universität mainz

LEHRSTUHL FÜR
ALLG. BWL UND WIRTSCHAFTSINFORMATIK
UNIV.-PROF. DR. HERBERT KARGL

Dandl, Jörg

Objektorientierte Prozeßmodellierung mit der UML und EPK

ARBEITSPAPIERE WI
Nr. 12/1999

Schriftleitung:
Dr. rer. pol. Axel C. Schwickert

Information

Reihe: Arbeitspapiere WI

Herausgeber: Univ.-Prof. Dr. Axel C. Schwickert
Professur für BWL und Wirtschaftsinformatik
Justus-Liebig-Universität Gießen
Fachbereich Wirtschaftswissenschaften
Licher Straße 70
D – 35394 Gießen
Telefon (0 64 1) 99-22611
Telefax (0 64 1) 99-22619
eMail: Axel.Schwickert@wirtschaft.uni-giessen.de
http://wi.uni-giessen.de

Bis Ende des Jahres 2000 lag die Herausgeberschaft bei:

Lehrstuhl für Allg. BWL und Wirtschaftsinformatik
Johannes Gutenberg-Universität Mainz
Fachbereich Rechts- und Wirtschaftswissenschaften
Welderweg 9
D - 55099 Mainz

Ziele: Die Arbeitspapiere dieser Reihe sollen konsistente Überblicke zu den Grundlagen der Wirtschaftsinformatik geben und sich mit speziellen Themenbereichen tiefergehend befassen. Ziel ist die verständliche Vermittlung theoretischer Grundlagen und deren Transfer in praxisorientiertes Wissen.

Zielgruppen: Als Zielgruppen sehen wir Forschende, Lehrende und Lernende in der Disziplin Wirtschaftsinformatik sowie das IuK-Management und Praktiker in Unternehmen.

Quellen: Die Arbeitspapiere entstanden aus Forschungsarbeiten, Diplom-, Studien- und Projektarbeiten sowie Begleitmaterialien zu Lehr- und Vortragsveranstaltungen des Lehrstuhls für Allg. Betriebswirtschaftslehre und Wirtschaftsinformatik Univ. Prof. Dr. Herbert Kargl an der Johannes Gutenberg-Universität Mainz.

Hinweise: Wir nehmen Ihre Anregungen und Kritik zu den Arbeitspapieren aufmerksam zur Kenntnis und werden uns auf Wunsch mit Ihnen in Verbindung setzen.
Falls Sie selbst ein Arbeitspapier in der Reihe veröffentlichen möchten, nehmen Sie bitte mit dem Herausgeber (Gießen) unter obiger Adresse Kontakt auf.
Informationen über die bisher erschienenen Arbeitspapiere dieser Reihe und deren Bezug erhalten Sie auf dem Schlußblatt eines jeden Arbeitspapiers und auf der Web Site des Lehrstuhls unter der Adresse http://wi.uni-giessen.de

Arbeitspapiere WI Nr. 12/1999

Autor: Dandl, Jörg

Titel: Objektorientierte Prozeßmodellierung mit der UML und EPK

Zitation: Dandl, Jörg: Objektorientierte Prozeßmodellierung mit der UML und EPK, in: Arbeitspapiere WI, Nr. 12/1999, Hrsg.: Lehrstuhl für Allg. BWL und Wirtschaftsinformatik, Johannes Gutenberg-Universität: Mainz 1999.

Kurzfassung: Die objekt- und prozeßorientierte Modellierung sind zwei zentrale Konzepte in der Analyse-, Entwurfs- und Implementierungsphase eines Softwareentwicklungsprozesses. Obwohl beide unterschiedlichen Paradigmen folgen, existieren Ansätze, sie miteinander zu verknüpfen, denn: objektorientierte Modelle decken nicht alle Aspekte prozeßorientierter Modelle ab, während prozeßorientierte Darstellungsformen wie die EPK des ARIS-Konzeptes Schwächen in der implementierungsnahen Modellierung aufweisen. Das vorliegende Arbeitspapier soll einen Überblick zum aktuellen Stand der objektorientierten Prozeßmodellierung geben. Zunächst wird die objektorientierte Modellierungssprache Unified Modeling Language (UML) mit ihren Sprachelementen und ausgewählten Diagrammtypen vorgestellt. Danach wird eine Möglichkeit zur Integration von UML-Elementen in die Notation der ereignisgesteuerten Prozeßkette (EPK) aufgezeigt und abschließend beispielhaft an einem Prozeß aus der forschenden pharmazeutischen Industrie umgesetzt.

Schlüsselwörter: Objektmodellierung, Prozeßmodellierung, UML, Unified Modeling Language, EPK, Ereignisgesteuerte Prozeßkette, ARIS, Klassendiagramm, Zustandsdiagramm, Anwendungsfalldiagramm, Aktivitätsdiagramm

Inhaltsverzeichnis

1 Ziel und Aufbau 3

2 Die objektorientierte Modellierungssprache UML 3

2.1 Grundlagen und Terminologie 3

2.2 Klassendiagramm 6

2.3 Zustandsdiagramm 8

2.4 Anwendungsfalldiagramm 10

2.5 Aktivitätsdiagramm 12

3 Objektorientierte Prozeßmodellierung 14

3.1 Einführung 14

3.2 EPK und Klassendiagramm 15

3.3 EPK und Zustandsdiagramm 17

3.4 EPK und Anwendungsfalldiagramm 18

4 Praxisbeispiel 19

4.1 Einführende Erläuterungen 19

4.2 Objektorientierte Prozeßmodellierung am Beispiel des Patentrechercheprozesses 21

Literaturverzeichnis 24

1 Ziel und Aufbau

Die objekt- und prozeßorientierte Modellierung sind zwei zentrale Konzepte in der Analyse-, Entwurfs- und Implementierungsphase eines Softwareentwicklungsprozesses. Obwohl beide unterschiedlichen Paradigmen folgen, existieren Ansätze, sie miteinander zu verknüpfen. Objektorientierte Modelle decken nicht alle Aspekte prozeßorientierter Modelle ab, da z. B. in einem Aktivitätsdiagramm der Unified Modeling Language (UML) keine Zuordnung von Daten zu Aktivitäten möglich ist, die Abbildung von Geschäftsregeln nur eingeschränkt darstellbar ist und aktivitätsauslösende Ereignisse nur indirekt über Zustandstransitionen modelliert werden können.[1] Prozeßorientierte Darstellungsformen wie die ereignisgesteuerte Prozeßkette (EPK) des ARIS-Konzeptes (Architektur integrierter Informationssysteme)[2] zeigen dagegen Schwächen in der implementierungsnahen Modellierung.

Das vorliegende Arbeitspapier soll einen Überblick zum aktuellen Stand der objektorientierten Prozeßmodellierung geben. Zunächst wird die objektorientierte Modellierungssprache Unified Modeling Language (UML) mit einigen ausgewählten Sprachelementen und Diagrammtypen vorgestellt. Danach wird eine Möglichkeit zur Integration von UML-Elementen in die Notation der ereignisgesteuerten Prozeßkette (EPK) aufgezeigt und abschließend in Kapitel 4 an einem Prozeß aus der forschenden, pharmazeutischen Industrie, der mit einem Dokumenten-Management-Systems (DMS) informationstechnisch unterstützt werden soll, exemplarisch umgesetzt.

2 Die objektorientierte Modellierungssprache UML

2.1 Grundlagen und Terminologie

Die Unified Modeling Language (UML) ist eine graphische Modellierungssprache zur objektorientierten Modellierung eines zu entwickelnden Anwendungssystems beliebiger Komplexität. Sie ist der Nachfolger der bis 1996 ehemals eigenständigen Methoden von

1 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), in: Arbeitsberichte des Instituts für Wirtschaftsinformatik, Heft 144/1998, Hrsg.: Scheer, August-Wilhelm, Saarbrücken 1998, S. 8.

2 Vgl. Scheer, August-Wilhelm: ARIS – Vom Geschäftsprozeß zum Anwendungssystem, Anwendungssystem, 3., völlig überarb. und erw. Aufl., Berlin et al.: Springer 1998. S. 39 f.

Booch, Jacobson und Rumbaugh,[3] wobei von diesen Design- und Analysemethoden die Sprachelemente, also die Notation (Modellierungstechnik), nicht aber die Methoden[4] selbst übernommen wurden.[5] Dies impliziert, daß für die Anwendung der UML innerhalb eines Softwareentwicklungsprozesses kein integrierter Leitfaden existiert, der vorgibt, was, wann und in welcher Reihenfolge getan werden muß, um ein Anwendungssystem zu entwickeln.

Die Sprachelemente der UML bieten die Möglichkeit, strukturelle, verhaltens- und zustandsorientierte Aspekte objektorientierter Modelle zu beschreiben. In diesem Zusammenhang unterscheidet die UML bei der Betrachtung eines zu entwickelnden Anwendungssystems zwischen verschiedenen Modellsichten.

In der Basissicht werden die Notationen des Objektes, der Klasse, der Operation und des Attributes beschrieben. In den statischen Sichten sind die Prinzipien Vererbung, Assoziation, Aggregation und Abstraktion dargestellt. Sie beschreiben das zu entwickelnde System mit seinen sich nicht verändernden Aspekten. Modelliert werden die grundlegenden Aktivitäten und Aufgaben, die das zukünftige System adäquat unterstützen soll (das „was“). Die dynamischen Sichten beschreiben das Anwendungssystemverhalten im Zeitverlauf (das „wie“). Dargestellt wird hier z. B. der Kommunikationsfluß (Austausch von Nachrichten mit Operationen) über die in den statischen Sichten definierten Kommunikationswege.[6]

Zudem verfügt die UML über eine große Anzahl von Konzepten, die sowohl für die Systemanforderungs- und Analysephase als auch für die Entwurfs- und Implementierungsphase[7] geeignet sind.[8] Darüber hinaus ist die UML-Version 1.1 seit November 1997 ein

---

3 Dies sind die Methoden “Object-Oriented Software-Engineering (OOSE)“ von Jacobson, die „Object Modeling Technique (OMT)“ von Rumbaugh und die „Object-Oriented Analysis and Design“ von Booch. Vgl. Object Management Group (Hrsg.): OMG Unified Modeling Language Specification (draft), Version 1.3 alpha R2, 1/1999, S. 1-12 und Schäfer, Steffen: Objektorientierte Entwurfsmethoden: Verfahren zum objektorientierten Systementwurf im Überblick, , Bonn et al.: Addison-Wesley 1994, S. 71 ff.

