

XML-basierte Geschäftsprozessmodellierung

Jan Mendling, Markus Nüttgens

Universität Trier

Zusammenfassung: In diesem Beitrag wird ein Konzept zur XML-basierten Geschäftsprozessmodellierung vorgestellt. Dieses ermöglicht eine Verbesserung der Integrationsfähigkeit von Werkzeugen und Methoden zum Geschäftsprozessmanagement. Mit der Entwicklung einer EPK-Markup-Language (EPML) wird hierbei ein werkzeugunabhängiges Beschreibungsformat für Ereignisgesteuerte Prozessketten (EPK) vorgeschlagen, welches den Entwurfsprinzipien Lesbarkeit, Erweiterbarkeit, Tool-Orientierung und Syntaktische Richtigkeit genügt. Das Austauschformat bietet Werkzeugherstellern und Endanwendern einen transparenten und produktunabhängigen Bezugsrahmen zur XML-basierten Geschäftsprozessmodellierung.

Schlüsselworte: Modellierungswerkzeuge, Geschäftsprozessmodellierung, Geschäftsprozessmanagement, Schnittstellen, EPK, EPML, XML, XSLT

1 Einführung

Das Angebot an Modellierungswerkzeugen zum Geschäftsprozessmanagement (GPM) hat sich seit Beginn der 90er Jahre zu einem eigenständigen Marktsegment entwickelt. Eine jährlich veröffentlichte Studie von Gartner Research schätzt das globale Marktvolumen gegenwärtig auf über 500 Millionen \$ und prognostiziert ein durchschnittliches Marktwachstum von ca. 20% für die kommenden Jahre. Eine weitere Prognose betrifft die Anzahl der kommerziell verfügbaren Produkte. So soll sich diese Anzahl von derzeit 35 Produkten in den kommenden Jahren tendenziell halbieren [Ga01, Ga02]. Ein wesentliches Leistungsmerkmal ist demnach ist die Integrationsfähigkeit der Modellierungswerkzeuge untereinander und zu Prozessausführungsumgebungen [BS01, Nü02].

In einer Umgebung heterogener Werkzeuge und Methoden schlagen [WHB02] drei alternative Strategien zur Konvertierung vor, die als Peer-to-Peer-Strategie, Ring-Strategie und Intermediär-Strategie bezeichnet werden (vgl. Abb. 1). Diesen Strategien werden typische Gesamtkosten zugeordnet, die sich aus Kosten für den Konverterbau, Kosten für die Ausführung der Konvertierung, Kosten durch Konvertierungsverluste und Kosten für die Definition eines intermediären Formates zusammensetzen.

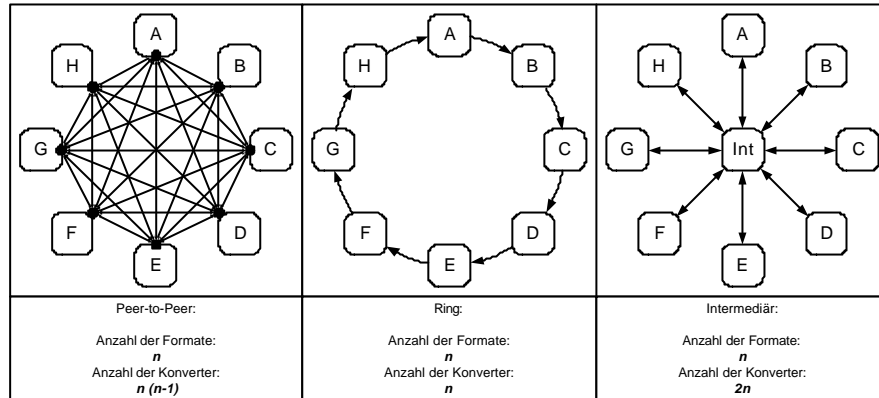


Abb. 1: Konvertierungsstrategien (vgl. [WHB02])

Die Peer-to-Peer-Strategie stellt die einfachste Variante dar. Hierbei werden zwischen jedem Paar von Formaten zwei Konverter, jeweils für die Hin- und die Rückrichtung, benötigt. Bei n Formaten ergeben sich dann $n(n-1)$ Konverter. Unterstellt man für den Konverterbau konstante Kosten, so wachsen diese quadratisch mit der Anzahl der Formate. Die Ausführung einer Konvertierung ist dagegen sehr günstig. Im Durchschnitt ist sie konstant mit den Kosten einer mittleren Konvertierung. Genauso ist von einem durchschnittlichen Konvertierungsverlust an Information auszugehen. Kosten für die Definition eines Intermediär-Formates fallen nicht an. Der Gesamtkostenverlauf in Abhängigkeit von der Anzahl der Formate dürfte also quadratisch sein. Bei sehr wenigen Formaten scheint diese Lösung dennoch sehr günstig zu sein.

Bei zwei Formaten ist die Ring-Strategie mit der Peer-to-Peer-Strategie identisch. Sie bietet jedoch den Vorteil, dass die Anzahl der Konverter der Anzahl der Formate entspricht. Folglich steigen die durchschnittlichen Kosten für den Konverterbau linear und nicht quadratisch mit der Anzahl der Formate, was gegenüber der Peer-to-Peer-Strategie schnell Vorteile einbringt. Die Kosten der Konvertierung steigen jedoch, da im Durchschnitt $n/2$ Konvertierungen ausgeführt werden müssen, bis das Zielformat erreicht ist. Besonders unvorteilhaft ist die Entwicklung der Konvertierungsverluste mit steigender Anzahl von Formaten. Im Extremfall bleibt lediglich die Schnittmenge an Information aller Formate erhalten. Es ist davon auszugehen, dass die Konvertierungsverluste überproportional mit der Anzahl der Formate zunehmen. Kosten für ein intermediäres Format fallen hier nicht an. Daraus ergibt sich die Vermutung, dass die Gesamtkosten der Konvertierung überproportional zu der Anzahl der Konverter ansteigen, jedoch bedeutend flacher als die Gesamtkosten der Peer-to-Peer-Strategie.

Die Intermediär-Strategie zeichnet sich dadurch aus, dass lediglich $2n$ Konverter benötigt werden, allerdings Kosten für die Definition eines intermediären Formates anfallen. Bis zur Erreichung des Zielformates sind zwei Schritte nötig. Die Ausführungskosten betragen dann das Doppelte des Durchschnittes einer Ausführung. Ebenso ist von konstanten Konvertierungsverlusten auszugehen, da sich Fehler nicht Potenzieren können. Zwar stellen die Kosten für die Definition eines intermediären Formates einen erheblichen Aufwand dar, dennoch schlagen sie nur einmal zu Buche und amortisieren sich mit steigender Anzahl an Formaten. Die Gesamtkosten dürften also linear verlaufen, wenn auch mit einem größeren Achsenabschnitt, der die Definitionskosten für das intermediäre Format darstellt.

