Conference, Lecture Notes in Computer Science Vol. 2224, Springer-Verlag, Berlin – Heidelberg, 2001, S. 133-148.

[Mer99] Merz, M.: *Electronic Commerce — Marktmodelle, Anwendungen und Technologien*, dpunkt.verlag, Heidelberg, 1999.

[MLM01] Mani, M.; Lee, D.; Muntz, R.R.: *Semantic Data Modeling Using XML Schemas*, in: Kunii, H.S.; Jajodia, S.; Sølvberg, A. (Hrsg.): ER2001 Conference, Lecture Notes in Computer Science Vol. 2224, Springer-Verlag, Berlin – Heidelberg, 2001, S. 149-163.

[Obe96] Oberweis, A.: Modellierung und Ausführung von Workflows mit Petri-Netzen, B.G. Teubner, Stuttgart – Leipzig, 1996.

[Ray01] Ray, E.T.: Learning XML, O'Reilly, Bejing et al., 2001.

[ReR98] Reisig, W., Rozenberg, G. (Hrsg.): *Lectures on Petri Nets Part I: Basic Models*, *Part II: Applications*, Lecture Notes in Computer Science Vol. 1491, Springer-Verlag, Berlin – Heidelberg, 1998.

[RLS98] Robie, J.; Lapp, J.; Schach, D.: XML Query Language (XQL), The Query Languages Workshop (QL'98), 1998 (http://www.w3.org/TandS/QL/QL98/pp/xql.html).

[WeK02] Weber, M.; Kindler, E.: *The Petri Net Markup Language*, eingereicht für: Petri Net Technology for Communication Based Systems, Lecture Notes in Computer Science Vol. 1491, Springer-Verlag, Berlin – Heidelberg, 1998.

[W3C00] World Wide Web Consortium: *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 6. Oktober 2000 (http://www.w3.org/TR/REC-xml).

[W3C01a] World Wide Web Consortium: *XML Schema Part 1: Structures*, *Part 2: Datatypes*, W3C Recommendation, 2. Mai 2001 (http://www.w3.org/TR/xmlschema-1/ und http://www.w3.org/TR/xmlschema-2/).

[W3C01b] World Wide Web Consortium: *XQuery 1.0: An XML Query Language*, W3C Working Draft, 20. Dezember 2001 (http://www.w3.org/TR/xmlquery).