Aspekte der komponentenorientierten Entwicklung adaptiver prozessorientierter Unternehmenssoftware*

Hilmar Acker[1], Colin Atkinson[2], Peter Dadam[1], Stefanie Rinderle[1], Manfred Reichert[1]

[1] Abteilung Datenbanken und Informationssysteme
Universität Ulm, D-89069 Ulm
{acker, dadam, rinderle, reichert}@informatik.uni-ulm.de

[2] Lehrstuhl für Softwaretechnik
Universität Mannheim, D-68161 Mannheim
atkinson@informatik.uni-mannheim.de

Abstract: Prozessorientierte Informationssysteme, Workflow-Management und komponentenorientierte Softwareerstellung sind Schlagworte, die in aller Munde sind. Jedoch werden diese Aspekte bzw. Technologien heute von kaum einem Unternehmen bei der Entwicklung ihrer Software und der Realisierung betrieblicher Informationssysteme wirklich konsequent um- bzw. eingesetzt – und wenn sie Verwendung finden, dann oft in einer Form, bei der das Potenzial dieser Ansätze nicht annähernd genutzt wird. In diesem Beitrag behandeln wir fundamentale Fragestellungen, die sich bei der Realisierung prozessorientierter Informationssysteme aus Anwendungskomponenten ergeben. Wir zeigen, wie diese Aspekte von uns im Aristaflow-Projekt aufgegriffen werden. Ziel ist die Entwicklung einer Software, die es erlauben wird, robuste und adaptive prozessorientierte Informationssysteme, mit Hilfe von Komponententechnologie und darauf zugeschnittenen Entwurfs- und Entwicklungsumgebungen im „Plug & Play"-Stil, zu realisieren.

1 Einleitung

Unternehmen, die am Markt erfolgreich sein wollen, müssen sich immer rascher auf neue Gegebenheiten einstellen können. Vielfach kommt es zu Änderungen in Organisationsstrukturen und Abläufen, zu neuen Formen der Zusammenarbeit mit Partnern, Zulieferern und Kunden oder zu anderen Veränderungen. Häufig betreffen solche Anpassungen nur einen bestimmten Geschäftsprozess, möglicherweise sogar nur einen einzelnen (wichtigen) Geschäftspartner oder Kunden. Die Fähigkeit, Geschäftsprozesse bei Bedarf zu individualisieren, wird deshalb zukünftig ein zentraler Wettbewerbsfaktor sein, ähnlich wie kundenspezifisch konfigurierte Serien-Pkw (vor nicht allzu vielen Jahren noch undenkbar) heute eine Selbstverständlichkeit sind [RD98, RRD03].

* Die Arbeit entstand im Rahmen des Aristaflow-Projektes (04/2004 – 03/2007), welches durch den Forschungverbund Unternehmenssoftware Baden-Württemberg gefördert wird.

Eine solche einzelfallbezogene Handhabung von Geschäftsprozessen lässt sich aber nur dann mit vertretbarem Aufwand und Fehlerrisiko bewältigen, wenn dies durch die eingesetzte Unternehmenssoftware in geeigneter Weise unterstützt wird. Diese Software (SW) muss eine prozessorientierte Sicht- und Denkweise aktiv unterstützen, und es erlauben, rasch und kostengünstig neue Prozesse (Prozesstypen) zu realisieren. Dabei sind Robustheit und Stabilität ebenso wichtig, wie die flexible Anpassbarkeit der Prozesse während ihrer Ausführung. So muss es für autorisierte Anwender möglich sein, bei Bedarf vom vorgeplanten Prozess abzuweichen, ohne dass es dadurch, im weiteren Prozessverlauf, systemseitig zu Fehlern oder Inkonsistenzen kommt. Eine starre Implementierung der Prozesse dagegen, wie für heutige Branchensoftware typisch, ist für viele Anwendungsdomänen nicht akzeptabel.

Dasselbe gilt für Änderungen am Geschäftsprozess (am Prozesstyp) selbst, etwa um diesen an gesetzliche Änderungen oder organisatorische Restrukturierungen anzupassen. Sie müssen systemseitig rasch und korrekt nachvollzogen, und – soweit gewünscht und möglich – auf bereits laufende Einzelprozesse dieses Typs übertragen werden können.

Prozessorientierte Unternehmenssoftware muss sowohl eigenentwickelte als auch zugekaufte Anwendungskomponenten integrieren. Es müssen auch Änderungen (z.B. Austausch oder Anpassung) der verwendeten SW-Bausteine unterstützt werden, wobei feststellbar sein muss, welche Prozesstypen infolge dieser Änderungen begleitend adaptiert werden müssen und welche Anpassungen daran (z.B. der Prozesslogik) vorzunehmen sind. Schließlich muss die Koexistenz von Komponenten unterschiedlicher Versionen möglich sein.

Technologische Antworten auf diese Herausforderungen sind adaptives Prozess-Management [RD98, RRD04a, RRD04b] und komponentenorientierte SW bzw. SW-Entwicklung [An03, GT00]. So befassen sich immer mehr Unternehmen damit, ihre Geschäftsprozesse zu optimieren und ihre Anwendungssysteme prozessorientiert auszurichten. Im Bereich der SW-Entwicklung wiederum wird Anwendungssoftware, aus Gründen der Wiederverwendung und zwecks Fehlerreduzierung, mehr und mehr komponentenorientiert gestaltet [At02, GMS02].

Trotz ihres komplementären Charakters werden diese beiden Paradigmen – Prozess- und Komponentenorientierung – bisher von den meisten Unternehmen bei der Entwicklung ihrer Software sowie der Realisierung ihrer betrieblichen Informationssysteme nicht wirklich konsequent ein- bzw. umgesetzt. Und wenn, dann meist in einer Form, bei der das gemeinsame Potenzial beider Ansätze brach liegt. Diese isolierte Betrachtung hat in der Vergangenheit z.B. dazu geführt, dass Prozesslogik oftmals im Anwendungscode fest „verdrahtet“ worden ist. Anpassungen sind dadurch nicht nur fehlerträchtig, sondern verursachen auch einen enorm hohen Zeitaufwand. Hinzu kommt die daraus resultierende Starrheit der Prozesse, die einen breiten Einsatz in der Praxis oftmals unmöglich macht. Bei Einsatz eines Workflow-Management-Systems (WfMS), mit expliziter Definition und Steuerung der Prozesse, hingegen werden interne Reihenfolgen- und Datenflussbeziehungen der verwendeten Komponenten meist nicht adäquat berücksichtigt. Dies wiederum erschwert die Wartung und Pflege der resultierenden Anwendungssysteme.

In diesem Beitrag behandeln wir ausgewählte Fragestellungen, die sich im Zusammenhang mit der komponentenorientierten Entwicklung adaptiver, prozessorientierter Unternehmenssoftware ergeben. In Kapitel 2 gehen wir auf den aktuellen Stand der Technik in den Bereichen Prozess-Management und Komponententechnologien ein. Anhand ausgewählter Aspekte vermitteln wir in Kapitel 3 einen Eindruck der Probleme, die sich beim Versuch, beide Paradigmen zusammenzuführen, ergeben. Entsprechende Fragestellungen werden von uns im Rahmen des Projekts Aristaflow untersucht, dessen Ziele und Inhalte wir in Kapitel 4 skizzieren. Der Beitrag schließt mit einer Zusammenfassung.

2 Stand der Technik

Die mangelhafte Integration von Prozess-Management- und Komponententechnologien ist darauf zurückzuführen, dass viele Probleme und Fragestellungen bislang isoliert betrachtet werden. Dies soll nachfolgend anhand einfacher Beispiele illustriert werden.