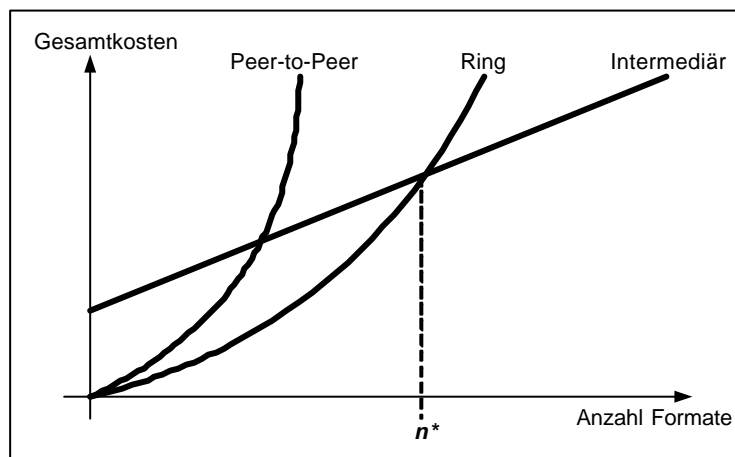


Abb. 2: Kosten für Strategien in Abhängigkeit von der Anzahl der Formate

Stellt man die Kostenverläufe in Abhängigkeit von der Anzahl der Formate gegenüber, so ergibt sich ein Bild wie in Abb. 2 dargestellt. Die Peer-to-Peer-Strategie ist bei zwei Formaten noch mit der Ring-Strategie identisch, zeigt dann jedoch einen steileren Kostenanstieg aufgrund der Vielzahl von benötigten Convertern. Die Intermediär-Strategie besitzt zwar hohe Anfangskosten durch die Definition eines intermediären Formates, besitzt dafür aber eine lineare Abhängigkeit von der Anzahl der Formate. Deshalb gibt es einen Punkt n^* , ab dem die Intermediär-Strategie kostengünstiger ist als die Ring-Strategie. Wir vermuten, dass diese Anzahl sehr niedrig ist. Dies ergibt sich zum einen aus der Heterogenität der Tools und der Methoden, welche sich ungünstig auf Konvertierungsverluste bei einer Ring-Strategie auswirken. Zum anderen ist die Definition eines intermediären Formates mit Hilfe von XML Schema-Sprachen effizient und kostengünstig möglich, was den Fixkostenanteil einer Intermediär-Strategie entlastet. Daraus ergibt sich die Motivation der Definition einer XML-Repräsentation für EPKs, genannt EPK-Markup-Language (EPML).

2 Geschäftsprozessmodelle und XML

Die eXtensible Markup Language (XML) [Br00] hat zwischenzeitlich eine enorme Verbreitung erreicht. Auch wenn XML selbst erst zu Beginn des Jahres 1998 vom World Wide Web Consortium (W3C) als Standard verabschiedet worden ist, wurde der Vorgänger Standard Generalized Markup Language (SGML) bereits 1986 als ISO 8879 standardisiert [ISO86]. So nutzt auch das World Wide Web mit HTML [RHJ99] eine SGML-Anwendung zur Seitenbeschreibung und demonstriert eindrucksvoll die Vorteile einer maschinen- und softwareunabhängigen Informationsaufbereitung. Da SGML selbst jedoch einige komplizierte Details beinhaltet, ist es für Online-Anwendungen nur bedingt geeignet [Lo00]. XML überwindet diese Unzulänglichkeiten und stellt eine flexible Möglichkeit zur anwendungsunabhängigen Informationsmodellierung dar.

2.1 XML-Standardfamilie

Einen guten Überblick über XML bietet das XML Information Set (XML Infoset) [CT01], das ein Metamodell für in XML-Dokumenten gespeicherte Information darstellt. Abb. 3 zeigt hierzu ein UML Modell. Die Objekte dieses Modells werden als „Information Items“ bezeichnet. Ankerpunkt ist das auf der linken Seite dargestellte Information Item „Document“. Es bildet den Container für alle weiteren Information Items. Wichtigste Struktureinheit sind „Elemente“, die weitere „Elemente“ und „Attribute“ als Kindknoten enthalten können. Textknoten von Elementen werden im XML Information Set als eine Liste von „Character“ Information Items beschrieben. Des Weiteren sieht das XML Infoset unter anderem „Document Type Declaration“ und „Namespace Declaration“ als Information Items vor.

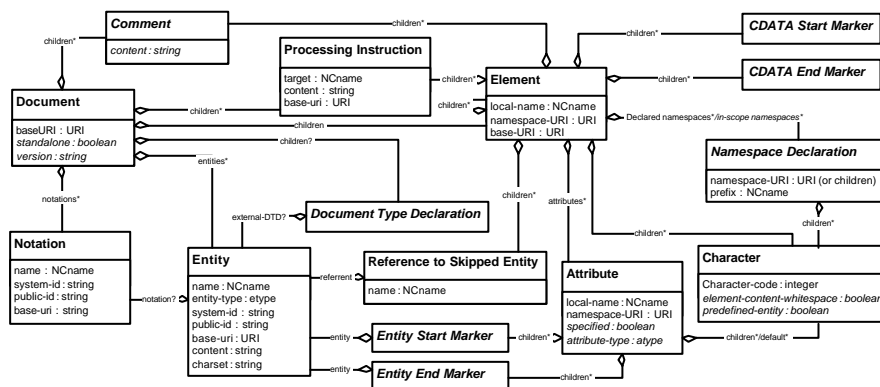


Abb. 3: UML Modell des XML Infosets (vgl. [BSL00])

XML beschreibt eine serialisierte Baumstruktur. Dabei muss ein XML-Dokument dem Kriterium der Wohlgeformtheit genügen [Br00]. Dies bedeutet, dass es ein oder mehrere Elemente enthalten muss. Genau eines davon, die Wurzel oder root, erscheint in keinem anderen Element. Für alle anderen Elemente gilt, dass der öffnende Starttag in demselben Element auftauchen muss wie auch der schließende Endtag. Namespaces [BHL99] dienen dazu, trotz der freien Definierbarkeit von XML-Vokabularien die Eindeutigkeit der Elemente zu sichern. Mit einem Namensraum wird ein Kontext definiert, in dem ein Element- oder Attributname eine bestimmte Bedeutung trägt. Der Namensraum wird als ein Uniform Resource Identifier (URI) spezifiziert, der meist auch die URL des entsprechenden Schemas bezeichnet. Somit wird es möglich, verschiedene XML-Vokabularien in einem XML-Dokument gemischt zu verwenden. Über den Namensraum der Elemente und Attribute ist es weiterhin möglich, die Validität in Bezug zu dem angegebenen Namensraum zu prüfen.

