Markus Nüttgens, Frank J. Rump (Hrsg.)

EPK 2002

Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten

Workshop der Gesellschaft für Informatik e.V. (GI)
und Treffen ihres Arbeitkreises „Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)“

21.-22. November 2002 in Trier

Proceedings

Veranstalter

veranstaltet vom GI-Arbeitskreis "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)" der GI-Fachgruppe WI-MobIS (FB-WI) in Kopperation mit der GI-Fachgruppe EMISA (FB-DBIS) und der GI-Fachgruppe Petrinetze (FB-GInf).

Dr. Markus Nüttgens (Sprecher)
Wirtschaftsinformatik II
(Lehrstuhlvertretung)
Universität Trier
Postfach 3825
D-54286 Trier
Email: markus@nuettgens.de

Prof. Dr. Frank J. Rump (stellv. Sprecher)
Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven
Fachbereich Technik
Constantiaplatz 4
D-26723 Emden
Email: rump@informatik-emden.de

EPK 2002 / Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Hrsg.: Markus Nüttgens, Frank J. Rump. – Trier 2002

Vorwort

Ereignisgesteuerte Prozessketten (EPK) haben sich in der Praxis als Beschreibungsmittel für betriebliche Abläufe etabliert. Mit dem Aufbau der Arbeitsgruppe "Formalisierung und Analyse Ereignisgesteuerter Prozessketten (EPK)" im Jahre 1997 wurde ein erster Schritt unternommen, einen organisatorischen Rahmen für Interessenten und Autoren wesentlicher Forschungsarbeiten zu schaffen und regelmäßige Arbeitstreffen durchzuführen (Organisatoren: Markus Nüttgens, Andreas Oberweis, Frank J. Rump). Im Jahr 2002 wurden die Arbeiten der "informellen“ Arbeitsgruppe in den GI-Arbeitskreis "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)" der GI-Fachgruppe WI-MobIS (FB-WI) überführt und inhaltlich erweitert. Der Arbeitskreis soll Praktikern und Wissenschaftlern als Forum zur Kontaktaufnahme, zur Diskussion und zum Informationsaustausch dienen. Die Aktivitäten des Arbeitskreises werden unter der Internetadresse http://www.epk-community.de dokumentiert (aktuell: 80 Mitglieder)

Der vorliegende Tagungsband enthält neun vom Programmkomitee ausgewählte und auf dem Workshop präsentierte Beiträge. Zu jedem Beitrag wurde in einem vereinfachten Begutachtungsverfahren zunächst ein Ausgangsgutachten erstellt und dem Programmkomitee zur Stellungnahme zugeleitet. Im Falle von abweichenden Stellungnahmen wurde ein Zweitgutachten angefertigt.

Die Beiträge decken ein breites Spektrum zur Spezifikation und Anwendung Ereignisgesteuerter Prozessketten (EPK) ab:

- Fuzzy-Ereignisgesteuerte Prozessketten,
- Prozessorientierte Risikomanagementsysteme,
- Refactoring von Ereignisgesteuerten Prozessketten,
- Workflow Management mit Ereignisgesteuerten Prozessketten,
- Syntax- und Semantikdefinition,
- XML-Schemata für Ereignisgesteuerte Prozessketten,
- Aspekte der Aus- und Weiterbildung von ModelliererInnen.

Der Tagungsband wurde ausschließlich in digitaler Form publiziert und ist frei verfügbar unter: http://www.epk-community.de/epk2002/epk2002-proceedings.pdf.

Wir danken den AutorInnen, den Mitgliedern des Programmkomitees und dem Sponsor HRW Consulting Factory AG (Saarbrücken) für die Beträge zur Realisierung des Workshops.

Trier und Emden, im November 2002

Markus Nüttgens
Frank J. Rump

Programmkomitee

Jörg Becker, Universität Münster
Jörg Desel, Katholische Universität Eichstätt
Markus Nüttgens, Universität Trier (Vorsitz)
Andreas Oberweis, Universität Frankfurt a.M.
Christian Reiter, HRW Consulting Factory AG, Saarbrücken
Peter Rittgen, Technische Universität Darmstadt
Frank J. Rump, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven
Reinhard Schütte, Universität Essen
Michael Seubert, SAP AG, Walldorf (Baden)

Organisation

Markus Nüttgens, Universität Trier
Frank J. Rump, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven

Sponsoren

HRW Consulting Factory AG, Saarbrücken
(http://www.hrw-consulting.com/)

Inhaltsverzeichnis

Fuzzy-Ereignisgesteuerte Prozessketten

Geschäftsprozessmodellierung unter Berücksichtigung unscharfer Daten

Oliver Thomas[1], Claus Hüsselmann[2], Otmar Adam[1]

[1] Institut für Wirtschaftsinformatik (IWi)
im Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3, Geb. 43.8
66123 Saarbrücken
{thomas|adam}@iwi.uni-sb.de

[2] IDS Scheer AG
Altenkesseler Straße 17
66115 Saarbrücken
c.huesselmann@ids-scheer.de

Abstract. Berührungspunkte zwischen Prozess- und Wissensmanagement treten auf, wenn gesammeltes Prozesswissen zur Gestaltung, Steuerung und Ausführung betrieblicher Vorgänge genutzt wird. Dieses Wissen liegt meist in unstrukturierter Form vor, da es sich um Erfahrungswerte handelt. Entscheidungen werden daher häufig aus unscharfen Bedingungen oder vage formulierten Zielvorstellungen abgeleitet. Zur computergestützten Gestaltung und automatisierten Steuerung der Prozesse mit Workflow-Management-Systemen wird das notwendige Wissen in die Sprache der Anwendungssysteme übersetzt, die von der Dichotomie der klassischen Aussagenlogik geprägt sind. Dabei gehen nicht nur wesentliche Teile des Wissens verloren, eine Änderung der Prozessbeschreibung wird außerdem erschwert. Dieser Beitrag untersucht, wie unscharfe Daten zum adäquaten Design wissensintensiver und schwach strukturierter Geschäftsprozesse und ihrer Implementierung in Anwendungssystemen genutzt werden können. Als notwendige Voraussetzung wird die Geschäftsprozessmodellierung in Form Fuzzy-Ereignisgesteuerter Prozessketten identifiziert. Diese Methodenerweiterung wird anhand eines Beispiels motiviert und in Form eines Metamodells spezifiziert.

1 Management wissensintensiver Geschäftsprozesse

Das Management von Geschäftsprozessen umfasst die Planung, Gestaltung, Steuerung und Kontrolle betrieblicher Abläufe [SNZ95, S. 426]. Geschäftsprozesse werden in Form von Modellen beschrieben, anhand derer untersucht werden kann, inwiefern sich Effizienzsteigerungspotenziale durch eine Reorganisation bestehender Prozesse ergeben [SET00, S. 22]. Diese Modelle dienen als Grundlage zur Steuerung der Geschäftsprozesse durch Workflow-Management-Systeme [Ja95; VB96]. Zur Modellierung und Ausführung von Geschäftsprozessen sind in der Vergangenheit eine Vielzahl von Konzepten, Methoden und Werkzeugen entwickelt worden. Semi-formale Modellierungsmethoden dienen dabei als Mittler zwischen den Mitarbeitern einer Organisation, die über aufgabenspezifisches Wissen verfügen, und denjenigen, die über methodenspezifisches Wissen verfügen, wie etwa Berater oder Anwendungsentwickler [He94, S. 44f; Or97, S.

11f]. Auch zu den Anwendungssystemen selbst wird eine Schnittstelle geschaffen, sofern diese die Modelle automatisiert umsetzen können. So wird bei Prozessen, deren Ausführung von Bedingungen und Entscheidungen abhängt, durch die Modellierung das notwendige Prozesswissen erhoben, visualisiert und gespeichert.

Bei stark strukturierten Prozessen, die ohne Veränderungen wiederholt werden (z. B. Buchhaltung, Rechnungslegung, Instandhaltung) kann der Ablauf automatisiert unterstützt werden. Die Informationen und Entscheidungen im Umfeld wissensintensiver, schwach strukturierter Prozesse (z. B. Softwareentwicklung, Konstruktion, Beratung) beruhen oftmals auf Erfahrungswerten und damit implizitem Wissen [St92; DJB96]. Dieses ist nur schwer extrahierbar und liegt in der Regel in verbaler, informaler Form vor.

Es besteht eine Inkompatibilität mit den von der Dichotomie der klassischen Aussagenlogik geprägten Anwendungssystemen: In die Ablaufsteuerung wissensintensiver Prozesse gehen oftmals vage, unscharfe Informationen als Stellgrößen ein. Entschlüsse werden meist aus unscharfen Prämissen, wie „angemessener Gewinn“, „akzeptable Stückkosten“ oder „hoher Objektwert“, abgeleitet. Obwohl die in diesen Prämissen verwendeten Adjektive nicht präzise sind, ist mit ihnen jedoch zur Erfassung konkreter Unternehmungssituationen eine zusätzliche und bedeutsame Information verbunden. Durch die Übersetzung vager Prozessbeschreibungen in die scharfen Konstrukte der Modellierungsmethoden und Computersprachen geht ein wesentlicher Teil des ursprünglich vorhandenen Prozesswissens verloren. Die automatisierte Unterstützung wissensintensiver Prozesse erfüllt ihre Aufgaben daher häufig nicht adäquat.

Im vorliegenden Beitrag werden zunächst die Erscheinungsformen von Unschärfe im Umfeld wissensintensiver und schwach strukturierter Geschäftsprozesse aufgezeigt. Anschließend werden existierende, auf der Fuzzy-Set-Theorie basierende Ansätze zur Verarbeitung unscharfer Informationen in der Unternehmungsmodellierung analysiert. Diese Überlegungen bilden die Grundlage zur Fuzzy-Erweiterung der Modellierungsmethode der Ereignisgesteuerten Prozesskette (EPK), die an einem vereinfachten Beispiel demonstriert wird. Abschließend werden Nutzenpotenziale und Perspektiven für das Geschäftsprozessmanagement dargelegt.

2 Unschärfe in schwach strukturierten Geschäftsprozessen

Entscheidungen auf der Basis vager oder qualitativer Informationen gehören zur Klasse der Entscheidungen bei Unsicherheit, deren fehlende Detailbetrachtung in der klassischen Abgrenzung gegen deterministische und stochastische Modelle kritisch hinterfragt wird [Bo93, S. 124f]. Als *Unschärfe* wird in diesem Beitrag die Unsicherheit hinsichtlich von Daten und ihrer Interdependenzen verstanden. Auslöser der Unschärfe können die Realität selbst, die Sprache als Modellbildung über die Realität oder die Verwendung der Sprache sein [Bo93, S. 76f]. Durch die Identifikation verschiedener Formen von Unschärfe, die sich über die Entstehungsgründe von Unschärfe definieren [Hö97, S. 35ff], ist eine Konkretisierung des Begriffs im Umfeld wissensintensiver Prozesse möglich.

Die Komplexität des Umweltsystems und die Wahrnehmungsgrenzen des Menschen bedingen *informationale* Unschärfe. So enthalten wissensintensive Prozesse kurzlebige In-

formationen aus einer Vielzahl von Quellen, sodass zu einem festen Zeitpunkt nur ein Teil des Gesamtprozesses erfasst werden kann, der jedoch während der Erfassung anderer Teilaspekte bereits veraltet. Menschliche Präferenzordnungen sind nicht exakt bestimmbar [Kü89, S.61 ff], sodass es zu einer mit der informationalen Unschärfe verwandten Vagheit des Zielsystems kommt. So impliziert z. B. das Ziel „wesentliche Verminderung der Durchlaufzeit“ zwar Maßnahmen, jedoch lassen sich wegen der nicht explizierten Höhe der angestrebten Änderung und der unklaren Wertungsinterdependenzen mit anderen Zielen keine exakten Handlungen ableiten.

Die Beschreibung der Realität mit natürlicher Sprache erzeugt die *intrinsische* (auch: *verbale* oder *linguistische*) Unschärfe. Sowohl die Bildung eines sprachlichen Modells als auch die Kontextsensitivität von sprachlichen Aussagen tragen zur Entstehung dieser Unschärfe bei. Dicht verwoben hiermit ist auch die Ungenauigkeit in sprachlichen Vergleichen. Die Aussage „der Objektwert ist viel höher als x“ ist ein Beispiel hierfür.

Das für den Menschen übliche *größenordnungsmäßige Erfassen* der Realität erzeugt ebenfalls Unschärfe. Die Verwendung ungenauer Daten kann jedoch vorteilhaft sein, wenn geeignete Messmethoden fehlen, der Realweltausschnitt von hoher Dynamik geprägt ist oder nicht exakt ermittelbare Abhängigkeiten bestehen.

Bei der Entwicklung betriebswirtschaftlicher Geschäftsprozessmodelle und der dazu notwendigen Quantifizierung verbaler Attribute sollte die Unschärfe qualitativer Problemstellungen als verhaltensrelevant akzeptiert werden.

Die *Fuzzy-Set-Theorie* versucht die Trennung zwischen einer modell- und verfahrenstechnisch notwendigen Präzision einerseits sowie einer empirisch wünschenswerten Berücksichtigung qualitativer Informationen andererseits zu überwinden und einen Anteil an fehlender Präzision sowie Vagheit und Unsicherheit bei Modellierungsprozessen zu tolerieren [BZ70; Kr96]. Kernpunkt der Fuzzy-Theorie ist es, Zustände (von Objekten) nicht ausschließlich mit „wahr“ oder „falsch“ zu bewerten, sondern Zwischenstufen zuzulassen. Der ursprünglichen Idee von ZADEH folgend [Za65] wird die klassische Mengenlehre, d. h. die Theorie der scharfen Mengen, durch die Beschreibung *unscharfer Mengen (Fuzzy-Mengen)* erweitert.[1] Mit Fuzzy-Mengen lassen sich *linguistische Variablen* formulieren, die nicht Zahlen, sondern Wörter als Werte annehmen.

Abbildung 1 zeigt links die linguistische Variable „Objektwert“. Sie weist die Terme „gering“ und „hoch“ auf. Die Zugehörigkeit eines Objektwerts zu diesen Mengen ist durch die Zugehörigkeitsfunktionen μ_{gering}, μ_{hoch} ausgedrückt.[2] Der Objektwert 26.000 €

---

[1] Für jedes Element ω einer vorgegebenen (scharfen) Grundmenge Ω wird der Grad der Zugehörigkeit zu einer Teilmenge $A \subseteq \Omega$ durch einen Wert $\mu_A(\omega)$ einer Abbildung $\mu_A : \Omega \rightarrow [0;1]$ ausgedrückt. Man wählt diese *Zugehörigkeitsgrade* aus dem Intervall [0;1] und gibt folgende Interpretation: Je größer der Zugehörigkeitsgrad eines Elementes bzgl. einer (unscharfen) Menge ist, desto mehr gehört das Element zu dieser Menge. μ_A wird *Zugehörigkeitsfunktion* der *unscharfen Menge (Fuzzy-Menge)* $\{(\omega;\mu_A(\omega))|\ \omega \in \Omega\}$ genannt. Zur Erläuterung grundlegender Konzepte der Fuzzy-Theorie sei auf [DP80; Zi01] verwiesen.

[2] Aus Vereinfachungsgründen wird im Hinblick auf das folgende Beispiel auf die Abbildung zusätzlicher Terme, wie „sehr gering“ oder „sehr hoch“, verzichtet.

gehört z. B. zu 0.82 der Fuzzy-Menge „gering“ und zu 0.47 der Fuzzy-Menge „hoch“ an. Diese Abbildung scharfer Werte auf unscharfe Mengen heißt *Fuzzifizierung*. In einem scharfen Kontext wäre es nur möglich, z. B. einen Objektwert ab 26.000 € als „hohen“ Objektwert zu charakterisieren, während 25.999 € bereits als „gering“ gelten würde.

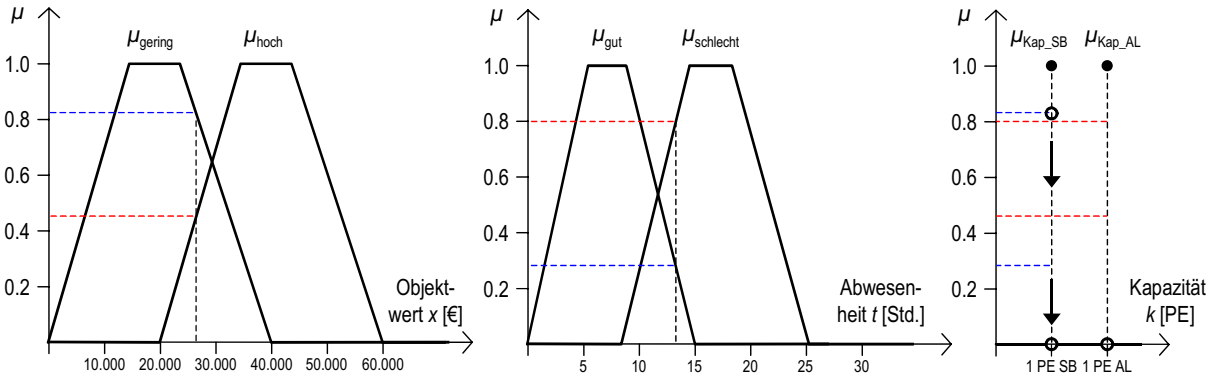


Abbildung 1: Zugehörigkeitsfunktionen

Ein *Fuzzy-System* besteht aus Eingangs- und Ausgangsvariablen, deren jeweilige Attribute durch Regeln, bestehend aus Prämissen- und Konklusionsteil, z. B. der Form „WENN Objektwert = gering DANN Priorität = mittel“, miteinander verknüpft sind. Durch Inferenzverfahren werden die Eingangs- und Ausgangsvariablen einander zugeordnet. Für eine ausführbare Aktion, z. B. „Priorität festlegen“, wird ein scharfer Wert der Ausgangsvariablen benötigt. Ein Defuzzifizierungsschritt liefert diesen scharfen Wert [Ro93; Zi01, S. 171ff].

Im Kontext der Modellierung und Ausführung wissensintensiver, schwach strukturierter Geschäftsprozesse gibt es viele Einflussgrößen, deren Ränder nicht scharf voneinander abgegrenzt sind, sondern fließende Übergänge besitzen. Mit Fuzzy-Mengen können diese modelliert werden.

3 Existierende Ansätze unscharfer Unternehmungsmodellierung

Im Bereich der Unternehmungsmodellierung wurden bereits Erweiterungen klassischer Methoden entwickelt, welche die positiven Effekte der Fuzzy-Theorie integrieren.

Die Fuzzy-Erweiterung des Entity-Relationship-Modells (ERM) [Ch76] wurde von ZVIELI und CHEN beschrieben [ZC86]. Hierbei können Entitytypen, Beziehungstypen und Attributmengen Fuzzy-Werte annehmen.

Die Berücksichtigung dieser fuzzifizierten Datenstrukturen führt konsequent zur Verarbeitung der unscharfen Daten in den entsprechenden betrieblichen Geschäftsprozessen. Insbesondere der Bereich der Produktionsplanung und -steuerung (PPS) tritt dabei in den Vordergrund [Bi97]. Im Dienstleistungssektor werden kaum geschäftsprozessorientierte Verfahren der Fuzzy-Technologie angewendet. Funktional orientierte Anwendungen, wie Entscheidungsprobleme im Bankenwesen, z. B. Kreditwürdigkeitsprüfung oder Aktienkursanalysen, zählen zu den Haupteinsatzgebieten [Po97].

Zur Beschreibung dynamischer Aspekte betrieblicher Informationssysteme werden unter anderem *Petri-Netze* [Pe62] eingesetzt. Das zweiwertige Verhalten von Stellen und Transitionen eines Petri-Netzes ist bei der Abbildung wissensintensiver und schwach strukturierter Prozesse jedoch von Nachteil. Um das Systemverhalten auch bei unscharfen Prozessbedingungen oder unvollständigen, vagen Informationen darstellen zu können, wurden Petri-Netze durch Fuzzy-Konzepte erweitert. Das *Fuzzy-Petri-Netz* [Li80; Li89] entsteht durch die Projektion mehrerer scharfer Petri-Netze, bei der die Strukturinformationen als unscharfe Mengen abgebildet werden.

Petri-Netze sind originär in einer technischen, systemnahen Philosophie beheimatet und werden wegen ihrer mathematisch-formalen Starrheit, die zu einer Minderung der Praxistauglichkeit führt, häufig kritisiert [UB99]. Im Gegensatz dazu ist die Geschäftsprozessmodellierung mit der Methode der *Ereignisgesteuerte Prozesskette (EPK)* [KNS92] aus einer eher betriebswirtschaftlich-organisatorisch orientierten Sichtweise motiviert. Die EPK besitzt, nicht zuletzt aufgrund ihrer Anwendungsorientierung und einer umfassenden Werkzeugunterstützung, einen hohen Grad der Verbreitung und Akzeptanz in der Praxis [Nü97]. Sie ist Bestandteil des ARIS-Toolset der IDS Scheer AG [Sc96] sowie des Business Engineering und Customizing des SAP R/3-Systems [KT98].

BECKER, REHFELDT, TUROWSKI [Be97] zeigen am Beispiel der industriellen Auftragsabwicklung mit unscharfen Informationsobjekten die Berücksichtigung unscharfer Daten in der Geschäftsprozessmodellierung mit Ereignisgesteuerten Prozessketten exemplarisch auf. Als wesentliche, mit Unschärfe in Form von Unsicherheit behaftete exogene Eingangsdaten werden vage Vertriebsinformationen betrachtet, die in vorläufige Kundenaufträge umgewandelt werden. Diese „unscharfe Ergänzung“ der Prozesse wird durch schattierte Objekte visualisiert.

4 Fuzzy-Ereignisgesteuerte Prozessketten

Die zuvor genannten Beispiele verdeutlichen, dass die Integration unscharfer Daten in die Ablauflogik der EPK keine Erweiterung der Methode bedingt. Sie beschreiben vielmehr das betriebswirtschaftliche Konzept zur Berücksichtigung unscharfer Daten und deren inhaltliche Abbildung im Modell. Anders verhält es sich mit der Steuerung des Kontrollflusses, wobei die Automatisierbarkeit durch Workflow-Management-Systeme sowie die flexible Optimierung der Ablauflogik wesentliche Aspekte darstellen. Hierbei ist es erforderlich, dass Syntax und Semantik der EPK erweitert werden.

Ausgangsbasis der Betrachtung ist der in Abbildung 2 gegebene einfache Verwaltungsprozess, bei dem die Weiterbearbeitung eines Objekts durch Aufgabenträger von der Größe des Objektwerts abhängt. Die Verfügbarkeit des Abteilungsleiters (AL) stellt einen Engpassfaktor dar, der zu Verzögerungen führen kann. Die Verfügbarkeit des Sachbearbeiters (SB) wird als prozessintern angenommen.

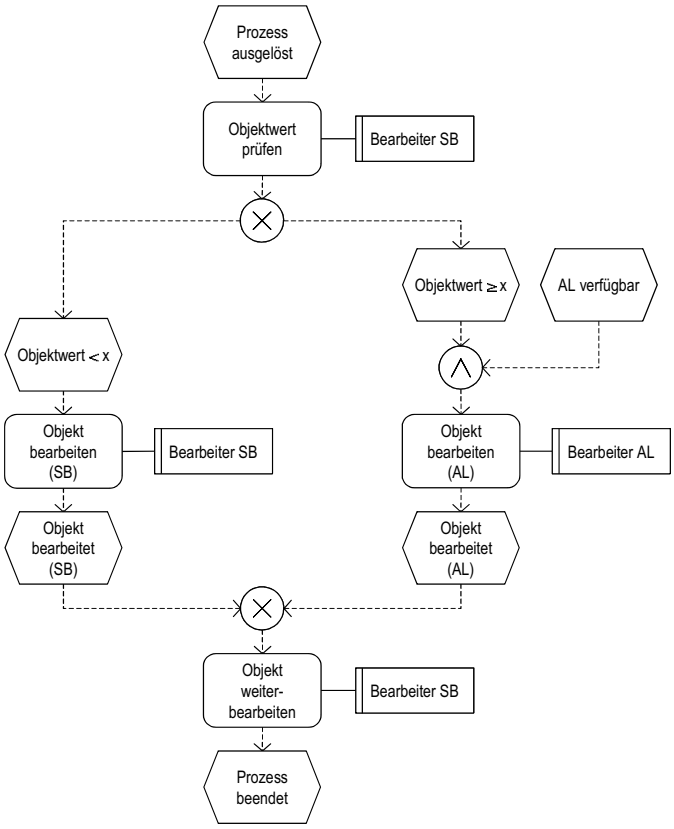


Abbildung 2: Ereignisgesteuerte Prozesskette „Objekt bearbeiten“

Die scharf formulierten Bedingungen werden durch die linguistischen Variablen „Objektwert“ und „Verfügbarkeit“ ausgedrückt. Als linguistische Terme werden die Begriffe „gering“ und „hoch“ für den Objektwert definiert sowie „gut“ und „schlecht“ für die Verfügbarkeit AL. Die semantischen Regeln der linguistischen Variablen seien beispielhaft durch ein vereinfachtes Regelsystem beschrieben:

(1)	WENN Objektwert = gering	(2)	WENN Objektwert = hoch
	UND Verfügbarkeit AL = schlecht		UND Verfügbarkeit AL = gut
	DANN Bearbeitung durch SB		DANN Bearbeitung durch AL

Die Zugehörigkeitsfunktionen μ_{gering}, μ_{hoch}, μ_{schlecht}, μ_{gut} sind in Abbildung 1 dargestellt. Im Beispiel wird eine einfache Repräsentation gewählt. Die Regeln führen in Abänderung des scharfen Kontrollflusses zu der in Abbildung 3 dargestellten Fuzzy-Ereignisgesteuerten Prozesskette – kurz: *Fuzzy-EPK*. Dabei ist dem Fuzzy-Operator „⊗̃“ das um die verwendeten Zugehörigkeitsfunktionen der linguistischen Terme ergänzte Regeldiagramm der Abbildung 3, rechts, hinterlegt. Zur Ermittlung des Ausgangswerts des Fuzzy-Operators wird das Prozessverhalten bettreffend der möglichen Bearbeitungsintensitäten von Funktionen spezifiziert. Dazu wird angenommen, dass Funktionen nur scharf schalten können, d. h. sie werden ausgeführt oder nicht. Damit stellen sich die Bearbeitungskapazitäten der Funktionen Objekt bearbeiten (SB) und Objekt bearbeiten (AL) als „Singletons“ über dem jeweiligen Aufgabenträger dar (Abbildung 1, rechts).

Zur Realisierung des Inferenzvorgangs wird die Min-Max-Methode gewählt. Da zur Bearbeitung der Funktionen Objekt bearbeiten (SB) und Objekt bearbeiten (AL) keine Unter- oder Überlastkapazitäten zugelassen sind und die Aufgabenträgerzuordnung eindeutig erfolgt, ergibt sich in Abbildung 1 ein scharfer Ausgangswert, d. h. die Kontrollflussentscheidung, die Bearbeitung von Bearbeiter SB durchführen zu lassen.

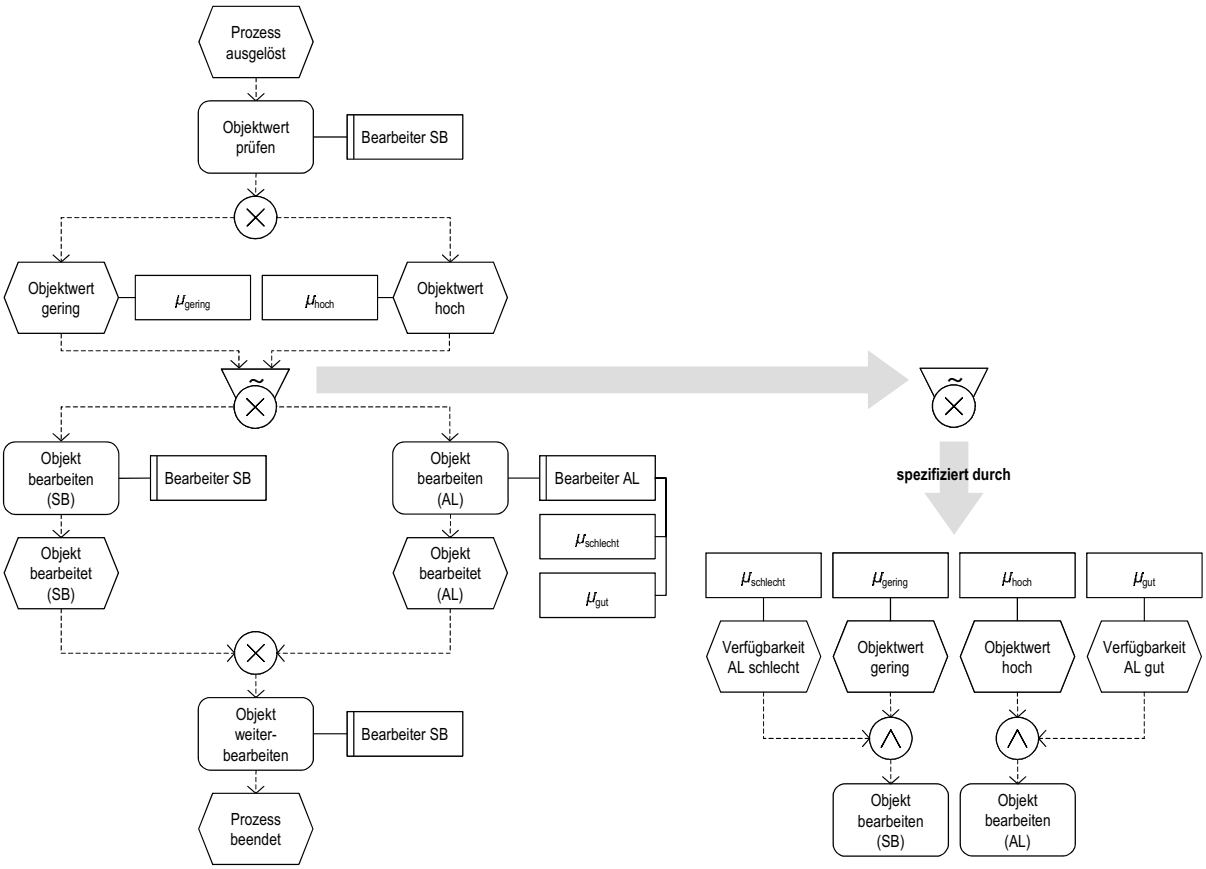


Abbildung 3: Fuzzy-Ereignisgesteuerte Prozesskette „Objekt bearbeiten“

Durch das Fundament der Fuzzy-Logik wird die Komplexität, die aufgrund der Flexibilisierung der Prozesssteuerung entsteht, durch eine dem Fachanwender zugängige Notationsform substituiert. Die syntaktischen Mittel der Fuzzy-EPK beschränken sich auf die in Abbildung 4 gegebenen Elemente, welche die „konventionelle“ EPK nur geringfügig ergänzen. Die Erweiterung beschränkt sich damit im Wesentlichen auf (unscharfe) Kontrollflussobjekte, die Kontrollflusssteuerung und Ressourcenkapazitäten. Damit wird ein weicher Übergang von einer „ungenauen“ zur systematisch fuzzifizierten Modellierung möglich. Insbesondere dem Endanwender in den Fachbereichen kommen die „Daumenregeln“ der Fuzzy-EPK entgegen. Das Regelwerk (Gestaltung der Zugehörigkeitsfunktionen, Inferenzregeln etc.) belastet die grafische Darstellung nicht. Ein solcher Effekt ist typisch für die Einsatzgebiete und -potenziale der Fuzzy-Logik: Die komplexe Realität wird durch ein einfaches Modell adäquat abgebildet [Zi93, S. 1f]. Auf diese Weise wird eine hohe Akzeptanz der Methodik erreicht.

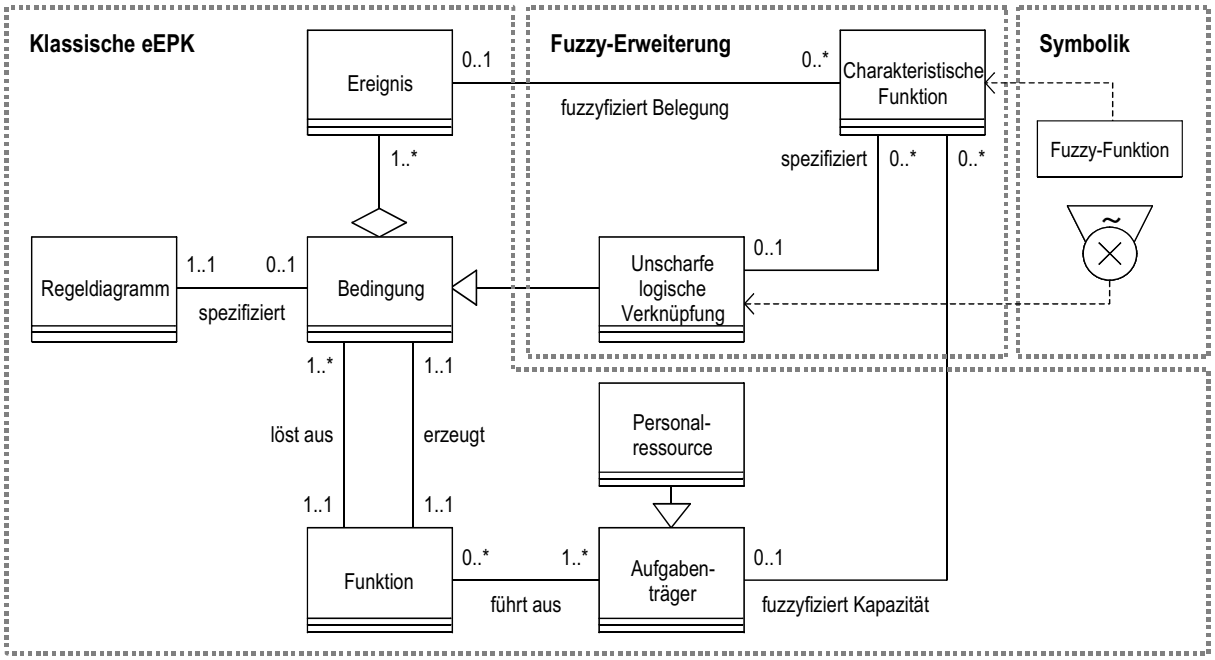


Abbildung 4: Metamodell und Symbolik der Fuzzy-EPK

5 Zusammenfassung und zukünftige Forschungsfragen

Die Abbildung von Prozessen und Strukturen von Unternehmungen und öffentlichen Verwaltungen in Form von Modellen trifft an vielen Stellen auf Unschärfe. Betriebswirtschaftlich-organisatorische Modelle berücksichtigen diese Unschärfe bis dato nicht durchgehend systematisch. Insbesondere die Geschäftsprozessmodellierung bietet Potenzial zur Umsetzung unscharfer Konzepte bis hin zur Etablierung und Steuerung flexiblerer Formen der Ablauforganisation.

Mit der Geschäftsprozessmodellierung auf der Basis Fuzzy-Ereignisgesteuerter Prozessketten wurde in dieser Arbeit ein Ansatz für die flexible und informationstechnisch gestützte Regelung und Steuerung von Geschäftsprozessen dargestellt. Speziell die Kontrollflusssteuerung von Prozessen erfuhr dabei eine konzeptionelle, semantische und syntaktische Erweiterung. Der Ansatz besteht in der kombinierten Anwendung von Geschäftsprozessmodellierung, Fuzzy Logic sowie Workflow Management und zeigt eine bisher kaum berücksichtigte Domäne des Geschäftsprozessmanagements auf.

Eine Konkretisierung der Zielsetzung im weiteren Verlauf ihrer Forschungstätigkeit sehen die Autoren in der (1) systematischen Erweiterung weiterer Prozess- und Organisationsaspekte durch Fuzzy-Technologien, (2) der Implementierung eines Werkzeugs zur unscharfen Unternehmungsmodellierung, und (3) in der Implementierung von Wissensspeicher, Netzwerk und Funktionalität eines Fuzzy-Workflow-Management-Systems, d. h. eines integrierten Anwendungssystems, das die flexible informationstechnisch gestützte Regelung und Steuerung wissensintensiver Geschäftsprozesse ermöglicht. Da im Bereich der Geschäftsprozessmodellierung [BS01] und der Steuerung von Geschäftsprozessen [Ha02] Produkte existieren, ist eine Neuentwicklung nicht erforderlich – sondern eine Erweiterung bestehender Systeme.

Literaturverzeichnis

[Be97] Becker, J.; Rehfeld, M. D.; Turowski, K.: Industrielle Auftragsabwicklung mit unscharfen Informationsobjekten. In: [Bi97], S. 85-98.

[Bi97] Biethahn, J., Hönerloh, A., Kuhl, J., Nissen, V. (Hrsg.): Fuzzy-Set-Theorie in betriebswirtschaftlichen Anwendungen, München 1997.

[Bo93] Bosch, H.: Entscheidung und Unschärfe. Bergisch Gladbach 1993.

[BS01] Bullinger, H.-J.; Schreiner, P. (Hrsg.): Business process management tools: eine evaluierende Marktstudie über aktuelle Werkzeuge. Stuttgart 2001.

[BZ70] Bellman, R. E.; Zadeh, L. A.: Decision-Making in a Fuzzy Environment. Management Science 17 (1970) 4, S. B141-B164.

[Ch76] Chen, P. P.-S.: The entity-relationship model – toward a unified view of data. ACM Transactions on Database Systems 1 (1976) 1, S. 9-36.

[DJB96] Davenport, T.; Jarvenpaa, S. L.; Beers, M. C.: Improving Knowledge Work Processes. Sloan Management Review 37 (1996) 4, S. 53-65.

[DP80] Dubois, D.; Prade, H. M.: Fuzzy sets and systems: theory and applications. New York 1980.

[Ha02] Hales, K.: Workflow Management Products. 10. Aufl., SODAN Uxbridge (England)2002.

