mehr zum thema:
www.gernert-partner.de

VON ARIS ZUR UML: TRANSFORMATIONEN IN DER PROZESSMODELLIERUNG

Für eine erfolgreiche IT-Systemgestaltung ist ein Zusammenwachsen von prozessorientiertem und objektorientiertem Design mehr denn je gefordert. Die Mitarbeiter der Fachbereiche denken vordergründig prozessorientiert und verwenden für ihre Modelle oft Notationen des ARIS-Ansatzes. Softwareentwickler nutzen dem gegenüber in der Regel die Notationsmöglichkeiten der UML, um Verhalten auszudrücken. Der Artikel stellt eine verlustfreie Transformation zwischen diesen beiden Arten der Verhaltensmodellierung vor. Es wird aufgezeigt, wie sich in ARIS modellierte betriebswirtschaftliche Prozessabläufe in technisch korrekte Steuerungsabläufe der UML 2.0 transformieren lassen.

Geschäftsprozesse bestimmen den IT-Einsatz

Die Dynamik der Märkte, zunehmender Konkurrenzdruck, kurze Innovationszyklen und sich ständig erneuernde Technologien zwingen Unternehmen neue Wege einzuschlagen. Unter diesem Druck vollzog sich in den letzten Jahren ein grundlegender Wandel. Wurden früher die Unternehmensprozesse entlang der eingesetzten Softwaresysteme gestaltet und dadurch mitunter unzulässig eingeschränkt, wird heutzutage nach der Maxime gehandelt, IT-Systeme optimal auf die Unternehmensziele auszurichten. Softwaresysteme unterstützen Geschäftsprozesse und nicht umgekehrt. Dass viele fachliche Anforderungen für die Entwicklung solch angepasster Softwaresysteme bereits in den Geschäftsprozessmodellen stecken, ist leicht nachvollziehbar. Korrekte Geschäftsprozessmodelle unterstützen die klare Abgrenzung fachlicher Anforderungen. Sie sollten deshalb den Softwareentwicklern als Grundlage für die resultierende Systemgestaltung neuer Software dienen (siehe Abb. 1).

In der Praxis stehen beide Seiten jedoch vor der Herausforderung, zunächst eine nicht unerhebliche Sprachbarriere überwinden zu müssen. Während Fachexperte und Organisator sich mit den Prozessen des Unternehmens beschäftigen, lebt der Softwareentwickler in einer Welt der Objekte und Daten. Beide benutzen unterschiedliche Denkansätze und Notationen, um das Verhalten von Systemen zu beschreiben. Konflikte zwischen Fach- und IT-Abteilung durch unpräzise Ausdrucksweisen, unterschiedliche Begriffspezifikationen oder mangelndes gegenseitiges Verständnis sind vorprogrammiert.

Eine Möglichkeit, um solche Missverständnisse zu verhindern, ist, dass die Softwareentwickler die fachliche Welt modellieren und so „Medienbrüche" von Anfang an vermieden werden. Was soll man jedoch tun, wenn das Unternehmen bereits in die betriebswirtschaftliche Prozessmodellierung investiert hat?

Ausgangspunkt von geschäftsprozessgetriebenem Anforderungsmanagement ist ein fundiertes Prozesswissen im Unternehmen. Idealerweise wurden die zu unterstützenden Geschäftsprozesse im Einklang mit einem unternehmensweiten Prozessmanagement bereits analysiert und konzipiert. Wichtige Prozessinformationen liegen bereits in Form von Modellen vor und sollten für das Design der IT-Unterstützung genutzt werden. Die Dokumentation dieser betriebswirtschaftlich orientierten Geschäftsprozessmodelle erfolgt jedoch oft nicht in der Sprache der IT. Die IT-konforme Nachdokumentation des vorhandenen Wissens ist kostenintensiv. Ein Transformationsansatz, der beide Sprachwelten ohne Informationsverlust ineinander überführt, bietet sich als Lösung an.

Nachfolgend wird ein Konzept vorgestellt, das eine verlustfreie Überführung betriebswirtschaftlicher Prozessabläufe (modelliert in ARIS) in technisch interpretierbare Verhaltensbeschreibungen (dargestellt in der UML 2.0) gewährleistet. Die Art und Weise der Transformation sowie einige Knackpunkte, die zu beachten sind, werden aufgezeigt und an einem praktischen Beispiel verdeutlicht. Ziel ist die Minimierung des Informationsverlusts zwischen den beiden Begriffs- bzw. Sprachwelten. Das vorgestellte Transformationskonzept bildet damit einen Ansatzpunkt, Softwareentwicklung erfolgreicher zu gestalten, vor allem jedoch die Möglichkeit bereits getätigte Investitionen langfristig zu sichern.

Prozesse modellieren

Für viele Unternehmen ist die kontinuierliche Optimierung der Geschäftsabläufe eine essenzielle Voraussetzung ihres Geschäfts-

die autoren

Dr.-Ing. Christiane Gernert (E-Mail: christiane.gernert@t-online.de) arbeitet als selbständige Beraterin und Trainerin in den Bereichen Projekt-, Prozess- und Anforderungsmanagement.