Bei der Validierung von XML-Dokumenten wird geprüft, ob die Elemente und deren Struktur den Bedingungen eines XML Schemas entsprechen. Die Zahl der XML Schema-Sprachen hat mittlerweile derart zugenommen, dass sich ein spezielles ISO Projekt mit der Systematisierung verschiedener Arten von Schemata unter dem Namen DSDL (Document Schema Definition Language) befasst [Ho02]. Die bekanntesten XML Schema-Sprachen lassen sich den Kategorien Regelbasiert, Grammatikbasiert und Objektorientiert zuordnen [VI02, S. 340]. Da XML als eine Untermenge von SGML konzipiert ist, übernimmt es aus diesem Kontext die Document Type Definition (DTD) [Br00], eine grammatik-basierte Schema-Sprache. Diese wird genutzt, um die Validität des XML-Dokumentes gemäß der Spezifikation des Schemas zu prüfen. Dabei kann ein XML-Dokument durchaus wohlgeformt, jedoch in Bezug auf eine bestimmte DTD nicht valide sein. Aufgrund einiger Nachteile von DTDs hat der World Wide Web Consortium einen neuen Standard namens XML Schema [Be01, BM01] verabschiedet. Im Folgenden wird diese Recommendation als W3C XML Schema bezeichnet, um im Vergleich mit anderen XML Schema-Sprachen Missverständnisse zu vermeiden.

Extensible Stylesheet Language Transformations (XSLT) [CI99] fußt ursprünglich auf der Idee, logische Dokumentstrukturen wie etwa ein XML-Dokument in eine betrachterorientierte Darstellung wie etwa HTML umzusetzen. Als Eingabe dient ein XML-Dokument. Ein XSLT-Prozessor führt darauf auf Basis eines XSLT-Skriptes verschiedene Transformationen durch, woraus letztlich eine XML-Ausgabedatei entsteht. Wichtigster Bestandteil von XSLT sind Template-Regeln und Muster. Ein Template definiert dabei, welche Ausgabe zu erzeugen ist, wenn ein bestimmtes Muster erkannt wird. Ein XSLT-Skript besteht aus einer Reihe solcher Template-Regeln, die über Template-Aufrufe ineinander verschachtelt sein können. Muster werden dabei in der XML Path Language (XPath) angegeben [CD99]. Diese benutzt eine nicht auf XML basierende Syntax, um eine Menge von Knoten in einem XML-Dokument zu beschreiben.

2.2 XML-Schnittstellen für Geschäftsprozessmodelle

Die zunehmende Verbreitung von XML-basierten Anwendungen und Austauschformaten birgt eine Reihe von Effizienzvorteilen bei der Datenkonvertierung. Ohne ein standardisiertes Serialisierungsformat wie XML ergibt sich folgende exemplarische Ausgangssituation: Zwei Anwendungen (1A und 2b) besitzen Exportschnittstellen und individuelle Exportformate. Die Konvertierung lässt sich dann in drei Schritte unterteilen: Schritt 1 mit dem Parsen des individuellen Exportformates von Anwendung 1A, Schritt 2 mit der Transformation von Format 1A nach Format 2b und Schritt 3 mit der Serialisierung der internen Darstellung im Konverter in das Exportformat von Anwendung 2b.

In einer Situation ohne XML ist der Programmierer gezwungen, einen Parser für jedes Dateiformat zu erstellen. Zudem muss er explizit programmieren, wie welche Daten eines Formates in Daten eines anderen Formates zu übertragen sind. Zuletzt muss er selbst programmieren, nach welchen Regeln die transformierten Daten in einem Format der Zielanwendung darzustellen sind. Diese aufwendigen Arbeiten werden meist dadurch erschwert, dass die Struktur der Exportfiles nicht transparent ist, oft sogar nicht einmal dokumentiert. Konverterbau ohne XML lässt sich somit als schwierig sowohl wegen des Programmieraufwandes als auch wegen der Intransparenz der Dateiformate beschreiben.

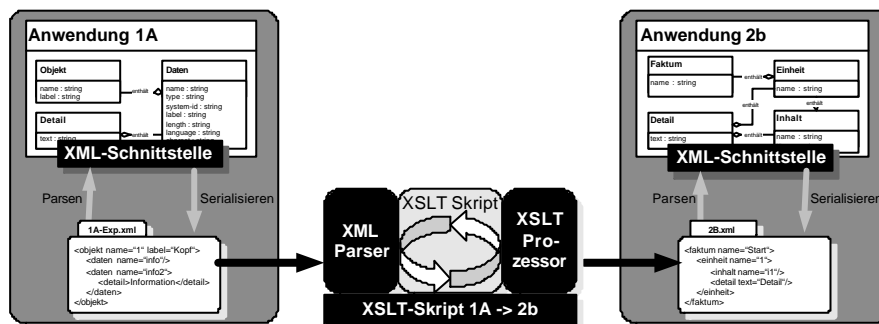


Abb. 4: XML-basierte Konvertierungsstrategie

Abbildung 4 zeigt die Situation basierend auf XML Technologien. Das Einlesen wird dem Programmierer durch frei verfügbare XML Parser abgenommen. Die Konvertierung selbst wird vereinfacht, da mit XSLT Template-Regeln angegeben werden können, für die spezifische XML-Strukturen erzeugt werden. Wie diese Template-Regeln die entsprechenden Strukturen im Eingabefile erkennen, wird vom Programmierer abgeschirmt. Ebenso werden die Ausgabestrukturen direkt in den Template-Regeln spezifiziert, was zu einer weiteren Entlastung führt. Des Weiteren werden Dateiformate mit Hilfe von XML Schema-Sprachen beschrieben, was eine ungleich größere Transparenz mit sich bringt. Der Programmierer

ist somit weitgehend von zugriffstechnischen Fragen entlastet und kann aus einer logischen Perspektive die Konvertierung mit XSLT realisieren. Dies senkt die Konvertierungskosten erheblich.

In Abb. 5 sind exemplarisch GPM-Modellierungswerkzeuge aufgeführt, welche über XML-Schnittstellen verfügen. Die Formate sind meist jedoch proprietär oder unterstützen Fremdformate nur unidirektional als Import. Daher besteht aus Sicht der Endanwender der Bedarf, herstellerunabhängige Formate auf Basis von XML bereitzustellen.

