

Modellierung von Softwareentwicklungsprozessen mit Hilfe von Aktivitätsdiagrammen

Stefan Hilmer

Resco GmbH
- Business Consulting -
Millerntorplatz 1
D-20359 Hamburg
Stefan.Hilmer@Resco.de

Zusammenfassung: In den vergangenen Jahren traten immer neue allgemeine vereinheitlichte Softwareentwicklungsprozesse in Erscheinung. Zudem setzen viele Industrieunternehmen heute auf Eigenentwicklungen in diesem Bereich. Im praktischen Einsatz wird man dabei mit immer neuen Notationen zur Prozessmodellierung und einem entsprechend hohen Einarbeitungsaufwand konfrontiert. Dieser Beitrag zeigt eine Möglichkeit zur Vereinheitlichung der Prozessmodellierung mit Hilfe von Aktivitätsdiagrammen. Diese bringen drei wesentliche Vorzüge mit sich: 1. Aktivitätsdiagramme sind als Bestandteil der Unified Modeling Language (UML) weit verbreitet und in der Regel dem überwiegenden Teil der Prozessbeteiligten bekannt, 2. bieten sie die Möglichkeit einer durchgängigen, prozessunabhängigen Modellierung und 3. lassen sich in solchen Darstellungen einfach und nachvollziehbar Anpassungen des Prozesses vornehmen.

1 Einleitung und Motivation

Wie führt man Softwareentwicklungsprojekte zum Erfolg? Dieser Frage widmen sich seit je her Vorgehensmodelle und Prozesse der Softwareentwicklung. Nicht zuletzt durch die Einführung objektorientierter Technologien werden Softwaresysteme und damit auch die Anforderungen an deren Entwicklung immer komplexer. Aus diesem Grund werden immer neue allgemeine vereinheitlichte Vorgehensmodelle und Prozesse vorgestellt, die häufig die objektorientierte Softwareentwicklung in ihren Mittelpunkt stellen. Zu diesen Prozessen gehören unter anderen der *Rational Unified Process (RUP)*, die *Process Patterns* und - im weitesten Sinne - auch das *eXtreme Programming (XP)*. Parallel zu diesen teilweise kommerziell angebotenen Produkten entwickeln immer mehr Unternehmen eigene unternehmensweit vereinheitlichte Prozesse, die an unternehmenseigene Rahmenbedingungen und Möglichkeiten angepasst sind.

Zur Modellierung sowohl allgemeiner als auch unternehmenseigener vereinheitlichter Prozesse werden in der Regel selbstentwickelte Notationen eingesetzt, wobei hier häufig eine angemessene Durchgängigkeit zu vermissen ist. Vor dem Hintergrund der Existenz einer standardisierten Modellierungssprache, wie der Unified Modeling Language (UML), ist dieser Zustand unerklärlich, zumal diese Sprache im Rahmen nahezu aller modernen Softwareentwicklungsprozesse Anwendung findet.

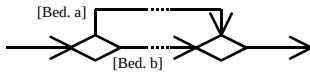
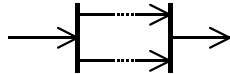
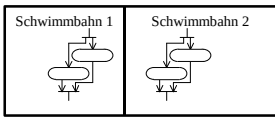
Die Vorzüge der UML sind also in der Regel den Designern von Softwareentwicklungsprozessen hinlänglich bekannt. Vereinheitlichung, Eindeutigkeit und Durchgängigkeit der Notationen gehören zu ihren Wesensmerkmalen und sind gewiss drei der Gründe für ihre weite Verbreitung. Vereinheitlichung ist aber auch erklärtes Ziel der geschilderten Prozesseinführungen. Dieser Grundgedanke ist also den Prozessen und der UML gemein und würde durch die Verwendung der UML zur Prozessbeschreibung stark unterstützt. In der Praxis - und insbesondere im Berateralltag - zeigt sich immer wieder, dass mit dem Vorhandensein einheitlicher Notationen ein nicht unerheblicher Teil der Einarbeitung entfällt. Dies gilt im gleichen Maße für die Darstellung eines zu verwendenden Prozesses. Außerdem kann über eine vereinheitlichte Darstellung ein besseres Verständnis aller Prozessbeteiligten, sowohl der Prozessdesigner als auch der -anwender, erreicht werden.

