

Objektorientierte Analyse mit EMK

Peter Rittgen

Universität Koblenz-Landau

Rheinau 1, D-56068 Koblenz

E-Mail: rittgen@uni-koblenz.de

Abstract

Der vorliegende Beitrag stellt eine prozessorientierte Analysemethode vor, die verschiedene Sichten auf das betriebliche Informationssystem integriert: Prozess, Struktur und Ressourcen. Im Mittelpunkt steht dabei die Prozesssicht. Hier werden die Kernprozesse des Unternehmens identifiziert und beschrieben. Als Modellierungssprache wird die Ereignis-Methoden-Kette verwendet, die auf der weit verbreiteten EPK basiert und daher für potentielle Nutzer leicht erlernbar ist. Durch die konsequente Ausrichtung am objektorientierten Paradigma und die Desambiguierung der Syntax fördert die EMK aber im stärkeren Maße einen nahtlosen Übergang von der Analyse zum objektorientierten Entwurf. Dies wird unterstützt durch die Integration der EMK in das Objektmodell (Struktursicht).

1 Einleitung

In der frühen Phase der Analyse steht der Modellierer eines betrieblichen Informationssystems in der Regel noch einem weitgehend unstrukturierten Gegenstandsbereich gegenüber. Die hohe Komplexität dieses Gegenstands schließt es aus, ihn etwa unter Zuhilfenahme nur eines Modells als monolithisches System zu realisieren. Die aus diesem Grund notwendige Zerlegung des Ganzen in die zu modellierenden Teilwelten könnte dabei im Grunde entlang beliebiger Dimensionen erfolgen. In der Betriebswirtschaft steht jedoch eine Dimension im Vordergrund: die Spezialisierung (oder zunehmende Konkretisierung) von der strategischen Ebene über die taktische (oder organisatorische) zur operationalen (hier: Informationssystem). Eine weitere Aufteilung kann, auf jeder dieser Ebenen, bezüglich der folgenden sogenannten Foki erfolgen:

- **Struktur:** statische Beschreibung der Objekte des Gegenstandsbereichs und ihres Verhältnisses zueinander
- **Prozess:** dynamische Systemdarstellung
- **Ressourcen:** zur Ausführung von Prozessen benötigte Mittel
- **Ziele:** zu erreichende Ergebnisse und ihre Beziehungen

Die sich daraus ergebenden 12 Teilmodelle spannen die 4×3-Matrix der Sichten von MEMO (Multi-perspective Enterprise MOdeling, [Frank 97]) auf. Abb. 1 zeigt ikonisierte Beispiele für die 12 Sichten. Der erste Buchstabe der Modellbezeichnung steht dabei für die Ebene: (S)trategie, (O)rganisation und (I)nformationssystem, der zweite für den Fokus: (P)rozess, (S)truktur, (R)essourcen, und (Z)iele. So bedeutet beispielsweise SPM Strategisches Prozessmodell. Es besteht aus einer Wertkette á la Porter [Porter 85], wobei die primären Prozesse (Einkauf, Produktion, Vertrieb, Marketing) die Basiskette formen (pfeilförmige Kästchen) und die sekundären Aktivitäten an den unterstützten primären fixiert sind (Fähnchen). Auf den Ebenen Organisation und Informationssystem wird die Verwendung einer EPK-ähnlichen Sprache zur Prozessmodellierung angedeutet. Wieviele Sprachen insgesamt nötig sind, ist noch Gegenstand aktueller Forschung, d. h. die 7 in Abb. 1 angedeuteten Sprachen sind noch nicht endgültig.