2.1 Komponententechnologien

Seitens der SW-Technologie sind in den letzten Jahren gute Fortschritte im Umfeld komponentenbasierter SW-Erstellung erzielt worden. So gibt es mittlerweile eine Reihe von Komponentenmodellen (z.B. COM+, CORBA, EJB, WebServices), für die mächtige Entwicklungs- und Ausführungsplattformen existieren (z.B. [DP00, Du02, GT00]).

Im Folgenden bezeichnen wir mit einer Komponente einen SW-Baustein, dessen Schnittstelle durch das zugrunde liegende Komponentenmodell vertraglich genau spezifiziert ist. Eine Komponente erwartet in ihrer Einsatzumgebung lediglich die im Komponentenmodell festgelegten Abhängigkeiten, so dass ein einheitlicher sowie kontext- und herstellerunabhängiger Zugriff auf sie möglich wird [Du02]. Die Beschreibung der Schnittstellen (z.B. Signatur aufrufbarer Operationen und Abhängigkeiten zwischen diesen Operationen) erfolgt in den gebräuchlichen Komponentenmodellen rein syntaktisch – beinhaltet also im Wesentlichen, welche „Steckdosen“ die Komponente anbietet.

Heute werden Komponenten vom Programmierer praktisch „von Hand“ ausgewählt und zu einer neuen Anwendung zusammengesetzt. Die Komposition einer (prozessorientierten) Anwendung wird meist in gängigen Programmier- oder Skriptsprachen realisiert, d.h. den „Kleber“ zwischen den Komponenten bildet Quellcode (vgl. Abb. 1). Es ist daher die Aufgabe des Anwendungsprogrammierers, festzulegen, in welcher Reihenfolge bzw. unter welchen Bedingungen die eingesetzten Komponenten ausgeführt und welche Daten zwischen ihnen ausgetauscht werden. Hierbei verbringt er viel Zeit mit der Konvertierung von Daten (zwecks Austausch zwischen Komponenten), der Überprüfung der Parameterversorgung von Komponenten, der Fehlerbehandlung oder der Einbeziehung trans-

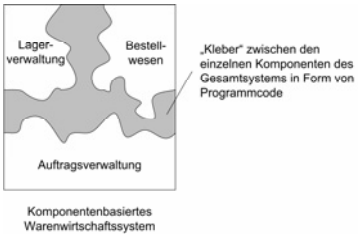

Abb. 1: Zusammensetzen von Komponenten mittels Programmcode

aktionaler Aspekte [AC04]. Weitergehende Prozessunterstützungsfunktionen, etwa für das Logging oder das partielle Zurücksetzen von Prozessen im Fehlerfall, müssen explizit ausprogrammiert werden. Dies alles führt dazu, dass Prozesslogik und systemnahe Aspekte im Quellcode „vermischt" sind, was die Pflege und Wartung solcher Anwendungen aufwendig gestaltet. Erschwerend kommt die mangelhafte Interoperabilität existierender Komponentenmodelle hinzu.

Eine wichtige Anforderung an prozessorientierte Anwendungen betrifft ihre Wartbarkeit. Wie eingangs motiviert, treten Änderungen auf zwei Ebenen auf: Zum einen können sich die Geschäftsprozesse selbst ändern, was rasche und korrekte Anpassungen der Ablauflogik der sie unterstützenden Anwendungen erfordert (vgl. Abb. 2, links). Zum anderen muss es möglich sein, einzelne Komponenten bei Bedarf abzuändern oder auch auszutauschen (vgl. Abb. 2 rechts). In beiden Fällen werden meist aufwendige Eingriffe in den Quellcode erforderlich, um das Anwendungssystem an die neuen Gegebenheiten anzupassen. Dabei müssen sowohl interne Beschränkungen der verwendeten Komponenten (z.B. Aufrufreihenfolgen ihrer Operationen) als auch globale Geschäftsregeln beachten werden.

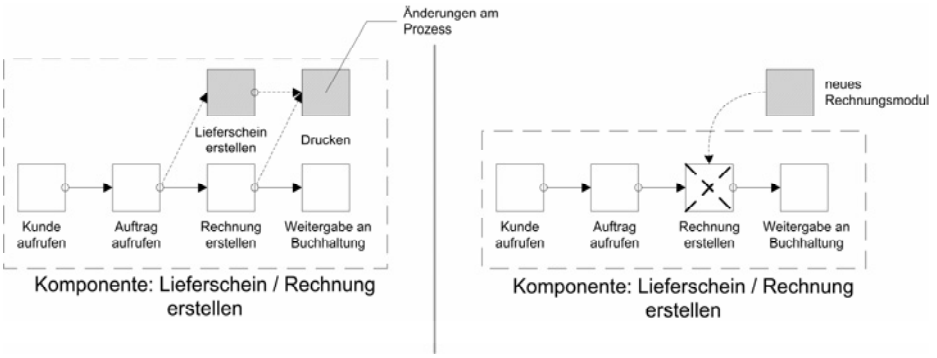

Abb. 2: Änderung prozessorientierter, komponentenbasierter Anwendungen

In der Praxis sind zur Prüfung der korrekten „Komponentenverschaltung" üblicherweise aufwendige Testfälle zu definieren, die dann im Änderungsfall durchgespielt werden müssen. Automatische Analysen zur Quellcodeprüfung bzw. zur Sicherstellung der oben genannten Aspekte, wie sie bei expliziter Modellierung der Prozesse (siehe unten) teilweise möglich sind, scheiden bei fester Verdrahtung der Abläufe im Programmcode im Allgemeinen aus. Diese Einschränkung soll bei *Model Driven Architectures* (MDA) entfallen [An03, OYP03]. Durch modellseitige Beschreibung und Validation der Struktur und des Verhaltens des Zielsystems (z.B. mittels UML) soll die Grundlage für die automatische Generierung von Programmcode geschaffen werden [An03]. Dadurch soll der Aufwand für Entwicklung und Wartung der Anwendungssysteme signifikant reduziert werden.

Unabhängig davon, ob der Programmcode für das Zusammensetzen von Komponenten manuell oder automatisch erzeugt wird, bestehen bei einige fundamentale Probleme: Nach Implementierung von Änderungen am Geschäftsprozess (d.h. am Prozesstyp) ist eine getrennte Ausführung von alten und neuen Prozessen nicht mehr möglich. Je nach

Umfang der Änderungen kann sogar eine totale Inkompatibilität des alten und des neuen Prozesses die Folge sein. Hier bieten existierende Ansätze keine vernünftige Lösung. Ein weiteres Problem betrifft die einzelfallbezogene Anpassung einzelner Instanzen. Sofern nicht als Spezialfälle in der Anwendung hinterlegt, lassen sich diese Ad-hoc-Abweichungen (z.B. dynamisches Auslassen oder Einfügen einzelner Prozessschritte), bei „Verstecken“ der Prozesslogik im Programmcode, nicht realisieren.

2.2 Workflow-Management-Technologie

Workflow-Management-Systeme (WfMS) bieten mächtige Entwicklungs- und Ausführungskomponenten für die Modellierung, Steuerung und Überwachung von Geschäftsprozessen. Charakteristisch ist die Trennung von Prozesslogik und Anwendungscode. Gegenüber der reinen Codierung im Anwendungsprogramm ermöglicht dies – zumindest vom Prinzip her – eine bessere Anpassbarkeit prozessorientierter Anwendungen.

