Konzeption eines XML-basierten Austauschformates für Ereignisgesteuerte Prozessketten (EPK)

Jan Mendling

Wirtschaftsuniversität Wien
jan.mendling@wu-wien.ac.at

Markus Nüttgens

Universität des Saarlandes
markus@nuettgens.de

Zusammenfassung: Der Erfolg und die schnelle Verbreitung des XML-Konzeptes haben dazu geführt, dass die Hersteller von Werkzeugen zur Geschäftsprozessmodellierung zunehmend XML Import- und Export-Schnittstellen anbieten. Die Existenz produktspezifischer Schnittstellen ist aber keine hinreichende Bedingung für einen XML-basierten Modelldatenaustausch über Produktgrenzen hinweg. Ein wesentlicher Lösungsansatz besteht in der Entwicklung und Unterstützung von XML-basierten Austauschformaten. Dieser Beitrag befasst sich mit dem Konzept einer EPK Markup Language (EPML) als XML-basiertes Austauschformat für Ereignisgesteuerte Prozessketten (EPK).

1 XML-basierter Modelldatenaustausch

Seit den 1990er Jahren haben sich Werkzeuge zur Modellierung von Geschäftsprozessen (Business Process Modelling) zu einem eigenständigen Marktsegment entwickelt. Eine Studie von Gartner Research aus dem Jahr 2002 bezieht sich auf 35 größere Werkzeug-Hersteller und prognostiziert, dass sich diese Zahl in den nächsten Jahren tendenziell halbieren wird [Ga02]. Interoperabilität und die Unterstützung von Standards wird hierbei als ein wichtiges Selektionskriterium genannt. Die Unterstützung von XML-basierten Import- und Export-Schnittstellen wird auch von den Werkzeugherstellern selbst zunehmend als geschäftskritische Produkteigenschaft angeführt [MN03b]. Dabei sind zwei Stufen von Standardunterstützung zu unterscheiden. Auf der einen Seite wird XML als standardisiertes Repräsentationsformat für die Definition von Austauschformaten benutzt, was wir als schwache Standardunterstützung bezeichnen. Andererseits können standardisierte Inhaltsformate unterstützt werden, welches wir als starke Standardunterstützung bezeichnen.

Schwache Standardunterstützung beschreibt eine Strategie eines Werkzeugherstellers, ein produktabhängiges und damit proprietäres XML Schema bereitzustellen. In einem Umfeld heterogener Werkzeuge, die schwache Standardunterstützung bieten, benötigt man Transformationsprogramme, um ein Modell von Tool A nach Tool B zu übertragen. Ein wesentlicher Vorteil des XML-Einsatzes in diesem Szenario ist die Verfügbarkeit spezieller Anwendungen. Mit Hilfe von XSLT [Cl99] existiert beispielsweise eine Skript-Sprache für die Transformation von XML-Daten, die dem Programmierer das

Mendling J., Nüttgens, M.: Konzeption eines XML-basierten Austauschformates für Ereignisgesteuerte Prozessketten (EPK),
in: Gesellschaft für Informatik (GI) e.V. (Hrsg.): Informationssystem Architekturen, Wirtschaftsinformatik Rundbrief der GI Fachgruppe
WI-MobIS, 10(2003)2, S. 89-103

„Parsen“ des Eingabe-Dokumentes, die Definition von Transformationsregeln und die Generierung des Outputs erheblich vereinfachen. Das grundlegende Problem der Heterogenität der produktspezifischen XML-Daten ist damit allerdings nicht gelöst.

Die starke Standardunterstützung setzt die Existenz anerkannter XML-Austauschformate und eine Unterstützung durch die Werkzeughersteller voraus. Für den Anwender ist dies ein effizienteres Szenario. Spezialisierte Werkzeuge können für unterschiedliche Aufgaben genutzt werden, wobei das standardisierte Austauschformat die Heterogenität überbrückt. Derartige Standards haben sich bereits erfolgreich für eine Reihe von Modellierungskonzepten etabliert. Für die Unified Modeling Language (UML) [OMG03a] wurde XML Metadata Interchange (XMI) [OMG03b] und für Petri Netze eine Petri Net Markup Language (PNML) spezifiziert [Bi03]. Für die Geschäftsprozessmodellierung mit Ereignisgesteuerten Prozessketten (EPK) [KNS92] befindet sich derzeit eine vergleichbare Spezifikation einer EPK Markup Language (EPML) in der Entwicklung [MN02, Me03, MN03b, MN03c].

Die Etablierung eines XML-Standardformates für EPK-Modelle hat aufgrund der zunehmenden Verbreitung und Akzeptanz des EPK-Konzeptes ein hohes Anwendungspotenzial. Der Austausch von Geschäftsprozessmodellen kann dabei unter zwei Dimensionen erfolgen: Der horizontale Modelldatenaustausch hat die Integration von Modellierungswerkzeugen zum Gegenstand, wohingegen ein vertikaler Modelldatenaustausch auf die Integration von Modellierungswerkzeugen mit Ausführungswerkzeugen wie z.B. Simulationswerkzeugen, Workflow-Engines oder Monitoring-Diensten abzielt [Wf02]. Damit kann ein wesentlicher Beitrag zur Integration des Entwicklungszyklus von Prozessmodellierung und –implementierung geleistet werden.

Bislang ist der Markt für Modellierungswerkzeuge durch eine Strategie der schwachen Standardunterstützung mit proprietären XML-Schnittstellen gekennzeichnet. Die Entwicklung von XML-Austauschformaten wie EPML sollte mittelfristig ein Umschwenken in Richtung einer starken Standardunterstützung begünstigen. Ein Zwischenschritt kann auch in der Nutzung von XML-Austauschformaten als intermediäres Format liegen. Bei einer größeren Anzahl von Werkzeugen kann ein gemeinsames intermediäres Format die Anzahl der Transformationsprogramme maßgeblich verringern [WHB02].

Der Beitrag diskutiert, wie solch ein Austauschformat für EPKs entworfen werden kann. Im zweiten Kapitel werden allgemeine XML-Designprinzipien und Entwurfsprinzipien für EPML besprochen. Kapitel 3 behandelt Prozess- und Graphrepräsentation in XML. Hier werden verschiedene Darstellungsmöglichkeiten anhand vorhandener Spezifikationen erläutert. In Kapitel 4 werden die syntaktischen Elemente eines EPK-Prozesses dargestellt. Dies umfasst auch Aspekte der Verwaltung von Modellhierarchien. Kapitel 5 bietet einen Überblick über die graphische Repräsentation wie etwa Koordinatensystem, Positionsangaben und Layout-Informationen. In Kapitel 6 wird eine Übersicht über Sichten auf Prozesse gegeben. Das siebte Kapitel schließt mit einer Zusammenfassung und einem Ausblick.

