Ein Transformationsansatz in der Geschäftsprozessmodellierung

Dipl.-Vw. Veit Köppen
Freie Universität, Berlin

Dipl.-Kfm. Thorsten Schwarz
Gernert und Partner, Berlin

Dr.-Ing. Christiane Gernert
Gernert und Partner, Ludwigsburg

Zusammenfassung

Bei der Projektkonzeption können Geschäftsprozessmodelle für eine klare Abgrenzung fachlicher Anforderungen sorgen. Im Fachbereich wird hauptsächlich prozessorientiert gedacht, während im IT-Bereich heutzutage die objektorientierte Modellierung von Softwaresystemen im Vordergrund steht. Eine Überführung der fachlichen in die technische Sicht bedeutet die verlustfreie Transformation eines prozessorientierten in ein objektorientiertes Modell. Anhand der Prozessmodellierung mit ARIS und UML 2.0 zeigt dieser Beitrag, wie sich Ereignisgesteuerte Prozessketten in Aktivitätsdiagramme ohne Informationsverlust transformieren lassen.

1. Einleitung

Für viele Unternehmen ist die kontinuierliche Optimierung der Geschäftsabläufe eine essentielle Voraussetzung ihres Geschäftserfolges. Diese, oft tiefgreifenden Prozessveränderungen erfordern ein zielgerichtetes und koordiniertes Vorgehen. Erst die Analyse der Geschäftsprozesse und ihre systematische Modellierung schafft das benötigte Wissen für erfolgreiche Optimierungen.

Viele Unternehmen haben in den letzten Jahren in dieser Hinsicht einen grundlegenden Wandel vollzogen. Es werden nicht mehr die Unternehmensprozesse entlang den eingesetzten Softwaresystemen gestaltet, sondern die Softwaresysteme richten sich auf die Bedürfnissen der Prozesse aus. Dass die fachlichen Anforderungen für die Entwicklung solch angepasster Softwaresysteme bereits in den Geschäftsprozessmodellen stecken, ist leicht nachvollziehbar. Die Geschäftsprozessmodelle sollten deshalb den Softwareentwicklern als Grundlage für die resultierende Systemgestaltung dienen (vgl. Abb. 1).

In der Praxis stehen beide Seiten jedoch vor der Herausforderung, oft zunächst eine nicht unerhebliche Sprachbarriere überwinden zu müssen. Während Fachexperte und Organisator sich in der Welt der Prozesse bewegen, lebt der Softwareentwickler in seiner Welt der Objekte. Beide benutzen unterschiedliche Denkansätze und Notationen um das Verhalten von Systemen zu modellieren. Die Minimierung des Informationsverlusts zwischen diesen beiden Begriffs- bzw. Sprachwelten bildet einen Ansatzpunkt Softwareprojekte erfolgreicher zu gestalten.