Kommerzielle WfMS konnten die in sie gesetzten Erwartungen, gerade wegen ihrer Starrheit, bisher nicht erfüllen. Dagegen hat es in den vergangenen Jahren in der Forschung innovative Entwicklungen gegeben, welche die rasche und fehlerfreie Adaption von Workflows (im laufenden Betrieb) ermöglichen. So ist es im Prinzip nun möglich, prozessorientierte Anwendungen zu entwickeln und unternehmensweit zum Einsatz zu bringen, bei denen die Anwender zur Laufzeit flexibel auf Ausnahmesituationen reagieren können [RRD04a, RRD04b]. So lassen sich einzelfallbezogen Prozessschritte (sog. Aktivitäten) dynamisch einfügen, löschen oder verschieben, ohne dass diese Ad-hoc-Änderungen im weiteren Verlauf der Ausführung zu Inkonsistenzen oder Fehlern führen. Ebenso gibt es umfassende Konzepte für die Unterstützung von Änderungen am Geschäftsprozess selbst (d.h. Änderungen auf Prozesstypebene) und deren Propagation auf laufende Prozessinstanzen (soweit gewünscht und möglich). – Die von uns entwickelte ADEPT-Technologie etwa, unterstützt beide Arten von Änderungen [RRD03].

Bisher gehen (adaptive) WfMS aber davon aus, dass die Anwendungsfunktionen bzw. -programme, die den verschiedenen Aktivitäten eines Prozessmodells zugeordnet werden, voneinander unabhängig sind. Diese Annahme trifft bei Einbeziehung von Komponenten, die mehrere Operationen mit internen Kontroll- und Datenabhängigkeiten umfassen können, bekanntermaßen nicht zu. Wird eine solche Operation einer Prozessaktivität zugewiesen, muss der Prozessmodellierer sicherstellen, dass evtl. Abhängigkeiten zu anderen Operationen derselben Komponente erfüllt sind. So dürfen z.B. die im Prozessmodell festgelegten Kontrollabhängigkeiten nicht im Widerspruch zu den von den Komponenten geforderten Reihenfolgenbeziehungen für Operationen stehen.

Allerdings besitzen heutige WfMS selbst keine Kenntnis von den beim Entwurf beachteten Komponenteneigenschaften und verwaltet diese auch nicht explizit. Dies ist deshalb problematisch, weil bei späteren Änderungen der Prozesse nicht unmittelbar erkennbar ist, ob die geforderten Beschränkungen seitens der Komponenten weiterhin erfüllt sind. Dies mag für reine Schemaänderungen durch einen Prozessmodellierer noch beherrschbar erscheinen, gestaltet sich aber bei der Realisierung von Ad-hoc-Änderungen durch Endanwender sehr problematisch. Auch bei späteren Änderungen der Komponenten selbst ist nicht unmittelbar erkennbar, welche Prozesse davon betroffen sind.

Abb. 3 zeigt am Beispiel eines Bestellprozesses, über welche Informationen ein WfMS typischerweise verfügt. Die Trennung von Prozesslogik und Anwendungskomponenten ist schematisch angedeutet. Das WfMS weiß nichts über Internas der Komponenten und umgekehrt. Es kann deshalb keine Unterstützung bei der Prozessmodellierung bieten.

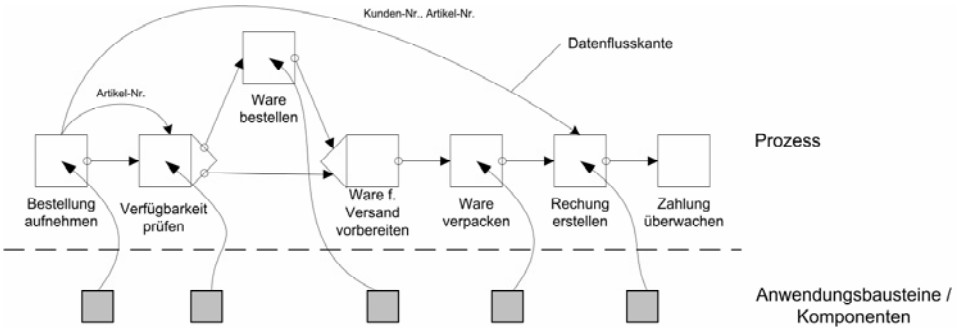

Abb. 3: Verwendung von Anwendungskomponenten in WfMS

2.3 Zwischenfazit

In existierenden WfMS muss der Modellierer die Komponenten von Hand in den Prozess einbinden und dafür Sorge tragen, dass diese in der angegebenen Reihenfolge ausführbar sind. Das WfMS gibt ihm dabei nur wenig Hilfestellung. Allenfalls kann geprüft werden, ob die angegebenen Parameter der gerufenen Prozessaktivitäten versorgt und die modellierten Datenflüsse korrekt sind. Da einzelfallbezogene Abweichungen vom vorgeplanten Ablauf unterstützt werden müssen, ist diese Situation unzureichend. Insbesondere müssen Ad-hoc-Abweichungen durch Endanwender festlegbar sein, ohne dass sie über Detailwissen zu den verwendeten Komponenten verfügen müssen.

Bis heute nicht wirklich verstanden ist, wie Komponenten zu den Aktivitäten eines Prozessmodells passen bzw. wie Prozesse und Komponenten zu beschreiben sind, damit sich beide auf harmonische Weise und systemunterstützt zusammenfügen lassen. So schließt die Verifikation von Prozessmodellen in WfMS nicht mit ein, ob die Rahmenbedingungen der Komponenten in diesem Kontext überhaupt erfüllt sind. Dies sicherzustellen ist Aufgabe des Prozessmodellierers, der dazu sehr viel Hintergrundwissen über die Komponenten besitzen muss. Lediglich der im Prozess explizit modellierte Datenfluss schränkt die Kombinationsmöglichkeiten der Komponenten ein.

Web Services bieten einen viel versprechenden Ansatz zur Verknüpfung der beiden Welten [AC04]. Sie reichen von einfachen zustandslosen Anwendungen, die auf eine Anfrage eine Antwort geben, bis zu prozessorientierten Anwendungen, die selbst wieder mehrere Web Services enthalten. Mittlerweile gibt es zahlreiche Standards, Methoden und Werkzeuge für die prozessorientierte Komposition und Orchestrierung von Web Services [AC04, MBE03, OYP03]. Einige davon beziehen Semantikaspekte in den Kompositionsprozess mit ein (z.B. [HG03, MBE03, SM03]). Hinzu kommen mächtige Entwicklungs- und Laufzeitplattformen, z.B. WebSphere Process Choreographer

[WS04] und MS BizTalk [BI04]. Die vorangehend aufgeworfenen Fragestellungen, die Änderung von Komponenten (Services) und Prozessen betreffend, bleiben aber bestehen. Aspekte der (dynamischen) Adaption von Service Flows wurden bisher kaum behandelt.

3 Ausgewählte Aspekte

Damit ein WfMS die Prozessmodellierer und Anwender bei der Erstellung und Änderung komponentenbasierter Prozesse optimal unterstützen kann, muss es über zusätzliche Informationen zu den verwendeten Komponenten verfügen. Diese sind vom Komponentenentwickler bereitzustellen und in das WfMS zu importieren (z.B. Zugriff auf Beschreibungsdateien, Nutzung von Abfrageschnittstellen). Letzteres macht deshalb Sinn, weil das WfMS in der Lage sein muss, einzelfallbezogene Ad-hoc-Änderungen performant zu unterstützen, d.h. ohne aufwendige Zugriffe auf Komponentenbeschreibungen aus einem Repository. Welche Art von Informationen ein adaptives WfMS benötigt, um das Zusammenwirken zwischen Workflow- und Komponententechnologie optimal zu gestalten, wird in diesem Kapitel anhand ausgewählter Beispiele skizziert.

