Event-Driven-Process-Chain-Markup-Language (EPML): Anforderungen zur Definition eines XML-Schemas für Ereignisgesteuerte Prozessketten (EPK)

Jan Mendling, Markus Nüttgens

Universität Trier
Wirtschaftsinformatik II
Postfach 3825, D-54286 Trier
jm@wiinfo.uni-trier.de
markus@nuettgens.de

Abstract: Nach einer Darstellung der Notwendigkeit eines neutralen Austauschformates für die Ereignisgesteuerte Prozesskette (EPK) in XML, genannt Event-Driven-Process-Chain-Markup-Language (EPML), werden bereits existierende Standardisierungsbemühungen unter XMI, PNML und BPML betrachtet. Aus der Diskussion ergeben sich Anforderungen für die Ausarbeitung einer EPML, die in methodenorientierte, toolorientierte, implementierungsorientierte und verifikationsorientiert Anforderungen untergliedert werden.

1 Notwendigkeit eines XML-Schemas für EPKs

Bei der Modellierung von Geschäftsprozessen mit Ereignisgesteuerten Prozessketten (EPK) stellt sich die Frage, auf welches Modellierungswerkzeug zurückgegriffen werden soll. Kriterien für die Auswahl finden sich unter anderem bei [Nu02]. Durch die Heterogenität der Werkzeuge sowohl in ihrem Persistenzformat als auch in ihrer Ausdrucksstärke ergeben sich jedoch Probleme beim Austausch von Modellen zwischen verschiedenen Tools, wie es etwa unter Projektpartnern oder bei der Konsolidierung nach einem Unternehmenszusammenschluss nötig wird.

[WHB02] beschreiben als Lösung für dieses Problem zwei Alternativen, Standardisierung einerseits und Konvertierung andererseits. Sind bereits Modelle angefertigt, so schließt sich die Alternative Standardisierung aus und eine Konvertierung ist notwendig. Dabei bieten sich drei Strategien zum Umwandeln von Modellen an. Bei der Ringstruktur werden für n Tools n Konverter benötigt, die aufeinander folgend angeordnet sind, wobei der letzte Konverter vom letzten Tool wieder zum ersten Tool umwandelt. Ungünstigerweise potenzieren sich dabei Auslassungen, so dass nach einem Durchlauf nur noch die Schnittmenge aller Tools an Informationen resultiert. Die Peer-to-Peer-Struktur benötigt für n Tools (n-1) Konverter. Dies ist schon ab einer kleinen Anzahl von Tools unpraktikabel. Bei einer größeren Anzahl Tools empfiehlt sich die Definition eines intermediären Standards, der deren Heterogenität überbrücken kann.

Ein solcher intermediärer Standard existiert für die EPK noch nicht. Als Definitionsformat bietet sich die eXtensible Markup Language (XML) [Br00] an, da hierfür mit XML Schema Definition (XSD) [BM01,Be01] ein Mechanismus zur allgemeinen Festlegung der Sprachelemente besteht. Des Weiteren können mit XSLT [Cl99] auf effiziente Weise Konverter zwischen verschiedenen XML-Sprachen, wie sie etwa individuelle Tools verwenden, programmiert werden.

Ziel ist es im Folgenden, der Definition einer Event-Driven-Process-Chain-Markup-Language (EPML) vorzuarbeiten. Zuerst werden vergleichbare Standardisierungsbemühungen kurz skizziert, die für Petri-Netze, UML-Diagramme und allgemein unter der Business Process Management Initiative (BPMI) unternommen werden. Anschließend werden Anforderungen aufgeführt, die bei der Ausarbeiten einer EPML Berücksichtigung finden sollten. Ein Ausblick schließt die Betrachtungen.

2 Vergleichbare Standardisierungsbemühungen

2.1 eXtensible Markup Language Metadata Interchange (XMI)

Im Rahmen der Unified Modeling Language (UML) [OMG01] ist die Problematik heterogener Tools bekannt [Je00]. Neben der Standardisierung der graphischen Notation ist eine XML-Sprache zum Austausch von Modellen definiert worden, die unter der Bezeichnung eXtensible Markup Language Metadata Interchange-Format (XMI) [OMG02] gepflegt wird.