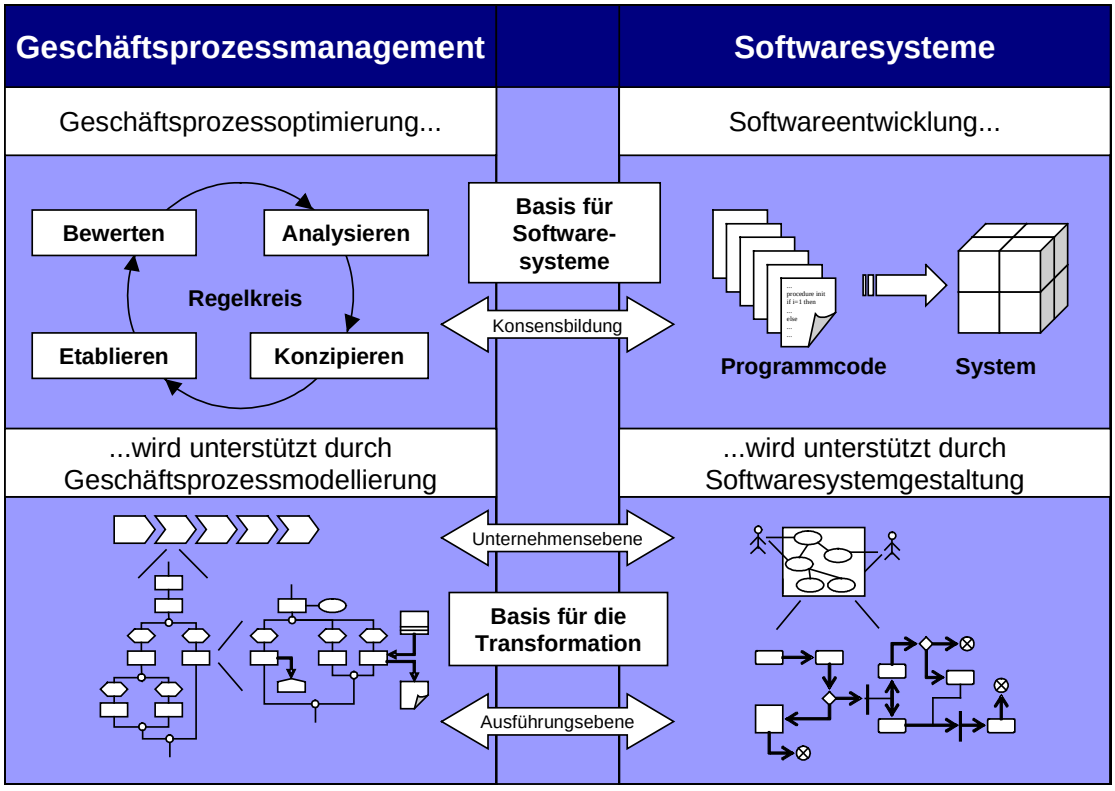


Abbildung 1: Geschäftsprozessmanagement und Softwaresysteme

2. Prozessmodellierung

Die Prozessmodellierung unterstützt die systematische und strukturierte Dokumentation von Prozessinformationen. Für die Modellierung von Prozessen existieren unterschiedliche methodische Ansätze und Darstellungsformen. Informelle (verbale) Darstellungen haben den Nachteil, dass sie nicht immer eindeutig sind, sondern selbst bei vorgegebenen Beschreibungsmustern einen breiten Spielraum für Interpretationen bieten. Eine weitere Herausforderung besteht in der verbalen Beschreibung von Prozessverzweigungen und Nebenläufigkeiten in einem Prozess. Grafische Darstellungen besitzen in diesem Bereich ihre Stärken. Einfach strukturierte grafische Elemente 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. Streng formale Beschreibungen (grafisch oder textuell) erfordern einen hohen Grad an Abstraktion. Die prozessorientierte und objektorientierte Modellierung sind die beiden bekanntesten methodischen Ansätze.

Die Tiefe der Modellierung und die Auswahl von Vorgehen, Methode und Werkzeugunterstützung werden maßgeblich von den Ausführenden und Beteiligten der Prozessmodellierung bestimmt. Sie hängen jedoch auch vom jeweiligen Modellierungsziel ab. Ist es zum Beispiel erklärtes Ziel im Nachgang der Konzeption die Prozesse durch eine selbstentwickelte Software zu unterstützen, setzt man in der Regel die im Kontext der objektorientierten Anwendungsentwicklung etablierte Beschreibungssprache Unified Modelling Language (UML) ein. Stehen Organisationsveränderungen im Vordergrund, wird man eher zu den Ereignisgesteuerten Prozessketten (EPK) greifen.

2.1 Die fachliche Sicht: Prozessmodellierung mit ARIS

Ausgangspunkt von geschäftsprozessgetriebenem Anforderungsmanagement für die Softwareentwicklung ist ein fundiertes Prozesswissen im Unternehmen [3].