Hersteller	Tool
ATOSS Software	AENEIS
Computas	METIS
IDS Scheer	ARIS Toolset
IntraWare	Bonapart
IvyTeam	IvyFrame
MEGA International	MEGA Process
Microsoft	Visio
Proforma	ProVision Workbench
Promatis	INCOME Process Designer
Pulinco	TopEASE
SILVERRUN	SILVERRUN BPM
ViCon	ViFlow

Abb. 5: Modellierungswerkzeuge mit XML-basierten Schnittstellen

2.3 XML-Repräsentationen für Geschäftsprozessmodelle

Ereignisgesteuerte Prozessketten (EPK) [KNS92], Petri-Netzen [Pe62] und Aktivitätsdiagramme [OMG01] sind die wesentlichen in der Theorie und Praxis verbreiteten Methoden zur Geschäftsprozessmodellierung. Während für Petri-Netze und Aktivitätsdiagramme bereits Standardisierungskonzepte für XML-Formate vorliegen, ist dies für EPK-Modelle noch nicht der Fall.

Im Rahmen der Unified Modeling Language (UML) [OMG01] ist die Problematik werkzeugbezogener Schnittstellenformate 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. Als Anwendungsmöglichkeiten von XMI nennt Jeckle [Je00] die Codegenerierung aus OO-Modellen, die Modellvalidierung, Metrikenberechnung, Persistenzhaltung, Versionsverwaltung und Export/Import-Möglichkeiten. Die Kodierung in XMI erfolgt in zwei Schritten entsprechend der Metamodellhierarchie der Object Management Group (OMG). Als erstes wird das Modell als Instanz des entsprechenden Metamodells beschrieben. Anschließend werden die konkreten Objekte hinzugefügt. Die Namensgebung der Elemente stützt sich 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.

Anders als bei der Definition von XMI können die Autoren der Petri Net Markup Language (PNML) [WK02] nicht auf eine standardisierte Version der Modellierungsmethode zurückgreifen. Daher ist ihr Vorschlag eines PNML-Datenformates auch gleichzeitig ein Standardisierungsversuch innerhalb der Petri-Netz-Community. 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, wie etwa Pages und Module. Mit PNML liegt ein Austauschformat für Petri-Netze vor, welches sich durch Flexibilität und durch Strukturierungsmöglichkeiten auszeichnet. 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 ist das Konzept hilfreich beim Austausch von Petri-Netz-Modellen.

2.4 XML-Integriationsebenen für Geschäftsprozeßmodelle

Als konzeptionellen Rahmen für die Konvertierung zwischen verschiedenen Dateiformaten schlagen [WHB02] ein Schichtenmodell vor. Angewandt auf das Problem der Integration von Geschäftsprozessmodellen ergeben sich drei Ebenen, die zueinander in Beziehung gesetzt werden müssen: Die Format-Ebene, die Methoden-Ebene und die Tool-Ebene (vgl. Abb. 6).

Das Konzept der Integriationsebenen für die Modellierung von Geschäftsprozessen zeigt von unten nach oben eine zunehmende Spezifität. Während in der Format-Ebene lediglich eine Serialisierungsform für die Modelle festgelegt wird, bindet die mittlere Ebene die Modelle an ein methodenspezifisches Format. In der Tool-Ebene werden die Modelle im jeweils tooleigenen Metamodell ausgedrückt.

Die drei Ebenen implizieren zwei Arten von Konvertierungen, zum einen die Integration von Tools über die Methoden-Ebene, und zum anderen die Integration von Methoden über die Format-Ebene. Die Integration von Tools über die

Methoden-Ebene vollzieht sich in zwei Schritten und entspricht vom Vorgehen einer Intermediär-Strategie. Zuerst wird ein Modell eines toolspezifischen Exportfiles in das entsprechende anwendungsneutrale Methodenformat konvertiert. Dieses wird dann in ein anderes toolspezifisches Format konvertiert.

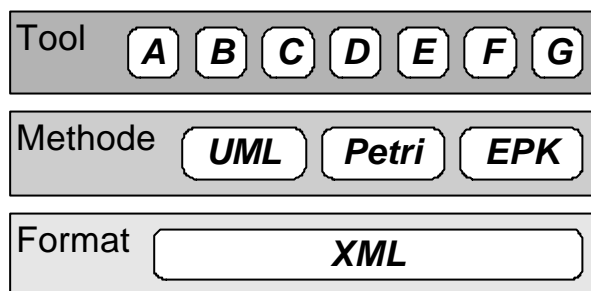


Abb. 6: XML-Integrations Ebenen für Geschäftsprozessmodelle

Die Integration von Tools und Methoden vollzieht sich in drei Schritten. Zuerst erfolgt die Konvertierung in das anwendungsneutrale Format der jeweiligen Methode. Danach erfolgt die Konvertierung zwischen den beteiligten Methoden über die Nutzung der gemeinsamen XML-Format-Ebene und vorzugsweise XSLT (vgl. 2.2.). Die methodischen Feinheiten der verschiedenen Modellierungstechniken erfordern hier eine Peer-to-Peer-Strategie. Zuletzt wird das toolneutrale Format der Zielmethode in das toolspezifische Format konvertiert.

Die Existenz anwendungsneutraler XML-Formate für verschiedene Modellierungstechniken ermöglicht es, die Frage der Konvertierbarkeit zwischen UML, Petri Netzen und EPKs losgelöst von einem oder mehreren Tools zu betrachten. Die weite Verbreitung von XML bietet hier viele Effizienzvorteile. XSLT ermöglicht die einfache Konvertierung zwischen verschiedenen XML-Vokabularien. Dies hat positive Effekte auf die Integrierbarkeit von Tools und Methoden. Vor diesem Hintergrund ist die Definition eines XML-basierten Austauschformates für EPKs, genannt EPML, zu verstehen.

3 EPK-Markup-Language (EPML)

3.1 Entwurfsaspekte

Die Entwurfsaspekte zur Definition einer EPK-Markup-Language (EPML) leiten sich aus dem Anwendungszusammenhang ab [MN02]. Anwender sollen in der Lage sein, die in EPML kodierten Prozessmodelle auch ohne spezielle Editier-

kenntnisse lesen und interpretieren zu können. Zudem sollen verschiedene Prozess-Sichten abgebildet werden können, was eine flexible Erweiterbarkeit der zugrundeliegenden Kontrollflussbeschreibung erfordert. Die Unterstützung des Austausches von Modelldaten zwischen heterogenen Modellierungswerkzeugen ist eine weitere zentrale Anforderung. Schliesslich sollte mit der Speicherung in EPML auch die syntaktische Richtigkeit der Modelle gesichert sein. Abbildung 7 fasst diese vier Aspekte zusammen.