XMI orientiert sich an der Metamodellhierarchie der Object Management Group (OMG). Hier wird zwischen vier Modellebenen unterschieden, vom Meta-Metamodell über Metamodell und Modell bis zur Ausprägung. Beispielweise kann eine Person namens Hans die Instanz einer Klasse Person (Modell) sein. Diese ist eine Instanz des Metamodellelements Klasse, welche wiederum die Instanz einer Metaklasse des Meta-Metamodells ist. XMI ist in der Lage, gleichermaßen Metamodelle als auch deren Ausprägungen zu übertragen.

Die Kodierung in XMI erfolgt in zwei Schritten [Je00]. Als erstes wird das Modell als Instanz des entsprechenden Metamodells beschrieben. Anschließend werden die konkreten Objekte hinzugefügt. Die Namensgebung der Elemente stützt sich wiederum auf die vier Modellebenen. Nach XMI-Nomenklatur beginnt der jeweilige Tag mit der Bezeichnung des Metamodell-Packages. Dann folgt der Name der Metaklasse und der Name des Metaattributs, jeweils mit einem Punkt separiert. Jedes Metamodellelement wird dabei über eine ID eindeutig identifiziert, dessen Korrektheit auf Ebene des XML-Parser validiert wird.

Die beschriebene Namensgebung schafft durch die vollständige Angabe der Objekthierarchie zum einen Lesbarkeit. Zum anderen werden die Tag-Namen schnell sehr lang, was ein ungünstiges Verhältnis zwischen Metadaten und Nutzdaten zu Folge hat. Nichtsdestotrotz ergeben sich viele Anwendungsmöglichkeiten von XMI. [Je00] nennt

die Codegenerierung aus OO-Modellen, die Modellvalidierung, Metrikenberechnung, Persistenzhaltung, Versionsverwaltung und Export/Import-Möglichkeiten.

2.2 Petri Net Markup Language (PNML)

Anders als bei der Definition von XMI können die Autoren der Petri Net Markup Language (PNML) [WK02] nicht auf eine standardisierte Version ihrer Modellierungsmethode zurückgreifen. Daher ist ihr Vorschlag eines PNML-Datenformates auch gleichzeitig ein Standardisierungsversion. Dabei stützen sie sich auf drei Prinzipien: die Lesbarkeit für einen menschlichen Betrachter, die Universalität jegliche Petri-Netze speichern zu können, als auch die Wechselseitigkeit beliebige Informationen mitführen zu können.

Neben den klassischen Petri-Netz-Elementen Stelle, Transition und Kanten führen sie pragmatisch weitere Elemente ein [WK02]. Zur Strukturierung werden zwei Konstrukte eingeführt. Mit Pages können Teile eines Petri-Netzes zu einer Seite gruppiert werden, wobei References von Stellen oder Transitionen einer Seite auf die Fortsetzung auf einer anderen Seite verweisen. Module hingegen stellen eine methodenlogische Gliederungsmöglichkeit dar. Hiermit können Systeme geschachtelt aus wieder verwendbaren Modulreferenzen zusammengesetzt werden.

Unter Objekten werden Pages, Stellen, Stellenreferenzen, Transitionen, Transitionsreferenzen und Kanten zusammengefasst. Objekte können Labels haben, um weitere Informationen zu hinterlegen, etwa einen Namen. Ein einfaches Label wird als Attribut bezeichnet. Darüber hinaus können mit einer Annotation auch Grafiken hinterlegt werden. Zu jedem Objekt sind Positionsangaben möglich, die ein Tool zur grafischen Ausrichtung des Objektes nutzen kann. Zuletzt ist es möglich in dem Element ToolInfo spezifische Informationen über das verwendete Tool abzulegen.

Mit PNML liegt ein Austauschformat für Petri-Netze vor, welches sich durch Flexibilität und durch Strukturierungsmöglichkeiten auszeichnet. Es ist gleichermaßen der Aufbau einer Hierarchie von Netzen möglich als auch die Verwaltung von Positionsinformationen für die einheitliche Darstellung in verschiedenen Modellierungstools. Als Nachteil für die Implementierung dürfte sich die Tatsache herausstellen, dass die Pflege alleine von den Autoren bewältigt wird. Somit fehlt die Autorität einer Standardisierungseinrichtung wie sie für UML und XMI gegeben ist. Doch selbst ohne dies dürfte sich das Konzept als nützlich für den Austausch von Petri-Netzen herausstellen.