Veit Köppen (E-Mail: koeppen@wiwiss.fu-berlin.de) ist wissenschaftlicher Mitarbeiter am Institut für Wirtschaftsinformatik der FU Berlin und arbeitet auf dem Gebiet der Business-Intelligenz.

Oliver Darkow ist in der Haufe Mediengruppe im Bereich IT-Business-Anwendungen tätig. Aktuelle Themen sind unter anderem Multiprojektmanagement und Aufwandschätzung von IT-Projekten.

Thorsten Schwarz (E-Mail: Thorsten.Schwarz@entory.com) ist Consultant bei der Entory AG und arbeitet in der Prozessberatung von Finanzdienstleistern.

erfolges. Erfolgreiche Unternehmen richten ihre Prozesse optimal auf die Kundenbedürfnisse und die Marktsituation aus. Diese mitunter tief greifenden Prozessveränderungen erfordern ein zielgerichtetes und koordiniertes Vorgehen. Die Analyse der Geschäftsprozesse und ihre systematische Aufbereitung in Prozessmodellen stellt das erforderliche Wissen für erfolgreiche Veränderungen bereit. Die Prozessmodellierung unterstützt die systematische und strukturierte Dokumentation von Prozessinformationen.

Ein Geschäftsprozess besteht aus einer Menge miteinander verknüpfter Aktivitäten, die in einer bestimmten Reihenfolge ausgeführt werden, um ein festgelegtes Ziel zu erreichen. Die verschiedenen Aktivitäten können sequenziell und/oder parallel gestartet und ausgeführt werden.

Für die Modellierung von Prozessen existieren unterschiedliche methodische Ansätze und Darstellungsformen. Informelle, meist verbale Darstellungen haben den Nachteil, dass sie nicht immer eindeutig sind. Selbst bei vorgegebenen Beschreibungsschablonen bieten sie einen breiten Spielraum für Interpretationen. Eine weitere Herausforderung besteht in der verbalen Beschreibung von Prozessverzweigungen und Nebenläufigkeiten in einem Prozess. Formale Notationen, wie z.B. Petrinetze, besitzen in diesem Bereich ihre Stärken. Mathematische Modelle ziehen jedoch einen hohen Grad an Abstraktion nach sich und sind insofern nicht immer für jeden *Stakeholder* verständlich. Grafische Notationen vereinfachen das Verständnis und verbessern damit die Kommunikation zwischen allen Beteiligten. Durch eine einheitliche Syntax und Semantik der Notation sind Prozessabläufe auch für Außenstehende leicht nachvollziehbar.

Die Tiefe der Modellierung und die Auswahl von Vorgehen, Methode und Werkzeugunterstützung sind abhängig von den Ausführenden und Beteiligten der Prozessmodellierung sowie dem jeweiligen Modellierungsziel.

Prozessorientierte Verhaltensmodellierung mit ARIS ...

Erfolgt die Analyse von Geschäftsprozessen aus einer prozessorientierten Sicht heraus, ist die Dokumentation des Prozessmodells mit dem ARIS-Toolset der IDS Scheer AG in der Praxis weit verbreitet (vgl. [IDS03]). Zur Beschreibung von Prozessabläufen wird innerhalb der ARIS-Architektur in erster Linie die *erweiterte Ereignisgesteuerte Prozesskette (eEPK)* verwendet.