Mit den Aktivitätsdiagrammen bietet die UML eine Darstellungsform, die sich zur Prozessbeschreibung im praktischen Einsatz bewährt hat. Ein konkretes Vorgehen wird im nachfolgenden Kapitel 2 beschrieben und im dritten Kapitel beispielhaft auf die Darstellungsformen der erwähnten allgemeinen vereinheitlichten Prozesse übertragen. Die UML-Darstellung unterstützt zudem den Anwender bei der Anpassung von Prozessen, denn in diesen Notationen können, wie im vierten Kapitel beschrieben, Vorschriften zur Änderung formalisiert werden. Außerdem werden die Änderungen leichter nachvollziehbar.

2 Umsetzung von Prozessbeschreibungen in Aktivitätsdiagramme

Zum Aktivitätsdiagramm heißt es in [Ba01]: „Es ist besonders gut dafür geeignet, komplexe Workflows übersichtlich darzustellen.“ Diesem Grundsatz folgt die Idee, Prozessbeschreibungen in UML-Aktivitätsdiagramme zu überführen. Diese Idee führt zu dem hier beschriebenen Ansatz. Die dabei eingesetzten Notationen stammen aus der UML-Spezifikation [OMG00]. Die Konzepte der Aktivitätsdiagramme sind demnach: *Aktivitäten*, *übergeordnete Aktivitäten*, *Transitionen*, *Entscheidungen*, *Synchronisationen*, *Schwimmbahnen*, *Objekte* und *Steuerungssignale* sowie aus den Zustandsdiagrammen entnommene *Anfangs-* und *Endpunkte*. Die Anwendung dieser Konzepte wird im Folgenden beschrieben, die zugehörigen grafischen Notationen der UML sind in Tab. 1 zusammengestellt.

Aktivitäten repräsentieren Vorgänge, die in einem beliebigen System, also auch im Rahmen eines Softwareentwicklungsprozesses, ablaufen. *Übergeordnete Aktivitäten* fassen andere Aktivitäten zusammen. Innerhalb eines Softwareentwicklungsprozesses können übergeordnete Aktivitäten zur Repräsentation von Teilprozessen oder Phasen eingesetzt werden. Die zusammengefassten Aktivitäten werden in der Regel in verschachtelten Aktivitätsdiagrammen dargestellt. *Transitionen* zwischen den Aktivitäten werden automatisch aktiviert, sobald die davor liegende Aktivität ausgeführt wurde. Jede Aktivität besitzt eine Eingangs- und eine Ausgangstransition. Zusätzliche Ausgangstransitionen sind möglich, sofern diese durch Bedingungen unterschieden werden.

Bezeichnung im Aktivitätsdiagramm	UML-Notation	Bezeichnung im Prozessmodell
Aktivität		Aktivität
Übergeordnete Aktivität		Teilprozess oder Phase
Transition		Übergang
Entscheidung		alternative Aktivitäten, Phasen oder Teilprozesse
Synchronisation		parallele Aktivitäten, Phasen oder Teilprozesse
Schwimmbahn		Rolle
Objekte		Datenaustausch, Zuständigkeiten
Steuerungssignal		Außenwirkungen
Anfangspunkt		Prozess- oder Phasenbeginn
Endpunkt		Prozess- oder Phasenende

Tab. 1 - UML-Notationen und ihre Funktion in Aktivitätsdiagrammen und Prozessmodellen

Alternativ zu mehrfachen Transitionen am Ausgang einer Aktivität werden auch spezielle sogenannte *Entscheidungen* eingesetzt. Auch ihre Ausgangstransitionen können durch Bedingungen unterschieden werden. Die Funktionsweise entspricht also den Aktivitäten mit mehreren Ausgängen, nur dass hier keine Vorgänge ausgeführt werden. Diese Entscheidungen werden genauso durch Rauten symbolisiert wie die Zusammenführungen nach derartigen Verzweigungen. Dabei sind dann allerdings mehrere Ein- und eine Ausgangstransition vorzusehen. Zur Aktivierung des Ausgangs genügt die Aktivierung eines der Eingänge. Gelegentlich wird eine Zusammenführung auch vereinfacht dargestellt, indem direkt mehrere Transitionen auf eine Aktivität zeigen. Die UML-Spezifikation [OMG00] sieht diese Darstellung jedoch nicht explizit vor. Da zudem das Verhalten identisch mit erstgenannter Notation ist und somit die Darstellungen austauschbar sind, wird letztere im Folgenden ohne Beschränkung der Allgemeinheit nicht berücksichtigt. *Synchronisationen* können mehrere Eingangs- und mehrere Ausgangstransitionen aufweisen. Sobald alle eingehenden Transitionen aktiviert wurden, werden auch alle Ausgangstransitionen aktiviert. Auf diesem Wege lassen sich parallele Teilprozesse und Aktivitäten darstellen. Synchronisationen und Entscheidungen können auch verschachtelt auftreten. *Schwimmbahnen* teilen Aktivitätsdiagramme in Bereiche, die im