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

Die Modellierung von Geschäftsprozessen mit dem ARIS-Toolkit der IDS Scheer AG ist heutzutage weit verbreitet [5]. Zur Beschreibung von Geschäftsprozessen wird innerhalb der ARIS-Architektur die Ereignisgesteuerte Prozesskette (EPK) verwendet.

Eine EPK besteht aus einem Graph mit einer alternierenden Abfolge von Ereignissen und Funktionen. Funktionen dienen der Modellierung eines betrieblichen Vorgangs. Ereignisse beschreiben den Zustand eines Informationsobjekts zu einem bestimmten Zeitpunkt. Verknüpfungsoperatoren steuern den Prozessfluss und ermöglichen die Darstellung optionaler und paralleler Abläufe. Abbildung 2 zeigt ein Beispiel für eine EPK.

Zusätzlich zum Ablauf des Geschäftsprozesses können in die EPK weitere Elemente integriert werden. So können zum Beispiel die ausführenden Organisationseinheiten, der Datenfluss zwischen den Funktionen oder die produzierten Leistungen dem Geschäftsprozessmodell hinzugefügt werden. Werden diese zusätzlichen Informationen modelliert, spricht man von erweiterten Ereignisgesteuerten Prozessketten (eEPK).

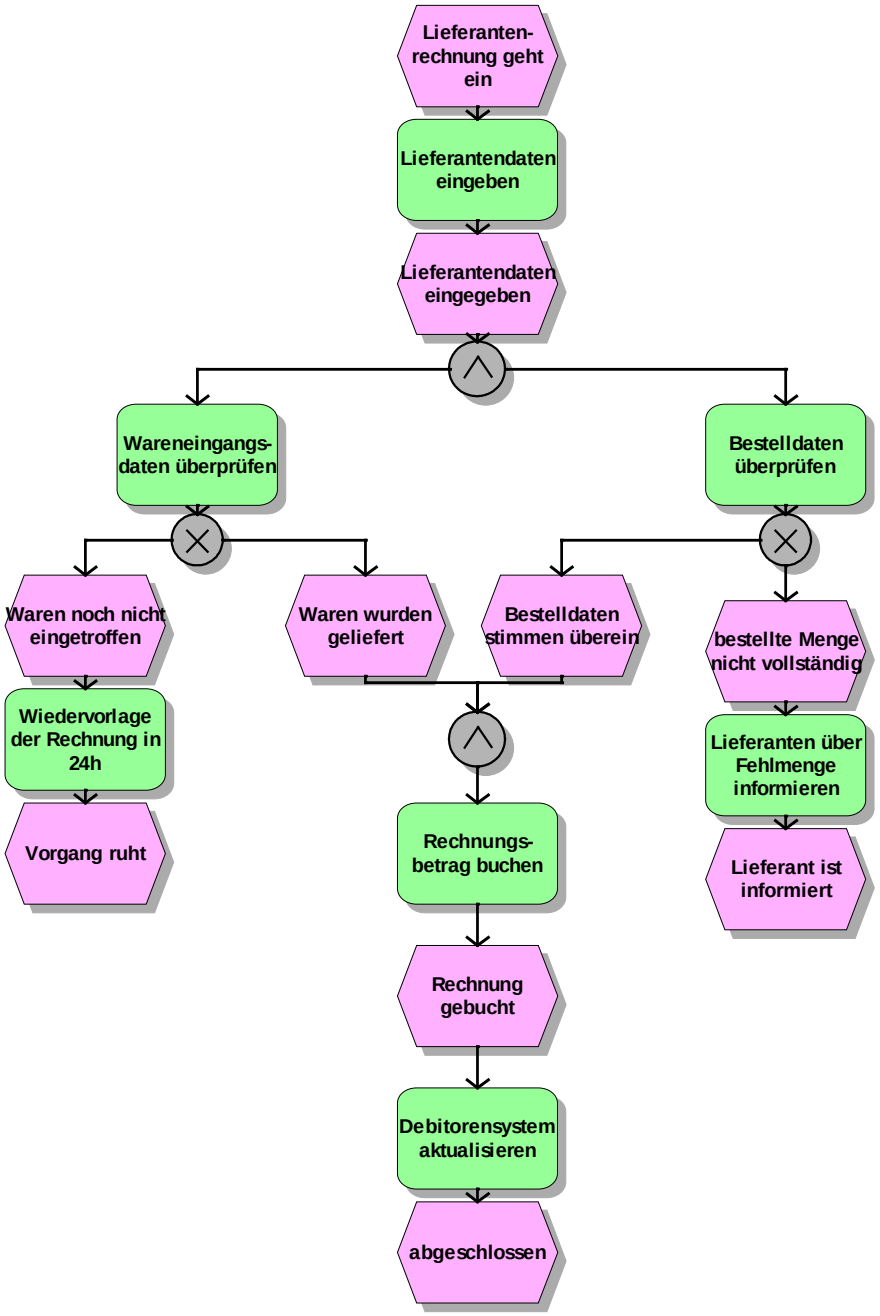


Abbildung 2: Beispiel Bearbeitung einer Lieferantenrechnung

2.2 Die technische Sicht: Prozessmodellierung mit UML 2.0

Die UML wurde in der Version 2.0 grundlegend überarbeitet [4,6]. Die in der Vergangenheit entstandene Komplexität einiger Diagramme wurde verringert und die Semantik vieler Notationselemente präzisiert. Ein Großteil der Änderungen fand dabei im Bereich der Verhaltensmodellierung statt. Insbesondere das für die Prozessmodellierung interessante Aktivitätsdiagramm ist nicht mehr eine Spezialform des Zustandsdiagramms, sondern besitzt nun ein eigenes Metamodell. Aus diesem Grund sind auch die viele Notationselemente des Aktivitätsdiagramms neu.