[He94] Hesse, W. et al.: Terminologie in der Softwaretechnik – Ein Begriffssystem für die Analyse und Modellierung von Anwendungssystemen. Teil 1: Begriffssystematik und Grundbegriffe. Informatik-Spektrum 17 (1994) 1, S. 39-47.

[Hö97] Hönerloh, A.: Unscharfe Simulation in der Betriebswirtschaftslehre. Göttingen 1997.

[Ja95] Jablonski, S.: Workflow-Management-Systeme: Modellierung und Architektur, Bonn 1995.

[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, Nr. 89, Saarbrücken 1992.

[Kr96] Kruse, R.: Fuzzy-Systeme – Positive Aspekte der Unvollkommenheit. Informatik Spektrum 19 (1996) 1, S. 4-11.

[KT98] Keller, G.; Teufel, T.: SAP R/3 prozeßorientiert anwenden: iteratives Prozeß-Prototyping zur Bildung von Wertschöpfungsketten. 2. Aufl., Bonn 1998.

[Kü89] Kühn, M. A.: Flexibilität in logistischen Systemen. Heidelberg 1989.

[Li80] Lipp, H.-P.: Die Anwendung der unscharfen Mengentheorie für ein Steuerungskonzept zur operativen Fuehrung komplexer Systeme. Karl-Marx-Stadt, Diss., 1980.

[Li89] Lipp, H.-P.: Ein Konzept eines unscharfen Petri-Netzes als Grundlage für operative Entscheidungsprozesse in komplexen Produktionssystemen. Karl-Marx-Stadt, Diss., 1989.

[Nü97] Nüttgens, M.: Ereignisgesteuerte Prozeßkette (EPK) – Forschungsansätze in der wissenschaftlichen Literatur und Praxis. In: Desel, J.; Reichel, H. (Hrsg.): Grundlagen der Parallelität, Technische Berichte der TU Dresden, Fakultät Informatik, Dresden 1997.

[Or97] Ortner, E.: Methodenneutraler Fachentwurf. Stuttgart 1997.

[Pe62] Petri, C. A.: Kommunikation mit Automaten. Darmstadt, Techn. Hochsch., Diss., 1962.

[Po97] Popp, H.: Einsatz der Fuzzy-Technik in Industrie und Dienstleistungsbereich – ein Überblick. In: [Bi97], S. 23-38.

[Ro93] Rommelfanger, H.: Fuzzy-Logik basierte Verarbeitung von Expertenregeln. OR-Spektrum 15 (1993), S. 31-42.

[Sc96] Scheer, A.-W.: ARIS-Toolset: Von Forschungs-Prototypen zum Produkt. Informatik-Spektrum 19 (1996) 2, S. 71-78.

[Sc96] Scheer, A.-W.: ARIS-House of Business Engineering. In: Scheer, A.-W. (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Nr. 133, Saarbrücken 1996.

[SET00] Scheer, A.-W.; Erbach, F.; Thomas, O.: E-Business – Wer geht? Wer bleibt? Wer kommt? In: Scheer, A.-W. (Hrsg.): E-Business – Wer geht? Wer bleibt? Wer kommt? Heidelberg 2000, S. 3-45.

[SNZ95] Scheer, A.-W.; Nüttgens, M.; Zimmermann, V.: Rahmenkonzept für ein integriertes Geschäftsprozeßmanagement. Wirtschaftsinformatik 37 (1995) 5, S. 426-434.

[St92] Starbuck, W.: Learning by knowledge-intensive firms. Journal of Management Studies 29 (1992) 6, S. 713-740.

[UB99] von Uthmann, C.; Becker, J.: Petri Nets for Modeling Business Processes – Potentials, Deficits and Recommendations. In: Colloquium on Petri Net Technologies for Modelling Communication Based Systems, Berlin, October 21-22, 1999, S. 127-167.

[VB96] Vossen, G.; Becker, J. (Hrsg.): Geschäftsprozeßmodellierung und Workflow-Management – Modelle, Methoden, Werkzeuge. Bonn 1996.

[Za65] Zadeh, L. A.: Fuzzy sets. Information and Control 8 (1965) 3, S. 338-353.

[ZC86] Zvieli, A.; Chen, P. P.-S.: Entity-Relationship Modeling and Fuzzy Databases. Proceedings of the Second International Conference on Data Engineering, February 5-7, 1986, IEEE Computer Society, Los Angeles 1986, S. 320-327.

[Zi01] Zimmermann, H.-J.: Fuzzy set theory – and its applications. 4. Aufl., Dordrecht 2001.

[Zi93] Zimmermann, H.-J. (Hrsg.): Fuzzy-Technologien: Prinzipien, Werkzeuge, Potentiale. Düsseldorf 1993.

Analyse und Gestaltung prozessorientierter Risikomangementsysteme mit Ereignisgesteuerten Prozessketten

Eric Brabänder,
Heike Ochs

IDS Scheer AG
Postfach 101534, D – 66015 Saarbrücken
E-Mail: e.brabaender@ids-scheer.de
E-Mail: h.ochs@ids-scheer.de

Abstract: Es werden die Möglichkeiten des Einsatzes Ereignisgesteuerter Prozessketten für die Analyse und Gestaltung eines prozessbasierten Risikomanagementsystems aufgezeigt. Dabei wird ausgehend von der Vorgehensweise zum Aufbau eines prozessorientierten Risikomanagementsystems dargelegt, welche Methodenerweiterungen der eEPK notwendig sind und welche Auswertungsmöglichkeiten für die Berechnung des Schadenspotenzials eingesetzt werden können.

1 Notwendigkeit der prozessorientierten Risikobetrachtung

Immer stärker kann man heute erkennen, dass sich nach vielen Jahren des Wachstums und verstärkter Akquisitionstätigkeiten die Unternehmen in den wichtigen globalen Märkten restrukturieren müssen. Viele Faktoren, in Deutschland nicht zuletzt die Wiedervereinigung, die übertriebene Börsenhausse sowie der Internet-Hype, haben die Unternehmen dazu verleitet, Strukturen und Prozesse nicht immer kritisch zu hinterfragen, sondern ohne Rücksicht auf vorhandene Ressourcen und bestehende Organisationsformen zu wachsen.

Das Abnehmerverhalten verändert sich ständig, die Produktlebenszyklen verkürzen sich und der Kostendruck wird für die Unternehmen enorm. Daher sind gerade in Zeiten der rezessiven Wirtschaftsentwicklung optimierte Abläufe entscheidend. Durch diese Situation richtet sich der Fokus zahlreicher Unternehmen auf die internen betrieblichen Abläufe und deren Schnittstellen zu Kunden, Lieferanten und Kooperationspartnern.

Die häufige vorhandene Intransparenz der Unternehmensprozesse mit den Folgen von Redundanz, Ineffizienz und fehlender Integration betriebswirtschaftlicher Anwendungssysteme soll dann im Rahmen zahlreicher Projekte verbessert werden. Zeit-, qualitäts- und kostenoptimierende Prozessverbesserungen werden in Angriff genommen. Oftmals wird im Rahmen solcher Projekte aber die Betrachtung der möglichen Prozessrisiken stark vernachlässigt. Die Unternehmensprozesse sind aber der Motor eines Unternehmens. Ein großer Teil unternehmerischer Risiken entstehen im operativen Geschäft. Die-

se **operationellen Risiken**[1] resultieren überwiegend aus Geschäftsabläufen, Organisationsstrukturen, DV-Systemen oder auch externen Einflüssen [vgl. OV01]. Um operationelle Risiken zu erkennen und effiziente Handlungsstrategien ableiten zu können, muss jedes Unternehmen ein Risikomanagementsystem aufbauen, das die operationellen Risiken und deren Auslöser innerhalb der Prozesse sichtbar macht. Treten in den Prozessen Risiken auf und sind dann die Notfall- und Kontrollmaßnahmen nicht bekannt, kann das für ein Unternehmen von verheerender Wirkung sein.

Beispiele für das Nichterkennen prozessbasierter Risiken und fehlender Kontrollmechanismen finden sich zahlreiche. So ist unter anderem der Zusammenbruch der Barings Bank im Jahre 1995 auf das Zusammenwirken zahlreicher Ursachen zurückzuführen. Eine dieser Ursachen war ein mangelndes prozessorientiertes Risikomanagement. Nur durch die Intransparenz des Abwicklungsprozesses von Wertpapiertransaktionen und fehlende Kontrollstrukturen war es möglich, dass Nick Leeson die durch Marktrisiken entstandenen Derivatverluste von ca. 827 Mio. £ über das speziell dafür eingerichtete Konto 88888 lange Zeit kaschieren konnte [vgl. OV02]. Dieser Fehler im Prozessdesign und das Ausnutzen dieser Lücke durch Nick Leeson führte erst zu dem enormen Ausmaß an Verlusten der Barings Bank. Wären der Prozess und seine Schwächen transparent gewesen und das potenzielle Risiko erkannt worden, dann hätte durch eine Veränderung des Prozessdesigns und die entsprechenden Kontrollmaßnahmen bereits im Voraus das Ausmaß der Verluste reduziert werden können [vgl. He02].

Ungesicherte Prozesse führten auch im Jahre 1997 bei Morgan Grenfell Asset Management zu Verlusten von mehr als 600 Mio. USD aufgrund von Verletzungen der Anlagevorschriften für Aktienfonds durch Peter Young [vgl. BK00, S.634].

Immer wieder wird klar, dass operationelle Risiken, die in der Regel aus der Geschäftstätigkeit und den Geschäftsprozessen eines Unternehmens resultieren, zu den am häufigsten vorzufindenden Risiken gehören. Sie können noch vor dem ersten Geschäftsabschluss, Produktionsstart oder der ersten Kreditvergabe durch eine Bank auftreten [vgl. GP01, S.789].

Im Gegensatz zu den vielfach bekannten Markt- und Kreditrisiken, die bewusst eingegangen werden, um die daraus resultierenden Gewinnchancen zu nutzen, steht beim Umgang mit prozessbasierten Risiken die Minimierung des Schadenspotenzials im Vordergrund, da diesen Risiken in der Regel keine direkten Chancen gegenüberstehen [vgl. BK00, S. 650]. Obwohl operationelle Risiken sehr wohl seit jeher zu den Kernrisiken eines jeden Betriebes gehören, erfolgt ihre Bewusstmachung und ihre Instrumentalisierung jedoch tatsächlich erst seit kurzer Zeit.

Vielfach haben diese Prozessrisiken nicht das oben beschriebene katastrophale Ausmaß an finanziellen Schäden sondern schlagen sich aufgrund vieler kleiner oft nicht erkannter oder berichteter Einzelschäden in Form erhöhter Prozesskosten nieder [vgl. OV00]. Aber auch hier können durch frühzeitiges Erkennen, Analysieren und durch das Ergreifen entsprechender Maßnahmen Verluste für das Unternehmen begrenzt werden. Erkennbar

[1] vgl. zum Begriff „operationelle Risiken“ [OV98], [OV98a], [OV01], [OV01b], [OV01c], [OV02a], [OV02b]

werden operationelle Risiken erst durch die hinreichende Kenntnis des zugrundeliegenden Prozesses. Deshalb ist eine methodisch sinnvolle und durchgängige Prozessanalyse unabdingbare Voraussetzung für ein funktionierendes prozessorientiertes Risikomanagement [OV01a, S. 64]. Erst in der Detailbetrachtung einzelner Prozessbestandteile können die dort auftretenden potenziellen Risiken erkannt und mit entsprechender Genauigkeit erfaßt und bewertet werden. Hierbei sehen sich große wie kleine Unternehmen vor einer Fülle scheinbar unlösbarer Aufgaben, wenn es um die organisierte bzw. instrumentalisierte Erfassung, Berechnung und Optimierung dieser Risiken geht.

Wie sieht also eine sinnvolle Vorgehensweise bei der prozessbasierten Erhebung, Analyse und Steuerung operationeller Risiken aus? Wie sollte ein prozessorientiertes Risikomanagementsystem gestaltet sein? Welche Methodik wird dabei eingesetzt, um einerseits den Anforderungen an ein effizientes Prozessmanagement und andererseits einem integrierten Risikomanagement gerecht zu werden? Wie können bestehende Methoden um das Thema Risikomanagement erweitert werden und welche Ansätze sind praktikabel?

Im folgenden Beitrag wird aufgezeigt, wie die Methode der Ereignisgesteuerten Prozesskette erweitert werden kann, um Geschäftsprozesse hinsichtlich Ihrer Risiken zu analysieren, deren Risikopotenzial zu bewerten und ein prozessorientiertes Risikomanagementsysteme zu gestalten. Dabei wird ausgehend von der Vorgehensweise beim Aufbau eines prozessorientierten Risikomanagementsystems anhand eines praktischen Beispiels dargestellt, welche Methodenerweiterungen und Auswertungsmöglichkeiten eingesetzt werden können.

2 Vorgehensweise zur Umsetzung eines prozessorientierten Risikomanagementsystems

Die **Implementierung eines prozessorientierten Risikomanagements** umfasst in der Regel mehrere Phasen:
- Festlegung der Risikopolitik,
- Prozessanalyse,
- Risikoidentifikation und Risikoanalyse,
- Soll-Konzeption,
- Risikoreporting und
- Risikocontrolling.

Diese Phasen werden jedoch nicht einmalig durchlaufen, sondern in Form eines Kreislaufs wird das Risikomanagementsystem immer wieder an veränderte Unternehmensprozesse, Umweltbedingungen oder eine eventuelle Neuausrichtung der Unternehmenspolitik angepaßt. Dadurch können neue, bisher nicht erkannte Risiken für die Unternehmensleitung und die betroffenen Mitarbeiter sichtbar und kalkulierbar gemacht werden. Das Problem der „Betriebsblindheit“ besonders im eigenen Unternehmen kommt in der Realität insbesondere bei den Unternehmensprozessen relativ häufig vor.

2.1 Festlegen der Risikopolitik

Die Einführung und Anwendung eines Risikomanagementsystems ist eine strategische Entscheidung der Unternehmensleitung. In der Phase der Festlegung der Risikopolitik werden alle notwendigen Vorbereitungen für eine erfolgreiche Projektrealisierung geleistet. Die Schwerpunkte liegen in dem Entwickeln bzw. Aktualisieren der Risikopolitik für das prozessorientierte Risikomanagementsystem. Somit wird eine gemeinsame Basis für alle weiteren Phasen und Projektschritte geschaffen.

Bei der Formulierung der Risikopolitik und der Ziele des Risikomanagements ist zu beachten, dass diese direkt mit anderen Unternehmenszielen zusammenhängen (z.B. Technologie-, Markt-, Produkt-, Qualitätsziele, Wirtschaftlichkeit etc.). Folgende Ziele sind in der Regel in der Risikopolitik formuliert [vgl. Ro00, S. 609]:
- Sicherung der Unternehmensziele
- Sicherung des künftigen Erfolges des Unternehmens
- Senkung der Risikokosten
- Grundsätze zum Umgang mit Risiken

2.2 Prozessanalyse

Um ein Bild von der Komplexität der Problemstellung des Managements operationeller Risiken und den Erweiterungsmöglichkeiten der Methode der Ereignisgesteuerten Prozesskette zu vermitteln, wird im Folgenden anhand eines Beispiels aus der Bankenbranche der gesamte Verlauf der Prozess- und Risikoanalyse sowie der Bewertung der Risiken durchgehend für einen Wertpapierhandelsprozess aufgezeigt. Gleichermaßen gelten die Ergebnisse und Schlussfolgerungen aber für viele andere Branchen, da es sich nicht primär um die Risiken innerhalb der hergestellten Produkte und Services handelt (das wären im Beispiel einer Bank Kredit- und Marktrisiken) sondern um diejenigen Risiken, die bei der Erstellung und dem Vertrieb von Produkten entstehen. Betrachtet werden also die internen Strukturen eines Unternehmens, wodurch es nachrangig wird, welche Services und Produkte erzeugt werden.

Der Wertpapierhandelsprozess kann in fünf Teilprozessen „Verkauf“, „Disposition“, „Handel“, „Fakturierung“ und „Controlling“ beschrieben werden. Diese Teilprozesse werden im Rahmen der ARIS Methode[2] in Form eines Wertschöpfungskettendiagramms dargestellt (vgl. Abb. 1). Allerdings reicht diese recht grobe Sicht auf den Gesamtprozess nicht aus, um alle Risiken der einzelnen Teilprozesse zu erkennen. Deshalb werden die Teilprozesse mit Hilfe von Ereignisgesteuerten Prozessketten weiter detailliert, um einerseits den Prozessverlauf genauer zu analysieren und andererseits alle am Prozess beteiligten und somit auch einflussnehmenden Faktoren und potenziellen Risikoträger oder -verursacher zu erkennen.

[2] zur ARIS Methode und deren praktischem Einsatz vgl. [Sc99] und [SJ02]

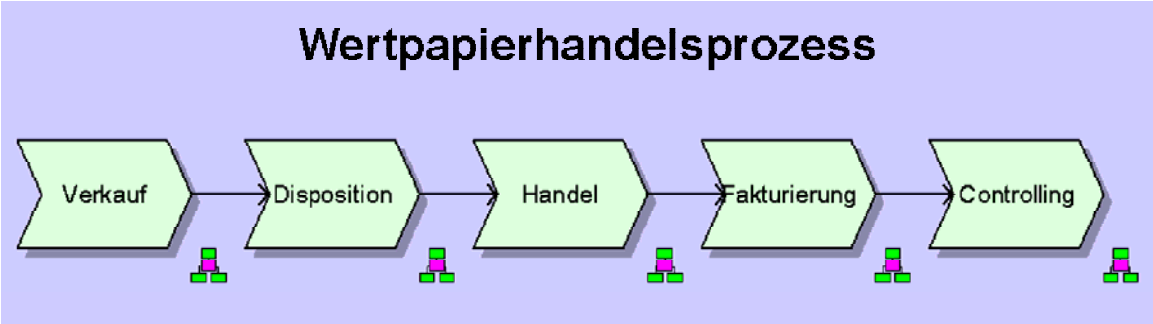


Abb.1: Teilprozesse des Wertpapierhandelsprozess als Wertschöpfungskettendiagramm

Bei einer sehr großen Zahl von Teilprozessen können im Rahmen einer Grobrisikoanalyse zuerst die Prozesse identifiziert werden, die voraussichtlich hohes Risikopotenzial enthalten. In der Regel sind das wichtige wertschöpfende Prozesse [vgl. An01, S.443] bzw. Prozesse, die hohe materielle oder immaterielle Werte verarbeiten (z.B. Handelsprozesse, Entwicklungs- und Produktionsprozesse etc.).

Im vorliegenden Beispiel können die Teilprozesse „Disposition“, „Handel“ und „Fakturierung“ als Prozesse mit hohem Risikopotenzial (A-Risikoprozesse) eingestuft werden, da klar erkennbar ist, dass über diese Prozesse der eigentliche Wertfluss verläuft. Hier ist eine Detailanalyse unverzichtbar, um zu erkennen, wie der Prozess verläuft, welche Systeme und Personen oder Rollen beteiligt sind und welche Leistungen erbracht werden sollen. Die Teilprozesse „Verkauf“ und „Controlling“ weisen ein verhältnismäßig geringeres Risikopotenzial auf (B-Risikoprozesse).

Der Teilprozess „Handel“ wird für das vorliegende Beispiel inhaltlich vereinfacht. Anhand einer erweiterten Ereignisgesteuerten Prozesskette (eEPK) wird eine Detailanalyse dieses Prozesses durchgeführt (vgl. Abb. 2). Im Rahmen dieser Analyse wird erkannt, dass der Prozess mehrere Verzweigungen aufweist. Der Teilprozess „Handel“ hat das Startereignis „Order ist freigegeben“. Sobald dieses Ereignis eintritt, folgt die Funktion „Order auf Vollständigkeit prüfen“, welche von einem Mitarbeiter durchgeführt wird, der die Rolle „Wertpapierhändler“ hat. Für diesen Prozessschritt wird in dem Beispiel das Ordersystem verwendet. Ist die Order nicht vollständig erfasst und übermittelt worden, wird zurück zum Teilprozess „Disposition“ verzweigt. War die Order vollständig, werden im folgenden Schritt die Wertpapierbeschränkungen geprüft. Auch hier kann der Prozess wiederum zum Teilprozess „Disposition“ verzweigen, falls Beschränkungen bestehen.

Sind keine Beschränkungen vorhanden, folgen die typischen Handelsaktivitäten, wie „Handelsplatz und Makler bestimmen“, „Preis feststellen“, „Order bestätigen“ und letztendlich die „bestätigte Order zurückmelden“. Der Teilprozess „Handel“ endet mit dem Ereignis „bestätigte Order zurückgemeldet“, welches wiederum als Startereignis für den nachfolgenden Teilprozess der „Fakturierung“ fungiert.

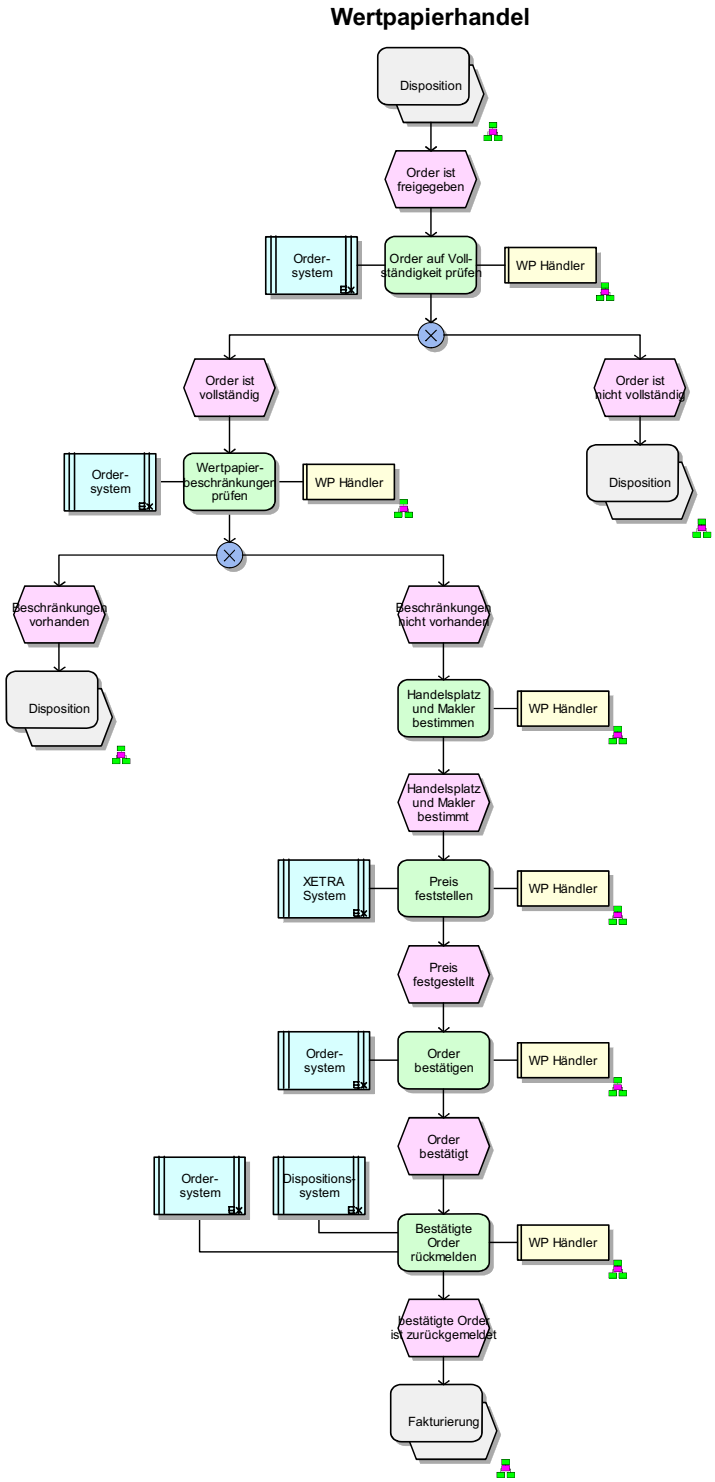


Abb. 2: Teilprozess „Wertpapierhandel“ als erweiterte Ereignisgesteuerte Prozesskette

2.3 Risikoidentifikation und Risikoanalyse

Bei der Risikoidentifikation sollen zum einen die Mitarbeiter bzw. die Prozessverantwortlichen als Experten ihrer täglichen Arbeit befragt werden als auch die interne Revision, die Mitglieder der Organisationsabteilung und die Mitglieder des Risikocontrol-

lings. Diese Abteilungen können ihre Erfahrungen (z.B. Prüfungs- und Revisionsberichte) im Umgang mit strukturbedingten Risiken in die Analyse einbringen. Mit ihrer eigenen Sichtweise können sie dabei helfen, von den Prozessverantwortlichen nicht beachtete Aspekte zu durchleuchten und subjektive Ansichten zu objektivieren.

Im Rahmen von **Risikoaudits** wird gemeinsam mit den involvierten Mitarbeitern der dokumentierte Prozess Schritt für Schritt besprochen und nach möglichen Risiken an den einzelnen Funktionen durchleuchtet. Die Prozesse sollten vor allem immer auf folgende **Schwachstellen** hin überprüft werden [We01, S.38]:
- Mehrfachbearbeitungen und Rückschleifen,
- manuelle / papiergestützte Bearbeitung,
- manuelle Datenerfassung anstatt automatischer Datenerfassung,
- Medienbrüche (Papier / Software),
- mangelnde Qualität der Systemunterstützung,
- redundante Datenerfassung in verschiedenen Systemen,
- fehlende Schnittstellen zwischen den Systemen,
- heterogene und veraltete Infrastruktur.

Auch bereits bestehende Kontrollen und Gegenmaßnahmen zur Minderung der Risiken sowie bekannte Mängel dieser Kontrollmechanismen müssen erfasst werden. Für jedes identifizierte Risiko werden folgende **zusätzliche Informationen** erhoben:
- Risikoverantwortlicher (Risk Owner), der das Risiko regelmäßig überwacht und zurückmeldet,
- weitere betroffene Prozesse und Wechselwirkungen,
- Datum der letzten Bewertung des Risikos,
- mögliche Frühwarnsignale,
- Kennzahlen zur Risikoüberwachung mit Eingriffsschwellen,
- mögliche Maßnahmen zur Risikooptimierung,
- Kontrollprozesse zur regelmäßigen Überwachung des Risikos.

Anhand dieser notwendigen Informationen zur detaillierten Analyse und Bewertung der einzelnen Prozessrisiken können die benötigten Erweiterungen der Ereignisgesteuerten Prozesskette für das Risikomanagement hergeleitet werden. So wird im Modell der eEPK ein Objekttyp für die Darstellung der einzelnen Risiken an den jeweiligen Funktionen benötigt. Wie diese Erweiterung um den Objekttyp „Risiko“ aussieht wird in Abbildung 3 dargestellt.

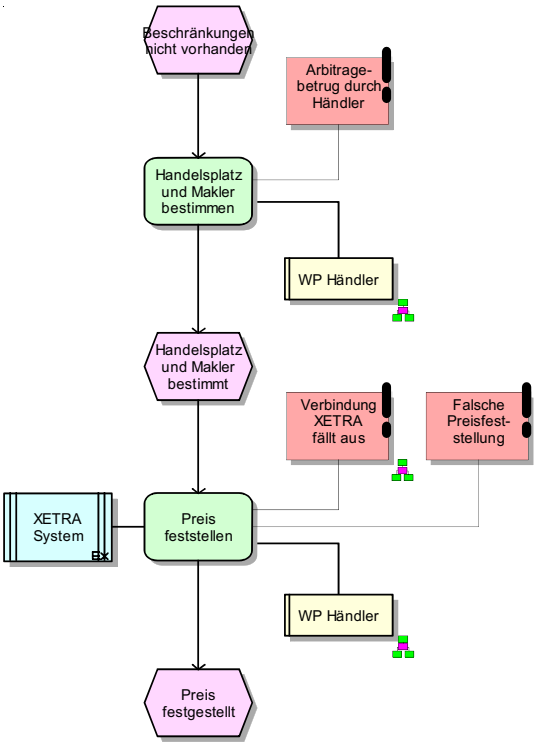


Abb. 3: Risikoidentifikation im Teilprozess „Wertpapierhandel“

So kann beispielsweise im Teilprozess „Handel“ an der Funktion „Handelsplatz und Makler bestimmen“ das Risiko des „Arbitragebetrug durch den Händler“ identifiziert werden sowie an der Funktion „Preis feststellen“ die Risiken „Verbindung XETRA fällt aus“ und „Falsche Preisfeststellung“ (vgl. Abb. 3).

Allerdings reicht die reine Identifizierung der Risiken innerhalb des Prozessmodells nicht aus, um alle zur Risikoanalyse und -bewertung notwendigen Informationen darzustellen. Aus diesem Grund wird jedes Risiko in ARIS in einem weiteren Detailmodell vom Typ „Kennzahlenzuordnungsdiagramm“ näher beschrieben.

Das Risiko „Verbindung XETRA fällt aus“ wird in diesem Detailmodell anhand der oben genannten Kriterien verfeinert (vgl. Abb. 4). So wird als Risikoverantwortlicher der „IT-Systemleiter“ für das XETRA System identifiziert. Kennzahlen zur Überwachung und Analyse des Risikos liefern die „Ausfallrate des XETRA-Systems“ sowie die „Wiederherstellungszeit“, die benötigt wird, bis das System nach einem Ausfall wieder verfügbar ist. Mögliche Maßnahmen zur Risikooptimierung wären in diesem Beispiel der Aufbau eines Zwischenspeicherungssystems oder der Aufbau von Redundanzleitungen zum XETRA System.

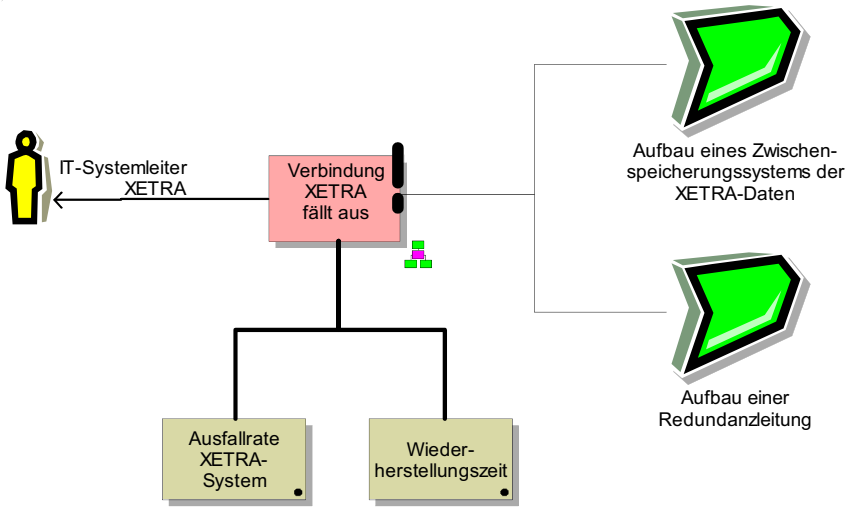


Abb. 4: Risikodetailanalyse mit ARIS

Zur Risikoanalyse gehört neben der Risikoidentifikation sowie der modellbasierten Beschreibung von Risiken und risikorelevanter Informationen auch die **Bewertung der Risiken** hinsichtlich ihres Schadenspotenzials. Erst diese Bewertung liefert wertvolle Daten für die weiteren Phasen, insbesondere für die Soll-Konzeption und das Risikoreporting. Ohne eine Bewertung des Risikos können für mögliche Maßnahmen keine sinnvollen Kosten-Nutzen-Analysen durchgeführt werden, da in der Regel die Kosten der risikooptimierenden Maßnahmen der Reduzierung (bzw. Optimierung) des Schadenspotenzials des Risikos gegenübergestellt werden.

Die einzelnen Risiken werden dabei in der Regel aufgrund der Analyse historischer Daten oder durch Expertenschätzungen nach ihrer Eintrittshäufigkeit bezogen auf einen bestimmten Zeitraum (in der Regel pro Jahr) und ihrem Schadensausmaß (im Sinne einer Verlustgröße) bewertet [vgl. JP01, S.54].

Input für die Schadensanalyse eines Risikos können alle Dokumente liefern, die mögliche Verlustdaten enthalten können wie beispielsweise Revisions- und Wirtschaftsprüfungsberichte, Dokumentationen der juristischen Abteilung (z.B. Betrugsfälle, Kundenklagen), der EDV-Abteilung (z.B. Protokolle über Systemausfälle), der Produktion (z.B. Wartungsanalysen einzelner Maschinen), der Controlling-Abteilung, der Personal-Abteilung sowie anderer Fachabteilungen.

Für viele prozessbasierte Risiken liegen allerdings bei zahlreichen Unternehmen noch keine Datensammlungen vor. In diesem Fall kann es durchaus sinnvoll sein, Datensammlungen externer Initiativen oder kommerzieller Anbieter zu erwerben, um einzelne Risiken bewerten zu können. Gerade große Verlustfälle treten nur sehr selten auf und können daher oft nicht anhand interner Datenanalysen erkannt bzw. abgeschätzt werden. Bei der Verwendung externer Daten sollte aber immer beachtet werden, aus welchen Branchen und Unternehmensbereichen die angebotenen Datensammlungen stammen und wie umfangreich bzw. vollständig die Daten sind. Problematisch sind externe Datensammlungen dann, wenn Kriterien unbekannt bleiben, die für die Zuführung zur eigenen Datensammlung nötig wären.

Die eigentliche Berechnung des zu erwartenden Schadenspotenzials kann anhand unterschiedlicher Verfahren durchgeführt werden. Drei Berechnungsverfahren stehen derzeit zur Diskussion. Zwei dieser Verfahren sind deterministisch, eines verwendet statistische Ansätze. Ebenfalls basieren zwei dieser Verfahren auf den Prozessverläufen, während eines stark vereinfacht ist und den Prozessverlauf nicht berücksichtigt:

1. **Die allgemeine Risikoberechnung** (deterministisch, nicht prozessorientiert)
2. **Die prozessorientierte Risikoberechnung** (deterministisch, prozessorientiert)
3. **Die simulationsbasierte Risikoberechnung** (statistisches Verfahren, prozessorientiert)

2.3.1 Die allgemeine Risikoberechnung

Die allgemeine Risikoberechnung geht von den zwei Kriterien „Verlusthöhe des Risikos i“ (VHR_i) und „Eintrittshäufigkeit des Risikos i pro Zeiteinheit“ (EHR_i) aus. Sind alle Schadensfälle eines Unternehmens für ein ganz bestimmtes Risiko vollständig bekannt und über einen längeren Zeitraum hin erfasst, lässt sich aus den gesammelten Daten für jedes der beiden Kriterien ein sinnvoller Durchschnittswert herleiten. Aus der Multiplikation beider Kriterien ergibt sich das pro Zeiteinheit zu erwartende „Schadenspotenzial eines bestimmten Risikos i“ (SPR_i).

$$SPR_i = VHR_i \bullet EHR_i$$

Diese Form der Risikoberechnung liefert nur sehr grobe Ergebnisse, ist aber in der Praxis aufgrund der einfachen und schnellen Durchführbarkeit häufig anzutreffen. Darin liegt auch der Vorteil der allgemeinen Risikoberechnung. Man erhält zu Beginn einer Risikoanalyse sehr schnell einen Überblick über die Risikosituation eines Unternehmens.

Die Nachteile der allgemeinen Risikoberechnung liegen vor allem in der Durchschnittsbildung der Eingangskriterien. Vielfach gehen gerade aufgrund der Durchschnittsbildung wertvolle Informationen verloren. So werden Extremwerte geglättet und es geht die Information verloren, wie hoch die Wahrscheinlichkeit des Auftretens gerade solcher Extremwerte ist. Diese Schwachstelle führt wiederum bei vielen Einzelrisiken zu notwendigen Detailanalysen. Durch die Bildung der Durchschnittswerte ist auch keine sinnvolle „Aggregation“ der errechneten Schadenspotenziale im Sinne eine Value-at-Risk-Analyse möglich.

Ein weiterer Nachteil der allgemeinen Berechnungsmethode ist eindeutig in der Unabhängigkeit der Risiken von den betrachteten Prozessen zu sehen. So sind die Risiken zwar einzelnen Funktionen innerhalb eines Prozesses zugeordnet, das Eintreten eines Risikos ist aber im Rahmen dieser Analyse letztendlich unabhängig von der Häufigkeit der Ausführung der Funktionen. Diese Information kann lediglich implizit durch das Kriterium „Eintrittshäufigkeit des Risikos pro Zeiteinheit“ in die Berechnung eingehen, in-

dem bei der vorherigen Analyse dieser Eintrittshäufigkeiten auch die mit dem Risiko verbundenen Prozesse und deren Ausführungshäufigkeiten untersucht werden.

Aufgrund der Unabhängigkeit der Berechnungsmethode von den Prozessinformationen kann bei der allgemeinen Berechnungsmethode auch nicht auf das Schadenspotenzial eines gesamten Prozesses geschlossen werden.

2.3.2 Die prozessorientierte Risikoberechnung

Die prozessorientierte Risikoberechnung schließt die Lücke zu den fehlenden Prozessinformationen bei der Berechnung des Schadenspotenzials für ein Risiko i (SPR_i). Anstatt des risikobezogenen Kriteriums der „Eintrittshäufigkeit des Risikos i" werden an der Funktion j die prozessorientierten Kriterien „Häufigkeit der Funktionsausführung" (HF_j) pro Zeiteinheit einer Funktion j und die „relative Eintrittswahrscheinlichkeit des Risikos i an Funktion j" (REW_{ij}) berücksichtigt.

Daraus kann wiederum, bei Kenntnis der Verlusthöhe pro Risiko i (VHR_i), dessen Schadenspotenzial wie folgt berechnet werden:

$$SPR_i = VHR_i \bullet \sum_{j=1}^{n} (HF_j \bullet REW_{ij})$$

Dabei kann die Verlusthöhe (wie bereits bei der allgemeinen Risikoberechnung) auf Basis gesammelter Verlustfälle errechnet werden. Die Kriterien „Häufigkeit der Funktionsausführung pro Zeiteinheit" und „relative Eintrittswahrscheinlichkeit" können entweder durch Interviews der Prozessverantwortlichen oder durch die Messung mit geeigneten Analysewerkzeugen (z.B. ARIS Process Performance Manager[3]) erhoben werden.

