Schwegmann, Ansgar; Schlagheck, Bernhard

Working Paper

Integration der Prozeßorientierung in das objektorientierte Paradigma: Klassenzuordnungsansatz vs. Prozeßklassenansatz

Arbeitsberichte des Instituts für Wirtschaftsinformatik, No. 60

Provided in Cooperation with:
University of Münster, Department of Information Systems

Arbeitsberichte des Instituts für Wirtschaftsinformatik

Herausgeber: Prof. Dr. J. Becker, Prof. Dr. H. L. Grob, Prof. Dr. St. Klein,

Prof. Dr. H. Kuchen, Prof. Dr. U. Müller-Funk, Prof. Dr. G. Vossen

Arbeitsbericht Nr. 60

Integration der Prozeßorientierung in das objektorientierte Paradigma: Klassenzuordnungsansatz vs. Prozeßklassenansatz

Ansgar Schwegmann, Bernhard Schlagheck

Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster

Steinfurter Str. 107, 48149 Münster, Tel. (0251) 83-38100, Fax (0251) 83-38109

Dezember 1997

Inhalt

Zusammenfassung

In den aktuell verfügbaren Methoden zur objektorientierten Softwareentwicklung wird der Modellierung und Implementierung von Geschäftsprozessen nur eine geringe oder keine Bedeutung beigemessen. Daraus resultiert das Problem, daß Anforderungen durch neue oder geänderte Geschäftsprozesse nicht direkt, d. h. ohne Methodenbruch, in die objektorientierte Softwareentwicklung einfließen können. Basierend auf dieser Erkenntnis wird ein Ansatz vorgestellt, der darauf abzielt, durch die explizite Berücksichtigung von Geschäftsprozessen und durch eine Trennung von Objekt- und Prozeßwissen, die Anpaßbarkeit und Erweiterbarkeit objektorientierter Informationssysteme zu steigern.

1 Motivation

Die kontinuierliche Optimierung und Anpassung der Aufbau- und Ablauforganisation ist zu einem wesentlichen Erfolgsfaktor für Unternehmen geworden. Insbesondere die Reorganisation der Prozesse[1] (Business Process Reengineering (BPR)) wird seit einigen Jahren intensiv in Theorie und Praxis diskutiert.[2] In diesem Zusammenhang wird der Informationstechnologie die Rolle eines Katalysators für das Business Process Reengineering zugewiesen.[3] Informations- und Kommunikationssysteme müssen die Prozesse eines Unternehmens optimal unterstützen, um Ziele wie beispielsweise hohe Qualität, Nähe zum Markt und Reduktion der Kosten zu erreichen.

An Informationssysteme lassen sich daraus verschiedene Anforderungen ableiten, wobei die flexible Anpaßbarkeit, Steuerbarkeit und Erweiterbarkeit betriebswirtschaftlicher Anwendungssysteme besonders wichtig erscheint. Es gibt verschiedene Konzepte und Technologien, mit denen versucht wird, dieses Ziel zu erreichen (z. B. objektorientierte Softwareentwicklung, Workflow-Management, Computer Aided Softwareengineering, Component Ware).

Die objektorientierte Softwaretechnologie hat in den letzten Jahren verstärkt an Bedeutung gewonnen.[4] Im Gegensatz zur funktionsorientierten Softwareentwicklung werden bei der objektorientierten Softwareentwicklung Struktur und Verhalten zu einer Einheit - dem Objekt bzw. der Klasse - gekapselt.[5] Neben einem hohen Wiederverwendungsgrad, einer durchgängigen Softwareentwicklung von der *Application* zur *Solution Domain* werden die leichtere Wartbarkeit, Anpaßbarkeit und Erweiterbarkeit als Vorteile der objektorientierten Softwareentwicklung genannt.[6] Aufgrund dieser Merkmale ist die objektorientierte Softwareentwicklung eine gute Ausgangsbasis (ihr Einsatz aber keine hinreichende Bedingung!), um die

---

1 Es existiert eine Fülle von Definitionen und eine Menge von Synonymen für den Begriff Geschäftsprozeß. Im folgenden wird dem Begriffsverständnis von BECKER und VOSSEN gefolgt, die einen Prozeß definieren als „inhaltlich abgeschlossene, zeitliche und sachlogische Abfolge von Funktionen, die zur Bearbeitung eines betriebswirtschaftlich relevanten Objektes notwendig sind. (...) Eine besondere Untermenge dieser Prozesse sind die Geschäftsprozesse. Die Geschäftsprozesse einer Unternehmung repräsentieren ihre Geschäftsarten, ergeben sich damit aus den obersten Sachzielen und weisen zwingend Schnittstellen zu externen Marktpartnern auf.“ Becker/Vossen (1995), S. 19.

2 Vgl. Davenport (1993), Hammer/Champy (1993).

3 Vgl. Davenport (1993), S. 37 ff.

4 Vgl. Schäfer (1994), S. 57 ff.

5 Eine umfassende Beschreibung des objektorientierten Paradigmas findet sich beispielsweise in Meyer (1997); Jacobson et al. (1992); Booch (1991); Rumbaugh et al. (1991).

6 Vgl. Schäfer (1994), S. 58 ff.

Restrukturierung und laufende Anpassung der Aufbau- und Ablauforganisation informationstechnisch zu unterstützen.

Bei der Implementierung eines betriebswirtschaftlichen Prozesses in einem objektorientierten Softwaresystem wird die Prozeßlogik auf alle beteiligten Objekte bzw. die entsprechenden Methoden verteilt. Dieses Vorgehen wird im folgenden als *Klassenzuordnungsansatz* bezeichnen. Durch die Verteilung des Prozeßwissens ist bei diesem Ansatz die softwaretechnische Umsetzung von betriebswirtschaftlichen Prozeßänderungen im Vergleich zum nachfolgend skizziertem Ansatz aufwendiger.

Um der Forderung nach einer kontinuierlichen Prozeßoptimierung besser gerecht zu werden, ist die Verwendung des *Prozeßklassenansatzes* ein möglicher Lösungsweg. Kern dieses Konzepts ist, daß durch die Extrahierung des Prozeßwissens in eine spezifische Klasse (Prozeßklasse) die Anpassungs- und Erweiterungsfähigkeit der Anwendungssysteme gesteigert werden kann.

Im folgenden wird in Kapitel 2 zunächst kurz auf die Abbildung von Prozessen im Rahmen der objektorientierten Analyse eingegangen. Darauf aufbauend wird im Kapitel 3 der Klassenzuordnungs- und in Kapitel 4 der Prozeßklassenansatz dargestellt. In Kapitel 5 werden Kriterien erarbeitet, unter welchen Bedingungen welchem der Ansätze der Vorzug zu geben ist. Kapitel 6 beinhaltet ein abschließendes Resümee und einen Ausblick.

2 State-of-the-art bei der Abbildung von Prozessen im Rahmen der objektorientierten Analyse

Um die Vorteile der objektorientierten Softwareentwicklung konsequent nutzen zu können, ist eine durchgehende Berücksichtigung dieses Paradigmas in allen Phasen der Softwareentwicklung notwendig. Die Entwicklung objektorientierter Softwaresysteme läßt sich grob in die Phasen Analyse, Design und Implementierung einteilen.[7] Im folgenden wird primär die Analysephase betrachtet. In dieser Phase werden die betriebswirtschaftlichen Prozesse beschrieben, die die Grundlage für die Arbeit der folgenden Phasen bilden.

In der Analysephase wird zunächst ein konzeptionelles Modell der zu erstellenden Software unabhängig von der Zielumgebung (Software- bzw. Hardwareplattform) entwickelt. Eine

Vielzahl von objektorientierten Analysemethoden sind für diesen Zweck in den letzten Jahren vorgestellt worden.[8] I. d. R. unterscheiden diese Methoden zwischen einem statischen und einem dynamischen Modell. Statische Modelle enthalten die Beziehungen zwischen Klassen bzw. zwischen Objekten. Dynamische Modelle bilden die Interaktionen der Klassen bzw. genauer der Objekte zur Laufzeit ab. Zur Abbildung von Statik und Dynamik sind verschiedene Diagrammtechniken entwickelt worden, die sich zwischen den einzelnen objektorientierten Analysemethoden nur geringfügig unterscheiden. Im Kontext dieses Beitrags ist die Modellierung der Dynamik bzw. der betriebswirtschaftlichen Prozesse wichtig.

Die Unified Modeling Language (UML)[9], die sich voraussichtlich als Standardnotation[10] zur objektorientierten Analyse etablieren wird, unterstützt vier verschiedene Diagrammtechniken zur Abbildung von Dynamik: Use Case-, Sequenz-, State Transition- und Collaboration-Diagramme. Die verfügbaren Notationen der UML oder anderer objektorientierter Analysemethoden sind eng mit der Informationstechnologie verknüpft. Sie sind jedoch kaum geeignet, betriebswirtschaftliche Prozesse für den in dieser Thematik ungeschulten Endanwender verständlich darzustellen.[11] Use Case-Modelle ermöglichen eine vergleichsweise einfache Darstellung. Es lassen sich aber nur rudimentäre Prozesse modellieren. Komplexe Prozesse mit Fallunterscheidungen, Wiederholungen usw. lassen sich nicht abbilden. Weiterhin ist die Modellierung von Ressourcen oder von Beziehungen zur Aufbauorganisation nicht vorgesehen.

Um diese bestehenden Schwachstellen objektorientierter Analysemethoden bei der Modellierung von Prozessen zu beseitigen, werden in der Literatur unterschiedliche Vorschläge diskutiert.

---

7 Vgl. z. B. Nerson (1992), S. 63-74, Schäfer (1994), S. 76 f. Zur Problematik der evolutionären Softwareentwicklung vgl. Hesse (1997), S. 21-28.

8 Ein Überblick über verschiedene OOA-Methoden findet sich beispielsweise in Stein (1993), S. 317-332 und Schäfer (1994). Die folgenden Ausführungen basieren im wesentlichen auf Untersuchung der OOA-Methoden OMT (Rumbaugh et al. (1991)), OOA nach BOOCH (Booch (1994)), OOA nach COAD und YOURDON (Coad/Yourdon (1991)) und Objectory (Jacobson et al. (1992)). Diese OOA-Methoden wurden wegen ihrer relativ großen Verbreitung bzw. ihres hohen Bekanntheitsgrades aus der Masse der existierenden Methoden ausgewählt.