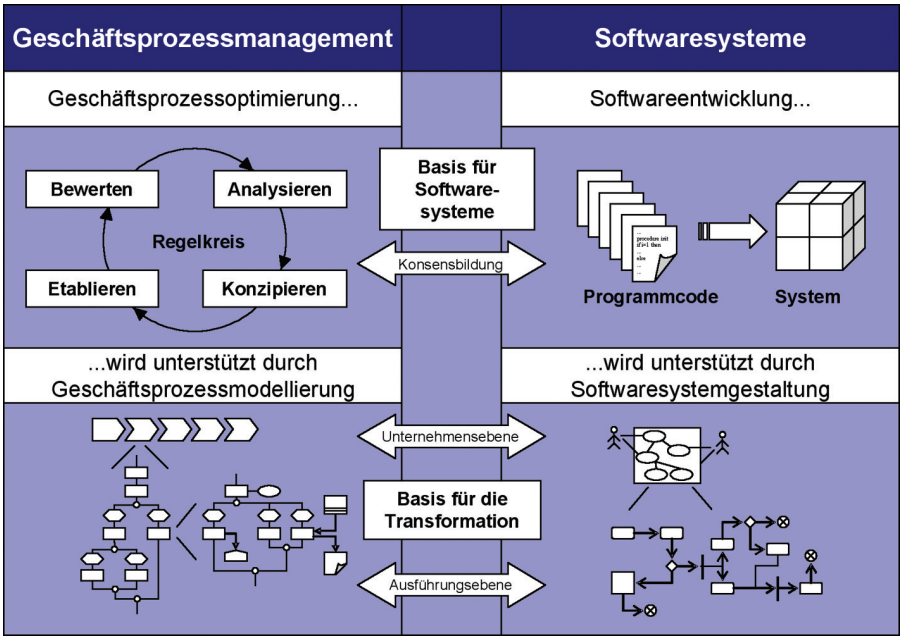


Abb. 1: Geschäftsprozessmanagement und Softwaresysteme

Der Prozessablauf wird als gerichteter Graph mit einer alternierenden Abfolge von Ereignissen und Funktionen beschrieben. Funktionen dienen der Modellierung eines betrieblichen Vorgangs. Ereignisse beschreiben den Zustand eines Informationsobjekts zu einem bestimmten Zeitpunkt im Prozessablauf. Durch Hinzufügen von Verknüpfungsoperatoren kann der Prozessablauf getrennt oder zusammengeführt werden. Man spricht in diesem Zusammenhang auch von der Modellierung des Kontrollflusses eines Geschäftsprozesses, in dem Ereignisse und Funktionen einem wechselseitigen Ablösemechanismus folgen. Die Steuerung des Prozessflusses kann durch drei verschiedene Verknüpfungsoperatoren abgebildet werden:

- Der Verknüpfungsoperator „UND" (UND) verdeutlicht parallele Prozessschritte.
- Zur Darstellung alternativ durchzuführender Funktionen steht das „Exklusive Oder" (XOR) zur Verfügung.
- Wird erst in der Realität entschieden, ob ein oder mehrere der nachfolgenden Teilstränge zu durchlaufen sind, verdeutlicht das „Inklusive Oder" (OR) diesen Sachverhalt im Modell.

In Abhängigkeit von der Zielsetzung der Modells können neben dem Ablauf des Prozesses weitere Prozessinformationen im Modell abgebildet werden, wie zum Beispiel der Datenfluss zwischen den Funktionen, Angaben zu Informationsobjekten, die ausführenden Organisationseinheiten, die produzierten Leistungen, unterstützende Informationssysteme oder notwendige Ressourcen. Kanten verbinden dabei die einzelnen Modellelemente. Werden diese zusätzlichen Informationen modelliert, spricht man von eEPKs.

Abbildung 2 zeigt exemplarisch einen mit den Notationsmitteln der eEPK modellierten Prozess. Von einem Sachbearbeiter (z.B. Agent im Call-Center) wird der Kunde mit Hilfe eines betrieblichen Informationssystems identifiziert.

...versus objektorientierte Verhaltensmodellierung mit UML 2.0

Für die Erstellung von Softwaresystemen hat sich im Rahmen der objektorientierten Modellierung die *Unified Modeling Language (UML)* als Modellierungsstandard etabliert. Mit der Version 2.0 wurde die UML in vielen Bereichen grundlegend überarbeitet (vgl. [Jec04], [OMG04]). Die in der Vergangenheit entstandene Komplexität einiger Diagramme wurde verringert und die Semantik vieler Notationselemente präzisiert. Viele Änderungen konzentrieren sich auf die Verhaltensmodellierung, wobei speziell das für die Prozessmodellierung interessante

