Workshop auf der D-CSCW'98

Flexibilität durch Selbstorganisation in Workflow-Systemen

Stefan Schreyjak

Stefan.Schreyjak@informatik.uni-stuttgart.de

29. September 1998

Veröffentlicht in:
SIEBERT, Reiner ; WESKE, Matthias (Hrsg.): *Workshop im Rahmen der D-CSCW'98: Flexibilität und Kooperation in Workflow-Management-Systemen*, Universität Münster, Angewandte Mathematik und Informatik, Bericht Nr. 18/98-I.
`http://www.informatik.uni-stuttgart.de/ipvr/veranstaltungen/DCSCW_WS/CfP.html`

Flexibilität ist eine wichtige Eigenschaft in Workflow-Systemen. Dieser Begriff wird anhand eines Schichtenmodells näher erläutert, das den Grad an Flexibilität über die Änderbarkeit und Erweiterbarkeit von Daten bestimmbar macht. Verschiedene Mechanismen zur Steigerung der Flexibilität werden im Kontext dieses Schichtenmodells erläutert. Der Flexibilitätsaspekt kann auch durch eine Sichtweise auf die Einsatzphasen eines Workflow-Systems beschrieben werden. Dazu wird ein Phasenmodell vorgestellt, in dem mehrere Zyklen zur Erreichung von Flexibilität vorhanden sind. Durch die Nutzung eines komponenten-orientierten Systems zur Erstellung und Anpassung von Anwendungsprogrammen durch den Benutzer ergeben sich Probleme, die durch die Einführung eines zusätzlichen Selbstorganisationszyklus gelöst werden können. In diesem Zyklus kann der Benutzer kompositive Anwendungen auf ähnliche Weise verändern wie das Modell eines Workflows.

Schlüsselwörter: Anpassung, Flexibilität, Komponente, komponenten-orientiertes System, Selbstorganisation, Workflow-Management-System

1 Einleitung

Der Trend zur zunehmenden Globalisierung [Ulr98] prägt die Wirtschaft immer mehr. Die Hauptakteure der sich globalisierenden Wirtschaft, die Unternehmen, transformieren dabei ihre internen Strukturen und ihre Beziehungen zu Kunden und Lieferanten. In den daraus resultierenden hochaktiven, entgrenzten Unternehmen ist Wandel die einzige Konstante. Daraus ergibt sich die Konsequenz, mit Ambivalenzen, Ambiguitäten, Unsicherheiten und Überraschungen umgehen zu können. Anpassungsfähige, sich selbst organisierende Unternehmen können die aus der Globalisierung entstehenden Anforderungen erfüllen.

Workflow-Management [JBS97] ist eine Technologie, die solche Unternehmen informationstechnisch unterstützt. Die Organisation von Unternehmen in Form von Geschäftsprozessen und deren Modellierung erlaubt im hohen Maße Wandelbarkeit, insbesondere verglichen mit traditionell monolithisch implementierten Programmen zur Arbeitsunterstützung. Die Kräfte zur Realisierung des Wandels müssen dabei nicht von außen kommen, sondern können vom Unternehmen selbst aufgebracht werden, indem Mitarbeiter des Unternehmens die Modellierung übernehmen. Anpassungsfähigkeit und Selbstorganisation sind daher wichtige Anforderungen an ein modernes Workflow-System.

Im folgenden Abschnitt wird zunächst der Begriff der Flexibilität näher erläutert und dazu ein Schichtenmodell vorgestellt, anhand dessen die Flexibilität eines Systems in grobe Klassen eingeteilt werden kann. Verschiedene Mechanismen zur Steigerung der Flexibilität im Kontext dieses Schichtenmodells werden erläutert. Nach dieser datenbezogenen Sichtweise auf den Flexibilitätsaspekt wird eine phasenbezogene Sichtweise vorgestellt. Der Einsatz eines Workflow-Systems wird mit einem Vorgangsmodell beschrieben und darin werden mehrere Zyklen zur Realisierung von Flexibilität identifiziert. Anschließend werden Probleme aufgezeigt, die sich aus der Nutzung eines Systems zur Erstellung und Anpassung der Anwendungsprogramme in den Aktivitäten durch den Benutzer ergeben und nicht in den vorhandenen Zyklen berücksichtigt werden. Um diese Probleme zu lösen, wird ein zusätzlicher Selbstorganisationszyklus eingeführt. In diesem Zyklus kann ein Workflow-System-Benutzer Anwendungsprogramme durch die Verwendung vorgefertigter Komponenten erstellen und verändern. Mit einer Zusammenfassung endet der Artikel.