9 Vgl. Burkhardt (1997), Fowler (1997), Oestereich (1997), Rational Software Corporation (1997). Die UML wurde 1997 von der OMG zum Industriestandard der Software Design Notationen erklärt.

10 Hier wurde bewußt der Begriff „Notation“ statt „Methode“ verwendet, da die UML 1.1 kein Vorgehensmodell enthält.

11 Vgl. Bungert/Heß (1995), S. 54.

BURKHARDT[12] untersucht die bestehenden Vorschläge hinsichtlich der Modellierung dynamischer Aspekte. Ergebnis seiner Untersuchung ist das *Objekt-Prozeßmodell*, das in ein generelles Vorgehensmodell eingebettet ist. Wesentlich bei dieser Vorgehensweise ist, daß der Prozeß die Basis des Systementwurfs darstellt. Objekte werden erst im zweiten Schritt identifiziert und anschließend Prozesse zu den einzelnen Objekten bzw. Klassen zugeordnet.

BUNGERT und HEß[13] stellen ebenfalls den Geschäftsprozeß an den Anfang ihres objektorientierten Ansatzes zur Geschäftsprozeßmodellierung. Ausgangspunkt ihrer Betrachtungen ist die Methodik der Ereignisgesteuerten Prozeßkette, die als Beschreibungsmittel von Geschäftsprozessen dient. Nachdem die Geschäftsprozesse zusammen mit den Fachabteilungen analysiert und modelliert wurden, erfolgt die Identifizierung der relevanten Objekte und anschließend die Zuordnung der einzelnen Teilfunktionen zu den zugehörigen Objekten. Auch hier werden demnach die einzelnen Prozesse zergliedert und auf eine Reihe von Einzelobjekten verteilt. BUNGERT und HEß stellen jedoch in ihrer Vorgehensweise zur Modellierung heraus, daß die Fachanwender mit einer zergliederten objektorientierten Darstellung eines Geschäftsprozesses überfordert sind und einer geschlossene Darstellungsform den Vorzug geben.[14] Eine integrierte Betrachtung von Objekten und Prozessen in einem gemeinsamen Modell findet, wie SCHEER, NÜTTGENS und ZIMMERMANN kritisieren,[15] nicht statt.

SCHEER, NÜTTGENS und ZIMMERMANN entwickeln, aufgrund der fehlenden Integration von Geschäftsprozessen und relevanten Objekten, den Ansatz einer *objektorientierten Ereignisgesteuerten Prozeßkette (oEPK).*[16] In diesem Ansatz werden Geschäftsobjekte, die Gegenstand der Bearbeitung durch einen Geschäftsprozeß sind, zusammen mit den Geschäftsprozessen in einem integrierten Modell dargestellt. Die Darstellungsform ist eine Kombination aus ERM, statischem Klassenmodell und EPK.

Der Anforderung nach einer Orientierung an Geschäftsprozessen wird in den modifizierten Ansätzen nur begrenzt Rechnung getragen. Insbesondere die fehlende Transparenz der Ab-

---

12 Burkhardt (1995).

13 Bungert/Heß (1995).

14 Vgl. Bungert/Heß (1995), S. 62.

15 Vgl. Scheer/Nüttgens/Zimmermann (1997), S. 6.

16 Scheer/Nüttgens/Zimmermann (1997).

lauflogik in den erstellten Objektmodellen führt dazu, daß aus Prozeßänderungen resultierender Anpassungsbedarf des Informationssystems nicht direkt umsetzbar ist.[17]

Zur Darstellung von Prozessen wird im folgenden auf die oEPK-Methode von SCHEER, NÜTTGENS und ZIMMERMANN zurückgegriffen. Abbildung 1 zeigt die verwendeten Modellierungselemente und ein einfaches Beispiel.

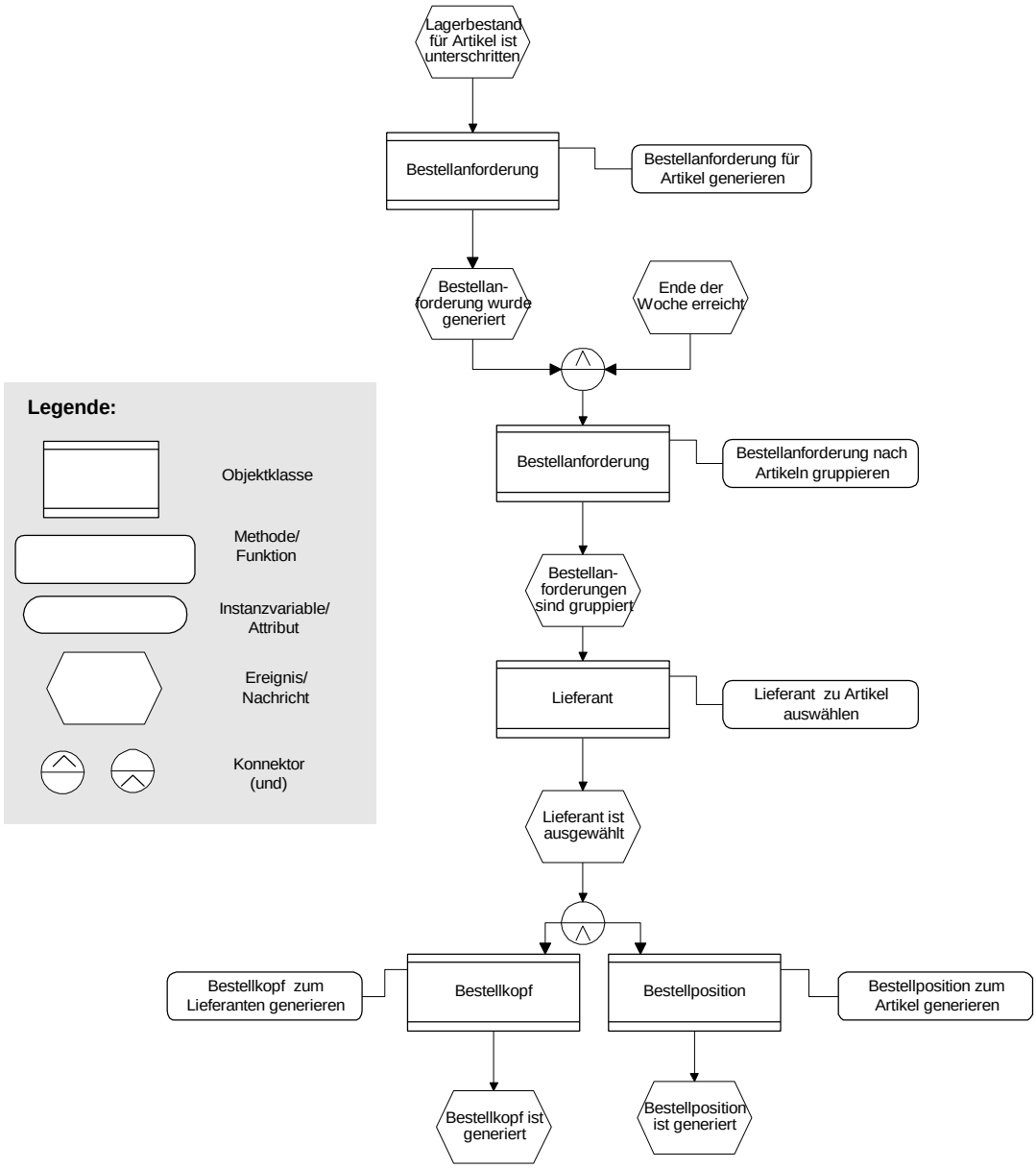

Abbildung 1: Notation und Beispiel zur oEPK

17 Siehe auch Kapitel 3.2.

3 Der Klassenzuordnungsansatz

3.1 Darstellung des Ansatzes

Das hier als *Klassenzuordnungsansatz* bezeichnete Konzept zur objektorientierten Modellierung von betrieblichen Informationssystemen entspricht dem „klassischen“ Vorgehen bei der objektorientierten Softwareerstellung. Bei der Mehrzahl der objektorientierten Softwareengineering-Methoden lassen sich grob folgende Schritte abgrenzen:[18]

1. Identifizieren der Klassen bzw. Objekte
   Im ersten Schritt erfolgt im Rahmen der objektorientierten Analyse die Identifizierung von Klassen bzw. Objekten. Als Basis kann beispielsweise eine Fachspezifikation dienen, die das Systemverhalten beschreibt.
2. Identifizieren und Zuweisen von Attributen und Methoden
   Im zweiten Schritt werden den identifizierten Klassen Attribute und Methoden zugewiesen.
3. Identifizieren der Beziehungen zwischen Klassen und Objekten
   Im dritten Schritt werden die Beziehungen zwischen den Klassen/Objekten modelliert und Klassenmodelle bzw. Objektmodelle erstellt. Sofern dies als notwendig erachtet wird, ist bestimmtes, für die Anwendung besonders wichtiges oder komplexes Verhalten in Zustands- oder Sequenzdiagrammen abzubilden.
4. Design und Implementierung
   Die erstellten Modelle werden anschließend bis zur Implementierung verfeinert.

Die beschriebenen Phasen werden nicht sukzessive durchlaufen, sondern i. d. R. ist eine mehrfache Wiederholung einzelner Schritte notwendig.

Im Kontext dieses Beitrags ist insbesondere Schritt 2, die Identifizierung und Zuordnung von Methoden zu den entsprechenden Klassen interessant. Ziel ist die Definition feingranularer Methoden, die ein exakt definiertes Verhalten aufweisen. Die Modellierung feingranularer Methoden erhöht die Wiederverwendbarkeit.[19]

---

18 Vgl. Schäfer (1994), S. 6, Jacobson et al. (1992), S. 77. Ein ausführlicheres Modell findet sich in Balzert (1997), S. 40 ff.