Ein Aktivitätsdiagramm der UML 2.0 beschreibt den Ablauf einer Aktivität anhand eines Graphen von Aktionen [1]. Eine Aktion ist eine elementare Einheit einer Verhaltensspezifikationen. Die einzelnen Aktionen ähneln dabei stark den Notationselementen moderner Programmiersprachen. Es existieren beispielsweise Aktionen zum Lesen / Schreiben von Attributen eines Objekts (ReadStructuralFeatureAction / WriteStructuralFeatureAction) und zum Erzeugen / Vernichten von Objekten (CreateObjectAction / DestroyObjectAction).

Durch die Verwendung der CallBehaviorAction können andere Verhaltensspezifikationen aufgerufen werden. Der Aufruf einer Aktivität aus einer anderen Aktivität heraus kann dafür genutzt werden, um eine einmal definierte Verhaltensspezifikationen mehrmals zu verwenden oder das Verhalten eines Systems auf verschiedenen Abstraktionsschichten zu modellieren.

Der Ablauf eines Aktivitätsdiagramms ist durch ein Tokenkonzept definiert. Das Tokenkonzept ähnelt dem der Petri-Netze. Stehen an allen eingehenden Kanten einer Aktion Token zur Verfügung, dann wird diese ausgeführt. Nach Beendigung der Aktion werden allen ausgehenden Kanten Token zur Verfügung gestellt. Diese Eigenschaft der Aktion kann dazu verwendet werden, um mehrere nachfolgende Aktionen (Teilprozesse) parallel zu starten. Ein paralleler Ablauf kann durch die Verwendung eines Parallelisierungsknotens aber auch explizit dargestellt werden. Die Synchronisation mehrerer Teilprozesse erfolgt mit dem Synchronisationsknoten. Sowohl Parallelisierungs- als auch Synchronisationsknoten werden in Form eines schwarzen Balkens dargestellt. Zur Darstellung optionaler Abläufe muss ein Entscheidungsknoten verwendet werden. Die Zusammenführung mehrerer optionaler Teilprozesse erfolgt durch Einsatz eines Zusammenführungsknotens. Beide Knoten werden wiederum durch dasselbe Symbol, eine Raute, dargestellt.