2 Flexibilität und Selbstorganisation

Flexibilität charakterisiert nach dem Webster Dictionary[1] das Vorhandensein der Fähigkeit, sich an neue, unterschiedliche oder sich verändernde Anforderungen anpassen zu können. Ein *anpassungsfähiges* System paßt sich aktiv an, d. h. aus sich heraus, während ein *anpaßbares* System von außen, d. h. passiv, angepaßt werden kann. Der Begriff *Selbstorganisation* [KW90] beschreibt ein Organisationskonzept, bei der die Koordination der Arbeit durch ihre autonom kooperierenden Bearbeiter selbst erfolgt. Selbstorganisation steht somit im Gegensatz zu einer hierarchisch organisierten Weisungsstruktur. Da der Benutzer zur Organisation Anpassungsarbeiten durchführen muß, setzt Selbstorganisation Flexibilität des Systems voraus.

Flexibilität in einem System bekommt man nicht umsonst. Sie muß schon beim Systementwurf berücksichtigt und explizit eingebaut werden. Flexibilität wird im wesentlichen durch die zwei Faktoren Abänderbarkeit und Erweiterbarkeit bestimmt [FC96]. *Abänderbarkeit* beschreibt den Grad der Leichtigkeit, mit der eine gewisse Fähigkeit eines Systems in eine neue Variante verändert werden kann. Die Ergänzung um neue

[1] *"flexibility: characterized by a ready capability to adapt to new, different, or changing requirements"*

Fähigkeiten ist hierin nicht enthalten. Ein Beispiel ist die Abänderung eines generischen Graphikeditors in einen Editor für eine bestimmte Diagrammart. *Erweiterbarkeit* hingegen bedeutet, daß ein System ohne großen Aufwand um Fähigkeiten erweitert werden kann. Einerseits kann das System um etwas erweitert werden, von dem es schon etwas ähnliches besitzt, wie beispielsweise ein Graphikeditor um weitere Graphikobjekte. Andererseits kann auch eine Erweiterung um etwas notwendig werden, das bisher keine Entsprechung im System besitzt. Die Unterstützung für diese erste Variante ist leichter in ein System einzubauen, da gewisse Ähnlichkeiten vorausgesetzt werden können.

2.1 Ein Schichtenmodell für Daten

In einem System können immer nur Daten abgeändert und erweitert werden. Das Systemverhalten kann nur geändert oder erweitert werden, wenn die Verhaltensbeschreibung als Datum aufgefaßt wird – also der Quelltext geändert wird.

Um die Flexibilität eines Systems genauer bestimmen zu können, werden daher die Daten des Systems in einzelne Schichten strukturiert, die dann einzeln auf die Eigenschaften Abänderbarkeit und Erweiterbarkeit untersucht werden können. Mit Hilfe dieses Schichtenmodells bekommt man ein grobes Maß für die Anpassungsfähigkeit eines Systems.

Für diesen Zweck wird hier ein Rahmenwerk eingeführt, das an den Standard für *Information Resource Dictionary Systems* (IRDS) [ISO90] angelehnt ist. Dieses Rahmenwerk kann auch auf andere Systeme, wie Datenbank-Management-Systeme, Workflow-Management-Systeme [JBS97, Kap. 6.8] oder sogar Anwendungsprogramme angewendet werden. Im IRDS-Standard sind vier Schichten definiert. Für unseren Zweck reicht es allerdings aus, die folgenden drei Schichten zu betrachten:

	DBMS	WFMS	Programme
Schicht 3	Schemabeschreibung	Workflow-Sprache	Programmiersprache
Schicht 2	Schema	Workflow-Schema	Quelltext
Schicht 1	Anwendungsdaten	Workflow-Instanz	Programminstanz

Abbildung 1: Das IRDS-Schichtenmodell bezogen auf drei Systeme

Die Daten solcher Systeme können in Schichten eingeteilt werden. Jedes benachbarte Schichtenpaar steht in einer *definiert–Typ*-Beziehung, d. h. in der übergeordneten Schicht wird ein Typ definiert, dessen Instanzen in der untergeordneten Schicht auftreten. Die Daten einer Schicht bilden ein Modell und beschreiben Teile der Realität. Jede Schicht ist eine weitere Indirektionsstufe, in der von der Beschreibung eines konkreten Modells abstrahiert wird. Das Modell der Metaschicht beschreibt vereinheitlichend mögliche Ausprägungen des darunterliegenden Modells.