19 Vgl. Booch (1991), S. 125.

Die betrachteten Modellierungsmethoden geben nur grobe Handlungsanweisungen, wie Methoden zu identifizieren sind. Prinzipiell lassen sich folgende Vorgehensweisen unterscheiden:

- Zugriffs-Methoden aus Attributen definieren
  Elementare Zugriffsoperationen wie Lesen oder Schreiben lassen sich aus den modellierten Attributen einer Klasse ableiten.[20]
- Methoden aus Zustandsübergangsdiagrammen ableiten
  Zustandsübergangsdiagramme betrachten die möglichen Zustände und Zustandsübergänge eines Objektes. Die definierten Zustandsübergänge sind potentielle Kandidaten für eine Methode.
- Methoden aus Ablaufbeschreibungen ableiten
  Im Gegensatz zu Zustandsübergangsdiagrammen betrachten Ablaufbeschreibungen den Kontrollfluß zwischen mehreren Objekten. OMT sieht dazu Datenfluß-Diagramme vor, Objectory oder die UML nutzen Use Case- bzw. Interaktionsdiagramme.[21] Jede Funktion eines Datenfluß-Diagrammes ist als Methode einer Klasse zuzuordnen.[22] In Interaktionsdiagrammen dargestellte Nachrichten machen die Definition einer entsprechenden Methode beim aufgerufenen Objekt erforderlich.
- Methoden sukzessive definieren und verfeinern
  Bei OOA-Methoden wie beispielsweise OOA nach BOOCH, OOA nach COAD/YOURDON oder Objectory definiert der Modellierer auf Basis seines Wissen über die Anwendungsdomäne die Methoden zu einer Klasse.[23] Im Laufe der Zeit bekommt der Modellierer ein tieferes Verständnis von der Anwendungsdomäne, erkennt Zusammenhänge und notwendige Funktionalitäten. Auf Basis neuer Erkenntnisse werden die Klassen und Methoden weiter verfeinert. Dabei kann es sich auch um Methoden handeln, die in dem betrachteten Anwendungsbereich (noch) nicht erforderlich sind, bei denen aber das betrachtete Objekt die Modellierung dieser Methode nahelegt (Shopping List Operations).[24]

---

20 Vgl. Rumbaugh et al. (1991), S. 184, Coad/Yourdon (1991), S. 146 ff.

21 Vgl. Rumbaugh et al. (1991), S. 184 f., Jacobson et al. (1994), S. 126 ff. Zu den Mängeln dieser Notationen siehe Abschnitt 2.

22 Vgl. Rumbaugh et al. (1991), S. 184.

23 Vgl. Coad/Yourdon (1991), S. 143 ff., Booch (1991), S. 125 f., Jacobson et al. (1992), S. 78.

24 Vgl. Rumbaugh et al. (1991), S. 185.

3.2 Bewertung des Ansatzes

Der dargestellte Ansatz weist einige wesentliche Schwachstellen auf, die aus der fehlenden Prozeßorientierung resultieren. Diese werden im folgenden dargestellt.

Keine oder nur rudimentäre Modellierung von Prozessen in der Analysephase

Um betriebswirtschaftliche Anwendungssysteme an sich ändernde Prozesse anpassen zu können, ist es notwendig, die Prozesse im Rahmen einer objektorientierten Analyse explizit zu modellieren. Die Ergebnisse der objektorientierten Analyse determinieren, welche Funktionalitäten und Strukturen in der folgenden Design- und Implementierungsphase realisiert werden. In Abschnitt 2 wurde bereits diskutiert, daß die aktuell verfügbaren OOA-Methoden in dieser Beziehung Defizite aufweisen.

Defizite bei der Wartbarkeit des Systems

Auch wenn betriebswirtschaftliche Prozesse im Rahmen der objektorientierten Analyse nicht oder nur rudimentär modelliert werden, so beinhalten die implementierten objektorientierten Softwaresysteme Informationen über die Ablaufsteuerung der betrieblichen Prozesse. Diese Informationen finden sich in codierter Form verteilt in den Objekten der Anwendung wieder; es erfolgt keine Trennung zwischen Prozeßwissen und Objektwissen. Prozeßwissen beinhaltet Informationen über die Ablaufsteuerung bzw. den Kontrollfluß zwischen mehreren Objekten. Objektwissen repräsentiert Struktur und Verhalten, das sich primär auf ein Objekt bezieht (Struktur des Objektes, mögliche Zustände und Zustandsübergänge, Integritätsbedingungen).[25] Durch die Vermischung von Prozeß- und Objektwissen wird eine Änderung und Erweiterung des Systems erschwert, da die Kopplung der Objekte hoch und die Kohäsion niedrig ist.[26]

Als Kopplung wird der Grad der Abhängigkeit zwischen Modulen (bzw. hier Klassen) bezeichnet.[27] Eine starke Kopplung hat vor allem zur Folge, daß Klassen aufgrund der Abhängigkeiten nicht so einfach wiederverwendet werden können.

Kohäsion wird als Bindung von Elementen eines Moduls untereinander verstanden.[28] Die Abbildung von Objekt- und Prozeßwissen durch entsprechende Attribute und Methoden in

---

25 Vgl. Seubert (1997), S. 59 ff.
26 Vgl. Leymann (1995), S. 91.
27 Vgl. Kappel/Schrefl (1996), S. 202 ff.
28 Vgl. Kappel/Schrefl (1996), S. 226 ff.

einer Klasse wirkt sich negativ auf die Kohäsion dieser Klasse aus. Eine niedrige Kohäsion vermindert die Wartbarkeit und Wiederverwendbarkeit der Klasse.

Mangelnde Transparenz der Ablauflogik

Aus der fehlenden Trennung von Prozeß- und Objektwissen resultiert eine mangelnde Transparenz der Ablauflogik für den Softwareentwickler. Dies macht eine intensive Analyse des Klassenmodells erforderlich, um den Ort einer notwendigen Änderung zu identifizieren, da Teilfunktionen eines Prozesses beim Klassenzuordnungsansatz auf alle beteiligten Objekte verteilt sind.[29] Das Sequenzdiagramm in Abbildung 2 zeigt die Abarbeitung eines vereinfachten Prozesses aus dem Bereich Einkauf, bei dem Bestellungen aus zuvor erzeugten Bestellanforderungen generiert werden. Wenn die Kontrollflüsse oder implementierte Geschäftsregeln geändert werden müssen, ist i. d. R. nicht direkt ersichtlich, in welcher Klasse Änderungen vorzunehmen sind bzw. wo das Prozeßwissen hinterlegt ist.

Bei Änderungsanforderungen an den Prozeß muß der Softwarentwickler zunächst untersuchen, wie die Klassen zusammenarbeiten, damit er im zweiten Schritt die richtigen Klassen anpassen kann. Der beschriebene Einkaufsprozeß könnte beispielsweise dahingehend erweitert werden, daß zu Artikeln, für die Rahmenverträge abgeschlossen wurden, keine Bestellung, sondern eine Abrufaufforderung generiert wird. Es ist leicht nachvollziehbar, daß solche Änderungsanforderungen bei komplexen betriebswirtschaftlichen Anwendungssystemen, die eine Vielzahl von Prozessen unterstützen, erheblichen Aufwand verursachen kann. Des weiteren wird durch das Konzept des Information Hiding[30], das Bestandteil des objektorientierten Paradigmas ist, die Identifikation der relevanten Klassen erschwert. Die Idee der Prozeßorientierung wird folglich durch die Objektorientierung nicht hinreichend unterstützt.[31]

---

29 Vgl. Jacobson et al. (1992), S. 133.

30 Vgl. Meyer (1997), S. 25, S. 51 ff.

31 Vgl. Kueng et al. (1996), S. 49.

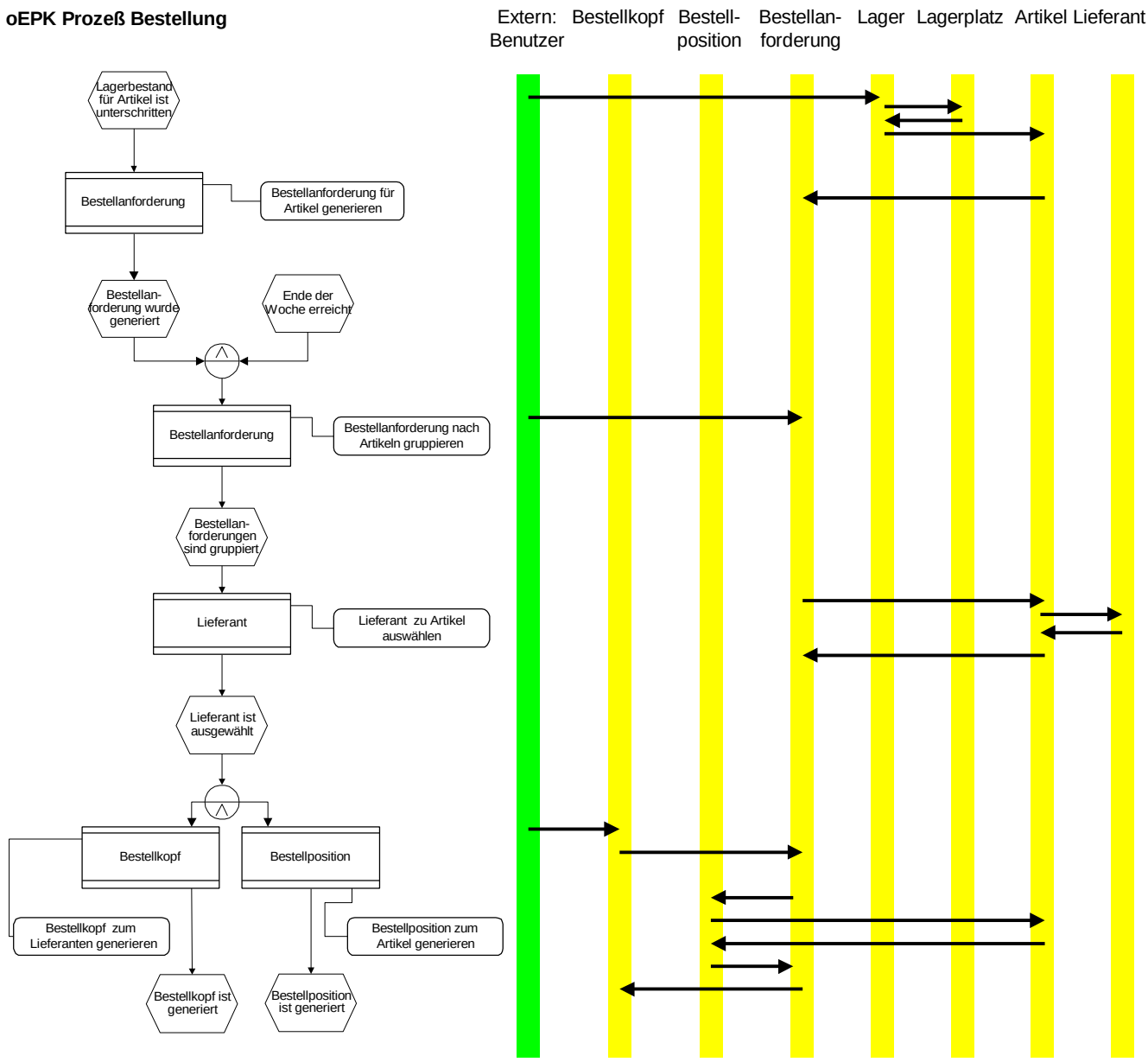

Abbildung 2: Nachrichtenbeziehungen des Prozesses „Bestellung aus Bestellanforderung generieren“[32]

Geringe Unterstützung bei der Identifizierung und Zuordnung von Methoden

In Abschnitt 3.1 wurde beschrieben, wie Methoden zu einem Objekt identifiziert und zugeordnet werden können. Die Identifizierung rudimentärer Zugriffsfunktionen ist dabei unproblematisch. Schwieriger ist die Identifizierung und Zuordnung komplexer Methoden.