Die prozessorientierte Risikoberechnung ermöglicht nicht nur die Berechnung des zu erwartenden Schadenspotenzials eines bestimmten Risikos i pro Zeiteinheit sondern auch die Kalkulation des Schadenspotenzials der Risiken einer bestimmten Funktion j. So errechnet sich das „Schadenspotenzial der Funktion" j (SPF_j) ebenfalls anhand der Kriterien der „Häufigkeit der Funktionsausführung" (HF_j) pro Zeiteinheit, der „relativen Eintrittswahrscheinlichkeit" (REW_{ij}) des Risikos i pro Funktion j sowie der „Verlusthöhe des Risikos i" (VHR_i):

$$SPF_j = HF_j \bullet \sum_{i=1}^{n} (VHR_i \bullet REW_{ij})$$

Aufgrund der Berücksichtigung der Häufigkeit der einzelnen Funktionsausführungen bei der Berechnung des zu erwartenden Schadenspotenzials kann bei einfachen Prozessstrukturen ein Schadenspotenzial für den gesamten Prozess abgeschätzt werden.

[3] vgl. zu ARIS PPM: http://www.ids-scheer.de/ppm

Diese Form der Risikoberechnung liefert Ergebnisse mit direktem Bezug zum zugrundeliegenden Prozess und dessen Struktur. Die Ergebnisse sind deshalb bereits wesentlich aussagekräftiger als die Ergebnisse der allgemeinen Risikoanalyse und liefern bereits deutliche Hinweise auf besonders kritische Prozesse im Unternehmen. Aufgrund der detaillierteren Analyse können bei diesem Berechnungsverfahren die später abzuleitenden Maßnahmen hinsichtlich Ihrer Kosten-Nutzen-Struktur genauer bewertet werden.

Die Nachteile des Verfahrens liegen einerseits in der deterministischen Berechnung des zu erwartenden Schadenspotenzials und der zugrundeliegenden Durchschnittswerte (z.B. durchschnittliche Häufigkeit der Funktionsausführung, durchschnittlicher Schadenswert etc.), andererseits in der nicht zu unterschätzenden Fülle an notwendigem Datenmaterial zur Durchführung der Berechnung. Der Aufwand der Erhebung dieser prozessrelevanten Daten macht den Einsatz des Verfahrens in erster Linie bei besonders kritischen Prozessen rentabel.

2.3.3 Die simulationsbasierte Risikoberechnung

Die simulationsbasierte Risikoberechnung kann als Erweiterung der prozessorientierten Risikoberechnung gesehen werden. Sie verwendet die gleichen Berechnungsformeln wie die prozessorientierte Risikoberechnung allerdings in Kombination mit dem Verfahren der Prozess- und Monte Carlo-Simulation[4].

So wird ein Prozess über einen bestimmten Zeitraum (z.B. ein Jahr) hinweg simuliert, indem die EPK basierend auf den Zeit- und Ressourcenvorgaben instanziiert und durchlaufen wird. Um die Prozesssimulation möglichst realitätsnah zu gestalten, können an jeder Funktion hinsichtlich ihrer Liege-, Einarbeitungs- und Bearbeitungszeiten verschiedene Verteilungsfunktionen vorgegeben werden.

Bei jedem Durchlauf des Prozesses wird an jeder ausgeführten Funktion überprüft, ob eines oder mehrere Risiken an dieser Funktion auftreten können. Basierend auf der relativen Eintrittswahrscheinlichkeit jedes Risikos an der Funktion wird im Simulationsdurchlauf der Risikoeintritt ausgelöst oder nicht ausgelöst. Wurde ein Risiko während eines Prozessdurchlaufs ausgelöst, kann nun auf Basis der Monte Carlo Simulation ein Schadenswert an dem Risiko ermittelt werden. Dabei wird der Schadenswert des Risikos in dem Modell nicht mehr als Durchschnittswert erfasst, sondern als Verteilungsfunktion (z.B. Konstant, Gleich-, Normal-, Exponentialverteilung etc.), aus welcher bei jedem Risikoeintritt ein entsprechender Schadenswert generiert wird.

Durch die Aggregation der eingetretenen Risiken pro Prozessdurchlauf lässt sich eine prozessorientierte Verlusthöhenverteilung für den gesamten Prozess im Sinne einer Value-at-Risk Analyse erstellen. Aber auch das zu erwartende Schadenspotenzial einzelner Funktionen kann entsprechend der statistischen Verteilung genau analysiert werden.

[4] vgl. zur Methodik der Monte Carlo Simulation [FN01]

Dieses Verfahren liefert den größten Detaillierungsgrad einer Risikoanalyse mit Ereignisgesteuerten Prozessketten. Dafür benötigt man zur Durchführung der entsprechenden Simulation auch sehr detaillierte Prozess- und Risikoinformationen. Aus diesem Grund ist diese Vorgehensweise bisher im praktischen Einsatz in Unternehmen noch nicht anzutreffen. Für besonders kritische Prozesse bietet die simulationsbasierte Risikoberechnung allerdings größte Aussagekraft.

2.4 Soll-Konzeption

Das Konzept des prozessorientierten Risikomanagementsystems mit Ereignisgesteuerten Prozessketten ermöglicht ausgehend von einer gemeinsamen Datenbasis das Erreichen zweier Ziele. Einerseits wird das Hauptziel - das Management operationeller Risiken und die Absicherung von Prozessen - umgesetzt, darüber hinaus werden gleichzeitig die untersuchten Unternehmensprozesse optimiert und somit effizienter gestaltet.

Die Analyse für jedes Risiko nach Schadenswert und Eintrittshäufigkeit (vgl. Abb. 5) sowie (je nach verwendeter Berechnungsmethode) die Analyse der einzelnen Prozesse und Funktionen hinsichtlich des zu erwartenden Schadenspotenzials bilden den Ausgangspunkt der weiteren Schritte im Rahmen der Soll-Konzeption. Für die Risiken, die sich im „roten Bereich“ befinden, herrscht dringender Handlungsbedarf hinsichtlich der Maßnahmen zur Reduktion des Schadenswertes und/oder der Eintrittshäufigkeit. Risiken, die sich im „gelben Bereich“ befinden werden individuell betrachtet und ebenfalls entsprechende Maßnahmen überprüft. Risiken im „grünen Bereich“ gilt es, zu beobachten und bei negativer Veränderung Maßnahmen zu ergreifen bzw. nur die Maßnahmen zu ergreifen, die ein solches Risiko ohne große Aufwände drastisch reduzieren oder gar eliminieren würden.

Je nachdem in welchem Bereich des **Analysecharts** sich ein Risiko befindet, können die vielfach bekannten Handlungsstrategien Versichern/Überwälzen, Vermeiden/Verhindern, Kontrollieren/Reduzieren oder Akzeptieren/Selbsttragen als Vorgabe für die zu treffenden Maßnahmen dienen [vgl. BK00, S.651].

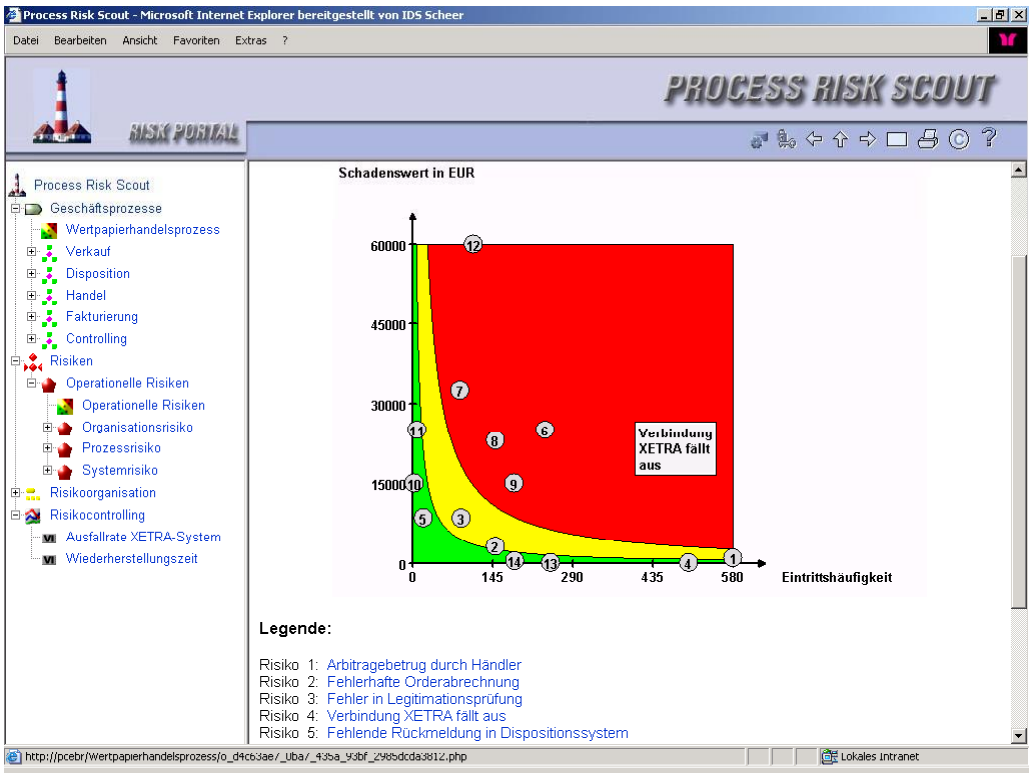


Abb. 5: Analysechart für die Schadenspotenzialanalyse operationeller Risiken

Dabei ist bei jeder Maßnahme zur Optimierung eines Risikos im Rahmen einer Kosten-Nutzen-Analyse abzuwägen, ob die Minderung der Schadenshöhe und/oder der Eintrittshäufigkeit die mit der Maßnahme verbundenen Kosten rechtfertigen.
Folgende **Maßnahmen** werden häufig bei der Umsetzung des prozessorientierten Risikomanagements eingesetzt:

- Risikooptimiertes Redesign des Prozesses
- Implementierung von Kontrollmechanismen und Qualitätspunkten im Prozessverlauf
- Standardisierung von Arbeitsgängen,
- Einführung einer einheitlichen Bearbeitungssoftware,
- Einführung eines Workflow-Systems,
- Überprüfung bestehender Berechtigungskonzepte, oder
- Aufbau von Redundanz- und Backupsystemen.

Im Beispiel des Risikos „Verbindung XETRA fällt aus“ könnten zwei Maßnahmen umgesetzt werden. Einerseits wäre der Aufbau eines Zwischenspeicherungssystems der XETRA-Daten sinnvoll. So könnten alle 30 Minuten automatisch die aktuellen Kurswerte in einer Datenbank zwischengespeichert werden, um im Falle eines Ausfalls auf diese Kurswerte zugreifen zu können. Andererseits kann durch den Aufbau einer zweiten, völlig unabhängigen Datenleitung, der Ausfall aufgrund von Leitungsüberlastungen verhindert werden. Die Analyse des Risikos zeigt jedoch, dass dieses Risiko im grünen Bereich liegt (vgl. Risiko Nr. 4 in Abb. 5). Eine Gegenüberstellung der Kosten, die mit den einzelnen Maßnahmen verbunden wären, ergibt, dass keine der Maßnahmen zu einem sinnvollen Kosten-Nutzen-Verhältnis führen würde.

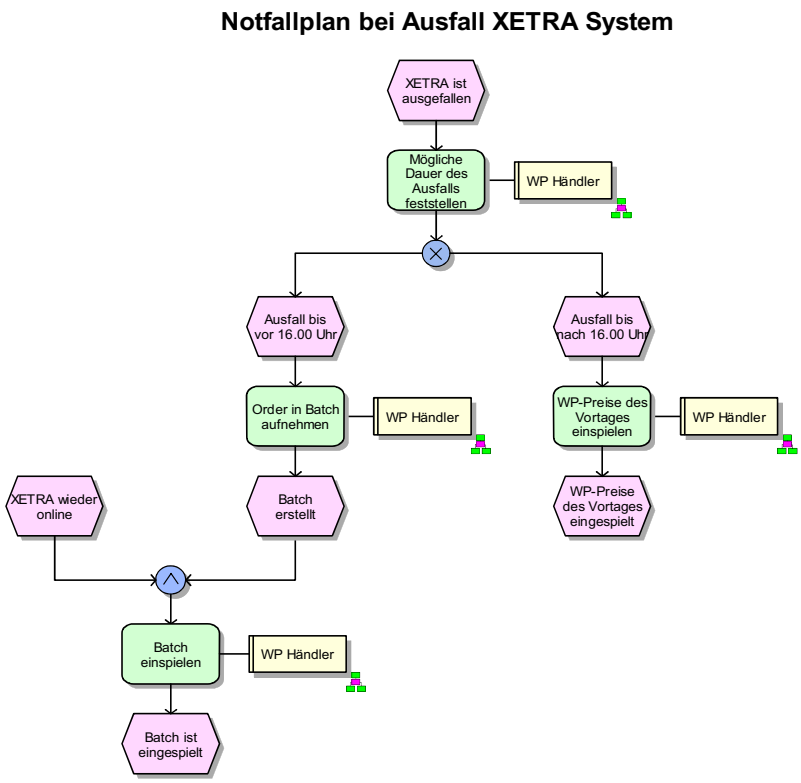


Abb. 6: Notfallbeschreibung für das operationelle Risiko „Ausfall des XETRA Systems“

Können die Prozesse durch entsprechende Maßnahmen oder Umstrukturierungen nicht vollständig gesichert werden oder stehen die Maßnahmen (wie in dem Beispiel) in keinem positiven Kosten-Nutzen-Verhältnis, so sollten für die **Risiken Notfall- und Alternativprozesse, Kontrollprozesse und Risikokennzahlen** zur Überwachung definiert werden [vgl. OV01a, S.65]. Im Beispiel des Wertpapierhandelsprozess muss der beteiligte Wertpapierhändler (WP-Händler) darüber informiert sein, was er im Falle eines Verbindungsausfalls zum XETRA System zu tun hat. Um den operativen Prozess also im Risikofall nicht zum vollständigen Stillstand kommen zu lassen, wird dem Risiko ein entsprechender Notfallprozess zugeordnet (vgl. Abb. 6).

Zur Beschreibung der Notfall-, Alternativ- oder Kontrollprozesse wird in der Regel wiederum die EPK eingesetzt. Gerade bei der Beschreibung der Alternativprozesse kann durch die Variation der Beschreibung des „normalen“ Prozessverlaufs sehr schnell verdeutlicht werden, wo die Veränderungen des Prozesses im Notfall liegen. So könnte im Beispiel des Systemausfalls eines dem Prozess zugrundeliegenden IT-Systems sehr schnell im variierten Alternativprozess der Informationsfluß anhand der nun zu verwendenden Dokumente aufgezeigt werden.

2.5 Risikoreporting

Für ein effektives Risikomanagementsystem ist die zielgerichtete Kommunikation aller Risiken, Maßnahmen und vor allem der Notfall-, Alternativ- und Kontrollprozesse von erheblicher Bedeutung. Es muss sichergestellt werden, dass die in den vorherigen Phasen

erhobenen Daten den zuständigen Entscheidungsträgern und Betroffenen im Unternehmen zeitnah zur Verfügung stehen. Dies kann durch ein rollen- und nutzergerechtes Publizieren gewährleistet werden. Am besten eignet sich dafür ein webbasiertes Portalsystem. Mit dem Rollenkonzept wird sichergestellt, dass jedem Mitarbeiter auch genau die für ihn bestimmten Informationen und alle ihn betreffenden Notfall- und Alternativprozesse kommuniziert werden.

Dazu sind in einem prozessorientierten Risikomanagementsystem grundsätzlich zwei **Rollen** notwendig:

- **Risk Owner:** Sie sind für die Kontrolle, Beobachtung und Rückmeldung einzelner Risiken verantwortlich und können auch nur die Bewertung der Risiken ihres Verantwortungsbereiches einsehen. Risk Owner sind in der Regel die Prozessverantwortlichen in den Fachabteilungen. Risk Owner werden den einzelnen Risiken zugeordnet, indem in ARIS die entsprechenden Personen oder Personentypen an den Objekttyp Risiko im Kennzahlenzuordnungsdiagramm über die Kante „ist verantwortlich für“ hinzugefügt werden (vgl. Abb. 4).

- **Risk Manager**: Sie sind für die Implementierung des prozessorientierten Risikomanagementsystems verantwortlich und analysieren und verwalten die identifizierten Risiken. Sie haben Einblick in die gesamte Risikosituation des Unternehmens und sind in der Regel Mitglieder der Revisions- und Controllingabteilungen.

2.6 Risikocontrolling und Risikoüberwachung

Ein Risikomanagementsystem dient der Kontrolle und Steuerung der Risiken und der möglichst zeitnahen Warnung vor existenzbedrohenden Risikoereignissen. Durch die Risikokontrolle und die Einführung eines Frühwarnsystems mit Hilfe von Kennzahlen wird das bewusste Auseinandersetzen mit operationellen Risiken gefördert, so dass die Risiken früher erkannt werden und Steuerungsmaßnahmen ergriffen werden können [vgl. An01, S.446]. Darüber hinaus können Risiken mit Hilfe spezieller Softwareanalysesysteme automatisiert erkannt und eingetretene Schäden berechnet werden.

Neben der Identifizierung und der automatischen Berechnung operationeller Risiken müssen die Risiken auch anhand von Frühwarn- bzw. Risikoindikatoren überwacht werden können. Dazu eignen sich bestimmte quantitative und qualitative Kennzahlen, die für jeden Prozess automatisiert aus den unterstützenden Systemen erhoben werden können [vgl. Ex02, S.70-73]. Derartige Kennzahlen finden sich für alle Unternehmen unterschiedlicher Branchen. **Typische Risikoindikatoren** für die prozessorientierten Risiken im Beispiel des Wertpapierhandelsprozess sind:

- Reklamationshäufigkeiten und -bearbeitungszeiten (z.B. Anzahl der Kundenreklamationen aufgrund falscher Wertpapierbuchungen)
- Häufigkeit von Stornobuchungen oder Buchungen gegen bestimmte Sonderkonten
- Umfang von Fehlerhinweislisten und Fehlerursachen

- Häufigkeit manueller Nachbearbeitungen (Hinweis auf die Qualität der Systemunterstützung)
- Gesamtdurchlaufzeiten eines Prozesses (z.B. Gesamtlaufzeit des Wertpapierhandelsprozess)
- Verletzung von Freigabeverfahren im Prozess
- Ausfallraten und Wiederherstellungszeiten bestimmter IT-Systeme (z.B. XETRA Ausfall- und Wiederherstellungsrate)

Sobald diese Kennzahlen bestimmte Limitvorgaben über- oder unterschreiten, muss das Analysesystem automatisch die prozess- und risikoverantwortlichen Mitarbeiter des Unternehmens informieren, damit sofort entsprechende Problemanalysen und Gegenmaßnahmen eingeleitet werden können. Nur so können die operativen Geschäftsprozesse eines Unternehmens zeitnah und effizient überwacht werden und die operationellen Risiken gesteuert und kontrolliert werden.

Die Kennzahlen und Frühindikatoren zur Überwachung der Risiken werden ebenfalls im Modelltyp „Kennzahlenzuordnungsdiagramm“ mit der Kante „wird gemessen durch“ festgelegt.

3 Ergebnisse

Die Notwendigkeit der Umsetzung eines prozessorientierten Risikomanagementsystems für Unternehmen steht ausser Frage. Im vorliegenden Artikel wurde anhand eines Beispiels aus der Bankenbranche eine praxisorientierte Vorgehensweise zur Gestaltung eines solchen prozessorientierten Risikomanagementsystems für den Umgang mit operationellen Risiken anhand der Methode der Ereignisgesteuerten Prozessketten dargestellt.

Dabei wurde vor allem aufgezeigt, wie ARIS und die Ereignisgesteuerten Prozessketten die Gestaltung und Umsetzung eines prozessorientierten Risikomanagementsystems unterstützen und welche Ansätze bezüglich der Kalkulation zu erwartender Schadenspotenziale auf Basis von EPKs derzeit zur Diskussion stehen.

Trotz des Implementierungsaufwandes prozessorientierter Risikomanagementsysteme überwiegen die damit verbundenen Vorteile. Risiken, die den Geschäftsablauf empfindlich stören oder stillegen könnten, werden bereits im Voraus identifiziert, Gegenmaßnahmen können umgesetzt und kennzahlenbasierte Kontrollmechanismen installiert werden. Natürlich bleibt die Frage offen, wie mit Risiken umgegangen werden soll, die im Rahmen der prozessbasierten Analyse nicht erkannt wurden. Diese Risiken können nur sehr schwer abgesichert werden und bilden somit einen noch bestehenden Schwachpunkt (Restrisiko) im Risikomanagementsystem. Aber durch die offene Diskussion und Analyse operationeller Risiken und deren zunehmende Wahrnehmung auf betrieblicher Ebene werden im Laufe der Zeit auch diese Risiken erkannt, in der zentralen Risikodatenbank erfaßt und entsprechend überwacht.

Klar strukturierte und beschriebene Prozesse sind eine notwendige Voraussetzung zur Erkennung und Quantifizierung von operationellen Verlusten und damit Bedingung für ein erfolgreiches Management operationeller Risiken.

Eine praxisorientierte Lösung für die Umsetzung eines prozessorientierten Risikomanagementsystems ist der **Process Risk Scout** der IDS Scheer AG[5]. Der Process Risk Scout ist ein Lösungspaket, das in detaillierter Form die Vorgehensweise und den Einsatz einzelner Werkzeuge (z.B. ARIS Toolset, ARIS Process Performance Manager etc.) beschreibt. Gleichzeitig sind die für eine Umsetzung erforderliche und hilfreiche Referenzinhalte in Form von Checklisten, ARIS Referenzdatenbanken, Musterformularen und Beispielen enthalten. Das Ergebnis der Umsetzung mit Hilfe des Process Risk Scout ist ein Risikoportal, in dem jeder Anwender einen rollen- und nutzerspezifischen Zugriff auf die Risikoinformationen hat.

Literaturverzeichnis:

[An01] Anders, U.: Qualitative Anforderungen an das Management operativer Risiken, in: Die Bank, Nr. 6, 2001, S. 442 - 446.

[BK00] Beeck, H.; Kaiser, T.: Quantifizierung von Operational Risk mit Value-at-Risk, in: Johannig, L.; Rudolph, B. (Hrsg.), Handbuch Risikomanagement (Band 1): Risikomanagement für Markt-, Kredit und operative Risiken, (Uhlenbruch) Bad Soden 2000, S. 633 – 553

[Ex02] Exeler, S.: Was fehlt beim Fehler?, in: eBanker Magazin, Nr. 2, 2002, S. 70 – 73

[FN01] Frey, H.C.; Nießen, G.: „Monte Carlo Simulation – Quantitative Risikoanalyse für die Versicherungsindustrie“, (Gerling Akademie) München 2001.

[GP01] Geiger, H.; Piaz, J.-M.: Identifikation und Bewertung operationeller Risiken, in: Schierenbeck, H.; Rolfes, B.; Schüller, S. (Hrsg.), Handbuch Bankcontrolling, 2. Aufl., (Gabler) Wiesbaden 2001, S 789 – 802

[He02] Hereford, N.: Calculated Risk: How banks make sure they stay off the Barings path; URL: http://www.garp.com/public/calcrisk.doc (2002-08-12).

[JP01] Jovic, D.; Piaz, J.-M.: Operationelle Risiken sind teuer, in Schweizer Bank, Nr. 7, 2001, S. 54 - 55.

[5] vgl. für Detailinformationen vgl. http://www.ids-scheer.de/risk-scout/

[OV98] o.V.: Enhancing Bank Transparency, Basel Committee Publications No. 41, September 1998, URL: http://www.bis.org/publ/bcbs41.htm (2002-10-22)

[OV98a] o.V.: Operational Risk Management, Basel Committee Publications No. 42, September 1998, URL: http://www.bis.org/publ/bcbs42.htm (2002-10-22)

[OV00] o.V.: Time for a New Look at Operational Risk – Executive Summary; Corporate Risk Management, Meridien Research, Inc, Februar 2000.

[OV01] o.V.: Basel Committee on Banking Supervision (Hrsg.): Operational Risk – Supporting Document to the New Basel Capital Accord, Januar 2001.

[OV01a] o.V.: Auch im Notfall betriebsbereit sein, in: Computerwoche, Nr. 41, 2001, S. 64 – 65.

[OV01b] o.V.: Working Paper on the Regulatory Treatment of Operational Risk, September 2001, URL: http://www.bis.org/bcbs/publ.htm (2002-10-22)

[OV01c] o.V.: Sound Practices for the Management and Supervision of Operational Risk, Basel Committee Publications No. 86, Dezember 2001, URL: http://www.bis.org/publ/bcbs86.htm (2002-10-22)

[OV02] o.V.: Fallstudie „Barings“;
URL: http://www.erisk.com/reference/case/ref_case_barings.asp
(2002-10-22)

[OV02a] o.V.: The New Basel Capital Accord, Juli 2002,
URL: http://www.bis.org/publ/bcbsca.htm (2002-10-22)

[OV02b] o.V.: The Quantitative Impact Study for Operational Risk: Overview of Individual Loss Data and Lessons Learned, Basel Committee , Januar 2002. URL: http://www.bis.org (2002-10-22)

[Ro00] Romeike, F.: IT Risiken und Grenzen traditioneller Risikofinanzierungsprodukte; Zeitschrift für Versicherungswesen Nr.17, 1. September 2000, Jg. 51, S. 603 – 610.

[Sc99] Scheer, A.-W.: ARIS – Vom Geschäftsprozess zum Anwendungssystem; 3. Aufl., (Springer) Heidelberg 1999, S. 32 – 37.

[SJ02] Scheer, A.-W.; Jost, W. (Hrsg.): ARIS in der Praxis; 1. Aufl., (Springer) Heidelberg 2002.

[We01] Weiß, D.: Die 4 Dimensionen des Prozessmanagements, in: is report, Nr. 6, 2001, 5. Jg, S. 34 – 38.

Refactoring von Ereignisgesteuerten Prozessketten

Peter Fettke, Peter Loos

Johannes Gutenberg-Universität Mainz
Information Systems & Management
Lehrstuhl Wirtschaftsinformatik und Betriebswirtschaftslehre
D-55099 Mainz, Germany
E-Mail: {fettke|loos}@wiwi.uni-mainz.de

Abstract: Unter dem Begriff Refactoring werden bisher innerhalb der Literatur vornehmlich Techniken zur Verbesserung der Qualität von *Programmcode* diskutiert. In dieser Arbeit wird der Begriff Refactoring ebenso auf die Verbesserung der Qualität von *Informationsmodellen* ausgeweitet. Zur Demonstration werden verschiedene Techniken zum Refactoring von Ereignisgesteuerten Prozessketten vorgestellt. Diese umfassen bspw. Techniken zur strukturierten Modellierung von Prozessen oder der Verwendung ausgewiesener Verknüpfungsoperatoren von Prozesssträngen. Derartige Model Refactoring-Techniken können die Funktionalität von Werkzeugen zur Prozessmodellierung sinnvoll ergänzen, indem dem Modellkonstrukteur abstraktionsreiche Operationen zur Verfügung gestellt werden.

1 Motivation

Unter dem Begriff Refactoring wird eine Menge von Techniken zusammengefasst, die darauf abzielen, die Qualität des Programmcodes zu verbessern, ohne die Funktionalität des Softwaresystems zu verändern [Re99]. Die Bedeutung von Refactoring-Techniken wird insbesondere von Vorgehensweisen der Agilen Softwareentwicklung betont [bspw. Be00]. Konventionelle Vorgehensweisen des Software Engineering sehen zwar ebenso Aktivitäten innerhalb der Wartungsphase eines Softwaresystems vor, die auf die Verbesserung der Qualität eines Softwaresystems abzielen. Derartige Aktivitäten sind indes zumeist darauf ausgelegt, Programmfehler zu beheben bzw. neue, bisher nicht für relevant erachtete Funktionalitäten zu implementieren [Ba01, S. 1090-1100]. Dahingegen ist der ausdrückliche Zweck von Refactoring, Architektur, Entwurf oder die Art der Codierung eines Softwaresystems zu verbessern.

Eine der ersten Arbeiten, die eine umfassende Vorstellung von Refactoring-Techniken präsentiert, wurde 1992 von Opdyke vorgelegt [Op92]. Einen aktuellen Überblick über Refactoring-Techniken findet sich in [Fo02] und die dort angegebenen Quellen. Die Durchsicht des Materials zeigt, dass bekannte Refactoring-Techniken primär darauf abzielen, den Programmcode eines Softwaresystems zu verändern. Demnach sind sie gemäß der Architektur Integrierter Informationssysteme (ARIS) [Sc01] auf der Implementierungsebene einzuordnen. Um diese Techniken effektiv zu unterstützen, gibt es inzwischen eine Reihe von Softwarewerkzeugen, die in der Lage sind, Refactoring-Techniken (halb-)automatisch durchzuführen. Derartige Werkzeuge werden als Refactoring Browser bezeichnet. Um eine reibungslose Zusammenarbeit zwischen Refactoring

Browser und der Entwicklungsumgebung zu ermöglichen, werden beide Werkzeuge zunehmend miteinander integriert.

Bei der Entwicklung von Informationssystemen setzt sich zunehmend eine modellbasierte Entwicklung durch [LS95; OMG02]. Vor diesem Hintergrund stellt sich die Frage, ob Refactoring-Techniken nicht ebenfalls direkt auf Modellebene durchgeführt werden können. Derartige Techniken wären folglich auf einer fachkonzeptionellen Ebene einzuordnen. Diese Idee erscheint vor einem weiteren Grund ebenso interessant: Fowler [Fo00] verwendet bei der Beschreibung von Refactoring-Techniken nicht ausschließlich entsprechende Programmcode-Auszüge. Vielmehr setzt er ebenso UML-Darstellungen zur Verdeutlichung der Refactoring-Techniken ein. Deshalb erscheint es möglich, Refactoring-Techniken ebenso auf Modellebene durchzuführen.

Die Idee eines modellbasierten Refactoring wird bereits in der Arbeit von [BSF02] aufgegriffen. Die Autoren erläutern Refactoring-Techniken für UML-Modelle, wobei insbesondere auf Class-, Activity- and State-Diagrams eingegangen wird. Darüber hinaus wird die Funktionsweise eines Werkzeuges für das Refactoring von UML-Modellen beschrieben.

In der vorliegenden Arbeit soll die Idee eines Refactoring auf Modellebene weitergeführt werden, indem Refactoring-Techniken für Ereignisgesteuerte Prozessketten (EPK) eingeführt werden. Die Arbeit ist wie folgt strukturiert: Nach diesem einleitenden Kapitel wird in Kapitel 2 der Begriff des Model Refactoring eingeführt. Kapitel 3 erläutert verschiedene Refactoring-Techniken für EPK. Die Arbeit schließt mit einer knappen Zusammenfassung und einem Ausblick auf sich anschließende Arbeiten.

2 Der Begriff Refactoring

Der Begriff des Model Refactoring soll hier aufbauend auf dem Begriff des Code Refactoring eingeführt werden. Der Begriff des Code Refactoring wird verstanden als „a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior“[Fo00, S. 53].

Wesentlich an dieser Begriffsauffassung sind zwei Aspekte: Einerseits zählen nur solche Veränderungen am Programmcode als Refactoring, die dazu führen, dass der Programmcode einfacher zu verstehen bzw. zu verändern ist. Diese Definition schließt es aus, dass bspw. Maßnahmen zur Performance-Verbesserung als Refactoring verstanden werden, da diese i. d. R. dazu führen, dass der Programmcode schlechter wartbar wird. Andererseits betont die Definition, dass durch eine Refactoring-Technik die Programmfunktionalität nicht verändert wird. Das bedeutet, dass aus der Sicht des Programmbenutzers die Änderungen am Programmcode nicht bemerkt werden.

Aufbauend auf der Definition des Code Refactoring soll der Begriff des Model Refactoring eingeführt werden. Hierzu ist zunächst der Begriff des Informationsmodells festzulegen: Gemäß einer konstruktivistischen Auffassung ist ein Informationsmodell „das Ergebnis einer Konstruktion eines Modellierers, der für Anwendungssystem- und Orga-

nisationsgestalter Informationen über zu modellierende Elemente eines Systems zu einer Zeit als relevant mit Hilfe einer Sprache deklariert.[im Original z. T. mit Hervorhebungen, die Autoren]“[Sc98, S. 63]. Ein Refactoring eines Informationsmodells wird verstanden als das Ergebnis einer alternativen Konstruktion desselben ursprünglichen Systems, die leichter zu verstehen oder zu verändern ist. Im Folgenden wird bei der Verwendung des Begriffs Refactoring stets Bezug genommen auf ein Model Refactoring. Das Informationsmodell, an dem eine Refactoring-Technik angewandt worden ist, soll hier als transformiertes Informationsmodell bezeichnet werden.

Wesentlich an der Definition des Begriffs des Model Refactoring sind zwei Aspekte: Einerseits haben sowohl das ursprüngliche als auch das transformierte Informationsmodell dasselbe System als Gegenstand. Durch Refactoring werden demnach zwei oder mehr äquivalente Repräsentationen eines Systems konstruiert. Mit anderen Worten soll durch Anwendung von Refactoring-Techniken die Semantik des Informationsmodells nicht verändert werden. Andererseits wird an das transformierte Informationsmodell der Anspruch gestellt, dass dieses leichter zu verstehen und zu verändern ist.

Refactoring geht über die Anwendung von Repräsentationsvorschriften wie Layout-Konventionen [siehe bspw. die Grundsätze ordnungsmäßiger Modellierung von BRS95] hinaus. Während Repräsentationsvorschriften für Informationsmodelle sich ausschließlich auf einer Darstellungsebene von Informationsmodellen bewegen, betreffen Refactoring-Techniken konzeptionelle Modellierungsaspekte. Durch die Anwendung einer Refactoring-Technik wird derselbe Sachverhalt durch alternative Modellierungskonzepte zum Ausdruck gebracht – und nicht durch alternative Repräsentationen des Informationsmodells.

Gleichzeitig hat die Anwendung von Refactoring-Techniken nicht die Reichweite von Techniken des Business Process (Re-)Engineering (BPR) [Ga94]. BPR-Techniken fokussieren die Optimierung von Prozessmodellen. Typische BPR-Techniken sind bspw. das Parallelisieren von Funktionen oder der Wegfall von redundanten bzw. nicht wertschöpfenden Funktionen. Durch Anwendung von BPR-Techniken wird implizit unterstellt, dass das ursprüngliche und das transformierte Informationsmodell verschiedene Systeme zum Gegenstand haben. Dies wird u. a. dadurch deutlich, dass zwischen einem Ist- und einem Soll-Modell begrifflich sowie konzeptionell zu unterscheiden ist. Somit können BPR-Techniken nicht als Refactoring verstanden werden. Vor diesem Hintergrund muss die Ansicht kritisch beurteilt werden, dass das Parallelisieren (oder Sequenzialisieren) von Aktivitäten in einem Activity Diagramm als Refactoring verstanden wird [BSF02, S. 367f.]. Derartige Transformationsschritte verändern die Modellsemantik und werden demnach hier nicht als Refactoring bezeichnet.

Refactoring-Techniken bieten im Vergleich zu Qualitätskriterien für Prozessmodelle den Vorteil, dass sie direkte Handlungsanweisungen geben, wie eine schlechte Modellierung in eine bessere überführt werden kann. Dagegen liefern Qualitätskriterien nur einen Maßstab zur Beurteilung eines Informationsmodells und keine direkten Handlungsanweisungen zur Modellverbesserung.

Refactoring-Techniken zielen nicht darauf ab, Syntax oder Semantik der Modellierungssprache zu spezifizieren. Statt dessen wird davon ausgegangen, dass Syntax und Semantik der Modellierungssprache bereits vollständig, konsistent und präzise festgelegt sind. Die Anwendung von Refactoring-Techniken führt demnach nicht zu einer Veränderung der Syntax und Semantik einer Modellierungssprache[1].

Zur weiteren Charakterisierung von Refactoring-Techniken erscheint der Vergleich zu Planungsheuristiken sinnvoll. Planungsheuristiken können als Strukturierungsregeln von Problemen verstanden werden, die in der vorliegenden Form nicht lösbar sind. Die Anwendung von Planungsheuristiken ist nicht formallogisch begründbar, sondern gründet ausschließlich auf Plausibilitätsüberlegungen [Ad93, S. 414-418]. Ebenso sind die Vorzüge von Refactoring-Techniken nicht formallogisch zu begründen, sondern im Hinblick auf plausible Einzelaspekte zu erklären. Ähnlich wie Planungsheuristiken sind Refactoring-Techniken problemspezifisch angelegt und erlauben eine Modellverbesserung nur in ganz spezifischen Modellierungssituationen.

3 Darstellung ausgewählter Refactoring-Techniken

3.1 Refactoring „Einführung des SEQ-Operators“

Ein Modell einer Menge von Funktionen, die ausschließlich sequentiell, aber in beliebiger Reihenfolge auszuführen sind, wird in Abbildung 1a dargestellt. Solche Funktionsfolgen können bspw. bei der Qualitätssicherung auftreten, bei der mehrere Qualitätstests sequentiell, aber in beliebiger Reihenfolge auszuführen sind. Die vorgenommene Modellierung ist aus verschiedenen Gründen unbefriedigend. Beispielsweise ist die Anzahl zu modellierender Prozessstränge bei einer größeren Menge von Funktionen relativ groß. Gleichzeitig wird die Neuaufnahme bzw. das Löschen von Funktionen innerhalb der Gesamtmenge der Funktionen relativ aufwendig.