4 Es ist anzumerken, daß bei der Entwicklung von Applikationen zumeist nur die Modellierungssprache als ein Bestandteil einer Methode zum Einsatz kommt. Der Prozeßaspekt, der die Handlungsanweisung zur systematischen Vorgehensweise für die Anwendungsentwicklung enthält, ist häufig nur skizzenhaft. Vgl. Fowler, Martin; Scott, Kendall: UML konzentriert: Die neue Standard-Objektmodellierungssprache anwenden, Bonn: Addison-Wesley 1998, S. 17.

5 Vgl. Fowler, Martin; Scott, Kendall: UML konzentriert: Die neue Standard-Objektmodellierungssprache anwenden, a. a. O., S. 17.

6 Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; in: Informatik Spektrum, 21/1998, S. 89 f.

7 In Anlehnung an die Life-Cycle-Phasen zur Softwareentwicklung. Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; a. a. O., S. 89

von der Object Management Group (OMG) verabschiedeter Standard,[9] der von bedeutenden Unternehmen wie IBM, Microsoft und Oracle durch eine Vielzahl von Entwicklungs- und Modellierungswerkzeugen getragen wird.[10]

Mit der UML wird die Entwicklung von komplexen Anwendungssystemen unterstützt. Sie verspricht, innerhalb des Life-Cycle-Phasenmodells den Übergang von der Entwurfs- zur Implementierungsphase zu erleichtern,[11] da die UML für die direkte Übersetzung in eine objektorientierte Programmiersprache, d. h. für die Code-Generierung mit Entwicklungswerkzeugen, entworfen wurde.[12] Sie stellt eine Vielzahl von phasenspezifischen Sprachelementen und Diagrammtypen bereit, die zur Beschreibung der statischen Struktur und des dynamischen Verhaltens von Objekten eines sich in der Analyse- oder Entwurfsphase befindlichen Anwendungssystems ebenso herangezogen werden können, wie zur Beschreibung von detaillierten Implementierungsanweisungen.[13]

Im Hinblick auf die Entwicklung von Anwendungssystemen besitzt die UML gegenüber anderen objektorientierten Modellierungssprachen Vorteile, die sich u. a. mit der Verfügbarkeit von Diagrammtypen[14] wie dem „Klassendiagramm (class diagram)“, dem „Anwendungsfalldiagramm (use case diagram)“ und verschiedenen Verhaltensdiagrammen wie dem „Zustandsdiagramm (state chart diagram)“ oder dem „Aktivitätsdiagramm (activity diagram)“ begründen.

8 Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; a. a. O., S. 89.

9 Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; a. a. O., S. 89.

10 Vgl. Object Management Group (Hrsg.): OMG Unified Modeling Language Specification (draft), a. a. O., S. 1-13 f.

11 Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; a. a. O., S. 89.

12 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; 4., akt. Auflage, München et al.: Oldenbourg 1998, S. 21.

13 Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; a. a. O., S. 89.

14 Die UML verfügt über insgesamt acht Diagrammtypen. Vgl. hierzu Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 204.

2.2 Klassendiagramm

Das zentrale Sprachelement der objektorientierten Modellierung ist die Klasse in unterschiedlichen Abstraktionsgraden (Klasse, abstrakte Klasse, Metaklasse)[15]. Ergänzend kommen verschiedene Basiselemente wie Objekt, Attribut, Operation, Schnittstelle, Paket oder Zusicherung hinzu. Auf der Grundlage des UML-Sprachelementes der Klasse sowie statischen Beziehungselementen sind Klassendiagramme als Technik zu verstehen, die miteinander in Beziehung stehende Klassen illustrieren. Zusätzlich zu den zueinander in Beziehungen stehenden Klassen können Bedingungen („Zusicherungen" bzw. „Constraints") weitere semantische Informationen z. B. als Attributwertebereich in dem Klassenmodell repräsentieren.[16]

Fowler unterscheidet in diesem Zusammenhang drei Sichtweisen[17] zur Erstellung von Klassendiagrammen, die in unterschiedlichen Phasen des Life-Cycle-Phasenmodells eines zu entwickelnden Anwendungssystems zur Anwendung kommen und letztendlich den zunehmenden Detaillierungsgrad im Entwicklungsprozeß des entstehenden Anwendungssystems repräsentieren.[18]

Eine alternative Unterscheidung bezüglich des Detaillierungsgrades nimmt Oestereich vor. Er unterscheidet zwischen zwei Klassenvarianten: die Geschäftsklasse und die Fachklasse. Eine Geschäftsklasse abstrahiert ein reales Geschäftsobjekt in einem Detaillierungsgrad, „wie er vor allem von Fachabteilungen und Managern verstanden"[19] wird (Makroebene). Auf die Geschäftsklasse wird in der Fachkonzeptphase zur unscharfen Darstellung (ohne Attribute und Operationen) der statischen Klassenstruktur eines in ein Anwendungssystem umzusetzenden Prozesses zurückgegriffen.[20]

Das Modell der Fachklasse wird von einer konkreten Geschäftsklasse abgeleitet und repräsentiert die fachlichen Anforderungen eines Prozesses in einem Detaillierungsgrad,

15 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 223 ff.

16 Vgl. Object Management Group (Hrsg.): OMG Unified Modeling Language Specification (draft), a. a. O., S. 3-23 f.

17 Im einzelnen sind das die „konzeptionelle", „spezifizierende" und die „implementierende Sichtweise". Vgl. Fowler, Martin; Scott, Kendall: UML konzentriert: Die neue Standard-Objektmodellierungssprache anwenden, a. a. O., S. 62 f.

18 Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; a. a. O., S. 89.

19 Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 156.

20 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 91.

dessen Ergebnis (Designmodell/DV-Konzept) die Entwickler bei der Implementierung der Systemlösung unterstützt (Mikroebene).[21] Die Differenzierung der Klasse in eine Geschäfts- und eine Fachklasse führt iterativ zu einer Verfeinerung der Attributs- sowie Beziehungsdarstellung (und im Fall der Operation) bis auf die Ebene der zu implementierenden Methode mit Parameterangaben.[22] Bei einer schrittweisen Verfeinerung können Geschäftsklassen in Fachklassen überführt werden, die so als Implementierungsanleitung für Entwickler dienen. Die Anwendung von Klassendiagrammen führt zu statischen Strukturen von Objekten/Aufgaben unterschiedlichen Detaillierungsgrades, die geeignet sind, softwaretechnisch in einem Anwendungssystem abgebildet zu werden.[23]

Die UML-Darstellungsnotation für Klassendiagramme umfaßt Rechtecke zur Repräsentation einer Klasse (Fach- und Geschäftsklasse), eine Linie mit Kardinalitätsangaben für die Darstellung einer Assoziation zwischen Klassen,[24] eine Vererbungsbeziehung mit einem leeren Pfeil in Richtung der vererbenden Klasse,[25] eine Aggregationsbeziehung als Linie zwischen zwei Klassen mit leerer Raute auf der Seite des Aggregats[26] und eine Kompositionsbeziehung als Linie zwischen zwei Klassen mit gefüllter Raute auf der Seite des Aggregats (siehe Abbildung 1).[27]

Im Hinblick auf die Entwicklung von Anwendungssystemen leisten die Geschäftsklassen einen Beitrag zum Dialog zwischen zukünftigen Benutzern und den Fachverantwortlichen zur Spezifikation der fachlichen Anforderung zur Umsetzung eines dokumentenbegleiteten Geschäftsprozesses in ein DMS.[28]

---

21 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 91 f.

22 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 92.

23 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 168 ff.

24 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 269.

25 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 262.

26 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 284 f.

27 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 286 f.

28 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 156.

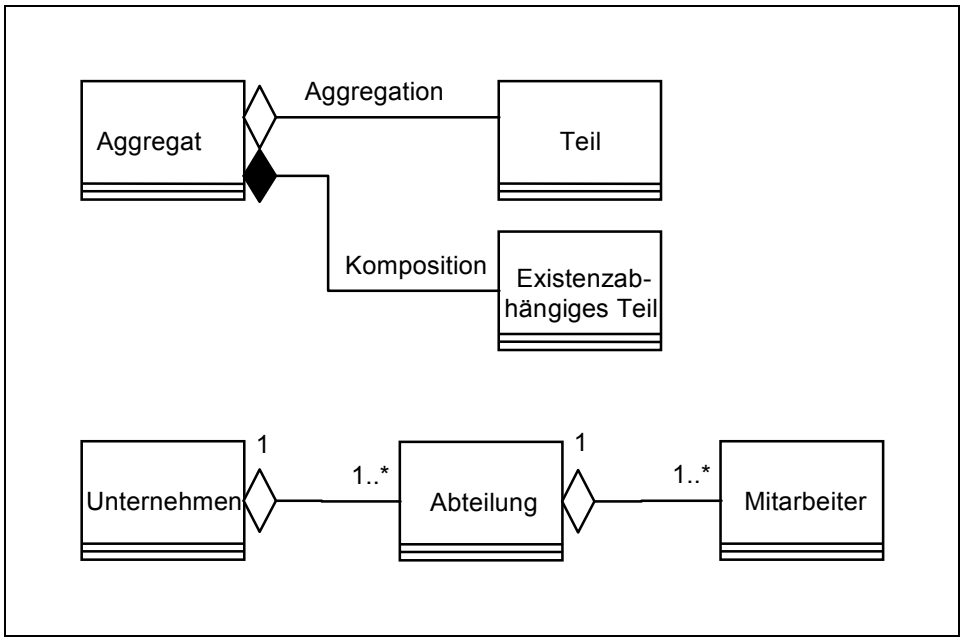

Abb. 1: Die UML-Beziehungselemente Aggregation und Komposition

2.3 Zustandsdiagramm

Das Zustandsdiagramm ist darauf ausgerichtet, das interne Verhalten eines Objektes anwendungsfallübergreifend im Zeitverlauf zu beschreiben. Ein derartiges Zustandsmodell formalisiert den Lebenszyklus eines Objektes als Sequenz von Zuständen vorgeschriebener Reihenfolge, die sich aufgrund definierter, äußerer Stimuli (Ereignisse) verändern können und die mit bestimmten Aktivitäten (Aktionen) verbunden sind.[29]

Konstituierende Elemente des Zustandsdiagrammes sind der Zustand, das Ereignis, die Aktion und der Zustandsübergang (Transition). Ein Zustand löst Aktionen aus, wartet auf eintretende Ereignisse und erfüllt Bedingungen. Als Zustand wird die Abstraktion von einer Menge bestimmter Attributausprägungen verstanden, die ein Objekt einer Klasse annehmen kann.[30] Da nicht jede Änderung eines Attributwertes (jede Attributwertänderung löst ein Ereignis aus) automatisch zu einer Zustandsänderung führt, werden im Zustandsdiagramm nur diejenigen Ereignisse berücksichtigt, die das Objektver-

29 Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; a. a. O., S. 90 und Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 310.