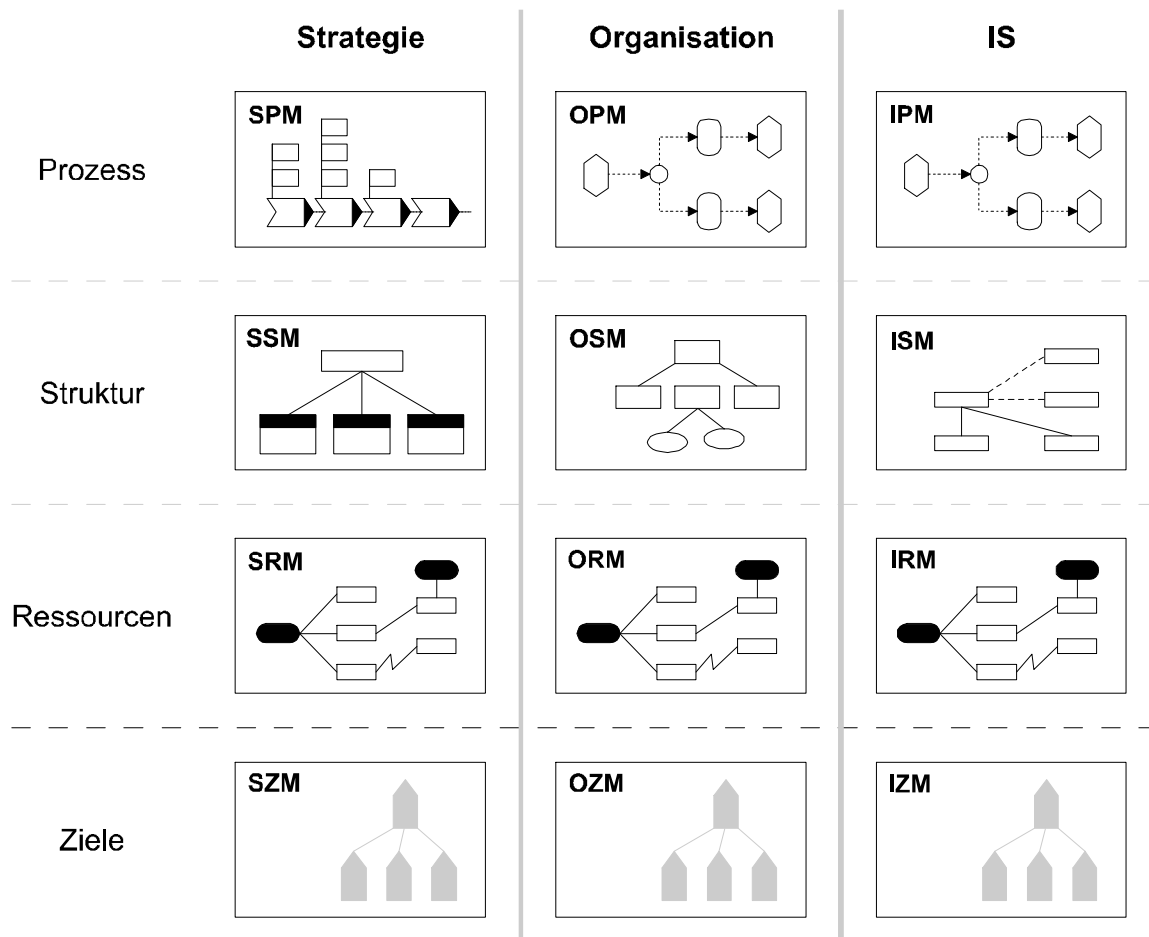


Abbildung 1: MEMO-Sichten auf ein Unternehmen

Im vorliegenden Beitrag versuchen wir nun, eine integrierte Modellierungssprache für die Informationssystemebene zu entwickeln, wobei wir vorerst von der Repräsentation von Zielen abstrahieren. Im Mittelpunkt unserer Betrachtungen steht dabei der Prozessfokus (siehe Abschnitt 3). Ein Ansatz hierzu ist die MEMO-OrgML (Organization Modeling Language, [Frank 99]), die neben der Ablauf- auch die Aufbauorganisation abdeckt. Der vorliegende Beitrag stellt eine Alternative zur Diskussion, die stärker die ablauforganisatorische Sicht betont, die sogenannte Ereignis-Methoden-Kette (MEMO-EMK). Die prozessualen Elemente dieses Modells, die Methoden, sind den jeweils zuständigen Objekten zugeordnet. Letztere sind im Objektmodell auf der Basis der MEMO-Object-Modeling-Language (MEMO-OML) beschrieben. Die zur Ausführung benötigten Ressourcen werden mittels Ressourcenzuordnungskanten annotiert. Der Zusammenhang zwischen den MEMO-Sichten und den Teilmodellen ist der Abbildung 2 zu entnehmen.

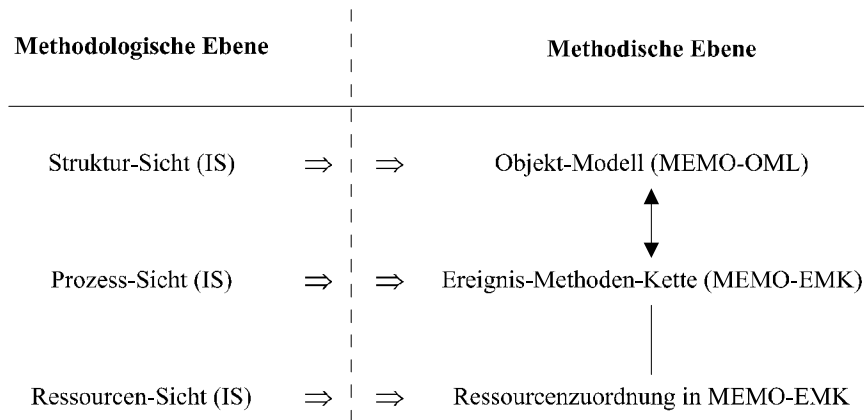


Abbildung 2: Repräsentation der Sichten in den Modellen

Für die Prozesssicht wurde eine Sprache gewählt, die den Ereignis-Prozess-Ketten (EPKn) der Architektur integrierter Informationssysteme (ARIS) von Scheer [Scheer 92] nicht unähnlich ist. Die Gründe hierfür sind vielfältig: Auf der praktischen Seite ist mit dem ARIS-Toolset schon seit längerem ein kommerzielles Werkzeug für EPKn verfügbar. Darüber hinaus trug nicht zuletzt der große Erfolg der betrieblichen Standardsoftware von SAP erheblich zu der Verbreitung dieser Methode bei. Auf der anderen Seite sind die EPKn aber auch wissenschaftlich recht gut untersucht, wie die folgenden Ausführungen noch zeigen werden.

Bei der Modellierung von Geschäftsprozessen mit ARIS werden wichtige Kernprozesse des Unternehmens identifiziert und mithilfe von EPKn dargestellt. Eine EPK besteht aus einer alternierenden Folge von Ereignissen (z. B. „Rechnung eingetroffen“) und Funktionen (z. B. „Rechnung buchen“). Zusätzlich können durch entsprechende Konnektoren alternative und parallele Abläufe beschrieben werden. Das Prozessmodell dient dabei der Dokumentation schon bestehender Abläufe, der Vorgabe neuer Abläufe, welche die Ziele der Unternehmung besser umsetzen sollen als die bisherigen, oder auch einer Kombination aus beiden. Falls die EPK neue Prozesse enthält, stellt sich die Frage, wie diese in einem Informationssystem umgesetzt werden sollen. Das „klassische“ Vorgehen besteht dabei in der Durchführung eines Reengineering-Projektes. In diesem Projekt werden die zu automatisierenden Teilprozesse identifiziert und in einem Software-Projekt realisiert.

Der zunehmende Anteil an automatisierten Prozessen führt jedoch zu entsprechend umfangreichen SW-Projekten, was eine schnelle Umsetzung erschwert. Dieses Dilemma wird entschärft, wenn man bereits bei der Analyse eine möglichst detaillierte und für die SW-

Entwicklung geeigneter Darstellung der Prozesse zuläßt und Fehler bei der Interpretation des Modells soweit wie möglich ausschließt (siehe Forderung 1). Eine weitere Unterstützung und Beschleunigung der SW-Erstellung kann auch durch frühzeitige Integration der von den Prozessen manipulierten Geschäftsobjekte erreicht werden (siehe Forderung 2). Hierbei wird unterstellt, daß die Implementation in einer objektorientierten Sprache erfolgen soll.