Bei Datenbank-Management-Systemen beschreiben die Daten auf der untersten Schicht ein Modell der Realität aus dem Anwendungsbereich, d. h. das Datum `John Smith` steht für die reale Person John Smith. Auf der nächsthöheren Schicht beschreiben die Daten ein Datenschema, d. h. die Struktur der Anwendungsdaten. Auf der dritten Schicht findet sich ein Modell zur Beschreibung von Datenmodellen, z. B. von Entity-Relationship-Diagrammen.

Im Workflow-Management beschreiben die Daten der untersten Schicht einen laufenden Workflow, die Instanz eines Workflow-Schemas. Auf der mittleren Schicht werden die Workflow-Schemata beschrieben. Auf der obersten Schicht befindet sich das Metamodell der Workflow-Schemata, die Workflow-Sprache. Die Workflow-Instanz ist eine Instanz des Schemas, die wiederum eine Instanz der Workflow-Sprache ist, d. h. in dieser formuliert wurde.

Bei Programmen stellt die Programminstanz die Daten der untersten Schicht dar. Die Daten auf der mittleren Schicht bilden den Quelltext des Programms. Der Quelltext ist wiederum eine Ausprägung einer Programmiersprache, deren Beschreibung auf der dritten Schicht zu finden ist.

Die Daten in den einzelnen Schichten und den Systemen sind entweder fest oder veränderbar bzw. erweiterbar. Das hängt allerdings auch von der Rolle ab, die der Benutzer der Daten innehat. So ist üblicherweise das Workflow-Schema unveränderbar für einen Akteur, nicht aber für einen Workflow-Modellierer. Analog ist der Quelltext eines Programms im allgemeinen für den Benutzer unveränderbar, da er keine Programmierkenntnisse hat. Für einen Software-Entwickler ist der Quelltext aber sehr wohl veränderbar.

Je mehr Daten nun für eine bestimmte Rolle veränderbar sind, desto größer ist die Anpassungsfähigkeit des Systems. Die Flexibilität kann also durch Betrachtung der unterschiedlichen Datenschichten und Einordnung in die Kategorien fest und veränderbar in Abhängigkeit von Rollen grob bestimmt werden.

Im Kontext dieses Artikels beschränken wir uns auf die Rolle des Workflow-System-Benutzers, der Anwendungsprogramme innerhalb von Aktivitäten ausführt. Dieser Personenkreis hat typischerweise keine oder nur geringe Programmierkenntnisse.

2.2 Anwendung des Schichtenmodells

Das Schichtenmodell erlaubt existierende Konzepte zur flexibleren Gestaltung von Workflow-Systemen zu klassifizieren, indem man bestimmt, in welcher Schicht sie Abänderungen oder Erweiterungen ermöglichen.

Eine Änderung oder Erweiterung der Workflow-Instanz-Daten auf Schicht 1 nennt man *Ad-hoc-Modifikationen*. Hierbei wird eine Instanz aufgrund einer einmalig vorkommenden Ausnahmesituation durch den Benutzer geändert, um von der Vorgabe (d. h. dem Schema) genau dieses eine Mal abweichen zu können. So können z. B. workflowrelevante Daten geändert werden, um eine andere Alternative im Workflow-Kontrollfluß zu ermöglichen, oder der Benutzer bestimmt den Akteur einer Aktivität selbst und läßt dies nicht vom Workflow-System machen. Das Einfügen zusätzlicher Aktivitäten, sowie das Umlenken des Kontrollflusses sind Beispiele für Erweiterungen der Instanzdaten. Da dabei nur eine temporäre, neue Variante des Schemas erzeugt wird, haben diese Ad-hoc-Modifikationen keinen Einfluß auf die Daten der Schicht 2, also auf das Workflow-Schema.

Veränderungen im Workflow-Schema sind Aufgabe des Modellierers. Im Sinne der Selbstorganisation sollte aber auch der Benutzer — zumindest eingeschränkt — sich bewährende Ad-hoc-Verbesserungen in das Workflow-Schema einfließen lassen können.

Auf Schicht 3 erlaubt Abänderbarkeit und Erweiterbarkeit der Daten, daß z. B. neue Kontrollkonstrukte erzeugt oder neue Konzepte, wie z. B. ein erweitertes Rollenkonzept

[Meh97], eingeführt werden können. Flexibilität auf dieser Schicht erweitert das Einsatzgebiet der Workflow-Systeme, da fehlende Konzepte vom Anwender realisiert werden können.