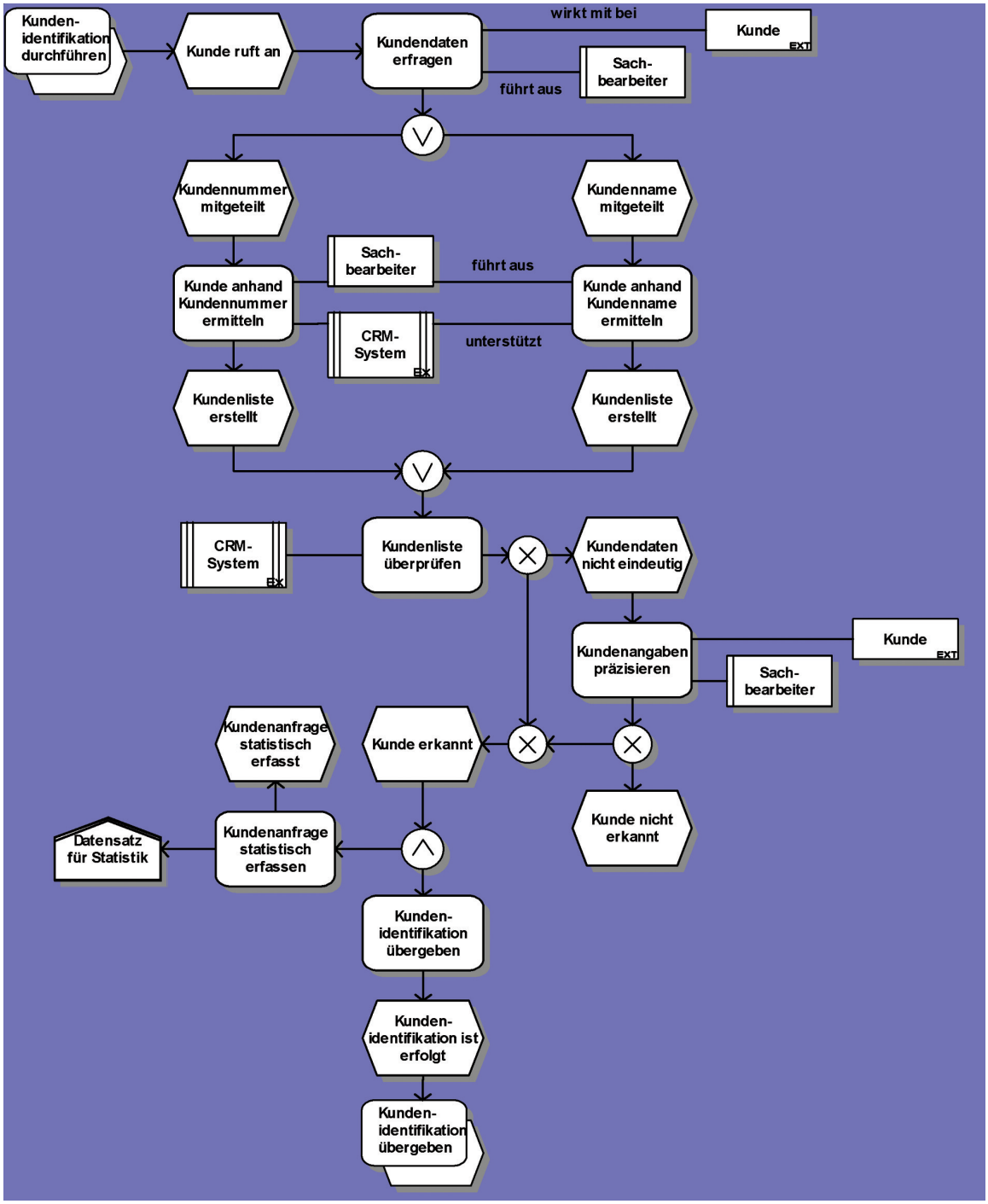


Abb. 2: Beispielprozess Kunde identifizieren (eEPK)

Aktivitätsdiagramm weit reichende neue Notationsmittel bekommen hat.

Ein *Aktivitätsdiagramm* der UML 2.0 beschreibt den Ablauf einer Aktivität anhand eines Grafen von Aktionen (vgl. [Boc03]). Eine Aktion ist eine elementare Einheit einer Verhaltensspezifikation. Die einzelnen Aktionen ähneln dabei stark den Notationselementen moderner Programmiersprachen. Beispielsweise existieren Aktionen zum Lesen bzw. Schreiben von Attributen eines Objekts (ReadStructuralFeatureAction bzw. WriteStructuralFeatureAction) und zum Erzeugen bzw. Vernichten von Objekten (CreateObjectAction bzw. DestroyObjectAction).

Der Ablauf einer Aktivität wird durch ein *Token*-Konzept definiert, das dem des Petrinetzes ähnelt. Kontrollknoten dienen der Steuerung des Ablaufs einer Aktivität. Der Kontrollfluss kann durch Entscheidungsknoten aufgeteilt und durch Zusammenführungsknoten wieder verbunden werden. Parallele Tätigkeiten einer Aktivität werden im Kontrollfluss durch Parallelisierungsknoten gestartet und durch Synchronisationsknoten zusammengeführt. Eine paarweise Verwendung ist dabei nicht mehr notwendig. Startknoten symbolisieren den Beginn einer Aktivität und sind zudem *Token*-Produzenten. Bei den Endknoten wird zwischen „Beenden der Aktivität" und „Beenden des Kontrollflusses" unterschieden. Erreicht ein *Token* einen Endknoten der Aktivität, so löscht dieser alle *Token* und beendet damit die gesamte Aktivität. Dem gegenüber beendet der Endknoten des Kontrollflusses nur diesen Teilstrang und vernichtet lediglich den *Token*, der diesen Endknoten erreicht.

ARIS	UML
Ereignis → Funktion → Ereignis	Aktion als CallBehaviorAction
Leistung	Objektknoten als PIN, Central Buffer, Datastore
Organisations-einheit	Partition als aktive oder passive Partition
Prozess-schnittstelle	Parameterknoten
Kein entsprechendes Symbol	Startknoten
Kein entsprechendes Symbol	Aktivitätsendknoten
Kein entsprechendes Symbol	Kontrollflussendknoten

Tabelle 1: Transformationsregeln

Zusätzlich zum Kontrollfluss kann in einem Aktivitätsdiagramm der Objektfluss dargestellt werden. Zur Beschreibung stehen dafür verschiedene Arten von Objektknoten zur Verfügung. Objektknoten treten hauptsächlich in Form von *Pins* in Erscheinung, die mit einer Aktion verbunden sind und auf diese Weise die eingehenden und ausgehenden Objekte einer Aktion beschreiben. Neben der *Pin*-Notation besteht die Möglichkeit, Objektknoten unabhängig von Aktionen als Sammel- und Verteilungsknoten für das so genannte *Pooling* zu benutzen. Diese werden als CentralBuffer bezeichnet. Als dritte Variante kann ein DataStore zur Verdeutlichung einer persistenten Speicherung verwendet werden.