3.1 Datenflüsse in prozess- und komponentenorientierten Anwendungen

Heutige WfMS beachten bei Analysen – wenn überhaupt – lediglich den im Prozessmodell explizit festgelegten Datenfluss zwischen Aktivitäten (materialisierter Datenfluss; siehe [Re00]). Dies ist in Verbindung mit Komponenten, die aus mehreren Operationen bestehen oder die auf einer gemeinsamen Datenbasis arbeiten, nicht ausreichend. Hier findet der Datenaustausch teilweise nicht über die Schnittstellen sondern unter der Oberfläche statt (z.B. über die Datenbank). Er ist deshalb im Prozessmodell nicht sichtbar und steht somit nicht unter der Kontrolle des WfMS. Es ergeben sich Abhängigkeiten bzgl. möglicher Aufrufreihenfolgen der Komponenten (bzw. ihrer Operationen), die aus den verfügbaren Schnittstellenbeschreibungen für das WfMS nicht erkennbar und somit modellseitig nicht prüfbar sind. Stattdessen muss die Einhaltung der richtigen Operationsreihenfolge vom Modellierer gewährleistet werden. Kennt bzw. beachtet er die internen Datenabhängigkeiten nicht, wird der Prozess vom WfMS zwar als (syntaktisch) korrekt bewertet, die von einer Komponente zur Laufzeit benötigten Daten liegen dann jedoch nicht vor. Schlimmstenfalls führt dies zu Fehlern bei der Komponentenausführung.

Diese Problematik lässt sich entschärfen, indem die in der Schnittstellenbeschreibung nicht aufgeführten („versteckten“) Datenflüsse explizit gemacht werden. Dies kann z.B. über *virtuelle* Datenkanten zwischen jeweils zwei abhängigen Komponenten (bzw. Operationen) erfolgen (siehe Abb. 4). Virtuelle Datenflüsse spielen für die Ausführung der Prozesse zwar keine Rolle, können aber dazu genutzt werden, die Korrektheit der Komposition von Komponenten oder deren Änderung zu prüfen. Um virtuelle Datenflüsse beim Einsetzen der Komponenten in Prozessmodelle automatisch ableiten zu können, müssen auch bei der Komponentenbeschreibung entsprechende Informationen angegeben werden.

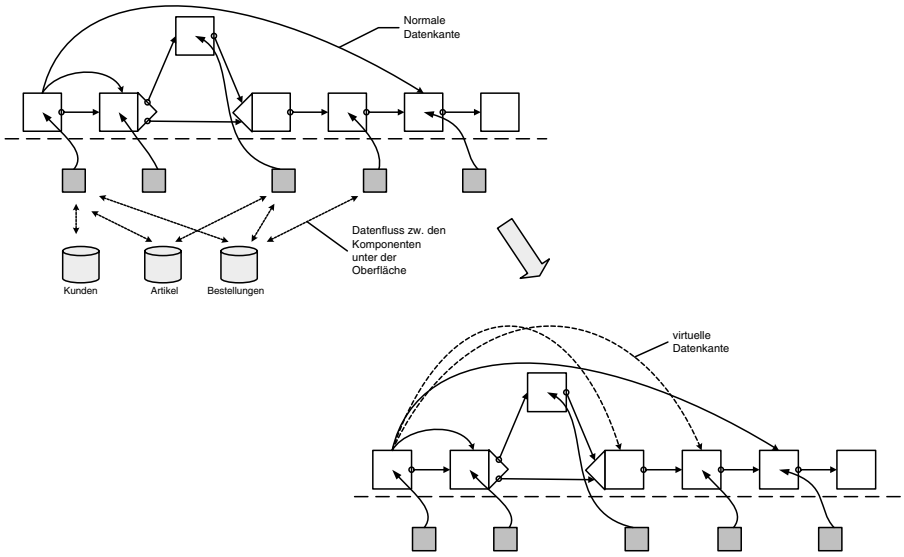

Abb. 4: Abbildung „interner“ Datenflüsse zwischen Komponenten auf virtuelle Datenflüsse

Ein weiterer Aspekt betrifft die (materialisierten) Daten, die zwischen Komponenten bzw. einzelnen Operationen über ihre Aufrufschnittstelle ausgetauscht werden. Die konkrete Versorgung und Übernahme der Aufrufparameter erfolgt über den Datenkontext der jeweiligen Prozessinstanz. Grundlegend ist die korrekte Festlegung der Datenflüsse im Prozessmodell. Um den Prozessmodellierer bei der Datenflusserstellung adäquat unterstützen zu können, müssen auch hierzu aussagekräftige Informationen zu den verwendeten Komponenten vorliegen. Von Vorteil ist bei der Beschreibung der Komponentenschnittstellen auch die Verwendung kontrollierter Bezeichner (aus einem domänenspezifischen Vokabular) und evtl. die Bereitstellung von Konvertierungsroutinen. Da diese Aspekte für die Komponenten von Drittherstellern meist nicht konsequent eingehalten werden, muss hier eine Anpassung der Parameterbezeichnungen beim Import der Komponente in das WfMS erfolgen. Zusätzlich zur Generierung virtueller Datenflüsse (siehe oben) ist die automatische Ableitung materialisierter bzw. expliziter Datenflüsse dann – zumindest vom Prinzip her – möglich. Dies ist vor allem für Ad-hoc-Änderungen einzelner Prozessinstanzen, welche von Endanwendern vorgenommen werden, nützlich.

3.2 Globale Prozesszustände und interne Zustände einer Komponente

Interne Datenflüsse zwischen Komponenten oder einzelnen Operationen einer Komponente stellen lediglich eine Rahmenbedingung bei deren Komposition dar. Bei Verwendung zustandsbehafteter Komponenten ergeben sich weitere Beschränkungen bzgl. zulässiger Operationsabfolgen und –anwendungen. Konkret ist das Verhalten einer zustandsbehafteten Komponente abhängig von ihrem aktuellen internen Zustand (siehe Abb. 5). Das bedeutet zum einen, dass je nach Zustand unterschiedliche Operationen

aufrufbar sein können, und zum anderen kann sich das Verhalten einer bestimmten Operation in verschiedenen Zuständen ändern. Beispielsweise könnte beim Aufruf einer Operation „Auftragsdruck“ der Komponente „Auftragsverwaltung“, je nach Zustand dieser Komponente, entweder der Lieferschein oder die Rechnung gedruckt werden.

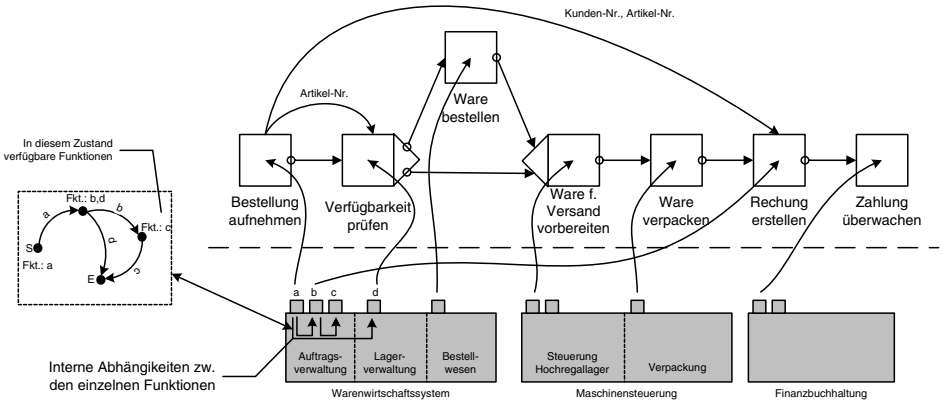

Abb. 5: Zustandsbehaftete Komponenten mit internen Abhängigkeiten ihrer Operationen

Solche verhaltensbezogenen Aspekte müssen bei der Komponentenbeschreibung wieder explizit gemacht werden. Nur durch genaue Kenntnis aller Vor- und Nachbedingungen der einzelnen Operationen, kann sichergestellt werden, dass ihre Verwendung innerhalb eines Prozessmodells konform mit den Beschränkungen der betreffenden Komponente ist. In der gängigen Praxis übernimmt der Prozessmodellierer die Aufgabe, diese Beschränkungen zu identifizieren und sie bei der Komposition zu beachten. Wie bereits im vorangehenden Abschnitt diskutiert, ist eine solche Vorgehensweise für die flexible Unterstützung von (dynamischen) Prozessänderungen nicht ausreichend. Hier müssen Abhängigkeiten zwischen den Operationen einer Komponente dem adaptiven WfMS bekannt gemacht und von diesem verwaltet werden, um z.B. Ad-hoc-Änderungen korrekt und performant bewerkstelligen zu können.