2 EPML Design Prinzipien

Das Ziel von EPML ist es, ein werkzeug- und plattformunabhängiges Austauschformat für EPKs zur Verfügung zu stellen. Daraus ergeben sich drei Fragen. Zum ersten stellt sich die Frage, *was* genau im Detail modelliert werden soll. Zum zweiten impliziert dies die Frage nach den übergeordneten Richtlinien, *wie* Sachverhalte in XML ausgedrückt werden sollen. Zum dritten steht die Frage im Raum, *welche* generellen Prinzipien die Modellierung leiten sollen. Wir werden diese Fragen der Reihe nach aufgreifen, beginnend mit der dritten, gefolgt von der zweiten, die beide in diesem Kapitel angesprochen werden.

2.1 Allgemeine EPML Design-Prinzipien

Einige der veröffentlichten XML Spezifikationen beschreiben explizit ihre Design-Prinzipien wie z.B. das ASC X12 Reference Model for XML Design (X12) [AN02]. Dieses beschreibt ein siebenschichtiges Model für die Spezifikation von Geschäftsdokumenten. Die Definition von X12 wurde von vier abstrakten Design-Prinzipien geleitet: Ausrichtung an anderen Standards, Einfachheit, Vordefiniertheit und begrenzte Freiräume. Ausrichtung an anderen Standards (*Alignment*) bezieht sich auf die spezifische Domäne der Geschäftsdokumente, zu der auch andere Organisationen wie OASIS und UN/CEFACT, das World Wide Web Consortium und OASIS UBL Beiträge leisten. Einfachheit (*Simplicity*) ist ein domänenunabhängiges Kriterium. Es verlangt, dass spezifische Aspekte und Wahlmöglichkeiten auf ein Minimum reduziert werden. Vordefiniertheit (*Prescriptiveness*) bezieht sich wiederum auf Geschäftsdokumente. Dieses Prinzip empfiehlt es, eher mehr präzise und spezifische Dokumente zu definieren als wenige allgemeine. Begrenzte Freiräume (*Limit Randomness*) bezieht sich auf den Gebrauch gewisser Konstrukte von XML Schema Sprachen, insbesondere Operatoren, die verschiedene Wahlmöglichkeiten in der Syntax eröffnen. Der Einsatz solcher Konstrukte sollte auf ein Minimum reduziert werden. Dieser Aspekt bezieht sich auf den Entwurf von XML Vokabularien im Allgemeinen.

Der PNML-Ansatz wird von den Prinzipien Flexibilität, Eindeutigkeit und Kompatibilität geleitet [Bi03]. Flexibilität (*Flexibility*) ist ein wichtiger Aspekt für Petri-Netze, da jegliche Arten von heute diskutierten als auch zukünftigen Klassen von Petri Netzen abbildbar sein sollen. Dies wird über Labels erreicht, die zur Annotation von Kanten und Knoten benutzt werden können. Eindeutigkeit (*No Ambiguity*) bezieht sich auf das Problem der Standardisierung dieser Labels. Dafür werden Petri Net Type Definitions eingesetzt, um erlaubte Labels eines speziellen Netztypes zu beschreiben. Kompatibilität (*Compatibility*) beschreibt Probleme, die sich durch semantisch äquivalente Labels ergeben, die in verschiedenen Petri-Netztypen benutzt werden. Diese überlappenden Labels sollten beim Austausch zwischen verschiedenen Netztypen erhalten bleiben.

Der EPML-Ansatz reflektiert diese verschiedenen Design-Prinzipien. Er wird von den Prinzipien Lesbarkeit, Erweiterbarkeit, Tool-Orientierung und syntaktische Richtigkeit geleitet [MN03b]. *Lesbarkeit* fordert, dass EPML-Elemente und –Attribute intuitive und sprechende Namen haben. Dies ist wichtig, da EPML-Dokumente nicht nur von Anwen-

dungen benutzt werden, sondern auch von Anwendern, die beispielsweise XSLT-Skripte zur Konvertierung zwischen EPML und anderen XML Vokabularien schreiben. Lesbarkeit steht teilweise in Bezug zu den Prinzipien Einfachheit und begrenzte Freiräume des X12-Ansatzes. *Erweiterbarkeit* betrachtet ein Problem, das in gewisser Weise analog zu der Darstellung verschiedener Petri-Netz-Typen ist. Ein wichtiger Aspekt der EPK-Modellierung ist die Bereitstellung verschiedener Sichten auf ein Prozessmodell. EPML soll in der Lage sein, jegliche Arten von Sichten abbilden zu können anstatt sich auf eine vorgegebene Menge zu beschränken. *Tool-Orientierung* zielt hauptsächlich auf die graphische Repräsentation von EPK-Modellen. Dies ist eine zentrale Eigenschaft, da Modellierungswerkzeuge eine graphische Benutzerschnitte zur Definition von Prozessen bereitstellen. EPML soll in der Lage sein, verschiedene Layout- Positionsangaben von EPK-Elementen abspeichern zu können. Diese Aspekte werden in Kapitel 6 detailliert aufgegriffen. Syntaktische Richtigkeit befasst sich mit den Elementen der EPK-Syntax und deren Beziehungen zueinander. Dieser Punkt wird in Kapitel 3 und Kapitel 4 behandelt. Der folgende Abschnitt umfasst allgemeine Richtlinien für das Design von XML Vokabularien.

2.2 Richtlinien für XML Design

Grundsätzlich können zwei Ansätze zur Begründung von XML Design Richtlinien unterschieden werden: Ein theoretischer, der sich auf Normalformen und informationstheoretische Maßzahlen wie Entropie stützt, und ein pragmatischer, der Empfehlungen bereitstellt, wann und wie welche XML-Sprachkonzepte eingesetzt werden sollen und wie die Benennung von Elementen und Attributen vorgenommen werden soll.