Aus diesem Grunde wird das Refactoring „Einführung des SEQ-Operators“ eingeführt. Dieses Refactoring sieht vor, den zugrunde gelegten Sachverhalt durch Verwendung des SEQ-Operators zu modellieren [Ro95, S. 240-243]. Ein entsprechendes transformiertes Modell ist in Abbildung 1b dargestellt. Sämtliche Prozessstränge, die innerhalb des SEQ-Operatoren eingeklammert sind, sind sequentiell, aber in beliebiger Reihenfolge zu durchlaufen.

Bei der Anwendung des SEQ-Operators ist eine Besonderheit zu beachten. Es Bedarf der Unterscheidung, ob die Reihenfolge der abzuarbeitenden Prozessstränge zu Beginn der Abarbeitung einmalig festgelegt wird oder ob die Reihenfolge nach jeder Abarbeitung einer Teilsequenz erneut ermittelt wird. Der letzte Fall würde bedeuten, dass nach der Abarbeitung jeder Teilsequenz eine entsprechende Fallunterscheidung eingeführt wird, in der bestimmt wird, welche nächste Teilsequenz ausgeführt werden soll. Eine solche Interpretation des SEQ-Operators wird in Anlehnung an Rosemann [Ro95, S. 241] als

[1] Die Syntax und Semantik von EPK werden von [Ru98, S. 79-102; NR02] formal eingeführt.

dynamische Sequenz bezeichnet. Das resultierende Prozessmodell ist bei einer dynamischen Sequenz komplizierter.

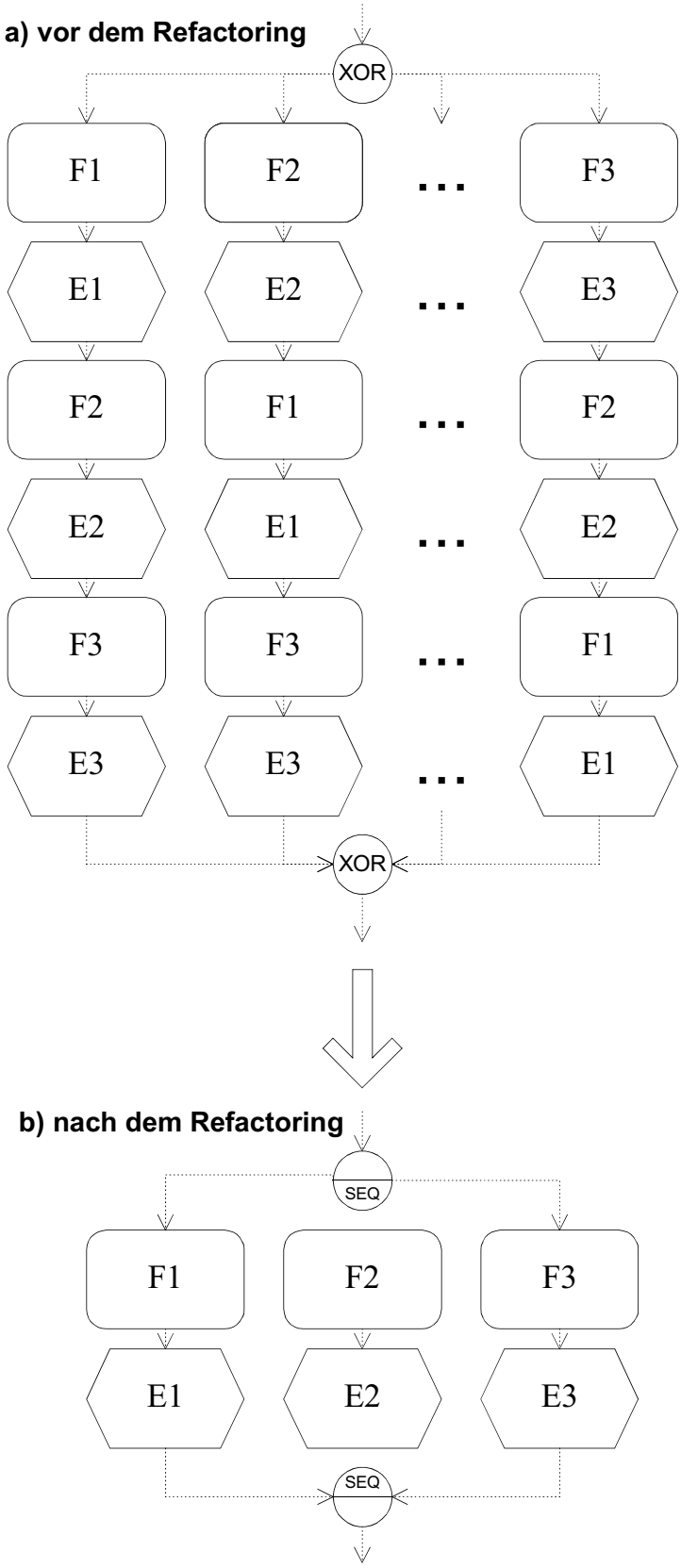


Abb. 1: Refactoring „Einführung des SEQ-Operators“

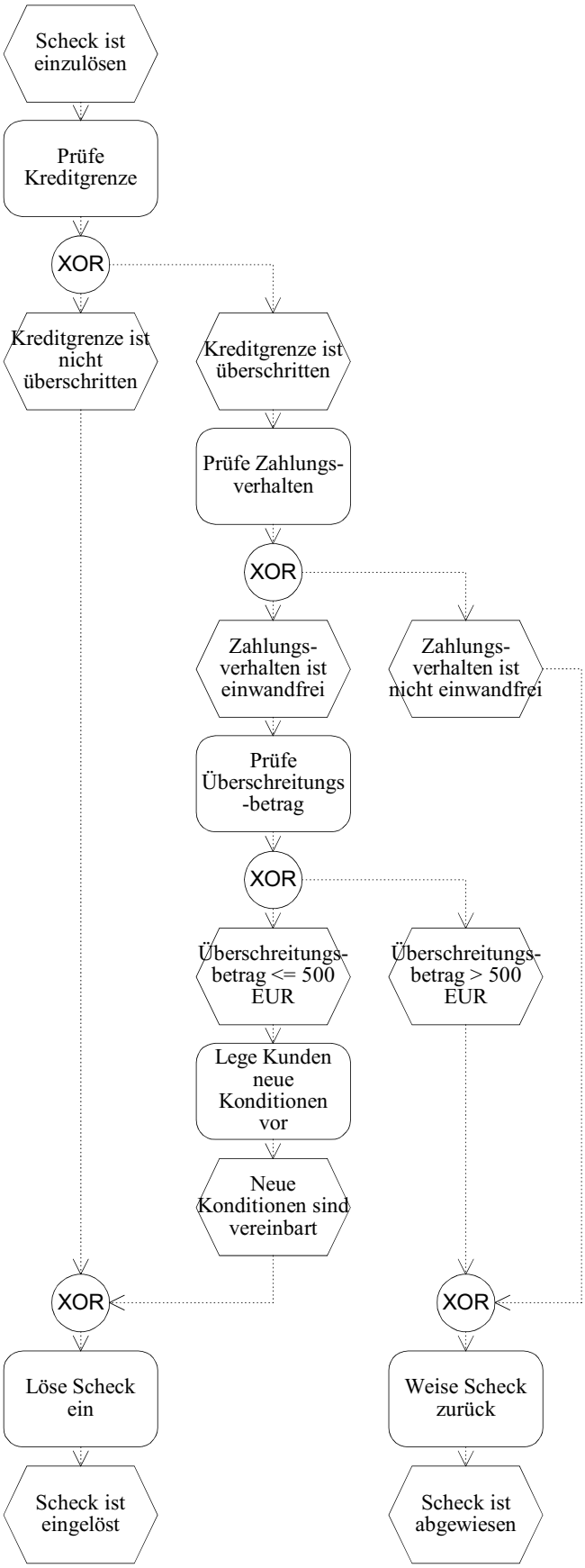
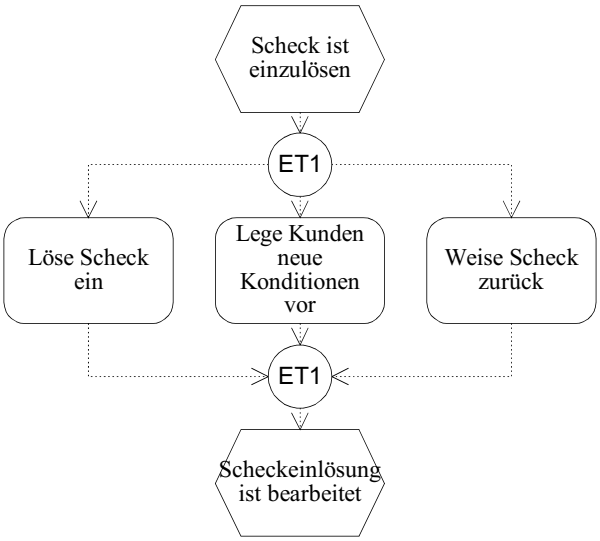


Abb. 2: Refactoring „Einführung des ET-Operators“ (vor dem Refactoring)



ET1: Scheck ist einzulösen		R1	R2	R3	R4
B1	Ist Kreditgrenze überschritten?	J	J	J	N
B1	Ist Zahlungsverhalten einwandfrei?	J	J	N	-
B1	Ist Überschreitungsbetrag <= 500 EUR?	J	N	-	J
F1	Löse Scheck ein	X	X		X
F2	Weise Scheck zurück			X	
F3	Lege Kunden neue Konditionen vor		X		

Abb. 3: Refactoring „Einführung des ET-Operators“ (nach dem Refactoring)

3.2 Refactoring „Einführung des ET-Operators“

In Abbildung 2 ist ein Modell eines Geschäftsprozesses zur Bearbeitung von Scheckeinreichungen bei einer Bank dargestellt. Dieses Modell beruht auf einer Falldarstellung bei [Ba01, S. 270f.]. Die wesentlichen Kernfunktionen im Modell umfassen „Löse Scheck ein“, „Weise Scheck zurück“ sowie „Lege Kunden neue Konditionen vor“. Bei den anderen Funktionen im Prozess handelt es sich um verschiedene Funktionen zur Überprüfung der Kreditgrenze, des Überschreitungsbetrages sowie des Zahlungsverhaltens des Kunden. Je nach Ergebnis dieser Prüfung sind dann verschiedene Kernfunktionen des Prozesses auszuführen. Im Ganzen handelt es sich bei diesem Sachverhalt um einen wohl-strukturierten Prozess, dessen Komplexität gerade darin besteht, dass gewisse Funktionen auf Grundlage einer bestimmten Entscheidungslogik auszuführen sind.

Die im Prozessmodell implizit enthaltene Entscheidungslogik ist aus der Prozesskette nicht unmittelbar ersichtlich. Daher ist es relativ schwer, den Prozess hinsichtlich Vollständigkeit aller möglichen Entscheidungsverzweigungen, Konsistenz der getroffenen Entscheidungen sowie etwaige Redundanzen zu überprüfen.

Aus diesem Grunde wird vorgeschlagen, die Entscheidungslogik durch Einführung einer Entscheidungstabelle explizit zu modellieren. Die Verwendung des ET-Operators in EPK wird in Abbildung 3 dargestellt. Auf Basis der Entscheidungstabelle ist es nun leicht möglich, entsprechende Vollständigkeits-, Konsistenz- oder Redundanzprüfungen vorzunehmen.

3.3 Refactoring „Einführung einer strukturierten Prozessmodellierung“

EPK können durch Schleifenkonstrukte beliebig geschachtelt werden. Dagegen hat sich auf der Ebene der Programmiersprachen die Erkenntnis durchgesetzt, stets eine Strukturierte Schleife zu verwenden, um Goto-Sprünge im Programmablauf zu vermeiden [Ba01, S. 265-268]. Aus diesem Grunde wird für nicht strukturierte EPK ein Refactoring vorgeschlagen, was eine strukturierte Prozessmodellierung ermöglicht.

Dieses Refactoring sieht so aus, dass der Schleifenrumpf eines Prozesses jeweils in einer eigenen EPK zu kapseln ist (vgl. Abbildung 4). Innerhalb einer EPK dürfen keine beliebigen Rücksprünge zu anderen Funktionen vorgenommen werden. Statt dessen dürfen solche Rücksprünge jeweils nur am Anfang bzw. Ende einer EPK vorgenommen werden. Auf diese Art und Weise wird eine strukturierte Prozessmodellierung erzwungen. In der Abbildung 4 ist das Refactoring graphisch dargestellt.

Ergänzend sei darauf hingewiesen, dass es alternativ ebenso möglich wäre, durch eine Veränderung des sprachbasierten Metamodells[2] von EPK eine gänzliche Vermeidung unstrukturierter Schleifen herbeizuführen. Dazu könnten bspw. explizite Modellierungskonstrukte eingeführt werden, die den Schleifenbeginn und das –ende markieren. Eine ähnliche Variante wurde ebenso für die Konstruktion des Metamodells von Programmablaufplänen gewählt [Ba01, S. 262-265]. Prinzipiell werden in der Strukturierten Programmierung drei Schleifen-Varianten unterschieden: FOR-, WHILE- und REPEAT-UNTIL-Schleifen. Dies bedeutet, dass das Metamodell von EPK um drei Modellierungskonzepte zu erweitern wäre. Dieser Lösungsansatz ist indes nicht als eine Refactoring-Technik einzustufen (vgl. Abschnitt 2).

[2] Ein sprachbasiertes Metamodell beschreibt die Syntax einer Modellierungssprache [St96, S. 19-24].

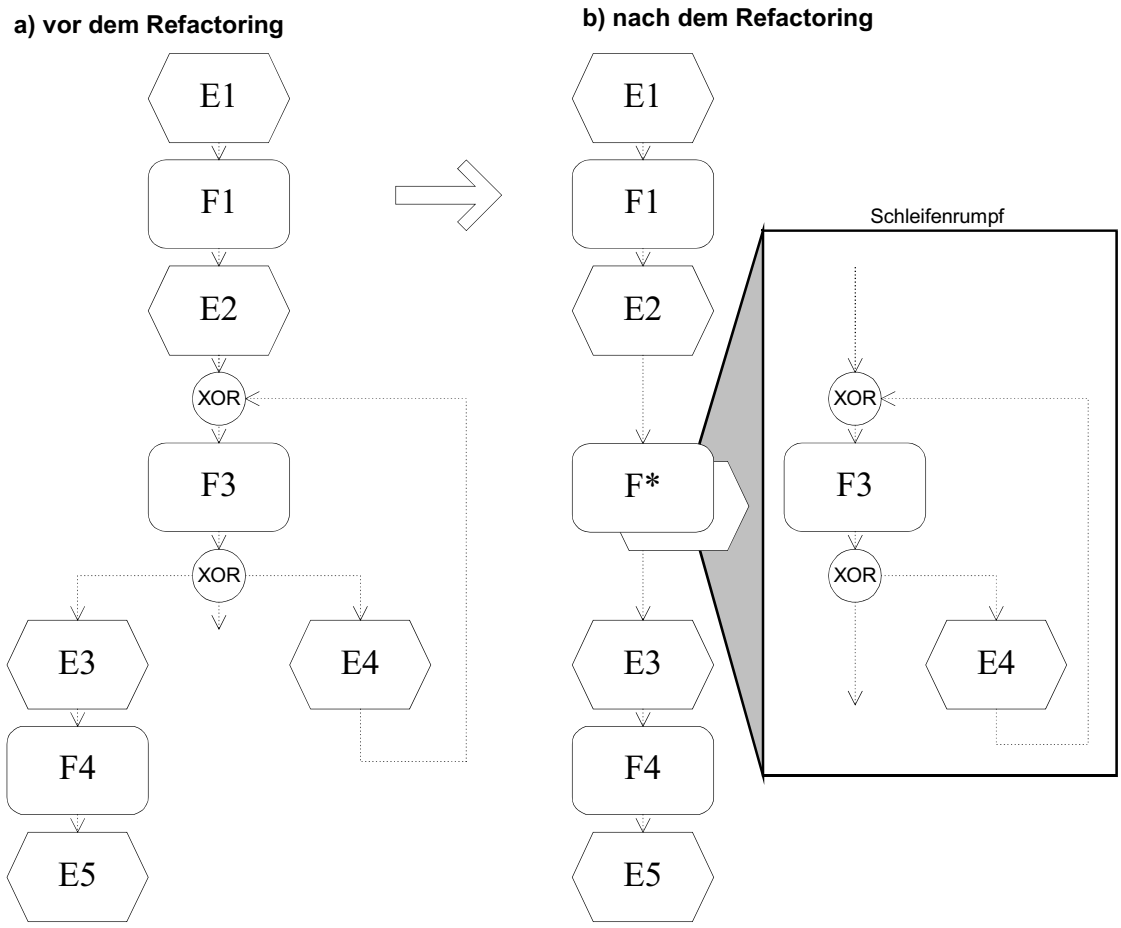


Abb. 4: Refactoring „Einführung einer strukturierten Prozessmodellierung“

3.4 Refactoring „Vermeidung unnötiger Entscheidungsverallgemeinerungen“

EPK erlauben das Formulieren von Entscheidungsalternativen, so dass es möglich ist, alternative Prozessstränge je nach Entscheidungssituation zu durchlaufen. Entsprechende Entscheidungsbedingungen können so formuliert sein, dass sie sich sowohl gegenseitig ein- als auch ausschließen. Falls sich die Entscheidungsbedingungen gegenseitig ausschließen, sollte dies ebenso durch einen Entscheidungsoperator gekennzeichnet werden, der nur eine ausschließliche Entscheidungssituation zulässt.

Diese Refactoring-Technik wird an einem Beispiel in Abbildung 5 näher beschrieben. In dem aufgegriffenen Modellausschnitt wird auf Grundlage der Auftragssumme entschieden, welche Schritte zur Auftragsbearbeitung notwendig sind. Bei einer Auftragssumme kleiner oder gleich 10.000 EUR wird der linke Prozessstrang, bei einer Auftragssumme größer 10.000 EUR der rechte Prozessstrang bearbeitet. Offensichtlich schließen sich beide Ereignisse gegenseitig aus. Indes ist die Entscheidungssituation mit Hilfe des OR-Operator modelliert, der prinzipiell eine einschließende Entscheidungssituation ausdrückt. Folglich stellt der gewählte OR-Operator eine unnötige Verallgemeinerung der Entscheidungssituation dar und sollte durch den XOR-Operator ersetzt werden. Das durch Anwendung der beschriebenen Refactoring-Technik transformierte Prozessmodell

ist in Abbildung 5b dargestellt. Das transformierte Prozessmodell enthält keine unnötige Entscheidungsverallgemeinerung mehr, vielmehr ist der Entscheidungsoperator der aufgeführten Entscheidung inhaltlich angemessen.

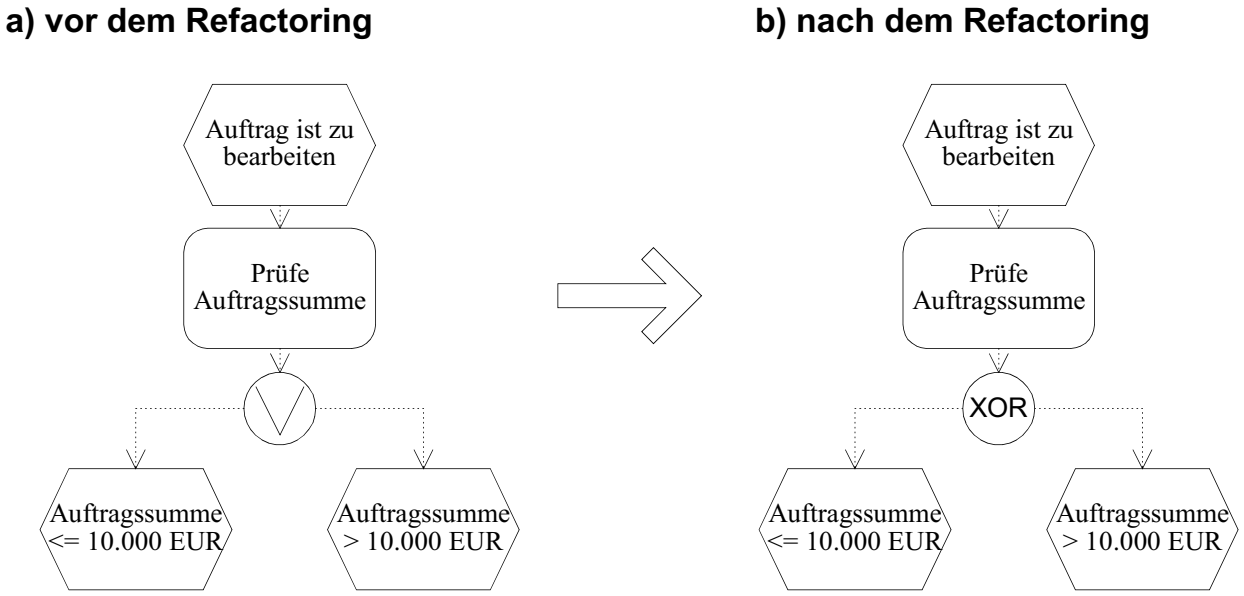


Abb. 5: Refactoring „Vermeidung unnötiger Entscheidungsverallgemeinerungen“

Es sei darauf hingewiesen, dass die zuvor dargestellten Refactoring-Techniken (vgl. Abschnitte 3.1 bis 3.3) auf einer formalen Ebene anzusiedeln sind und prinzipiell vollautomatisiert angewendet werden können. Dagegen bewegt sich die in diesem Abschnitt eingeführte Refactoring-Technik auf einer inhaltlichen Ebene. Die Anwendung der Technik erfordert die inhaltliche Analyse, ob sich die möglichen Entscheidungsalternativen gegenseitig ein- bzw. ausschließen. Diese Analyse kann nicht vollautomatisiert, sondern ausschließlich vom Modellkonstrukteur durchgeführt werden.

3.5 Weitere Refactoring-Techniken

Innerhalb der Literatur existieren Arbeiten, die Transformationstechniken für EPK oder andere Prozessmodellierungssprachen beschreiben, ohne sich dabei explizit auf den Begriff Refactoring zu beziehen. Derartige Ansätze werden im Folgenden kurz aufgegriffen.

Rodenhagen beschreibt eine Reihe von Transformationstechniken für EPK [Ro97]. Die primäre Zielstellung seiner Arbeit ist es, eine „widerspruchsfreie und plausible Semantik für die Prozesskette zu definieren“[Ro97, S. 11, im Original hervorgehoben, die Autoren]. Darüber hinaus wird als zweites Ziel angestrebt, „eine Reihe von Empfehlungen zur Modellierung *korrekter* Prozessketten aufzustellen“[Ro97, S. 11, Hervorhebungen nicht im Original, die Autoren]. Damit verfolgt seine Arbeit nicht dieselbe Zielstellung

wie Refactoring-Techniken, da letztere nicht darauf abzielen, syntaktische oder semantische Unzulänglichkeiten der Modellierungssprache zu beheben (vgl. Abschnitt 2). Nichtsdestotrotz stellt der Autor auch Transformationstechniken vor, die als Refactoring-Techniken einzustufen sind, da diese weder Syntax noch Semantik von EPK betreffen. Der Autor beschreibt bspw. als eine Transformationstechnik die Reduktion binärer OR-, AND- und XOR-Operatorenketten, die kaskadenförmig angeordnet sind, in einen entsprechenden n-ären OR-, AND- bzw. XOR-Operator [Ro97, S. 117-119]. Ein zweites Beispiel ist die vom Autor erläuterte Möglichkeit zur Transformation von Operatorenketten beliebiger Schachtelungstiefe in eine äquivalente zweistufige Prozesskette [Ro97, S. 119-123]. Beide genannten Transformationstechniken sind als Refactoring-Techniken einzustufen. Andere Transformationstechniken (bspw. zur Vermeidung von Ereigniszusammenführungen mit XOR-Operatoren [Ro97, S. 56]) sind dagegen nicht als Refactoring-Techniken zu verstehen.

Ebenso verfolgt Rump im Rahmen eines durchgängigen Managements von Geschäftsprozessen die Zielsetzung, die Syntax und Semantik von EPK formal zu spezifizieren [Ru98, S. 14f.]. Der Autor beschreibt bspw. die Überführung n-ärer Operatoren in binäre Operatoren [Ru98, S. 89f.]. Durch diese Transformation wird die Modellkomplexität im Sinne der Anzahl der Elemente und Beziehungen vergrößert und kann daher nicht als Refactoring-Technik verstanden werden. Die vom Autor vorgestellten Verfahren zum Nachweis allgemeiner (bspw. Verklemmungsfreiheit) und spezifischer Eigenschaften (bspw. Auslösung bestimmter Ereignisse) von EPK unterstützen zwar die Qualitätssicherung von EPK und geben implizit auch Hinweise zur Verbesserung von EPK. Allerdings stellen diese Verfahren keine fallspezifischen Transformationsregeln bereit und sind daher nicht als Refactoring-Techniken zu verstehen.

Verschiedene Transformationstechniken für Prozessmodelle auf Basis einer Workflow Modeling Language werden von Sadiq und Orlowska beschrieben [SO00]. Einerseits erläutern die Autoren Transformationstechniken für eine äquivalente Modelltransformation, die als Refectoring-Techniken zu verstehen sind. Diese behandeln bspw. unnötige Synchronisationsbedingungen oder kaskadierende Kontrollflussverknüpfungen. Andererseits werden ebenso Transformationstechniken vorgestellt, die explizit den im Modell dargestellten Sachverhalt verändern.

4 Zusammenfassung und Ausblick

Es wurde gezeigt, dass Refactoring-Techniken nicht nur auf der Ebene von Programmcode angewendet werden können, sondern auch auf der Ebene von Informationsmodellen. Derartige Techniken wurden als Model Refactoring bezeichnet. In dieser Arbeit wurden drei verschiedene Refactoring-Techniken für EPK vorgestellt. In kommenden Arbeiten sind weitere EPK-Refactoring-Techniken zu identifizieren und zu beschreiben. Ferner erscheint es sinnvoll, derartige Refactoring-Techniken in einem Werkzeug zur EPK-Modellierung zu integrieren, um den Benutzer bei der Prozessmodellierung weiter zu unterstützen. Refactoring-Techniken bieten damit eine abstraktionsreiche Möglichkeit, EPK zu manipulieren. Durch Refactoring-Techniken ist es dem Benutzer möglich, EPK über dem Niveau von elementaren Operationen wie „Füge Funktion ein“ oder „De-

finiere Kontrollfluss“ zu verändern. Bspw. ist es denkbar, dass die Refactoring-Technik „Einführung des SEQ-Operator“ in der Art werkzeuggestützt angewendet werden kann, dass zunächst alle Prozessstränge mit den entsprechenden Funktionen selektiert werden. Anschließend kann aus einem (Kontext-)Menü die Refactoring-Technik aufgerufen werden. Die entsprechenden Modelltransformationen können vom Werkzeug automatisiert vorgenommen werden. Gleichzeitig hat das Modellierungswerkzeug die Möglichkeit, notwendige Konsistenzsicherungen bei der Anwendung der Refactoring-Technik zu überwachen. Durch die Entwicklung derartiger Funktionen erhalten Modellierungswerkzeuge einen weiteren Vorteil gegenüber reinen Werkzeugen zur graphischen Visualisierung von Informationsmodellen, wodurch die Identifikation und Dokumentation neuer Refactoring-Techniken fruchtbar erscheint.

Literaturverzeichnis

[Ad93] Adam, D.: Planung und Entscheidung - Modelle - Ziele - Methoden. 3. Aufl., Wiesbaden 1993.

[Ba01] Balzert, H.: Lehrbuch der Software-Technik - Software-Entwicklung. 2. Aufl., Heidelberg, Berlin 2001.

[Be00] Beck, K.: Extreme Programming Explained - Embrace Change. Boston et al. 2000.

[BRS95] Becker, J.; Rosemann, M.; Schütte, R.: Grundsätze ordnungsmäßiger Modellierung. In: Wirtschaftsinformatik 37 (1995) 4, S. 435-445.

[BSF02] Boger, M.; Sturm, T.; Fragemann, P.: Refactoring Browser for UML. In: Organisatoren der Net.ObjectDays (Hrsg.): Net.ObjectDays 2002 - Tagungsband, Hauptkonferenz, 07. - 10. Oktober 2002, Messekongresszentrum Messe Erfurt. Erfurt 2002, S. 364-376.

[Fo00] Fowler, M.: Refactoring - Improving the Design of Existing Code. Boston et al. 2000.

[Fo02] Fowler, M.: Refactoring Home Page. http://www.refactoring.com/, Abruf am 2002-10-18.

[Ga94] Gaitanides, M.; Scholz, R.; Vrohlings, A.; Raster, M.: Prozeßmanagement - Konzepte, Umsetzungen und Erfahrung des Reengineering. München, Wien 1994.

[LS95] Loos, P.; Scheer, A.-W.: Vom Informationsmodell zum Anwendungssystem - Nutzenpotentiale für den effizienten Einsatz von Informationssystemen. In: W. König (Hrsg.): Wirtschaftsinformatik '95 - Wettbewerbsfähigkeit, Innovation, Wirtschaftlichkeit. Heidelberg 1995, S. 185-201.

[NR02] Nüttgens, M.; Rump, F. J.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: J. Desel; M. Weske (Hrsg.): Promise 2002 - Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen - Proceedings des GI-Workshops und Fachgruppentreffens (Potsdam, Oktober 2002). Bonn 2002, S. 64-66.

[OMG02] OMG: OMG Model Driven Architecture. http://www.omg.org/mda/, Abruf am 2002-10-19.

[Op92] Opdyke, W. F.: Refactoring Object-Oriented Frameworks. PhD Thesis, 1992.

[Re99] Reißing, R.: Refactoring. In: Informatik-Spektrum 22 (1999), S. 210-211.

[Ro97] Rodenhagen, J.: Darstellung ereignisgesteuerter Prozeßketten (EPK) mit Hilfe von Petrinetzen. Diplomarbeit, Hamburg 1997.

[Ro95] Rosemann, M.: Erstellung und Integration von Prozeßmodellen - Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung. Diss., Münster 1995.

[Ru98] Rump, F.: Durchgängiges Management von Geschäftsprozessen auf Basis ereignisgesteuerter Prozeßketten. Diss., Oldenburg 1998.

[SO00] Sadiq, W.; Orlowska, M. E.: On Business Process Model Transformations. In: A. H. F. Laender; S. W. Liddle; V. C. Storey (Hrsg.): Conceptual Modeling - ER 2000 - 19th International Conference on Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000 Proceedings. Berlin et al. 2000, S. 267-280.

[Sc01] Scheer, A.-W.: ARIS: Modellierungsmethoden, Metamodelle, Anwendungen. 4. Aufl., Berlin et al. 2001.

[Sc98] Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung - Konstruktion konfigurations- und anpassungsorientierter Modelle. Wiesbaden 1998. (Zugl.: Diss., Münster 1997)

[St96] Strahringer, S.: Metamodellierung als Instrument des Methodenvergleichs - Eine Evaluierung am Beispiel objektorientierter Analysemethoden. Aachen 1996. (Zugl.: Diss., Darmstadt 1996)

Making EPCs fit for Workflow Management

Juliane Dehnert, TU Berlin, CIS
email: dehnert@cs.tu-berlin.de

Abstract: EPCs are widely accepted for the modeling of business processes. They provide a common understanding between different participants and are therefore extremely suitable as a starting point for Workflow management management. But Workflow management comprises more than modeling, namely the analysis of the processes, their improvement and finally their support during run-time. Although the EPC tool support through the ARIS-toolset is mainly responsible for their wide use, it is still sobering which functionality is provided. Mainly, EPCs can be drawn and watched. There is no behavior analysis, only limited simulation facilities, and no execution support at all. One reason for the pure support is the disunity about the EPC semantic. In this paper we propose the use of a pragmatic interpretation of the correctness of EPCs which enables us to provide a comprehensive approach for workflow management. The proposed approach is based on the modeling with EPCs and leads the modeler to a sound process description which can be used as base for a reliable process execution at run-time.

Introduction

In recent years, workflow management was seen as one of the most promising reserves in order to straighten up their business processes. Many companies started business reengineering projects to identify and use existing optimization potential.

Many of these projects got stuck, as the support was limited to the gathering of the existing processes, but did not provide any help thereafter. Beside the modeling of the existing business processes, a comprehensive approach for workflow management should also provide means for the analysis, the optimization, and the execution support of the described processes.

One of the core problems was the identification of a suitable modeling technique that meets an adequate abstraction of the real world and suffices the various requirements posed within the different phases: modeling, analysis, and execution support.

Also, if the focus is set on the core - the control flow aspects only, it has been found impossible to find a modeling technique that fits the purpose of the different phases adequately.

The main demands on a modeling technique within the different phases are summarized in the following:

Modeling The most important features of a modeling technique used for the first phase - the modeling, are intuitive concepts and tool support. Only, if the elements and their composition are easy to understand and to composite, the technique will win recognition at the user side.

Analysis/Optimization For the analysis, the picture is already completely different. The significant features that support analysability are formal foundation together with a theoretical background. Preferably, there should be a wide range of criteria in combination with tools that support their implementation.

Execution Support Workflow management is completed if the execution of the modeled, and possibly optimized, process is supported at run-time. This is done through a workflow management system which reacts on external events according to the internal specification of the process. The internal representation should therefore reflect a reactive behavior, guaranteeing a reliable and prompt execution.

As already said, many of the approaches used in practice, e.g.[Sch94, LR99, Mar00, JBS97] consider mainly the modeling. Accordingly, there exist many techniques that meet the requirements of this phase. One that won most recognition in the last years are *Event driven Process Chains*. They prevail the market (at least in Germany) due to their intuitive modeling concepts, their expressive power, and last but not least, their tool support through the ARIS-toolset.

The objective of the approach presented is the use of EPCs as communication language between the domain experts and the application developers throughout the whole procedure. It is clear that they are used for the modeling itself. For this purpose their applicability has been proven already.

We do not use EPCs as base for the analysis nor for the execution support. Here, dedicated classes of Petri nets are employed which were developed especially for these domains and are supported through a rich theory and supporting tools. The use of Petri nets within these phases balances the lack of formal foundation EPCs suffer from. Still, the domain experts do not have to become Petri net experts as the understanding is maintained providing precise interfaces, in form of transformations, between the different techniques.

The main challenge of the proposed approach was to overcome the deficiencies of EPCs, namely their inherent ambiguity, and to come up with a sound model that meets the primary intuition of the modeler. The clue of the presented approach is the use of a relaxed correctness criteria that provides an adequate measure for the correctness of EPCs. Once the EPC is, what we call ”relaxed sound”, it is possible to transform the corresponding Petri net into a sound Petri net which then can be used to guarantee reliable execution at run-time. Within the proposed procedure the process description is developed gradually. The performed steps can be automatized.

On the remaining pages, the proposed approach will be sketched. A more detailed version can be found in [Deh02a].

A Comprehensive Process Model for Workflow Management

The proposed procedure is iterative and consists out of five steps. Support for modeling and analysis is provided through the steps one to three. The execution of the process then gets supported within step four and five. Figure 1 shows the general process model.

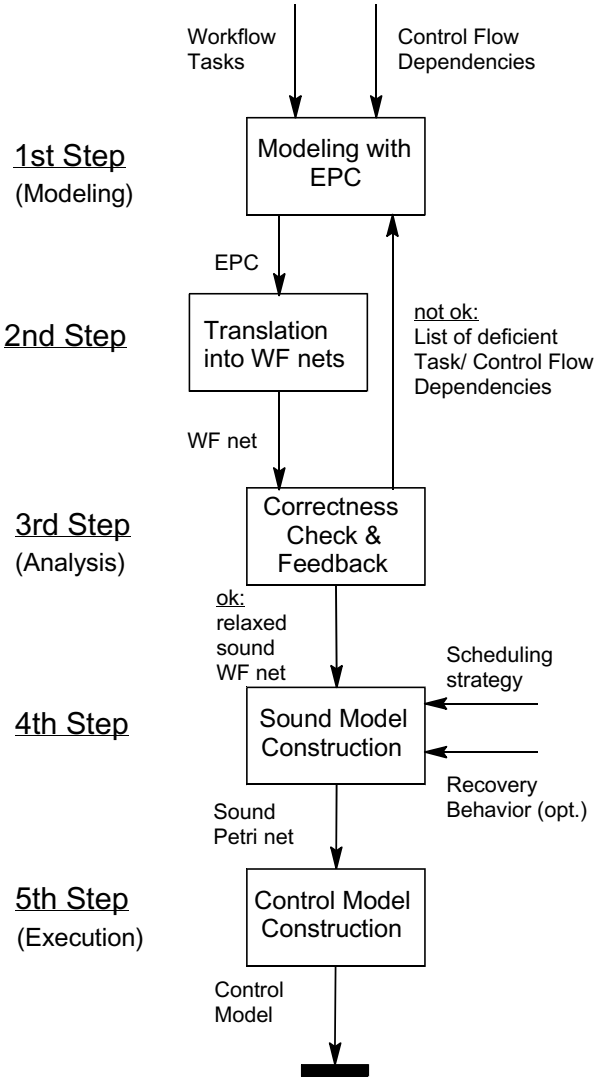


Figure 1: A Process Model for Business Process Modeling

The five steps are: “Modeling with EPC,” “Transformation into WF nets,” “Correctness Check and Feedback,”“Sound Model Construction,” and “Control Model Construction.” They are explained in the following.

1st Step: Modeling with EPCs

For the modeling of business processes we propose the use of Event-driven Process Chains (EPCs) of the Architecture of integrated Information Systems (ARIS) described in [Sch94]. EPCs are widely accepted in practice due to its comprehensive tool support. They provide a set of workflow modeling pattern with an intuitive graphical notation which has proven to meet the intuition of the application developer.