Interessante Perspektiven bieten hier Ansätze zur expliziten Beschreibung aller Vor- und Nachbedingungen von Operationen. Beispielhaft sei hier die *Object Constraint Language* (OCL) genannt [WK99]. Allerdings haben sich entsprechende Ansätze in der Praxis bisher noch nicht in erhofftem Maße durchsetzen können.

3.3 Ganzheitlicher Entwurf prozessorientierter Anwendungen

Eine aus der Komposition von Einzelkomponenten resultierende Anwendung kann selbst wieder als Komponente dienen. Bei herkömmlicher Vorgehensweise besteht eine Komponente dem entsprechend aus Teilkomponenten und dem sie verknüpfenden Quellcode. Übertragen auf WfMS bedeutet dies, dass ein (Teil-)Prozess wieder als Komponente

aufgefasst werden kann. Somit können häufig genutzte Prozessfragmente nicht nur als syntaktische Vorlage im System hinterlegt werden, sondern auch als voll funktionsfähige Komponente dienen. Eine der Fragen, die für eine derartige Unterstützung in WfMS geklärt werden muss, ist, wie sich die Schnittstelle dieser neuen Komponente aus den Schnittstellen der Einzelkomponenten ergibt bzw. ob diese aus den Informationen der Teilkomponenten automatisch generiert werden kann.

Bei der komponentenbasierten Entwicklung prozessorientierter Anwendungen stellt sich die grundlegende Frage, wie viel Ablaufsteuerung innerhalb einer Komponente angesiedelt wird (z.B. um eine Folge von Masken zu steuern) und welche Prozesssteuerung „oben“ im WfMS stattfinden soll. In dem von uns verfolgten Ansatz (siehe Kapitel 4) streben wir eine einheitliche Beschreibung der internen Ablauflogik einer Anwendungskomponente (*Mikroprozess*) und der im WfMS hinterlegten Prozesslogik (*Makroprozess*) an. Dies schafft die Möglichkeit, den Gesamtprozess zunächst „ganzheitlich“ zu entwerfen und erst relativ spät zu entscheiden, welche Teile davon innerhalb von Komponenten realisiert werden und welche durch ein WfMS ausgeführt werden sollen (vgl. Abb. 6). Denkbar ist in diesem Zusammenhang auch, dass Mikroprozesse (z.B. einfache Maskenflüsse mit ein und demselben Bearbeiter) nicht in der Komponente verankert, sondern durch ein leichtgewichtiges WfMS (auf dem betreffenden Klientenrechner) ausgeführt werden. Hierdurch ließen sich viele Prozessunterstützungsfunktionen, die man auf Makroprozesse anwenden kann (z.B. einzelfallbezogene Abweichung vom vorgegebenen Maskenfluss), auch auf den Mikroprozess übertragen.

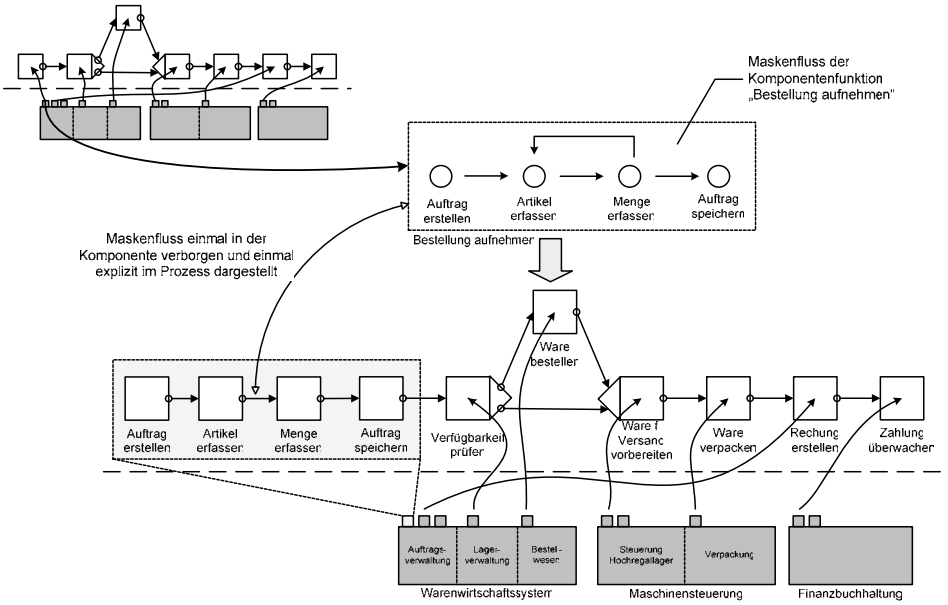

Abb. 6: Makro- und Mikroprozesse

3.4 Transaktionale Aspekte

Fundamentale Anforderungen resultieren aus der Unterstützung von Transaktionen (vgl. Abb. 7). Zum einen muss gewährleistet sein, dass interne Zustandsübergänge einer Komponente und das Weiterschalten im Workflow synchronisiert werden. So darf eine persistente Komponente nicht mit „Commit“ enden, wenn dieses Commit vom WfMS nicht bestätigt wird (horizontale Transaktion). Zum anderen muss es möglich sein, mehrere Prozessaktivitäten in einer atomaren Transaktion zu kapseln (vertikale Transaktion). Hierbei ist auch relevant, inwieweit man die Transaktionslogik, ähnlich wie bei Enterprise Java Beans, außerhalb der Komponenten definieren kann [DP00]. Des Weiteren kann es sein, dass eine Komponente nur in gewissen Zuständen transaktionsfähig ist oder die Ausführung nur durch eine entsprechende Kompensationsaktivität rückgängig gemacht werden kann. Ein Beispiel für einen solchen Fall ist die Finanzbuchhaltung. Fehlbuchungen dürfen hier nicht gelöscht, sondern nur durch eine entsprechende Gegenbuchung aufgehoben werden.

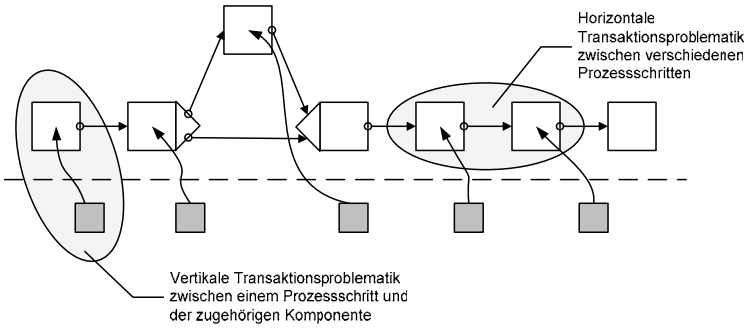

Abb. 7: Transaktionale Aspekte

3.5 Zusammenfassung

Im Zusammenhang mit der komponentenbasierten Entwicklung prozessorientierter Anwendungen gibt es zahlreiche Fragestellungen, die einer systematischen Klärung bedürfen, bevor man die Vision des „Plug & Play“ von Anwendungskomponenten in die Tat umsetzen kann:

- Wie muss eine Komponente beschrieben sein, damit „Plug & Play“ im Zusammenwirken mit einem WfMS reibungslos funktioniert? Wie werden Abhängigkeiten zwischen Komponenten beschrieben? Wer stellt diese Informationen bereit? Inwieweit lassen sich diese automatisch bei der Komponentenentwicklung ableiten?