---

[32] Die in der EPK abgebildete Ausführung paralleler Zweige ist im Sequenzdiagramm nicht adäquat darstellbar.

Methoden in mehreren Schritten zu definieren und zu verfeinern ist ein praktikabler aber ineffizienter Weg. Einerseits ist nie sichergestellt, daß alle relevanten Methoden gefunden wurden. Andererseits sind möglicherweise zahlreiche Überarbeitungen notwendig, bis ein stabiles Modell als Ergebnis dieses Analyseprozesses entsteht.

Auch die Identifizierung von Methoden über Zustandsdiagramme ist unbefriedigend, da Zustandsdiagramme immer nur ein Objekt betrachten. Beziehungen zu anderen Objekten werden vernachlässigt und es besteht die Gefahr, relevante Methoden zu vergessen.

Vielversprechender ist der Weg, über Ablaufbeschreibungen die notwendigen Methoden zu finden. Bei diesem Vorgehen ist – ein adäquates Prozeßmodell vorausgesetzt – gewährleistet, daß alle betriebswirtschaftlich notwendigen Funktionen identifiziert werden. Es bleibt dann die Aufgabe, gefundene Teilfunktionalitäten zu strukturieren und den Methoden einer Klasse zuzuordnen.[33] Dies ist unter Umständen keine triviale Aufgabe. Folgendes Beispiel soll das verdeutlichen: Die Funktion `Bestellung_aus_Bestellanforderung_generieren` (siehe Beispiel Abbildung 2) könnte sowohl der Klasse `Bestellung`, der Klasse `Bestellanforderung`, der Klasse `Lieferant` als auch der Klasse `Einkauf` zugeordnet werden (letztere repräsentiert eine organisatorische Einheit und ist für die Ausführung der Funktion verantwortlich). Eine redundante Zuordnung sollte vermieden werden, um den Wartungsaufwand zu minimieren. Die richtige Zuordnung der Methode zu einer Klasse beeinflußt die spätere Anpaßbarkeit und Wiederverwendbarkeit der Klassen. Die untersuchten objektorientierten Analysemethoden geben für dieses Problem keine Hilfestellung.[34]

3.3 Fazit

Zusammenfassend läßt sich konstatieren, daß die Idee der Prozeßorientierung durch die Objektorientierung nicht hinreichend unterstützt wird. In allen Phasen der objektorientierten Softwareentwicklung fehlen oder existieren nur mangelhafte Konstrukte zur Repräsentation von Prozessen. Dies wirkt sich negativ auf die Anpassungsfähigkeit der objektorientierten Informationssysteme bei geänderten oder neuen Prozessen aus. Der geforderten Flexibilität der Informationssysteme wird nicht Rechnung getragen.

---

33 Vgl. Hruschka (1996), S. 84-85, Jacobson et al. (1992), S. 191, Müller-Luschnat/Hesse/Heydenreich (1993), S. 79, Kurbel/Teubner (1996), S. 245.

34 In den Publikationen zu den untersuchten Analysemethoden finden sich keine Hinweise bzw. nur grobe Heuristiken, wie das Zuordnungsproblem von Methoden zu Klassen gelöst werden kann. Vgl. Coad, Yourdon (1991), S. 143 ff., Rumbaugh et al. (1991), S. 183 ff., Booch (1991), S. 125 f., Jacobson et al. (1992), S. 78.

4 Der Prozeßklassenansatz

4.1 Darstellung des Ansatzes

Die zentrale Idee des Prozeßklassenansatzes[35] ist es, Ablaufsteuerinformationen betriebswirtschaftlicher Prozesse in einer eigenen Klassen, der Prozeßklasse, zu modellieren und zu implementieren. Eine Instanz dieser Prozeßklasse hat die Aufgabe, einen betriebswirtschaftlichen Prozeß, wie beispielsweise den in Abbildung 4, von der Instantiierung bis zur Archivierung zu steuern. Prozeßklassen stellen für die Steuerung des Prozesses entsprechende Methoden bereit; z. B. `Prozeß_Starten`, `Prozeß_unterbrechen`, `Prozeß_beenden` oder `Prozeßstatus_anzeigen`. Die Attribute einer Prozeßklasse dienen primär dazu, die Ablauflogik des Prozesses, den Prozeßzustand und Verweise auf die im Prozeß herangezogenen *Ressourcenobjekte* zu speichern. Abbildung 3 zeigt beispielhaft die Struktur einer Prozeßklasse.

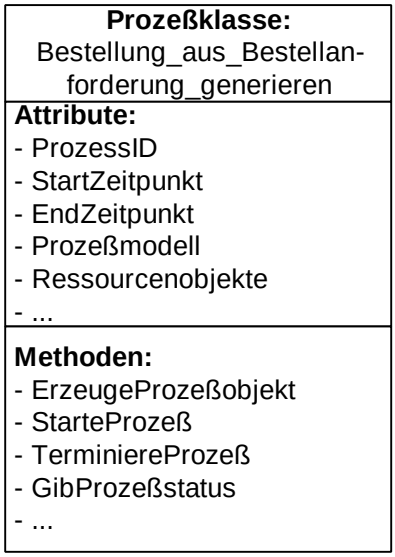

Abbildung 3: Beispiel für eine Prozeßklasse

Die Prozeßklasse nutzt als Client die Dienste von *Ressourcenobjekten*.[36] Zu jeder Aktivität eines Prozesses (der durch das Prozeßobjekt gesteuert wird) erfolgt der Aufruf einer oder mehrerer Methoden von Ressourcenobjekten.[37] Ressourcenobjekte verwalten die Nutzdaten

---

35 Das hier als Prozeßklassenansatz bezeichnete Konzept ist grundsätzlich nicht neu. Eine Auflistung vergleichbarer Ansätze findet sich unten.

36 Andere Bezeichungen für Ressourcenobjekte/-Klassen der Literatur sind beispielsweise *Bestandsklassen* (Müller-Luschnat/Hesse/Heydenreich (1993), S. 82) oder *Informationsobjekte* (Turowski (1997), S. 106 f).

37 Leymann (1995), S. 92 f.

des Informationssystems und enthalten im Gegensatz zu den Prozeßobjekten nur Mikroverhalten.[38] Ressourcenobjekte sind beispielsweise Instanzen der Klassen `Bestellanforderung`, `Lieferant` oder `Artikel`. Abbildung 4 veranschaulicht die Verwendung von Ressourcenobjekten durch Prozeßobjekte auf Basis des oben bereits verwendeten Beispiels.

Um einen Prozeß steuern zu können, enthält eine Prozeßklasse alle relevanten Ablauf- bzw. Kontrollflußinformationen über den entsprechenden Prozeß. Diese Kontrollflußinformationen definieren, welche Methodenaufrufe (von Ressourcenobjekt-Methoden) auf Basis des aktuellen Prozeßzustands möglich sind und welche Folgezustände bei erfolgreicher Methodenausführung erreicht werden. Weiterhin werden ggf. Daten über die Zuordnung zur ausführenden organisatorischen Einheiten (z. B. über das Konstrukt der Rolle) hinterlegt. Die Informationen können entweder in den Methoden des Prozeßobjekts (z. B. in der Methode `Prozeß_Starten`) codiert oder in entsprechenden Datenstrukturen gespeichert werden.

Das Beispiel in Abbildung 4 enthält nur ein Prozeßobjekt. Auch zwischen Prozeßobjekten können Beziehungen bestehen, die hier zur Vereinfachung nicht dargestellt wurden. Zum einen sind hierarchische Beziehungen möglich, bei denen ein untergeordnetes Prozeßobjekt einen Teilprozeß ausführt und anschließend die Kontrolle an den übergeordneten Prozeß zurückgibt. Andererseits existieren Reihenfolgebeziehungen, bei denen ein Prozeßobjekt zu einem bestimmten Zeitpunkt einen oder mehrere Folgeprozesse anstößt.

Das zweite wesentliche Merkmal dieses Ansatzes ist, daß auch das Vorgehen zur Erstellung des Analysemodells von dem des Klassenzuordnungsansatzes abweicht. Zunächst werden (Geschäfts-)Prozeßmodelle erstellt. Diese Prozeßmodelle sind Grundlage zur Identifizierung der Prozeß- und Ressourcenklassen. Es ist zu erwarten, daß das Problem des „how to find the objects“[39] durch diesen Ansatz systematisiert werden kann.[40]