Aus dem Gesagten leiten sich also folgende Forderungen ab:

1. Syntax und Semantik der Modellierungssprache sollten möglichst präzise und reichhaltig sein, damit genügend Informationen für die Implementierung zur Verfügung steht und massive Änderungen bzw. Neuentwicklungen soweit möglich vermieden werden können.
2. Die benötigten Geschäftsobjekte sollten in das Modell integriert sein. Um die Programmierung in einer objektorientierten Programmiersprache zu erleichtern, sollten die Konzepte der Objektorientierung verwendet werden.

Im den folgenden Abschnitten wird dies am Beispiel der Ereignis-Prozess-Ketten als Prozessmodell verdeutlicht.

2 Von EPK zu EMK

Seit die EPKn von Scheer eingeführt wurden, gab es zahlreiche Ansichten darüber, wie denn eine „korrekte“ EPK auszusehen habe. Die Vorschläge erstreckten sich dabei sowohl auf die syntaktische Struktur (Welche Knoten dürfen wie miteinander verbunden werden?) als auch auf die Bedeutung der Strukturen (Was bewirkt ein Konnektor?). Auf der syntaktischen Ebene haben sich dabei einige Regeln etabliert, die heute allgemein anerkannt sind. Beispiele für diese Regeln [KT 97] sind:

- K1: Es existieren keine isolierten Knoten.
- K3/4: Funktionen und Ereignisse haben genau eine eingehende und eine ausgehende Kante (außer Start-/Endereignisse).
- K6: Konnektoren sind entweder Verteiler (1 Eingang, mehrere Ausgänge) oder Verknüpfen (mehrere Eingänge, 1 Ausgang).
- K8/9: Auf ein Ereignis folgt immer eine Funktion und umgekehrt (modulo Konnektoren).

Manchmal wird (wenn auch meist implizit) gefordert, daß jedem öffnenden Konnektor (auch Verteiler genannt) ein gleichartiger schließender (Verknüpfen) zugeordnet werden muß. Für

diesen Fall schlage ich eine Erweiterung der EPK-Syntax vor: Der Modellierer sollte die Konnektoren numerieren, wobei zusammengehörenden Konnektoren die gleiche Nummer zugeteilt wird. Diese sollte dann direkt am Konnektor vermerkt werden (siehe Abbildung 3, rechts). Es kann auch vorkommen, daß ein schließender Konnektor alleine auftritt, z. B. wenn eine EPK mehrere Startereignisse hat, die unabhängig voneinander den Ablauf triggern (siehe Abbildung 3, links). Dann kann im Zweifelsfalle die Bedeutung dieses Konnektors durch eine Kommentarfahne erläutert werden.

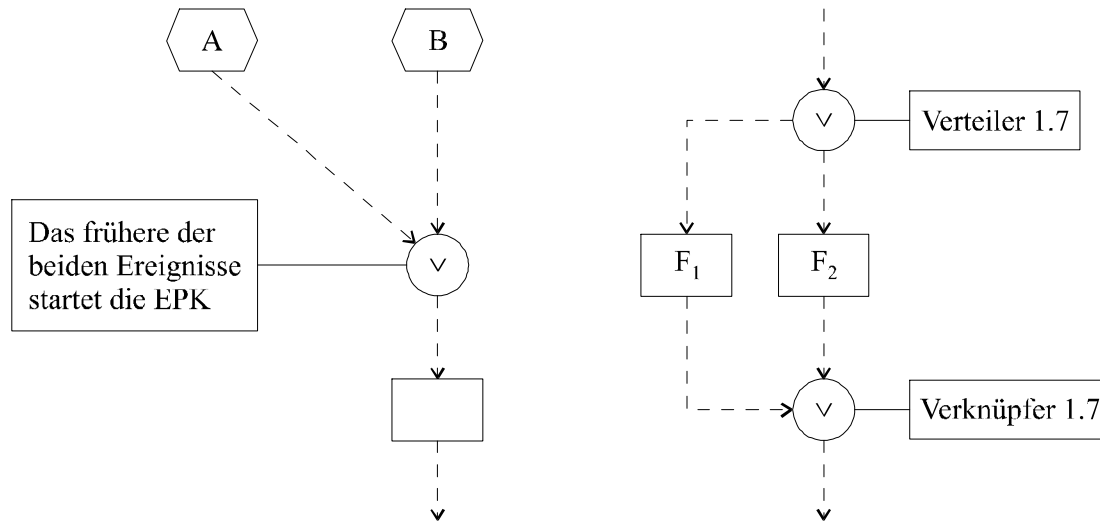


Abbildung 3: Beispiele für annotierte Konnektoren

Die einfachste Möglichkeit, einer EPK eine präzise und eindeutige, d. h. formale, Semantik zuzuordnen, bieten die Petrinetze. Abbildung 4 zeigt die S/T-Netze für die Konnektorenpaare XOR und UND. Beim öffnenden XOR schaltet entweder die obere oder die untere Transition und aktiviert damit den entsprechenden Pfad (gestrichelter Pfeil). Zusätzlich wird in jedem Fall eine Kontrollmarke auf die mittlere Stelle gelegt. Diese soll verhindern, daß der XOR-Verknüpfender ohne vorherige Aktivierung des Verteilers fehlerhaft ausgelöst wird, z. B. durch einen seitlichen Einsprung in einen der beiden Pfade. Der Verknüpfender schaltet erst dann, wenn beide Marken angekommen sind, also sowohl die Kontrollmarke als auch die Marke, die über den ausgewählten Pfad läuft. Der UND-Verteiler schickt je eine Marke über beide Pfade und eine auf die Kontrollstelle. Das schließende UND schaltet, wenn alle drei Marken vorliegen. Da die Kontrollmarke immer vom Verknüpfender direkt an den zugehörigen Verteiler übergeben wird, ist auch klar, daß die Zusammengehörigkeit syntaktisch repräsentiert sein muß, z. B. durch Numerierung mit der gleichen Nummer (siehe oben).