30 Vgl. Scheer, August-Wilhelm: ARIS – Modellierungsmethoden, Metamodelle, Anwendungen, 3., völlig neubearb. und erw. Aufl., Berlin et al.: Springer 1998, S. 128.

halten modifizieren. Ein Zustand kann auch als das Intervall zwischen dem Eintreten eines Ereignisses, das zu einem Zustand hinführt, und dem Eintreten eines zweiten (späteren) Ereignisses, das den nächsten Zustand bestimmt, veranschaulicht werden.[31]

Abbildung 2 ist ein Vorschlag der Notationsanwendung des UML-Konzeptes zum Zustandsdiagramm für das Objekt „Kundenanfrage“. Ein Objekt braucht einen Stimulus, damit es einen eingenommen Zustand verläßt. Dieser wird in objektorientierten Modellen durch ein Ereignis ausgelöst und führt zu einem Zustandsübergang, der als Transition bezeichnet wird.[32] Ein Ereignis wird ausgelöst, wenn eine für die Transition definierte Bedingung „wahr“[33] (siehe Abbildung 2 „Condition: Letzter Kontakt > 12 Monate“) wird, wenn das Objekt eine Nachricht von einem anderen Objekt erhält (eine Operation wird ausgeführt) oder wenn eine bestimmte Zeitspanne überschritten bzw. ein bestimmter Zeitpunkt eingetreten ist.[34] Das Eintreten eines neuen Zustands kann erneut eine Operation anstoßen und damit wiederum ereignisauslösend sein, so daß ein Zustandsdiagramm eine dynamische Abfolge von Zustand, Operation und Ereignis realisiert.

Die Notation eines Zustandes bildet ein abgerundetes Rechteck, wobei im oberen Teil der Zustandsname und unterhalb optional eine Aktionsbeschreibung abgebildet ist. Das Erreichen eines neuen Zustandes kann eine Aktion auslösen beim Eintritt des Zustandes („entry/“), solange ein Zustand vorherrscht („do/“) oder beim Verlassen des Zustandes („exit/“).[35] Aktionen korrespondieren dabei mit Aktivitäten/Funktionen anderer UML-Diagrammtypen.[36] Eine Transition wird durch einen Pfeil symbolisiert, der von einem Zustand (Quellzustand) zum nächsten Zustand (Zielzustand) zeigt. Die Transition wird mit dem Transitions-auslösenden Ereignis und/oder einer Bedingung beschriftet.[37]

---

31 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 312 f.

32 Vgl. Burkhard, Rainer: UML – Unified Modeling Language: Objektorientierte Modelle für die Praxis, 2. Aufl., Bonn: Addison-Wesley 1998, S. 250.

33 Das Bedingungsergebnis wird hier als Boolscher Operator verstanden.

34 Vgl. Object Management Group (Hrsg.): OMG Unified Modeling Language Specification (draft), a. a. O., S. 3-28 f.

35 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 313.

36 Vgl. hierzu die Erläuterungen in Kap. 0

37 Vgl. Object Management Group (Hrsg.): OMG Unified Modeling Language Specification (draft), a. a. O., S. 3-131.

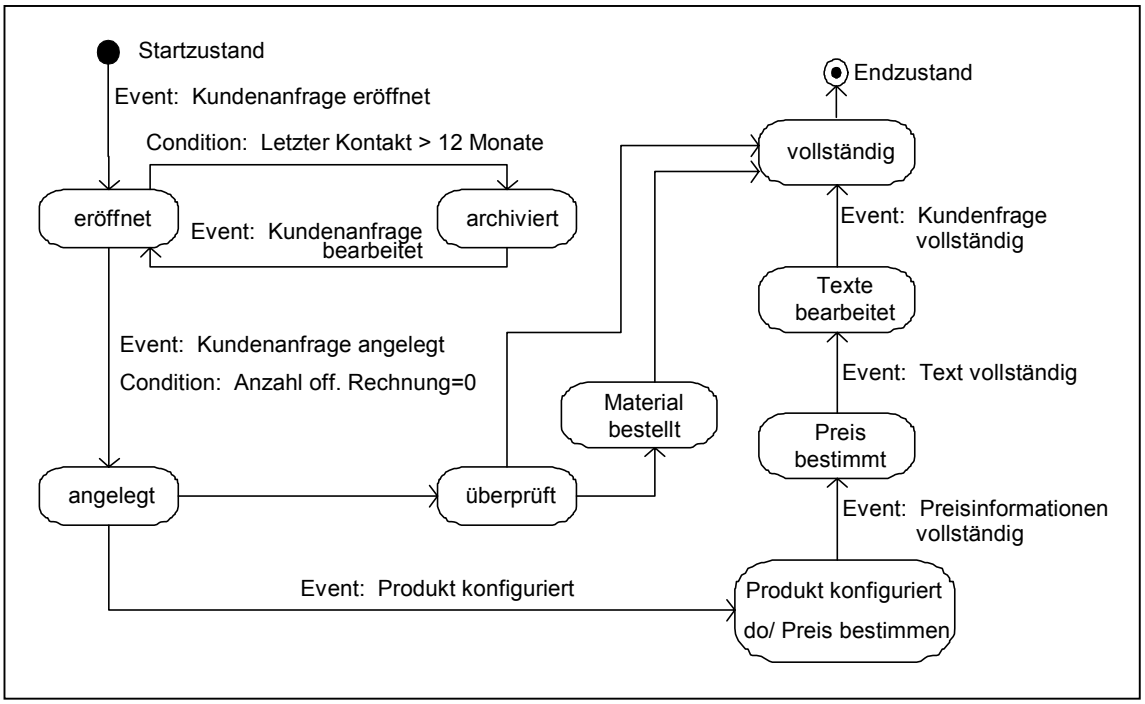

Abb. 2: Zustandsdiagramm für das Objekt "Kundenanfrage"

2.4 Anwendungsfalldiagramm

Ein weiterer Vorzug der UML liegt in der Verwendung von UML-Diagrammen zur Modellierung von Geschäftsprozessen. Das Anwendungsfalldiagramm eignet sich zur Abbildung von Geschäftsprozeßszenarien[38], die durch ein Anwendungssystem unterstützt werden sollen, betrachtet dies aber ausschließlich aus der Sicht des zukünftigen Benutzers.[39] Das Diagramm vereinfacht die Kommunikation zwischen dem zukünftigen personellen Aufgabenträger („Akteur") und den Fachverantwortlichen bei der Anforderungsermittlung.[40]

38 In dem ARIS-Konzept wird das Anwendungsfalldiagramms als eine neben anderen Möglichkeiten zur modellhaften Darstellung der Beziehung zwischen Funktion und Organisationseinheit auf der Ebene der Fachkonzepterstellung eingesetzt, wobei nur zukünftige Benutzer des Anwendungssystems als Organisationseinheiten zulässig sind. Vgl. Scheer, August-Wilhelm: ARIS – Vom Geschäftsprozeß zum Anwendungssystem, a. a. O., S. 40 ff.

39 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 215.

40 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 208.

Zentrales Element dieses Diagrammtyps ist der Anwendungsfall, der als Technik zur Abbildung eines Teilprozesses in einem zukünftigen Anwendungssystem geeignet ist und damit in der Systemanforderungsphase spezifiziert, „was“ das zu entwickelnde System aus Sicht des Anwenders leisten muß. Unterschiedliche Anwendungsfallausprägungen konkretisieren alternative Prozeßszenarien, so daß ein Anwendungsfall eine Klasse von Prozessen (Objekten) darstellt.[41] Die Zuordnung eines zukünftigen Benutzers („Kundenbetreuer“ in Abbildung 3) zu einem Anwendungsfall („Kundenanfrage ermitteln“ in Abbildung 3) komplettiert das Anwendungsfalldiagramm. Abschließend ist festzuhalten, daß Anwendungsfalldiagramme eine vergleichsweise einfache Darstellung ermöglichen, mit der sich Prozesse aber nur rudimentär modellieren lassen, da sich z. B. Fallunterscheidungen oder Wiederholungen nicht abbilden lassen.

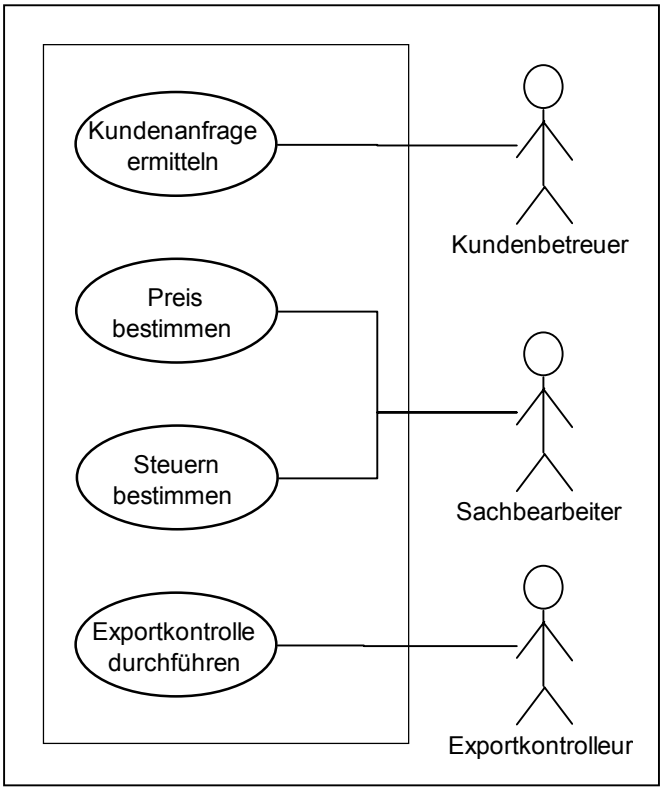

Abb. 3: Anwendungsfalldiagramm „Kundenanfragebearbeitung“

41 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 207.