Der theoretische Ansatz gründet sich auf Erkenntnisse der Datenbank-Theorie. Für relationale Datenbankmodelle wurden Funktionale Abhängigkeit (FD), Mehrwertige Abhängigkeiten (MVD) und Join Abhängigkeit (JD) konzipiert [Bi95]. Um Schemata mit formal guten Eigenschaften abzuleiten, gibt es Dekompositionsalgorithmen, um verschiedene Stufen von Normalformen zu erreichen. Diese Normalformen beseitigen Redundanzen und Anomalien durch Operationen auf relationalen Daten. Analog hierzu sind Normalformen für XML (XNF) entwickelt worden [EM01, AL02]. In [AL03] wird ein informationstheoretischer Ansatz vorgestellt, der die konzeptionelle Lücke zwischen relationaler und XML-Repräsentation überbrücken kann. Aufbauend auf Entropie-Maßen wird ein konzeptunabhängiges Verständnis zwischen Redundanzen und Normalformen erarbeitet. Ein Schema wird als *well-designed* bezeichnet, wenn es keine Instanzdaten enthalten kann, die weniger als das Maximum an Information bezüglich der bedingten Entropie tragen [AL03]. Darauf basierend kann gezeigt werden, dass ein Schema mit lediglich FDs und weder MVDs noch JDs *well-designed* ist, wenn und nur wenn es in Boyce-Codd-Normalform ist. FDs für XML-Daten treten dann auf, wenn Pfade von der Wurzel zu Knoten von anderen Pfaden abhängen. Dementsprechend wird definiert, dass ein XML Schema bezüglich einer Menge von FDs well-designed ist, wenn und nur wenn es in XNF ist [AL03]. Eine Verletzung dieser Forderung bedeutet, dass Redundanzen in der Art vorliegen, dass Pfade zu verschiedenen Knoten führen, die jedoch denselben Wert haben. Hier hilft ein Normalisierungsalgorithmus, der die Position von Attributen verändert und neue Elemente erzeugt bis XNF erfüllt ist [AL03]. Für

den Entwurf von XML Vokabularien bedeutet dies, dass es keine XPath [CD99] Anweisungen geben darf, die eine Knotenmenge mit jeweils demselben Wert zurückgeben. In diesem Fall ist XNF erfüllt und das Schema ist well-designed.

Pragmatische Ansätze zielen hauptsächlich auf die Erweiterbarkeit und mögliche Freiräume bei der Definition von XML Vokabularien ab. Dokumente von ISO [ISO01], SWIFT [SW01], MISMO [MIS02] und X12 [AN02] begründen eine Reihe von Regeln, um Mehrdeutigkeiten in XML Schemata zu minimieren. Diese XML Entwurfsrichtlinien umfassen Konventionen für Namen, für die Wahl zwischen Attributen und Elementen, für den Einsatz von speziellen Operatoren von XML Schema-Sprachen und für den Einsatz von Namespaces. *Namenskonventionen* beziehen sich auf die Bezeichnung von Elementen und Attributen. ISO, SWIFT, MISMO, und X12 fordern übereinstimmend den Gebrauch von ausschließlich englischen Wörtern. Namen können aus mehreren Wörtern bestehen, die gemäß des so genannten Upper Camel Case abgetrennt werden sollen (keine trennenden Leerzeichen, sondern jedes einzelne Wort beginnend mit einem Großbuchstaben). Abkürzungen und Akronyme sollen auf ein Minimum reduziert werden. *Stilkonventionen* leiten die Entscheidung zwischen Elementen und Attributen. X12 empfiehlt die Benutzung von Attributen für Metadaten und Elemente für Anwendungsdaten [AN02]. In diesem Zusammenhang sollte man identifizierte Schlüsselattribute als Metadaten auffassen und in Attribute setzen. Dies ermöglicht den DTD-konformen Gebrauch von ID, IDREF und IDREFS Datentypen und entsprechende key- und keyref-Deklarationen in W3C XML Schema [Be01, BM01]. Des Weiteren wird angemerkt, dass Attribute zu einer besseren Lesbarkeit des Inhaltes beitragen [Me01, AN02]. Deshalb sollten Daten, die nie erweitert werden können, in Attribute gepackt werden. *Schemakonventionen* empfehlen bewusst, nur einen Teil der Ausdrucksstärke einer XML Schema-Sprache einzusetzen. X12 rät beispielsweise, gemischte Inhaltstypen, Substitutionsgruppen und Gruppen-Redefinitionen zu vermeiden. Es sollten u.a. nur benannte Inhaltstypen und vordefinierte einfache Datentypen benutzt werden. Eine tiefergehende Diskussion findet sich in [AN02]. Namespace-Konventionen verweisen auf den Gebrauch von Namespaces in Dokumenten. X12 empfiehlt, explizite Namespace-Referenzen nur für die Wurzel zu benutzen.

Die theoretischen und pragmatischen Ansätze liefern komplementäre Richtlinien für die Definition „guter“ XML Schemata. Die aufgeführten Richtlinien sind in den EPML-Vorschlag eingeflossen. Das folgende Kapitel über Prozessgraph-Repräsentation in XML beginnt die Reihe der Kapitel, die einzelne Design-Aspekte von EPML vorstellen.

3 Prozessgraph-Repräsentation

Ein Graph ist ein Paar von Knoten V und Kanten E, wobei E eine Untermenge des kartesischen Produktes über V ist. In der Informatik spielen Graphen in den verschiedensten Bereichen eine Rolle. Zum Beispiel werden Entity-Relationship-Modelle zum konzeptionellen Entwurf von relationalen Datenbanken eingesetzt [Ch76]. Entitäten können dabei als spezielle Art von Knoten betrachtet werden und Relationships als spezielle Sorte von Kanten. Ein anderes Beispiel ist die objekt-orientierte Software-Entwicklung. Die Unified Modeling Language (UML) [OMG03a] erlaubt die Modellierung von Be-

ziehungen und Vererbungshierarchien, die ebenfalls als Graphen betrachtet werden können. Graphstrukturen von Software-Programmen werden im Bereich Software Reengineering eingesetzt [FGW03]. Da auch Geschäftsprozessmodelle formal auf gerichteten Graphen basieren, sollte ein Ansatz zur Repräsentation von EPK-Modellen Erfahrungen aus diesen Bereichen einbeziehen.

In der Informatik werden verschiedene Datenstrukturen für Graphen diskutiert, hauptsächlich mit Fokus auf die effiziente Ausführung von Graph-Algorithmen. Die drei grundlegenden Darstellungsformen sind Adjazenzmatrizen, Adjazenzlisten und Kantenlisten [Eb87]. Eine Adjanzenzmatrix repräsentiert einen gerichteten Graphen mit n Knoten mit Hilfe einer n × n Matrix, wobei der Eintrag an der Stelle (i,j) 1 ist, wenn eine Kante von Knoten i zu Knoten j verläuft, andernfalls ist der Eintrag 0 [Bl03]. Im Gegensatz dazu beschreibt die Adjazenzliste einen gerichteten Graph mit n Knoten als einen Array von n Knotenlisten. Ein Knoten j ist in der Liste i enthalten, wenn es eine Kante von Knoten i zu Knoten j gibt. Kantenlisten kommen der Mengendefinition von Graphen am nächsten. Eine Kante von einem Knoten i zu einem Knoten j wird als ein Paar (i,j) gespeichert.