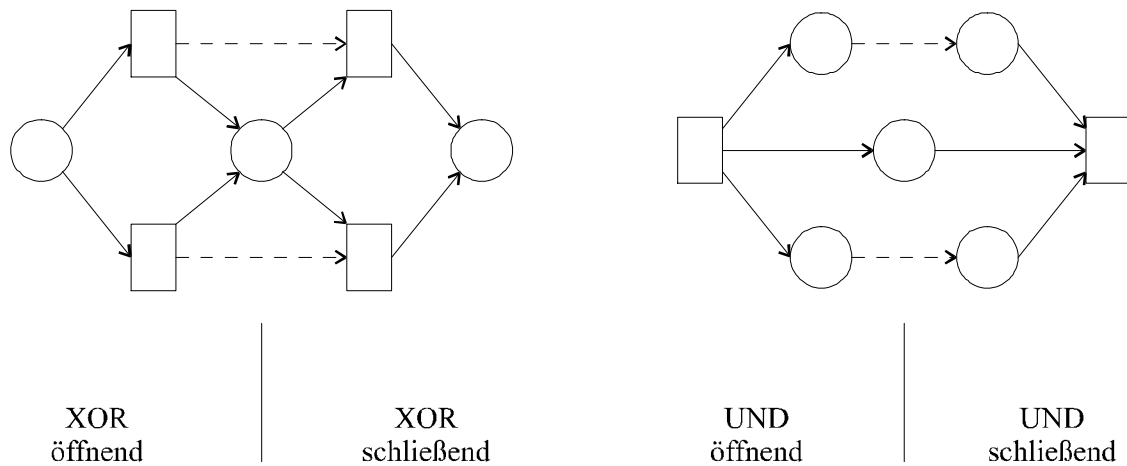


Abbildung 4: S/T-Netze der XOR- und UND-Konnektoren

Etwas schwieriger ist die Situation beim ODER-Konnektor: Hier genügt es nicht, daß der Verteiler dem Verknüpfer nur mitteilt, daß überhaupt etwas kommt. Das schließende ODER muß auch „wissen“, auf wieviele Marken es warten soll, weil der Verteiler einen oder beide Zweige aktiviert haben könnte. Aus diesem Grund haben Chen und Scheer in [CS 94] unterschiedliche Marken für die Zweige eingeführt. Hat man z. B. einen ODER-Verteiler mit zwei Pfaden, so wird über den einen Pfad die Marke „a“ und/oder über den anderen die Marke „b“ geschickt. Gleichzeitig teilt man dem ODER-Verknüpfer mit, welche Marken losgeschickt wurden, damit dieser „weiß“, auf was er warten muß.

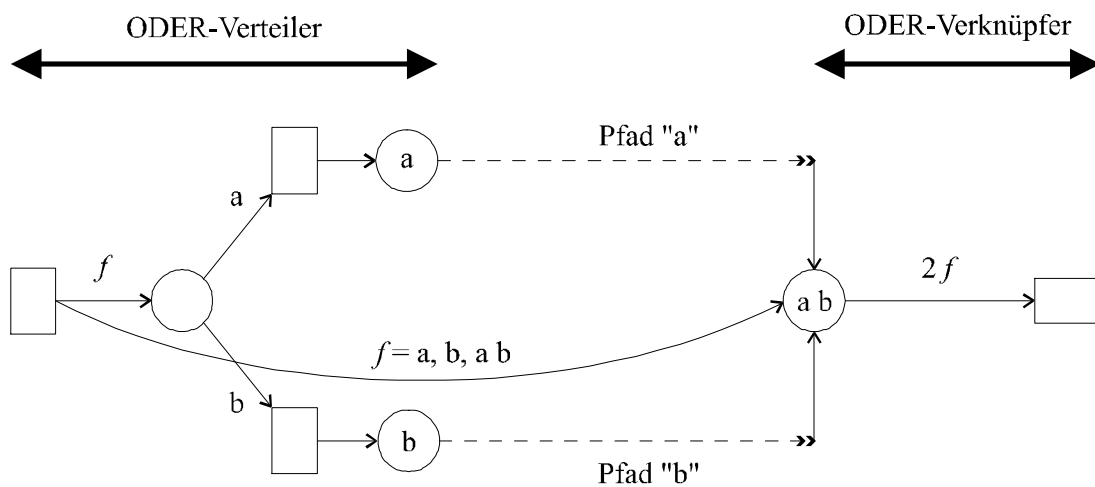


Abbildung 5: Semantik der ODER-Konnektoren nach [CS 94]

Bei dem Beispiel in Abbildung 5 soll der ODER-Verteiler beide Pfade aktivieren. f ist somit „a+b“, und die erste Transition legt die Marken „a“ und „b“ auf jede der beiden

Nachfolgerstellen. Die ersten beiden wandern über „ihren“ Pfad, also „a“ über den oberen, „b“ über den unteren, die beiden anderen signalisieren dem Verknüpfen, auf die „wandernden“ Marken aus Pfad „a“ **und** Pfad „b“ zu warten. Sind beide Pfade bearbeitet, dann liegen 4 Marken in der Stelle des ODER-Verknüpfers (nämlich 2 $f = 2$ a und 2 b) und die Transition schaltet. Dasselbe läßt sich auch mit gewöhnlichen S/T-Netzen erreichen (siehe [Ritt 99]).

Die dargestellte Semantik paßt, wie man leicht zeigen kann, exakt zu der vorgeschlagenen Syntax mit numerierte Konnektoren. Dadurch wird eine fehlerfreie Interpretation jeder EPK ermöglicht. Die EPK kann direkt in ein simulationsfähiges Petrinetz übersetzt werden, ohne daß die sonst üblichen Modifikationen (siehe z. B. [LSW 97]) nötig wären. Zudem kann das Petrinetz auch als Grundlage für die Implementation dienen, weil es hinreichend formal ist. Auf Basis einer Syntax mit numerierten Konnektoren und der vorgestellten Semantik können nun also Prozessmodelle leichter in Software umgesetzt werden.