2.5 Aktivitätsdiagramm

Das Aktivitätsdiagramm ist eine spezielle Ausprägung des Zustandsdiagrammes und verfügt über die Fähigkeit, einen Geschäftsprozeß oder Vorgang (Workflow)[42] auf Basis eines „Anwendungsfalls“[43] als eine Aneinanderreihung von Aktivitäten/Funktionen darzustellen.[44] In diesem Zusammenhang veranschaulicht ein Aktivitätsdiagramm die dynamischen Details eines Anwendungsfalles (auch anwendungsfallübergreifend) und zielt darauf ab, die internen Abläufe eines Prozesses in Abhängigkeit von Geschäftsregeln und Bedingungen darzustellen.[45] Insofern stellen Aktivitätsdiagramme eine Detaillierung von Anwendungsfällen dar.

Im Unterschied zu Flußdiagrammen (als Vertreter der dynamischen Funktionsmodelle), die auf sequentielle Funktionsdarstellungen beschränkt sind, kann in Aktivitätsdiagrammen verdeutlicht werden, ob Aktivitäten parallel oder sequentiell durchgeführt werden müssen.[46] Parallele Aktivitäten bieten die Chance, die Effizienz und die Durchlaufzeit von Prozessen zu verbessern. Sie müssen aber an definierten Diagrammstellen durch sogenannte „Synchronisationsbalken“ untereinander synchronisiert werden; d. h., zwei parallele Aktivitäten müssen vollständig realisiert sein, bevor die nachfolgende Aktivität ausgelöst werden kann (siehe Abbildung 4 „Synchronisation“). Zusätzlich erlauben Aktivitätsdiagramme zu den einzelnen Aktivitäten/Funktionen die Objektzustände festzuhalten; d. h., der semantische Informationsgehalt wird ausgebaut.[47]

Abbildung 4 zeigt einen Ausschnitt aus dem Geschäftsprozeß „Kundenanfragebearbeitung“ als Aktivitätsdiagramm. Die Synchronisationslinie unterhalb der Aktivität „Produkt konfigurieren“ teilt den Aktivitätsfluß auf. Die Linie synchronisiert ausgehende Transitionen, so daß die wegführende Transition erst ausgeführt wird, wenn alle eingehenden eingetreten sind.

---

42 Vgl. Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; a. a. O., S. 89.

43 Das UML-Element des „Anwendungsfalls“ ist ein Aufgabenbereich/Prozeßteil und beschreibt, „was“ ein Anwendungssystem aus Sicht des Anwenders („Akteurs“) zu leisten hat. Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 207.

44 Vgl. Fowler, Martin; Scott, Kendall: UML konzentriert: Die neue Standard-Objektmodellierungssprache anwenden, a. a. O., S. 131.

45 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; a. a. O., S. 296.

46 Vgl. Fowler, Martin; Scott, Kendall: UML konzentriert: Die neue Standard-Objektmodellierungssprache anwenden, a. a. O., S. 133.

47 Vgl. Fowler, Martin; Scott, Kendall: UML konzentriert: Die neue Standard-Objektmodellierungssprache anwenden, a. a. O., S. 133 ff.

Zwischen einem Aktivitätsdiagramm und einer ereignisgesteuerten Prozeßkette (EPK) ist eine Affinität zu registrieren. In einem Aktivitätsdiagramm wird sowohl der Kontrollfluß als auch der Objektfluß ähnlich der Darstellung des Kontrollflusses und des Datenflusses in der EPK entlang einer Aktivität/Funktion abgebildet. Beide Formen (Aktivitätsdiagramm und EPK) eignen sich zur Darstellung von Geschäftsprozessen, wobei Aktivitäten Funktionen entsprechen und EPK-Ereignisse sich in den Transitionspfeilen zwischen zwei Objektzuständen verbergen, da Ereignisse in der Notation des Aktivitätsdiagramms nicht explizit berücksichtigt werden.

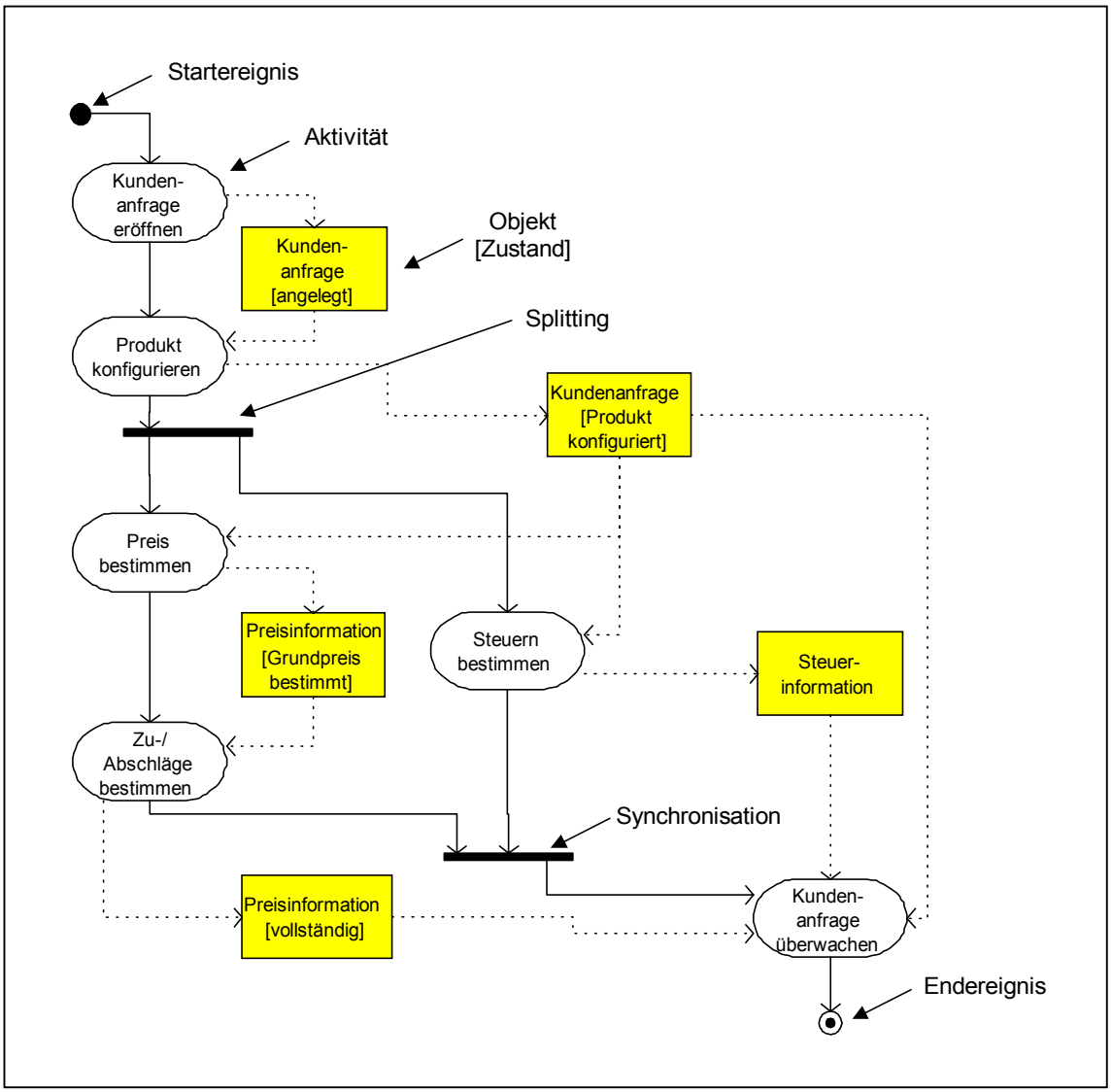

Abb. 4: Aktivitätsdiagramm „Kundenanfragebearbeitung“

3 Objektorientierte Prozeßmodellierung

3.1 Einführung

Ziel der objektorientierten Prozeßmodellierung ist die integrierte Darstellung der unternehmensrelevanten Geschäftsprozesse und Geschäftsobjekte (Business Objects), die Gegenstand einer Geschäftsprozeßbearbeitung sind, in einem Modell.[48] In einem Geschäftsobjekt wie z. B. Kunde, Auftrag oder Bestellung ist gekapseltes Objektwissen (Informationen als Attribute und Verhalten als Operationen) enthalten. Geschäftsobjekte können in ihrer inneren Struktur eine Komposition von unterschiedlichen Klassen sein, beschreiben einen betriebswirtschaftlichen Zusammenhang und repräsentieren die zur unternehmerischen Leistungserstellung relevanten Entitäten.[49]

Mit dem Ansatz von Bungert/Heß[50] kann eine EPK in ein äquivalentes Objektmodell überführt werden, wobei vorausgesetzt wird, daß die in der EPK verwendeten Daten (Datensicht) als Klassen (z. B. Kunde, Kundenauftrag) vorliegen. Weitere Schritte der Transformation sind die Zuordnung der prozeßrelevanten Funktionen zu den definierten Klassen, die Definition von auslösenden Ereignissen für jede Funktion und die Bestimmung der ereignisauslösenden Vorgängerfunktion.

Nüttgens/Zimmermann[51] wählen den Ansatz der objektorientierten EPK (oEPK), eine kombinierte Darstellungsform von ERM, statischen Klassen und EPK; dies mit dem Ziel, eine gleichzeitige Darstellung von EPK und Interaktion zwischen Geschäftsobjekten zu erreichen. Funktionen werden durch Klassen ersetzt, in denen die betriebswirtschaftlichen Funktionen gekapselt sind, und über Ereignisse miteinander verbunden werden.[52] Ein Nachteil dieser Darstellung ist, daß zur Bearbeitung einer betriebswirtschaftlichen Funktion oftmals mehrere Klassen herangezogen werden müssen und dies nur unzulänglich darstellbar ist.

---

48 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 3.

49 Vgl. Seubert, Michael: Business-Objekte und objektorientiertes Prozeßdesign, in: Referenzmodellerriung: State-of-the-Art und Entwicklungsperspektiven, Hrsg.: Becker, Jörg; Rosemann, Michael; Schütte, Reinhard, Heidelberg: Physica-Verlag 1999, S. 112 ff.

50 Vgl. Bungert, Winfried; Heß, Helge: Objektorientierte Geschäftsprozeßmodellierung, in: Information Management, 1/1995, S. 59 ff.

51 Vgl. Scheer, August-Wilhelm: ARIS – Modellierungsmethoden, Metamodelle, Anwendungen, a. a. O., S. 136.

52 Vgl. Scheer, August-Wilhelm; Nüttgens, Markus, Zimmermann, Volker: Objektorientierte Ereignisgesteuerte Prozeßkette (oEPK) - Methode und Anwendung, in: Arbeitsberichte des Instituts für Wirtschaftsinformatik, Heft 141/1997, Hrsg.: Scheer, August-Wilhelm, Saarbrücken 1997, S. 18.