Wenn solch eine generische Graph-Datenstruktur in XML ausgedrückt werden soll, müssen Anpassungen vorgenommen werden, um der Baumstruktur von XML Rechnung zu tragen. Das bedeutet, dass ID, IDREF, IDREFS Datentypen der Document Type Definition (DTD) [Br00] oder xs:key, xs:keyref Bedingungen von XML Schema [Be01, BM01] eingesetzt werden müssen, um beliebige Kanten zu beschreiben. Um Best Practices in der Beschreibung von Graphen mit XML zu identifizieren, wurden eine Reihe von graphbasierten Konzepten mit einer XML Syntax untersucht, darunter sowohl akademische Vorschläge als auch Industriestandards und werkzeugspezifische Formate. Deren Graph-Repräsentationen lassen sich in drei Kategorien aufteilen: block-orientierte Repräsentation, Adjazenz-Subelement-Listen und Kanten-Element-Listen.

Die *Block-orientierte Repräsentation* wird von den Notationen im Bereich Web Services wie BPML [Ar02] oder BPEL4WS [An03] genutzt. Dieses Paradigma ist inspiriert von Prozessalgebren wie dem Pi-Calculus [Mi99], die ihnen als theoretischer Rahmen dienen. Block-orientierte Sprachen bieten eine Menge von einfachen (simple in BPML oder basic in BPEL4WS) und komplexen (complex in BPML oder structured in BPEL4WS) Operationen, die den Kontrollfluss repräsentieren. Dabei gibt es einige Unterschiede in der Benennung zwischen BPML und BPEL4WS, die Konzepte sind jedoch sehr ähnlich [MM03]. Komplexe Operationen ermöglichen die Definition paralleler Ausführung, Sequenz, Verzweifungen und Schleifen. Sie können ineinander geschachtelt sein. Die reine Blockstruktur ist jedoch nicht im Stande, beliebige Kontrollflüsse auszudrücken. Daher besitzen BPML und BPEL4WS zusätzlich Links, die solche Kontrollflussbeschreibungen ermöglichen. Ein Vorteil der Block-Orientierung liegt darin, dass Code (wenn nicht zu tief geschachtelt) dank seiner sequentiellen Darstellung gut lesbar ist und mit wenigen Befehlen recht komplexe Sachverhalte ausgedrückt werden können. Der Nachteil besteht darin, dass Block-Orientierung mit anderen Konzepten wie Links gemischt werden muss, um bestimmtes Verhalten abbilden zu können und somit für eine graphische Aufbereitung schlecht geeignet ist. Es sind komplexe Abbildungen zwischen Modellierungswerkzeugen und block-orientierter Darstellung erforderlich, wie sie beispielsweise im Rahmen der Business Process Modelling Notation (BPMN) [Wh03]

diskutiert werden. EPML ist jedoch auch mit dem Fokus auf graphische Modellierungswerkzeuge konzipiert. Somit ist eine block-orientierte Repräsentation weniger geeignet.

Adjazenz-Subelement-Listen beschreiben einen Prozessgraphen mit Hilfe einer ungeordneten Liste von Knoten, die über ein ID-Attribut identifiziert werden. Eine Kante wird als Subelement ihres Ausgangsknotens abgespeichert. Die Kante besitzt ein Attribut, das eine ID-Referenz auf den Zielknoten enthält. Diese Repräsentation wird beispielsweise vom XML Import- und Exportformat der ARIS-Werkzeugfamilie benutzt [IDS01, IDS03a]. Sein Vorteil liegt darin, dass man einen schnellen Überblick über die von einem Knoten ausgehenden Kanten hat. Jedoch reduziert die Benutzung von ID-Referenzen die Lesbarkeit. Ein weiterer Nachteil gründet sich auf die konzeptionellen Implikationen dieses Darstellungsstils: Es können keine Kanten gespeichert werden, die nicht zumindest einen Ausgangsknoten haben. Im Rahmen der Geschäftsprozessmodellierung kann es jedoch durchaus sinnvoll sein, Prozessmodelle auszutauschen, die noch nicht fertig gestellt sind bzw. Kanten enthalten, die keinen Start- und nur einen Endknoten haben. Eine Adjazenz-Subelement-List kann solche Informationen nicht abbilden.

Kanten-Element-Listen stehen in enger Beziehung zu einer mengenorientierten Definition von Graphen. Kanten werden als first-class Objects behandelt und enthalten Felder für ID-Referenzen auf den Ausgangs- und den Zielknoten. Spezifikationen wie GXL [WKR02] und PNML [WK02] unterstreichen dies, indem sie Kanten genau wie Knoten ein ID-Attribut zuteilen. Die Kanten-Element-Listen-Repräsentation ist weit verbreitet. Sie wird von GXL, PNML, Visio's VDX [Mi03], XMI [OMG03b] und vom XPDL-Format der Workflow Management Coaltion [Wf02] unterstützt. Ein Vorteil dieser Darstellung ist ihre Flexibilität. Beliebige Graphen können mit ihr beschrieben werden und es können Kanten ohne Anfangs- und Endknoten gespeichert werden. Ein Nachteil besteht durch den Einsatz von IDs und IDREFs, die die Lesbarkeit des Formates beeinträchtigen.

Die Paradigmen zur Prozessrepräsentation betreffen die EPML Design-Prinzipien Lesbarkeit und Tool-Orientierung. Lesbarkeit wird am besten von der block-orientierten Repräsentation unterstützt, da bis auf Links keine IDs und ID-Referenzen benutzt werden. Jedoch müsste eine graphische Darstellung komplexe Abbildungsregeln definieren, was der Tool-Orientierung widerspricht. Kanten-Element-Listen sind zwar weniger leicht lesbar, dafür jedoch sehr flexibel. Da sie auch von anderen Spezifikationen genutzt werden, erleichtert dies werkzeug- und methodenbezogene Transformationen. Daher werden EPK-Prozessgraphen in EPML mit Hilfe von Kanten-Element-Listen beschrieben.

4 Prozessgraph-Elemente und deren Beziehungen

In diesem Kapitel wird das EPML-Verständnis eines EPK-Kontrollflusses dargestellt. Sichten auf den Kontrollfluss werden in Kapitel 6 besprochen. Da EPML auf dem Konzept einer EPK-Schemamenge aufbaut [NR02], ist es möglich, mehrere EPK-Modelle in einem EPML-File abzuspeichern. Zuerst wird eine Übersicht über die Organisation mehrerer EPK-Modelle und deren modellübergreifende Beziehungen in einem EPML-File

behandelt. Danach werden die einzelnen Elemente eines EPK-Modells in ihrer EPML-Syntax vorgestellt.