Unter Berücksichtigung dieser syntaktischen Änderungen gelangt man zur Kontrollstruktur der Ereignis-Methoden-Kette (EMK). Diese bildet die Grundlage für die Prozesssicht von MEMO.

3 Die Prozesssicht

Eine schnelle und einfache Umsetzung der Kontrollstrukturen allein reicht aber nicht aus, um aus Modellen komplexer Geschäftsprozesse Software herzustellen. Vielmehr sind bei nahezu allen betrieblichen Prozessen immer auch eine Reihe von Geschäftsobjekten involviert (wie z. B. Dokumente). Berücksichtigt man diese nicht schon von Anfang an im Prozessmodell, so wird eine nachträgliche Integration sehr schwierig und erfordert in der Regel umfangreiche Änderungen des Prozessmodells bis hin zu einer Neuentwicklung. Aus diesem Grund wurden die EPKn von Scheer, Nüttgens und Zimmermann in [SNZ 97] objektorientiert erweitert. Das Grundprinzip dieser oEPKn lautet, daß ein Ereignis nun nicht mehr eine Funktion sondern eine Objektklasse auslöst. Diese Klasse kümmert sich um die Behandlung des Ereignisses und aktiviert die entsprechenden Methoden, welche rechts an der Klasse annotiert werden. Die dazu benötigten Attribute werden auf der linken Seite aufgeführt. Die Syntax für eine oEPK zeigt Abbildung 6.

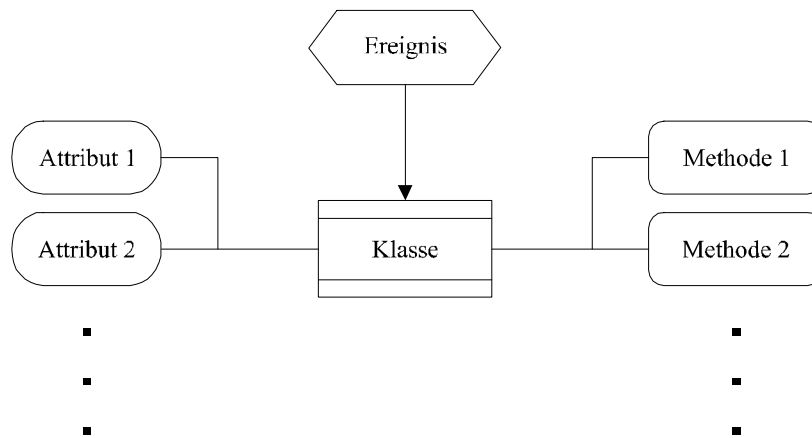


Abbildung 6: Syntax einer oEPK

Das Konzept einer Klasse, die von einem Ereignis ausgelöst wird, erscheint kontraintuitiv. Zudem fehlt in dieser Art der Darstellung die logische Ablaufstruktur der Methodenaufrufe. Der Aufbau dieser Struktur wird in dem geschilderten Ansatz auf die späteren Phasen der Softwareentwicklung verschoben. Die dort entdeckten Schwachstellen wie z. B. fehlende Methoden oder solche, die in der spezifizierten Form nicht realisierbar sind, machen dann wiederum eine Überarbeitung der oEPK und damit einen Rücksprung in die Modellierungsphase nötig. Um solche Zyklen zu vermeiden, sollten die Methoden und nicht die Klassen den Platz der Funktionen einnehmen. Dies entspricht auch der Intuition, daß ein Ereignis einen Vorgang auslöst, nämlich eine Methode. Die zur Methode gehörende Klasse wird über eine Kante angebunden. Ebenso werden alle Attribute angeschlossen. Die Klasse selbst taucht nur ein einziges Mal auf. Die Syntax ist in Abbildung 7 dargestellt.

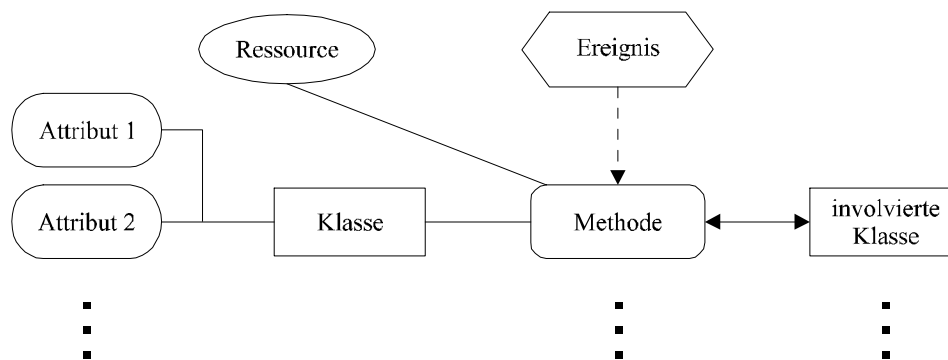


Abbildung 7: Syntax einer EMK

Im Zentrum der EMK steht die Klasse eines Objekts, denotiert durch ein Rechteck. Sie wird in der Struktursicht (Objektmodell) definiert (siehe Abschnitt 4) und hier nur referenziert,

wobei eine Vorwärts-Referenz im Sinne einer späteren Definition im Objektmodell zulässig ist. Bei Bedarf kann der Modellierer Attribute für diese Klasse spezifizieren, falls dies noch nicht im Objektmodell geschehen ist. In der Regel sind das solche, die sich aus dem aktuellen Anwendungskontext erst ergeben. Da es sich um ein Analysemodell handelt, müssen diese nicht auch notwendigerweise Attribute des Entwurfs- oder Implementationsmodells sein. (z. B. Alter einer Person).