Um die Verknüpfung von Aktionen und Verantwortlichkeiten darzustellen, ist die Verwendung von Partitionen möglich. Solche Partitionen (auch Aktivitätsbereich genannt) dienen in der UML zur Gruppierung von Aktionen. Partitionen sollten gemeinsame Eigenschaften aufweisen und können hierarchisch oder mehrdimensional modelliert werden.

Das Verhalten innerhalb eines Aktivitätsdiagramms wird ablauforientiert beschrieben. Die beteiligen Objekte spielen eine untergeordnete Rolle. Aus diesem Grund eignet sich das Aktivitätsdiagramm am besten zur Beschreibung von Geschäftsprozessen.

Das Transformationskonzept

Sind die Notationen der eEPKs in ARIS und der Verhaltensmodellierung in der UML 2.0 auch verschieden, so verfolgen beide das gleiche Ziel – das Verhalten eines Systems möglichst korrekt wiederzugeben. Nachfolgend wird beschrieben, wie diese gemeinsame Basis genutzt werden kann, um beide Sprachwelten zu verbinden.

In IT-Systemen muss das Verhalten sehr detailliert beschrieben werden. Darüber hinaus ist in einem objektorientierten Modell jede Verhaltensspezifikation an das Objekt gebunden, dessen Zustand es verändert. Eine objektübergreifende Verhaltensspezifikation existiert zunächst nicht. Das Verhalten des Gesamtsystems ergibt sich nur indirekt aus der Summe des Verhaltens der einzelnen Objekte.

Möchte man eine abstrakte Beschreibung des Verhaltens von einem objektorientierten Modell erstellen, dann muss die Interaktion der Objekte beschrieben werden. Aus diesem Grund unterscheidet die UML 2.0 grundsätzlich zwischen Ausführungsverhalten und auftretendem Verhalten (vgl. [Bor04]). Das Ausführungsverhalten stellt die klassische Form der Verhaltensmodellierung dar. Nur diese Form der Verhaltensspezifikation kann tatsächlich zu Zustandsänderungen im System führen. Dabei kann es sich beispielsweise um eine Methode einer Klasse handeln. Auftretendes Verhalten hingegen ergibt sich aus der Interaktion der Objekte im System. In der Regel werden Sequenzdiagramme als klassische Form der Modellierung für diese Interaktionen von Objekten genutzt.

Mit dem Aktivitätsdiagramm der UML 2.0 können sowohl Ausführungsverhalten als auch auftretendes Verhalten beschrieben werden. Das Aktivitätsdiagramm realisiert die Beschreibung von auftretendem Verhalten durch die Verwendung der CallBehaviorAction. Durch den Aufruf einer Aktivität aus einer zweiten Aktivität heraus wird das Verhalten der ersten Aktivität implizit eingebunden. Auf diese Weise können komplexe Systeme mit beliebig vielen Hierarchiestufen modelliert werden. Durch eine beliebige Hierarchisierung kann eine Angleichung an das Abstraktionsniveau der Funktionen in ARIS erreicht werden.

Die Transformation einer Funktion in ARIS zu einer CallBehaviorAction in einem Aktivitätsdiagramm ist jedoch nicht ohne weiteres möglich. Eine CallBehaviorAction stellt nur den Aufruf einer Verhaltensspezifikation, nicht die Verhaltensspezifikation selbst dar. Die Lösung dieses Problems besteht darin, eine Funktion in eine Aktivität zu transformieren. Diese wird dann von der CallBehaviorAction aufgerufen. Allgemein ist eine Aktivität als eine zeitliche Operation definiert, die zwischen einem Anfangs- und Endereignis liegt und die den Zustand eines Objekts verändern kann. In der EPK wird dieses Konstrukt in der Folge von Ereignis/Funktion/ Ereignis abgebildet. Aus diesem Grund müssen diese drei zusammenhängenden Elemente in eine Aktivität übersetzt werden. Ein in ARIS modellierter Geschäftsprozess, der aus Funktionen und Ereignissen besteht, wird deshalb im UML-Aktivitätsdiagramm als Folge von CallBehavior Actions dargestellt.

Die in einer eEPK dargestellte *Leistung* kann eindeutig als spezialisierter Objektknoten im Aktivitätsdiagramm abgebildet werden. Durch die Modellierung des Leistungs-Outputs in der eEPK heben wir den Gedanken der Kunden- und Leistungsorientierung von Prozessen stärker hervor. Weitere Informationsobjekte werden ebenfalls in spezialisierte Objektknoten transformiert, so zum Beispiel der Dokumenten- und Datenfluss. Um die Übersicht des Modells zu wahren, sollten jedoch Redundanzen vermieden werden.