Bei Anwendungsprogrammen sind üblicherweise die Daten der zweiten und dritten Schicht für den Benutzer fest vorgegeben. Auch die erste Schicht ist meist nur sehr eingeschränkt änderbar, häufig findet sich hier nur die Konfiguration des Programms.

2.3 Flexibilität im Workflow-Systemen

Auch über die Art und Weise wie ein Workflow-System eingesetzt wird, kann man die Flexibilität eines Workflow-Systems bestimmen. Der Einsatz vieler Workflow-Systeme kann durch einen Prozeß mit Zyklen beschrieben werden [Hei94]. Der Einsatzprozeß ist Teil eines Vorgehensmodells zur Entwicklung von Workflow-Anwendungen [JBS97, Kap. 10.2]. Abbildung 2 zeigt einen solchen Prozeß. Jedes Oval stellt eine Einsatzphase dar. Im Oval sind links oben die Eingangsdaten und rechts oben die Ausgangsdaten angegeben. Es wird zwischen Workflow-Instanzen und Workflow-Schemata unterschieden. Unter dem Oval sind die ausführenden Rollen angegeben.

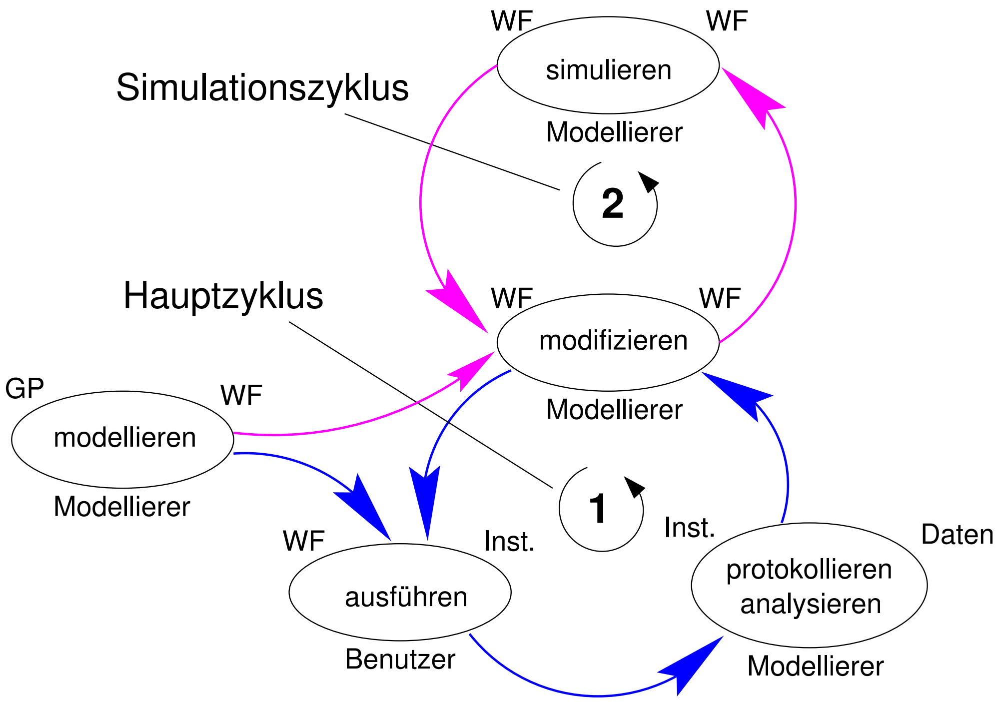

Abbildung 2: Der Einsatzprozeß eines Workflow-Systems (frei nach [Hei94])

Nach dem *Business Process Reengineering* [Ham90] im Unternehmen werden die identifizierten Geschäftsprozesse als Workflows modelliert. Während der Ausführung dieser Workflows werden Laufzeitdaten protokolliert, die in einer anschließenden Analysephase ausgewertet werden können. Die Ergebnisse der Analyse dienen dann als Entscheidungsgrundlage für die Veränderungen des Workflows. Indem diese Phasen in einem Zyklus mehrmals durchlaufen werden, können die Geschäftsprozesse sukzessive optimiert werden. Neben diesem Hauptzyklus gibt es manchmal noch einen weiteren Zyklus, in der Abbildung mit 2 markiert, in dem Laufzeitdaten einer simulierten Workflow-Ausführung zur Optimierung der Workflow-Schemata verwendet werden.