Jeder zu bearbeitende Vorgang (hier Methode genannt) wird ebenso wie ein Attribut über eine einfache Kante mit derjenigen Klasse assoziiert, die diese Methode zur Verfügung stellt. Sie wird durch ein abgerundetes Rechteck repräsentiert. Dabei können zur Bearbeitung einer Methode auch Methoden anderer Klassen benutzt werden. Solche involvierten Klassen können über einen Doppelpfeil mit der entsprechenden Methode verknüpft werden. Mithilfe gestrichelter Einfachfeile sind Methoden außerdem in die Kontrollstruktur aus Ereignissen und Methoden eingebunden. Etwaig für die Ausführung benötigte Ressourcen (Ellipsen) können an der Methode annotiert werden.

Ein Beispiel für eine solche „Ereignis-Methoden-Kette“ (EMK) zeigt Abbildung 8.

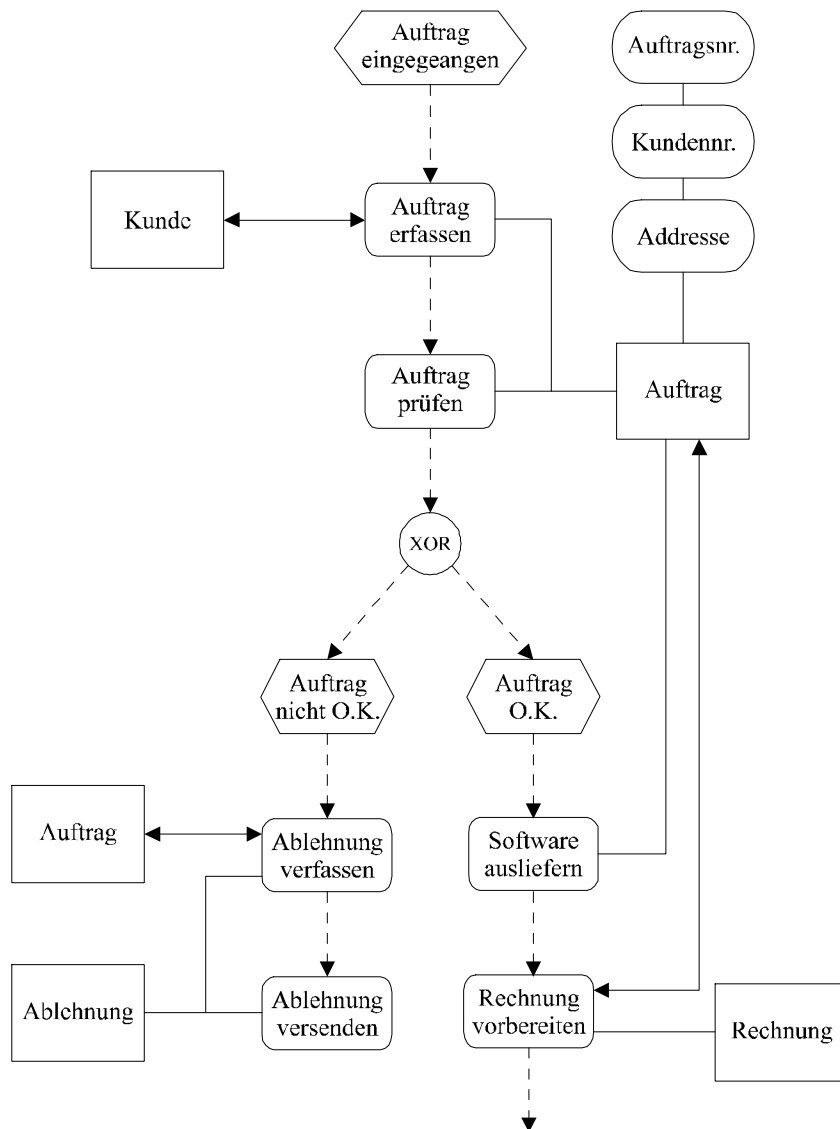


Abbildung 8: Ereignis-Methoden-Kette des Anwendungsbeispiels

Die Syntax einer EMK entspricht im wesentlichen der einer EPK. Allerdings erlaubt die EMK das Aufeinanderfolgen mehrerer Methoden ohne dazwischenliegende Ereignisse und umgekehrt (siehe Abbildung 9). Nach der Eintreffen einer Rechnung wird diese überprüft. Da dieser Vorgang in unserem Beispiel manuell erfolgt, abstrahiert der Anwender in seinem Modell für das IS folgerichtig davon. Analoges gilt für das Ereignis „Rechnung erfaßt“.

Außerdem sind zusammengehörnde Konnektoren mit dem gleichen Etikett versehen (K1 in Abbildung 8). Eine EMK verfügt auch über eine reichhaltigere Semantik: Eine oEPK für das Beispiel in Abbildung 8 in etwa ließe nur erkennen, daß die Klasse „Auftrag“ die Klasse

„Kunde“ benötigt, in der EMK wird darüber hinaus klar, an welcher Stelle sie gebraucht wird: nämlich bei der Erfassung des Auftrags (Übernahme der Kundennummer).

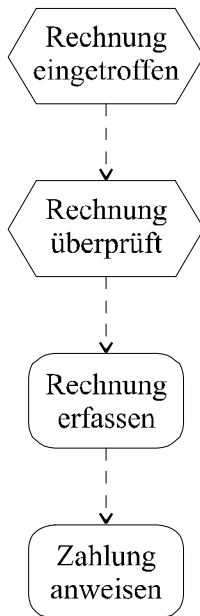


Abbildung 9: Einfache EMK für die Bearbeitung einer Eingangsrechnung

Bei einer oEPK existiert eine Klasse zudem in der Regel an mehreren Stellen des Diagramms. Zur eigentlichen Definition der Klasse muß man also die Attribute und Methoden an allen diesen Stellen zusammensuchen. Mehrfachnennungen von Attributen und Methoden sind dadurch unausweichlich. Damit werden die Diagramme mit Redundanzen überfrachtet und Fehler durch unterschiedliche Schreibweisen desselben Attributs provoziert. In der EMK wird jedes Attribut und jede Klasse nur genau einmal erwähnt. Dies ist hilfreich beim Aufbau des Objektmodells der Struktursicht.