Prozessschnittstellen sind spezielle Funktionen, die jeweils paarweise auftreten. Sie verknüpfen getrennte Prozessstränge die in

unterschiedlichen eEPKs modelliert wurden. Modellierer nutzen Prozessschnittstellen auch als Elemente zur übersichtlichen Verkettung von hierarchischen eEPKs. Im Aktivitätsdiagramm können Prozessschnittstellen durch Parameterknoten beschrieben werden. Die Funktion der Prozessschnittstelle wird dabei als Parameterübergabe interpretiert.

Organisationseinheiten und *Informationsobjekte* aus der eEPK, denen Prozessschritte zugeordnet sind, können in einer Aktivität durch aktive oder passive Partitionen visualisiert werden. Unter einer passiven Partition werden dabei Notationselemente verstanden, die zwar für eine Aktion benötigt werden, jedoch keinen Einfluss auf das Verhalten dieser Aktion ausüben. Aktive Partitionen hingegen sind unmittelbar an der Aktion beteiligt und gestalten diese mit. So sind vorliegende Dokumente in den meisten Geschäftsaktionen passiv und ein Informationssystem bzw. der Prozessausführende aktiv. Wenn die Bearbeitung durch mehrere direkt oder indirekt beteiligte Objekte darzustellen ist, wird die eindimensionale Sichtweise schnell unübersichtlich. Hier bieten sich dem Modellierer zwei Alternativen an: Innerhalb der betroffenen Aktion ist es möglich, alle beteiligten Partitionen oberhalb des Aktionsnamens untergliedert in aktive und passive Einflüsse anzugeben. Dies kann aber punktuell leicht zur Informationsüberflutung führen. Die andere – unserer Meinung nach übersichtlichere – Alternative ist in den Prozessfluss integrierbar. Aktive Partitionen werden in den Partitionen geführt, passive hingegen als Objekte behandelt. Wie im Beispiel erkenntlich, lassen sich dann anwendungsunterstützende Systeme und mitgeführte Informationsquellen in den spezialisierten Objektknoten modellieren. Dadurch wird der Partitionsraum eingeschränkt und die Übersichtlichkeit bleibt gewahrt.

Der *Start-* bzw. *Endknoten* aus dem Aktivitätsdiagramm wird in der eEPK nicht verwendet. Hier legen die Ereignisse Beginn und Ende einer Prozesskette fest. Wenn eine eEPK durch *mehrere Startereignisse* gekennzeichnet ist, dann muss der Prozessfluss im Aktivitätsdiagramm ebenfalls mehrere Startknoten besitzen. Bei abhängigen Ereignissen wird nach dem Startpunkt eine Entscheidung hinzugefügt und alle von diesem Entscheidungsknoten abgehenden Kanten werden mit Wächterbedingungen versehen. Ereignisse, die