Mit der zunehmenden Verbreitung von Workflow-Systemen zeigte sich, daß diese Zyklen nicht ausreichend sind. Sie unterstützen zwar die Ausführung und Optimierung von Workflows, erlauben aber nur geringe Flexibilität im Betrieb. In vielen Anwendungsgebieten sind Geschäftsprozesse *a priori* nicht immer präzise beschreibbar [RD98]. Es wird häufig notwendig, vom normalen Ablauf abzuweichen. Wann abgewichen werden muß, wird aber erst zur Laufzeit bekannt.

Eine Lösung für dieses Problem stellt die Einführung eines zusätzlichen Zyklus im Einsatzprozeß eines Workflow-Systems dar. In einem *Ad-hoc-Zyklus* kann der Benutzer zur Laufzeit, durch gewisse Rechte eingeschränkt, Modifikation an der Workflow-Instanz vornehmen. Die Veränderungen werden nur für diesen einen Zweck durchgeführt und wirken sich nicht auf das Workflow-Schema aus. Mittlerweile finden sich auch erste Ansätze für die Unterstützung des Ad-hoc-Zyklus in kommerziellen Systemen, z. B. InConcert [Oly96] oder TeamWARE Flow [Tea97].

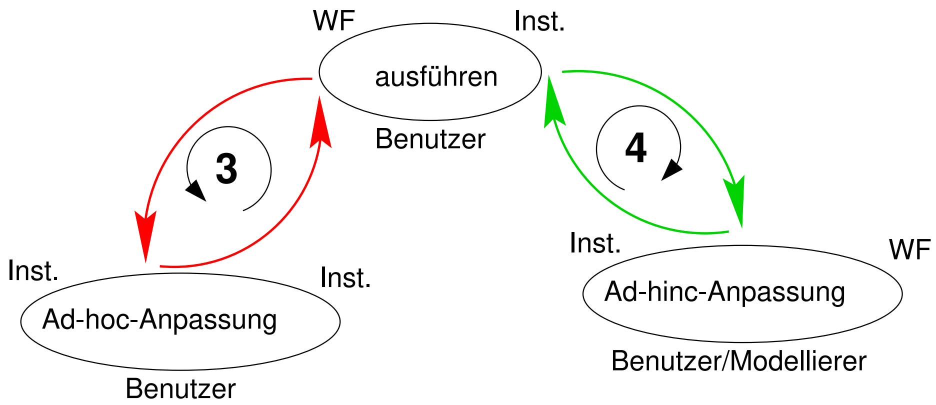

Abbildung 3: Erweiterung des Phasenprozesses um einen Ad-hoc-Zyklus (3) und einen Ad-hinc-Zyklus (4)

3 Problemstellung

Der Ad-hoc-Zyklus bietet leider keine direkte, sondern lediglich eine indirekte Unterstützung für die Selbstorganisation. So kann der Benutzer selbst zwar keine Workflow-Schemata ändern, aber er kann Änderungen über die im Unternehmen bereitstehenden Modellierer bewirken. Dann werden alle künftigen Instanzen mit dieser Änderung durchgeführt. Auch die Übernahme einer häufig durchgeführten Ad-hoc-Modifikation für eine eingegrenzte Menge an künftigen Instanzen kann durch eine Abspaltung einer Schemavariante geschehen. In traditionellen Workflow-Systemen, d. h. unter Ausgrenzung der Anwendungsprogramme, ist daher kein weiterer Zyklus für die Selbstorganisation nötig.

Im Workflow-Management wird traditionell davon ausgegangen, daß die in den Aktivitäten benutzten Anwendungsprogramme für den Benutzer vorgegeben und unveränderbar sind. Programme werden z. B. über Wrapping-Technologien [VK96] ins

Workflow-System integriert. Flexibilität äußert sich nur in der Auswahl vorhandener Anwendungsprogramme. Diese Art der Flexibilität ist allerdings nicht ausreichend für die Selbstorganisation. Was soll der Benutzer tun, wenn sich die Anforderungen im Unternehmen geringfügig ändern und nicht nur Auswirkungen auf den Prozeß, sondern auch auf die benutzten Anwendungsprogramme haben? Das erste Problem ist also, daß Anwendungsprogramme weder vom Benutzer noch vom Modellierer erstellt oder angepaßt werden können. Selbstorganisation, die auch die Anwendungsprogramme umfaßt, ist nicht möglich.