Ein Token wird zu Beginn der Ausführung einer Aktivität durch einen Startknoten (InitialNode) erzeugt. Analog dazu wird ein Token vernichtet, wenn er einen Endknoten des Kontrollflusses (FlowFinalNode) erreicht. Darüber hinaus existiert noch der Endknoten der Aktivität (ActivityFinalNode), der alle existierenden Token innerhalb der Aktivität vernichtet und somit die Ausführung aller Aktionen beendet.

Zusätzlich zum reinen Ablauf einer Aktivität kann der Objektfluss beschrieben werden. Die Modellierung des Objektflusses erfolgt durch die zusätzliche Verwendung von Objektknoten, die zur Darstellung der eingehenden und ausgehenden Objekte einer Aktion dienen. Diese Objektknoten werden als Eingabe- bzw. als Ausgabepins

bezeichnet. Die Kanten zwischen Objektknoten transportieren nun reale Objekte (Objekttoken) und nicht mehr die imaginären Marken (Kontrolltoken). Da in diesem Beitrag nur die Transformation von EPK und Aktivitätsdiagramm dargestellt werden soll, wird an dieser Stelle nicht weiter auf die Modellierung des Objektflusses in UML 2.0 eingegangen.

3. Transformation beider Welten

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[4] und auftretendem Verhalten[5] [2]. 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 (Kommunikation) der Objekte im System. Die klassische Form der Modellierung der Interaktion von Objekten ist das Sequenzdiagramm. Auf diese Weise kann das Verhalten objektübergreifend beschrieben werden.

Mit dem Aktivitätsdiagramm der UML 2.0 kann sowohl Ausführungsverhalten als auch auftretendes Verhalten beschrieben werden. Darüber hinaus wird das Verhalten innerhalb eines Aktivitätsdiagramms ablauforientiert beschrieben. Die beteiligen Objekte spielen eine untergeordnete Rolle. Aus diesem Grund eignet sich das Aktivitätsdiagramm am besten zur Beschreibung von Geschäftsprozessen.

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.

Nun ist eine Transformation einer Funktion in ARIS zu einer CallBehaviorAction in einem Aktivitätsdiagramm nicht ohne weiteres möglich. Eine CallBehaviorAction stellt nur den Aufruf einer Verhaltensspezifikation, nicht die Verhaltensspezifikation selbst dar. Die Lösung dieses Problems besteht darin, dass eine Funktion in eine Aktivität transformiert wird. Diese wird dann von der CallBehaviorAction aufgerufen. Allgemein ist eine Aktivität als eine zeitliche Operation definiert, die zwischen einem Anfangs- und Endereignis liegt und den Zustand eines Objektes verändern kann. In

[4] engl.: Executing Behavior
[5] engl.: Emergent Behavior

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 Abfolge von CallBehaviorActions umgesetzt (s. Abbildung 3).

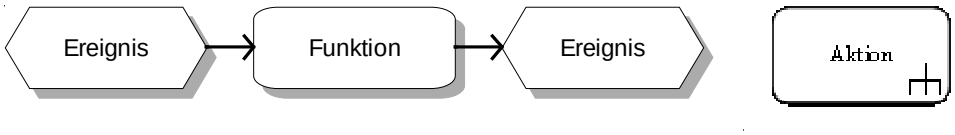


Abbildung 3: Transformation von Verhaltensnotationen

Eine implizite Darstellung von optionalen und parallelen Abläufen führt in der Praxis häufig zu Missverständnissen, da die Semantik des Tokenkonzeptes zwischen Kontroll- und Objektfluss entgegengesetzt definiert ist. Weiterhin wird in der EPK eine explizite Darstellung der Ablaufsteuerung vorgenommen. Aus diesem Grund muss auch in der UML-Modellierung für eine Transformation die Verwendung der expliziten Modellierung von optionaler und paralleler Ablaufsteuerung gefordert werden. In der UML existieren nur Kontrollknoten für das logische UND und für das Exklusive ODER. In der EPK sollten deshalb auch nur diese beiden Steuerungselemente verwendet werden. Durch Kombination dieser beiden Operatoren können jedoch auch alle anderen logischen Operatoren, die in der EPK zur Modellierung eingesetzt werden, ausgedrückt werden.