4.1 Hierarchien von EPK-Modellen

<epml> ist das Wurzelelement eines EPML-Files. Wie alle anderen Elemente kann es <documentation> und <toolInfo> Kindelemente enthalten. Diese können zusätzliche Daten enthalten wie etwa Kommentare und werkzeugspezifische Informationen. Es wird jedoch empfohlen, lediglich standardisierte Dublin Core Metadata Elemente [DC03] für die Dokumentation eines EPML-Files zu benutzen und nur solche anwendungsspezifischen Daten beizufügen, die sich auf die interne Speicherung von Modellen in spezifischen Tools beziehen, aber nicht die graphische Präsentation der Modelle beeinflussen. Allgemeine Graphik-Einstellungen könnten mit Hilfe des <graphicsDefault> Elementes vorgenommen werden (siehe Kapitel 5). Das <coordinates> Element dient der genauen Angabe der Ausrichtung des Koordinatensystems und damit der Interpretation der Positionsangaben der graphischen EPML-Elemente. Das @xOrigin Attribut kann die Werte“leftToRight” oder “rightToLeft” annehmen und das @yOrigin Attribut “topToBottom” oder “bottomToTop”. Es wird empfohlen, stets die Ausrichtung “leftToRight” und “topToBottom” zu benutzen. Dies ist allgemein üblich. Dennoch gibt es Ausnahmen wie zum Beispiel MS Visio, dessen Y-Achse von unten nach oben verläuft. Es wird empfohlen, solche Koordinaten zu transformieren, bevor das Modell in EPML angelegt wird.

In [NR02] wird eine EPK-Schemamenge als eine Menge hierarchischer EPK-Schemata definiert. Ein einzelnes EPK-Schema besteht aus einem flachen EPK-Schema, dessen Funktionen und Prozess-Schnittstellen Hierarchierelationen besitzen können. Eine detaillierte Betrachtung von flachen EPK-Schemata und deren Syntax-Elementen wird im folgenden Abschnitt vorgenommen. Syntaktisch stellt eine Hierarchierelation ein Paar von einer Funktion oder einer Prozess-Schnittstelle einerseits und einem EPK-Modell andererseits dar. Semantisch ist dies als ein Aufruf eines Sub-Prozesses zu verstehen. <epml> hat außerdem <definitions> Kindelemente, die im Zusammenhang mit <directory> Elementen erklärt werden. Das <view> Element wird in Kapitel 6 vorgestellt.

In EPML wird eine Hierarchie von Prozessen mit Hilfe von Verzeichnissen organisiert. Ein <directory> beinhaltet ein @name Attribut, andere Verzeichnisse und/oder EPK-Modelle. Jede <epc> wird über ein @epcId Attribut identifiziert und hat zudem ein @name Attribut. Die @epcId kann von Hierarchierelationen in Funktionen und Prozess-Schnittstellen aus referenziert werden. In einer Hierarchie von EPK-Modellen können Redundanzen auftreten, wenn eine Funktion oder ein anderes Prozess-Element in zwei oder mehr Prozessen benutzt wird. In diesem Fall benötigt man eine Stelle, an der man solche Informationen ablegen kann, um dann im konkreten Prozess darauf zu referenzieren. Dies ist die Funktion des <definitions> Elements. Dieses dient als Container für Kontrollfluss-Elemente, die mehrmals in der Modellhierarchie benutzt werden.

4.2 EPK-Modelle in EPML-Syntax

Eine formale Spezifikation der EPK-Syntax und –Semantik geben [NR02] und [Ki03]. In [KNS92] wird das EPK-Konzept eingeführt, um die zeitlichen und logischen Abhängigkeiten in Geschäftsprozessmodellen abzubilden. Elemente der EPK können vom Typ Funktion (aktive Elemente) symbolisiert durch <function>, vom Typ Ereignis (passive Elemente) repräsentiert von <event> oder vom Konnektor-Typ AND, OR, oder XOR sein, wobei die Konnektoren entweder Split- oder Join-Operatoren sein müssen. Sie werden mit den EPML-Elementen <and>, <or> und <xor> beschrieben. Diese Objekte sind über <arc> Elemente verbunden und stellen den Kontrollfluss dar. Im Zusammenhang mit den Einsatzerfahrungen mit dem SAP Referenzmodell wurden zusätzlich Prozess-Schnittstellen und hierarchische Funktionen eingeführt [KM94]. Das <processInterface> wird dazu benutzt, um vom Ende eines Prozesses auf einen Folgeprozess zu verweisen. Eine hierarchische <function> ermöglicht es, Makro-Prozesse zu definieren, die auf Subprozesse verweisen. Beide Arten von Relationen werden mit Hilfe des <toProcess> Elements beschrieben, dessen @linkToEpcId eine Relation mit einem anderen EPK-Prozess repräsentiert. Ereignisse, Funktionen, Prozess-Schnittstellen, Konnektoren und Kontrollflusskanten sind die syntaktischen Elemente einer flachen EPK-Schemamenge [NR02]. Sie alle besitzen ein @id Attribut, ein <name> Element, ein <description> Element, ein <graphics> Element (beschrieben in Kapitel 5) und ein <syntaxInfo> Element, das zusätzliche Informationen bezüglich des impliziten Elementtyps enthalten kann. Syntaxinformation dient dem Design-Prinzip der syntaktischen Richtigkeit und erlaubt eine leichtere Verifikation von EPK-Syntaxeigenschaften. Für eine Darstellung von impliziten Elementtypen und EPK-Syntaxeigenschaften sei auf [MN03a, MN03c] verwiesen.

Manche Kontrollflusselemente haben spezielle Elemente. Möglicherweise werden dieselben Ereignisse, Funktionen und Prozess-Schnittstellen mehrmals in einer Hierarchie von EPKs verwendet. Um daraus resultierende Redundanzen zu vermeiden, kann das <reference> Element anstelle des <name> und <description> Elementes benutzt werden. Es referenziert die Definition eines Ereignisses, einer Funktion oder einer Prozess-Schnittstelle, die zentral den Namen und die Beschreibung abspeichert. Funktionen können zudem eine <unitReference> besitzen. Sie stellt den Verweis auf eine Sicht dar und wird in Kapitel 6 beschrieben. Kanten müssen in der Lage sein, zwei andere Knontrollflusselemente zu verbinden. Dazu dient das Element <flow>. Es enthält zwei Attribute, die jeweils auf ein id-Attribut verweisen: @source und @target.

5 Grafik-Information

Graphische Information bezieht sich auf die Präsentation von EPK-Modellen in graphischen Modellierungswerkzeugen. Im Rahmen der Petri Net Markup Language (PNML) existiert hierzu ein umfassender Vorschlag, welcher weitgehend auch für EPML übernommen werden kann. Ähnlich wie das <graphics> Element der Kontrollflusselemente kann das Toplevel-Element <graphicsDefault> sowohl <fill>, <line> und <font> Standardeinstellungen definieren, jedoch kein <position> Element.

Alle vier Attribute des <position> Elements beziehen sich auf das kleinste Rechteck, das parallel zu den Achsen um das polygone Symbol eines Objektes herum gezeichnet werden kann. Die @x und @y Attribute beschreiben den Abstand jener Ecke des Symbols vom Ursprung des Koordinatensystems, der am nächsten am Ursprung ist. @width und @height beschreiben die Länge der Kanten des einschließenden Rechteckes. In PNML wird ein separates Dimension-Element für die Repräsentation dieser Aspekte benutzt. Kanten können mehrere Positions-Elemente besitzen, die Ankerpunkte des Kantenverlaufes beschreiben, sie haben keine width- und height-Attribute.