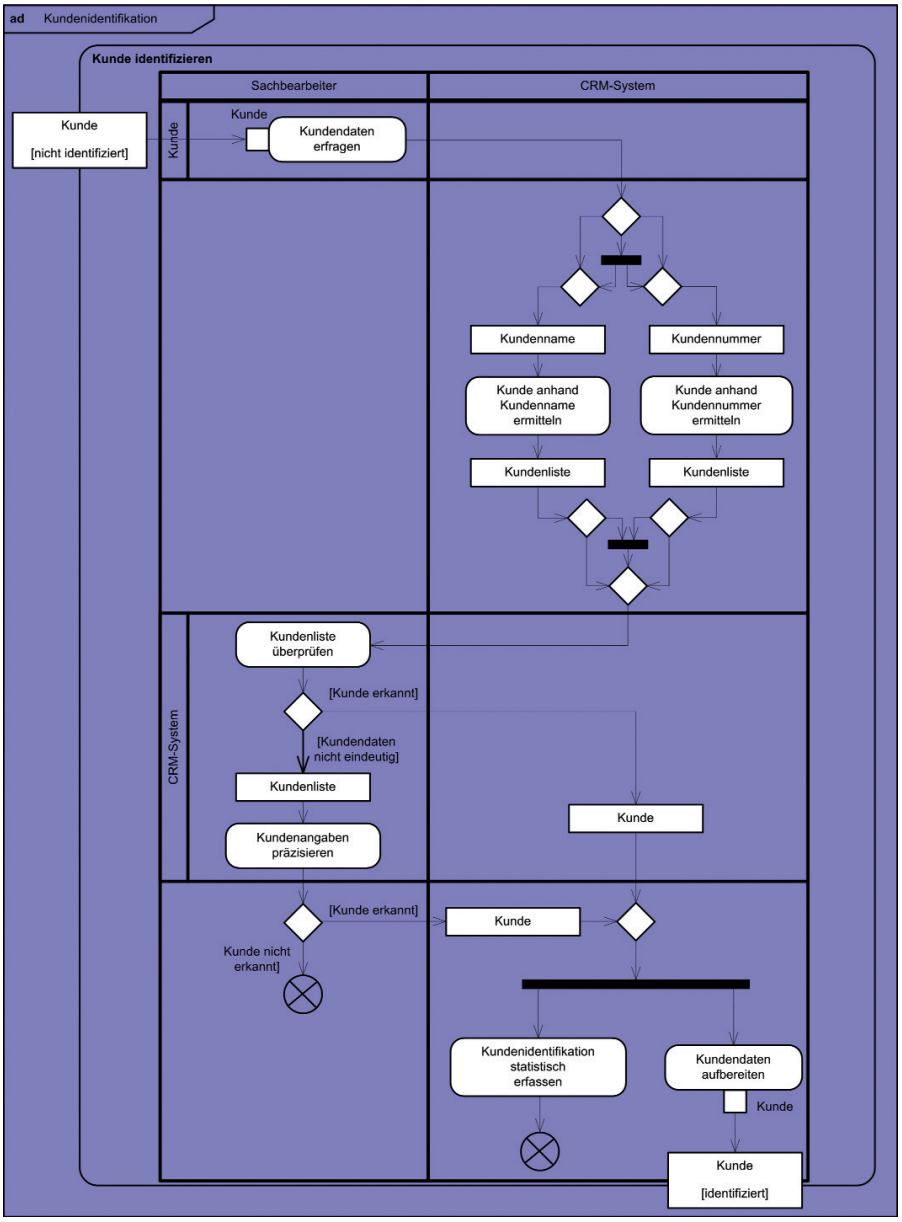


Abb. 3: *Beispielprozess Kunde identifizieren (UML-Aktivitätsdiagramm)*

eEPKs beenden, heißen Endereignisse. Gegenüber dem Aktivitätsdiagramm existiert demnach kein spezielles Symbol in ARIS, das die gesamte Prozesskette beendet. Ein im Aktivitätsdiagramm modellierter Aktivitätsendknoten, der alle *Token* eliminiert und somit alle Aktionen stoppt, kann nur dann modelliert werden, wenn kein weiterer aktiver Zweig existiert. Wenn es nur ein Endereignis in der eEPK gibt oder aber das Ereignis auf jeden Fall als letztes einritt, ist das Beenden der Aktivität zulässig. Ist dies nicht der Fall, ist die Darstellungsweise der Kontrollflussendknoten notwendig.

Tabelle 1 fasst die vorstellten Transformationen zusammen.

Wir haben diese Transformationsregeln auf unseren in Abbildung 2 dargestellten Beispielprozess „Kunde identifizieren" angewandt. Abbildung 3 veranschaulicht das Ergebnis der Transformation.

Vergleicht man den modellierten Prozessablauf mit der eEPK aus Abbildung 2, dann fällt zunächst die etwas technischer wirkende Modellierungsweise ins Auge. Zugleich scheint in der eEPK der Prozessfluss durch die vertikale Darstellung leichter erkennbar zu sein als im Aktivitätsdiagramm. Dennoch folgen alle Notationselemente im Beispielprozess „Kunde identifizieren" unseren Transformationsregeln und sind im Aktivitätsdiagramm wieder aufzufinden. Dass eine Transformation an manchen Stellen nicht immer einfach ist, möchten wir nachfolgend am Beispiel der Ablaufsteuerung näher beleuchten.

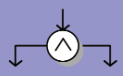
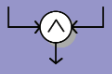
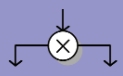
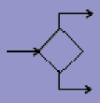
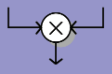
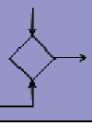
ARIS		UML	
Aufteilendes UND			Parallelisierungsknoten
Zusammenführendes UND			Synchronisationsknoten
Aufteilendes XOR			Entscheidungsknoten
Zusammenführendes XOR			Zusammenführungsknoten

Tabelle 2: Transformation der Ablaufsteuerung von UND und XOR

Knackpunkt Ablaufsteuerung

Die Semantik der Ablaufsteuerung von erweiterten eEPKs in ARIS und der Aktivitätsdiagramme in UML 2.0 ist nicht identisch. Durch die Verwendung expliziter Kontrollstrukturen kann aber eine eindeutige Transformation erreicht werden. Tabelle 2 verdeutlicht die Transformationsregeln für das UND und das XOR.

Die Kontrollflusssteuerung durch das aufteilende UND ist gleichbedeutend mit dem Parallelisierungsknoten. Das zusammenführende UND kann in der UML 2.0 durch den Synchronisationsknoten interpretiert werden. Ein XOR wird als Entscheidungsknoten modelliert. Der Zusammenführungsknoten hingegen fügt die einzelnen Ablaufstränge sequenziell zusammen. Nur wenn sichergestellt werden kann, dass jeweils nur ein Strang aktiv ist, kann die Zusammenführung als XOR interpretiert werden (auf technischer Ebene leistet dies eine so genannte Konkurrenzstelle). Weder UND noch XOR stellen für die Transformation eine große Herausforderung dar, wenn die Interpretation durch die Notationselemente der UML 2.0 wie hier dargestellt erfolgt.