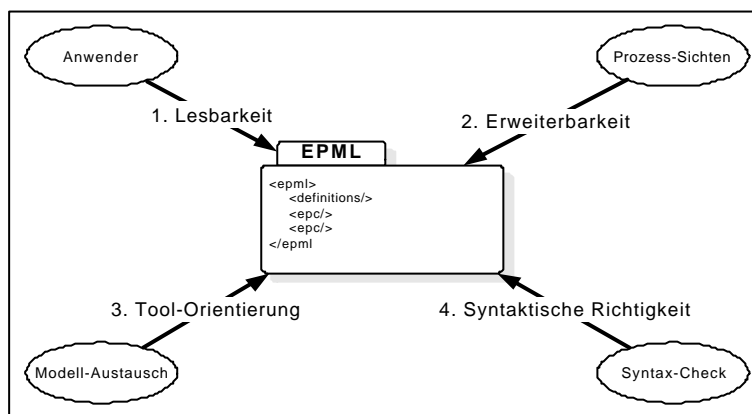


Abb. 7: EPML-Entwurfsaspekte

3.1.1 Lesbarkeit

Lesbarkeit als Entwurfsaspekt wird bereits explizit von den Autoren der PNML genannt [WK02]. Im Kontext der EPK bedeutet dies, dass EPK-Modelle unter einem Elementknoten <EPC> auftreten. Als Kindelemente sind dann nur die Elemente der EPK nach der ursprünglichen Definition von [KNS92] zugelassen, nämlich das Ereignis, die Funktion, Prozess-Schnittstelle, die Konnektoren AND, OR und XOR, und die Kontrollflusskante. Die Elemente werden mit einer Id identifiziert, die von den Kanten in einem FromId-Attribut und einem ToId-Attribut referenziert werden kann.

3.1.2 Erweiterbarkeit

Erweiterbarkeit zielt auf die Definition beliebiger Sichten auf einen EPK-Kontrollfluss ab. Dies erfordert einen Definitionsabschnitt zu Beginn einer EPML-Datei in der verschiedene Sichten beschrieben werden können. Des Weiteren müssen die Elemente einer Sicht unter einem Funktionsknoten referenziert und mit Rollen bzw. Werten belegt werden können. Zuletzt müssen auch Beziehungen unter Elementen einer Sicht abgebildet werden. Abbildung 8 verdeutlicht dies.

```

<EPML>
  <Definitions>
    <View Name="Organisation">
      <Unit Name="CEO" UnitId="100" />
      <Unit Name="Assistant" UnitId="101" />
      <Hierarchy RefId="100" SubId="101" />
    </View>
    <View Name="Costs" />
      <Unit Name="Average Duration" UnitId="200" />
    </View>
  </Definitions>
  <EPC EpcId="1">
    <Function Id="12">
      <UnitRef UnitRef="101" Role="Execution" />
      <UnitRef UnitRef="200" Role="5 min" />
    </Function>
  </EPC>
</EPML>

```

Abb. 8: EPML-Entwurfsaspekt „Erweiterbarkeit“

3.1.3 Tool-Orientierung

Tool-Orientierung beinhaltet drei Aspekte: Das Verwalten von Objektdefinitionen und -referenzen, das Mitführen von graphischen Positionsdaten, sowie das Bereitstellen von Annotations-, Dokumentations- und ToolInfo-Feldern zur semantischen Anreicherung des EPML-Files. Das Verwalten von gleichen Funktionen und Ereignissen in verschiedenen Prozessmodellen erfordert einerseits die Definition eines Objektes und andererseits dessen Referenzierung. Somit kann etwa eine Funktion „Abrechnung“ mehrfach in einem Modell auftauchen, aber logisch als Referenz auf eine Funktionsdefinition „Abrechnung“ verwaltet werden. Da eine solche Wiederverwendung nur für Ereignisse, Funktionen und Prozess-Schnittstellen sinnvoll erscheint, werden Konnektoren hiervon ausgenommen. Diese treten nur als Instanz in Prozessmodellen auf. Um Objekte eines Prozessmodells in Modellierungswerkzeugen graphisch darstellen zu können, werden Positions- und Größendaten gespeichert. Annotations-, Dokumentations- und ToolInfo-Felder ermöglichen die strukturierte Verwaltung von Zusatzinformationen, die für Endanwender oder für spezifische Anwendungen von Bedeutung sein können. Abbildung 9 erweitert den EPML-Code um diese Aspekte.

```

<EPML>
  <Annotation>
    <Documentation>EPML Intro</Documentation>
  </Annotation>

```

```

<Definitions>
  <ControlFlow>
    <EventDef DefId="1" Name="Start"/>
    <FunctionDef DefId="2" Name="Cook Coffee"/>
  </ControlFlow>
  <View Name="Organisation">
    <Unit Name="CEO" UnitId="100"/>
    <Unit Name="Assistant" UnitId="101"/>
    <Hierarchy RefId="100" SubId="101"/>
  </View>
  <View Name="Costs"/>
    <Unit Name="Average Duration" UnitId="200"/>
  </View>
</Definitions>
<EPC EpcId="1">
  <Event Id="11">
    <Graphic X="100" Y="100" H="20" W="120"/>
  </Event>
  <Function Id="12">
    <Graphic X="100" Y="150" H="20" W="120"/>
    <UnitRef UnitRef="101" Role="Execution"/>
    <UnitRef UnitRef="200" Role="5 min"/>
    <Arc FromId="11" ToId="12">
      <Graphic X="160" Y="120"/>
      <Graphic X="160" Y="150"/>
    </Arc>
  </EPC>
</EPML>

```

Abb. 9: EPML-Entwurfsaspekt „Tool-Orientierung“

3.1.4 Syntaktische Richtigkeit

Die drei ersten Aspekte beschreiben, welche Elemente in EPML enthalten sein müssen. Die syntaktische Richtigkeit der einzelnen EPKs und auch die Konsistenz einer Reihe von EPKs lässt sich hingegen nicht mit der Struktur der Markup-Elemente alleine beschreiben. Nüttgens/Rump [NR02] definieren eine Reihe von Eigenschaften, die von einer EPK erfüllt sein müssen. Diese lassen sich zu vier Blöcken von Bedingungen zusammenfassen, die hier knapp aufgeführt werden.