Das <fill> Element definiert die Füllfarbe des Innenraums eines Objektes. Kanten haben keine fill-Elemente. Das @color Attribut muss einen RGB-Wert oder eine vordefinierte Farbe gemäß Cascading Stylesheets 2 (CSS2) [Bo98] enthalten. Um eine kontinuierliche Variation der Füllfarbe darzustellen, kann eine optionale @gradient-color definiert werden. Die @gradient-rotation setzt die Orientierung des Gradienten auf entweder vertikal, horizontal oder diagonal. Falls eine URI im @image Attribut aufgeführt wird, werden die anderen Attribute ignoriert. Das <line> Element definiert die Außenlinie des Objektes. Das @shape Attribut verweist darauf, wie Kanten dargestellt werden: der Wert „line" repräsentiert eine lineare Verbindung der Ankerpunkte in Form eines Polygons; der Wert „curve" steht für eine quadratische Bezier-Kurve. Das <font> Element umfasst @family, @style, @weight, @size und @decoration Attribute, wie mit CSS2 conform sind. Zusätzlich zu PNML kann eine Schriftfarbe angegeben werden. @verticalAlign und @horizontalAlign spezifizieren die Ausrichtung des Textes. In PNML entspricht das align-Attribut dem horizontalAlign-Attribut von EPML, und das verticalAlign-Attribut ist über das PNML offset-Element abgedeckt. @rotation bezeichnet die Rotation des Textes im Uhrzeigersinn des Textes genau wie in PNML auch.

6 Sichten auf Prozesse

Sichtenbildung spielt eine wichtige Rolle für die Analyse und Konzeption von Prozessmodellen [Fr94]. Perspektiven im Allgemeinen haben sich als wertvoll für die Untergliederung der Spezifikation eines komplexen Systems erwiesen [Fi92]. Dieser Ansatz wurde für EPKs erweitert, um eine personalisierte Präsentation von Prozessmodellen zu ermöglichen [Be03].

Es gibt eine ganze Reihe von Vorschlägen an relevanten Perspektiven und Sichten für die Modellierung von Geschäftsprozessen. Die Architektur Integrierter Informationssysteme (ARIS) erweitert die EPK mit einer daten-, funktions-, organisations- und leistungsorientierten Sicht [Sc00]. Das PROMET-Konzept differenziert zwischen Geschäftsdimensionen, wobei explizit Organisation, Daten, Funktionen und Personal genannt werden [Ös95]. Eine tiefgehende Übersicht über die Definitionsmöglichkeiten von Organisationseinheiten in Workflow-Systemen findet sich in [RM98]. Die Beziehung zwischen rollenbasierter Zugriffskontrolle (RBAC) und Geschäftsszenarien wird in [NS02] analysiert und dient der Entwicklung einer Methode zur Ableitung von Rollenhierarchien. Mit speziellem Fokus auf Delegation wird in [AKV03] die Organisations-

perspektive in Workflow-Systemen strukturiert, welches in einem Metamodell mit Ressourcen, Organisationseinheiten, Benutzern und Rollen mündet. In [Wh03] und [BAN03] werden „swim lanes“ und „pools“ als Metapher zur graphischen Strukturierung von mehreren Parteien in einem Geschäftsprozess empfohlen. Neuerdings spielen WSDL-Beschreibungen [Ch01] von Web Services eine Rolle in Sprachen wie BPEL4WS als eine neue Kategorie von Ressourcen. Jenseits von Ressourcen wurden weitere Sichten wie etwa Risiko [BO02] oder Performance-Maßzahlen [IDS03b] vorgeschlagen, um nur einige zu nennen.

Die DAML-S Initiative hat sich das Ziel gesetzt, eine standardisierte Geschäftsprozess-Ontologie für Web Services zu entwickeln [DA03]. Dies ist eine schwierige Aufgabe, wenn man sich die Vielfalt der möglichen Perspektiven vor Augen führt. Es gibt Zweifel, ob eine standardisierte Ontologie überhaupt wünschenswert ist, da verschiedene Domänen und Geschäftssektoren maßgeschneiderte Metamodelle brauchen, die ihrem Geschäftsmodell am besten entsprechen [KK02]. Diese Argumente haben dazu geführt, dass EPML von dem Prinzip der Erweiterbarkeit geleitet wird und folglich keine standardisierten Sichten vordefiniert. Das `<view>` Element dient als Container für Einheiten einer speziellen Sicht und deren Beziehungen. Das `<unit>` Element beschreibt eine Einheit innerhalb einer Sicht mit Hilfe einer `@unitId` und eines `@name` Attributes. Die `<unitRelation>` drückt eine hierarchische Beziehung mit einem `@unitRef` Attribut und einem `@subUnitRef` Attribut aus. Die `@annotation` kann für eine detaillierte Bezeichnung der Relation genutzt werden. Es gibt zudem eine `@relationId`, um logisch zwischen verschiedenen Beziehungen zwischen zwei gleichen Elementen zu unterscheiden. Funktionselemente des Kontrollflusses können eine `<unitReference>` enthalten. Das `@role` und das `@value` Attribut ermöglichen die Spezifikation von zusätzlichen Informationen bezüglich der Beziehung zwischen einer Funktion und einer Unit.

7 Ausblick

In diesem Beitrag wurde ein Konzept für eine EPK Markup Language (EPML) vorgestellt. Dieser Ansatz dient dazu, ein Austauschformat für EPK-Modelle bereitzustellen. Es ist von den Prinzipien Lesbarkeit, Erweiterbarkeit, Tool-Orientierung und Syntaktische Richtigkeit geleitet. In den vorangegangenen Kapiteln wurden auch Best-Practices anderer graphenorientierten Spezifikationen betrachtet und die Entwurfsentscheidungen transparent gemacht. Dies beinhaltet eine detaillierte Diskussion der Prozessrepräsentation, der EPK-Prozesselemente und ihren Beziehungen untereinander, graphische Informationen sowie die Betrachtung von Sichten auf ein Geschäftsprozessmodell.