In ARIS wird eine EPK durch Startereignisse angestoßen. Durch den Aufruf einer Aktivität produzieren die Startknoten Kontrolltoken. Dadurch wird die Aktivität in der UML gestartet. Zur Transformation müssen somit alle Startereignisse in Startknoten transformiert werden. Endereignisse sind das Ende eines Stranges des Geschäftsprozesses. Im Aktivitätsdiagramm wird für die Beendigung eines Kontrollflusses der Kontrollflussendknoten genutzt. Auf eine Verwendung des Aktivitätsendknotens sollte verzichtet werden.

Die Transformation der Verknüpfungsoperatoren der EPK und der Kontrollknoten des Aktivitätsdiagramms ist in Abbildung 4 dargestellt.

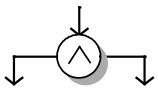
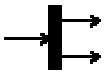
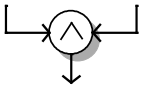
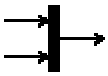
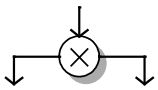
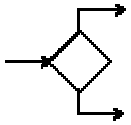
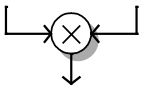
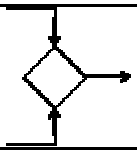
ARIS		UML	
Aufteilendes UND			Parallelisierungsknoten
Zusammenführendes UND			Synchronisationsknoten
Aufteilendes XOR			Entscheidungsknoten
Zusammenführendes XOR			Zusammenführungsknoten

Abbildung 4: Transformation der Verknüpfungsoperatoren und Kontrollknoten

Durch eine Transformation können die Informationen verlustfrei zwischen den Modellen ausgetauscht werden. Die folgenden Regeln für eine Transformation müssen jedoch beachtet werden:

1. Verwendung von Call Behavior Actions,
2. explizite Modellierung der Kontrollknoten,
3. Verwendung der logischen UND- und XOR-Verknüpfung in ARIS.

Das Ergebnis der Transformation der Abbildung 4 dargestellten EPK in ein UML-Aktivitätsdiagramm zeigt nachfolgende Abbildung .