2.3 Business Process Markup Language (BPML)

Die historische Entwicklung der Business Process Modeling Language (BPML) [Ar02] unterscheidet sich stark von dem der beiden obigen Modellierungsmethoden. Zum einen entstand sie aus einem starken Implementierungsfokus. Daher ist es zum zweiten auch nicht verwunderlich, dass die Definition der XML-Syntax der Festlegung einer graphischen Modellierungsmethode vorauseilt. Ziel der BPML ist es, ein Modell für abstrakte und ausführbare Prozesse zu definieren. Dabei sollen sämtliche Aspekte von Geschäfts-

prozessen abgebildet werden. Das umfaßt nach BPML Aktivitäten verschiedener Komplexität, Transaktionen und deren Kompensation, Datenmanagement, Concurrency, Ausnahmebehandlung und operationale Semantik [Ar02].

Ein Geschäftsprozess wird als adaptive Struktur für Aktivitäten verstanden, in denen eine Reihe von Teilnehmern verschiedene Rollen einnehmen kann [AA01]. Der Begriff Teilnehmer versteht sich umfassend als Generalisierung für IT-Systeme, Anwendungen, Anwender, Geschäftspartner und andere Prozesse. Geschäftsprozessmanagement ziele darauf ab, die Zusammenarbeit von Teilnehmern auf eine zuverlässige, skalierbare und sichere Art und Weise zu realisieren. Ähnliche Anforderungen führten im Datenbankbereich zur Entwicklung des Transaktionskonzeptes und des ACID-Paradigmas [HR83]. Für verteilte Prozesse sind mitunter andere Konzepte notwendig, die sich auch im Metamodell von BPML widerspiegeln.

Aus dem Sprachumfang von BPML sei hier nur der Ausschnitt der Prozessdefinition dargestellt. Ein Prozess wird durch eine Nachricht, einen anderen Prozess oder ein Ereignis angestoßen. Er hat unter anderem einen Namen, Input- und Outputspezifikationen und läuft dabei einem Kontext ab, der für das Ereignis- und Ausnahmekonzept benötigt wird. Ein Prozess besteht aus einer Reihe von Aktivitäten, die atomar oder komplex sein können. Dabei kann eine komplexe Aktivität wiederum ein anderer Prozess sein, wodurch eine Schachtelung möglich wird. An atomaren Aktivitäten gibt es 16 verschiedene, die 5 definierte Zustände annehmen können.

Die BPML realisiert ein Nachrichtenkonzept und ermöglicht synchrone wie asynchrone Kommunikation. Mit ihrer Komplexität und ihrer Umsetzung technischer Konzepte ist sie für eine betriebswirtschaftliche Geschäftsprozessmodellierung, wie sie zur Kommunikation bei Optimierungsprojekten eingesetzt wird, unter Umständen in fortgeschrittenen Phasen nützlich, für den Einstieg in das Projekt zu komplex. Dennoch hat BPML ein großes Potenzial, die Lücke zwischen Modellierung und Implementierung zu schließen.

2.4 Lessons learned

Die drei obigen Modellierungsmethoden haben große Unterschiede in verschiedenen Dimensionen aufgezeigt. So sind BPML und XMI von großen internationalen Organisation standardisiert. PNML versucht dies durch eine flexible Modellierung zu kompensieren. XMI wie auch UML zeigt eine klare hierarchische Struktur und stellt somit einen top-down-Ansatz dar, während BPML die Vollständigkeit möglicher Semantiken betont und aus dem Implementierungsfokus geboren eher bottom-up definiert wurde. Zudem ist der Komplexitätsgrad ganz unterschiedlich. Während BPML und XMI eine Vielzahl an verschiedenen Tags anbieten, ist deren Anzahl in PNML überschaubar.

Im Folgenden sollen nun aus der Diskussion der drei Konzepte Anforderungen für die Definition einer Ereignisgesteuerte-Prozessketten-Markup-Language (EPML) erarbeitet werden. Die Betrachtungen gliedern sich in methodische, toolorientierte, implementierungsorientierte und verifikationsorientierte Anforderungen.