Das EPML-Konzept ist offen für Diskussionen und Anregungen, um innerhalb der EPK-Community einen Konsens bezüglich der Repräsentation von EPKs in EPML zu erreichen und EPML-Anwendungen zu fördern. Es gibt eine Reihe von Themen, die in der Zukunft in Angriff genommen werden. Zum ersten werden Transformationsskripte entwickelt werden, die die Nutzung von EPML als Austauschformat ermöglichen werden. Als zweites wird eine Transformation von EPML nach Scalable Vector Graphics (SVG)

[Fe03] die optische Erscheinung von EPML-Modellen standardisieren. Zum dritten werden XSLT-Skripte [Cl99] eine XML-basierte Syntax-Validierung von EPK-Modellen ermöglichen [MN03c]. Und zuletzt bedarf es weiterer Forschung im Bereich Sichten. Methodisch werden hierfür Techniken der Metamodellierung und des Semantic Web Berücksichtigung finden. Zudem bedarf es weitergehender Untersuchungen, um spezifische Sichten formal zu spezifizieren. Die Administration von dezentralen, lose verbundenen Modellen wird dabei ebenfalls eine Rolle spielen. In diesem Zusammenhang kann die Entwicklung von EPML – jenseits ihrer kurzfristigen Ausrichtung als Austauschformat – als Katalysator für all diese Themen dienen.

Literaturverzeichnis

[AKV03] van der Aalst, W.M.P.; Kumar, A.; Verbeek, H.M.W.: Organizational Modeling in UML and XML in the Context of Workflow Systems. In: Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), March 9-12, 2003, Melbourne, Florida, USA, S. 603-608.

[AL02] Arenas, M.; Libkin, L.: A normal form for XML documents. In: Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02), June 3-6, 2002, Madison, Wisconsin, USA, S. 85-96.

[AL03] Arenas, M.; Libkin, L.: An Information-Theoretic Approach to Normal Forms for Relational and XML Data. In: Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'03), June 9-12, 2003, San Diego, California, USA, S. 15-26.

[AN02] ANSI (Hrsg.): ASC X12 Reference Model for XML Design, July 2002. http://www.x12.org/x12org/comments/X12Reference_Model_For_XML_Design.pdf.

[An03] Andrews, T.; Curbera, F.; Dholakia, H.; Goland, Y.; Klein, J.; Leymann, F.; Liu, K.; Roller, D.; Smith, D.; Thatte, S.; Trickovic, I.; Weerawarana, S.: Business Process Execution Language for Web Services (BPEL4WS) Version 1.1. BEA, IBM, Microsoft, SAP, Siebel, 2003.

[Ar02] Arkin, A.: Business Process Modeling Language (BPML). BPMI.org, 2002.

[BAN03] Becker, J.; Algermissen, L.; Niehaves, B.: Prozessmodellierung in eGovernment-Projekten mit der eEPK. In: Nüttgens, M.; Rump, F.J. (Hrsg.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des 2. GI-Workshop EPK 2003, Bamberg 2003, S. 31-44.

[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. http://w3c.org/TR/2001/REC-xmlschema-1-20010502/.

[Be03] Becker, J.; Delfmann, P.; Falk, T.; Knackstedt, R.: Multiperspektivische ereignisgesteuerte Prozessketten. In: Nüttgens, M.; Rump, F.J. (Hrsg.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des 2. GI-Workshop EPK 2003, Bamberg 2003, S. 45-60.

[Bi95] Biskup, J.: Achievements of relational database schema design theory revisited. In: Libkin, L.; Thalheim, B.: Semantics in Databases, LNCS 1358, 1998, S. 29-54.

[Bi03] Billington, J.; Christensen, S.; van Hee, K.E.; Kindler, E.; Kummer, O.; Petrucci, L.; Post, R.; Stehno, C.; Weber, M.: The Petri Net Markup Language: Concepts, Technology, and Tools. In: van der Aalst, W.M.P.; Best, E. (Hrsg.): Applications and Theory of Petri Nets 2003, 24th International Conference, ICATPN 2003, Eindhoven, The Netherlands, June 23-27, 2003, Proceedings. LNCS 2679, 2003, S. 483-505.

[Bl03] Black, P.E.: NIST Dictionary of Algorithms and Data Structures, 2003. http://www.nist.gov/dads/.

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

[Bo98] Bos, B.; Lie, H.W.; Lilley, C.; Jacobs, I. (Hrsg.): Cascading Style Sheets, level 2 – CSS2 Specification. http://w3c.org/TR/CSS2, 1998.

[BO02] Brabänder, E.; Ochs, H.: Analyse und Gestaltung prozessorientierter Risikomanagementsysteme mit Ereignisgesteuerten Prozessketten. In: Nüttgens, M.; Rump, F.J. (Hrsg.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des 1. GI-Workshop EPK 2002, Trier 2002, S. 17-34.

[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. http://www.w3c.org/TR/2000/REC-xml-20001006/.

[CD99] Clark, J.; DeRose, S.: XML Path Language (XPath) Version 1.0, World Wide Web Consortium, Boston, USA, 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[Ch76] Chen, P.: The Entity-Relationship Model – Towards a Unitied view of Data. In: ACM Transactions on Database Systems (TODS) 1 (1976), S. 9-36.

[Ch01] Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S.: Web Service Description Language (WSDL) 1.1, World Wide Web Consortium, Boston, USA, 2001. http://www.w3.org/TR/wsdl.

[Cl99] Clark, J. (Hrsg.): XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium, Boston, USA, 1999. http://w3c.org/TR/1999/REC-xslt-19991116/.

[DA03] The DAML Services Coalition (Hrsg.): DAML-S: Semantic Markup for Web Services. Whitepaper Version 0.9. http://www.daml.org/services, 2003.

[DC03] Dublin Core Metadata Initiative: Dublin Core Metadata Element Set, Version 1.1: Reference Description. 2003. http://dublincore.org/documents/2003/02/04/dces/.

[Eb87] Ebert, J.: A Versatile Data Structure for Edge-Oriented Graph Algorithms. In: CACM (30)6, 1987, S. 513-519.

[EM01] Embley, D.W.; Mok, W.Y.: Developing XML documents with guaranteed “good” properties. In: Kunii, H.S.; Jajodia, S.; Sølvberg, A. (Hrsg.): Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling, LNCS 2224, 2001, S. 426 – 441.

[Fe03] Ferraiolo, J.; Jun, F.; Jackson, D. (Hrsg.): Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3c.org/TR/SVG11, 2003.

[FGW03] Favre, J.-M.; Godfrey, M.; Winter, A.: First International Workshop on Meta-Models and Schemas for Reverse Engineering - Workshop Description, to appear in: Proceedings Working Conference on Reverse Engineering (WCRE 2003), IEEE Computer Society, 2003.

[Fi92] Finkelstein, A.; Kramer, J.; Nuseibeh, B.; Finkelstein, L.; Goedicke, M.: Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. In: International Journal of Software Engineering and Knowledge Engineering. 2 (1992) 1, S. 31-57.

[Fr94] Frank, U.: Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung, GMD-Bericht Nr. 225, München et al. 1994.

[Ga02] Gartner Research: The BPA Market Catches Another Major Updraft. Gartner's Application Development & Maintenance Research Note M-16-8153, 12 June 2002.