Ein zweites Problem liegt darin, daß in vielen Workflow-Systemen nur zwei Möglichkeiten bestehen, die Reichweite einer Anpassung zu bestimmen. Im ersten Fall findet die Anpassung im Ad-hoc-Zyklus statt. Die Reichweite ist dann zeitlich und räumlich auf eine Instanz beschränkt. Im anderen Fall wird im Hauptzyklus das Schema verändert. Die Anpassung wird global gültig und gilt ab sofort für alle künftigen Instanzen. Diese zweiwertige Reichweite für Anpassungen ist unbefriedigend. Bei einer Workflow-Änderung stellt sich die Frage, ob und wann die Änderung auf bereits laufende Workflow-Instanzen desselben Schemas propagiert werden soll. In [LOL98] wird dieses Problem mit der Einführung einer Sprache gelöst, mit der die Vorgehensweise bei der Versionsüberführung beschrieben wird. Bei einer Anwendungsprogrammänderung vergrößert sich das Problem noch, da es hier nötig wird, die Reichweite auch noch bearbeiterspezifisch festlegen zu können. Nur eine Programminstanz anzupassen, ist wenig sinnvoll, da die Änderung mit dem Ende der Aktivität verloren geht. Eine Anpassung sofort global bekanntzugeben, ist dagegen etwas leichtsinnig, da die Änderung noch nicht getestet ist und da sie unter Umständen nicht für alle Bearbeiter oder Workflows gleich gut geeignet ist. Die schrittweise Vergrößerung der Reichweite je nach Erfolg bzw. Anwendungszweck ist wünschenswert. Der Benutzer sollte eine differenzierte Wahl treffen können, für wen, wann und wo die von ihm durchgeführten Anpassungen sichtbar sein sollen. Eine Alternative ist, für jede Anpassung eines Benutzers eine eigene Programmvariante zu erstellen und diese dann entsprechend im Workflow-Schema aufzurufen. Dies würde aber bei anhaltender Selbstorganisation zu einer Explosion der Anzahl der Anwendungsprogramme führen, die nur noch schwer zu verwalten wären.

4 Lösungsansatz

4.1 Ad-hinc-Zyklus für inkrementelles Entwickeln

Die beschriebenen Probleme zeigen, daß Selbstorganisation unter Einbezug der Anwendungsprogramme eine neue Art der Anpassung nötig macht. Zur Repräsentation dieser neuen Anpassungsart wird ein zusätzlicher Zyklus eingeführt, der *Ad-hinc-Zyklus* oder *Selbstorganisationszyklus* (siehe Abbildung 3) genannt wird. In der Ad-hinc-Anpassungsphase können Anwendungen im Sinne der Selbstorganisation erstellt und angepaßt werden. Innerhalb dieses neuen Zyklus findet Selbstorganisation nicht nur im Rahmen des ganzen Unternehmens, sondern durch jeden Benutzer statt. Es wird kein umfassendes Neudefinieren der Prozesse betrieben, sondern kleine Verbesserungen durchgeführt. *Ad hinc* („von jetzt an“) soll im Gegensatz zu *ad hoc* („eigens zu diesem Zweck“) die unterschiedlichen Reichweiten der Anpassung verdeutlichen.

Ein inkrementelles Entwickeln computerunterstützter Geschäftsprozesse wird mit diesem Zyklus möglich. Ein Workflow kann zu Beginn mit funktional beschränkten Anwendungen instanziiert werden. Einzelne Aktivitäten können dabei noch von Hand oder mit geringer Computerunterstützung durchgeführt werden. Die Bearbeiter können mit dieser Workflow-Version Erfahrungen sammeln, sie probeweise verändern (Zyklus 3 und 4), und sie bei Erfolg sukzessive durch Änderungen im Workflow-Schema (Zyklus 1) und durch Anpassungen der Anwendungen (Zyklus 4) in den Geschäftsprozeß einfließen lassen. Im Laufe der Zeit kann somit ein Workflow aus einer Rohfassung in einen stark computerunterstützten, gut angepaßten Prozeß verfeinert werden. Dieses erhöhte Optimierungspotential beschleunigt die Entwicklung effizienter Geschäftsprozesse unter Mithilfe der Bearbeiter.

4.2 Komponenten-orientiertes System

Der Ad-hinc-Zyklus ist insbesondere dann sinnvoll, wenn der Benutzer Anwendungen erstellen und anpassen kann. Das Workflow-System sollte daher um ein System ergänzt werden, das dies erlaubt.