Der erste Block definiert Eigenschaften eines flachen EPK-Schemas. Ein flaches EPK-Schema ist ein Graph mit Elementen vom Typ Ereignis, Funktion, Konnektor und Prozess-Schnittstelle sowie einer Untermenge aus dem kartesischen Produkt dieser Elemente als Kanten. Dieser Graph ist gerichtet, einfach, zu-

sammenhängend und antisymmetrisch und enthält keine Zyklen, die nur aus Konnektoren bestehen. Es gibt mindestens ein Start- und mindestens ein Endereignis. Der zweite Block bezieht sich auf Kardinalitätseigenschaften der Elemente. Start-Ereignisse bzw. –Prozess-Schnittstellen haben keinen Vorgänger und genau einen Nachfolger. End-Ereignisse bzw. –Prozess-Schnittstellen haben genau einen Vorgänger und keinen Nachfolger. Andere Ereignisse und Funktionen haben genau einen Vorgänger und genau einen Nachfolger. Konnektoren haben entweder einen Vorgänger und mehr als einen Nachfolger oder mehr als einen Vorgänger und einen Nachfolger. Der dritte Block beschreibt Typkonsistenzregeln für diese Elemente. Ereignisse dürfen nur mit Funktionen und Prozess-Schnittstellen verbunden sein, eventuell über Konnektoren. Funktionen dürften nur mit Ereignissen und Prozess-Schnittstellen verbunden sein, möglicherweise über Konnektoren. Nach Ereignissen ist kein XOR- oder OR-Split erlaubt. Der vierte Block behandelt Eigenschaften, die von einer Menge von EPKs mit Hierarchierelationen erfüllt werden müssen. Jeder Prozess-Schnittstelle ist über eine Hierarchierelation genau eine andere EPK zuzuordnen. Jeder Funktion ist maximal eine andere EPK zugewiesen. Die Menge der vorangehenden Ereignisse einer hierarchisierten Funktion entspricht der Menge der Startereignisse der referenzierten EPK, die Menge der nachfolgenden Endereignisse entspricht der Menge der Endereignisse der referenzierten EPK. Bei Prozess-Schnittstellen ist die Menge der vorangehenden Ereignisse eine Teilmenge der Startereignisse der referenzierten EPK, und die Menge der nachfolgenden Ereignisse eine Teilmenge der Endereignisse der referenzierten EPK. Zudem darf eine EPK nicht über Hierarchierelationen mit sich selbst verbunden sein.

3.2 Umsetzungsaspekte

Die vier Entwurfsaspekte Lesbarkeit, Erweiterbarkeit, Tool-Orientierung und syntaktische Richtigkeit stellen unterschiedlich schwierige Anforderungen an eine XML Schema-Sprache, die zur Definition eines Austauschformates unentbehrlich ist. Die ersten drei beziehen sich auf die Anwesenheit von bestimmten Markup-Elementen und deren Struktur. Diese Anforderungen vermögen sämtliche XML Schema-Sprachen zu erfüllen. Von besonderer Schwierigkeit ist der vierte Aspekt der syntaktischen Richtigkeit. Da XML einer Baumstruktur aufbaut, mit EPML jedoch eine Reihe von EPK-Graphen serialisiert dargestellt werden sollen, bedarf es weitreichender Prüfungsmöglichkeiten jenseits der Struktur der Markup-Elemente. In wieweit die drei bekanntesten XML Schema-Sprachen diese Anforderungen erfüllen können, soll nachfolgend untersucht werden. W3C XML Schema ist dabei ein Vertreter der objektorientierten, RELAX NG eine grammatikbasierte und Schematron eine regelbasierte XML Schema-Sprache.

3.2.1 W3C XML Schema

W3C XML Schema ist eine Recommendation des World Wide Web Consortiums. Diese neue Schema-Sprache wurde entwickelt, um Unzulänglichkeiten der DTD in Sachen Namensraumunterstützung, Non-Konformität zur XML-Syntax und unzureichender Möglichkeiten zum Ausdruck von Bedingungen zu überwinden. W3C XML Schema gliedert sich in zwei Teile, Teil 1 für Strukturen [Be01] und Teil 2 für Datentypen [BM01]. Es unterstützt Namensräume, wird in XML-Syntax dargestellt und besitzt mit `<unique>`, `<key>` und `<keyref>` ein bedeutend flexibleres Repertoire als DTDs, um Eindeutigkeitsbedingungen auszudrücken. Zudem können Datentypen über reguläre Ausdrücke spezifiziert werden.

W3C XML Schema hat drei Nachteile. Als erstes können die syntaktischen Regeln der EPK nur ausgedrückt werden, wenn sie sich als Eindeutigkeitsbedingungen beschreiben lassen. Dies ist für einige Bedingungen möglich, etwa dass der Graph einfach sein soll. Mehrfachkanten zwischen zwei Elementen lassen sich über Eindeutigkeit verbieten, aber reflexive Kanten lassen sich somit nicht verhindern. Der zweite Nachteil ist die so genannte Unique Particle Attribution Rule, die ein Relikt aus der DTD-Zeit darstellt. Sie verbietet nicht-deterministische Inhaltsmodelle. Die Existenzbedingung von mindestens einem Startereignis kann deshalb nicht über ein nicht-deterministisches Inhaltsmodell erzwungen werden. Als drittes verbietet die Consistent Declaration Rule gleiche Elementnamen mit verschiedenen Datentyp-Deklarationen. W3C XML Schema kommt also lediglich dazu in Frage, die Struktur von EPML gemäß den Entwurfsaspekten Lesbarkeit, Erweiterbarkeit und Tool-Orientierung zu beschreiben. Eine Syntaxkontrolle ist nur äußerst begrenzt möglich.

3.2.2 RELAX NG

RELAX NG basiert auf dem Konzept der regulären Baum-Grammatik [CM01]. Seine Ausdrucksfähigkeit geht über die Mächtigkeit von W3C XML Schema hinaus [MLM00]. Da keinerlei Vergleichbares zu der Unique Particle Attribution Rule existiert, können nicht-deterministische Inhaltsmodelle beschrieben werden. In Sachen Bedingungen ist RELAX NG allerdings auf externe Datentypen angewiesen. Das bedeutet, dass lediglich die klassischen DTD-Typen wie ID, IDREF und IDREFS benutzt werden können, um Eindeutigkeitsbedingungen zu kodieren. Dies ist für die Umsetzung der EPML ein entscheidender Nachteil.