We assume the reader to be familiar with the EPC-notation. We therefore, do not describe their modeling features but just give some small examples that illustrate their modeling power. The examples have been chosen with respect to a current matter of dispute

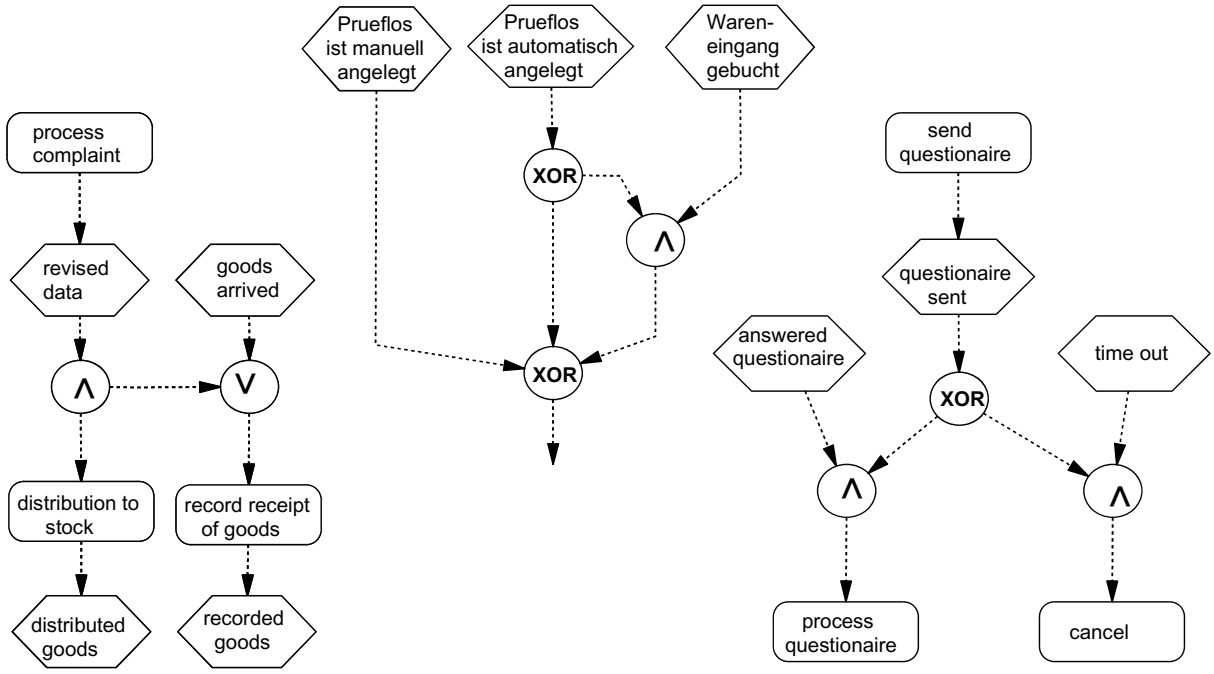


Figure 2: (a) An EPC with an OR-connector (b) One event only occurs together with another one (c) an XOR-connector following an event

within the EPC-community, regarding a formalization of EPCs (c.f. [NR02, DR01, MR00, Rum99, Aal99]). The examples emphasize the lack of a formal semantic of EPCs as they contain ambiguity and vagueness. Before going on, we want to ask the reader to consider the examples and to decide whether he or she thinks the details may depict correct behavior or not. The first picture (c.f. Figure2:a) is a detail of a larger EPC from [LSW98] and addresses the first controversial issue, namely the semantic of the OR-connector. The second detail (c.f. Figure2:b) was found within the modeling results of a SAP R/3 implementation project. It addresses the question whether EPCs have an operational semantic or not. The third detail (c.f. Figure2:c) finally contains a combination of EPC-elements that was precluded in the first description [KNS92].

Within this paper, we represent the opinion that these examples comprise correct behavior. Therefore, we start from the assumption that EPCs itself do **not** have an operational

semantic but just describe desired executions. This means, an EPC can be interpreted as a pattern that describes accepted executions. Deficient executions are only described implicitly, as the set of executions which do not fit the described pattern.

We claim that, the non-operational semantic fits an intuitive modeling understanding and is one reason why EPCs are said to be easy to learn and to understand. Modeling business

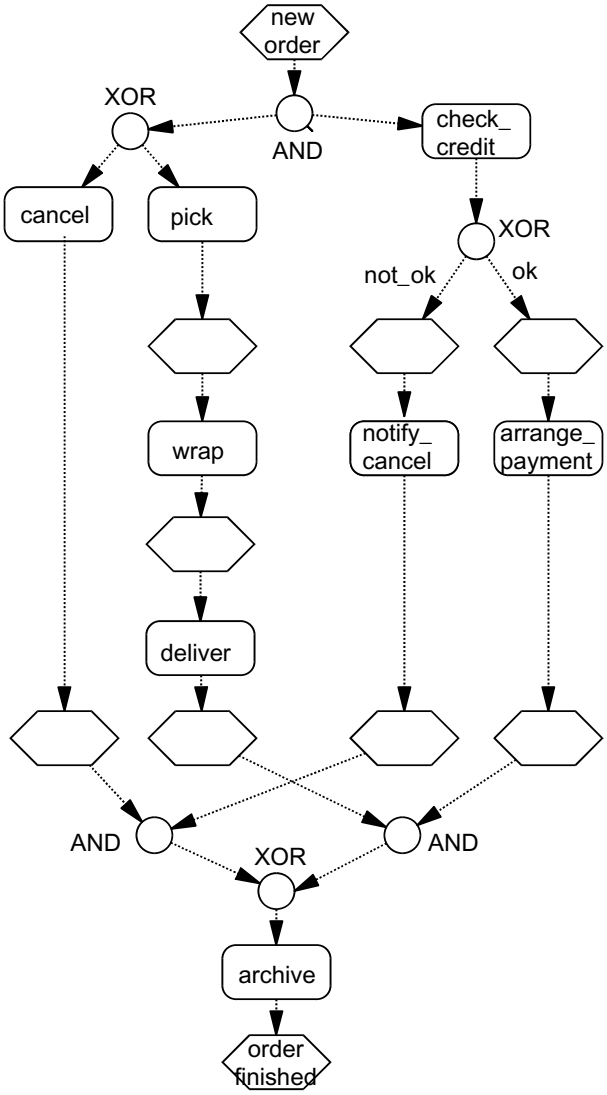


Figure 3: Handling of incoming order

processes, the modeler normally starts by describing ”What the execution should look like”, hence describes a set of good/accepted executions. This procedure fits the non-operational semantic. Modeling then means not to think about all possible executions but just specify the set of accepted executions.

We will explain the non-operational semantic again by the help of an example. Figure 3 shows an EPC modeling the process ”Handling of incoming order”.

It represents a reduced version of a real life process of a telephone company. The process models the ordering of a mobile phone which involves two departments: the accountancy

handling the payment and the sales handling the distribution.

The process starts with the event *new order*. After that, the execution is split into two parallel paths (AND split), the right one models the accounting, whereas the left one models the sales. In the accounting, the customer standing is checked (*check_credit*) first. The result of this task is either *ok* or *not_ok*. In case the result is positive, the payment is arranged (*arrange_payment*), in the latter case the instance is canceled.

The left path models the tasks on the sales side. Here, the order is either handled executing tasks *pick*, *wrap* and *deliver*, or *cancel*.

The two AND-connectors at the end make sure that only executions are accepted where both sides, the accountancy and the sales department, either cancel the instance or proceed the order. The process "Handling of incoming order" is finished by archiving information on that instance (*archive*).

An accepted execution of the EPC from Figure 3 is *check_credit, arrange_payment, pick, wrap, deliver, archive* but also *cancel, check_credit, notify_cancel, archive*.

The EPC only describes executions where the two departments work together correctly: they either both accept the order (*AND_accept*) or both reject it (*AND_cancel*). Hence, the execution *check_credit, notify_cancel, pick, wrap, deliver, archive* is not described by the EPC.

The EPC does not determine "how" the accepted executions are achieved. It does not stipulate the order of the two possible choices. It, therefore, accepts executions where the two departments work in parallel as well as executions where the two departments work sequentially. In an early design phase, this abstraction is appreciated, because it relieves the designer thinking about efficiency aspects of the execution at this point in time.

We hope that, under the assumption of a non-operational semantic, everybody will agree that the EPCs from Figure2 and Figure3 describe useful behavior and should therefore be considered correct.

In the next step of the proposed procedure, we will provide a formal semantic for EPCs by mapping them to Petri nets. The speciality about this transformation is, that the ambiguity of the EPC is preserved but made explicit. Correspondingly, the resulting Petri net will not satisfy traditional correctness measures. We will see that, deadlocks and/or spare tokens may occur. We, therefore, provide an adapted correctness criteria, which takes this into account.

2nd Step: Transformation into WF Nets

We suggest to transform the EPCs into a formal specification based on Petri nets. As suitable Petri net type we chose Workflow nets (WF nets), cf.[Aal98]. The transformation of EPCs into WF nets was introduced in [DR01] (see also appendix). It takes place in several steps. First, elements of the EPC are mapped onto Petri net-modules. Events and functions are transformed into places and transitions respectively including in- and

outgoing arcs. Routing constructs such as *AND split, AND join, XOR split, XOR join, OR split* and *OR join* are mapped to small net-pattern. The net-pattern describe the behavior of the routing constructs explicitly. This is primarily relevant for the OR, because its semantic has not been described consistently. We exemplarily show the translation of Figure2:a) containing the *OR join*. The EPC as well as the Petri net have the semantics

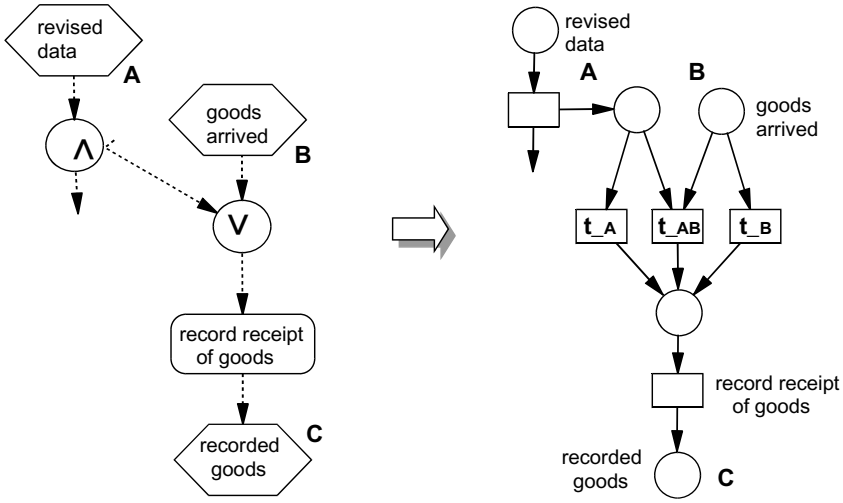


Figure 4: Transformation of the OR-Connector

that C can be reached if either A or B or both occur. In the EPC, all these different cases are described through one connector whereas in the net-pattern all possibilities are modeled explicitly via the transitions t_A, t_{AB} and t_B. The behavior of the EPC and the Petri net are equivalent, because both accept the same executions. Note that the case that C is reached twice if A and B occur sequentially has not been excluded.

After mapping the EPC-elements to net elements or net-pattern, the different parts are combined to form a complex process model. If the derived Petri net has more than one input and/or output place, one further step becomes necessary. In that case, the net is enhanced, such that a WF net is obtained. The transformation is well-defined. Applying the proposed rules each EPC is assigned to exactly one WF net. The transformation does not need any user interaction and was implemented within a prototypical framework developed at the Technical University Berlin. Here, an EPC -description based on the XML is transformed into a PNML-file. PNML is the XML format developed for Petri nets. PNML is an input format which is already accepted by various existing Petri net tools, such as Woflan or LoLA. Figure 5 shows the application of the rules to the EPC from Figure 3.

Petri nets do have an operational semantic. A Petri net model also describes ”how” an execution is reached. Where an EPC only describes a set of accepted executions, a Petri net describes all possible behavior. This difference was neglected in previous attempts of mapping EPCs to Petri nets. As a result, all EPCs have been considered malicious if the corresponding Petri nets were deficient, cf. [Aal99, LSW98]. As a consequence, the modeling facilities of EPCs have often been restricted, such that only correct Petri nets have been gained through the transformation. Referring to the examples from Figure 3 and Figure 2, previous approaches would reject the process specification because of deficiencies

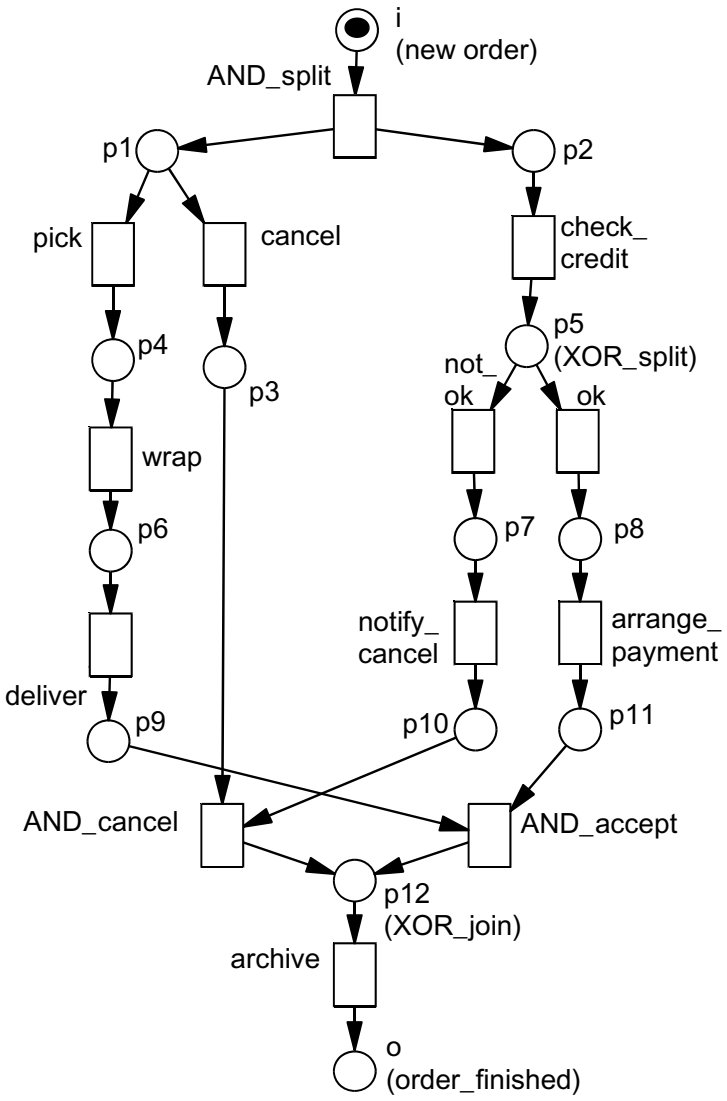


Figure 5: WF net: ”Handling of incoming order”

of the corresponding Petri net, c.f. Figure 5. In the following, we will see that the Petri net is vulnerable to deadlocks.

3rd Step: Correctness Check and Feedback

As WF nets are a special type of Petri nets, there exist a lot of analysis techniques that can be applied. Beside the check of reasonable correctness criteria, such as boundedness we propose to check the resulting net for relaxed soundness as introduced in [DDGJ01]. Relaxed soundness is an alleviated criteria which has been derived from Soundness [Aal98]. It only checks for some minimum requirements the resulting WF net should satisfy. Relaxed soundness only requires to find enough sound firing sequences, where enough means at least so many that each transition is contained in one of them. If the WF net is not relaxed sound, it is not sound either, as soundness implies relaxed soundness. In the context of modeling and transforming EPCs, relaxed soundness is more pragmatic than soundness. Relaxed Soundness provides a measure for good EPCs, as the corresponding WF net is checked to cover some reasonable behavior.

The results from the correctness check of the WF net can directly be transferred to the primary model. If the result of the relaxed soundness check is positive, we can conclude that the EPC represents reasonable behavior. If the result is negative, the modeler gets a list of transitions which are not contained in any sound firing sequence. According to the proposed transformation the deficient transitions either corresponds to a task or to a connector within the EPC. This means that either the task or one of the possible choices described by a connector are not included in an execution that terminates properly. It can be concluded that the corresponding part in the EPC needs improvement. This way, a precise feedback is provided which will helps the modeler to improve the process description until the corresponding WF net fits the property.

The check for relaxed soundness has been automated. The criterion can be checked with the help of existing Petri net tools, such as LoLA [Sch99] (Low Level Petri Net Analyzer) or Woflan [VA00].

Running Example

The resulting WF net is relaxed sound. A set of sound firing sequences which contains all transitions is:

1. $AND\_split, pick, wrap, check\_credit, ok, deliver, arrange\_payment, AND\_accept, archive$ and
2. $AND\_split, check\_credit, not\_ok, notify\_cancel, cancel, AND\_cancel, archive$.

The resulting WF net is not sound as there exist firing sequences that deadlock, e.g.:

- $AND\_split, pick, wrap, check\_credit, not\_ok, deliver, notify\_cancel.$

WF nets Containing OR-Subnets

If the primary EPC contained an OR-connector it is transformed as described in Figure 4. The resulting WF net will only be relaxed sound if all three transitions (t_A, t_{AB} and t_B) are contained in some sound firing sequence.

In most cases, the OR-connector was used to express the combination of one AND-connector (t_{AB}) and one XOR-connector (e.g.t_B), the second XOR-connector (t_A) often does not express a desired choice. However, the relaxed soundness check will detect the failure prone alternative and disclose it to the modeler. The feedback could be enhanced proposing to replace the OR connector through e.g. an AND connector and a XOR connector. After that replacement, the translation of the revised EPC will be relaxed sound.

With the second step, modeling and analysis is finished. The results are an EPC describing reasonable behavior and a relaxed sound WF net. But Workflow management comprehends more than modeling and analysis. Its final objective is the execution support of the process at run-time. In order to arrive at this destination, it is necessary to generate a process description that can be used as input format for a workflow management system guaranteeing a reliable execution at run-time.

The generation of such a process description is the objective of the remaining steps within the proposed process model.

4th Step: Sound Model Construction

For the process description which is used as base for the execution support, we again suggest the use of Petri nets. Properties that advice this choice are their precise formal foundation in combination with their operational semantic. Note, that it is not possible to just use the derived relaxed sound WF net as input format for the workflow management system. Relaxed soundness only states that at least all intended behavior has been described correctly, but it does not guarantee that malicious executions do not exist, as the resulting WF net must not be sound.

As the workflow management system shall guarantee a reliable execution, it must operate on a sound model. In the next step of the proposed procedure we therefore concern the construction of a sound model starting from the relaxed sound specification that has been reached so far. The required model is an extension of the relaxed sound WF net restricting its behavior to sound firing sequences only. This is done by introducing, what we call, synchronization pattern. Through their incorporation a certain scheduling strategy becomes implemented. Thereby the efficiency of the execution becomes determined.

Synchronization pattern refer to additional places that restrict the behavior according to a desired strategy. For its computation we partially refer to the results of Petri net con-

troller synthesis [Giu96, CDLX02, GRX02], where according algorithms were proposed. The used algorithm works on the reachability graph of the WF net. It first computes the maximal permissive behavior. At this, it determines the set of forbidden state transitions[1]. After that, it computes additional places, which disable the forbidden state transitions. The first part of the algorithm was provided in [Deh02b], whereas the second part has been implemented in the tool Synet[2] [Cai97].

Decide Upon a Strategy

Possible are optimistic and pessimistic strategies. Following a pessimistic scheduling strategy, deadlocks are avoided by waiting for decisions to be taken in advance. This would be implemented synchronizing both processes at the decision points. Figure 6 shows a sound model implementing a pessimistic scheduling strategy for the running example. The process model has been enhanced by two places *S1, S2* and corresponding arcs. Through the introduced pattern the two transition *ok* and *pick*, and transition *not_ok* and *cancel* become synchronized. Using this sound model as input for the workflow management system the execution of the process becomes serialized. The customer check would always been executed before the ordering. Here all sound but parallel firing sequences of the relaxed sound WF net have been eliminated.

In case the delivery process takes a long time and the customer check takes a long time, this would be very inefficient. This is especially undesirable if it is very rare that a customer check results in a *not_ok*. Then this way of pessimistic scheduling would be annoying. It would be more efficient to just start the delivery of the order to the customer hoping the customer check will be *ok*, i.e. following an optimistic approach. Only, in the rare case that the decision *not_ok* was taken, the order should be returned to stock and should be canceled after all.

To build an optimistic sound model an additional intermediate step: the specification of the recovery behavior, becomes necessary. This step can not be provided automatically, as it requires background knowledge about the process and its context. The domain expert has to determine: from which states recovery is possible and not too expensive, what has to be done for the recovery and which are the points where the process can start again. These information can be gained from long-term observation of the process or by simulation of the relaxed sound process model. In the latter case, the model should be enhanced by some probability distributions at the decisions points and appraisals of activity duration and -costs.

As results of this investigation, the modeler enhances the EPC model incorporating the recovery behavior. The enhanced model will be used then to build the sound model implementing an optimistic scheduling strategy.

For the running example, we assume that tasks *pick* and *wrap* can be reset without extraor-

[1] State transition which do not lead to state ”o”

[2] Synet: A Synthesizer of Distributable Bounded Petri-Nets from Finite Automata, Version 2.0b, http://www.irisa.fr/s4/tools/synet

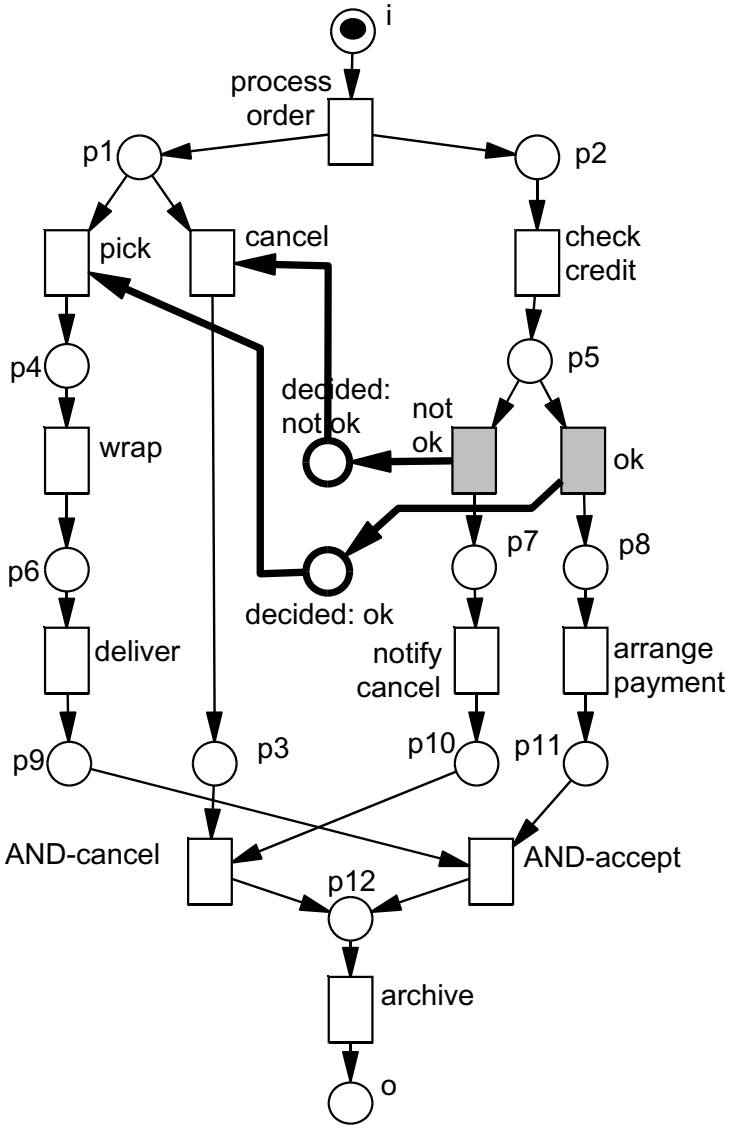


Figure 6: An implementation of a pessimistic scheduling strategy

dinary charges, whereas task *deliver* is considered to be nonreversible. The compensation tasks are *return* and *unwrap*. After the item has been returned to stock, the instance should be canceled.

The enhanced EPC model incorporating the recovery behavior as well as the corresponding WF net are shown in Figure 7.

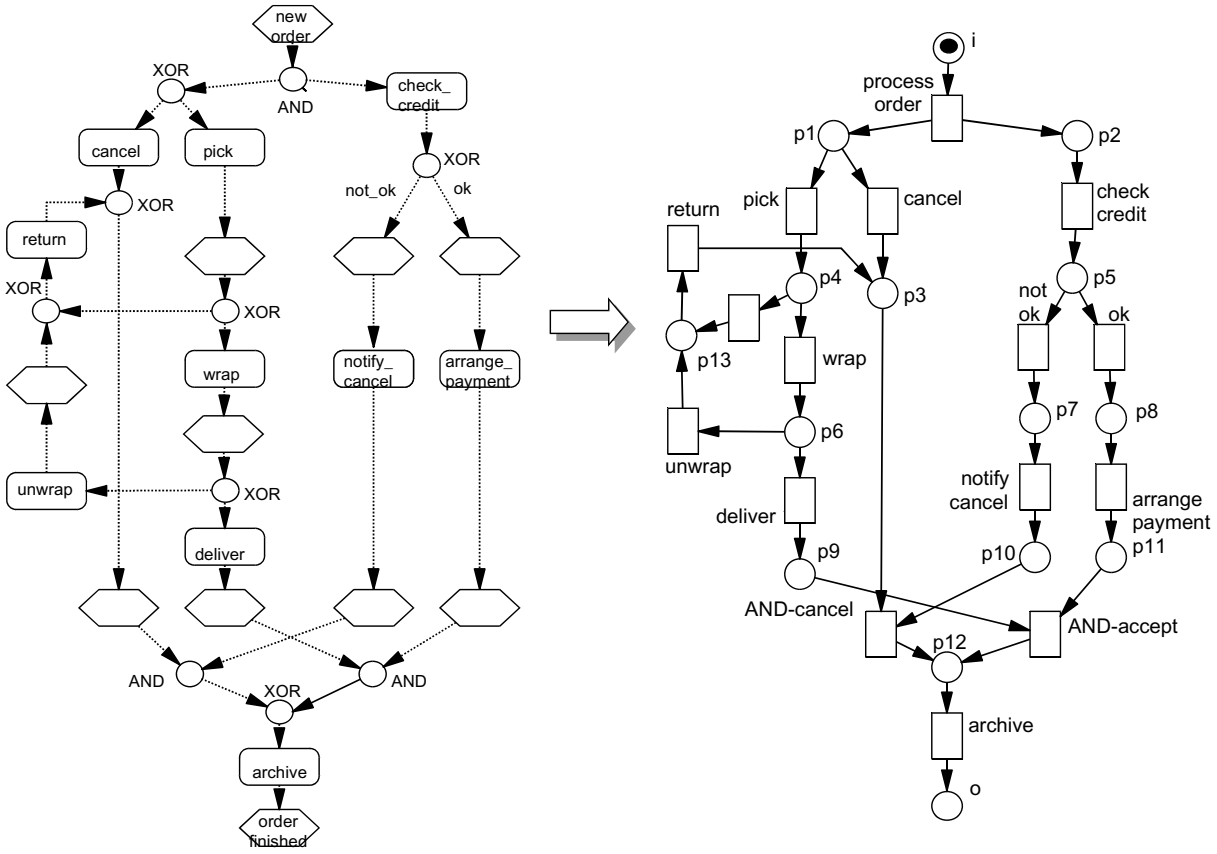


Figure 7: The enhanced Process models incorporating possible recovery behavior

The implementation of the optimistic scheduling strategy under these assumptions is shown in Figure 8.

Using this optimistic sound model as input for the workflow management system, the two departments can operate in parallel. The customer check of the accountancy can be executed concurrently to preparative tasks of the sales handling. Here, all sound firing sequences of the relaxed sound WF net are maintained. As consequence, some additional and not so effective executions are accepted as well.

Both models, the pessimistic as well as the optimistic, are sound. There exist no firing sequences that deadlock or do not terminate properly. Furthermore there exists no dead task. Using either of these models as base fot the internal process description of a workflow management system a reliable process execution at run-time can be guaranteed.

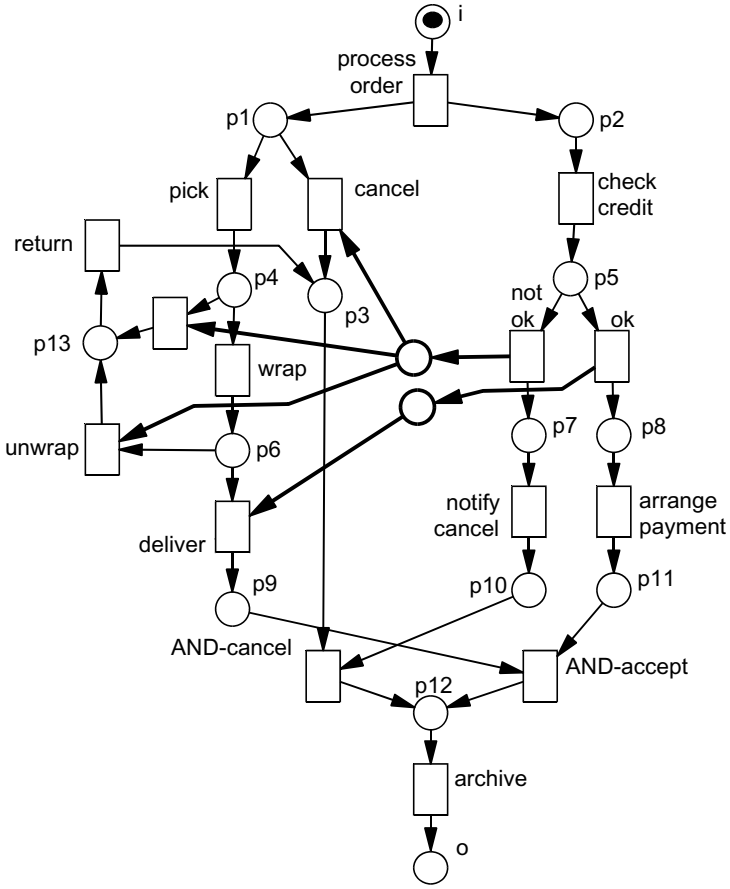


Figure 8: An implementation of an optimistic scheduling strategy

Feedback for the EPC Experts

The computation and integration of the synchronization patterns works on the base of a relaxed sound WF net. Once, the synchronization pattern have been computed and added to the WF net, it would be desirable to give the EPC expert feedback about the introduced changes. Therefore there should be a translation from WF nets back to EPCs. This backward translation is an issue of current research. It is clear, that the rules provided for the EPC→PN direction cannot simply be applied in reverse order. Difficulties arise with respect to the demarcation of net-pattern that correspond to connectors, as well as the translation of choices that are not free-choice. Their translation is not bijective.

Coming back to the process model, some adoptions remain in order to feed the resulting sound process description to a workflow management system. The last step on the way towards the final process description, which we will call *control model*, concerns the incorporation of requirements imposed through the reactive character of a workflow management system.

5th Step: Control Model Construction

The workflow management system is a reactive system, thus it runs in parallel with its environment and tries to enforce certain desirable effects in the environment. To be more precise the workflow management system monitors activities performed by actors (people or software). Monitoring comprises activating tasks by assigning them to certain actors and waiting for the tasks to complete (completion event). Which tasks are enabled at a certain point in time depends on the workflow model describing the process perspective.

WF nets which have been used so far have an active behavior. This is introduced through the following three facts:

1. Tasks are modeled through transitions, which are the active parts within the Petri nets semantic.
2. Transitions fire instantaneous. This does not match with the requirement to model tasks performed by external actors as time consuming entities.
3. The MAY-firing rule, which introduces unintended non-determinism leaving open either to execute an enabled task or to defer its execution.

In the last step the WF net is transformed into a *reactive* WF net, which is a Petri net that can serve as input format for a workflow management system, specifying what the workflow engine should do. Therefore, transitions modeling tasks get refined, and the May-firing is changed towards Must-firing in the corresponding cases. The changes do not compromise the validity of the proven properties. Finally, a sound model with a reactive

behavior is obtained. This process description can be used as input format for a workflow management system and will guarantee reliable execution at run-time.

Benefit of the Proposed Procedure

In this paper, a comprehensive process model for the management of workflow was sketched. The benefit of the proposed approach is obvious. It supports the management of business processes through all its phases, as modeling, analysis and execution, bridging the gap between low modeling expertise on the modeler side and high requirements on the system design side.

The procedure starts with the modeling of processes using the widely accepted EPCs. In contrary to previous approaches it supports the whole spectrum of control-flow elements. Particularly, the use of the OR-connector is not forbidden although ambiguities may become introduced. It, furthermore, does not restrict the modeling to well-formed EPCs only. Following the EPC semantic, the modeler is not required to think about all possible behavior but only has to specify the accepted executions. Thereby, the modeler is relieved from thinking about efficiency aspects during the first phases of the development process.

The correctness means that are provided during the analysis phase have been adapted to these prerequisites. Furthermore, the feedback the application developer gets from the tests is precise. It exactly points at these parts of the model where deficiencies, that occur during the execution, stem from. The requirement posed through the modeling are not very strict. This becomes well-balanced again through the provided execution support. In the last steps, the present model, which possibly also describes undesired executions is enhanced automatically to a sound model. Only here, efficiency aspects get introduced. To postpone the efficiency aspect determination to the end of the procedure is especially desirable as corresponding informations, e.g. the occurrence probability of a certain failure, costs of failure compensation, or priorities often only get available or even change during run-time. Their late incorporation extends the possibilities to reuse modeling results under changing priorities.

The resulting description can be used as base for the input format of a Workflow management system. Giving it a reactive semantic the workflow engine can operate on the derived process description, guaranteeing a reliable and prompt execution at run-time.

References

[Aal98] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[Aal99] W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.

[Cai97] B. Caillaud. Synet : A Tool for the Synthesis of Bounded Petri-Nets, Applications.

Technical Report RR-3155, Inria, Institut National de Recherche en Informatique et en Automatique, 1997.

[CDLX02] B. Caillaud, P. Darondeau, L. Lavagno, and X. Xie, editors. *Synthesis and Control of Discrete Event Systems*. Kluwer Academic Press, 2002.

[DDGJ01] W. Derks, J. Dehnert, P. Grefen, and W. Jonker. Customized Atomicity Specification for Transactional Workflow. In H. Lu and S. Spaccapietra, editors, *Proceedings of the Third International Symposium on Cooperative Database Systems and Applications (CODAS'01)*, pages 155–164. IEEE Computer Society, April 2001.

[Deh02a] J. Dehnert. Four Steps Towards Sound Business Process Model. In G. Rozenberg and H Ehrig, editors, *PNT-Volume*, LNCS, Advances in Petri Nets. Springer Verlag, 2002. to appear.

[Deh02b] J. Dehnert. Non-Controllable Choice Robustness: Expressing the Controllability of Workflow Processes. In J. Esparza and C. Lakos, editors, *ICATPN 2002, 23rd Int. Conf. on Application and Theory of Petri Nets*, volume 2360 of *LNCS*, pages 121–141. Springer Verlag, 2002.

[DR01] J. Dehnert and P. Rittgen. Relaxed Soundness of Business Processes. In K.L. Dittrich, A. Geppert, and M.C. Norrie, editors, *Advanced Information System Engineering, CAISE 2001*, volume 2068 of *LNCS*, pages 157–170. Springer Verlag, 2001.

[Giu96] A. Giua. Petri Net Techniques for Supervisory Control of Discrete Event Systems. In *First Int. Work. on Manufacturing and Petri Nets*, pages 1–3, June 1996.

[GRX02] A. Ghaffari, N. Rezg, and X. Xie. Net Transformation and Theory of Regions for Optimal Control of Petri Nets. In *15th Triennial World Congress of the International Federation of Automatic Control*, Barcelona, Spain, 2002.

[JBS97] S. Jablonski, M. Böhm, and W. Schulze. *Workflow-Management; Entwicklung von Anwendungen und Systemen*. dpunkt.verlag, Heidelberg, 1997.

[KNS92] G. Keller, M. Nüttgens, and A. W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

[LR99] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice Hall PTR (ECS Professional), Upper Saddle River, 1999.

[LSW98] P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event driven Process Chains. In J. Desel and M. Silva, editors, *Application and Theory of Petri nets*, volume 1420 of *LNCS*, pages 286–305. Springer Verlag, Berlin, 1998.

[LSW00] P. Langner, C. Schneider, and J. Wehler. In *Business Process Mangement: Models, Techniques, and Empirical Studie*, volume 1806 of *LNCS*, pages 161–183. Springer Verlag, 2000.

[Mar00] C. Marshall. *Enterprise Modeling With UML: Designing Successfull Software Through Business Analysis*. Addison Wesley, Reading, Massachusetts, 2000.

[MR00] D. Moldt and J. Rodenhagen. Ereignisgesteuerte Prozessketten und Petrinetze zur Modellierung von Workflows. In *Visuelle Verhaltensmodellierung verteilter und nebenläufiger Software-Systeme*, volume 24/00-I of *Fachberichte Informatik*, pages 57–63, 2000.

[NR02] M. Nüttgens and F. J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In J. Desel and M. Weske, editors, *Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen*, LN in Informatics. GI-Edition, 2002.

[Rum99] F. J. Rump. *Geschäftsprozessmanagement auf der Basis ereignisgesteuerter Prozessketten. Formalisierung, Analyse und Ausführung von EPKs (in German)*. Teubner, Stuttgart, 1999.

[Sch94] A. W. Scheer. *Business Process Engineering, ARIS-Navigator for Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.

[Sch99] K. Schmidt. LoLA: A Low Level Analyser. In *Proc. Int. Conf. Application and Theory of Petri net*, volume 1825 of *LNCS*, pages 465–474, 1999.

[VA00] H.M.W. Verbeek and W.M.P. van der Aalst. Woflan 2.0: A Petri-net-based Workflow Diagnosis Tool. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets*, volume 1825 of *LNCS*, pages 475–484. Springer, 2000.