Einer etwas genaueren Betrachtung bedarf das OR. Es wird vor allem dann in der Geschäftsprozessmodellierung genutzt, wenn nicht weiter spezifizierbare Abläufe vorliegen. Dieses Notationselement muss logisch korrekt in die transformierbaren Ausdrucksweisen überführt werden. Die Umformung eines OR in eine logisch äquivalente Aussage erfolgt durch Zusammenfügen der Ausdrucksweisen von UND und XOR. Abbildung 4 verdeutlicht dies für ein aufzuteilendes OR. Die Zusammenführung erfolgt analog.

Die äquivalente logische Umformung eines OR mit weiteren Ausgängen kann sukzessiv in eine Darstellung mit jeweils nur zwei Ausgängen abgebildet werden. Abbildung 5 zeigt eine solche Umformung anhand eines aufteilenden OR mit drei Ausgängen. Für *n* Ausgänge werden somit *(n-1)* OR benötigt. Diese müssen wiederum durch jeweils ein XOR und ein UND ersetzt werden. Das führt zu einer Erhöhung der verwendeten Verknüpfungen auf *2*(n-1)* Elemente. Die Aussage des Modells wird dabei nicht verändert. Wie in der Darstellungsweise des OR sind in der transformierten Abbildung *2n-1* Fälle modelliert. Dadurch wird eine Transformation eindeutig und verlustfrei gewährleistet. Bei komplexen Prozessen wäre die Einführung einer automatisierten Zwischentransformation in ARIS hilfreich und denkbar. Ähnlich einem Semantikcheck könnte dann an dieser Stelle die Übersichtlichkeit auf Ausführungsebene bereits bei der betriebswirtschaftlichen Prozessmodellierung und damit vor der Überführung der relevanten Geschäftsprozesse in die objektorientierte Verhaltensmodellierung erfolgen.

Fazit

Wir haben in diesem Artikel gezeigt, wie Unternehmensprozesse als Basis für die Entwicklung von Softwaresystemen genutzt werden können. Prozessorientierte und objektorientierte Ablaufbeschreibungen werden häufig parallel verwendet. Die Informationsgewinnung für beide Methoden sollte dabei nicht redundant erfolgen. Eine Transformation der beiden Modellwelten kann helfen, Kosten und Zeit zu sparen, insbesondere jedoch einen Informationsverlust zu vermeiden. Durch die Transformation betriebswirtschaftlich orientierte Modelle in technische Verhaltensspezifikationen wurde ein Weg aufgezeigt, wie geschäftsprozess-getriebenes Anforderungsmanagement für eine Softwaresystemgestaltung praxisorientiert etabliert werden kann. Es wurde deutlich, dass bei korrekter Prozessmodellierung eine Transformation verlustfrei möglich ist. In jeder Hinsicht bleibt es spannend, sich mit dem Thema Prozessmodellierung und Prozessmanagement weiterhin zu beschäftigen. ■

Literatur & Links

[Boc03] C. Bock, UML 2 Activity and Action Models, in: Journal of Object Technology, vol. 2 (4), 2003, S. 43-53

[Bor40] M. Born et al, Softwareentwicklung mit der UML 2, Markt und Technik Verlag, 2004

[IDS03] IDS Scheer AG, ARIS 6 Collaborative Suite – Version 6.2.1 – ARIS, Methode, ARIS 6 CD – Handbuch, 2003

[Jec04] M. Jeckle, C. Rupp, J. Hahn, B. Zengler, S. Queins, UML 2 glasklar, Hanser-Verlag, 2004

[OMG04] OMG: UML 2.0 Superstructure Specification, 27.6.04 (siehe: www.omg.org/docs/ptc/04-04-08.pdf)

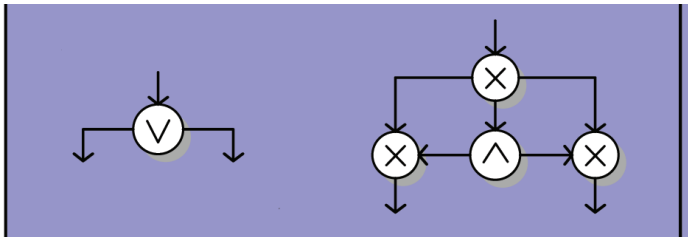


Abb. 4: Aufteilendes „OR" wird zur „XOR-UND"-Darstellung

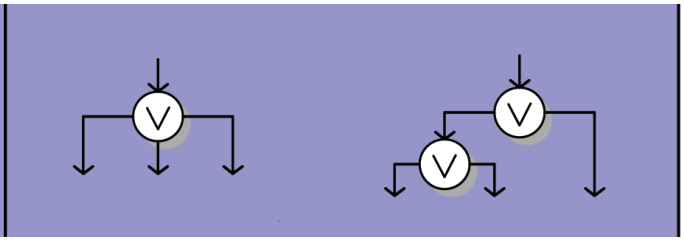


Abb. 5: Aufteilendes „OR" mit drei Ausgängen