- Wie viel Ablaufsteuerung soll innerhalb einer Komponente und wie viel „oben“ auf der Prozessebene angesiedelt sein? Macht es Sinn, zunächst „ganzheitlich“ zu entwerfen und diese Aufspaltung erst relativ spät vorzunehmen oder soll dieser Split bereits bei der Prozessmodellierung bzw. Komponentenentwicklung als gegeben vorausgesetzt werden?

- Auf welcher Ebene werden Transaktionsaspekte realisiert (einfaches Commit, Zwei-Phasen-Commit, Kompensation) – auf Prozessebene, in den Komponenten, in der Ausführungsumgebung der Komponenten, ...?

- Welche Unterstützung für das Zurücksetzen von Prozessen und evtl. Storno-Aktivitäten soll systemseitig zur Verfügung gestellt werden? Was ist hierzu auf E-bene des WfMS zu tun, was muss hierfür auf Komponentenebene bzw. im Rahmen der Komponentenimplementierung getan werden?

- Welche Daten werden über das WfMS verwaltet, welche Daten verwaltet die Anwendung, wie geht man mit impliziten Datenflüssen um? (Dies ist insbesondere bei der Integration bereits existierender Anwendungen wichtig.)

- Wie gestaltet man interaktive bzw. graphische Komponenten, so dass sie sich leicht an verschiedene Ausführungskontexte anpassen lassen?

4 Aristaflow – „Plug & Play" von Anwendungskomponenten

Die vorangehend motivierten Fragestellungen (und weitere) werden von uns eingehend im Projekt Aristaflow untersucht. Ziel dieses Verbundprojektes, dem Partner aus Wissenschaft und Industrie angehören[1], ist die Entwicklung einer umfassenden SW-Technologie, die es erlaubt, robuste und adaptive prozessorientierte Informationssysteme mit Hilfe von Komponententechnologie und darauf zugeschnittenen Entwurfs- und Entwicklungsumgebungen im „Plug & Play"-Stil zu realisieren (vgl. Abb. 8). Das resultierende Gesamtkonzept soll einen signifikanten technologischen Fortschritt gegenüber dem bisherigen Stand der Technik darstellen und die Vorzüge von Komponententechnologie und adaptiven Prozess-Management-Systemen miteinander kombinieren.

Die geforderte Robustheit erreichen wir, indem wir sowohl bei der Entwicklung der Komponenten als auch bei deren Komposition zu einem Prozess eine Reihe von Verifikations- und Testmethoden (z.B. Ausschluss von Verklemmungen zur Laufzeit, Gewährleistung korrekt modellierter Datenflüsse) einsetzen. Auf Prozessebene etwa stellen wir bereits zur Entwicklungszeit sicher, dass beim späteren Aufruf einer Komponente ihre Eingabeparameter aus dem Prozessdatenkontext korrekt versorgt werden können. Auch interne (Daten-)Abhängigkeiten zwischen Komponenten bzw. einzelnen Operationen (vgl. Kapitel 3) können den Prozessmodellen über virtuelle Datenkanten bekannt gemacht werden [Kr03]. Dadurch erreichen wir, dass bei Änderungen dieser Modelle die Beschränkungen der verwendeten Komponenten nicht verletzt werden.

[1] Nähere Angaben zu Projektpartnern und –zielen finden sich unter www.aristaflow.de!

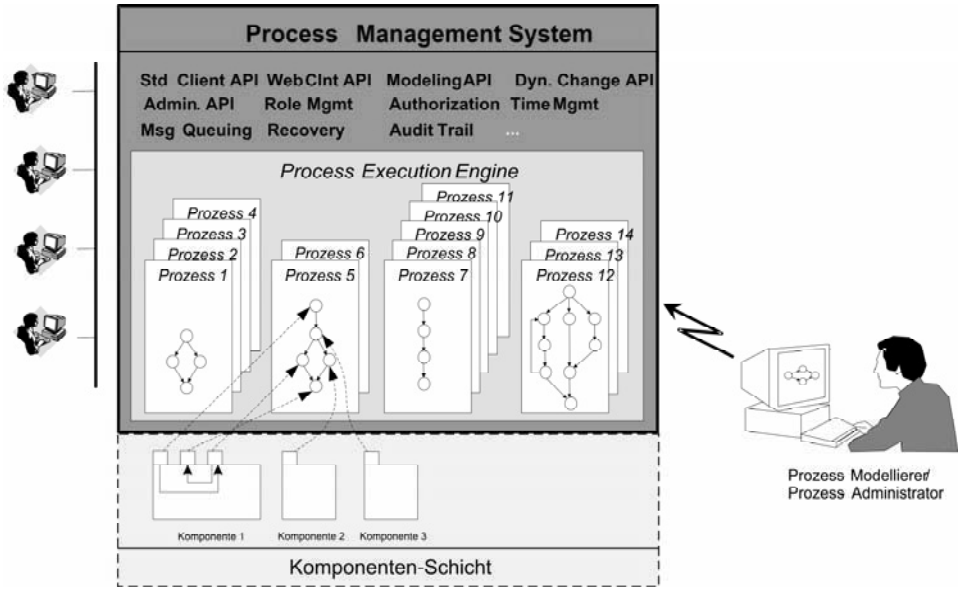

Abb. 8: Adaptives, komponentenorientiertes Prozess Management System (vereinfacht)

Adaptivität soll in mehrfacher Hinsicht erreicht werden. Zum einen soll es die Komponententechnologie ermöglichen, dass eine bestimmte Anwendungskomponente für verschiedene prozessorientierte Anwendungen verwendet werden kann.[2] Zum anderen ist Adaptivität hinsichtlich der Ausführung von Prozessen, d.h. zur Laufzeit, gegeben. Sehr gut verstanden sind von uns mittlerweile Konzepte zur dynamischen Adaption komponentenbasierter, prozessorientierter Anwendungen im laufenden Betrieb (z.B. [RRD04a, RRD04b, RRD03]). Die von uns entwickelte ADEPT Prozess-Management-Technologie unterstützt sowohl einzelfallbezogene Ad-hoc-Änderungen [RD98] als auch die kontrollierte Evolution von Prozess-Schemata (inkl. der Propagation solcher Schemaänderungen auf laufende Prozessinstanzen [RRD03, RRD04a], siehe Abb. 9). Neben umfassenden Konzepten haben wir mit dem ADEPT-System einen mächtigen SW-Prototyp entwickelt, der auch von verschiedenen Partnern zur Entwicklung fortschrittlicher prozessorientierter Anwendungssysteme genutzt wird (z.B. [MGR04, BKK04]).

Ein Alleinstellungsmerkmal des ADEPT-Ansatzes ist die umfassende und korrekte Unterstützung von Ad-hoc-Änderungen im Verlauf der Ausführung von Prozessinstanzen. Autorisierte Anwender können Prozessaktivitäten dynamisch einfügen, löschen oder verschieben. Durch Kombination solcher elementaren Änderungsoperationen lassen sich zudem semantisch höherwertige Änderungen verwirklichen, etwa zur Umsetzung spontaner Vorwärts- und Rückwärtssprünge im Workflow (z.B. vorzeitige Bearbeitung einer Aktivität mit Nachholen der übersprungenen Schritte).

[2] Bei der Komponentenbeschreibung orientieren wir uns an der *Object Constraint Language* (OCL) [WK99].