3 Anforderungen an eine EPML

3.1 Methodische Anforderungen

Seit der erstmaligen Beschreibung Ereignisgesteuerter Prozessketten (EPK) [KNS92] ist das Konzept kontinuierlich weiter formalisiert worden, zuletzt bei [NR02] worauf sich die folgenden Ausführungen beziehen.

Methodisch ist von einer EPML zu fordern, dass alle Sprachelemente der EPK explizit definiert werden. Eine metamodellartige Darstellung durch einen Typ Objekt, der über ein freies Typbezeichnungsfeld verfügt, ist abzulehnen, da so die Abgeschlossenheit der Methode syntaktisch nicht sicher gestellt werden kann. Mit dieser Forderung lassen sich auch einfacher Integritätsbedingungen in die Syntax einarbeiten, etwa das auf eine Funktion nicht zwei Ereignisse folgen dürfen ohne das ein Konnektor zwischen geschaltet ist. Neben der Explizität ist die Vollständigkeit der Sprachelemente zu fordern. Neben den Elementen des flachen EPK-Schemas sind somit Hierarchien wie auch Zustände des EPK-Schemas mit zu modellieren.

3.2 Toolorientierte Anforderungen

Für die Verarbeitung in grafischen Modellierungs-Tools ist die Mitführung von Positionsdaten der einzelnen Sprachelemente notwendig. Dies impliziert die Definition der Ausrichtung eines Koordinatensystems als auch die Festlegung eines Ankerpunktes für jedes Objekt, sowie dessen Höhe und Breite.

Zudem ist die Verwaltung eines großen Repositories von Modellen zu unterstützen. Dies impliziert eine betrachtungslogische Hierarchisierung der Modellen mit der Möglichkeit, flexibel Verweise zu definieren. Dies ist schwerlich ohne eine wie auch immer konzipierte Verzeichnisstruktur möglich.

Zuletzt ist es nützlich, ein Objekt sowohl als Original als auch als Verweis auf ein Objekt darstellen zu können. Dies ist erforderlich, um etwa eine Funktion an mehreren Stellen übersichtlich zu verwenden, ohne sie logisch neu zu erzeugen. Es wird damit erforderlich separate Objektdefinitionen und Objektinstanzen zu verwalten, wie dies das ARIS-Toolset der IDS Scheer AG [IDS02] bereitstellt.

3.3 Implementierungsorientierte Anforderungen

Aus Implementierungssicht stellt sich die Forderung, zusätzliche Information zu den einzelnen Objekten angeben zu können. Dies sollte vorerst über eine flexible Hierarchie von Elementen und Attributen erreicht werden. Damit wird es einerseits möglich, das Kontrollflussmodell sukzessive zu erweitern, bis eine Konvertierung in ein Workflow-Modell oder BPML möglich ist. Gerade das Fehlen von expliziten booleschen Schaltregeln wurde in der Vergangenheit vermisst [Ch01]. Andererseits kann hiermit sämtliche

Information aus implementierungsnäheren Formaten extrahiert und gegebenenfalls dargestellt werden.

Zusätzlich wäre es nützlich in Anlehnung an BPML die möglichen Zustände eines jeweiligen Objektes über einen Zustandsautomat zu definieren. Dadurch ließen sich explizit Prozess-Zustände einer Prozess-Instanz verwalten und portieren.

3.4 Verifikationsorientierte Anforderungen

Als verifikationsorientierte Anforderungen ergibt sich gleichermaßen die explizite Modellierung der Objekte der Methode. Somit können Integritätsbedingungen direkt einkodiert werden. Zudem ist es wichtig, dass sich aus dem Format effizient eine Darstellung erzeugen läßt, die Aussagen über Erreichbarkeit einzelner Knoten und Degenerierungen wie Dead Lock oder Live Lock ermöglicht. Denn mit solchen Prüfungen erschließt sich erst der Wert der Formalisierung einer Methode.

4 Ausblick

Ziel der weiteren Forschung wird es sein, eine formale Definition der EPK mit Hilfe einer XML-basierten EPML vorzunehmen und somit den Nutzen einer Standardisierung hinsichtlich Modelldatenaustausch und Modellverifikation auszuarbeiten.