---

38 Siehe Abschnitt 5.

39 Vgl. McDavid (1996), S. 134 ff., Jacobson et al. (1992), S. 72 f., Meyer (1997), S. 719 ff.

40 Vgl. Jacobson et al. (1994), S. 127.

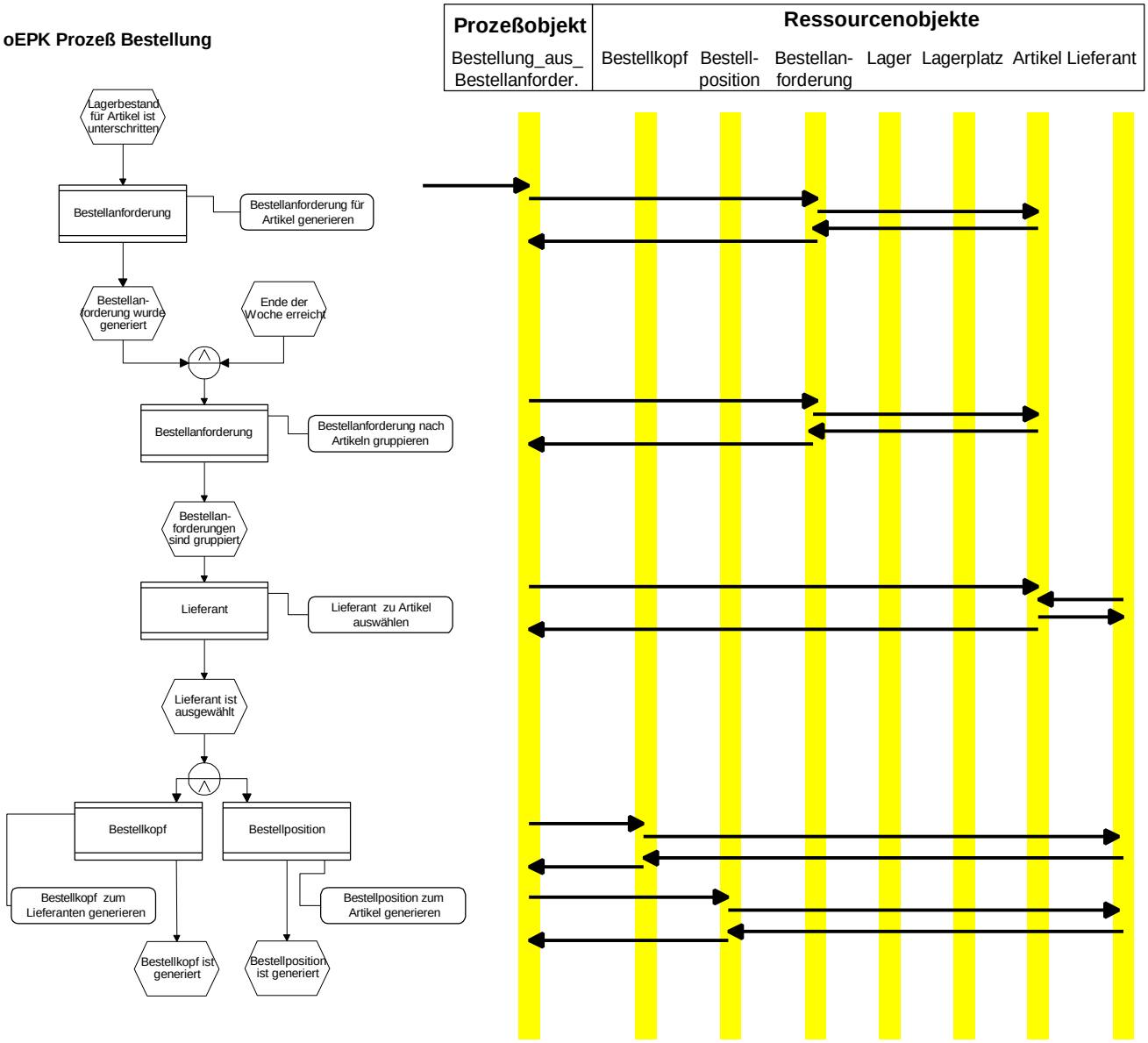

Abbildung 4: Nachrichtenbeziehungen des Prozesses „Bestellung aus Bestellanforderung generieren“ mit Prozeßobjekt

Mit dem Prozeßklassenansatz vergleichbare Ansätze wurden schon von verschiedenen Autoren vorgestellt:

- control objects (Jacobson et al. (1992), S. 190 ff.),
- Geschäftsfallklassen (Kueng/Bichler/Schrefl (1996), S. 48),
- Planungsobjekte (Turowski (1997), S. 100 ff.),
- process objects (Taylor (1995), S. 108 ff.),
- Prozeß-Klassen (Kohl (1996), S. 70 f.),
- Vorgangsobjekttypen (Ferstl/Sinz (1993), S. 160 ff.),

- Vorgangsklasse (Müller-Luschnat/Hesse/Heydenreich (1993), S. 82),
- Workflowtyp-Klasse (Jablonski/Stein (1995), S. 107 f.).

Eine vergleichende Darstellung dieser Ansätze würde den Rahmen dieses Beitrags sprengen. Grundsätzlich basieren alle aufgelisteten Ansätze auf der gleichen Idee: der Steuerung von Prozeßverhalten durch eine eigene Klasse. In den genannten Beiträgen fehlt jedoch eine Diskussion der positiven und negativen Konsequenzen eines Prozeßklassenansatzes und eine Aussage darüber, unter welchen Bedingungen dieser zu bevorzugen ist. Diese Aspekte werden in den folgenden Abschnitten betrachtet.

4.2 Bewertung des Ansatzes

Durch die Verwendung von Prozeßklassen ergeben sich im Vergleich zum Klassenzuordnungsansatz verschiedene Vorteile.

Verbesserte Anpaßbarkeit und Erweiterbarkeit durch die Trennung von Objekt- und Prozeßwissen

Eine Anpassung des Softwaresystems an neue oder geänderte Prozesse wird durch die Trennung von Objekt- und Prozeßwissen erheblich vereinfacht. Bei Veränderungen von Prozessen bzw. des Kontrollflusses muß im Idealfall - wenn die Ressourcenobjekte über alle benötigten Dienstleistungen verfügen und nicht geändert werden müssen – nur eine Klasse, die Prozeßklasse, angepaßt werden.

Weiterhin verbessern sich die Wiederverwendungsmöglichkeiten der Ressourcenobjekte. Durch die Einführung von Prozeßobjekten wird die Kopplung von Ressourcenobjekt-Klassen vermindert und die Kohäsion erhöht. Die Verminderung der Kopplung ergibt sich durch die Vereinfachung der Nachrichtenbeziehungen über die Prozeßobjekte; Ressourcenobjekte bieten elementare Dienstleistungen an, die von einem Prozeßobjekt genutzt werden. Die Kohäsion einer Ressourcenobjektklasse erhöht sich dadurch, daß komplexe Methoden, die einer Ressourcenobjektklasse nicht eindeutig zugeordnet werden können, als Prozeßklassen realisiert werden. Die verbleibenden Methoden einer Klasse weisen insgesamt eine höhere Kohäsion auf.

Verbesserte Anpaßbarkeit und Erweiterbarkeit durch vereinfachte Nachrichtenbeziehungen

Aus der Trennung von Objekt- und Prozeßwissen resultiert eine erhebliche Komplexitätsreduktion des Objektmodells durch übersichtlichere Nachrichtenbeziehungen.[41] Wenn Änderungen an Ressourcenobjekten erforderlich sind, so ist es durch die vereinfachten Nachrichtenbeziehungen leichter, den Ort einer notwendigen Änderung zu identifizieren und die Konsequenzen einer Änderung zu beurteilen (siehe Abbildung 4 im Vergleich zu Abbildung 2).

Einfache Realisierung von Prozeßmanagement-Funktionen

Funktionalitäten zum Prozeßmanagement sind leicht zu ergänzen. Für eine effiziente Steuerung und Kontrolle der Prozesse (Prozeßmonitoring[42]) sind einerseits Daten über die Prozesse zu sammeln (z. B. minimale/durchschnittliche/maximale Durchlaufzeit, Bearbeitungszeit, Liegezeit usw. für einzelne oder gruppierte Prozesse, z. B. nach Mitarbeitern oder bestimmten Auftragsarten). Andererseits sind besondere Funktionalitäten zur Verwaltung der Prozesse erforderlich (z. B. Starten, Unterbrechen, Wiederaufnehmen, Beenden eines Prozesses). Prinzipiell sind alle Funktionen denkbar, die in verfügbaren Workflow-Management Systemen (Workparty (SNI), Flowmark (IBM), usw.) realisiert sind.

Solche Funktionaliäten lassen sich durch eine Prozeßklasse verhältnismäßig leicht realisieren bzw. modifizieren. Beim Klassenzuordnungsansatz hingegen erschwert die Verteilung der Ablauflogik über verschiedene Klassen eine Erweiterung um die beschriebenen Funktionen deutlich. Eine klare Zuordnung von benötigten Attributen und Methoden ist nicht möglich, da i. d. R. mehrere Objekte als Kandidaten für diese Aufgabe in Frage kommen. Des weiteren wird eine umfassende Auswertung der Prozesse durch die Verteilung der Prozeßdaten komplizierter.

Einfache Erstellung von Prozeßvarianten

Auf Basis vorhandener Prozeßklassen können in einfacher Weise Varianten abgeleitet werden. Der objektorientierte Vererbungsmechanismus erleichtert die Erstellung dieser Varianten. Auch eine Wiederverwendung der Ablauflogik ist möglich. In Abhängigkeit davon, wie die Ablaufinformationen in der Prozeßklasse „gespeichert“ sind,[43] wird entweder der Code oder

---

41 Vgl. Jacobson et al. (1992), S. 133, Seubert (1997), S. 60.

42 Vgl. Grob (1996), S. 151 ff.

43 Siehe Abschnitt 4.1.

die hinterlegten Daten modifiziert. MÜLLER-LUSCHNAT, HESSE und HEYDENREICH veranschaulichen, wie eine Vererbung von Ablaufinformationen realisiert werden kann.[44]

Keine Zuordnung von komplexen Methoden zu Klassen erforderlich

Das in Abschnitt 3.2 beschriebene Zuordnungsproblem von Methoden zu Klassen wird vermieden. Das Zuordnungsproblem für komplexe Methoden wird dadurch umgangen, daß die Ablaufinformationen in einer eigenen Klasse, der Prozeßklasse, abgebildet werden.

4.3 Offene Fragen

Modellierung ausführbarer Prozesse

Ein generelles Problem der Prozeßmodellierung bzw. der Modellierung von ausführbaren Prozessen (=Workflow) besteht darin, daß sich zahlreiche betriebswirtschaftliche Prozesse nicht oder nur auf einer hohen Abstraktionsebene abbilden lassen. Ad-hoc-Abläufe, die dynamisch erzeugt werden bzw. sich permanent ändern, lassen sich jedoch nicht durch Prozeßobjekte abbilden und steuern.[45] In diesem Fall sind dem Benutzer Mikroprozesse[46] oder Methoden der Ressourcenobjekte zur individuellen Steuerung eines Prozesses zur Verfügung zu stellen.

Prozesse, die eine gewisse Strukturkonstanz aufweisen und somit für die Steuerung durch ein Informationssystem geeignet sind, müssen in adäquater Form modelliert werden.[47] Wenn ein Prozeß beispielsweise zu grob abgebildet wurde, ist es nicht möglich, Prozeßschritte genau einer Methode eines Ressourcenobjektes zuzuordnen (vgl. Abbildung 1).