[IDS01] IDS Scheer AG (Hrsg.): XML-Export und-Import mit ARIS 5.0, Stand Januar 2001, Saarbrücken, 2001.

[IDS03a] IDS Scheer AG (Hrsg.): Schnittstellen zu ARIS 6.0x / 6.1x / 6.2, Whitepaper Juli 2003, Saarbrücken. www.ids-scheer.de/sixcms/media.php/1049/ Uebersicht+Schnittstellen+ARIS+2003-07.pdf.

[IDS03b] IDS Scheer AG (Hrsg.): ARIS Process Performance Manager, Whitepaper 2003, Saarbrücken. www.ids-scheer.com/sixcms/media.php/1186/aris_ppm_whitepaper_e_v500.pdf.

[ISO01] Ketels, K.: ISO 15022 XML Design Rules, Technical Specification, 2001. http://xml.coverpages.org/ISO15022-XMLDesignRulesV23a.pdf.

[Ki03] Kindler, E.: On the semantics of EPCs: A framework for resolving the vicious circle (Extended Abstract). In: Nüttgens, M.; Rump, F.J. (Hrsg.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des 2. GI-Workshop EPK 2003, Bamberg 2003, S. 7-18.

[KK02] Karagiannis, D.; Kühn, H.: Metamodelling Platforms. Invited Paper. In: Bauknecht, K.; Min Tjoa, A.; Quirchmayer, G. (Hrsg.): Proceedings of the 3rd International Conference EC-Web 2002 - Dexa 2002, Aix-en-Provence, France, September 2002, LNCS 2455, Springer-Verlag, p. 182-196.

[KM94] Keller, G.; Meinhardt, S.: SAP R/3-Analyzer: Optimierung von Geschäftsprozessen auf der Basis des R/3-Referenzmodells, Walldorf, 1994.

[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.

[Me01] Mertz, D.: Subelement contents versus tag attributes. IBM DeveloperWorks – XML Zone, Nov 2001. http://www-106.ibm.com/developerworks/xml/library/x-tipsub.html.

[Me03] Mendling, Jan: Event-Driven-Process-Chain-Markup-Language (EPML): Anforderungen, Konzeption und Anwendung eines XML Schemas für Ereignisgesteuerte Prozessketten (EPK). In: Höpfner, H.; Saake, G. (Hrsg.): Proceedings des Studentenprogramms anlässlich des 10. Symposium "Datenbanksysteme für Business, Technologie und Web", Magdeburg 2003, S. 48-50.

[Mi99] Milner, R.: Communicating and Mobile Systems: The π-Calculus. Cambridge 1999.

[Mi03] Microsoft (Hrsg.): About the XML for Visio Schema. MSDN Library, 2003. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devref/HTML/XMLR_XMLBasics_818.asp

[MIS02] Mortgage Bankers Association of America (MISMO) (Hrsg.): MISMO XML Design Rules and Guidelines. Draft 2.0 RC3, 2002. http://www.mismo.org/mismo/docs/drftspc/mismoengguidelines.pdf.

[MM03] Mendling, J.; Müller, M.: A Comparison of BPML and BPEL4WS. In: Tolksdorf, R.; Eckstein, R.: Proceedings der 1. Konferenz "Berliner XML-Tage", Berlin, 2003, S. 305-316.

[MN02] Mendling, J.; Nüttgens, M.: Event-Driven-Process-Chain-Markup-Language (EPML): Anforderungen zur Definition eines XML-Schemas für Ereignisgesteuerte Prozessketten (EPK). In: Nüttgens, M.; Rump, F. (Hrsg.): EPK 2002 – Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des GI-Workshop EPK 2002, Trier 2002, S. 87-93.

[MN03a] Mendling, J.; Nüttgens, M.: EPC Modelling based on Implicit Arc Types. In: Godlevsky, M.; Liddle, S.W.; Mayr, H.C.: Proc. of the 2nd International Conference on Information Systems Technology and its Applications (ISTA), LNI Vol. P-30, Bonn 2003, S. 131-142.

[MN03b] Mendling, J.; Nüttgens, M.: XML-basierte Geschäftsprozessmodellierung. In: Uhr, W., Esswein, W.; Schoop, E. (Hrsg.): Wirtschaftsinformatik 2003 / Band II, Heidelberg, 2003, S. 161 -180.

[MN03c] Mendling, J.; Nüttgens, M.: EPC Syntax Validation with XML Schema Languages. In: Nüttgens, M.; Rump, F.J. (Hrsg.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des 2. GI-Workshop EPK 2003, Bamberg 2003, S. 19-30.

[NR02] Nüttgens, M.; Rump, J.F.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: Desel, J.; Weske, M. (Hrsg.): Promise 2002 - Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen, Proceedings GI-Workshop und Fachgruppentreffen (Potsdam, Oktober 2002), LNI Vol. P-21, Bonn 2002, S. 64-77.

[NS02] Neumann, G.; Strembeck, M.: A scenario-driven role engineering process for functional RBAC roles. In: 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002), S. 33-42.

[OMG03a] Object Management Group (Hrsg.).: Unified Modeling Language (UML) Specification, Marc 2003, Version 1.5, 2003.

[OMG03b] Object Management Group (Hrsg.): XML Metadata Interchange (XMI) Specification, May 2003, Version 2.0, 2003.

[Ös95] Österle, H.: Business Engineering. Prozess- und Systementwicklung, Band 1, Entwurfstechniken, Berlin, 1995.

[RM98] Rosemann, M.; zur Mühlen, M.: Evaluation of Workflow Management Systems - A Meta Model Approach. In: Australian Journal of Information Systems 6 (1998) 1, S. 103-116.

[Sc00] Scheer, A.-W.: ARIS business process modelling, Berlin et al., 2000.

[SW01] SWIFT (Hrsg.): SWIFTStandards XML Design Rules Version 2.3, Technical Specification, 2001. http://xml.coverpages.org/EBTWG-SWIFTStandards-XML200110.pdf.

[Wf02] Workflow Management Coalition (Hrsg.): Workflow Process Definition Interface – XML Process Definition Language, Document Number WFMC-TC-1025, October 25, 2002, Version 1.0, Lighthouse Point, USA, 2002.

[Wh03] White, S.A: Business Process Modeling Notation – Working Draft 1.0, Aug. 25, 2003. BPMI.org, 2003.

[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] M. Weber, E. Kindler: The Petri Net Markup Language. In: H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber (Hrsg.): Petri Net Technology for Communication Based Systems. LNCS 2472, 2002.

[WKR02] Winter, A.; Kullbach, B.; Riediger, V.: An Overview of the GXL Graph Exchange Language. In: Diehl, S. (Hrsg.) Software Visualization - International Seminar Dagstuhl Castle, LNCS 2269, 2001.