Fälle einer Prozessdarstellung einzelnen Rollen zugeordnet werden können. *Objekte* können sowohl Zuständigkeiten als auch Objektflüsse beschreiben. Zuständigkeiten beziehen sich in der Regel auf Rollen, also auf am Prozess beteiligte, und werden diesen grafisch zugeordnet. Objektflüsse zeigen die Übergabe von Objekten, in einem Softwareentwicklungsprozess in der Regel von Dokumenten, Modellen, o.ä., in die oder aus den einzelnen Aktivitäten. Wird ein Objekt von einer Aktivität zur anderen übergeben, ersetzt dies eine Transition. Den Objekten, also auch den Dokumenten kann dabei ein Status zugeordnet werden. *Steuerungssignale* gehören zu einer Transition. In der Darstellung wird eine Transition zweigeteilt, eine führt in das Signalsymbol hinein und eine hinaus. Es werden zwei Signale unterschieden, eingehende und ausgehende Signale. Beide sind mit Objektflüssen kombinierbar. So lassen sich in der Prozessdarstellung Außenwirkungen darstellen. Jede in einem Aktivitätsdiagramm dargestellte Kette von Aktivitäten beginnt mit einem *Anfangspunkt*, von dem eine Transition abgeht, und hört an einem *Endpunkt* auf, zu dem eine Transition führt. Beginn und Ende eines Prozesses lassen sich so darstellen.

3 Übertragbarkeit der Darstellung auf allgemeine Softwareentwicklungsprozesse

Ob sich alle Softwareentwicklungsprozesse durch Aktivitätsdiagramme modellieren lassen, kann nur schwer bewiesen werden. Im Gegenteil werden mit Sicherheit irgendwo Prozessdarstellungen zu finden sein, die Notationen verwenden, die sich, wenn überhaupt, nur mit größter Mühe in UML-konforme Darstellungen überführen lassen. In der Praxis treten solche Besonderheiten allerdings nur höchst selten auf. Hingegen lässt sich anhand der in der Einleitung benannten allgemeinen vereinheitlichten Prozesse exemplarisch zeigen, dass übliche Konzepte einer Prozessmodellierung durchaus in den UML-Notationen der Aktivitätsdiagramme darstellbar sind. Mit der hier getroffenen Auswahl an Prozessen wurden bewusst verschiedene Prozessformen einbezogen. Der RUP gehört zu den Prozessen, die einen starren Workflow vorgeben. Das XP hingegen zählt zu den agilen Prozessen, deren Workflow stark projektabhängig sein kann und Process Patterns liegen in dem Grenzbereich dazwischen.

3.1 eXtreme Programming (XP)

Nach [Be 00] ist XP kein Prozess, sondern eine Disziplin oder vielmehr eine „vergnügli- che Art und Weise der Softwareentwicklung.“ Dennoch wird XP in der gleichen Quelle gegen andere Prozesse abgegrenzt und soll daher auch hier als solcher eingeordnet werden. Häufig wird das XP inzwischen auch sehr treffend als agiler Prozess bezeichnet. Auf eine starre Prozessbeschreibung wird vollends verzichtet. Es werden zunächst vier Aktivitäten verbal beschrieben, auf denen die extreme Softwareentwicklung beruht, zu- dem werden einige Ergebnistypen zu diesen Aktivitäten benannt. Das Planen dieser Aktivitäten (vergl. [BF01]) ist Bestandteil des Prozesses selbst. Diese Planung bildet eine der Phasen des Lebenszyklus eines idealen XP-Projekts. Wer dabei welche Aufga- ben übernimmt wird ebenfalls nicht eindeutig beschrieben, es wird jedoch eine Reihe

von grundlegenden Rollen beschrieben. All diese Dinge lassen sich, wie Tab. 1 zeigt, in UML-Notationen ausdrücken.