Appendix: The Transformation EPCs→ WF nets

The transformation of EPCs into Petri nets takes place in three steps. First, the elements of the EPC are mapped onto Petri net-modules. In the second step, the derived modules are combined to form a complex process model. The modules are either connected by fusion of arcs or through the unification of nodes. During the last step, the derived net is supplemented to satisfy the WF net syntax, namely the regimentation to exactly one start and one sink place. The last step is only required if the EPC has more than one start or end events. In this case a new start place and/or a new sink place are added and connected to the Petri net-modules which initialize (clean up) the places representing the start and end events of the EPC. The connection is done with the help of connectors that complement the first (last) connector on the paths from the start (end) events.

The first step of the proposed transformation is shown in Figure 9 whereas Figure 10 illustrates the gluing of the individual Petri net modules.

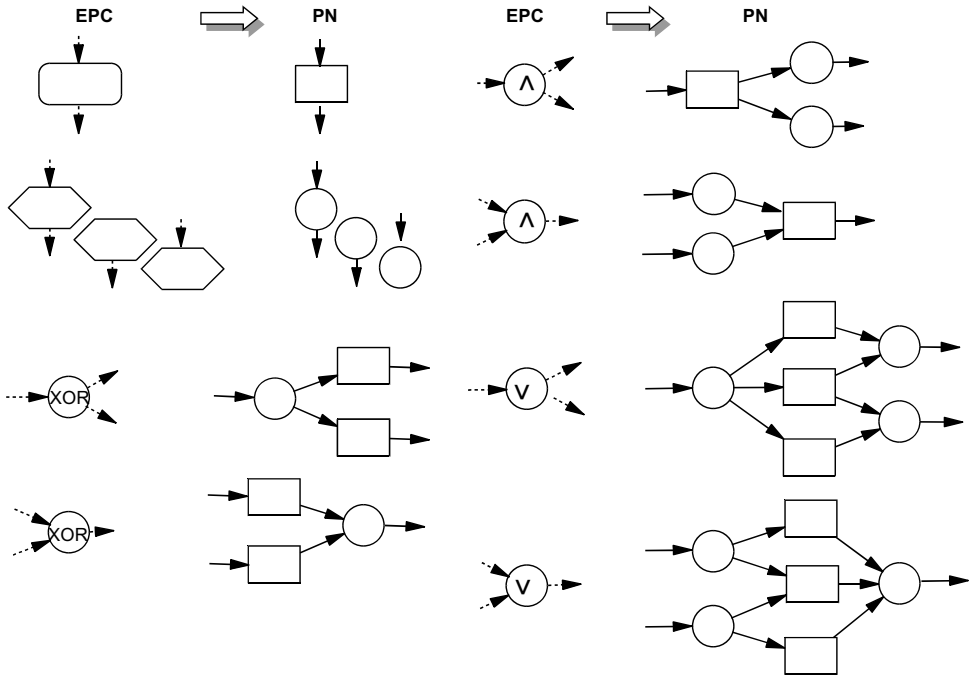


Figure 9: Mapping of EPC-elements to PN-pattern

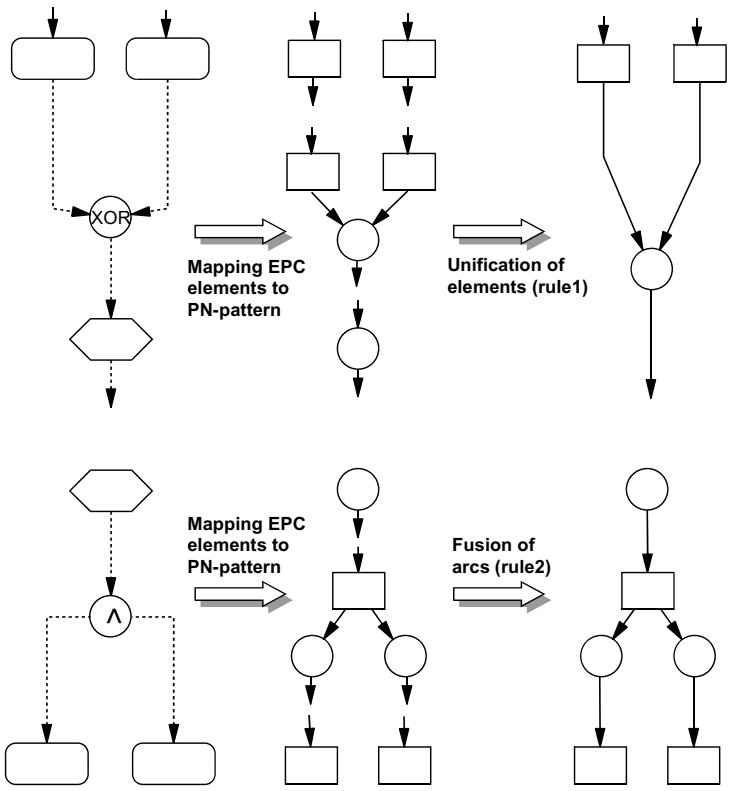


Figure 10: Assembling the Net Modules

On the semantics of EPCs: A vicious circle

Wil van der Aalst
Eindhoven University of Technology
w.m.p.v.d.aalst@tm.tue.nl

Jörg Desel
Katholische Universität Eichstätt-Ingolstadt
joerg.desel@ku-eichstaett.de

Ekkart Kindler
Universität Paderborn
kindler@upb.de

Abstract: Recently, Nüttgens and Rump proposed a formal semantics for Event driven Process Chains (EPCs), which should be fully compliant with the informal semantics of EPCs. But, their semantics has a severe flaw. This flaw reveals that there is a fundamental problem with the informal semantics of EPCs. Here, we pin-point the cause of this problem, we show that there is no sound formal semantics for EPCs that is fully compliant with the informal semantics, and we discuss some consequences.

1 Introduction

About ten years ago, *Event driven Process Chains* (*EPCs*) have been introduced for modelling business processes [KNS92]. Ever since, there have been different proposals of formal semantics for EPCs. Most of these semantics, however, consider only a fragment of EPCs or do not fully comply with the informal semantics of EPCs. Recently, Nüttgens and Rump proposed a formal semantics for EPCs [Rum99, NR02], and they claimed that the deficiencies of earlier formalizations have now been overcome.

The semantics of Nüttgens and Rump, however, has several insufficiencies, from a technical point of view, from a conceptual point of view, as well as from a practical point of view. These insufficiencies will be discussed in this paper. Moreover, we will show that these insufficiencies are inherent to the informal semantics of EPCs, to the effect that there cannot be a formal semantics of EPCs that is fully compliant with the informal semantics. Therefore, any formal semantics for EPCs will impose some restrictions on EPCs or will deviate form the informal semantics to some extend. Which restrictions or deviations are adequate, is a matter to be discussed. In this paper, we provide some input to such a discussion.

2 The informal semantics of EPCs

We start with a brief discussion of the informal semantics of EPCs, where we focus on one speciality of the semantics of EPCs, which we call *non-locality*. Figure 1 shows a simple example of an EPC. The dynamic behaviour of the EPC is best illustrated by *process folders*, which are propagated between the different nodes of the EPC along its control flow arcs. The *connectors*, which are represented as circles, may join and split process folders. This way, the connectors define the routing and the synchronization of process folders. For our example, we assume that, initially, there is one process folder on each of the two events Start1 and Start2.

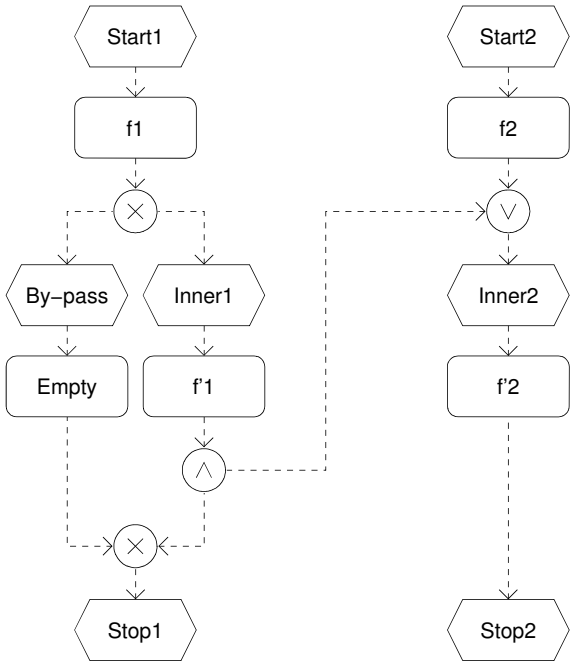


Figure 1: A simple EPC

First, we discuss what happens to the process folder on Start1: This process folder is passed to function f1. At the XOR-split connector below f1, the process folder is either propagated to the By-pass event or to the Inner1 event. If the process folder is propagated to the By-pass event, it is further propagated to the Empty function, and then passed on to the Stop1 event via the XOR-join connector. If the folder is passed to the Inner1 event, it is further propagated to the function f'1 and then reaches the AND-split connector. At the AND-split the process folder is duplicated. The two copies are propagated via the two outgoing arcs. One process folder is propagated to the XOR-join, the other is propagated to the OR-join on the right-hand side.

Second, we discuss what happens to the process folder on Start2: This process folder is propagated to function f2. What happens at the OR-join below function f2 depends on the behaviour of the left-hand part of the EPC. If there is the possibility that a process folder will arrive from the left-hand part, the OR-join delays the propagation of the process

folder. In our example, this is the case as long as there is a process folder on Start1, f1, Inner1, or f'1. If no process folder can arrive from the left-hand part anymore, the OR-join propagates the folder from f2 to Inner2. In our example, this is the case, once the process folder has by-passed the AND-split. If eventually a process folder from the left-hand part arrives at the OR-join, the process folders from both incoming arcs are merged and a single process folder is propagated to the Inner2 event. From the Inner2 event, the process folder is propagated down to f'2 and Stop2.

In the following, we will focus on the semantics of the OR-join connector. Its main characteristics is the delay of the propagation of a process folder on one of its incoming arcs as long as a process folder could possibly arrive at one of the other incoming arcs. Thus, the semantics of the OR-join is *non-local*[1]. In order to decide whether a process folder at an OR-join can be immediately propagated or whether the propagation should be delayed, we need to know whether there will be a process folder arriving at one of the other arcs at some time in the future. This information is not local to the OR-join, but may depend on all other parts of the EPC. In our example, it depends on the choice taken at the XOR-split in the left-hand part of the EPC.

3 A technical problem

In principle, there is no problem with the non-locality of the informal semantics of EPCs. But, a formalization requires some care, because the definition of such a semantics in some way refers to itself again: The semantics to be defined (propagation of process folders) occurs in its own definition (propagation of process folders to an incoming arc of an OR-join). This may easily result in a cyclic definition without any meaning. And this is what happened in the definition of the semantics in [Rum99, NR02]: There, a state is the assignment of process folders to the different nodes of the EPCs, and a *transition relation* $_{TR}$ between these states defines the state changes. Unfortunately, the transition relation $_{TR}$ occurs in its own definition again.

One way out of this cyclic definition would be to consider the definition as a fixed-point equation. Then, $_{TR}$ would be the least fixed-point of this equation. Unfortunately, the relation $_{TR}$ occurs under a negation in its definition, to the effect that the fixed-point does not exist in all cases[2]. Fortunately, fixed-points seem to exist in all practically relevant cases. In particular, a fixed-point exists for the example given in the next section.

[1] Here, we focus on the non-locality of the OR-join connector. According to Nüttgens and Rump [NR02], the XOR-join connector has a non-local semantics, too. But we do not deal with the XOR-join in the rest of the paper.

[2] For the experts, the Appendix shows an example of an EPC, for which a recursive interpretation of the definition of $_{TR}$ runs into an infinite cycle.

4 A conceptual problem

Figure 2 shows another EPC[3] with two OR-joins in a feedback loop, which is a vicious circle, as we will see. With the above mentioned fixed-point interpretation, the semantics of [NR02] is that the process folders are stuck at f1 and f2. The two OR-joins will not propagate the process folders to the Inner events.

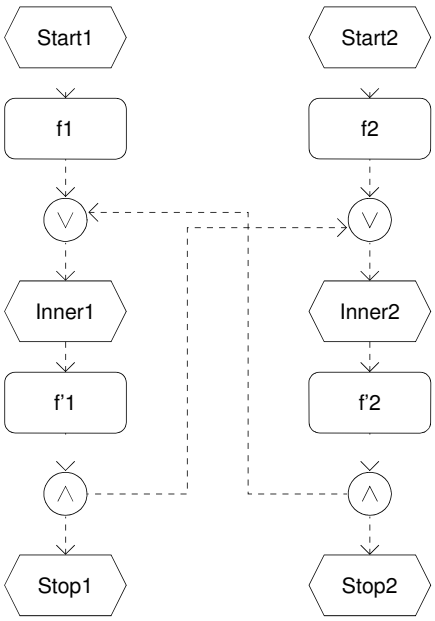


Figure 2: A vicious circle

Is this the intended semantics of this EPC? We will argue that it is not. To this end, we consider the OR-join above the Inner1 event. Since the Inner2 event will never occur, we know that no process folder will ever arrive at the other incoming arc of the OR-join. So, according to the informal semantics, the OR-join should propagate the process folder from f1 to the event Inner1. Symmetrically, we can argue that the process folder from f2 should be propagated to Inner2. So, we have shown that the process folders should not be delayed at f1 and f2 according to the informal semantics of EPCs.

Is this the intended semantics of this EPCs? Again, we will argue that it is not. We will argue that the OR-joins should not propagate the process folders from f1 and f2. To this end, we consider the OR-join before the Inner1 event again. Since Inner2 will eventually occur, we know that eventually there will be a process folder arriving at the second incoming arc. According to the informal semantics, this implies that the OR-join should wait with the propagation of the process folder until the second folder arrives. Symmetrically, we can argue that the process folder from f2 should not be propagated. So, we know that the process folders should be delayed at f1 and f2 according to the informal semantics of EPCs.

[3] Rump [Rum99] gives a similar example. But his point is that, in some situations, OR-joins may result in a deadlock. Here, we argue that the situation is much worse: the intuitive semantics of EPCs fails.

Altogether, we have established a vicious circle. In a situation with process folders on f1 and f2, we do not know whether the OR-joins should propagate the process folders or not. Both alternatives result in a contradiction to the informal semantics of EPCs. This shows that there is no formal semantics that precisely captures the informal semantics of EPCs.

Every formalization will deviate from the informal semantics in some way or the other. Either it will impose some syntactical restrictions on EPCs, or it will deviate from the informal semantics in some way. We believe that this choice depends on the purpose of the semantics.

5 A practical problem

In the previous sections, we have shown that the non-locality of the semantics of EPCs results in some technical and conceptual problems. But, non-locality is problematic from a practical point of view, too. The reason is that, for routing the process folders through an EPC, one needs to know the complete EPC, since process folders could arrive at an OR-join from far away parts of the EPC. This is acceptable for workflows of a single organization[4]. For inter-organizational workflows, however, this is not acceptable anymore, because different parts of the workflow belong to different organizations, which will not reveal their part to the others. Due to the non-locality of the semantics of EPCs, this would be necessary. Another solution to this problem would be a mechanism for a check-back with the other organizations whether a process folder will be arriving from them in the future. This does not not appear to be a realistic situation. Even worse, there could be a vicious cycle of check-backs.

6 Pragmatic solutions

Though the non-locality of the OR-join has its complications, it is a pattern frequently occurring in a wide variety of workflow processes. Therefore, dealing with this problem is a practical issue, and vendors of workflow management systems have found ways to deal with it. Most of the systems require the workflow designer to replace OR-splits and OR-joins by AND/XOR-splits and AND/XOR-joins. However, there are also systems that directly offer constructs that resemble the OR-join as discussed in this paper. For an overview of these systems we refer to [AHKB02, Kie02] where 15 systems are evaluated on the basis of 20 workflow patterns. Pattern 7, also referred to as the *synchronizing merge*, corresponds to the OR-join with a non-local semantics. The synchronization merge is fully or partially supported by InConcert, Eastman, Domino Workflow, eProcess, and MQSeries Workflow. In these systems, the designer does not have to specify the type of the join; this is automatically handled by the system. Here, we will briefly discuss how these systems support the synchronizing merge, i. e. the OR-join with a non-local semantics.

[4] Actually, it is only acceptable if the routing is performed by a workflow management system. Nobody wants to calculate the complete semantics of an EPC by hand just for routing the process folders.

InConcert (TIBCO, [Tib00]) only allows for acyclic workflow graphs and does not allow for XOR-splits. The only way to model alternatives is through adding conditions to tasks, i. e., a task is simply skipped when the condition evaluates to *false*. This way the problem of the OR-join is avoided and every join becomes an AND-join. This solution seems to be very limiting but is motivated by the fact that InConcert allows for ad-hoc workflows. End-users can make changes on-the-fly. Therefore, the language has to be very simple and it should be impossible to design a workflow with deadlocks or livelocks.

Eastman (Eastman, [Sof98]) directly supports the synchronizing merge in most cases. There are no syntactical requirements like the absence of cycles. However, in certain circumstances the OR-join does not behave as expected. It appears that, in the system, there are some heuristics to determine whether to wait or not. In most practical cases these heuristics work. However, for more involved situations the heuristics fail in the sense that the OR-join progresses while there is still some input on its way.

Domino Workflow (Lotus/IBM, [NEG 00]) also supports the synchronizing merge without imposing syntactical requirements. It is not clear how the engine detects whether it should wait or not. For all practical examples, the solution seems to work satisfactory. However, Domino Workflow has not been tested using nasty examples like the "vicious circle".

eProcess (FileNET, [Fil02]) supports the synchronizing merge, but requires a one-to-one correspondence between AND/OR-splits and AND/OR-joins, and every path starting in an AND/OR-split should lead to the corresponding AND/OR-join. Then, the OR-join can be implemented by a counter which is set by the split and inspected and decreased by the join. After executing the split, the counter is set to the number of parallel threads[5] initiated. Then, the counter is decreased by one for every thread reaching the join. The moment the counter is set to zero, processing continues. It is interesting to see that the simple syntactical restriction to a one-to-one correspondence between splits and joins makes the semantics of the OR-join local.

MQSeries Workflow (IBM, [IBM99]) is one of the most interesting systems when it comes to the OR-join. MQSeries Workflow only allows for the iteration of entire subprocesses, i. e., cycles are not allowed in the workflow graph. Note that this requirement does not imply that there is one-to-one correspondence between splits and joins (like for eProcess). Nevertheless this requirement is sufficient for implementing the synchronizing merge using a local rule. MQSeries workflow uses a technique called "dead-path elimination" [LR99, IBM99]: Initially, each input arc is in state "unevaluated". As long as one of the input arcs is in this state, the activity is not enabled. The state of an input arc is changed to true the moment the preceding activity is executed. However, to avoid deadlocks, the input arc is set to false the moment it becomes clear that it will not fire. By propagating these false signals, deadlocks are excluded and the resulting semantics is that of a synchronizing merge. The solution used in MQSeries workflow is similar to having true and false tokens in Petri nets: true tokens should be executed while false tokens are simply passed on. The idea of having true and false tokens to address complex synchronizations was already reported in [GT84]. However, the bipolar synchronization schemes presented in

[5] In the semantics of EPCs, the threads could be considered as process folders.

[GT84] are primarily aimed at avoiding constructs such as the synchronizing merge, i. e., the nodes are pure AND/XOR-splits/joins and partial synchronization is neither supported nor investigated[6].

The examples listed above show that the issue of non-local OR-joins is addressed by contemporary workflow management systems. Systems like InConcert, eProcess, and MQSeries have solved the OR-join problem using syntactical restrictions. Other systems like Eastman and Domino Workflow seem to use a non-local semantics similar to the one discussed in this paper. Such a non-local semantics may lead to unexpected results. Moreover, a non-local semantics may result in poor performance as is stated in the manual of Eastman: "Parallel instances can accumulate at a Join workstep if the instances are routed to the workstep by preprocessing rules. These instances will eventually be joined by a RouteEngine subprocess (thread) that examines Join worksteps for such instances. This *Join scavenger thread* reduces system efficiency, so routing to Join worksteps using preprocessing rules should be avoided" [Sof98].

7 Conclusion

In this paper, we have discussed a problem of the EPC semantics of Nüttgens and Rump [Rum99, NR02]. Actually, the problem is inherent to the informal semantics of EPCs. It came into focus thanks to Nüttgen's and Rump's attempt to precisely capture the informal semantics. We have shown that this is impossible, and that a fully compliant formal semantics does not exist.

Maybe, there is a sound formal semantics that captures the most important aspects of non-locality of the informal semantics. We see several possibilities to achieve this goal. One way would be the above mentioned fixed-point interpretation; but this would require to syntactically exclude those EPCs for which the fixed-point does not exist. Another way would be to define two transition relations; the first would be completely local, the second would be non-local, but would only use the first for checking whether a folder can arrive at an OR-join. But, all these semantics will be quite complex and will have pitfalls, to the effect that the semantics neither will help to understand EPCs, nor will help to develop analysis methods, nor will provide efficient routing algorithms.

There are several proposals of much simpler semantics for EPCs, which deliberately have chosen to be local – either by excluding the problematic connectors (i. e. the OR-join) or by giving them a local semantics. Clearly, they do not fully capture the informal semantics, but there simplicity is an argument in favour of these semantics.

Before starting another effort in defining the semantics of EPCs, we think that a discussion of the degree of non-locality that a semantics for EPCs should and could have is in order. For this discussion, we need a set of well-accepted practical EPC models of business processes, which cover the typical situations in which OR-joins and XOR-joins occur, along with the intended behaviour.

[6] Note that Langner et al. [LSW98] build on the work in [GT84] to analyze EPCs.

References

[AHKB02] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. QUT Technical report, FIT-TR-2002-02, Queensland University of Technology, Brisbane, 2002. (Accepted for publication in Distributed and Parallel Databases, also see http://www.tm.tue.nl/it/research/patterns.).

[Fil02] FileNET. *Panagon eProcess Designer 4.2.2*. FileNET Corporation, Costa Mesa, CA, USA, 2002.

[GT84] H. J. Genrich and P. S. Thiagarajan. A Theory of Bipolar Synchronization Schemes. *Theoretical Computer Science*, 30(3):241–318, 1984.

[IBM99] IBM. *IBM MQSeries Workflow - Getting Started With Buildtime*. IBM Deutschland Entwicklung GmbH, Boeblingen, Germany, 1999.

[Kie02] B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows (submitted)*. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2002.

[KNS92] G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Heft 89, Universität des Saarlandes, January 1992.

[LR99] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, 1999.

[LSW98] P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event driven Process Chains. In J. Desel and M. Silva, editors, *Application and Theory of Petri Nets 1998*, *LNCS* 1420, pages 286–305. Springer, 1998.

[NEG 00] S.P. Nielsen, C. Easthope, P. Gosselink, K. Gutsze, and J. Roele. *Using Lotus Domino Workflow 2.0, Redbook SG24-5963-00*. IBM, Poughkeepsie, USA, 2000.

[NR02] Markus Nüttgens and Frank J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In *PROMISE 2002, Prozessorientierte Methoden und Werkzeuge fürr die Entwicklung von Informationssystemen*, *GI Lecture Notes in Informatics* P-21, pages 64–77. Gesellschaft für Informatik, 2002.

[Rum99] Frank J. Rump. *Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten*. Teubner-Reihe Wirtschaftsinformatik. B.G.Teubner, 1999.

[Sof98] Eastman Software. *RouteBuilder Tool User's Guide*. Eastman Software, Inc, Billerica, MA, USA, 1998.

[Tib00] Tibco. *TIB/InConcert Process Designer User's Guide*. Tibco Software Inc., Palo Alto, CA, USA, 2000.

Acknowledgment We would like to thank Matthias Gehrke for his comments on an earlier version of this paper.

Appendix

Figure 3 shows a nasty example of an EPC. For this EPC, a recursive interpretation of the definition of TR in [NR02] runs into an infinite cycle, when started with a process folder on Start1 and Start2.

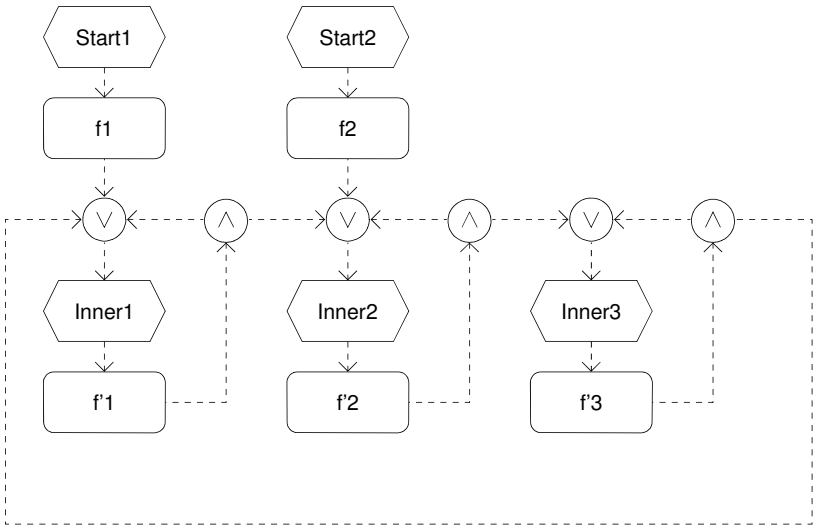


Figure 3: A vicious circle worse than the one from Fig. 2

Eine XML-Notation für Ereignisgesteuerte Prozessketten (EPK)

Marco Geissler, TU Berlin
email: gyvermac@cs.tu-berlin.de

Andreas Krüger, TU Berlin, CIS
email: akrueger@cs.tu-berlin.de

Abstract: Ereignisgesteuerte Prozessketten dienen in der Praxis als gängiges Mittel, um Geschäftsprozesse zu modellieren bzw. zu dokumentieren. Werkzeuge, die auf der Basis von Ereignisgesteuerten Prozessketten arbeiten, sollten eine einheitliche Notation für die EPKs benutzen, damit eine Austauschbarkeit der Modelle zwischen den einzelnen Tools gegeben ist. Als Sprache für eine solche Notation hat sich in der Praxis XML durchgesetzt. Ziel dieses Papiers ist es deshalb, eine XML-Notation für EPKs vorzustellen.

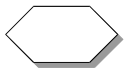
Einleitung

Für die Modellierung sowie für die Analyse und Verarbeitung von EPKs existieren inzwischen eine grosse Zahl von Werkzeugen. Bisher benutzen diese Tools jeweils eigene Methoden/Formate zum Abspeichern bzw. Einlesen der EPKs. In Bezug auf einer für den Anwender freien Kombinierbarkeit eines Modellierungstools mit einem Analyse- oder Verarbeitungstool ist es notwendig, dass eine gemeinsame Schnittstelle zwischen diesen Programmen existiert. In einer immer stärker werdenden komponentenbasierten Entwicklung und Anwendung von Programmen, sind einheitliche Schnittstellen unabdingbar. Aus diesem Grund wird hier für EPKs ein einheitliches Format entwickelt und somit im Bereich der auf EPKs arbeitenden Programme eine einheitliche Schnittstelle geschaffen. Da sich für solche Zwecke XML als Sprache durchgesetzt hat, wird hier eine XML-basierte Notation verwendet.
Die XML-Notation ist im Rahmen der Entwicklung eines Tools entstanden, welches EPKs zur weiteren Analyse nach [DR01] in Workflownetze übersetzt.

1 Überführung der EPK-Symbole in eine XML-Darstellung

Um jedes Symbol für die Modellierung von EPKs in geeigneter Weise in eine XML-Notation zu bringen, werden zuerst die einzelnen Symbole bezüglich ihrer Bedeutung und dargestellten Information betrachtet. Nachfolgend sind die Symbole, ihre Bedeutung nach [KNS92] und die verwendete Notation erklärt.

Allen Elementen wird in der XML-Notation eine eindeutige ID zugewiesen. Bis auf die Konnektoren haben alle Elemente auch den vom Modellierer vergebenen Namen vermerkt.



Ereignis

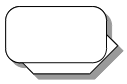
```
<event id="..." name="..." />
```

Für ein Ereignis sind keine weiteren Informationen in der Notation abzuspeichern.

Funktion

```
<function id="..." name="...">
  <control_flow_function>
    <in from="..." /> <out to="..."  />
  </control_flow_function>
  <information_flow>
    <in from="..." /> <out to="..."  /> ...
  </information_flow>
  <ressources>
    <ressource unit="..." /> ...
  </ressources>
</function>
```

Funktionen sind zusätzlich durch einen ein- und ausgehenden Kontrollfluss gekennzeichnet. Desweiteren können Funktionen Informationsflüsse beinhalten und Ressourcen benutzen. Für eingehende und ausgehende Kanten, sowie für Ressourcen werden als Referenz die IDs der Vorgänger- bzw. Nachfolger-Elemente in der Notation vermerkt.



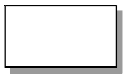
Prozesswegweiser

```
<process_marker id="..." name="..." link="...">
  <control_flow_pm>
    <in from="..." /> bzw. <out to="..."  />
  </control_flow_pm>
</process_marker>
```

Ein Prozesswegweiser hat entweder einen eingehenden oder einen ausgehenden Kontrollfluss und verweist auf einen anderen Prozess.

Organisationseinheit

```
<organisation_unit id="..." name="..." />
```



Informationsobjekt

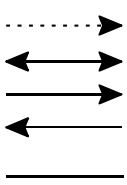
```
<information_object id="..." name="..." />
```

Organisationseinheit und Informationsobjekt benötigen keine zusätzlichen Informationen.

Verknüpfungsoperatoren

```
<connector id="..." relation="..." >
  <control_flow_connector>
    <in from="..." /> <out to="..." /> <out to="..." />
  </control_flow_connector>
</connector>
```

Verknüpfungsoperatoren haben je nach Typ (Split oder Join) eine Eingangskante und mehrere Ausgangskanten oder mehrere Eingangskanten und eine Ausgangskante, die durch die korrespondierenden IDs gekennzeichnet sind. Die Konnektoren können von der Relation OR, AND oder XOR sein.



Kontrollfluss

Informationsfluss

Zuordnung von Ressourcen

Kontroll- und Informationsfluss, sowie die Zuordnung von Ressourcen sind in die Notation von Funktionen und Verknüpfungsoperatoren eingebettet.

Eine Ereignisgesteuerte Prozesskette besteht aus mehreren Prozessen. Die einzelnen Prozesse sind durch Kontroll- und Informationsflüsse, zwischen den oben beschriebenen Elementen charakterisiert.
Die Grobstruktur sieht wie folgt aus:

```
<EPC>
  <process id="PROC1" name="...">
  ...  beliebige Aneinanderreihung der oben beschriebenen
       Elemente
  </process>
  <process id="PROC2" name="..."> ... </process>
  ...
</EPC>
```

Die vollständige DTD ist im Anhang aufgeführt.

2 Demonstration anhand eines Beispiels

In diesem Kapitel wird die Anwendung der XML-Notation anhand eines Beispiels demonstriert. Das Beispiel ist in Abbildung 1 zu sehen und wird auszugsweise in die XML-Notation überführt.

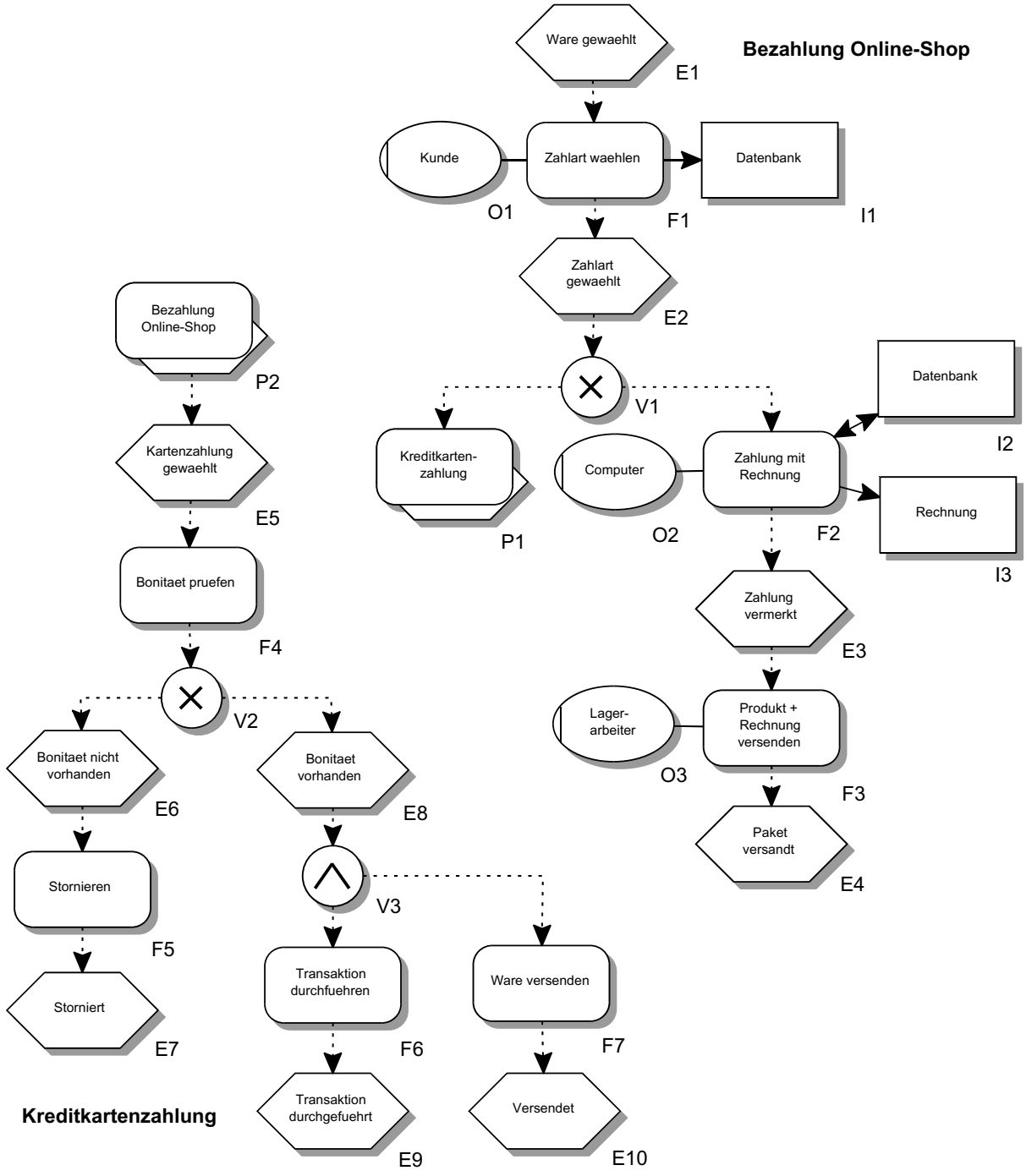


Abbildung 1: EPK-Symbole

In Abbildung 2 wurde exemplarisch ein Teil der Elemente aus dem Prozess *Bezahlung*

Online-Shop in die XML-Notation überführt.

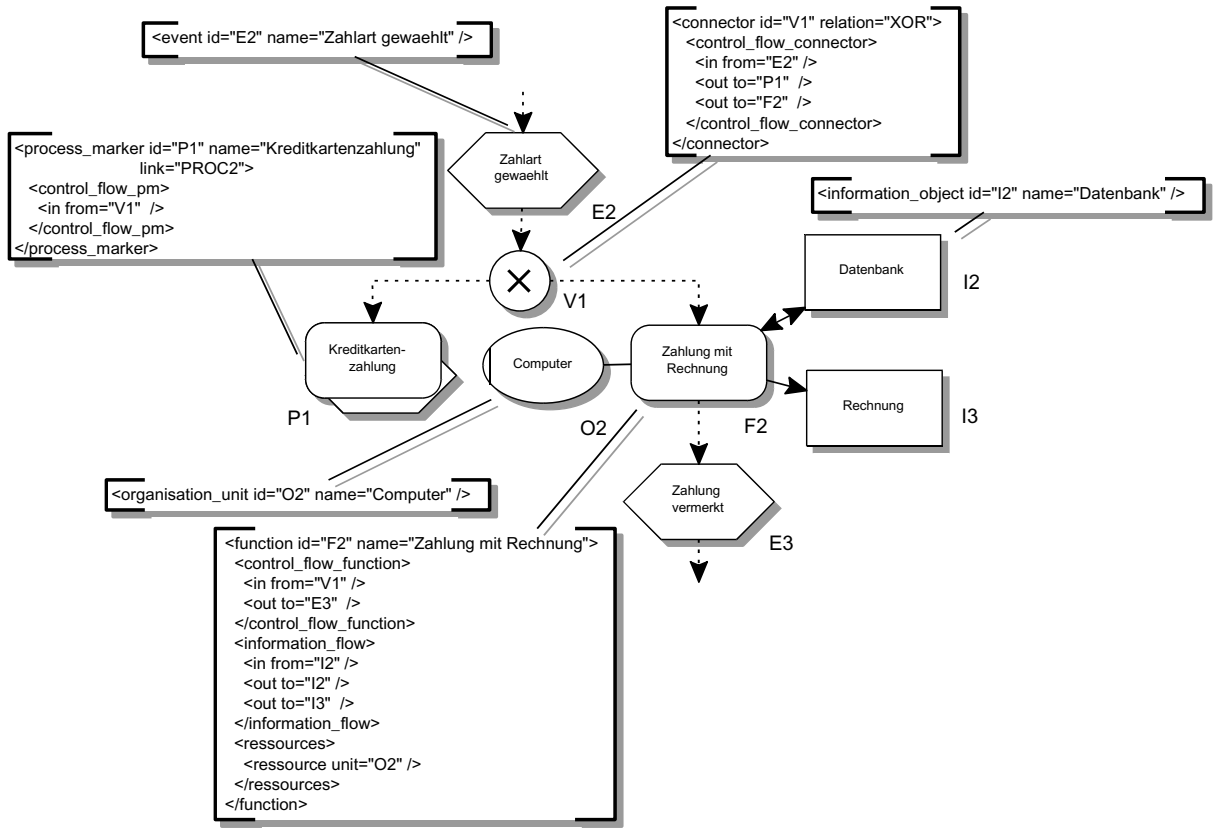


Abbildung 2: EPK-Ausschnitt und XML-Übersetzung

Schlussbemerkung

Mit der beschriebenen XML-Notation ist es möglich, Ereignisgesteuerte Prozessketten vollständig in einem einheitlichen Format darzustellen. Mit der Umsetzung dieser Notation wird Entwicklern die Möglichkeit gegeben, ihre EPK-Modelle ohne Beachtung toolspezifischer Speicherformate mit allen Programmen modellieren, analysieren und verarbeiten zu können. Damit wären bisher hinderliche Einschränkungen dieser Art nicht mehr von Bedeutung.
Weitere Informationen sowie die DTD sind auf http://www.cs.tu-berlin.de/˜akrueger zu finden.