Dies lässt sich anschaulich am Beispiel des Verbotes von Mehrfachkanten zwischen Objekten verdeutlichen. Kanten müssen in ihren FromId- und ToId-Attributen Verweise auf andere Objekte enthalten, daher erhalten sie den Typ IDREF. Während mit W3C XML Schema immer noch eine Eindeutigkeitsbedingung für dieses Attributpaar über `<unique>` definierbar ist, kann dies in RELAX NG nicht ausgedrückt werden. Ein Vorteil für RELAX NG ergibt sich immerhin bei der Definition der Mindestexistenz von einem Ereignis und einer

Funktion in einer EPK. Hier können nicht-deterministische Inhaltsmodelle diese Bedingung erfüllen. Jedoch bieten sich in RELAX NG gleichermaßen wie in W3C XML Schema keine Möglichkeiten, die verbleibenden Bedingungen adäquat auszudrücken.

3.2.3 Schematron

Schematron [Je02] als regelbasierte XML Schema-Sprache folgt einem anderen Paradigma als W3C XML Schema oder RELAX NG. Statt der Beschreibung erlaubter Strukturen bietet Schematron Überprüfungen von nicht-erlaubten Strukturen. Dafür ermöglicht Schematron die Definition von Assertions, die positive Aussagen über Eigenschaften des Dokumentes in XPath formulieren. Damit erschließt sich die gesamte Flexibilität von XPath und dessen eingebauten Funktionen wie etwa `count()` oder String-Manipulation. Damit lassen sich eine Vielzahl von Bedingungen formulieren, die weder in RELAX NG noch in W3C XML Schema abgebildet werden können. Abbildung 10 zeigt die Prüfung von Reflexivität für Kanten in Schematron. Gleichermäßen lassen sich Mehrfachkanten ausschließen.

```
<sch:rule context="Arc">
  <sch:assert test="./@FromId!=./@ToId">
    An arc is not reflexive
  </sch:assert>
</sch:rule>
```

Abb. 10: Reflexivitätsprüfung mit Schematron

Probleme bekommt Schematron erst, wenn Knoten expandiert oder Abschlüsse berechnet werden müssen. Daher kann weder der Zusammenhang des EPK-Graphen noch die Existenz von Zyklen aus Konnektoren überprüft werden. Existenzbedingungen lassen sich jedoch einfach ausdrücken. Probleme ergeben sich bei der Prüfung von hierarchischen EPKs. Im Besonderen die Hierarchierelation und ihre Nicht-Reflexivitätsbedingung erzwingen die Berechnung transitiver Abschlüsse, was mit Schematron nicht möglich ist.

Zusammenfassend lässt sich sagen, dass Schematron trotz seiner Einschränkungen eindeutig am besten in der Lage ist, die syntaktischen Bedingungen der EPK zu prüfen. Wenn auch keine Möglichkeit zur Prüfung von transitiven Abhängigkeiten und Knotenexpansion besteht, so können doch immerhin eine große Zahl an einfachen Bedingungen formuliert werden.

3.3 Anwendungsaspekte

Nachfolgend wird in Abb. 11 und 12 schematisch ein Anwendungsbeispiel zur Abbildung einer EPK-Modellgrafik als EPML-Notation erläutert, wobei auf die Beschreibung der Grafikinformatoren verzichtet wird. Dem Ereignis „Start“ des Prozesses „Requirements Engineering“ folgt die Funktion „List requirements“. Im Anschluss an diese Funktion kann entweder das Ereignis „Can be fulfilled“ oder „Cannot be fulfilled“ eintreten. Der „Design Process“ ist über das gemeinsame Ereignis „Can be fulfilled“ mit dem „Requirements Process“ verbunden. Der zweite Prozess startet mit derselben Prozess-Schnittstelle und demselben Ereignis.

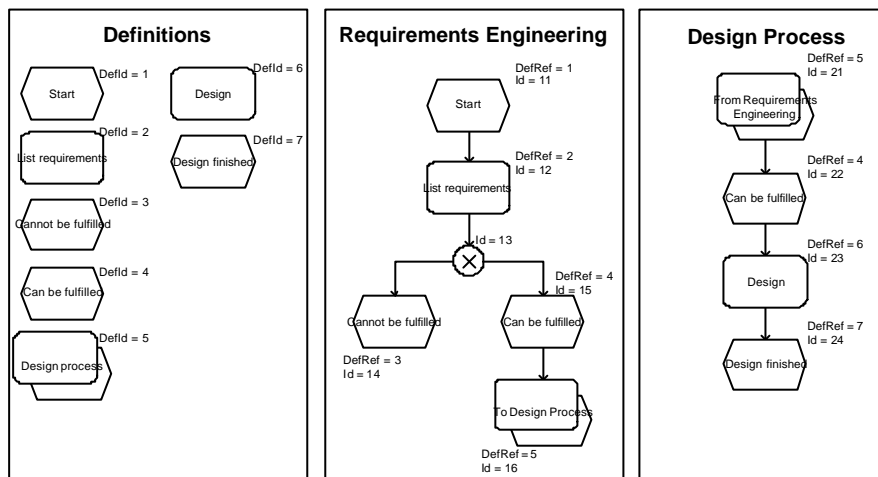


Abb. 11: EPK-Anwendungsbeispiel (mit EPML Attributen)

```
<EPML>
  <Definitions>
    <ControlFlow>
      <EventDef DefId="1" name="Start" />
      <FunctionDef DefId="2" name="List Requirements" />
      <EventDef DefId="3" name="Cannot be fulfilled" />
      <EventDef DefId="4" name="Can be fulfilled" />
      <ProcessInterfaceDef DefId="5" toname="To Design
        Process" fromname="From Requirements Engineering" />
      <FunctionDef DefId="6" name="Design" />
    </ControlFlow>
  </Definitions>
</EPML>
```

```
        <EventDef DefId="7" name="Design finished"/>
    </ControlFlow>
</Definitions>
<Dir Name="Hauptverzeichnis"/>
    <EPC EpcId="1" name="Requirements Engineering">
        <Event DefRef="1" Id="11"/>
        <Function DefRef="2" Id="12"/>
        <Arc FromId="11" ToId="12"/>
        <XOR Id="13"/>
        <Arc FromId="12" ToId="13"/>
        <Event DefRef="3" Id="14"/>
        <Arc FromId="13" ToId="14"/>
        <Event DefRef="4" Id="15"/>
        <Arc FromId="13" ToId="15"/>
        <ProcessIF DefRef="5" Id="16">
            <ToProcess LinkToEpcId="2"/>
        </ProcessIF>
        <Arc FromId="15" ToId="16"/>
    </EPC>
    <EPC EpcId="2" name="Design Process">
        <ProcessIF DefRef="5" Id="21"/>
        <Event DefRef="4" Id="22"/>
        <Arc FromId="21" ToId="22"/>
        <Function DefRef="6" Id="23"/>
        <Arc FromId="22" ToId="23"/>
        <Event DefRef="7" Id="24"/>
        <Arc FromId="23" ToId="24"/>
    </EPC>
</Dir>
</EPML>
```