Konzepte des XP	Ausprägungen	Anwendbare UML-Notation
Aktivitäten	Programmieren, Testen, Zuhören und Design entwerfen	
Ergebnisse	Code, Testfälle und Tests, Anforderungen und Design	
Phasen	Erforschung, Planung, Iteration bis zur ersten Version, Produktion, Wartung und Tod	
Rollen	Programmierer, Kunde, Tester, Manager, Terminmanager, Coach, Berater und Big Boss	

Tab. 2 - Konzepte des eXtreme Programming und ihre Darstellung in Aktivitätsdiagrammen

Die Verbindungen zwischen den Konzepten bzw. ihren Ausprägungen festzulegen ist Inhalt der Planungsphasen und kann somit hier nicht fest angegeben werden. Mit Transitionen, Entscheidungen oder Synchronisationen (vergl. Tab. 1) werden hier mit der UML jedoch einige Konzepte angeboten.

3.2 Process Patterns

Mit den Process Patterns wird in [Am99a] und [Am99b] der Objektorientierte Software Prozess (OOSP), wie in Abb. 1 gezeigt, eingeführt.

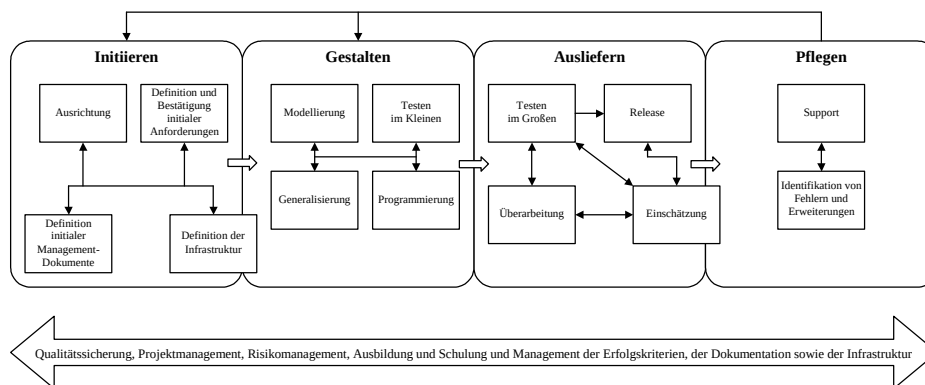


Abb. 1 - Darstellung des OOSP nach [Am99a]

Im Kern dieses Prozesses steht der Grundsatz, seriell im Großen, iterativ im Kleinen. Dies bedeutet, dass die wesentlichen vier Projekt-Phasen, Initiieren, Gestalten, Ausliefern und Pflegen seriell durchlaufen werden. Parallel dazu sind verschiedene Managementaufgaben, wie Qualitätssicherung, Projektmanagement, Risikomanagement, Risikomanagement, Reuse-Management usw., auszuführen. Diese Zusammenhänge sind in dem Aktivitätsdiagramm in Abb. 2 in der Notation der UML dargestellt.

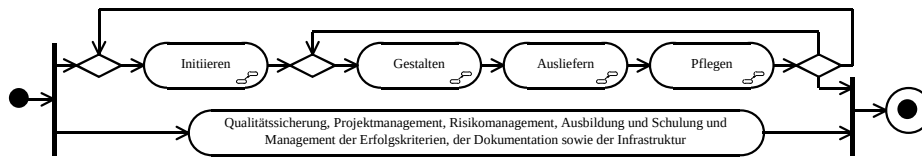


Abb. 2 - Darstellung des OOSP als Aktivitätsdiagramm

Die vier primären Phasen des OOSP sollen hier inhaltlich nicht weiter betrachtet werden. Wichtig ist hier lediglich, dass jeder dieser Phasen eines der vier Aktivitätsdiagramme in Abb. 3 zugeordnet werden kann. Diese vier Diagramme weisen eine hohe Zahl von Entscheidungen auf, die nicht nur im Verlauf des Projektes zu treffen sind, sondern deren Bedingungen in der Regel auch projektabhängig definiert werden. Hierin liegt die Flexibilität des Prozesses und die Bedeutung der Bezeichnung Process Patterns, Prozessmuster.