4 Die Struktursicht

Das Ziel der Struktursicht besteht in der Darstellung der invariablen Teile des Informationssystems. Die in der Prozesssicht identifizierten Objekte werden hier in Beziehung zueinander gebracht. Fehlende, strukturbildende Klassen (z. B. abstrakte Klassen) können dabei noch ergänzt und das dynamische Modell konsolidiert werden. Ein Objektmodell für das Beispiel aus Abbildung 8 zeigt Abbildung 10. Es wurde in der MEMO-Object-Modeling-Language (MEMO-OML) abgefaßt. Eine detaillierte Darstellung dieser Sprache findet man in [Frank 98]. Die Subklassenbeziehung wird mit einfachen Pfeilen von der Unter- zu Oberklasse

dargestellt. Assoziationen sind einfache Kanten, deren Beschriftung die Art der Beziehung kennzeichnet. Ein Dreieck gibt dabei die Lesrichtung an.

Geht man von den Objekten aus, die sich unmittelbar aus dem Kontext des Anwendungsbeispiels der Abbildung 8 ergeben, so enthält das Objektmodell zunächst nur Kunde, Auftrag, Rechnung und Ablehnung als isolierte Objekte. Bei näherer Betrachtung dieser Objekte fällt auf, daß es sich bei den drei Letztgenannten um Dokumente handelt. Dies legt die Bildung einer gemeinsamen Oberklasse diesen Namens nahe. Im Bereich Auftragsverwaltung gibt es aber einen wesentlichen Unterschied zwischen diesen Dokumenten: Während der Auftrag ein eingehendes Dokument ist, sind Rechnung und Ablehnung ausgehende. Da eine Rechnung den Auftrag im positiven Sinne (der Erfüllung) bestätigt und die Ablehnung im negativen, werden beide unter der Oberklasse Bestätigung zusammengefaßt (von einer herkömmlichen Auftragsbestätigung soll hier abgesehen werden). Bestätigung und Auftrag sind dann die Unterklassen von Dokument. Auf der Basis dieser Klassenstruktur ist es nun auch möglich, die Beziehungen des Kunden zu den Dokumententypen zu etablieren: Der Kunde „erteilt“ den Auftrag und „erhält“ die Bestätigung (Ablehnung oder Rechnung). In einem weiteren Schritt identifiziert man vielleicht noch den Kunden als einen speziellen Geschäftspartner neben dem Lieferanten.