Funktionen/Daten versus Prozeßobjekte/Ressourcenobjekte

Verfechter der „originären" Objektorientierung werden gegen den vorgestellten Prozeßklassenansatz einwenden, daß die Modellierung und Implementierung von Prozeßklassen der funktionsorientierten Trennung von Funktionen und Daten entspricht. Die Modellierung einer Prozeßklasse für jeden, auch für sehr einfache Prozesse, an denen beispielsweise nur ein Ressourcenobjekt beteiligt ist, würde tatsächlich einer Trennung von Funktionen und Daten nahe kommen und den objektorientierten Ansatz konterkarieren. Es gilt also, Entscheidungsregeln

---

44 Vgl. Müller-Luschnat/Hesse/Heydenreich (1993), S. 82 f.

45 Vgl. Picot/Rohrbach (1995), S. 30 ff.

46 Siehe Abschnitt 5.

47 Vgl. dazu Jablonski et al. (1997), S. 65 ff.

zu formulieren, wann Ablaufsteuerinformationen in einem Ressourcenobjekt zu kapseln sind und wann sinnvollerweise eine Prozeßklasse modelliert werden sollte. Diesem Problem widmet sich Abschnitt 5.

Prozeßklassen versus Workflow-Management-System

Als Vorteil des Prozeßklassenansatzes wurde herausgestellt, daß eine Prozeßklasse um diverse Funktionalitäten erweiterbar ist, die auch in marktgängigen Workflow-Management-Tools verfügbar sind. Hierbei stellt sich die Frage, warum nicht direkt ein solches Tool verwandt wird. Grundsätzlich wäre es denkbar, die Ablaufinformationen einer Prozeßklassen in einem Workflow-Management-Tool zu modellieren und den Prozeß durch dieses Tools steuern zu lassen. Die Implementierung der Prozeßklassen könnte entfallen und die modellierten Ressourcenobjekte würden durch das Workflow-Management-Tool direkt aufgerufen.

Gegen eine generelle Ersetzung von Prozeßklassen durch Workflow-Management-Tools sprechen einerseits technischen Probleme, wie die noch ungeklärte Definition von Schnittstellen und Performanceprobleme. Weiterhin müßte bei diesem Szenario eine sehr aufwendige Modellierung innerhalb des Workflow-Management-Tools erfolgen. Jeder Zugriff auf ein Ressourcenobjekt wäre zu programmieren. Ein hybrider Ansatz, bei dem einfache bzw. strukturkonstante Workflows durch Prozeßklassen und änderungsanfällige Workflows durch Workflow-Management-Tools abgebildet werden, ist möglicherweise der richtige Mittelweg.

5 Prozeßklassenansatz versus Klassenzuordnungsansatz

Wie bereits in Abschnitt 4.3 erläutert wurde, ist der Prozeßklassenansatz nicht in jedem Fall dem Klassenzuordnungsansatz vorzuziehen. Es gilt, einen Weg zu finden, bei dem einerseits die Vorteile der Objektorientierung nicht verloren gehen und andererseits der erforderlichen Anpassungsfreundlichkeit an geänderte bzw. neue Prozesse stärker Rechnung getragen werden kann.

Die Problematik soll im folgenden noch einmal verdeutlicht werden.

Wie in Abschnitt 3.2 ausgeführt wurde, entstehen bei der Verwendung des Klassenzuordnungsansatzes komplexe Nachrichtenbeziehungen (vgl. Abbildung 5) mit den erwähnten Nachteilen für die Änderungs- und Anpassungsfreundlichkeit des Informationssystems.

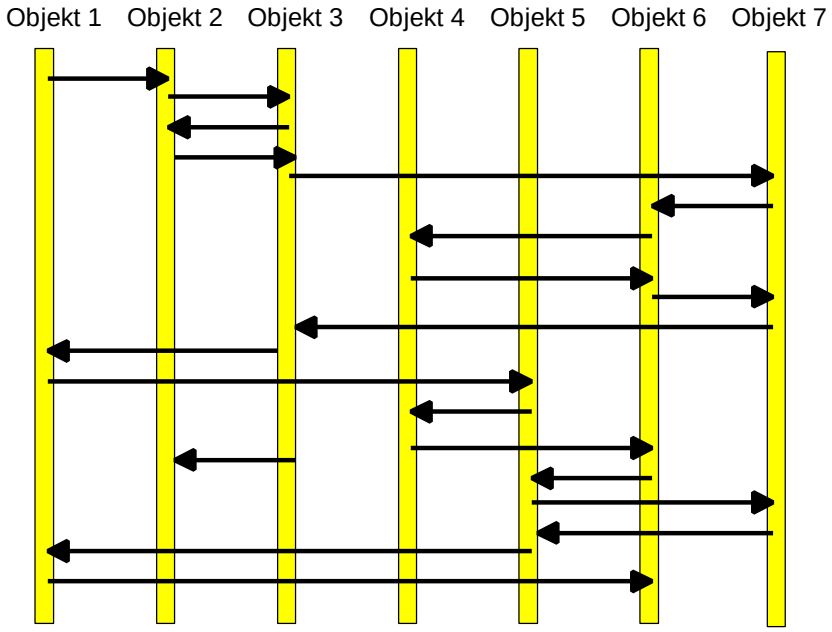

Abbildung 5: Komplexe Nachrichtenbeziehungen
innerhalb des Klassenzuordnungsansatzes

Gemäß dem objektorientierten Paradigma kapseln Objekte Struktur und Verhalten. Um den Anforderungen durch die Prozeßorientierung besser gerecht zu werden, wird die Verhaltenskomponente der Objekte in Mikroverhalten (abgebildet in Ressourcenobjekten bzw. genauer in Ressourcenobjekt*klassen*) und Makroverhalten (abgebildet in Prozeßobjekten bzw. genauer in Prozeßobjekten*klassen*) unterteilt.

Mikroverhalten bzw. ein Mikroprozeß wird als eine Methode eines Ressourcenobjektes implementiert. Diese Methoden steuern die Prozesse, die aus sachlogischen Gründen genau diesem Objekt zuzuordnen sind. Ressourcenobjekte werden dadurch weitgehend kontrollflußunabhängig.[48] D. h. eine Änderung von Prozessen wirkt sich auf die Ressourcenobjekte bzw. Ressourcenobjektklassen nur aus, wenn der Prozeß zusätzliche Dienstleistungen verlangt.

Makroverhalten bzw. ein Makroprozeß wird durch ein Prozeßobjekt gesteuert. Das Prozeßobjekt ist weitgehend unabhängig von den Ressourcenobjekten. D. h. Änderungen der Ressourcenobjekte bzw. der Ressourcenobjektklassen werden i. d. R. keine Auswirkungen auf die Prozeßobjekte haben.

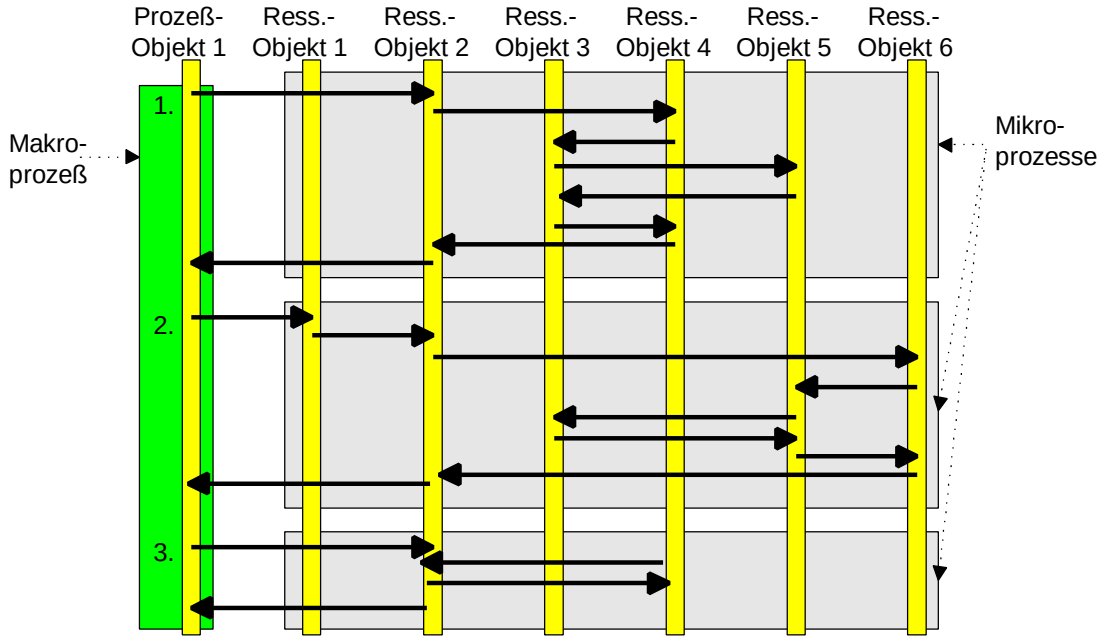

Abbildung 6: Trennung von Mikro- und Makroprozessen

Abbildung 6 zeigt für einen einfachen Fall, wie die Trennung zwischen Mikro- und Makroprozessen erfolgt. In diesem Fall handelt es sich um einen Prozeß, der sequentiell dreimal die Methode eines Ressourcenobjektes aufruft. Beziehungen zu anderen Prozessen wurden hier aus Vereinfachungsgründen nicht dargestellt.

Zu klären bleibt, wie Mikro- und Makroverhalten getrennt werden können bzw. wann Verhalten einem Ressourcenobjekt und wann einem Prozeßobjekt zugewiesen wird. Die folgenden Kriterien können hierbei eine Entscheidungshilfe geben:[49]

---

48 Vgl. Leymann (1995), S. 92.

49 Vgl. Müller-Luschnat/Hesse/Heydenreich (1993), S. 85, sowie Taylor (1995), S. 132.

Arbeitszusammenhang

Der Arbeitszusammenhang drückt aus, inwieweit es betriebswirtschaftlich sinnvoll ist, elementare Aktivitäten zu trennen (Logical Unit of Work).[50] Was betriebswirtschaftlich sinnvoll ist, hängt zudem von der Anwendungsdomäne ab.