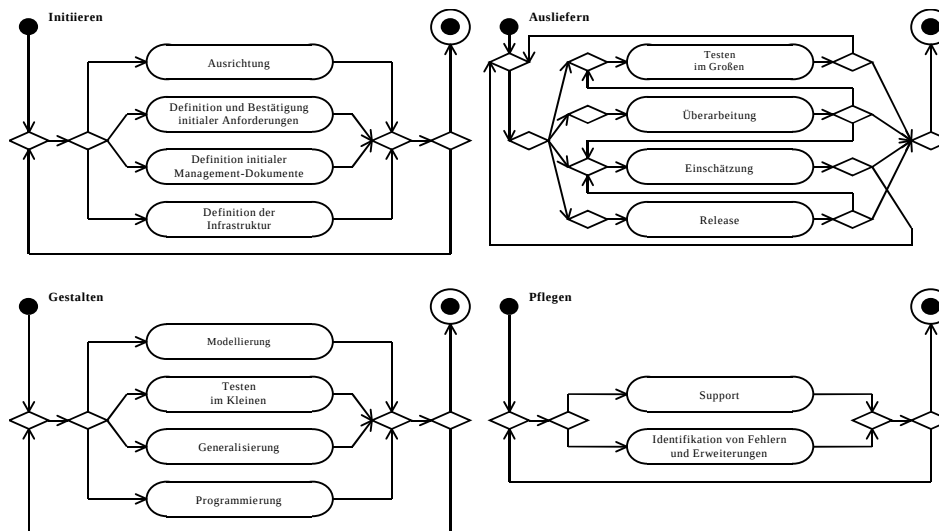


Abb. 3 - Darstellung der Phasen des OOSP in Form von Aktivitätsdiagrammen

Deutlich wird an dieser Stelle aber auch, dass zur Darstellung des in Abb. 1 gezeigten Zusammenhangs insgesamt fünf Aktivitätsdiagramme benötigt werden.

3.3 Rational Unified Process (RUP)

Vom Rational Unified Process (RUP) erwartet man am ehesten eine Prozessbeschreibung in UML-Notationen. Diese Erwartung wird jedoch enttäuscht. Für diesen Prozess wird eine eigene Notation definiert.

In [Kr99] wird der RUP als Prozess mit einer zweidimensionalen Struktur dargestellt. Dabei wird die Organisation bezüglich des Inhalts und die Organisation bezüglich der Zeit unterschieden. Da letzter zwei Ebenen aufweist, Phasen und Iteration, und beide Organisationen tabellarisch gegenübergestellt werden, führt dies zu einer dreistufig verschachtelten Struktur, 1. werden zeitlich vier Phasen unterschieden (Konzeptionalisierung, Entwurf, Konstruktion und Übergang), 2. werden jeder Phase eine Anzahl Iterationen zugeordnet und 3. werden in jeder Phase einige oder alle inhaltlichen Kernprozesse, wie Analyse & Design, Implementierung, aber auch Projektmanagement usw. ausgeführt. Jede dieser Phasen, Iterationen oder Kernprozesse können mit Hilfe übergeordneter Aktivitäten dargestellt werden, denen jeweils ein eigenes Aktivitätsdiagramm zur detaillierten Beschreibung zugeordnet werden kann.

Zur Beschreibung des Prozesses unterhalb dieser drei Ebenen werden nun besagte, für den RUP definierte Notationen verwendet. Dafür werden drei wesentliche Entitäten definiert, Worker, Aktivitäten und Konzepte, wobei die Verbindungen untereinander mit vorgegeben werden. Zur ergänzenden Prozessbeschreibung werden Richtlinien, Vorlagen und Toolmentoren definiert. Alles zusammen wird, wie in Abb. 4 gezeigt, eingesetzt und lässt sich gemäß Abb. 5 in Aktivitätsdiagramme überführen.

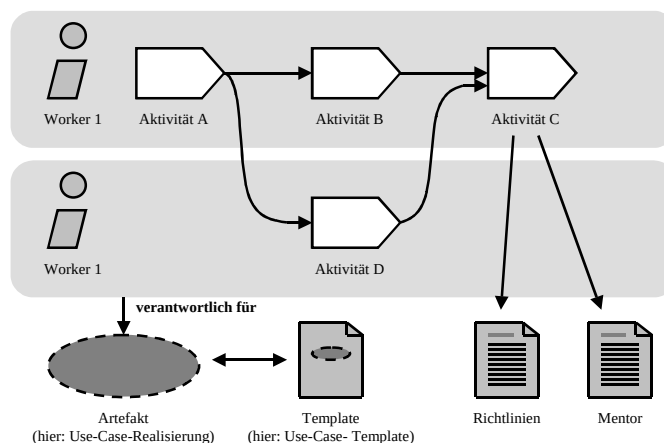


Abb. 4 - Notationen zur Modellierung des RUP nach [Kr99]