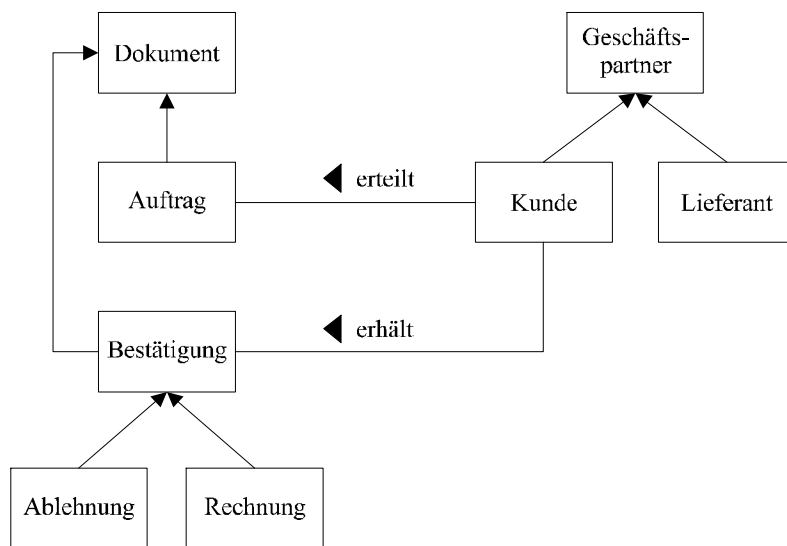


Abbildung 10: Ausschnitt aus einem Objektmodell in MEMO-OML

5 Metamodelle

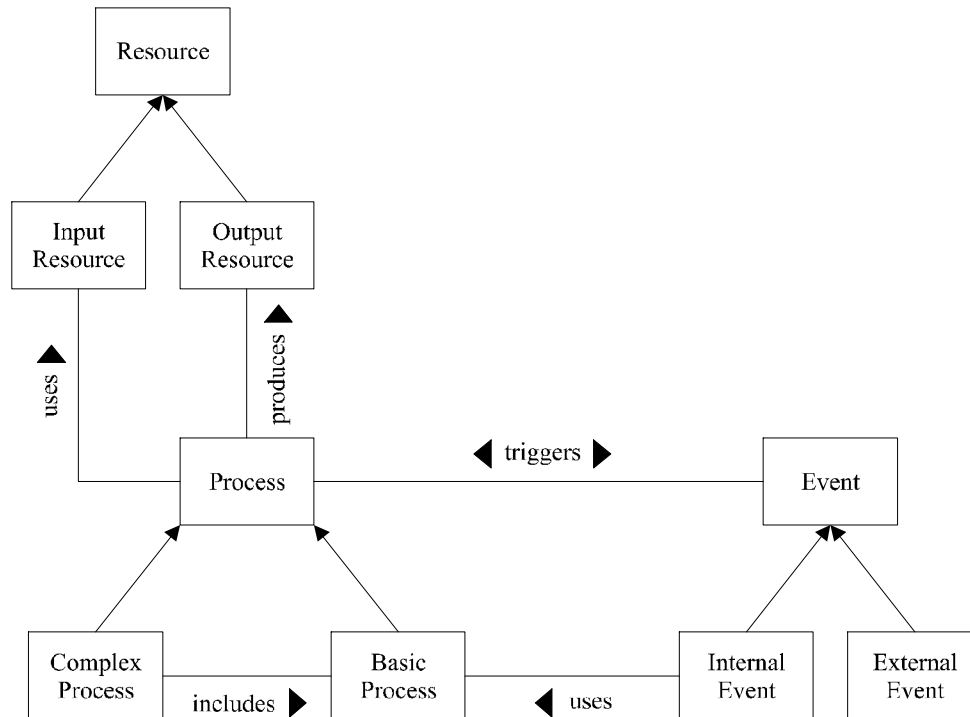


Abbildung 11: Metamodell der Prozess- und Ressourcensicht

Das Metamodell für die Struktursicht wurde bereits in [Frank 97] eingehend dargestellt. Ein Ausschnitt aus dem Metamodell für die Prozess- und Ressourcensichten zeigt Abbildung 11. Ein Prozess ist entweder atomar (BasicProcess) oder zusammengesetzt aus anderen Prozessen (ComplexProcess). Bei diesen Prozessobjekten handelt es sich immer um *Prozesstypen*. Wird ein Prozesstyp verwendet, dann spricht man von einem Ereignis (siehe [Fracz 95]): Wenn man also z. B. den Prozesstyp „bohren“ zweimal hintereinander ausführt oder auszuführen plant, so erhält man die Ereignisse „1. Loch bohren“ und „2. Loch bohren“. Ereignisse, die innerhalb des Informationssystems geschehen, heißen interne Ereignisse, die übrigen externe. Startereignisse lösen Prozesse aus, Endereignisse werden von Prozessen ausgelöst. Prozesse können Input-Ressourcen benutzen und Output-Ressourcen erzeugen.

6 Ausblick

Der vorliegende Beitrag stellt eine prozessorientierte Analysemethode vor, die verschiedene Sichten auf das betriebliche Informationssystem integriert: Prozess, Struktur und Ressourcen. Im Mittelpunkt steht dabei die Prozesssicht. Hier werden die Kernprozesse des Unternehmens

identifiziert und beschrieben. Als Modellierungssprache wird die Ereignis-Methoden-Kette verwendet, die auf der weit verbreiteten EPK basiert und daher für potentielle Nutzer leicht erlernbar ist. Durch die konsequente Ausrichtung am objektorientierten Paradigma und die Desambiguierung der Syntax fördert die EMK aber im stärkeren Maße einen nahtlosen Übergang von der Analyse zum objektorientierten Entwurf. Dies wird unterstützt durch die Integration der EMK in das Objektmodell (Struktursicht).

Dennoch sind damit die Probleme bei der Modellierung betrieblicher Systeme noch nicht vollständig gelöst. Die Akzeptanz der EPK (und damit wahrscheinlich auch der EMK) ist nach [Speck 98] auf der Ebene der Geschäftsführung bzw. des Projektleitungsgremiums gering. Dort wird ein solcher Ansatz als zu formal empfunden. Hier muß die oEPK/EMK noch an die Bedürfnisse solcher Benutzer angepaßt werden. Darüber hinaus muß ein Ansatz zur Unternehmensmodellierung auch die Ebenen Organisation und Strategie berücksichtigen. In diesen Problemen sehen wir einen wesentlichen Kern zukünftiger Forschung.

Literatur

- [CS 94] Chen, R.; Scheer, A.-W.: *Modellierung von Prozessketten mittels Petri-Netz-Theorie*, IWi-Heft 107, Institut für Wirtschaftsinformatik, Universität Saarbrücken, 1994
- [Fracz 95] Fraczak, W.: *Multi-Action Process Algebra*, in: Kanchanasut, K.; Lévy, J.-J.: *Algorithms, Concurrency and Knowledge*, 1995 Asian Computing Science Conference ACSC '95, Pothumthani, Thailand, 11.-13. Dezember 1995, Proceedings, Lecture Notes in Computer Science 1023, Springer, Berlin, 1995, S. 126-140
- [Frank 97] Frank, U.: *Enriching Object-Oriented Methods with Domain Specific Knowledge: Outline of a Method for Enterprise Modelling*. Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 4, Universität Koblenz-Landau, 1997
- [Frank 98] Frank, U.: *The Memo Object Modelling Language (MEMO-OML)*, Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 10, Universität Koblenz-Landau, 1998

- [Frank 99] Frank, U.: *Eine Architektur zur Spezifikation von Sprachen und Werkzeugen für die Unternehmensmodellierung*, in: Rundbrief des GI-Fachausschusses 5.2, 6. Jahrgang, Heft XX, Bamberg, September 1999, S. XX - XX
- [KT 97] Keller, G.; Teufel, Th.: *SAP R/3 prozessorientiert anwenden - iteratives Prozess-Prototyping zur Bildung von Wertschöpfungsketten*, Addison-Wesley, Bonn, 1997
- [LSW 97] Langner, P.; Schneider, Ch.; Wehler, J.: *Prozessmodellierung mit Ereignis-gesteuerten Prozessketten (EPKs) und Petri-Netzen*, WIRTSCHAFTS-INFORMATIK, 39 (1997) 5, S. 479-489
- [Porter 85] Porter, M.E.: *Competitive Advantage: Creating and Sustaining Superior Performance*. Simon & Schuster, New York, 1985
- [Ritt 99] Rittgen, P.: Quo vadis EPK in ARIS ? – Ansätze zu syntaktischen Erweiterungen und einer formalen Semantik, erscheint in: WIRTSCHAFTS-INFORMATIK, 1999
- [Scheer 92] Scheer, A.-W.: *Architektur integrierter Informationssysteme*, 2. Auflage, Springer, Berlin, 1992
- [SNZ 97] Scheer, A.-W.; Nüttgens, M.; Zimmermann, V.: *Objektorientierte Ereignis-gesteuerte Prozesskette (oEPK) – Methode und Anwendung*, IWi-Heft 141, Institut für Wirtschaftsinformatik, Universität Saarbrücken, 1997
- [Speck 98] Speck, M.: *Akzeptanz und Operationalität von EPK in der Modellierungspraxis – ein Erfahrungsbericht aus einem Reengineering-Projekt*, Arbeitskreistreffen Formalisierung der EPK, Münster, 1998-03-17, http://www-is.informatik.uni-oldenburg.de/~epk/treffen_170398/speck.ps, Abruf am 1998-08-11