Komplexität der Prozesse

Die Komplexität eines Prozesses kann an verschiedenen Maßgrößen festgemacht werden. Neben der Anzahl der beteiligen Ressourcenobjekte ist insbesondere die Komplexität der Ablauflogik ein Kriterium dafür, ob die Implementierung eines Prozeßobjektes sinnvoll ist. Die Komplexität der Ablauflogik drückt sich vor allem durch die Anzahl der parallelen Prozeßstränge und durch die Anzahl der Verzweigungen und Iterationen aus.

Die Implementierung eines Prozeßobjektes erscheint nur sinnvoll, wenn mindestens zwei Ressourcenobjekte an dem Prozeß beteiligt sind und der Prozeß Verzweigungen oder parallele Zweige aufweist bzw. solche Strukturen bei zukünftigen Prozeßerweiterungen benötigt werden.

Änderungshäufigkeit

Die Änderungshäufigkeit ist ein weiteres wichtiges Kriterium dafür, ob der Einsatz eines Prozeßobjektes sinnvoll ist. Wenn ein Ablauf häufig an neue Anforderungen angepaßt werden muß, spricht dies für die Modellierung und Implementierung eines Prozeßobjektes.

Notwendigkeit eines Prozeßmanagements

Wenn Funktionen zum Prozeßmanagement erforderlich sind, lassen sich diese, wie in Abschnitt 4.2 beschrieben, auf Basis von Prozeßklassen leichter realisieren als bei einer Verteilung des Prozeßwissens auf verschiedene Klassen.

Die folgende Tabelle gibt einen Überblick, wann welcher Ansatz zu bevorzugen ist. Zusätzlich ist der in Abschnitt 4.3 skizzierte hybride Ansatz aufgeführt, bei dem Prozesse durch ein Workflow-Management-System gesteuert werden.

---

50 Vgl. Seubert (1997), S. 54.

	Klassenzuordnungs-ansatz	**Prozeßklassenansatz**	**Prozeßklassenansatz und WfM-Tool**
Arbeitszusammenhang	unteilbar	teilbar (abhängig vom Kontext)	teilbar (abhängig vom Kontext)
Komplexität des Prozesses	niedrig	hoch	hoch
Änderungshäufigkeit	selten bis nie	selten bis häufig	häufig
Prozeßmanagement	nicht erforderlich	erforderlich (ggf. erst später)	auf jeden Fall erforderlich

Auswahlmatrix für die beschriebenen Ansätze

6 Resümee und Ausblick

Ausgangspunkt dieses Beitrags war die Notwendigkeit flexibler Softwaresysteme, die sich kontinuierlich an die sich verändernde Aufbau- und insbesondere Ablauforganisation einer Unternehmung anpassen lassen. Die Softwareentwicklung auf Basis des objektorientierten Paradigmas bietet dafür eine gute Grundlage. Jedoch wird bisher die Bedeutung der Modellierung und Implementierung der Ablauforganisation respektive der Prozesse bei der objektorientierten Softwareentwicklung nicht genügend Rechnung getragen.

Der „traditionelle“ Klassenzuordnungsansatz ist nur bedingt geeignet, um flexible, leicht änderbare Prozesse zu implementieren. Dies kann der Prozeßklassenansatz durch eine Trennung von Objekt- und Prozeßwissen leisten. Andererseits ist es nicht immer sinnvoll, eine Prozeßklasse zu modellieren bzw. zu implementieren, da dies im Extremfall der klassischen Trennung von Funktionen und Daten gleichkäme. Kriterien, wann welchem Ansatz der Vorzug gegeben werden sollte, wurden in Abschnitt 5 vorgestellt.

Weiterer Forschungsbedarf besteht insbesondere in folgenden Bereichen. Durch den praktischen Einsatz des Prozeßklassenansatzes ist zu evaluieren, inwieweit sich die angegebenen Vorteile bei großen objektorientierten Softwaresystemen tatsächlich realisieren lassen. Weiterhin sind die objektorientierten Analysemethoden um eine geeignete Prozeßmodellierungsnotation zu ergänzen, um die Softwareentwicklung durchgängig zu gestalten. Es ist zu untersuchen wie das Problem des „how to find the objects“ durch diesen Ansatz beeinflußt wird und wie Vererbung bei der Prozeßmodellierung (Vererbung von Kontrollflußinformationen[51]) genutzt werden kann, um die Objektorientierung und die Prozeßmodellierung zusammenzuführen.

---

51 Vgl. Müller-Luschnat/Hesse/Heydenreich (1993), S. 82 f.

7 Literatur

Balzert, H.: Wie erstellt man ein objektorientiertes Analysemodell? Informatik-Spektrum. 20 (1997) 1, S. 38-47.

Becker, J.; Vossen, G.: Geschäftsprozeßmodellierung und Workflow-Management: Eine Einführung, In: Geschäftsprozeßmodellierung und Workflow-Management. Hrsg.: Vossen, G.; Becker, J. Bonn, Albany 1996, S. 17-26.

Booch, G.: Objectoriented Design with Applications. Redwood City 1991.

Bungert, W.; Heß, H.: Objektorientierte Geschäftsprozeßmodellierung. Information Management. 10 (1995) 1, S. 52-63.

Burkhardt, R.: Modellierung dynamischer Aspekte mit dem Objekt-Prozeß-Modell. Dissertation, Universität Ilmenau, Ilmenau 1995.

Burkhardt, R.: Die Unified Modeling Language. Bonn et al. 1997.

Coad, P; Yourdon, E.: Object-Oriented Analysis. Englewood Cliffs 1991.

Davenport, T. H.: Process innovation: reengineering work through information technology. Boston, Mass. 1993.

Ferstl, O. K.; Sinz, E. J.: Grundlagen der Wirtschaftsinformatik, Bd. 1. München, Wien 1993.

Fowler, M.: UML destilled - Applying the standard Object Modeling Language. Bonn et al. 1997.

Grob, H. L.: Positionsbestimmung des Controlling, In: Rechnungswesen und EDV, Kundenorientierung in Industrie, Dienstleistung und Verwaltung, 17. Saarbrücker Arbeitstagung 1996. Hrsg.: Scheer, A.-W., Heidelberg 1996, S. 137-158.

Hammer, M.; Champy, J.: Reengineering the corporation: a manifesto for business revolution. New York 1993.

Hesse, W.: Wie evolutionär sind die objektorientierten Analysemethoden? Ein kritischer Vergleich. Informatik-Spektrum. 20 (1997) 1, S. 21-28.

Hruschka, P.: Wohin mit den Funktionen im Objektmodell. OBJEKTspektrum. o. J. (1996) 3, S. 84-85.

Jablonski, S.; Böhm, M.; Schulze, W.: Workflow-Management: Entwicklung von Anwendungen und Systemen. Heidelberg 1997.

Jablonski, S.; Stein, K.: Die Eignung objektorientierter Analysemethoden für das Workflow-Management. HMD. o. J. (1995) 185, S. 95-115.

Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.: Object-Oriented Software Engineering: A Use Case Driven Approach. Wokingham et al. 1992.

Jacobson, I.; Ericson, M.; Jacobson, M.: The Object Advantage: Business Process Reengineering with Object Technology. Wokingham et al. 1994.

Kappel, G.; Schrefl, M.: Objektorientierte Informationssysteme: Konzepte, Darstellungsmittel, Methoden. Wien 1996.

Kohl, C.: Objektorientierte Analysekonzepte in der Unternehmensmodellierung, In: Geschäftsprozeßmodellierung und Workflow-Management. Hrsg.: Vossen, G.; Becker, J., Bonn, Albany 1996, S. 63-79.

Kueng, P.; Bichler, P.; Schrefl, M.: Geschäftsprozeßmodellierung: ein zielbasierter Ansatz. Information Management o. J. (1996) 2, S. 40-50.

Kurbel, K.; Teubner, A.: Integrating Information-system Development into Business Process Reenginering: An Evaluation of Software Engineering Paradigms. In: Proceedings of the International Conference on Information Systems Analysis and Synthesis. Hrsg.: Nagib, C.; Callaos, C., Orlando 1996, S. 240-246.

Leymann, F.: Workflows make Objects really useful. In: Proc. of the 6th Int. Workshop on High Performance Systems (HPTS) 17-20 September 1995. Asilomar, CA. 1995.

McDavid, D. W.: Business language analysis for object-oriented information systems. IBM Systems Journal. 35 (1996) 2, S. 128-150.

Meyer, B.: Object-Oriented Software Construction. 2. Aufl., Upper Saddle River, New Jersey 1997.

Müller-Luschnat, G.; Hesse, W.; Heydenreich, N.: Objektorientierte Analyse und Geschäftsvorfallmodellierung, In: Objektorientierte Methoden für Informationssysteme. Hrsg.: Mayr, H. C.; Wagner, R., Berlin, Heidelberg, New York 1993, S. 74-90.

Nerson, J.-M.: Applying Object-Oriented Analysis und Design. Communication of the ACM. 35 (1992) 9, S. 63-74.

Oestereich, B.: Objektorientierte Sotwareentwicklung: Analyse und Design; Mit der Unified Method Language. 2. Aufl., München, Wien, Oldenburg 1997.

Picot, A.; Rohrbach, P.: Organisatorische Aspekte von Workflow-Management-Systemen. Information Management. o. J. (1995) 1, S. 28-35.

Rational Software Corporation et al., UML Summary, Version 1.1, 1997. Online im Internet URL: http://www.rational.com/uml [Stand 24.11.97].

Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.: Object-Oriented Modeling and Design. Englewood Cliffs 1991.

Schäfer, S.: Objektorientierte Entwurfsmethoden: Verfahren zum objektorientierten Softwareentwurf im Überblick. Bonn et al. 1994.

Scheer, A.-W.; Nüttgens, M.; Zimmermann, V.: Objektorientierte Ereignisgesteuerte Prozeßkette (oEPK) - Methode und Anwendung. Arbeitsbericht des Instituts für Wirtschaftsinformatik Saarbrücken, Nr. 141. Saarbrücken 1997.

Seubert, M.: Business Objekte und objektorientiertes Prozeßdesign, In: Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung. Hrsg.: Becker, J.; Rosemann, M.; Schütte, R., Arbeitsberichte des Institut für Wirtschaftsinformatik, Nr. 52, Münster 1997, S. 47-65.

Stein, W.: Objektorientierte Analysemethoden – ein Vergleich. Informatik-Spektrum. 16 (1993) 6, S. 317-332.