Ein dritter Ansatz von Loos/Allweyer[53] zielt darauf ab, die prozeßorientierte EPK-Darstellung mit UML-Diagrammtypen zu verbinden, wobei UML-Notationselemente mit EPK-Elementen in Beziehung stehen oder durch diese ersetzt werden.[54] In diesem Zusammenhang führt eine weiterhin strikte Trennung von Objektwissen und Prozeßwissen zu einer Komplexitätsreduktion bei der Analyse und dem Entwurf von Anwendungsszenarien.[55] Im folgenden werden drei Konzepte vorgestellt, die eine EPK mit UML-Sprachelementen verbinden:

3.2 EPK und Klassendiagramm

In diesem Kontext zielt die Kombination einer EPK mit Elementen eines Klassendiagramms darauf ab, zu beschreiben, welche EPK-Funktionen auf welche Klassen zugreifen bzw. welche Objekte von einer Funktion erzeugt oder modifiziert werden.[56] Beim objektorientierten Prozeßdesign setzt sich die Funktionalität eines zu entwickelnden Anwendungssystems aus mehreren (Geschäfts-) Objekten zusammen, die von einem Geschäftsprozeß gesteuert und über Operationen angesprochen werden. Ein Geschäftsprozeß setzt sich dann aus sequentiell aufgerufenen Operationen einzelner Objekte zusammen.[57]

Abbildung 5 zeigt im linken Bildteil einen Ausschnitt des Kundenanfragebearbeitungsprozesses in EPK-Notation. Die Funktion „Anfrage eröffnen“ greift auf Informationen (Objekte) der Klassen „Material“ und „Sachbearbeiter“ zu und erzeugt gleichzeitig ein neues Objekt der Klasse „Kundenanfrage“. Der rechte Bildteil stellt auf einer detaillierteren Ebene die Verbindung von Funktionen mit einzelnen Operationen und Attributen dar. Die Funktion „Anfrage eröffnen“ ruft die Operation „neu anlegen“ der Klasse „Kundenanfrage“ auf. Abbildung 6 (Klassendiagramm „Kundenanfrage“) präzisiert die Zugehörigkeit der in der EPK verwendeten Operationen und Attribute zu Klassen.

---

53 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 1 ff.

54 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 1.

55 Vgl. Seubert, Michael: Business-Objekte und objektorientiertes Prozeßdesign, a. a. O., S. 122.

56 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 10.

57 Vgl. Seubert, Michael: Business-Objekte und objektorientiertes Prozeßdesign, a. a. O., S. 121.

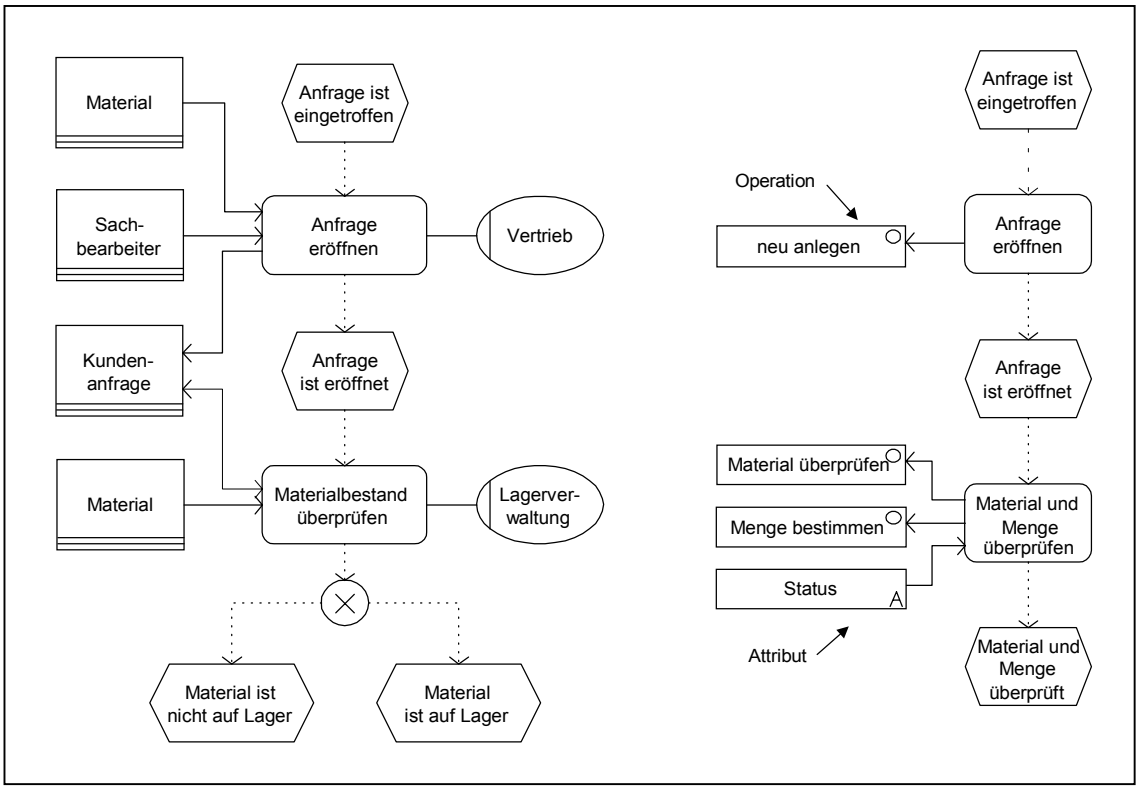

Abb. 5: EPK „Kundenanfragebearbeitung“ mit Klassen-, Operationen- und Attributzuordnung[58]

Die unterschiedlichen Detaillierungsgrade in Abbildung 5 zielen auf die verschiedenen Phasen eines Softwareentwicklungsprozesses ab. So kann in der Analysephase eines zu entwickelnden Anwendungssystems die Verbindung zwischen Klassen und Funktionen auf Makroebene dargestellt werden (linker Bildteil), wohingegen in der Entwurfsphase implementierungsnahe Methoden und Attribute für einzelne Funktionen spezifiziert werden (Mikroebene, rechter Bildteil).

58 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 9 f.

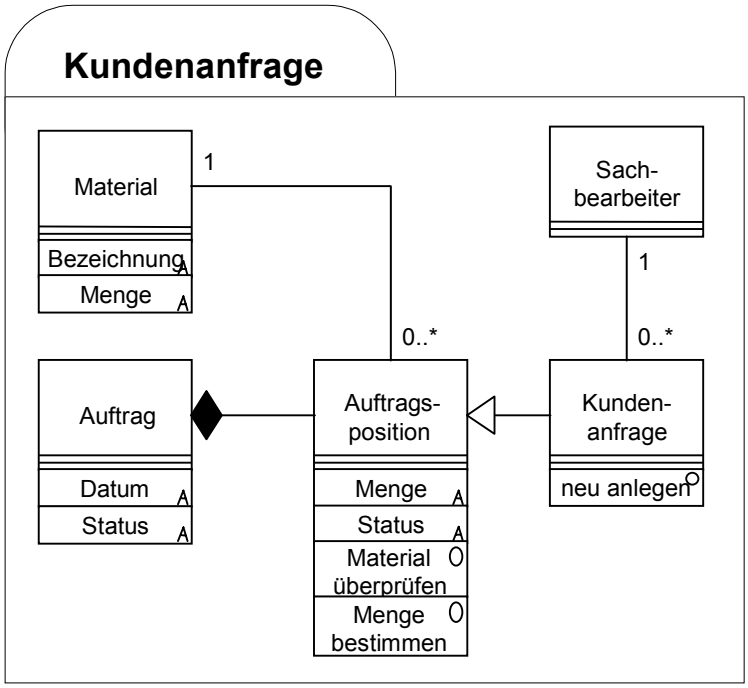

Abb. 6: Klassendiagramm „Kundenanfrage“[59]

3.3 EPK und Zustandsdiagramm

In Orientierung an der Kombination von EPK-Elementen mit Klassenelementen eröffnet darüber hinaus die Verbindung einer EPK mit Elementen eines Zustandsdiagrammes die Möglichkeit, zu spezifizieren, in welchem Zustand sich ein Objekt vor und nach einer Funktionsausübung befindet. Es werden Input- und Output-Verbindungen zwischen Funktionen und Zuständen hergestellt, wobei die in einer EPK verwendeten Zustände konsistent mit den zuvor modellierten Transitionen eines objektbezogenen Zustandsdiagrammes sein müssen.[60] Dieser Verknüpfungsschritt eröffnet die Möglichkeit, den Lebenslauf von Objekten in einem Prozeß zu verfolgen bei gleichzeitiger Trennung von Objekt- und Prozeßwissen. Zum Beispiel ist eine Materialbestellung nach Eintreffen einer Kundenanfrage erst dann erforderlich, wenn diese mit dem aktuellen Materialbestand abgeglichen wurde. Abbildung 7 illustriert in der linken Bildhälfte, daß eine Mate-

59 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 11.

60 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 12.

rialbestellung (Funktion „Material bestellen“) erst dann erfolgen kann, wenn sich das Objekt „Kundenanfrage“ im Zustand „überprüft“ befindet und das Ereignis „Material nicht auf Lager“ eintritt. Der rechte Bildteil zeigt das zugehörige Zustandsdiagramm für das Objekt „Kundenanfrage“, das wiederum einen Diagrammausschnitt der Abbildung 1 wiedergibt.