Literatur

[DR01] J. Dehnert and P. Rittgen. Relaxed Soundness of Business Processes. In K.L. Dittrich, A. Geppert, and M.C. Norrie, editors, *Advanced Information System Engineering, CAISE 2001*, volume 2068 of *LNCS*, pages 157–170. Springer Verlag, 2001.

[KNS92] G. Keller, M. Nüttgens, and A. W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

Anhang: Eine DTD für EPKs

```
<!ELEMENT EPC (process)+>
<!ELEMENT process (event | function | connector | organisation_unit |
          information_object | process_marker)*>
<!ELEMENT event EMPTY>
<!ELEMENT function (control_flow_function, information_flow, ressources)>
<!ELEMENT connector (control_flow_connector)>
<!ELEMENT organisation_unit EMPTY>
<!ELEMENT information_object EMPTY>
<!ELEMENT process_marker (control_flow_pm)>
<!ELEMENT control_flow_function (in, out)>
<!ELEMENT control_flow_pm (in | out)>
<!ELEMENT control_flow_connector ((in, out, out+) | (in, in+, out))>
<!ELEMENT information_flow (in*, out*)>
<!ELEMENT ressources (ressource+)>
<!ELEMENT link_to EMPTY>
<!ELEMENT link_from EMPTY>
<!ELEMENT in EMPTY>
<!ELEMENT out EMPTY>
<!ELEMENT ressource EMPTY>
<!ATTLIST EPC
          id ID #REQUIRED
          name CDATA #REQUIRED>
<!ATTLIST process
          id ID #REQUIRED
          name CDATA #REQUIRED>
<!ATTLIST event
          id ID #REQUIRED
          name CDATA #REQUIRED>
<!ATTLIST function
          id ID #REQUIRED
          name CDATA #REQUIRED>
<!ATTLIST connector
          id ID #REQUIRED
          relation (AND | OR | XOR) #REQUIRED
<!ATTLIST organisation_unit
          id ID #REQUIRED
          name CDATA #REQUIRED>
<!ATTLIST information_unit
          id ID #REQUIRED
          name CDATA #REQUIRED>
<!ATTLIST process_marker
          id ID #REQUIRED
          name CDATA #REQUIRED>
          link IDREF #REQUIRED>
<!ATTLIST in from IDREF #REQUIRED>
<!ATTLIST out to IDREF #REQUIRED>
<!ATTLIST ressource unit ID #REQUIRED>
```

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.

Ereignisgesteuerte Prozessketten (EPK) – Multi-Instanziierungsfähigkeit und referentielle Persistenz –

Jörg Rodenhagen

Detecon International GmbH
Competence Practice Organisation
Oberkasseler Str. 2, D-53227 Bonn
E-Mail: Joerg.Rodenhagen@detecon.com

Abstract: Die Ereignisgesteuerte Prozesskette (EPK) hat sich als praxisnahe Technik zur Modellierung von Geschäftsprozessen durchgesetzt. Derartige Prozessmodelle werden zunehmend eingesetzt, um Aussagen zur Effizienz realer Prozesse in komplexen Kontexten zu generieren und dort Optimierungspotentiale zu identifizieren. Dieser Beitrag interpretiert einen Prozess als Lebenszyklus eines zentralen Objektes beliebiger Komplexität innerhalb der festgelegten Systemgrenzen. Die Praxis zeigt, dass sich wesentliche Effizienzhemmnisse oft in detaillierten Subprozessen und Strukturen verbergen, so dass das Aufbrechen des zentralen Objektes in seine Substrukturen sowie eine detaillierte Betrachtung der korrespondierenden Prozesse auf verschiedenen Abstraktionsebenen erforderlich sind. Am Beispiel eines Auftragsbearbeitungsprozesses werden mit der Multiinstanziierungsfähigkeit und der referentiellen Persistenz zwei Systemeigenschaften diskutiert, die von der EPK nicht unterstützt werden und deren Ausdrucksmächtigkeit deutlich einschränken. Damit lässt sich auch die Zuverlässigkeit der aus einem EPK-Modell generierten Analyseergebnisse in Frage stellen. Unter Multiinstanziierungsfähigkeit wird die Eigenschaft verstanden, aus einem Prozess beliebig viele identische Instanzen eines Subprozesses generieren zu können. Die referentielle Persistenz stellt auf die Speicherung der Beziehung zwischen einem Objekt und seinen Substrukturen in einem verteilten System ab. Anhand eines Vergleiches mit Petrinetzen werden Potentiale zur Erweiterung der EPK skizziert.

1 Einleitung

Zur Gestaltung, Analyse und Optimierung von Geschäftsprozessen (GPO) werden zur Bewältigung der heute vorherrschenden Komplexität zunehmend standardisierte und werkzeuggestützte Methoden und Techniken eingesetzt. Eine in diesem Zusammenhang vielfach eingesetzte Technik ist die Ereignisgesteuerte Prozesskette (EPK). Sie ist die zentrale Technik des ARIS Konzeptes [Sc99, Sc01] und des Werkzeuges „ARIS Toolset“ der IDS Scheer AG. In der Praxis findet die EPK ihre Einsatzfelder in der individuellen Prozessspezifikation [EW95], [Ro00], der Einführung des R/3-Systems der SAP AG und in der Dokumentation des R/3-Referenzmodells [KM94].

Beschränkte sich in der Vergangenheit die individuelle Prozessspezifikation noch vorwiegend auf die Visualisierung und Explizitmachung des implizit vorhandenen Wissens über Abläufe im Unternehmen, so werden Prozessmodelle heutzutage in zunehmend anspruchsvolleren Anwendungskontexten eingesetzt, unter anderem zur Effizienzbewertung realer Geschäftsprozesse und zur Identifikation und Lokalisierung von Optimierungspotentialen. Mit der Erweiterung des Einsatzkontextes erhöhen sich zugleich die formalen Anforderungen an die eingesetzte Modellierungstechnik. In zahlreichen Beiträgen konnten wesentliche Mängel in den originären Definitionen der Syntax und Semantik von EPK-Modellen aufgezeigt werden, u.a. durch Langner et al. [LSW97, LSW98], Rump [Ru97, Ru99], van der Aalst [Aa99] und Rodenhagen [Ro97, MR00]. Zur Behebung der erkannten Mängel wurden zahlreiche konstruktive Vorschläge unterbreitet, dennoch kann bis heute nicht von einer vollständigen, ausdrucksstarken und allgemein anerkannten Spezifikation der EPK ausgegangen werden.

Neben der formalen Grundlage stellt sich die Frage nach der Ausdrucksmächtigkeit der EPK. Die meisten Verbesserungsvorschläge beziehen sich auf Syntax und Semantik der originären EPK-Definition. Die Frage, ob diese den aktuellen Ansprüchen an Prozessmodelle noch gerecht wird, rückte dabei meist in den Hintergrund. Erfolgt die Modellbildung heute primär unter der Zielsetzung der Prozessanalyse, so reicht die Widerspruchsfreiheit der bestehenden syntaktischen Konstrukte alleine nicht mehr aus; auch das Verhalten der zu analysierenden (realen) Prozesse muss in einer problemadäquaten Form spezifizierbar und validierbar sein. In diesem Beitrag werden mit der Multiinstanziierungsfähigkeit und der referentiellen Persistenz zwei Modelleigenschaften vorgestellt, die von der EPK nicht unterstützt werden. Insbesondere in Anwendungskontexten, in denen Mengengerüste und deren Steuerung von Prozessen auf verschiedenen Granularitätsstufen eine entscheidende Rolle spielen, wie oftmals in der Logistik, führen diese Defizite zu einer eingeschränkten Spezifizierbarkeit realer Prozesse.

2 Ereignisgesteuerte Prozessketten (EPK)

Die Ereignisgesteuerte Prozesskette (EPK) wurde im Jahre 1992 am Institut für Wirtschaftsinformatik (IWI) der Universität des Saarlandes zusammen mit der SAP AG als Technik[1] zur visualisierten Dokumentation von Abläufen und Geschäftsprozessen entwickelt [KNS92; SNZ95]. Das EPK-„Grundmodell“ konzentriert sich auf die Beschreibung des Kontrollflusses von Geschäftsprozessen unter Verwendung von Ereignissen, Funktionen und Prozesswegweiser sowie logischer Konnektoren und Kontrollflusskanten. In seiner einfachsten Form wird ein Kontrollfluss ausgedrückt durch eine alternierende Folge von Ereignissen und Funktionen.

[1] zur Begriffsbildung: im Folgenden sei mit der EPK die Modellierungstechnik im Sinne eines Diagrammtyps, mit einem EPK-Modell ein Prozessmodell gemäß der Syntax und Semantik der EPK, mit einem Prozesspfad ein maximaler unverzweigter Ast im EPK-Modell und mit einem Prozess eine konkrete ablauffähige Instanz des EPK-Modells bezeichnet.

Zur Gestaltung komplexerer Prozesse dienen logische Konnektoren der Typen AND, XOR und OR, die den gleichnamigen Junktoren der Aussagenlogik entsprechen sollen[2]. Optional kann dieses Grundmodell um weitere Elemente (Datenelemente, Ressourcen, organisatorische Elemente) zur „eEPK“ erweitert werden. Zur Reduktion der Modellkomplexität kann ein EPK-Modell zum einen „zerschnitten“ und auf mehrere Seiten verteilt werden (horizontale Dekomposition). Zum anderen lässt sich eine (komplexe) Funktion in einem EPK-Modell hierarchisieren und ihr Verhalten durch ein separates EPK-Modell grafisch beschreiben (vertikale Dekomposition). Da zu jedem dekomponierten EPK-Modell ein nichtdekomponiertes Äquivalent existiert, beschränken sich die folgenden Ausführungen auf nichtdekomponierte Modelle. Abbildung 1 zeigt exemplarisch einen Ausschnitt aus dem EPK-Modell „SD Vertrieb, Terminauftragsbearbeitung“ des R/3-Referenzmodells der SAP AG (in Anlehnung an [KM94]).

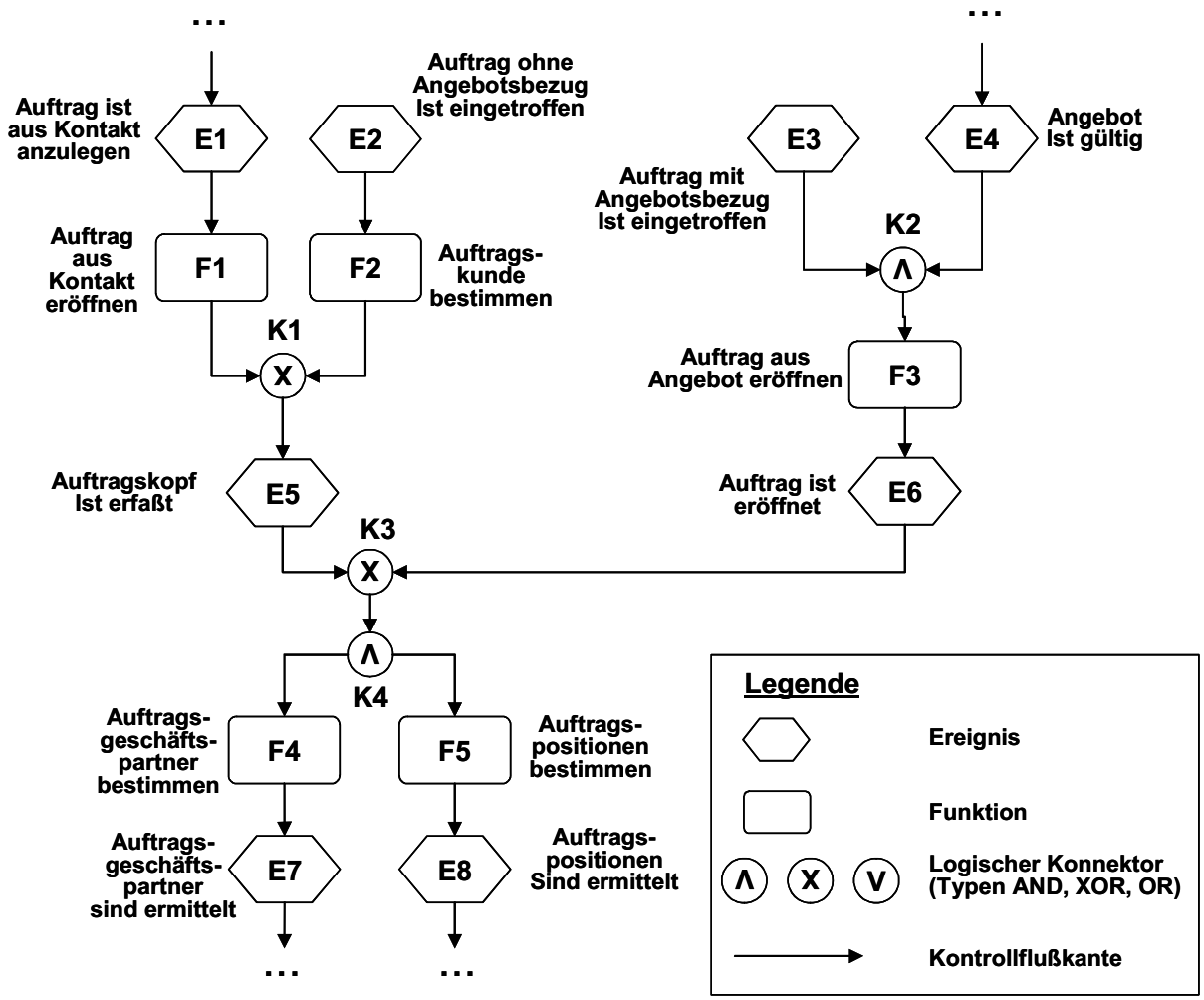


Abbildung 1: EPK-Modell „SD Vertrieb, Terminausftragsbearbeitung“ (Ausschnitt)

[2] in [Ro97] wurde anhand der Assoziativitätseigenschaft gezeigt, dass durch die Erweiterung des binären XOR-Junktors aus der Aussagenlogik zum n-ären Konnektor in der EPK auch die Semantik des XOR verändert wurde. Dem XOR in der EPK liegt tatsächlich eine „1 aus n“-Semantik zugrunde, die nur im Fall n=2 derjenigen des XOR-Junktors entspricht.

Durch Instanziierung eines EPK-Modells (genauer: seiner Prozessstruktur) lassen sich ablauffähige Prozesse generieren. Zur eindeutigen Identifikation einer jeden Instanz wird dieser eine Prozess-ID zugeteilt. Die Initialisierung des Prozesses erfolgt durch Markierung einer Teilmenge der Startereignisse, von denen jedes markierte Ereignis mit einer Prozessmappe versehen wird. Eine Prozessmappe enthält neben der Prozess-ID der zugehörigen Instanz ein Protokoll über den Prozessablauf hinsichtlich Zeiten und Kosten, welches mit jedem Prozessfortschritt dynamisch fortgeschrieben wird. Gemäß der EPK-Semantik wird die Prozessmappe während der Prozessausführung an das jeweils nächste Element der Prozessstruktur weitergereicht, bis ein Endereignis erreicht ist.

Mit Hilfe des AND-Konnektors lässt sich ein Prozess – entsprechend der Anzahl sich öffnender Prozesspfade – in mehrere Teilprozesse zerlegen (AND-Split), die nebenläufig ausgeführt werden; eine Synchronisation der Teilprozesse erfolgt mit dem AND-Join. Nach der originären EPK-Definition darf sich in einer Instanz (d.h. in einem sich in Ausführung befindlichen Prozess) in jedem Prozesspfad maximal eine Prozessmappe aufhalten. Laufen mehrere Prozesse desselben EPK-Modells simultan ab, so lassen sich anhand der Prozess-ID nur Teilprozesse miteinander synchronisieren, die zu demselben Prozess gehören. Auf diese Weise wird verhindert, dass verschiedene Prozesse miteinander vermischt werden.

Der XOR-Split (OR-Split) steuert die Fortsetzung eines Prozesses auf genau einem (einer nichtleeren Teilmenge) der sich öffnenden Prozesspfade. Die Semantik des XOR-Join sowie insbesondere des OR-Join ist in der Literatur bis heute umstritten. Den Autoren der originären EPK-Definition wird in zahlreichen Beiträgen – unter anderem in [LSW97], [LSW98], [Ru97], [Ru99], [Aa99], [Ro97] und [MR00] – Unvollständigkeit und Interpretationsfähigkeit vorgeworfen. Diese Problematik soll hier nicht weiter diskutiert werden. Es sei nur angemerkt, dass sich bis heute kein allgemein anerkannter Standard zur Interpretation der genannten Konstrukte herausgebildet hat.

3 Modellierung von Geschäftsprozessen mit der EPK

Eine Vielzahl von Geschäftsprozessen zeichnet sich aus durch die Existenz eines „zentralen Objektes“. Exemplarisch sei der Kernprozess „Auftragsbearbeitung“ mit seinem zentralen Objekt „Kundenauftrag“ genannt. Mit diesem „objektbasierten“[3] Prozessverständnis soll ein Geschäftsprozess verstanden werden als Beschreibung des Lebenszyklus’ seines zentralen Objektes innerhalb der festgelegten Systemgrenzen. Neben dem zentralen Objekt gehen verschiedene weitere Objekte in den Prozess ein. Der hier verwendete Objektbegriff umfasst dingliche und abstrakte Gegenstände sowie Informationen beliebiger Komplexität, von atomaren Items bis hin zu hochkomplexen Gebilden, die sich wiederum aus anderen Objekten zusammensetzen können. Ein Kundenauftrag besteht beispielsweise aus n Auftragspositionen (Abbildung 2), wobei die Anzahl n der Auftragspositionen von Auftrag zu Auftrag variieren kann.

[3] der Begriff der Objektorientierung soll hier nicht verwendet werden, da in der Softwaretechnik mit ihm Konzepte und Eigenschaften wie Vererbung, Polymorphie, Datenkapselung uvm. verbunden sind, die hier keinen Eingang finden.

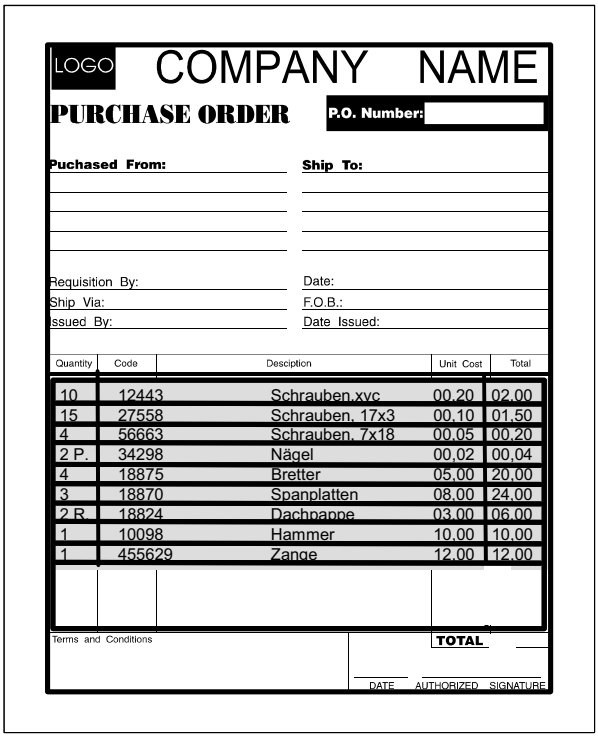


Abbildung 2: Kundenauftrag mit Auftragspositionen

Diese Beziehung zwischen Auftrag und Auftragsposition entspricht einer 1:n-Beziehung im relationalen Datenmodell bzw. Klassendiagramm. Eine Auftragsposition kann ihrerseits wiederum als zentrales Objekt eines „Auftragspositionsprozesses“ verstanden werden. Der 1:n-Beziehung zwischen dem Objekt „Auftrag“ und seinen n Teilobjekten vom Typ „Auftragsposition“ wird eine korrespondierende 1:n-Beziehung zwischen den jeweiligen Prozessen gegenübergestellt (Abbildung 3).

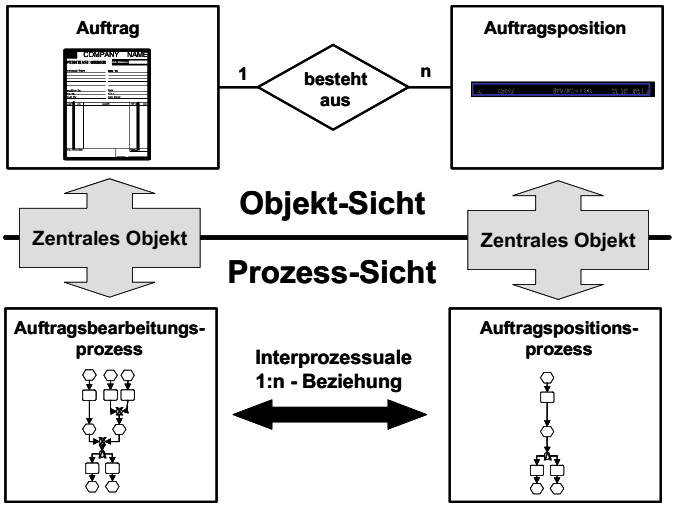


Abbildung 3: Korrespondierende 1:n-Beziehungen zwischen Objekten und Prozessen

Unter der Zielsetzung der Prozessoptimierung erscheint eine Disaggregation des zentralen Objektes und eine damit einhergehende differenzierte Betrachtung der Prozess-Granularitätsebenen[4] stets dann sinnvoll, wenn wesentliche Effizienzhemmnisse (Deadlocks, Livelocks, Synchronisationen von Teilprozessen etc.) in Details vermutet werden. Wird der Auftrag als eine unteilbare Einheit verstanden, so wird die Gesamtdurchlaufzeit der Auftragsbearbeitung determiniert durch die Bearbeitungsdauer der aufwendigsten Auftragsposition. Lässt sich dagegen bei den Durchlaufzeiten der einzelnen Auftragspositionen eine hohe Varianz nachweisen, so lohnt es sich durchaus, über eine Auslieferung des Auftrages in mehreren Margen nachzudenken. Auf diesem Wege lässt sich eine Reduktion der partiellen Liegezeiten aller bereits fertigen Positionen erreichen.

Die Analyse eines komplexen Prozesses über mehrere Granularitätsebenen verlangt eine saubere Spezifikation der interprozessualen 1:n-Beziehung. Hier stößt die EPK an die Grenzen ihrer Ausdrucksmächtigkeit. Unter dem Begriff der „**Multiinstanziierungsfähigkeit**“ wird diese Diskussion in Sektion 3.1 vertieft.

An einen realen Prozess wird in der Praxis zunehmend der Anspruch gestellt, jederzeit Auskunft über seinen Ausführungsfortschritt geben zu können („Prozessmonitoring“). Im Sinne der Kundenorientierung soll es einem Kunden jederzeit möglich sein, den Fertigstellungsgrad seines erteilten Auftrages sowie die erwartete Restlaufzeit abzurufen. Um eine Aussage über den Status eines Auftrages im System treffen zu können, sind die partiellen Stati aller zugehörigen Auftragspositionen zu ermitteln und diese Informationen zu einer Gesamtaussage zu verdichten. Dazu ist die Referenz zwischen einem Kundenauftrag und den im System verteilten zugehörigen Auftragspositionen während des gesamten Lebenszyklus’ des Kundenauftrages aufrecht zu erhalten. Die damit geforderte „**Referentielle Persistenz**“ ist Gegenstand der Sektion 3.2. Es wird ergänzend gezeigt, dass die referentielle Persistenz zusätzlich erforderlich ist, um die im Rahmen einer Multiinstanziierung erzeugten Instanzen des Auftragspositionsprozesses wieder zu dem ursprünglichen Auftrag zusammensetzen zu können (Multiinstanzen-Synchronisation).

3.1 Multi-Instanziierungsfähigkeit

Als Multi-Instanziierungsfähigkeit sei die Eigenschaft einer Modellierungstechnik bezeichnet, in einem Prozess endlich viele identische[5] und paarweise unabhängige Instanzen eines anderen (abhängigen) Prozesses generieren zu können. In unserem Beispiel sollen aus dem Auftragsbearbeitungsprozess n paarweise unabhängige Instanzen eines Auftragspositionsprozesses erzeugt werden. Die originäre EPK stellt dafür kein spezielles Konzept bereit. Mit folgenden syntaktisch zulässigen Konstrukten lässt sich die interprozessuale 1:n-Schnittstelle allenfalls skizzieren:

[4] man beachte, dass mit den Granularitätsebenen keineswegs zwingend Prozessmodelle auf verschiedenen Hierarchieebenen im Sinne hierarchischer EPK-Modelle gemeint sind.

[5] es handelt sich hier um Prozessinstanzen mit einer paarweise identischen Struktur, diese im Sinne einer kausalen und zeitlichen Halbordnung der Aktivitäten. Nicht zwingend identisch ist das Laufzeitverhalten. Es wird im Rahmen einer Prozessausführung für jede Instanz durch deren Kontext individuell bestimmt.

1. **Strukturelle Replikation**: Die Struktur des Auftragspositionsprozesses wird n-fach repliziert (Abbildung 4a) und mit Hilfe eines AND-Split auf jeder Replik genau eine Instanz erzeugt. Zwar wird auf diese Weise die geforderte paarweise Unabhängigkeit der Prozessinstanzen sichergestellt. Da von Auftrag zu Auftrag die Anzahl der Auftragspositionen jedoch variieren kann, sollte von einer strukturellen „harten" Codierung der Prozessinstanzen abgesehen werden. Vielmehr empfiehlt es sich, das n variabel zu halten, um die Anzahl der Positionen generisch anpassen zu können.

2. **Serialisierung**: Mit Hilfe eines Schleifenkonstruktes (Abbildung 4b) wird der Auftrag in n Iterationen sukzessive durchlaufen, dabei jeweils eine Auftragsposition abgespalten und instanziiert. Dieses Konstrukt ist einerseits syntaktisch zulässig, andererseits verbietet die originäre EPK-Semantik die Existenz mehrerer Instanzen in einem Prozesspfad. Unabhängig von dieser Problematik führt diese Lösung zu einer künstlichen Serialisierung der Prozessinstanzen, die Forderung nach paarweiser Unabhängigkeit ist nicht erfüllt. Durch die verfälschte Darstellung des realen Prozessverhaltens wird dem Modell die Grundlage zur Prozessoptimierung entzogen, denn mit ausreichend Ressourcen könnten im realen Prozess im Extremfall alle Instanzen parallel bearbeitet werden, so dass die Durchlaufzeit aller Instanzen in diesem Fall $\max_i(Durchlaufzeit(\text{Prozess}_i))$ beträgt.

   Im Falle der erzwungenen Serialisierung dagegen beträgt die gesamte Durchlaufzeit stets $\sum_i(Durchlaufzeit(\text{Prozess}_i))$. Die Optimalität des Prozessverhaltens ist damit aus dem Modell heraus nicht mehr quantifizierbar.

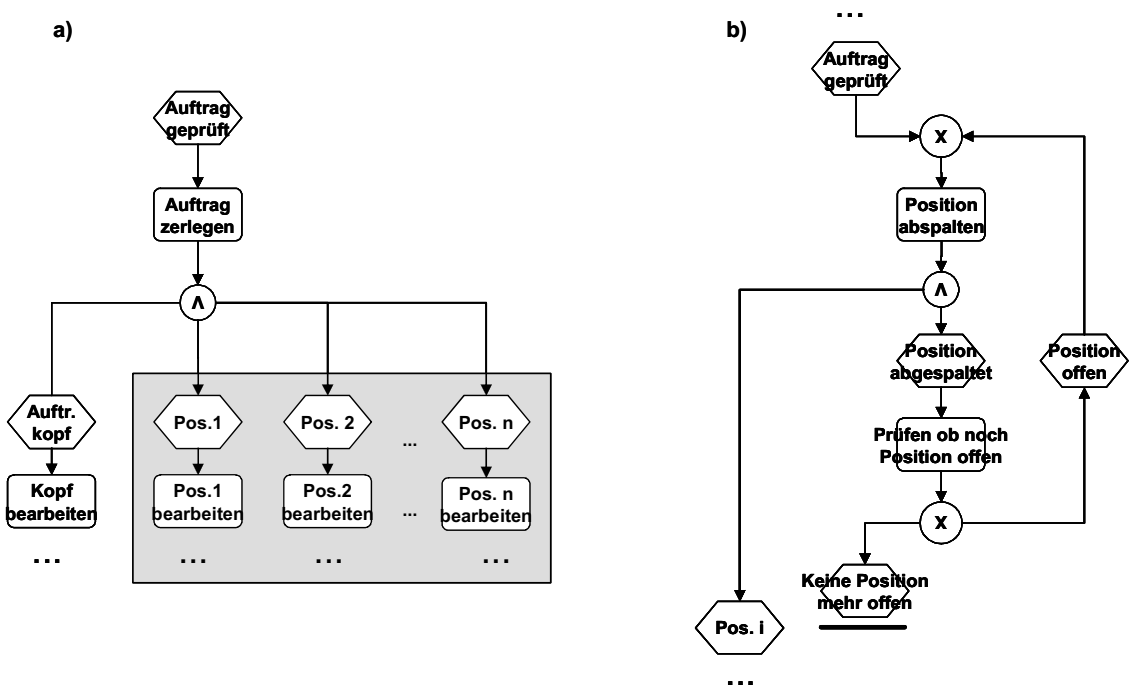


Abbildung 4: a) Strukturelle Replikation und b) Serialisierung

3. **Speichersynchronisation**: Hier wird die interprozessuale 1:n-Beziehung außerhalb der Prozessmodelle dargestellt. Dazu sei ein Speicher modelliert, in den auf der einen Seite der Auftragsbearbeitungsprozess alle n Auftragspositionen als Objekte hineinlegt. Auf der anderen Seite nimmt sich jede der n Auftragspositionsprozessinstanzen genau eines dieser Objekte heraus und verarbeitet es (Abbildung 5). Bei dieser Option handelt es sich lediglich um ein grafisches Hilfskonstrukt, um das intendierte Verhalten der Prozesse zu skizzieren.

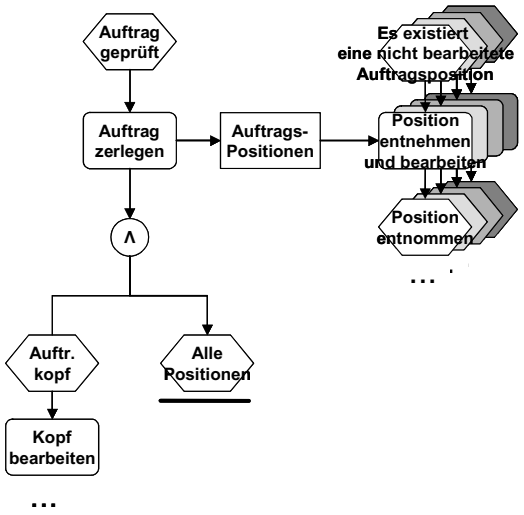


Abbildung 5: Speichersynchronisation

Eine saubere Lösung zur Spezifikation der Multiinstanziierung bieten Petrinetze[6], insbesondere Stellen-/Transitionsnetze (S/T-Netze), vgl. Abbildung 6a. Mit Hilfe eines Kantengewichtes generiert die Transition T_1 simultan n Auftragspositionen und legt diese in der Stelle S_3 ab. Deren Verarbeitung erfolgt auf dem rechten Pfad paarweise unabhängig, da jede einzelne Auftragsposition die nachfolgende Transition T_3 individuell aktiviert im Sinne des Prinzips der „lokalen Kausalität“, das Petrinetze grundsätzlich auszeichnet. Damit erhalten wir die gewünschte Ausdrucksmächtigkeit zur Multi-Instanziierung eines Prozesses, die ein EPK-Modell nicht bietet.

Es wäre überlegenswert, die EPK-Definition um Kantengewichte oder einen speziellen „Multiinstanziierungs-Konnektor“ (n-Split und n-Join) zu erweitern (Abbildung 6b). Allerdings ist dann die Einschränkung, dass sich in jedem Prozesspfad nur eine Prozessmappe aufhalten darf, zumindest für den durch das n-Split und das n-Join gekapselten Pfad nicht mehr haltbar. Die Syntax und die Semantik der EPK wäre sowohl für die lokalen Konstrukte als auch für globale Strukturen zu überdenken.

[6] Eine Einführung in Petrinetze und eine Übersicht über verschiedene Netzklassen geben unter anderem [BRR86, Re86, Ba90, St90, PNW02].

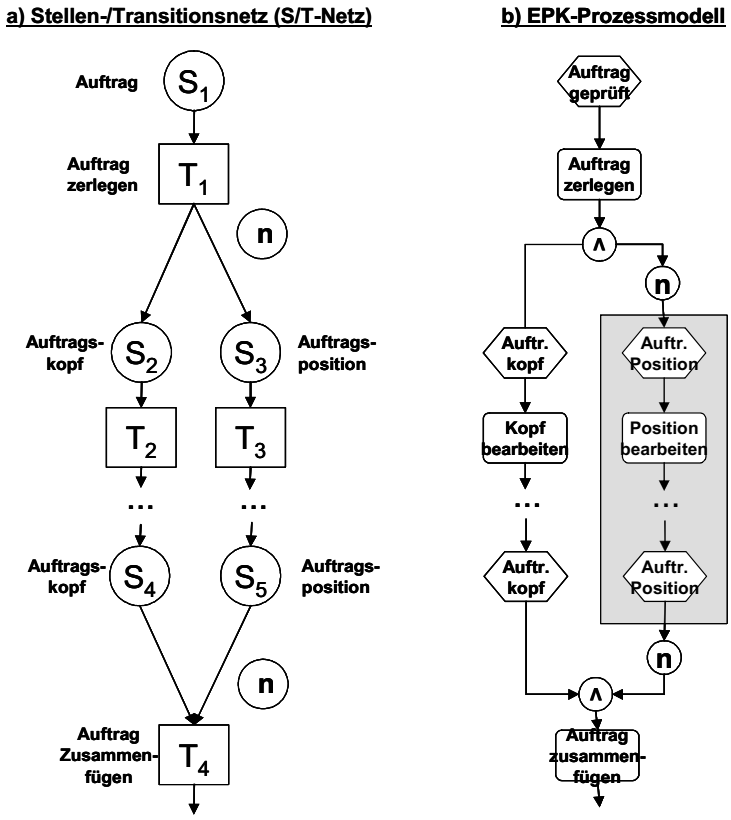


Abbildung 6: Multiinstanziierung im EPK-Modell und im S/T-Netz

3.2 Referentielle Persistenz

Als „referentielle Persistenz“ sei die Eigenschaft eines Systems bezeichnet, eine referentielle Beziehung zwischen zwei Objekten unabhängig von deren aktuellen Aufenthaltsorten im Modell bzw. System derart zu speichern, dass diese Information zu einem beliebigen Zeitpunkt abrufbar ist. Die Speicherung dieser referentiellen Beziehung soll für die Dauer der Existenz der beteiligten Objekte im System Bestand haben (es sei denn, dass sie durch eine entsprechende Aktivität gewollt aufgelöst wird).

In unserem Beispiel soll die Referenz einer jeden Auftragsposition zum übergeordneten Auftrag auch nach der erfolgten Übergabe an den „Auftragspositionsprozess“ persistent gehalten werden. Zum einen ist diese Referenz notwendig, um zu einem späteren Zeitpunkt alle Positionen des Auftrages synchronisieren und damit den Auftrag wieder vollständig rekonstruieren zu können. Zum anderen ist bei einer Verteilung der einzelnen Auftragspositionen im System eine Statusabfrage des gesamten Auftrages nur mit den genannten Referenzen möglich. Es stellt sich die Frage, wo die Referenzen im Modell gespeichert werden sollen. Ein erneuter Vergleich mit Petrinetzen offeriert zwei potentielle Ansätze:

1. **Zentrale persistente Speicher**: es werden persistente Speicher erzeugt, in denen Referenzen zentral abgelegt und verwaltet werden. Beliebige Elemente des Systems erhalten Lesezugriff auf diese Speicher, das Löschen der Speicherinhalte dagegen ist besonderen „Clearing-Funktionen“ vorbehalten. Die originäre

EPK stellt grundsätzlich kein Element mit expliziter Speicherfunktionalität bereit. Das Ereignis als einziges passives Element der EPK ist definiert als „das Eingetretensein eines betriebswirtschaftlich relevanten Zustandes eines Informationsobjektes, der den weiteren Ablauf des Geschäftsprozesses steuert oder beeinflusst (...). Im Gegensatz zu einer Funktion, die ein zeitverbrauchendes Geschehen dargestellt, ist ein Ereignis auf einen Zeitpunkt bezogen." [IDS00, S. 4-93]. Mit seinem Zeitpunkt-Bezug ist es für persistente Speicherungen ungeeignet. Dass Ereignisse dennoch interpretierbar sind als „Faltung" des Eintretens in einen Zustand mit dem Austreten aus diesem Zustand und damit einen (zeitbehafteten) Zustand implizit kapseln können sowie syntaktisch zulässige Konstrukte mit impliziter Speicherfunktionalität existieren [Ro97], soll hier vernachlässigt werden.