Der Vergleich zwischen Abb. 4 und Abb. 5 zeigt aber auch, dass die UML-Notationen in der Ausdrucksstärke gegenüber den RUP-Notationen zurückstehen. Dies ist nicht überraschend. Generell scheinen bei grafischen Notationen Allgemeingültigkeit und Ausdruckskraft in einem umgekehrt-proportionalen Verhältnis zueinander zu stehen. Eine

Fortführung des Vergleichs der hier gezeigten Prozessdarstellungen hin zu Petri-Netzen unterstreicht diese Vermutung.

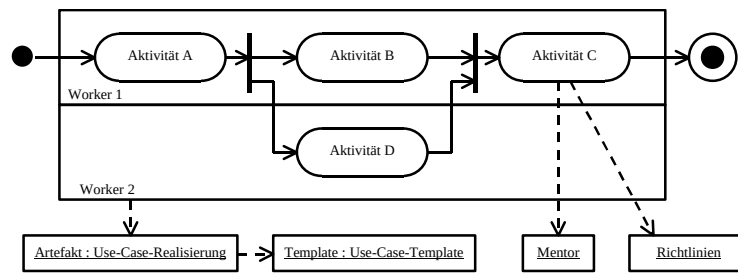


Abb. 5 - Darstellung des RUP als Aktivitätsdiagramm

4 Anpassung von Softwareentwicklungsprozessen

In der Regel ist im praktischen Projekteinsatz jeder vereinheitlichte Softwareentwicklungsprozess anzupassen, um den jeweiligen Gegebenheiten und Randbedingungen gerecht zu werden.

Insbesondere die agilen Prozesse begründen auf diesen Umstand sogar ihre Existenz. Daher unterscheiden sie sich auch in der Form, in der Anpassungen vorgenommen werden können bzw. müssen. Das Beispiel des XP zeigt, dass hier für weite Teile des Prozesses der Workflow nicht definiert wird. Diese Aufgabe wird, wie in Abschnitt 3.1 beschrieben, dem Anwender überlassen. Er muss dafür jedoch über ein weitreichendes Prozessverständnis verfügen. Aus diesem Grund wird der eigentliche Prozess in diesem Fall so einfach wie möglich gehalten. Eine Prozessdarstellung in Aktivitätsdiagrammen zur Prozesslaufzeit erfordert die Festlegung aller erforderlichen Transitionen, Entscheidungen und Synchronisationen. Hierfür anwendbare Formalismen anzugeben, die über die in Kapitel 2 vorgestellten Beschreibungen hinausgehen, erscheint an dieser Stelle schwierig.

Auch die Process Patterns bieten per se die Möglichkeit zur Anpassung des Prozesses an das Projekt. Dies geschieht in den zum Prozess gehörenden Aktivitätsdiagrammen gemäß Abschnitt 3.2 durch die Festlegung von Bedingungen. Bedingungen sind in den Notationen jedoch frei zu formulieren. Zudem ist für ihre Festlegung ein inhaltlicher Bezug zu den Aktivitäten oder ihren Ergebnissen herzustellen. Auch dieses Vorgehen erscheint nur schwer formalisierbar zu sein.

Starre Prozesse wie der RUP gewinnen ihre Anpassungsfähigkeit in der Regel aus der Möglichkeit einzelne Aktivitäten zu streichen. Dieses Vorgehen wird oft auch als Tailoring bezeichnet. Sind die Prozesse in Aktivitätsdiagrammen dargestellt, können die zu streichenden Aktivitäten in den Diagrammen problemlos entfernt werden. Bei der Verwendung eines CASE-Tools ist dies mühelos möglich. In der Regel werden dann aber auch die zur Aktivität gehörenden Eingangs- und Ausgangstransitionen gestrichen.

Dies kann jedoch zu unbeabsichtigten Prozessveränderungen führen. Beispielsweise könnte der Workflow vollständig unterbrochen werden. Aus diesem Grunde sind Regeln zu schaffen, die festlegen, wie im Falle des Tailorings die Diagramme zu pflegen sind. Tab. 3 zeigt vier praktische Regeln, mit deren diese Pflege gewährleistet werden kann.