Literaturverzeichnis

[AA01] Agrawal, A.; Arkin, A.: BPML 0.4 Working Draft, 2001. Abruf von http://www.bpmi.org am 09.04.2002.

[Ar02] Arkin, A.: BPML 1.0 Last Call Working Draft. BPMI.org, 2002. Abruf von http://www.bpmi.org am 12.10.2002.

[Be01] Beech, D.; Lawrence, S.; Moloney, M.; Mendelsohn, N.; Thompson, H.S. (Hrsg.): XML Schema Part 1: Structures. World Wide Web Consortium, Boston, USA, 2001. Abruf von http://w3c.org/TR/2001/REC-xmlschema-1-20010502/ am 20.10.2002.

[BM01] Biron, P.V.; Malhotra, A. (Hrsg.): XML Schema Part 2: Datatypes. World Wide Web Consortium, Boston, USA, 2001. Abruf von http://w3c.org/TR/2001/REC-xmlschema-2-20010502/ am 20.10.2002.

[Br00] Bray, T.; Paoli, J.; Sperberg-McQueen, C.M.; Maler, E. (Hrsg.): Extensible Markup Language (XML) 1.0 (Second Edition). World Wide Web Consortium, Boston, USA, 2000. Abruf von http://www.w3c.org/TR/2000/REC-xml-20001006/ am 20.10.2002.

[Ch01] Christensen, D.; Kraiß, A.; Syri, A.; Weikum, G.: Automatische Übersetzung von Geschäftsprozessmodellen in ausführbare Workflows. In (Heuer, A.; Leymann, F.; Priebe, D.): Datenbanksysteme in Büro, Technik und Wissenschaft (BTW). Berlin, Heidelberg, 2001, S. 505-513.

[Cl99] Clark, J. (Hrsg.): XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium, Boston, USA, 1999. Abruf von http://w3c.org/TR/xslt/ am 20.10.2002.

[HR83] Härder, T.; Reuter, A.: Principles of Transaction-Oriented Database Recovery. In ACM Computing Surveys, 15(4), 1983: S. 287-317.

[IDS02] IDS Scheer AG: XML-Export und Import mit ARIS 6 Collaborative Suite. Saarbrücken, 2002. Abruf von ftp://ftp.ids-scheer.de/pub/ARIS/HELPDESK/EXPORT/ARIS60/de_xmlexp60.pdf am 20.10.2002.

[Je00] Jeckle, M. C.: Metamodellierung als Ansatz zur Lösung der Modellaustausch-problematik im Umfeld objektorientierter Modellierungssprachen. In: Proceedings International Knowledge Technology Forum, Leipzig 2000. Abruf von http://www.jeckle.de am 20.10.2002.

[KNS92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“. In: Scheer, A.-W. (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, Saarbrücken, 1992. Abruf von http://www.iwi.uni-sb.de/iwi-hefte/heft089.zip am 20.10.2002.

[NR02] Nüttgens, M.; Rump, F.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen (Promise‘2002), Potsdam, 2002.

[Nu02] Nüttgens, M.: Rahmenkonzept zur Evaluierung von Modellierungswerkzeugen zum Geschäftsprozessmanagement. In (Gesellschaft für Informatik (GI) e.V.): Informationssytem-Architekturen, Wirtschaftsinformatik Rundbrief der GI Fachgruppe WI-MobIS, 9, 2002.

[OMG01] OMG: Unified Modeling Language Specification – Version 1.4. OMG, 2001. Abruf von http://www.omg.org am 12.10.2002.

[OMG02] OMG: XML Metadata Interchange (XMI) Specification. OMG, 2002. Abruf von http://www.omg.org am 12.10.2002.

[WHB02] Wüstner, E.; Hotzel, T.; Buxmann, P.: Converting Business Documents: A Classification of Problems and Solutions using XML/XSLT. In: Proceedings of the 4th International Workshop on Advanced Issues of E-Commerce and Web-based Systems (WECWIS 2002).

[WK02] Weber, M.; Kindler, E.: The Petri Net Markup Language. In (Ehrig, H.; Reisig, W.; Rozenberg, G.; Weber, H.): Petri Net Technology for Communication Based Systems. Berlin, Heidelberg: Springer, 2002.