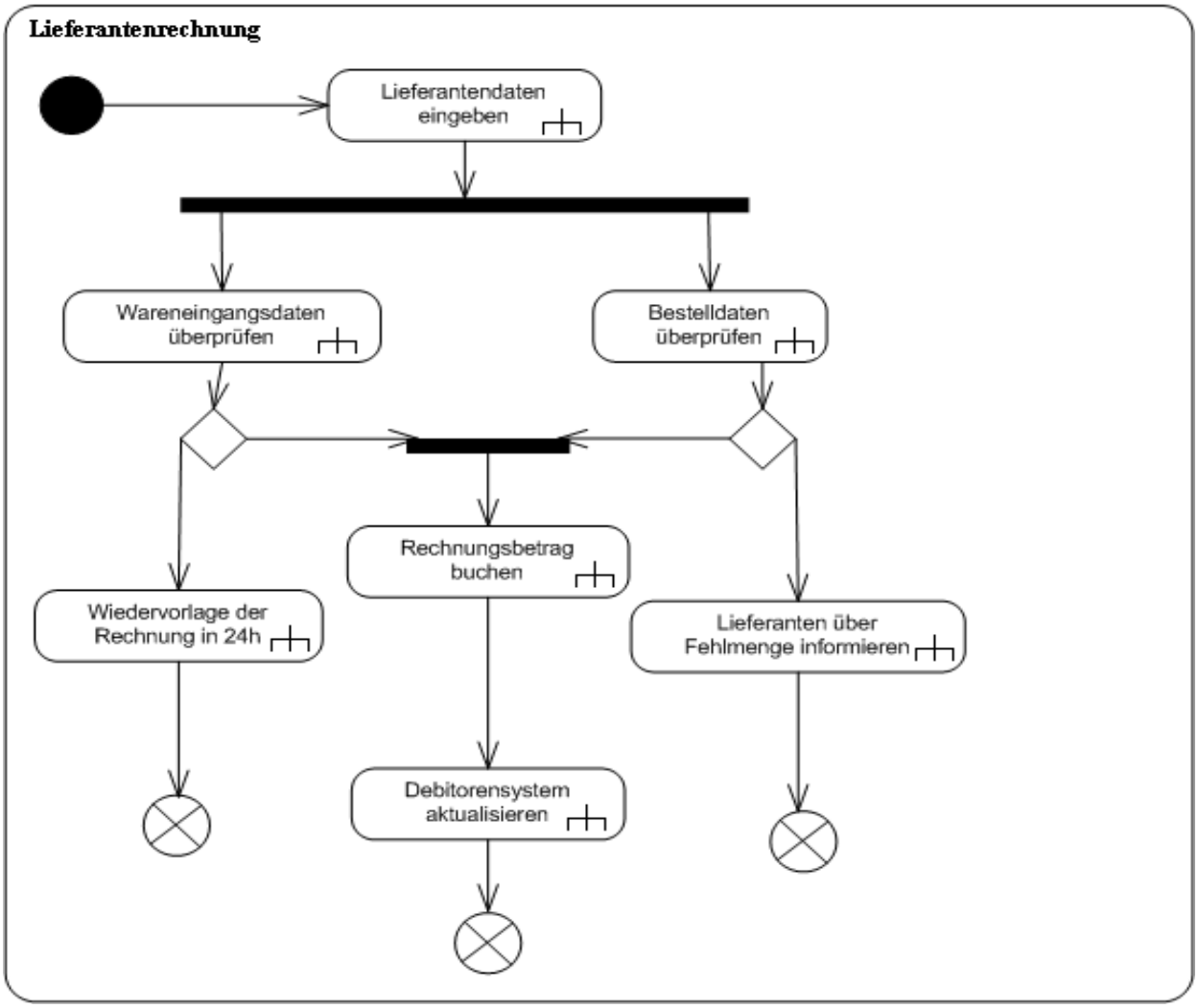


Abbildung 5: Aktivitätsdiagramm der Lieferantenrechnung

4. Ausblick

Prozessorientierte und Objektorientierte Ablaufbeschreibungen werden häufig parallel verwendet. Die Informationsgewinnung für beide Methoden wird dabei meist redundant durchgeführt. Eine Transformation der beiden Modellwelten kann helfen, Kosten und Zeit zu sparen, insbesondere jedoch einen Informationsverlust zu vermeiden. Transformationsregeln sind dafür unabdingbar.

Die Anwendung der UML als Modellsprache eröffnet die Möglichkeit, innerhalb eines IT-Projektes von der Geschäftsidee bis zur Realisierung eine einheitliche Notation zu nutzen. Die weiterentwickelte UML 2.0 bietet zusätzliche leistungsfähige Ausdrucksmittel für das methodische Identifizieren und Definieren der Geschäftsprozesse.

Literatur

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

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

[3] Hammer, M, Champy, J.: Reengineering the corporation: a manifesto for Business revolution, Brealey, Landon, 2001.

[4] Jeckle, M. et al.: UML glasklar, Hanser Verlag, München, 2004.

[5] Scheer, A.-W.: ARIS – Vom Geschäftsprozeß zum Anwendungssystem, 3. Auflage, Springer Verlag, Berlin, 1998.

[6] UML 2.0 Superstructure Specification, Object Management Group, www.omg.org/uml, 2004.