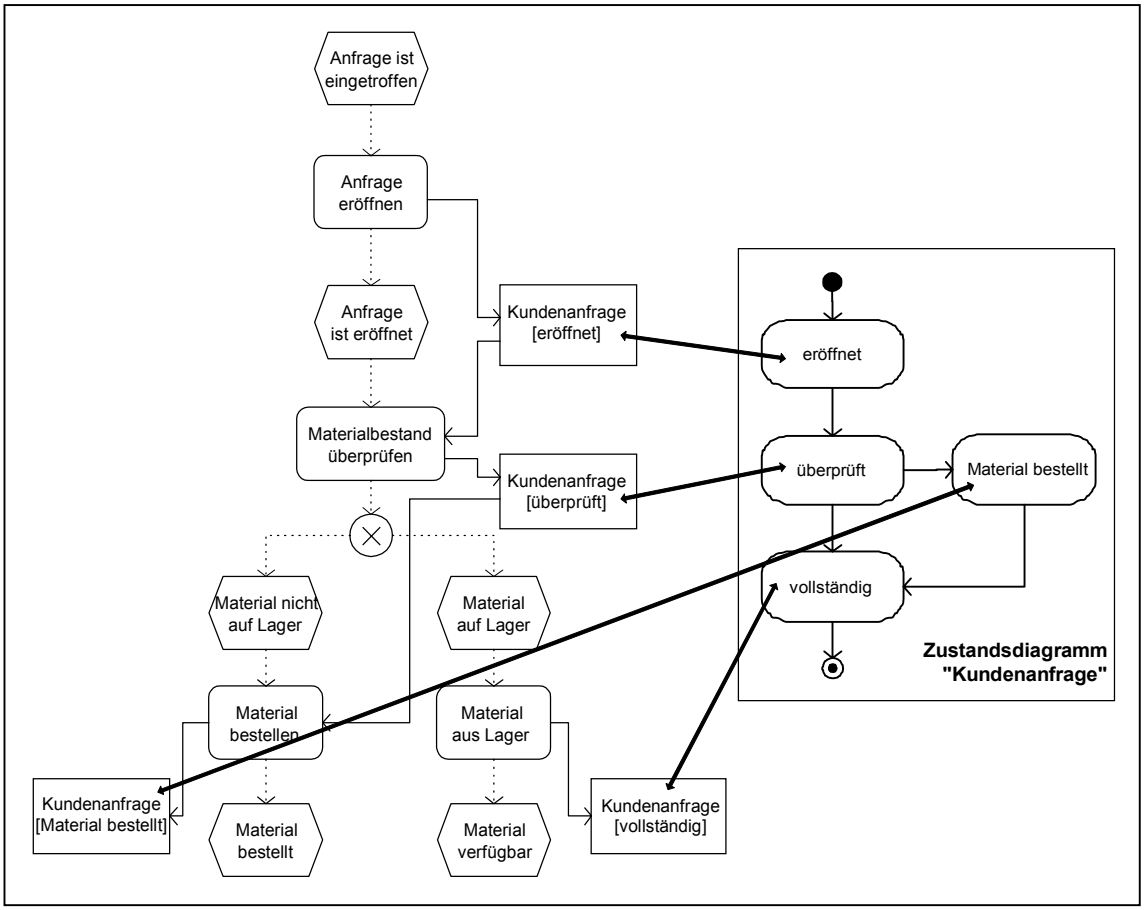

Abb. 7: Verwendung von UML-Zuständen in einer EPK[61]

3.4 EPK und Anwendungsfalldiagramm

Eine dritte Möglichkeit der Kombination von objekt- mit prozeßorientierten Techniken stellt die Detaillierung von EPK-Funktionen mit Anwendungsfalldiagrammen bzw. im umgekehrten Fall die Verfeinerung eines Anwendungsfalles unter Mitwirkung einer

61 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 12.

EPK dar. Da ein Anwendungsfall definitionsgemäß nicht eine betriebswirtschaftlich relevante Aktivität abbildet, sondern vielmehr die Interaktion eines Benutzers mit einem Anwendungssystem im Vordergrund steht,[62] hilft ein Anwendungsfalldiagramm die zur Funktionsausführung (EPK-Funktion) notwendigen anwendungsspezifischen Benutzerschritte zu detaillieren. Im umgekehrten Fall können die einem umfangreichen Anwendungsfall wie z. B. die „Kundenanfragebearbeitung“ zugrundeliegenden Einzelaktivitäten mit einer EPK gegliedert werden.[63]

4 Praxisbeispiel

4.1 Einführende Erläuterungen

Im Rahmen der Einführung eines Dokumenten-Management-Systems (DMS) in der Patentabteilung eines weltweit tätigen Life-Science-Unternehmens, wird deren Prozeß der „Patentrecherche“ untersucht, der mit verschiedensten Dokumenten verzahnt ist und daher mit einem DMS informationstechnisch unterstützt werden soll.

Der Prozeß der Patentrecherche (siehe Abbildung 8), der von Mitarbeitern der Abteilung Patente bearbeitet wird, beschäftigt sich mit der Klärung von Patentsituationen. Der Prozeß wird durch eine Rechercheanfrage eines Mitarbeiters aus der Forschung oder Entwicklung ausgelöst und führt im Prozeßverlauf ggfs. zu einem Patentrechercheauftrag. Der Informationsfluß dieses Prozesses ist z. Z. überwiegend papiergebunden. Die Patentabteilung archiviert alle prozeßrelevanten Dokumente in Aktenordnern, was zu einer Aktenzunahme von mehreren tausend Seiten pro Monat führt. Mit der Einführung eines DMS wird zum einen das Ziel verfolgt, die Aktenablagefläche zu reduzieren. Darüberhinaus können Dokumente (z. B. Patentschriften und -gutachten) bei späteren Recherchen wiederverwendet werden, was zu einer Kostenreduzierung für die Beschaffung, einer beschleunigten Auftragsbearbeitung und zu einer Vermeidung einer redundanten Dokumentenarchivierung führt.

62 Vgl. Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language, a. a. O., S. 207.

63 Vgl. Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), a. a. O., S. 13.

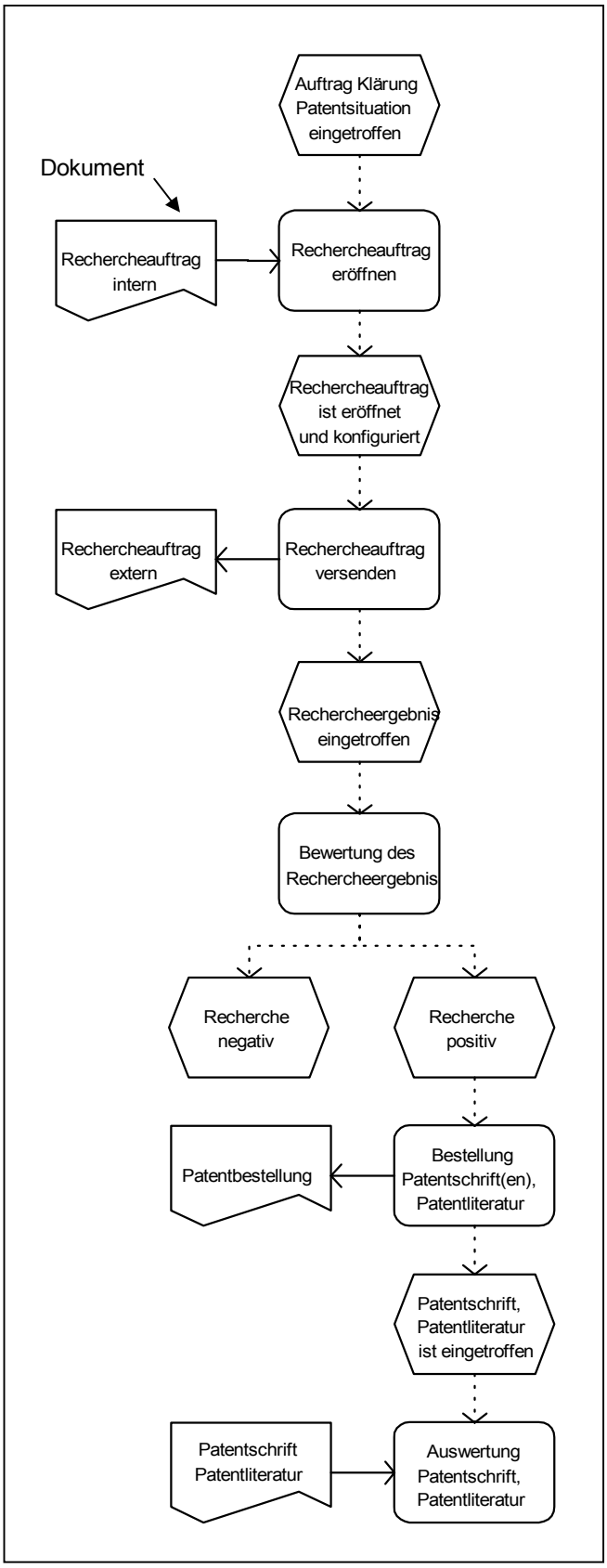

Abb. 8: Prozeß „Patentrecherche“ ohne DMS-Unterstützung

4.2 Objektorientierte Prozeßmodellierung am Beispiel des Patentrechercheprozesses

Das primäre Ziel von modellhaften Prozeßdarstellungen ist, Prozeßabläufe derart zu visualisieren, daß die unternehmensrelevanten Prozesse für alle beteiligten Organisationseinheiten und Mitarbeiter transparent sind und damit die Kommunikation zwischen Fachverantwortlichen und Informationsmanagement vereinfacht wird. Hierzu sind in mehreren Teilschritten die folgende Aufgaben bewältigen:

Das Ergebnis der Prozeßaufnahme und -dokumentation ist ein Ist-Prozeßmodell, bei dem sich ein Optimierungspotential durch Unterstützung mit einer DMS-Software ergibt, da dieser Prozeß von unterschiedlichen (Papier-) Dokumenten (z. B. „Rechercheauftrag intern“, „Rechercheauftrag extern“, „Patentbestellung“, „Patentschrift/Patentliteratur“ in Abbildung 8) begleitet wird. Die sich daran anschließende Prozeßanalyse, -bewertung und -gestaltung führt iterativ zu einer Prozeßverbesserung bei gleichzeitiger Ausrichtung auf die Unternehmensziele. Resultat ist ein Soll-Prozeßmodell, das anschließend in ein Anwendungssystem umgesetzt wird.[64]

In der Phase der Prozeßgestaltung erfolgt die Modellierung des Prozesses, der durch ein DMS unterstützt werden soll (bei der Verwendung des ARIS-Konzeptes, die Modellierung der einzelnen ARIS-Sichten und -Ebenen). Ausgehend von dem Prozeß der Patentrecherche werden die Geschäftsobjekte, die Gegenstand einer Geschäftsprozeßbearbeitung sind, in Klassendiagrammen modelliert. In einem weiteren Schritt erfolgt die Integration von UML-Elementen mit EPK-Elementen der Patentrecherche; d. h., es wird beispielsweise festgelegt, welche EPK-Funktion auf welche Klassen zugreift bzw. welche Objekte von einer Funktion erzeugt oder modifiziert werden.

Abbildung 9 zeigt in der linken Bildhälfte den dokumentenbehafteten aber noch nicht DMS-unterstützen Prozeß („Geschäftsprozeß“ in Abbildung 9), der in der Prozeßgestaltungsphase mit Hilfe von UML-Elementen konfiguriert wird. Das Ergebnis ist ein mit DMS-unterstützter Soll-Prozeß (in der unteren Bildhälfte), der in Abhängigkeit vom Detaillierungsgrad die Customizing- bzw. Implementierungsanleitung für Entwickler enthält.