Ein *komponenten-orientiertes System* [Szy96] ermöglicht dem Benutzer, *kompositive Anwendungen* aus gegebenen Komponenten zu erzeugen oder anzupassen, die als Anwendungsprogramme in den Aktivitäten eingesetzt werden können. Der Benutzer kann Komponenten zur Laufzeit der kompositiven Anwendung einfügen, austauschen oder entfernen, ganz analog wie er Medienstücke in Verbunddokumente [OHE96] einfügen, austauschen und entfernen kann. Zusätzlich existiert eine benutzerveränderbare Beschreibung des Kontrollflusses auf den Methodenaufrufen der Komponenten. Ein komponenten-orientiertes System kann damit als ein spezialisiertes „Mini-Workflow-System“ innerhalb von Aktivitäten angesehen werden, das in manchen Aspekten eingeschränkt und in anderen erweitert wurde.

Die Anwendung des Schichtenmodells auf komponenten-orientierte Systeme zeigt die größere Flexibilität dieser Art von Programmen, da die Daten auf der untersten Schicht — also die Daten, die das Programm bilden, — vom Benutzer verändert werden können. Bei den kompositiven Anwendungen beschreiben diese Daten den Aufbau der Anwendung aus Komponenten und die Reihenfolge, in der die Komponenten benutzt werden.

Zur besseren Nutzung eines komponenten-orientierten Systems in einem Workflow-System müssen beide Systeme eng gekoppelt werden. Einige Synergieeffekte, die sich aus der Kopplung ergeben, sind in [Sch98] beschrieben. Aus Platzgründen kann hier die Software-Architektur eines solchen gekoppelten Systems nicht näher beschreiben werden, mehr Details finden sich in [Sch98].

4.3 Differenzierte Reichweitenunterstützung

Für eine differenzierte Unterstützung unterschiedlicher Reichweiten einer Anpassung müssen vom Workflow-System zusätzliche Dienste bereitgestellt werden. Diese Dienste sind Teil der Kopplung. Das Workflow-System macht *Prozeßkontexte* auf den Workflow-Schemata und dem Organisationsmodell des Unternehmens definierbar und verwaltet diese Kontexte [Sch98]. Ein Prozeßkontext besteht dabei aus einer spezifizierten Menge an Aktivitäten, die bestimmte Anpassungen gemeinsam haben sollen. Der Benutzer kann für jede Anpassung individuell auswählen, in welchen zukünftig zu bearbeitenden Aktivitäten die Anpassung sichtbar werden soll.

5 Zusammenfassung und Ausblick

Der Zwang in Unternehmen zur ständigen Anpassung an sich verändernde geschäftliche Bedingungen ist einer der Hauptgründe für die Entwicklung und den Einsatz von Workflow-Systemen. Folglich ist Flexibilität eine sehr wichtige Eigenschaft in diesen Systemen. Durch einen Vergleich von Workflow-Systemen mit allgemeinen Informationssystemen konnte ein Schichtenmodell in Anlehnung an den IRDS-Standard aufgestellt werden. Mit Hilfe dieses Modells kann die Flexibilität eines Informationssystems anhand der Eigenschaft Veränderbarkeit und Erweiterbarkeit bestimmter Daten grob bestimmt werden. Verschiedene Flexibilisierungskonzepte, wie z. B. Ad-hoc-Modifikationen, sind in das Schichtenmodell eingeordnet worden.

Die Flexibilität eines Systems kann auch durch die Art und Weise des Systemeinsatzes bestimmt werden. Dazu wurde der Einsatzprozeß eines Workflow-Systems betrachtet, der verschiedene Zyklen zur Anpassung eines Workflows enthält. Insbesondere bei der Nutzung eines komponenten-orientierten Systems zur Erstellung und Veränderung von kompositiven Anwendungen durch den Benutzer ergibt sich die Notwendigkeit für einen weiteren Anpassungszyklus. In diesem Selbstorganisationszyklus können kompositive Anwendungen vom Benutzer verändert werden.