2. **Referenz-Informationen als Teil der Objekte**: Wie weiter oben ausgeführt, wird bei der Instanziierung eines EPK-Modells eine Prozess-ID generiert und an diejenigen Prozessmappen gebunden, mit denen die Initialisierung der Startereignisse erfolgt. Im Rahmen der Ausführung eines Prozesses wandern diese Prozessmappen auf paarweise verschiedenen Prozesspfaden durch das EPK-Modell, bis alle Mappen in Endereignissen angekommen sind und damit der Prozess terminiert. Wird das zentrale Objekt im Prozessverlauf in Unterobjekte zerlegt, die ihrerseits wieder dekomponierbare Objekte darstellen können, so tragen alle auf diese Weise erzeugten Objekte dieselbe einheitliche Prozess-ID mit sich. Es wird auf diese Weise zwar verhindert, dass Auftragspositionen verschiedener Aufträge miteinander synchronisiert und vereinigt werden. Nicht verhindern lässt sich jedoch, dass innerhalb einer gemeinsamen Prozess-ID Unterobjekte bzw. die dazu korrespondierenden Teilprozesse verschiedener Granularitätsebenen synchronisiert werden.

Vor diesem Hintergrund erscheint es sinnvoll, einem während der Prozessausführung generierten Objekt neben dem eigenen Identifikator auch den des jeweils übergeordneten Objektes mitzugeben. Um das Problem der referentiellen Persistenz mit Petrinetzen lösen zu können, ist eine Netzklasse zu bemühen, die eine Individualisierung der Objekte (Marken) gestattet. Diese Fähigkeit besitzen so genannte „High Level Petrinetze" wie beispielsweise gefärbte Petrinetze (Coloured Petri Nets (CPN), [Je92]) oder Prädikat/Transitionen-Netze (Pr/T-Netze, [Ge86]). In [Ro97] wurde auf Basis von CPN ein Vorschlag zur Vergabe von Objekt- bzw. Prozess-Identifikatoren unterbreitet, der Umgang mit Referenzen zwischen verschiedenen Objekten im Rahmen einer Objektdekomposition jedoch nicht behandelt. Ein Lösungsvorschlag ist in Abbildung 7 skizziert.

4 Zusammenfassung und Ausblick

Mit dem zunehmenden Anspruch, mit Hilfe von Prozessmodellen fundierte Aussagen über das Verhalten und die Effizienz real ablaufender Geschäftsprozesse generieren sowie Optimierungspotentiale lokalisieren zu können, wächst auch der Anspruch an die Ausdrucksmächtigkeit der verwendeten Modellierungstechniken. Insbesondere im Bereich der Logistik, in dem zahlreiche Geschäftsprozesse durch Mengen und komplexe

Objekte gesteuert werden, können viele Effizienzhemmnisse erst durch eine Spezifikation und Analyse sowohl der zentralen Objekte als auch der damit korrespondierenden Prozesse auf verschiedenen Granularitätsebenen erkannt werden. Dazu ist unter anderem eine durchgängige und saubere Spezifikation der jeweiligen Schnittstellen zwischen Objekten und/oder Prozessen unterschiedlicher Granularität erforderlich. In diesem Zusammenhang wurden mit der Multiinstanziierungsfähigkeit und der referentiellen Persistenz zwei Modell- bzw. Systemeigenschaften diskutiert, die von der EPK in ihrer originären Fassung nicht unterstützt werden und die zu einer signifikanten Einschränkung der Ausdrucksmächtigkeit der EPK führen. Unter Bezugnahme auf Petrinetze wurden Ansätze zur potentiellen Erweiterung der EPK skizziert.

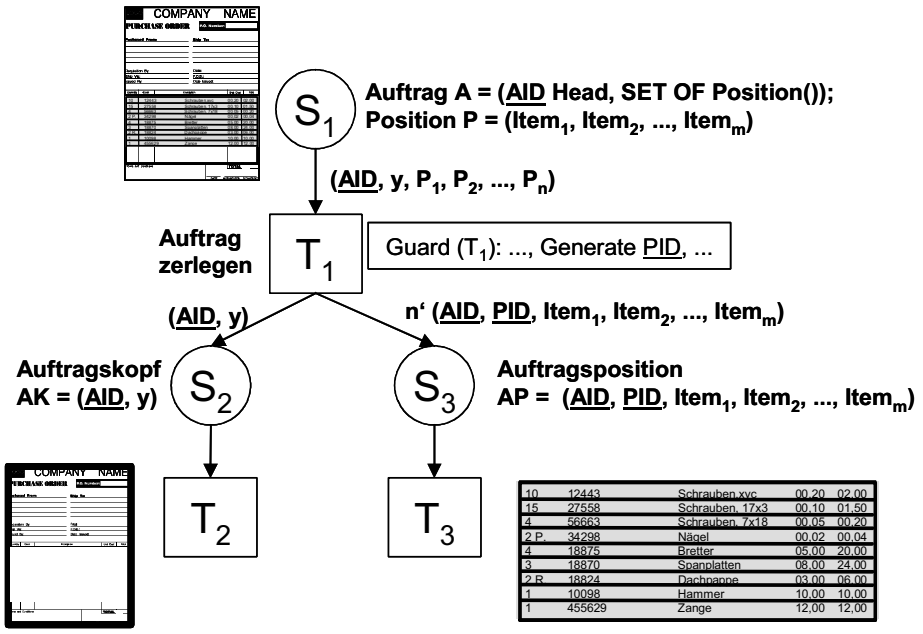


Abbildung 7: Vergabe von Identifikatoren im CPN

Offen bleibt die Frage, ob die diskutierten Eigenschaften von derart essentieller Bedeutung sind, dass sie in die Spezifikation des EPK-Basismodells mit aufgenommen werden sollten oder ob – analog zu den verschiedenen Klassen von Petrinetzen – neben der originären EPK eine neue ausdrucksmächtigere EPK-Klasse für spezielle Anwendungszwecke entsteht. Eine formale Ausarbeitung der in diesem Beitrag skizzierten Ansätze einschließlich einer Ausdehnung auf vertikal und horizontal dekomponierte EPK-Modelle könnte zur Beantwortung dieser Frage wertvolle Beiträge liefern. Dazu bietet die GI-Arbeitsgruppe „Formalisierung uns Analyse ereignisgesteuerter Prozessketten (EPK)“ [EPK97], die derzeit als GI-Arbeitskreis „Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten“ [EPK02] eine Renaissance erlebt, sicher eine geeignete organisatorische Basis. Es bleibt abzuwarten, ob sich mit den vorgeschlagenen Erweiterungen ein neuer und allgemein anerkannter EPK-Standard entwickeln läßt, der dann auch Eingang in die entsprechenden Werkzeuge, wie beispielsweise in das ARIS Toolset der IDS Scheer AG, findet.

Literaturverzeichnis

[Aa99] van der Aalst, W.M.P.: Formalization and Verification of Event-driven Process Chains, in: Information and Software Technology 41(1999)10, S. 639-650. URL: http://tmitwww.tm.tue.nl/staff/wvdaalst/Publications/ p74.pdf (2002-11-10).

[Ba90] Baumgarten, B.: Petri-Netze: Grundlagen und Anwendungen; BI-Wissenschaftsverlag Mannheim, Wien, Zürich 1990.

[BRR86] Brauer, W.; Reisig, W.; Rozenberg, G. (Hrsg.): Petri Nets: Central Models and Their Properties; in: Advances in Petri Nets 1986, Part I Proceedings of an Advanced Course Bad Honnef, September 1986; Lecture Notes in Computer Science (LNCS) No. 254 Springer Verlag Berlin et al. 1986.

[EPK97] Arbeitsgruppe „Formalisierung uns Analyse ereignisgesteuerter Prozessketten (EPK)“ der Gesellschaft für Informatik (GI). URL: http://www-is.informatik.uni-oldenburg.de/~epk/ (2002-11-10).

[EPK02] Arbeitskreis „Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten“ der Gesellschaft für Informatik (GI). URL: http://www.epk-community.de/ (2002-11-10).

[EW95] Esser, M.; Wittges, H.: Konzept für den Einsatz der „Architektur integrierter Informationssysteme (ARIS) bei der Stadtwerke Düsseldorf AG; in: Informationssystem Architekturen, Heft 1/1995, S. 93-99.

[Ge86] Genrich, H.: Predicate/Transition Nets; in [BRR86], S. 207-247.

[IDS00] IDS Scheer AG: ARIS Methodenhandbuch, Version 5, Stand Mai 2000.

[Je92] Jensen, K.: Coloured Petri Nets. Basic concepts, analysis methods and practical use; Volume 1; EATCS monographs on Theoretical Computer Science, Springer Verlag, New York 1992.

[Ke99] Keller, G. & Partner: SAP/ R3 prozeßorientiert anwenden. Iteratives Prozeß-Prototyping mit Ereignisgesteuerten Prozeßketten und Knowledge Maps, Addison Wesley Longman, Bonn et al. 1999.

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

[LSW97] Langner, P.; Schneider, C.; Wehler, J.: Ereignisgesteuerte Prozeßketten und Petri-Netze, in: Valk, R.; Jantzen, M. (Hrsg.): Bericht Nr. 196, Fachbereich Informatik der Universität Hamburg, Hamburg 1997.

[LSW98] Langner, P.; Schneider, C.; Wehler, J.: Petri Net Based Certification of Event driven Process Chains, in: Desel, J.; Silva, M. (Hrsg.): Application and Theory of Petri Nets 1998, Lecture notes in Computer Science Vol. 1420, (Springer) Berlin et al. 1998, S. 286-305.

[MR00] Moldt, D.; Rodenhagen, J.: Ereignisgesteuerte Prozeßketten und Petrinetze zur Modellierung von Workflows, in: Giese, H.; Philippi, S. (Hrsg.): Visuelle Verhaltensmodellierung verteilter und nebenläufiger Software-Systeme, Proceedings zum 8. Workshop des Arbeitskreises "Grundlagen objektorientierter Modellierung" (GROOM) der GI-Fachgruppe 2.1.9 ("Objektorientierte Softwareentwicklung"), Bericht Nr. 24/00-I, Münster 2000, S. 57-63. URL: http://wwwmath.uni-muenster.de/cs/u/versys/workshops/VVVNS2000/ papers/Moldt.pdf (2002-10-18).

[PNW02] Homepage der „Petrinet World“; http://www.daimi.aau.dk/~petrinet/ (2002-11-10).

[Re86] Reisig, W.: Petrinetze - Eine Einführung; 2. Auflage, Springer Verlag Berlin et al. 1986.

[Ro97] Rodenhagen, J.: Darstellung ereignisgesteuerter Prozeßketten (EPK) mit Hilfe von Petrinetzen, Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 1997.

[Ro00] Rodenhagen, J.: Die papierlose Bank – Business Process Modelling und elektronischer Workflow; interner Vortrag, Infomatec AG, Augsburg, Juli 2000

[Ru97] Rump. F.J.: Erreichbarkeitsgraphbasierte Analyse ereignisgesteuerter Prozeßketten, Technischer Bericht Fachbereich Informatik Universität Oldenburg, Oldenburg 1997.

[Ru99] Rump, F.J.: Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten - Formalisierung, Analyse und Ausführung von EPKs, Teubner, Stuttgart et al.1999.

[Sc99] Scheer, A.-W.: ARIS - Vom Geschäftsprozeß zum Anwendungssystem, 4. Aufl., (Springer) Berlin et al. 1998.

[Sc01] Scheer, A.-W.: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen, 4. Aufl., (Springer) Berlin et al. 2001.

[SNZ95] Scheer, A,-W.; Nüttgens, M.; Zimmermann, V.: Rahmenkonzept für ein integriertes Geschäftsprozessmanagement; in: Wirtschaftsinformatik 37 (1995) 5, S. 426-434.

[St90] Starke, P.H.: Analyse von Petri-Netz-Modellen; B.G. Teubner Verlag Stuttgart, 1990

Sprachgerechte unternehmensnahe Modellierung von Ereignisgesteuerten Prozessketten
-
Zur ädaquaten Aus- und Weiterbildung von ModelliererInnen

Christian Fichtenbauer, Max Rumpfhuber, Christian Stary

CF-CONSULT
Am Fischhof 2/3
A-1010 Wien
&
Universität Linz
Institut für Wirtschaftsinformatik
Communications Engineering
Freistädterstraße 315
A-4040 Linz
E-Mail: christian.fichtenbauer@cf-consult.at, max.rumpfhuber@cf-consult.at,
christian.stary@jku.at

Abstract: Der Einsatz von Werkzeugen zur Modellierung von Ereignisgesteuerten Prozessketten wie ARIS zeigt, dass die semantische Beherrschung der Notation eine unabdingbare Voraussetzung für eine unternehmensgerechte Abbildung von Geschäftsprozessen in Ereignisgesteuerte Prozessketten ist. Daher müssen im Vorfeld der Modellierung sowie begleitende Maßnahmen, zur eigentlichen Modellierung gesetzt werden. Das Projekt INTELIBUS (INTELligence In BUSiness modeling) versucht, mit Hilfe empirisch begründbarer fachdidaktischer Maßnahmen Prozessketten-Modellierung ARIS-sprachgerecht zu modellieren. Hauptaktivität in diesem Projekt stellt zunächst die Verständnisbildung der feststellbaren Schulungs- und Anwendungsprobleme dar. Mit Hilfe semantischer Inhaltsanalysen kann danach bestimmt werden, welche fachdidaktischen Konzepte zur sprachgerechten Verwendung erfolgreich eingesetzt werden können bzw. welche grundlegende Verständnisprobleme vermieden werden können. Aus diesen Erkenntnissen können Hilfsmittel und Methoden (wie Szenarien und Frage/Antwort-Sequenzen) entwickelt werden. INTELIBUS soll zu einem Werkzeug führen, welche die Modellierung von Ereignis gesteuerten Prozessketten ohne zeit- und kostenintensive Intervention von TrainerInnen ermöglicht. Durch semantisch und syntaktisch richtige (sprachgerechte) Modelle, kann die Qualität des organisationalen Unternehmenswissens signifikant erhöht werden.

1 Einleitung

Die Modellierung von Unternehmen und ihrer Geschäftsprozesse gewinnt vermehrt an Bedeutung. Grund dafür ist der vielfältige Einsatz von Unternehmensmodellen und die

Nutzungsmöglichkeiten entsprechender Spezifikationen. Ereignisgesteuerte Prozessketten stellen die Grundlage zur zielgerichteten Organisationsentwicklung und –optimierung dar. Sie können weiters als Kommunikationsplattform zwischen Fachbereich und IT-Abteilung dienen und können sowohl zu Analysezwecken entwickelt, als auch bei Designaufgaben eingesetzt werden (vgl. [DR01]. In allen Fällen muss die Qualität der Modelle und Spezifikationen, den Einsatz des abgebildeten Wissens unterstützen. Nur dann ist eine zweckgerichtete Nutzung des Unternehmenswissens erreichbar. Diese Nutzung betrifft nicht nur die unmittelbare Anwendung und Umsetzung von Prozesswissen, sondern auch die Weiter- bzw. Wieder-Verwendung von Unternehmensmodellen (vgl. [BES98], [De01]).

Aufgrund mehrerer Projekt-Vorhaben (z.B. Grundsätze der ordentlichen Modellierung [BES98]) wird evident, dass die derzeit mittels Techniken und Werkzeugen erreichte Qualität von Unternehmensmodellen nicht für deren Einsatzzweck ausreicht und daher zu erhöhen ist. Unterschiedliche Sachverhalte und Probleme erschweren die Erreichung der erforderlichen Modell-Qualität: So sind Ereignisgesteuerte Prozessketten äußerst unterschiedlich in ihrer Tiefe und Abstraktionsebene gestaltet, meist in Abhängigkeit unterschiedlich verfolgter Modellierungsziele (z.B. Simulation, Kernprozess-Definition, Workflow-Spezifikation). Weiters lassen die verwendeten Modellierungssprachen meist mehrfache Interpretation der spezifizierten Inhalte zu, sei es bei der logisch-kausalen Zusammenführung von Operationen, oder bei der Strukturierung der Daten. Schließlich bestimmt die Erfahrung des Modellierers/der ModelliererIn, wie syntaktisch oder/und semantisch korrekt eine Spezifikationssprache eingesetzt wird und somit, wie nahe das Modell ein realitätsnahes Abbild des betroffenen Unternehmens darstellt.

Will folglich ein Unternehmen Mehrwert aus Prozessmodellen erzielen, dann stellt die semantisch und syntaktisch korrekte Verwendung von Modellierungssprachen eine Grundvoraussetzung für sämtliche Aktivitäten der Organisationsentwicklung und –optimierung dar. Dies kann an den folgenden Aussagen zum Stellenwert und den wesentlichen Charakteristika von Geschäftsprozessmodellen gezeigt werden:

- Das Unternehmensmodell ist die Basis für das Verständnis eines Unternehmens. Es umfasst eine Aufzählung der relevanten strukturellen und dynamischen Komponenten des Unternehmens sowie Informationen darüber, wie die Komponenten interagieren. [Ve96]
- Es bietet eine vereinfachte Sicht auf die Unternehmensstruktur und bildet damit eine für alle Anspruchsgruppen gemeinsame Basis für Innovationen und Prozessverbesserungen. [BE01]
- Auch definiert es die Anforderungen an die prozessunterstützenden Informationssysteme und gibt die Rahmenbedingungen der Modellierung vor. [BE01]

In der Folge wird das Projekt INTELIBUS vorgestellt, welches zur Zielerreichung der Anforderungsdefinition für prozessunterstützende Informationssysteme und der Abbildung von Ereignisgesteuerten Prozessketten beitragen soll.

Abschnitt 2 beschreibt die Zielsetzungen sowie den beschrittenen Lösungsweg. Abschnitt 3 beinhaltet erste Ergebnisse und Abschnitt 4 beschließt den Aufsatz mit zwei Schlussfolgerungen.

2 Das Projekt INTELIBUS

Um für Unternehmen mittels Prozessmodellen für die Organisationsentwicklung und –optimierung geeignete Voraussetzungen zu schaffen, wurde seitens CF-Consult gemeinsam mit der Universität Linz (Institut für Wirtschaftsinformatik) das Forschungsprojekt INTELIBUS (INTELligence In BUSiness modeling) initiiert. Es verfolgt das Ziel, UnternehmensmodelliererInnen zu qualifizieren, sodass sie die eingesetzte Modellierungssprache nicht nur syntaktisch und semantisch korrekt beherrschen (vgl. [NR02]), sondern entsprechend dem jeweiligen Unternehmensziel die erforderliche Information auch tatsächlich in dem Modell abbilden. INTELIBUS soll folglich nicht nur die EntwicklerInnen von Unternehmensmodellen qualifizieren, d.h. den Prozess der Erstellung von Modellen durch kompetente MitarbeiterInnen optimieren helfen, sondern auch langfristig sicherstellen, dass die erstellten Modelle auch für die betroffenen Unternehmen den Möglichkeiten der Modellierungssprache entsprechend einsetz- und nutzbar sind.

Grundlage für das Vorhaben stellt ARIS dar, da diese IS-Architektur in der Lage ist, auch komplexes Unternehmenswissen aufzunehmen und darüber hinaus von einem entsprechenden Werkzeug zur Modellierung von Ereignis gesteuerten Prozessketten und zur Prozess-Optimierung unterstützt wird. Schließlich haben die Entwicklungen von ARIS dazu beigetragen, Unternehmen ganzheitlich zu modellieren, anstelle mit verschiedensten Werkzeugen und Sprachen bei der Spezifikation der unterschiedlichen Aspekte des Unternehmenswissens zu operieren.

Das Projekt wurde im November 2001 begonnen und soll 2003 abgeschlossen sein. Es verläuft nach folgendem Phasenschema: Vorbereitung, Explorierung und Evaluierung, Konsolidierung.

2.1 Phase 1: Vorbereitung

In dieser Phase wurden folgende Aktivitäten gesetzt:

- Sammlung und Strukturierung (semantische Inhaltsanalysen) von Erfahrungsberichten zu
  - Problemen bei der Modellierung von Unternehmen
  - Problemen bei der Schulung von ModelliererInnen zur sprachgerechten und unternehmensnahen Abbildung von Unternehmenswissen.
- Erhebung und Auswertung von Fachdidaktiken zur
  - Verwendung konzeptioneller oder Spezifikations-Sprachen in den Disziplinen Organisationslehre, Wirtschaftsinformatik und Informatik

- Schulung von Prozessdenken im Sinne ganzheitlicher Abbildung von Unternehmenswissen
- Schulung von OrganisationsentwicklerInnen, unter anderem anhand von firmenspezifischen Schulungen
- Schulung von Studierenden, und zwar im Rahmen der WirtschaftsinformatikerInnen-Ausbildung.

Ausgewertet wurden das Verständnis der Modellierungskonstrukte und die Fähigkeit, beobachtbare Unternehmenssachverhalte auf die Modellierungskonstrukte abzubilden.

- Analyse von Tele-Teaching/Learning-Ansätzen bezüglich
  - Fachdidaktiken zur Schulung von konzeptionellen Sprachkonstrukten und deren Nutzung an praktischen Fallbeispielen
  - Best practice und success stories in der Anwendung von IT-gestützten Fachdidaktiken im akademischen und unternehmerischen Bereich.

2.2 Phase 2: Explorierung und Evaluierung

In dieser Phase wurden folgende Aktivitäten gesetzt:

- Entwicklung einer Fachdidaktik zur sprachgerechten und unternehmensnahen Modellierung von (organisationalem) Unternehmenswissen beruhend auf
  - Meta-Modellen von ARIS
  - Szenario-basierten Schulungen von ModelliererInnen.

  Im Zentrum der Überlegungen standen dabei die Beziehungen zwischen Modellierungskonstrukten sowie die Schaffung von Situationsbeschreibungen, in denen diese Modellierungskonstrukte ohne zusätzlichen mentalen Aufwand eingesetzt werden können. Durch diesen damit erreichten intuitiven Zugang können Sprachkonstrukte und ihre Anwendung leichter verinnerlicht werden.

- Entwicklung eines Software-Prototyps zur
  - Testung möglicher Web-Architekturen
  - Schnittstellen-Spezifikation mit ARIS
  - IT-gestützten Schulung.

  Da die angestrebte Lösung Web-basiert einer (virtuellen) Gemeinschaft zugänglich gemacht werden soll, stand die Verteilung einer entsprechenden Anwendung zur Disposition. Ebenso mussten die ARIS-Anbindung sowie die mögliche Software-Implementierung von Fachdidaktiken laufend untersucht werden.

- Evaluierung von
  - Fachdidaktiken, etwa situationsspezifischen Q&A-cycles
  - Software-technischen Implementierungskonzepten
  - Schnittstellendefinitionen, z.B. zu ARIS
  - Werkzeugen und Software-technischen Hilfsmitteln.

2.3 Phase 3: Konsolidierung

In dieser Phase werden zur Zeit folgende Aktivitäten gesetzt:

- Design eines Schulungsprodukts
  - Data Management
  - Business Logic
  - User Interface.

  Das Ergebnis dient der Implementierung einer skalierbaren und Software-technisch wie inhaltlich offenen Schulungsumgebung.
- Design von Schulungsszenarien basierend auf
  - Meta-Modellen
  - Kritischen Sprachkonstrukten und Modellierungssituationen.

  Dieser Design-Schritt dient der Weiterentwicklung und Vielfalt der entwickelbaren Schulungs- und Ausbildungsszenarien.
- Durchführung von Fallstudien im
  - Akademischen Anwendungsbereich
  - Unternehmensspezifischen Bereich.

  In der akademischen Lehre kann der Erfolg der Verständnisbildung und Ausbildungsbemühungen direkt im Rahmen der erforderlichen Leistungsüberprüfung gemessen werden, und zwar bevor im Rahmen wirtschaftsnaher Schulungen Aufwand in die Erstellung bestimmter Schulungsunterlagen, didaktischer Modelle und Werkzeuge investiert wird.

Daran angeschlossen werden die Implementierung und Release-Entwicklung der marktfähigen Version 1.

3 Ausgewählte Ergebnisse

In diesem Abschnitt werden erste Ergebnisse skizziert. Abschnitt 3.1 gibt erhobene Modellierungsprobleme sowie mögliche Ursachen wieder. Abschnitt 3.2 zeigt eine Anwendung szenariobasierter Frage/Antwort-Sequenzen anhand des entwickelten Prototypen des Werkzeuges.

3.1 Probleme der Modellierung (Phase 1)

Da die Ereignisgesteuerte Prozesskette in der Praxis immer mehr an Bedeutung gewinnt, wurde diese für die Untersuchung von Modellierungssprachen herangezogen (vgl. [NR02], [Aa98]).

Laut Davis sind 90 % der auftretenden Fehler bei der EPK-Modellierung

- Unangemessene Duplikation und Replikation von Objekten,
- Inkorrekte Modellierung von Entscheidungen und
- Fehler beim Anwenden der EPK-Methodik (vgl. [Da01]).

Weiters erwähnt Davis, dass Modellierer die Ausnahmepfade in den Prozessmodellen oft übersehen, da die Ereignisse nicht sinngemäß eingesetzt werden (vgl. [Da01]).

Abbildung 1 zeigt den Vorteil der Verwendung von Ereignissen. Modell a scheint korrekt zu sein. Meist erkennt der Modellierer erst nach Einfügen der Ereignisse (Modell b), dass es mehrere Prozesspfade geben kann. So sollte das Ergebnis das korrekte Modell c sein.

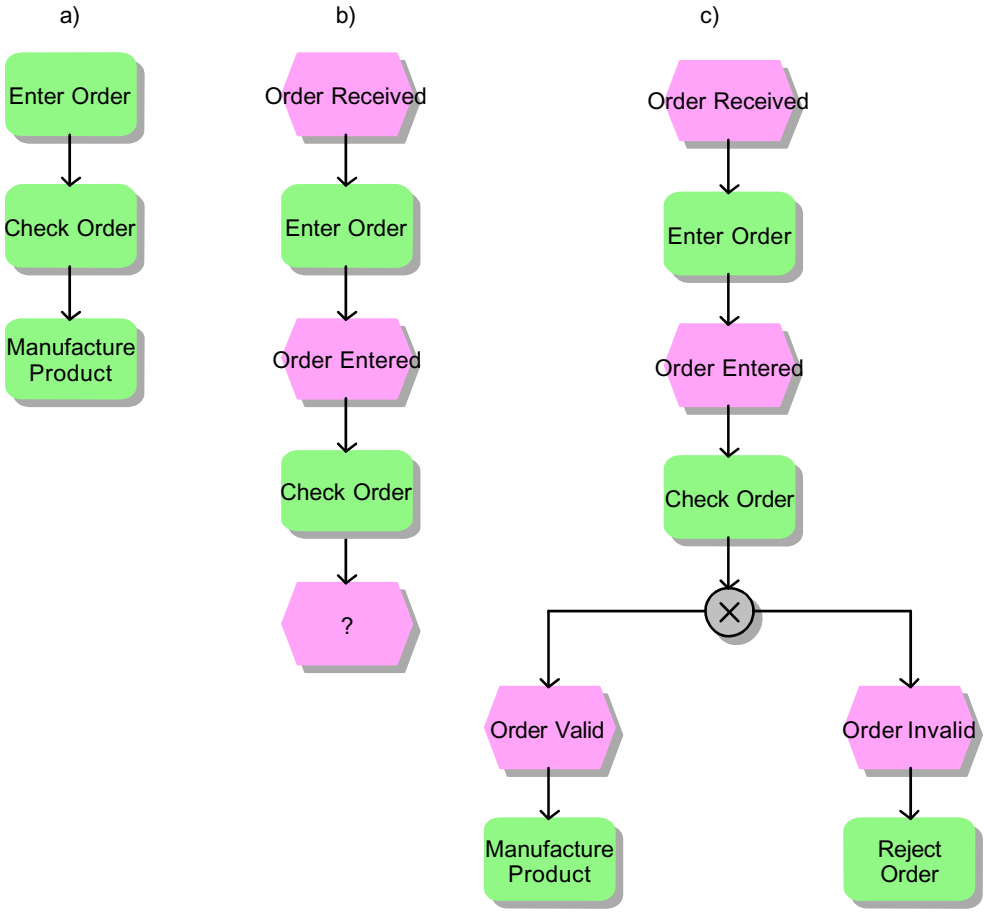


Abbildung 1. Vorteile der Ereignisse bei der EPK [Da01]

Laut der IDS-Hotline treten folgenden methodischen Probleme beim Einsatz der EPK am häufigsten auf (vgl. [He 02]):

- Wie können Prozessschnittstellen genutzt werden?
- Muss jedes Ereignis modelliert werden (Trivialereignisse)?
- Wie ist die maximale Größe eines Modells und wie modelliert man größere Modelle?

3.2 Prototyp des Ausbildungs- und Schulungswerkzeugs (Phase 2)

Den Fachexperten werden mittels Frage- und Antwortspiel kombiniert mit Szenarien die oben beschriebenen notwendigen Qualifikationen präsentiert. Die Szenarien implementieren dabei elementare und zusammengesetzte didaktische Konzepte. Bei der Erstellung der Szenarien wird zum einen von geclusterten Problemsituationen bei der Anwendung der EPK-Konstrukten, und zum anderen von abstrahierten EPK-Elementen ausgegangen, welche zu Situationsbeschreibungen kombiniert wurden, um für Modellierer einen realitätsnahen Einstieg zu definieren.

Abbildung 2 zeigt einen bereits im Einsatz befindlichen Prototypen (http://intelibus.ce.jku.at) des Schulungs- und Ausbildungswerkzeugs. Da dem Prototypen bereits ein didaktisches Framework (Abholen, notwendiger Input, Praxis und Reflexion) zugrunde liegt, wird zuerst der Wissensstand des Benutzers festgestellt (Abholen) und abhängig davon notwendige Inputs gegeben. Somit wird nach jeder Wissensakquisitions-Frage, Feedback mit der richtigen Antwort und einer Erläuterung gegeben um den Wissenstransfer zu verbessern. Nach einer praktischen Wissensüberprüfung mit Hilfe von Szenarien, wird nochmals am Ende jedes Kursmodules (z. B.: Ereignis, Funktion, Konnektoren, Situation, EPK, …) über die Inhalte reflektiert.

In welchem Modul bzw. Abschnitt sich der Benutzer gerade befindet, erkennt er in der Navigationsleiste links oben (Home-> Modul X -> Abschnitt Y). Das bedeutet, für jedes Kursmodul, gibt es mehrer Abschnitte, welche dem didaktischen Framework (Abholen, Input, …) entsprechen.

Rechts neben der Navigationsleiste befindet sich die Matrikelnummer des Benutzers und sein Status, welcher den sequentiellen Fortschritt im Lernprogramm widerspiegelt. 1/100 entspricht dem Anfang des Lehrprogrammes und 100/100 dem Ende.

Unten links befinden sich ein oder mehrere Szenarien, worauf die rechts befindliche Fragestellung bezug nimmt, um das Wissen des Benutzers zu messen.

In Abbildung 2 wird beispielsweise eine Situation (Ereignis-Funktion-Ereignis) mit einer Entscheidungsfunktion verwendet, welche aufgrund der Semantik einen XOR-Konnektor nach sich zieht und zusätzlich in einem Prozesspfad einen OR-Konnektor benötigt. Um eine richtige Antwort geben zu können, muss der Benutzer die unterschiedlichen Bedeutungen der Konnektoren (XOR, OR und ev. UND) kennen. Somit zielt diese Situation auf die Schulung der Semanik ab. Gibt der Benutzer nun eine

falsche Antwort, wird ihm das Modul Konnektoren (nochmals) erklärt und er bekommt am Ende des Modules in der Refelexionsrunde ein Szenario welches dasselbe Lehrziel abfragt.

In Abbildung 2 beginnt das Szenario mit dem Ereignis „Fahrradreifen zu prüfen“. Nach dem Ereignis folgt die Funktion „Fahrradreifen prüfen“, welche mehrere Ergebnisse haben kann bzw. muss. Entweder der Fahrradreifen ist „fahrbereit“ oder er ist „glatt“ oder /und „platt“ (also nicht fahrbereit).

Abbildung 2 Prototyp - Semantiktest

Das Szenario in Abbildung 3 überprüft das Syntax- und Semantikwissen mit ähnlicher Fragestellung wie in Abbildung 2. Es beginnt mit dem Ereignis „Reiseziel bestimmt“ und teilt sich dann entweder in die Funktion „schnellste Route fahren“ oder in die Funktion „kürzeste Route fahren“ auf und endet jeweils mit einem Ereignis („Ziel über schnellsten Weg erreicht“ bzw. „Ziel über kürzesten Weg erreicht“).

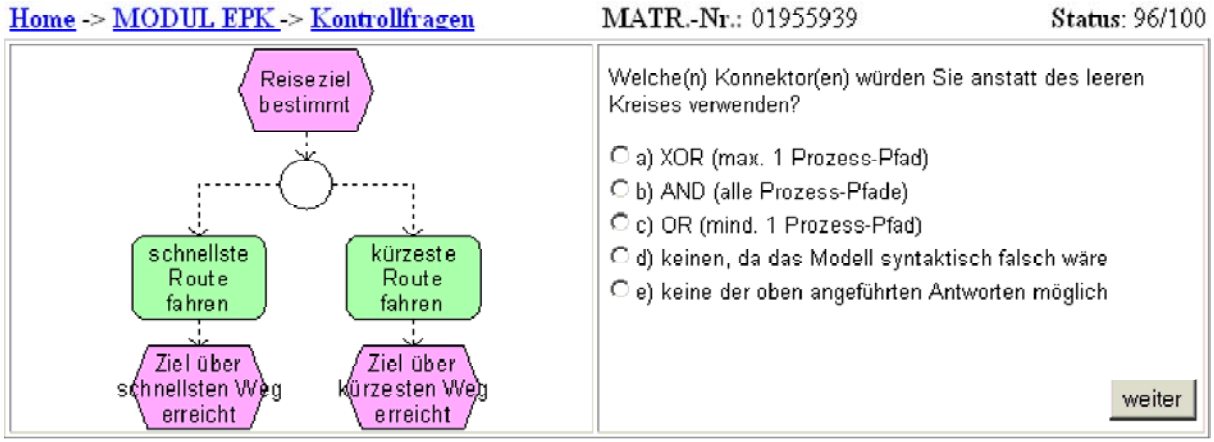


Abbildung 3 Prototyp – Syntax- und Semantiktest

Wenn der Benutzer die Antwort a wählt, hat er die Bedeutung des XOR-Konnektors verstanden (Semantiktest), auf die syntaktische Regel: „kein XOR-Konnektor nach einem Ereignis“ (vgl. [NR02]) muss jedoch noch eingegangen werden (Syntaxtest).

Die Antwort d ist zwar richtig, jedoch muss trotzdem anschließend mittels detaillierten Fragestellungen festgestellt werden ob er die entsprechende Regelverletzung „kein XOR-Konnektor nach einem Ereignis“ (Semantik- und Syntaxtest) erkannt hat. Somit ist dann auch eine Aussage über semantischen und syntaktisches Wissen möglich.

4 Schlussfolgerungen

Der Prototyping-Ansatz hat sich dahingehend bewährt, als bereits in den ersten drei Monaten des Projekts die szenario-basierte Interaktion mit Benutzern und die Schnittstelle zu ARIS erprobt werden konnten. Neben der Möglichkeit, Prototypen unmittelbar in Fallstudien einzusetzen, war es auch möglich, unterschiedliche fachdidaktische Ansätze zu evaluieren und zu adaptieren.

Insgesamt kann erwartet werden, dass durch den multi- und interdisziplinären Ansatz der Projektbemühungen ein erfolgreiches Aus- und Weiterbildungsprodukt entsteht, welchen seinen katalytischen Effekt dahingehend zeigt, dass die damit geschulten ModelliererInnen Unternehmenswissen sprachgerecht und unternehmensnahe in ARIS ausdrücken, und zwar sowohl bei der Analyse bestehender Organisationen, als auch bei der Weiterentwicklung bzw. beim Re-Engineering derselben.

Literaturverzeichnis

[Aa98] van der Aalst, W.M.P.: Formalization and Verification of Event-driven Process Chains, in: Backhouse, R.C.; Baeten, J.C.M. (Hrsg.): Computing Science Reports 98/01, Eindhoven University of Technology, Eindhoven 1998.

[BE01] Brücher, H.;Endl, R.: Erweiterung von UML zur geschäftsregelorientierten Prozessmodellierung 2001, http://www.wi.uni-muenster.de/is/tagung/ref2001/Kurzbeitrag13.pdf, Abruf: 2002-10-03

[BES98] Becker, J.; Ehlers, L.; Schütte, R.: Grundsätze ordnungsmäßiger Modellierung; Konzeption, Vorgehensmodelle. technische Realisierung, Nutzen. Münster, 1998, http://www.wi.uni-muenster.de/is/mitarbeiter/isresc/resc_Statutstagung.pdf, Abruf: 2002-10-03

[Da01] Davis, R.: Business Process Modeling with ARIS: A Practical Guide. London, 2001

[De01] Dehnert, J.: Four Systematic Steps Towards Sound Business Process Models, Proceedings of the 2nd International Colloquium on Petri Net Technologies for Modelling Communication Based Systems, Fraunhofer Gesellschaft ISST, Berlin 2001, pp. 55-64.

[DR01] Dehnert, J.; Rittgen, P.: Relaxed Soundness of Business Processes, Proceedings of the 13th Conference on Advanced Information Systems Engineering (CAISE'01) , Dittrich, K.L. and Geppert, A. and Norrie, M.C. (Eds.), Springer Verlag, LNCS 2068, Interlaken, Switzerland, 2001, S. 157-170.

[He02] Herzog, C.: Persönliche Kommunikation mit Herzog Christian/Call Center IDS. 2002

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

[Nü97] Nüttgens, M.: Ereignisgesteuerte Prozeßkette (EPK)-Forschungsansätze in der wissenschaftlichen Literatur und Praxis, in: Desel, J.; Reichel, H. (Hrsg.): Grundlagen der Parallelität, Workshop der GI-Fachgruppen 0.0.1 und 0.1.7 im Rahmen der INFORMATIK´97, Technische Berichte der TU Dresden, Fakultät Informatik, Dresden 1997.

[Ve96] Vernadat, F. B.: Enterprise Modeling and Integration: Principles and Applications. London, 1996