64 Vgl. Scheer, August-Wilhelm: ARIS – Vom Geschäftsprozeß zum Anwendungssystem, a. a. O., S. 150 ff. und Hirschmann, Petra; Scheer, August-Wilhelm: Konzeption einer DV-Unterstützung für das überbetriebliche Prozeßmanagement, in: Arbeitsberichte des Instituts für Wirtschaftsinformatik, Heft 113/1994, Hrsg.: Scheer, August-Wilhelm, Saarbrücken 1992, S. 3 f.

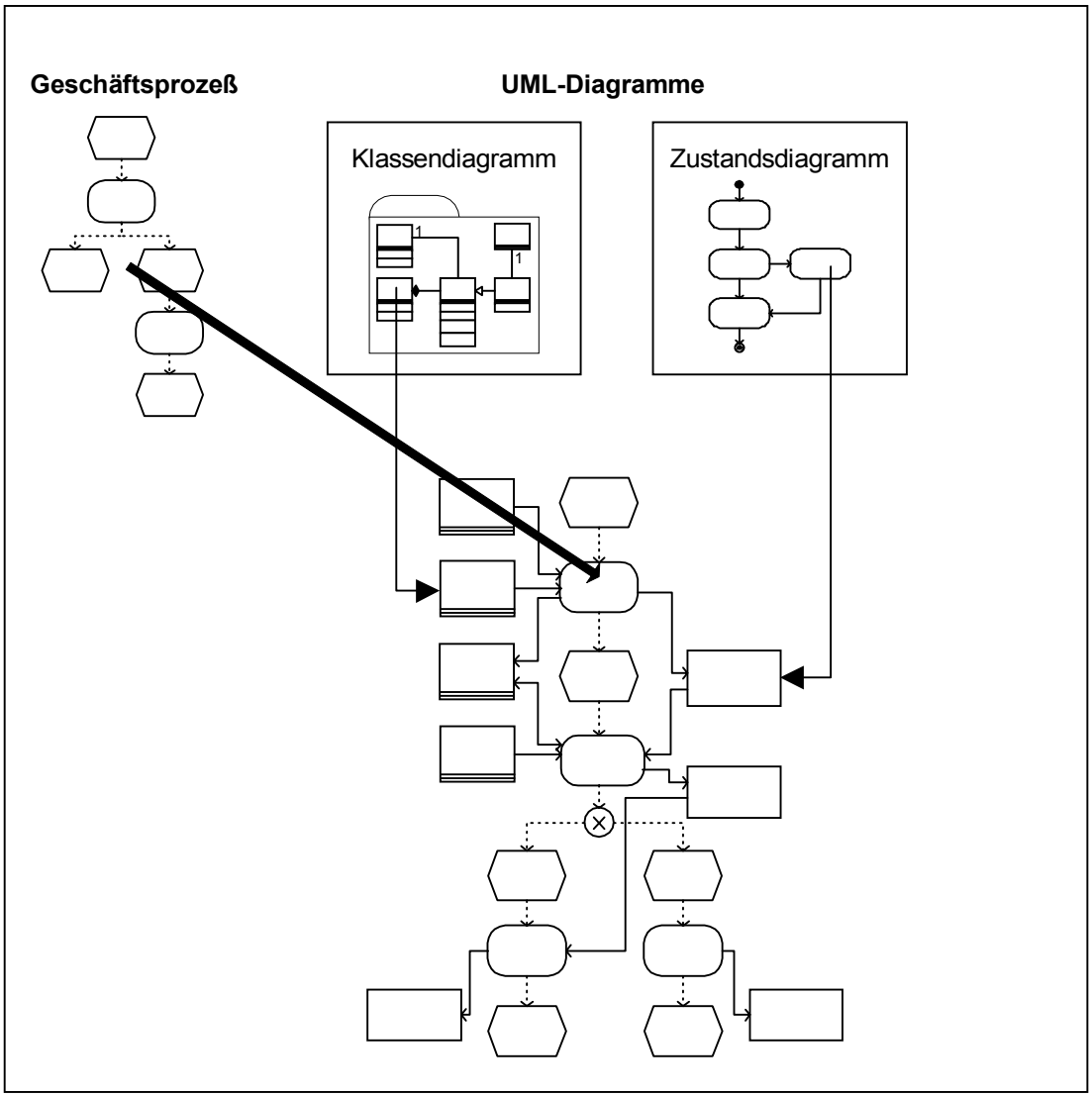

Abb. 9: Integration von UML-Elementen in eine EPK

Abbildung 10 zeigt die Integration von UML-Elementen in eine EPK am Beispiel des Patentrechercheprozesses, der durch ein DMS unterstützt werden soll. Der Prozeß wird durch eine unternehmensinterne Rechercheanfrage eines Mitarbeiters aus der Forschung oder Entwicklung ausgelöst. Diese Anfragen erreichen die Patentabteilung auf „klassischen“ Kommunikationswegen wie z. B. per Telefon oder Hauspost und sollen zukünftig durch Email-Anfragen oder elektronische Formulare abgelöst werden. Der Mitarbeiter prüft in dem DMS über eine Suchanfrage, ob bereits in der Vergangenheit eine ähnliche Anfrage zu einem Rechercheauftrag geführt hat. Sind keine Rechercheergebnisse vorhanden bzw. können bereits recherchierte Patentschriften ggfs. in der neuen Recherche wiederverwendet werden, ist ein Rechercheauftrag zu eröffnen (Funktion „Rechercheauftrag eröffnen“).

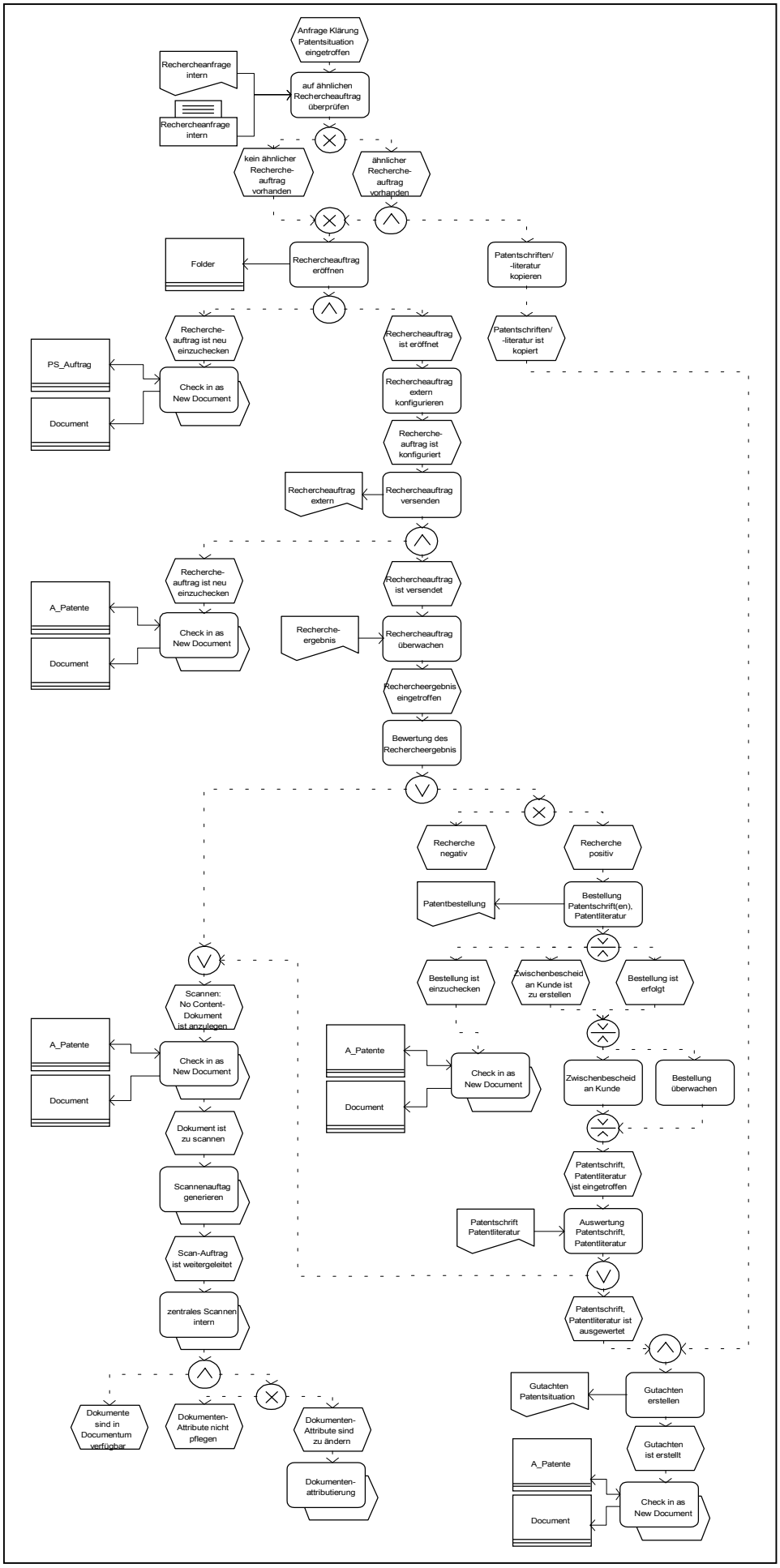

Abb. 10: Prozeß „Patentrecherche“ – Integration von UML in EPK

Literaturverzeichnis

Bungert, Winfried; Heß, Helge: Objektorientierte Geschäftsprozeßmodellierung, in: Information Management, 1/1995, S. 52-63.

Burkhard, Rainer: UML – Unified Modeling Language: Objektorientierte Modelle für die Praxis, 2. Aufl., Bonn: Addison-Wesley 1998.

Fowler, Martin; Scott, Kendall: UML konzentriert: Die neue Standard-Objektmodellierungssprache anwenden, Bonn: Addison-Wesley 1998.

Hirschmann, Petra; Scheer, August-Wilhelm: Konzeption einer DV-Unterstützung für das überbetriebliche Prozeßmanagement, in: Arbeitsberichte des Instituts für Wirtschaftsinformatik, Heft 113/1994, Hrsg.: Scheer, August-Wilhelm, Saarbrücken 1992.