Abb. 12: EPK-Anwendungsbeispiel in EPML-Notation (ohne Grafikdaten)

4 Ausblick

Für den Kunden bietet eine XML-basierte Geschäftsprozessmodellierung erhebliche Vorteile. Neben einem verbesserten Austausch von Prozessdokumentationen zwischen Organisationseinheiten entstehen neue Einsatzfelder wie beispielsweise der Begleitung von Post Merger Integrationen. Offene und konfigurierbare Schnittstellen zu anderen Modellierungswerkzeugen, Workflow-Management-Systemen, Prozesskostenrechnungssystemen, Simulationswerkzeugen oder Qualitätssicherungssystemen lassen sich effizienter implementieren und anpassen. Kostengünstigere Low-End Werkzeuge können im Front End Bereich zur Erfassung der Modelle genutzt werden, wo sie eine weite Verbreitung haben und High End Werkzeuge können im Back End für die Konsolidierung und Analyse eingesetzt werden. Dies ermöglicht die Umsetzung von Konzepten einer dezentral-koordinierten Geschäftsprozessmodellierung. Des Weiteren wird eine Wiederverwendung von betriebswirtschaftlichen Modellen für die Softwareentwicklung unterstützt.

Neben diesen Effekten ist es von weitreichender Bedeutung, dass die Unternehmen in einem XML-basierten Szenario keine Konzernentscheidung für ein Modellierungswerkzeug treffen müssen. Vielmehr können die Mitarbeiter das Werkzeug und die Methode einsetzen, mit dem Sie am besten vertraut sind. Hieraus resultiert eine höhere Akzeptanz des Business Engineerings und eine größere Unabhängigkeit von den Anbietern der Modellierungswerkzeuge.

Literaturverzeichnis

- [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.
- [BHL99] Bray, Tim; Hollander, Dave; Layman, Andrew: Namespaces in XML. World Wide Web Consortium, Boston, USA, 1999. Abruf von <http://www.w3.org/TR/1999/REC-xml-names-19990114/> 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.w3.org/TR/2000/REC-xml-20001006/> am 20.10.2002.
- [BS01] Bullinger, H.-J.; Schreiner, P. (Hrsg.): Business Process Management Tools - Eine evaluierende Marktstudie über aktuelle Werkzeuge, Fraunhofer IRB Verlag, Stuttgart 2001.

- [BSL00] Box, Don; Skonnard, Aaron; Lam, John: Essential XML – Beyond Markup, 2nd print, Boston et. al., 2000.
- [CD99] Clark, J.; DeRose, S.: XML Path Language (XPath) Version 1.0, World Wide Web Consortium, Boston, USA, 1999. Abruf von <http://www.w3.org/TR/1999/REC-xpath-19991116> am 20.10.2002.
- [CI99] Clark, J. (Hrsg.): XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium, Boston, USA, 1999. Abruf von <http://www.w3.org/TR/1999/REC-xslt-19991116/> am 20.10.2002.
- [CM01] Clark, J.; Murata, M.: RELAX NG Specification, 3 December 2001. URL: <http://www.oasis-open.org/committees/relax-ng/spec.html>.
- [CT01] Cowen, John; Tobin, Richard (Hrsg.): XML Information Set. World Wide Web Consortium, Boston, USA, 2001. Abruf von <http://www.w3.org/TR/2001/REC-xml-info-set-20011024/> am 20.10.2002.
- [Ga01] Gartner Research: The BPA/M Market Gets a Boost From New Features. Gartner's Applications Development & Management Research Note, M-13-5295, 16 May 2001.
- [Ga02] Gartner Research: The BPA Market Catches Another Major Updraft. Gartner's Application Development & Maintenance Research Note M-16-8153, 12 June 2002.
- [Ho02] Holman, Ken: ISO/IEC 19757 JTC 1/SC 34 – DSDL Document Schema Definition Languages, Date 2002/06/10, Canada, 2002. Abruf von <http://www.dSDL.org> am 20.10.2003.
- [ISO86] International Organisation for Standardization (Hrsg.): ISO 8879:1986. Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML), Genf, 1986.
- [Je00] Jeckle, M. C.: Metamodellierung als Ansatz zur Lösung der Modellaustauschproblematik im Umfeld objektorientierter Modellierungssprachen. In: Proceedings International Knowledge Technology Forum, Leipzig 2000. Abruf von <http://www.jeckle.de> am 20.10.2002.
- [Je02] Jelliffe, R.: The Schematron Assertion Language 1.5, 2002-10-01. URL: <http://www.ascc.net/xml/resource/schematron/Schematron2000.html>.
- [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.
- [Li00] Lienert, Christian: Integration of business processes with XML, in: Proceedings of XML Europe 2000, 13-16 June 2000, Paris, 2000. Abruf von <http://www.infoloom.com/gcaconfs/WEB/paris2000/S01-04.HTM> am 20.01.2003.
- [Lo00] Lobin, Henning: Informationsmodellierung in XML und SGML, Berlin et. al., 2000.
- [MLM00] Murata, M.; Lee, D.; Mani, M.: Taxonomy of XML Schema Languages using Formal Language Theory. In: Extreme Markup Languages, Montreal 2001. URL: <http://www.cobase.cs.ucla.edu/tech-docs/dongwon/mura0619.pdf>.

- [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 GI-Arbeitskreistreffen (Trier, November 2002), Trier 2002, S. 87-93.
- [Nu02] Nüttgens, M.: Rahmenkonzept zur Evaluierung von Modellierungswerkzeugen zum Geschäftsprozessmanagement. In (Gesellschaft für Informatik (GI) e.V.): Informationssystem-Architekturen, Wirtschaftsinformatik Rundbrief der GI Fachgruppe WI-MobIS, 9, 2002.
- [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.
- [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.
- [Pe62] Petri, Carl: Kommunikation mit Automaten. Schriften des RheinischWestfälischen Institutes für instrumentelle Mathematik, Bonn, 1962.
- [RHJ99] Raggett, D.; Le Hors, A.; Jacobs, I.: HTML 4.01 Specification, World Wide Web Consortium, Boston, USA, 1999. Abruf von <http://www.w3.org/TR/html401> am 20.10.2002
- [VI02] van der Vlist, E.: XML Schema, Sebastopol et. al. 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.