1. Regel	<i>Zu streichende Aktivitäten, deren Eingangstransition aus einer anderen Aktivität kommt <u>oder</u> deren Ausgangstransition in eine weitere Aktivität führt, werden durch eine durchgehende Transition ersetzt.</i>
2. Regel	<i>Zu streichende Aktivitäten, die über ihre Ein- <u>und</u> Ausgangstransition jeweils mit einer Synchronisationslinie oder einer Verzweigung verbunden sind, werden zusammen mit den zugehörigen Transitionen aus dem Aktivitätsdiagramm entfernt.</i>
3. Regel	<i>Beim Wegfall einer Aktivität mit mehreren Ausgangstransitionen, ist zu prüfen, welche der angegebenen Bedingungen per se erhalten bleibt. Alle anderen zugehörigen Transitionen können dann gestrichen werden. Danach kann die Aktivität nach den o. a. Regeln aus dem Aktivitätsdiagramm entfernt werden. Rekursiv sind anschließend alle Aktivitäten, die über keine Eingangstransition mehr verfügen, zusammen mit ihren Ausgangstransitionen zu entfernen.</i>
4. Regel (Bereinigung)	<i>Synchronisationslinien oder Verzweigungen, die am Ende des Tailorings nur noch eine Eingangs- <u>und</u> eine Ausgangstransition aufweisen, werden durch eine durchgehende Transition ersetzt. Zudem werden Objekte und Signale, die keine Verbindungen zu Aktivitäten besitzen, entfernt. Selbiges geschieht mit leeren Schwimmbahnen.</i>

Tab. 3 - Regeln zum Tailoring starrer Prozesse

5 Fazit

In der praktischen Anwendung vereinheitlichter Softwareentwicklungsprozesse, egal ob allgemeine oder unternehmenseigene, hat sich die Verwendung der Unified Modeling Language (UML), konkret der Aktivitätsdiagramme, zur Prozessmodellierung bewährt. Dieses Vorgehen bringt drei wesentliche Vorzüge mit sich:

- Eine durchgängige, einheitliche Prozessbeschreibung.
- Eine prozessunabhängige Darstellung.
- Die Möglichkeit der formalisierten Prozessanpassung.

Mit der Anwendung der Aktivitätsdiagramme wurden aber auch Nachteile gegenüber anderen Prozessbeschreibungen deutlich, die offensichtlich miteinander verknüpft sind:

- Die Darstellungen können an Ausdruckstärke verlieren, so dass
- zum Teil mehrere Aktivitätsdiagramme benötigt werden, um eine Prozessdarstellung zu ersetzen.

Insgesamt hat sich jedoch gezeigt, dass diese Darstellungsform prinzipiell dem Problem der Prozessanalyse angemessen erscheint. Es bieten sich jedoch noch Erweiterungen dieses Vorgehens an:

- Die Formalisierung der Prozessanpassung für nicht-starre Prozesse.
- Die Überführung von Aktivitätsdiagrammen in Petri-Netze, um so eine formale Prozessanalyse, beispielsweise auf der Basis der in [De98] beschriebenen Verfahren, zu erreichen.

Insbesondere letztere Erweiterung bietet ein sehr weites Feld zur Fortführung des hier beschriebenen Verfahren.

Literaturverzeichnis

- [Am99a] Ambler, S. W.: Process Patterns. Cambridge University Press, 1999
- [Am99b] Ambler, S. W.: More Process Patterns. Cambridge University Press, 1999
- [Ba01] Balzert, H.: UML kompakt. Spektrum Akademischer Verlag, Heidelberg, Berlin, 2001
- [Be00] Beck, K.: Extreme Programming. Addison-Wesley, München, 2000
- [BF01] Beck, K.; Fowler, M.: Extreme Programming planen. Addison-Wesley, München, 2001
- [De98] Desel, J.: Petri-Netze, lineare Algebra und lineare Programmierung. Teubner Verlag, Leipzig, 1998
- [Kr99] Kruchten, P. von: Der Rational Unified Process. Addison-Wesley, München, 1999
- [OMG00] Object Management Group: Unified Modeling Language Specification (V. 1.3). <http://cgi.omg.org/cgi-bin/doc?formal/00-03-01.pdf>, 2000