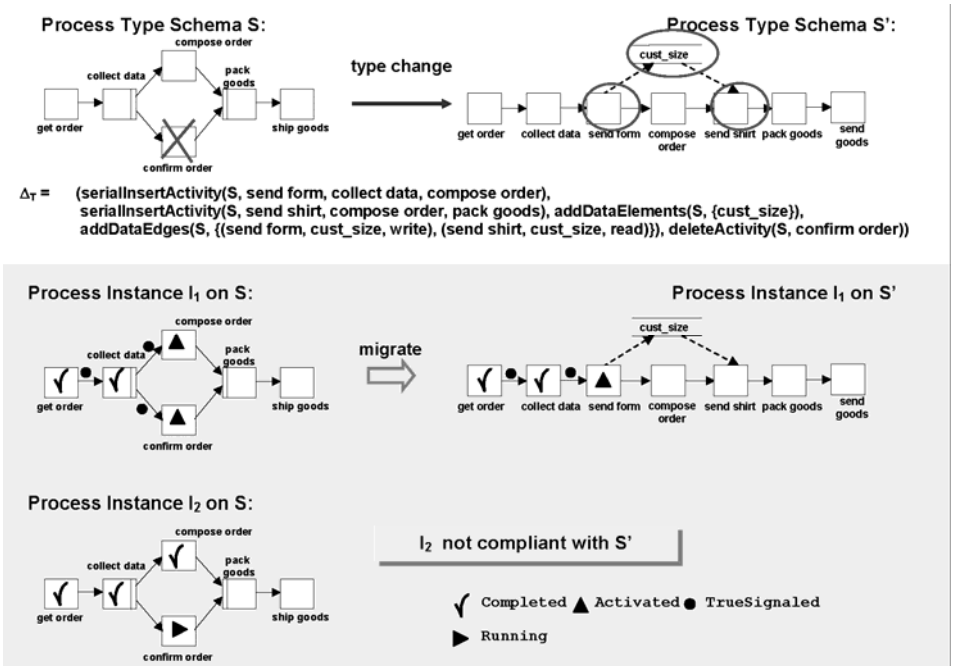

Abb. 9: Propagierung einer Schemaänderung auf die laufenden Instanzen unter Beachtung ihres Ausführungszustandes

Instanzänderungen sind in ADEPT nur dann zulässig, wenn sie die Korrektheit und Konsistenz der Prozessinstanzen nicht verletzen. Das ADEPT-System stellt bei einzelfallbezogenen Instanzänderungen nicht nur strukturelle und dynamische Eigenschaften der zugrunde liegenden Prozessmodelle sicher, sondern schränkt die Anwendbarkeit von Änderungsoperationen weiter, durch den aktuellen Zustand der jeweils betrachteten Prozessinstanz, ein. So dürfen bereits beendete Aktivitäten nicht mehr gelöscht oder ihre Ausführungsattribute nachträglich verändert werden. Ebenso wenig darf eine neue Prozessaktivität in einen bereits abgearbeiteten Teil einer Prozessinstanz eingefügt werden (vgl. Instanz I_2 aus Abb. 9). Über virtuelle Abhängigkeiten im zugrunde liegenden Prozessmodell kann zudem sichergestellt werden, dass bei der Abänderung einzelner Instanzen keine Verletzungen der geforderten Komponenteneigenschaften auftreten (siehe Abb. 10).

Abb. 10 zeigt, wie interne Komponentenabhängigkeiten in ADEPT bei der Durchführung von (dynamischen) Prozessänderungen berücksichtigt werden: Abhängigkeiten, zwischen einzelnen Operationen einer Komponente etwa, werden auf virtuelle Kanten zwischen den entsprechenden Prozessaktivitäten abgebildet (siehe oben). Mit diesen Informationen kann bei Ad-hoc-Änderungen zur Laufzeit die Einhaltung der internen Abhängigkeiten direkt auf Basis des Prozessschemas geprüft werden, ohne das dazu ein Zugriff auf das Komponentenrepository notwendig wird.

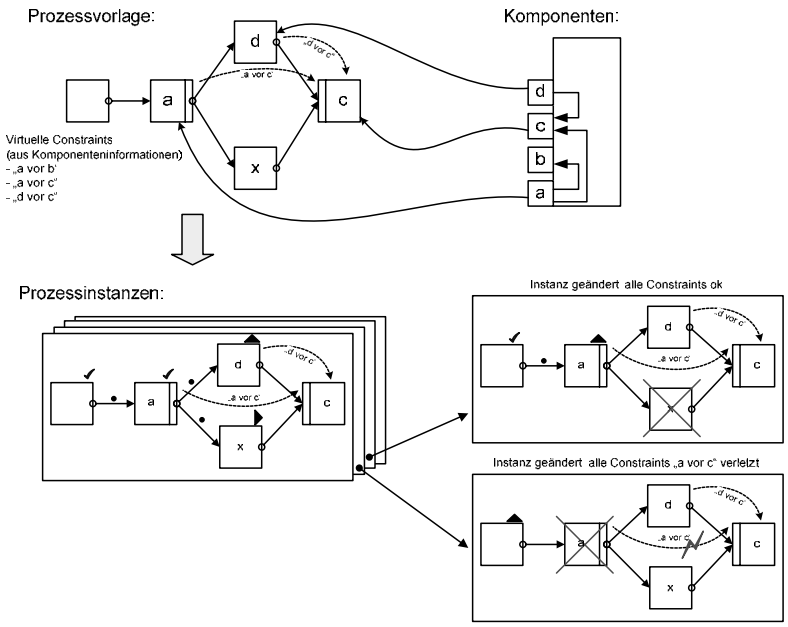

Abb. 10: Beachtung von Komponentenbeschränkungen bei Workflow-Änderungen

Entsprechende Überprüfungen werden vor Anwendung einer Änderungsoperation durchgeführt. Darüber hinaus wird nach der strukturellen Abänderung eines Ausführungsgraphen sein Status neu bewertet und ggf. angepasst, z.B. durch die Zuordnung von Zustandsmarkierungen zu neu eingefügten Aktivitäten. Anschließend wird mit der Prozessausführung in einem konsistenten Zustand fortgefahren.

Auch Schemaänderungen auf Prozesstypeebene (d.h. Änderungen am Geschäftsprozess) lassen sich im ADEPT-System rasch und korrekt umsetzen, etwa wenn das prozessorientierte Anwendungssystem an geänderte gesetzliche oder betriebliche Rahmenbedingungen adaptiert werden muss. In jedem Fall können die auf dem alten Prozessschema basierenden Instanzen ungestört zu Ende geführt werden, während – parallel dazu – neue Instanzen auf Grundlage des neuen Schemas erzeugt und ausgeführt werden können. D.h. die Koexistenz von Prozessinstanzen alter und neuer Form ist möglich. Darüber hinaus können solche Schemaänderungen – soweit gewünscht und möglich – automatisch an bereits laufende Prozessinstanzen propagiert werden (siehe Abb. 9) [RRD04a]. Diese Möglichkeit besteht selbst dann, wenn Instanzen in ihrem Verlauf zuvor individuell modifiziert worden sind (siehe Abb. 11). Bei einer solchen Prozessschema-Evolution werden wieder, wie schon bei einzelfallbezogenen Prozessänderungen, Änderungspropagationen nur dann vollzogen, wenn sie die korrekte und konsistente Ausführbarkeit der Prozessinstanz nicht verletzen. Kann eine Instanz nicht auf das neue Schema migriert werden, so wird sie auf dem Originalschema belassen. Der Prozessmodellierer kann dann von Fall zu Fall entscheiden, ob er die betreffende Instanz ggf. noch manuell anpassen oder unverändert lassen will.