Taylor, D. A.: Business engineering with object technologie, New York et al. 1995.

Turowski, K.: Flexible Verteilung von PPS-Systemen. Wiesbaden 1997.

Arbeitsberichte des Instituts für Wirtschaftsinformatik

Nr. 1 Bolte, Ch., Kurbel, K., Moazzami, M., Pietsch, W.: Erfahrungen bei der Entwicklung eines Informationssystems auf RDBMS- und 4GL-Basis; Februar 1991.

Nr. 2 Kurbel, K.: Das technologische Umfeld der Informationsverarbeitung - Ein subjektiver 'State of the Art'-Report über Hardware, Software und Paradigmen; März 1991.

Nr. 3 Kurbel, K.: CA-Techniken und CIM; Mai 1991.

Nr. 4 Nietsch, M., Nietsch, T., Rautenstrauch, C., Rinschede, M., Siedentopf, J.: Anforderungen mittelständischer Industriebetriebe an einen elektronischen Leitstand - Ergebnisse einer Untersuchung bei zwölf Unternehmen; Juli 1991.

Nr. 5 Becker, J., Prischmann, M.: Konnektionistische Modelle - Grundlagen und Konzepte; September 1991.

Nr. 6 Grob, H. L.: Ein produktivitätsorientierter Ansatz zur Evaluierung von Beratungserfolgen; September 1991.

Nr. 7 Becker, J.: CIM und Logistik; Oktober 1991.

Nr. 8 Burgholz, M., Kurbel, K., Nietsch, Th., Rautenstrauch, C.: Erfahrungen bei der Entwicklung und Portierung eines elektronischen Leitstands; Januar 1992.

Nr. 9 Becker, J., Prischmann, M.: Anwendung konnektionistischer Systeme; Februar 1992.

Nr. 10 Becker, J.: Computer Integrated Manufacturing aus Sicht der Betriebswirtschaftslehre und der Wirtschaftsinformatik; April 1992.

Nr. 11 Kurbel, K., Dornhoff, P.: A System for Case-Based Effort Estimation for Software-Development Projects; Juli 1992.

Nr. 12 Dornhoff, P.: Aufwandsplanung zur Unterstützung des Managements von Softwareentwicklungsprojekten; August 1992.

Nr. 13 Eicker, S., Schnieder, T.: Reengineering; August 1992.

Nr. 14 Erkelenz, F.: KVD2 - Ein integriertes wissensbasiertes Modul zur Bemessung von Krankenhausverweildauern - Problemstellung, Konzeption und Realisierung; Dezember 1992.

Nr. 15 Horster, B., Schneider, B., Siedentopf, J.: Kriterien zur Auswahl konnektionistischer Verfahren für betriebliche Probleme; März 1993.

Nr. 16 Jung, R.: Wirtschaftlichkeitsfaktoren beim integrationsorientierten Reengineering: Verteilungsarchitektur und Integrationsschritte aus ökonomischer Sicht; Juli 1993.

Nr. 17 Miller, C., Weiland, R.: Der Übergang von proprietären zu offenen Systemen aus Sicht der Transaktionskostentheorie; Juli 1993.

Nr. 18 Becker, J., Rosemann, M.: Design for Logistics - Ein Beispiel für die logistikgerechte Gestaltung des Computer Integrated Manufacturing; Juli 1993.

Nr. 19 Becker, J., Rosemann, M.: Informationswirtschaftliche Integrationsschwerpunkte innerhalb der logistischen Subsysteme - Ein Beitrag zu einem produktionsübergreifenden Verständnis von CIM; Juli 1993.

Nr. 20 Becker, J.: Neue Verfahren der entwurfs- und konstruktionsbegleitenden Kalkulation und ihre Grenzen in der praktischen Anwendung; Juli 1993.

Nr. 21 Becker, K., Prischmann, M.: VESKONN - Prototypische Umsetzung eines modularen Konzepts zur Konstruktionsunterstützung mit konnektionistischen Methoden; November 1993

Nr. 22 Schneider, B.: Neuronale Netze für betriebliche Anwendungen: Anwendungspotentiale und existierende Systeme; November 1993.

Nr. 23 Nietsch, T., Rautenstrauch, C., Rehfeldt, M., Rosemann, M., Turowski, K.: Ansätze für die Verbesserung von PPS-Systemen durch Fuzzy-Logik; Dezember 1993.

Nr. 24 Nietsch, M., Rinschede, M., Rautenstrauch, C.: Werkzeuggestützte Individualisierung des objektorientierten Leitstands ooL; Dezember 1993.

Nr. 25 Meckenstock, A., Unland, R., Zimmer, D.: Flexible Unterstützung kooperativer Entwurfsumgebungen durch einen Transaktions-Baukasten; Dezember 1993.

Nr. 26 Grob, H. L.: Computer Assisted Learning (CAL) durch Berechnungsexperimente; Januar 1994.

Nr. 27 Kirn, St., Unland, R. (Hrsg.): Tagungsband zum Workshop "Unterstützung Organisatorischer Prozesse durch CSCW". In Kooperation mit GI-Fachausschuß 5.5 "Betriebliche Kommunikations- und Informationssysteme" und Arbeitskreis 5.5.1 "Computer Supported Cooperative Work", Westfälische Wilhelms-Universität Münster, 4.-5. November 1993

Nr. 28 Kirn, St., Unland, R.: Zur Verbundintelligenz integrierter Mensch-Computer-Teams: Ein organisationstheoretischer Ansatz; März 1994.

Nr. 29 Kirn, St., Unland, R.: Workflow-Management mit kooperativen Softwaresystemen: State of the Art und Problemabriß; März 1994.

Nr. 30 Unland, R.: Optimistic Concurrency Control Revisited; März 1994.

Nr. 31 Unland, R.: Semantics-Based Locking: From Isolation to Cooperation; März 1994.

Nr. 32 Meckenstock, A., Unland, R., Zimmer, D.: Controlling Cooperation and Recovery in Nested Transactions; März 1994.

Nr. 33 Kurbel, K., Schnieder, T.: Integration Issues of Information Engineering Based I-CASE Tools; September 1994.

Nr. 34 Unland, R.: TOPAZ: A Tool Kit for the Construction of Application Specific Transaction; November 1994.

Nr. 35 Unland, R.: Organizational Intelligence and Negotiation Based DAI Systems - Theoretical Foundations and Experimental Results; November 1994.

Nr. 36 Unland, R., Kirn, St., Wanka, U., O'Hare, G.M.P., Abbas, S.: AEGIS: AGENT ORIENTED ORGANISATIONS; Februar 1995.

Nr. 37 Jung, R., Rimpler, A., Schnieder, T., Teubner, A.: Eine empirische Untersuchung von Kosteneinflußfaktoren bei integrationsorientierten Reengineering-Projekten; März 1995.

Nr. 38 Kirn, St.: Organisatorische Flexibilität durch Workflow-Management-Systeme?; Juli 1995.

Nr. 39 Kirn, St.: Cooperative Knowledge Processing: The Key Technology for Future Organizations; Juli 1995.

Nr. 40 Kirn, St.: Organisational Intelligence and Distributed AI; Juli 1995.

Nr. 41 Fischer, K., Kirn, St., Weinhard, Ch. (Hrsg.): Organisationsaspekte in Multiagentensystemen; September 1995.

Nr. 42 Grob, H. L., Lange, W.: Zum Wandel des Berufsbildes bei Wirtschaftsinformatikern, Eine empirische Analyse auf der Basis von Stellenanzeigen, Oktober 1995.

Nr. 43 Abu-Alwan, I., Schlagheck, B., Unland, R.: Evaluierung des objektorientierten Datenbankmanagementsystems ObjectStore, Dezember 1995.

Nr. 44 Winter, R., Using Formalized Invariant Properties of an Extended Conceptual Model to Generate Reusable Consistency Control for Information Systems; Dezember 1995.

Nr. 45 Winter, R., Design and Implementation of Derivation Rules in Information Systems; Februar 1996.

Nr. 46 Becker, J.: Eine Architektur für Handelsinformationssysteme; März 1996.

Nr. 47 Becker, J., Rosemann, M. (Hrsg.): Workflowmanagement - State-of-the-Art aus Sicht von Theorie und Praxis, Proceedings zum Workshop vom 10. April 1996; April 1996.

Nr. 48 Rosemann, M., zur Mühlen, M.: Der Lösungsbeitrag von Metadatenmodellen beim Vergleich von Workflowmanagementsystemen; Juni 1996.

Nr. 49 Rosemann, M., Denecke, Th., Püttmann, M.: Konzeption und prototypische Realisierung eines Informationssystems für das Prozeßmonitoring und -controlling; September 1996.

Nr. 50 v. Uthmann, C., Turowski, K. unter Mitarbeit von Rehfeldt, M., Skall, M.: Workflow-basierte Geschäftsprozeßregelung als Konzept für das Management von Produktentwicklungs-prozessen; November 1996.

Nr. 51 Eicker, S., Jung, R., Nietsch, M., Winter, R.: Entwicklung eines Data Warehouse für das Produktionscontrolling: Konzepte und Erfahrungen; November 1996.

Nr. 52 Becker, J., Rosemann, M., Schütte, R. (Hrsg.): Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung, Proceedings zur Veranstaltung vom 10. März 1997; März 1997.

Nr. 53 Loos, P.: Capture More Data Semantic Through The Expanded Entity-Relationship Model (PERM); Februar 1997.

Nr. 54 Becker, J., Rosemann, M. (Hrsg.): Organisatorische und technische Aspekte beim Einsatz von Workflowmanagementsystemen. Proceedings zur Veranstaltung vom 10. April 1997; April 1997.

Nr. 55 Holten, R., Knackstedt, R.: Führungsinformationssysteme - Historische Entwicklung und Konzeption; April 1997.

Nr. 56 Holten, R.: Die drei Dimensionen des Inhaltsaspektes von Führungsinformationssystemen; April 1997.

Nr. 57 Holten, R., Striemer, R., Weske, M.: Ansätze zur Entwicklung von Workflow-basierten Anwendungssystemen - Eine vergleichende Darstellung -, April 1997.

Nr. 58 Kuchen, H.: Arbeitstagung Programmiersprachen, Tagungsband, Juli 1997.

Nr. 59 Vering, O.: Berücksichtigung von Unschärfe im betrieblichen Informationssystem – Einsatzfelder und Nutzenpotentiale am Beispiel der PPS; September 1997.