Ein komponenten-orientiertes System kann als ein spezialisiertes Mini-Workflow-System angesehen werden, das mit dem Workflow-System kooperiert. Diese Kopplung ist z. B. für die Realisierung unterschiedlicher Reichweiten der Anpassungen sinnvoll. Der Einsatz kompositiver Anwendungen in Workflow-Aktivitäten versieht diese mit einer inneren Struktur, die ähnlich leicht wie ein Workflow zu verändern ist. Die Benutzeränderbarkeit kompositiver Anwendungen erlaubt damit eine neue Art von Flexibilität in Workflow-Systemen. Die Selbstorganisation der Mitarbeiter eines Unternehmens ist nicht mehr nur auf den Workflow beschränkt, sondern umfaßt auch die Anwendungen.

Inwieweit der Ansatz funktioniert, Technologien des Workflow-Managements auch in kompositiven Anwendungen einzusetzen, um dem Benutzer zu ermöglichen, aus vorgefertigten Komponenten Anwendungsprogramme zu erstellen und abzuändern, steht bis zu seiner Realisierung noch offen. Dazu muß der Entwurf eines gekoppelten Workflow- und komponenten-orientierten Systems weiter verfeinert und schließlich erprobt werden.

Literatur

[Oly96] Olympic Software NZ Ltd.: 1996. – `http://www.olympic.co.nz/InConcert.htm`

[ISO90] ISO: Information Resource Dictionary System Framework / ISO. 1990.

[Sch98] Schreyjak, Stefan: Using Components in Workflow Activities. **In:** Sutherland, Jeff (Hrsg.) ; Patel, Dilip (Hrsg.): *Proceedings of the Second and Third International Workshop on Business Objects (at OOPSLA'96/'97)*, Springer Verlag, 1998. – to appear

[Ham90] Hammer, Michael: Reengineering Work: Don't Automate, Obliterate. **In:** *Harvard Business Review* (1990), July–August, Nr. 4, S. 104–112

[FC96] Fayad, Mohamed ; Cline, Marshall P.: Aspects of Software Adaptability. **In:** *Communications of the ACM* 39 (1996), October, Nr. 10, S. 58–59

[KW90] Kratky, Karl W. (Hrsg.) ; Wallner, Friedrich (Hrsg.): *Grundprinzipien der Selbstorganisation*. Darmstadt : Wiss. Buchgesellschaft, 1990

[Ulr98] Ulrich Steger (Hrsg.): Wirkmuster der Globalisierung: nichts geht mehr, aber alles geht / Gottlieb Daimler- und Karl Benz-Stiftung. 1998. – Forschungsbericht

[JBS97] Jablonski, Stefan ; Böhm, Markus ; Schulze, Wolfgang: *Workflow-Management: Entwicklung von Anwendungen und Systemen*. dpunkt Verlag, 1997

[Meh97] Mehandjiev, Nikolay. User enhancability for Information Systems through Visual Programming. Disseration, September 1997

[Hei94] Heilmann, H.: Workflow Management: Integration von Organisation und Informationsverarbeitung. **In:** *HMD* 176 (1994), S. 8–21

[RD98] Reichert, M. ; Dadam, P.: Supporting Dynamic Changes of Workflows Without Loosing Control. **In:** *Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management* 10 (1998), Nr. 2

[Tea97] TeamWARE: *TeamwareFlow – Effektive Geschäftsprozesse durch kooperativen Workflow*. Juni 1997

[VK96] Valetto, Giuseppe ; Kaiser, Gail E.: Enveloping Sophisticated Tools into Process-Centered Environments. **In:** *Journal of Automated Software Engineering* 3 (1996), S. 309–345

[LOL98] Liu, Chengfei ; Orlowska, Maria E. ; Li, Hui: Automating Handover in Dynamic Workflow Environments. **In:** Pernici, Barbara (Hrsg.) ; Thanos, Costantino (Hrsg.): *Advanced Information Systems Engineering*, Springer, Juni 1998, S. 159–171

[Szy96] Szyperski, Clemens: Independently Extensible Systems - Software Engineering Potential and Challenges. **In:** *Australian Computer Science Communications* 18 (1996), Februar, Nr. 1, S. 203–212

[OHE96] Orfali, Robert ; Harkey, Dan ; Edwards, Jeri: *The Essential Distributed Objects Survival Guide*. New York, NY [u.a.] : Wiley & Sons, 1996

[Sch98] Schreyjak, Stefan: Synergies in a Coupled Workflow and Component-Oriented System. **In:** Grundy, John (Hrsg.): *Proceedings of CBISE'98 — CAiSE*98 Workshop on Component Based Information Systems Engineering*, 1998. – `http://www.cs.waikato.ac.nz/~jgrundy/caise98_workshop`. ISSN 1170–487X, S. 43–50