Loos, Peter; Allweyer, Thomas: Process-Orientation and Object-Orientation - An approach for integration UML and Event-driven Process Chains (EPC), in: Arbeitsberichte des Instituts für Wirtschaftsinformatik, Heft 144/1998, Hrsg.: Scheer, August-Wilhelm, Saarbrücken 1998.

Object Management Group (Hrsg.): OMG Unified Modeling Language Specification (draft), Version 1.3 alpha R2, 1/1999.

Oestereich, Bernd: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language; 4., akt. Auflage, München et al.: Oldenbourg 1998.

Schäfer, Steffen: Objektorientierte Entwurfsmethoden: Verfahren zum objektorientierten Systementwurf im Überblick, , Bonn et al.: Addison-Wesley 1994.

Scheer, August-Wilhelm: ARIS – Modellierungsmethoden, Metamodelle, Anwendungen, 3., völlig neubearb. und erw. Aufl., Berlin et al.: Springer 1998.

Scheer, August-Wilhelm: ARIS – Vom Geschäftsprozeß zum Anwendungssystem, Anwendungssystem, 3., völlig überarb. und erw. Aufl., Berlin et al.: Springer 1998.

Scheer, August-Wilhelm; Nüttgens, Markus, Zimmermann, Volker: Objektorientierte Ereignisgesteuerte Prozeßkette (oEPK) - Methode und Anwendung, in: Arbeitsberichte des Instituts für Wirtschaftsinformatik, Heft 141/1997, Hrsg.: Scheer, August-Wilhelm, Saarbrücken 1997.

Seemann, Jürgen; Wolff von Gudenberg, Jürgen: UML – Unified Modeling Language; in: Informatik Spektrum, 21/1998, S. 89-90.

Seubert, Michael: Business-Objekte und objektorientiertes Prozeßdesign, in: Referenzmodelleriung: State-of-the-Art und Entwicklungsperspektiven, Hrsg.: Becker, Jörg; Rosemann, Michael; Schütte, Reinhard, Heidelberg: Physica-Verlag 1999, S. 107-128.

Bisher erschienen

Stand: Dezember 2000 – Den aktuellen Stand der Reihe erfahren Sie über unsere Web Site unter http//wi.uni-giessen.de

Nr. 1/1996	Grundlagen des Client/Server-Konzepts	Schwickert/Grimbs
Nr. 2/1996	Wettbewerbs- und Organisationsrelevanz des Client/Server-Konzepts	Schwickert/Grimbs
Nr. 3/1996	Realisierungsaspekte des Client/Server-Konzepts	Schwickert/Grimbs
Nr. 4/1996	Der Geschäftsprozeß als formaler Prozeß - Definition, Eigenschaften, Arten	Schwickert/Fischer
Nr. 5/1996	Manuelle und elektronische Vorgangssteuerung	Schwickert/Rey
Nr. 6/1996	Das Internet im Unternehmen - Neue Chancen und Risiken	Schwickert/Ramp
Nr. 7/1996	HTML und Java im World Wide Web	Gröning/Schwickert
Nr. 8/1996	Electronic-Payment-Systeme im Internet	Schwickert/Franke
Nr. 9/1996	Von der Prozeßorientierung zum Workflow-Management - Teil 1: Grundgedanken, Kernelemente, Kritik	Maurer
Nr. 10/1996	Von der Prozeßorientierung zum Workflow- Management - Teil 2: Prozeßmanagement und Workfflow	Maurer
Nr. 11/1996	Informationelle Unhygiene im Internet	Schwickert/Dietrich/Klein
Nr. 12/1996	Towards the theory of Virtual Organisations: A description of their formation and figure	Appel/Behr
Nr. 1/1997	Der Wandel von der DV-Abteilung zum IT-Profitcenter: Mehr als eine Umorganisation	Kargl
Nr. 2/1997	Der Online-Markt - Abgrenzung, Bestandteile, Kenngrößen	Schwickert/Pörtner
Nr. 3/1997	Netzwerkmanagement, OSI Framework und Internet SNMP	Klein/Schwickert
Nr. 4/1997	Künstliche Neuronale Netze - Einordnung, Klassifikation und Abgrenzung aus betriebswirtschaftlicher Sicht	Strecker/Schwickert
Nr. 5/1997	Sachzielintegration bei Prozeßgestaltungsmaßnahmen	Delnef
Nr. 6/1997	HTML, Java, ActiveX - Strukturen und Zusammenhänge	Schwickert/Dandl
Nr. 7/1997	Lotus Notes als Plattform für die Informationsversorgung von Beratungsunternehmen	Appel/Schwaab
Nr. 8/1997	Web Site Engineering - Modelltheoretische und methodische Erfahrungen aus der Praxis	Schwickert
Nr. 9/1997	Kritische Anmerkungen zur Prozeßorientierung	Maurer/Schwickert
Nr. 10/1997	Künstliche Neuronale Netze - Aufbau und Funktionsweise	Strecker
Nr. 11/1997	Workflow-Management-Systeme in virtuellen Unternehmen	Maurer/Schramke
Nr. 12/1997	CORBA-basierte Workflow-Architekturen - Die objektorientierte Kernanwendung der Bausparkasse Mainz AG	Maurer
Nr. 1/1998	Ökonomische Analyse Elektronischer Märkte	Steyer
Nr. 2/1998	Demokratiepolitische Potentiale des Internet in Deutschland	Muzic/Schwickert
Nr. 3/1998	Geschäftsprozeß- und Funktionsorientierung - Ein Vergleich (Teil 1)	Delnef
Nr. 4/1998	Geschäftsprozeß- und Funktionsorientierung - Ein Vergleich (Teil 2)	Delnef
Nr. 5/1998	Betriebswirtschaftlich-organisatorische Aspekte der Telearbeit	Polak
Nr. 6/1998	Das Controlling des Outsourcings von IV-Leistungen	Jäger-Goy
Nr. 7/1998	Eine kritische Beurteilung des Outsourcings von IV-Leistungen	Jäger-Goy
Nr. 8/1998	Online-Monitoring - Gewinnung und Verwertung von Online-Daten	Guba/Gebert
Nr. 9/1998	GUI - Graphical User Interface	Maul
Nr. 10/1998	Institutionenökonomische Grundlagen und Implikationen für Electronic Business	Schwickert
Nr. 11/1998	Zur Charakterisierung des Konstrukts “Web Site”	Schwickert
Nr. 12/1998	Web Site Engineering - Ein Komponentenmodell	Schwickert
Nr. 1/1999	Requirements Engineering im Web Site Engineering – Einordnung und Grundlagen	Schwickert/Wild
Nr. 2/1999	Electronic Commerce auf lokalen Märkten	Schwickert/Lüders
Nr. 3/1999	Intranet-basiertes Workgroup Computing	Kunow/Schwickert
Nr. 4/1999	Web-Portale: Stand und Entwicklungstendenzen	Schumacher/Schwickert
Nr. 5/1999	Web Site Security	Schwickert/Häusler
Nr. 6/1999	Wissensmanagement - Grundlagen und IT-Instrumentarium	Gaßen
Nr. 7/1999	Web Site Controlling	Schwickert/Beiser
Nr. 8/1999	Web Site Promotion	Schwickert/Arnold
Nr. 9/1999	Dokumenten-Management-Systeme – Eine Einführung	Dandl
Nr. 10/1999	Sicherheit von eBusiness-Anwendungen – Eine Fallstudie	Harper/Schwickert
Nr. 11/1999	Innovative Führungsinstrumente für die Informationsverarbeitung	Jäger-Goy
Nr. 12/1999	Objektorientierte Prozeßmodellierung mit der UML und EPK	Dandl
Nr. 1/2000	Total Cost of Ownership (TCO) – Ein Überblick	Wild/Herges
Nr. 2/2000	Implikationen des Einsatzes der eXtensible Markup Language – Teil 1: XML-Grundlagen	Franke/Sulzbach
Nr. 3/2000	Implikationen des Einsatzes der eXtensible Markup Language – Teil 2: Der Einsatz im Unternehmen	Franke/Sulzbach
Nr. 4/2000	Web-Site-spezifisches Requirements Engineering – Ein Formalisierungsansatz	Wild/Schwickert
Nr. 5/2000	Elektronische Marktplätze – Formen, Beteiligte, Zutrittsbarrieren	Schwickert/Pfeiffer
Nr. 6/2000	Web Site Monitoring – Teil 1: Einordnung, Handlungsebenen, Adressaten	Schwickert/Wendt
Nr. 7/2000	Web Site Monitoring – Teil 2: Datenquellen, Web-Logfile-Analyse, Logfile-Analyzer	Schwickert/Wendt
Nr. 8/2000	Controlling-Kennzahlen für Web Sites	Schwickert/Wendt
Nr. 9/2000	eUniversity – Web-Site-Generierung und Content Management für Hochschuleinrichtungen	Schwickert/Ostheimer/Franke

Bestellung (bitte kopieren, ausfüllen, zusenden/zufaxen)

Adressat:

Professur für BWL und Wirtschaftsinformatik
Fachbereich Wirtschaftswissenschaften
Licher Straße 70
D – 35394 Gießen
Telefax: (0 641) 99-22619

Hiermit bestelle ich gegen Rechnung die angegebenen Arbeitspapiere zu einem Kostenbeitrag von DM 10,- pro Exemplar (MwSt. entfällt) zzgl. DM 5,- Versandkosten pro Sendung.

Nr.	An
1/1996	
2/1996	
3/1996	
4/1996	
5/1996	
6/1996	
7/1996	
8/1996	
9/1996	
10/1996	
11/1996	
12/1996	

Nr.	An
1/1997	
2/1997	
3/1997	
4/1997	
5/1997	
6/1997	
7/1997	
8/1997	
9/1997	
10/1997	
11/1997	
12/1997	

Nr.	Anz
1/1998	
2/1998	
3/1998	
4/1998	
5/1998	
6/1998	
7/1998	
8/1998	
9/1998	
10/1998	
11/1998	
12/1998	

Nr.	Anz
1/1999	
2/1999	
3/1999	
4/1999	
5/1999	
6/1999	
7/1999	
8/1999	
9/1999	
10/1999	
11/1999	
12/1999	

Nr.	Anz
1/2000	
2/2000	
3/2000	
4/2000	
5/2000	
6/2000	
7/2000	
8/2000	
9/2000	

Absender:

_____
Organisation

_____
Abteilung

_____
Nachname, Vorname

_____
Straße

_____
Plz/Ort

_____
Telefon Telefax eMail

_____
Ort, Datum Unterschrift