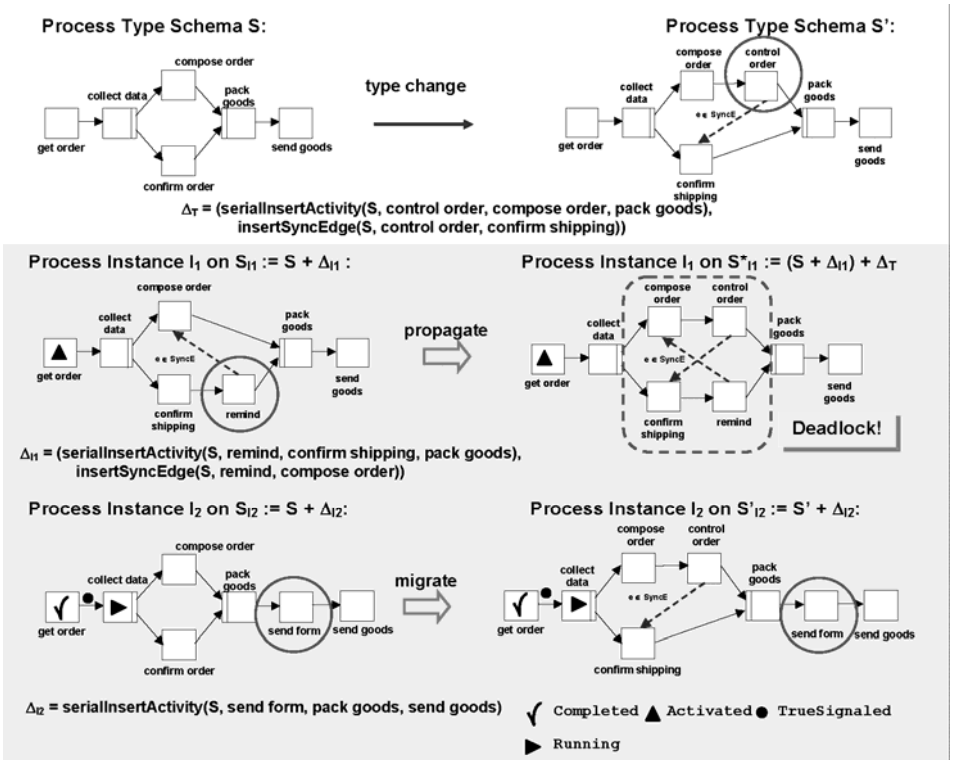

Abb. 11: Propagierung einer Schemaänderung auf individuell geänderte Instanzen

Die Evaluation der Aristaflow-Konzepte und -Software selbst wird anhand konkreter Anwendungsszenarien unserer Projektpartner[3] erfolgen. Am Beispiel der Anwendungsdomäne „Krankenhaus“ sollen insbesondere prozessorientierte Informationssysteme in Verbindung mit Ad-hoc-Abweichungen untersucht werden. Im Bereich der Automobilindustrie werden u.a. nachträgliche Änderungen am Prozess („Prozess-Schema-Evolution“) näher betrachtet. Der Bereich Energiewirtschaft wird als Muster für eine Branchensoftware und die kundenspezifische Konfiguration dieser Software dienen.

[3] Am AristaFlow-Projekt sind neben den Universitäten Ulm und Mannheim die folgenden Industriepartner beteiligt: DaimlerChrysler AG, SAP AG, All-for-One Systemhaus AG und Wilken GmbH.

5 Zusammenfassung

In vielen Anwendungsbereichen gewinnt die Realisierung prozessorientierter Anwendungen sowie die Anpassung von Prozessen im laufenden Betrieb immer mehr an Bedeutung. Die Zukunft liegt mit Sicherheit bei solchen Systemen, welche die Entwicklung prozessorientierter Anwendungen im „Plug & Play"-Stil ermöglichen. Sie müssen es zudem erlauben, dass bei Bedarf im Einzelfall flexibel vom vorgeplanten Ablauf abgewichen werden kann. Außerdem wird es für die Zukunft unabdingbar sein, dass Änderungen am Prozessmodell (etwa infolge einer internen Reorganisation) auf bereits laufende Prozessinstanzen übertragen werden können („Prozess-Schema-Evolution"). In beiden Fällen muss dies für den Benutzer einfach handhabbar sein und darf die Robustheit des Systems sowie die Konsistenz der Daten nicht gefährden.

Im Projekt Aristaflow versuchen wir grundsätzliche Fragen des Zusammenwirkens von Prozess- und Komponententechnologie zu klären. Ziel ist der Entwurf und die Implementierung einer komponentenbasierten SW-Entwicklungsumgebung, welche im Zusammenwirken mit einem Prozess-Management-System die komponentenbasierte Entwicklung von adaptiven prozessorientierten Anwendungen ermöglicht.

Literatur

[An03] A. Andresen: Komponentebasierte Softwareentwicklung mit MDA, UML und XML, Carl Hanser, 2003

[AC04] G. Alonso, F. Casati et al.: Web Services - Concepts, Architectures and Applications, Springer, 2004

[At02] C. Atkinson, J. Bayer, C. Bunse et al: Component-based Product Line Engineering with UML, Addison-Wesley, 2002

[BKK04] S. Bassil. R. Keller, P. Kropf: A workflow-oriented system architecture for the management of container transportation. Proc. BPM'04, Potsdam, 2004, S. 116-131

[BI04] Microsoft BizTalk Server 2004, http://www.microsoft.com/biztalk (April 2004)

[DP00] S. Denninger, I. Peters: Enterprise JavaBeans, Addison-Wesley, 2000

[Du02] A. Duffner: Evaluierung aktueller Technologien zur Anwendungsintegration in Workflow-Management-Systemen, Diplomarbeit, Universität Ulm, 2002

[GT00] V. Gruhn, A. Thiel: Komponentenmodelle, Addison-Wesley, 2000

[GMS02] D. Gruntz, S. Murer, C. Szyperski: Component Software – Beyond Object-Oriented Programming – Second Edition, Addison-Wesley, ACM Press, 2002

[HG03] Y. Han, H. Geng et al.: VINCA – A Visual and Personalized Business-Level Composition Language for Chaining Web-Based Services, LNCS 2910, 2003, S. 165-177

[Kr03] D. Krotr: Abbildung virtueller Kontroll- und Datenflüsse in Workflow Management Systemen, Diplomarbeit, Universität Ulm, 2003

[MBE03] B. Medjahed, A. Bouguettaya, A. Elmagarmid: Composing Web Services on the Semantic Web, VLDB Journal 12(03):333-51 (2003)

[MGR04] R. Müller, U. Greiner, E. Rahm: AgentWork – A Workflow System Supporting Rule-based Workflow Adaptation. Data & Knowledge Engineering, 2004 (to appear)

[OYP03] B. Orriens, J. Yang, M. Papyoglou: Model Driven Service Composition, LNCS 2910, 2003, S. 75-90

[RD98] M. Reichert, P. Dadam: $ADEPT_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control. JIIS, 10(2):93-129, 1998

[Re00] M. Reichert: Dynamische Ablaufänderungen in Workflow-Management-Systemen. Dissertation, Universität Ulm, Juli 2000

[RRD03] M. Reichert, S. Rinderle, P. Dadam: On the Common Support of Workflow Type and Instance Changes Under Correctness Constraints. Proc. CoopIS'03, Catania, Italien, November 2003, S. 407-425

[RRD04a] S. Rinderle, M. Reichert, P. Dadam: Flexible Support of Team Processes by Adaptive Workflow Systems. Distributed and Parallel Databases, 16(1):91-116 (2004)

[RRD04b] S. Rinderle, M. Reichert, P. Dadam: Correctness Criteria For Dynamic Changes in Workflow Systems – A Survey. Data and Knowledge Engineering, 50(1):9-34 (2004)

[SM03] K. Sivashanmugam, J.Miller: Framework for Semantic Web Process Composition, Technical Report 03-008, LSDIS Lab, University of Georgia, Juni 2003

[Tu01] K. Turowski: Spezifikation und Standardisierung von Fachkomponenten. Wirtschafts-informatik, 43(3):269-82 (2001)

[WK99] J. Warmer, A. Kleppe: OCL: The Object Constraint Language for the UML. JOOP, 12(2):10-13.

[WS04] IBM WebSphere Business Integration Server Foundation Process Choreographer, www-106.ibm.com/developerworks/websphere/zones/was/wpc.html (April 2004)