Universität Stuttgart
Software–Labor
Projekt 1.1:
Workflow–Management–Systeme
Breitwiesenstraße 20–22
D-70565 Stuttgart

Fakultätsbericht 1996/17
Software–Labor Bericht SL–3/96
CR–Klassifikation H.2.0, H.2.4

Fehlertolerante Abwicklung von Geschäftsprozessen in Workflow–Management–Systemen*

Stefan Schreyjak
Stefan.Schreyjak@informatik.uni-stuttgart.de
Hubert Bildstein

19. Dezember 1996

*Diese Arbeit wird von der IBM Deutschland Entwicklung GmbH und dem Ministerium für Wissenschaft und Forschung, Baden Württemberg, unterstützt.

Zusammenfassung

Die fehlertolerante Abwicklung von Geschäftsprozessen ist ein wichtiges Kriterium bei der Entwicklung von Workflowsystemen.

Dieser Bericht führt in die Problematik ein, die sich bei der Fehlerbehandlung in Workflowsystemen ergibt. Der Begriff des Fehlers wird dabei sehr weit ausgelegt und umschließt alle Ausnahmesituationen, die während eines Geschäftsprozesses auftreten können. Diese Fehlerarten werden klassifiziert. Wir stellen daraufhin Anforderungen an die Fehlerbehandlung in einem Workflowsystem auf. Anschließend werden im Überblick verschiedene Lösungsansätze vorgestellt: Durch eine Präferenzrelation auf den Aktivitäten können alternative Pfade spezifiziert werden. Ad–hoc–Modifikationen ermöglichen flexible Reaktionen auf Fehler. Durch Workflow–Transaktionen werden ACID–Eigenschaften in Workflows nutzbar. Der Lösungsansatz Kompensationssphären stellt besonders wenig Anforderungen an die Aktivitäten.

Inhaltsverzeichnis

1 Einführung und Motivation **5**
1.1 Allgemeine Anforderungen 6

2 Fehlermodell **7**
2.1 Grundlegende Begriffe 7
2.2 Fehlerbehandlungsmechanismen 8
2.2.1 Fehlerdetektion 10
2.3 Fehler in Workflow–Management–Systemen 10
2.3.1 Systemdienstebene 10
2.3.2 Workflowdienstebene 11
2.3.3 Aktivitätenebene 11

3 Funktionale Anforderungen an die Fehlerbehandlung **13**
3.1 Anforderungen an transparente Fehlerbehandlung 13
3.2 Anforderungen an automatische Fehlerbehandlung 14
3.3 Anforderungen an manuelle Fehlerbehandlung 15

4 Lösungsansätze **17**
4.1 Modellierung im Workflow 18
4.2 Alternative Aktivitäten und Pfade 19
4.3 Ad–hoc–Modifikationen 21

5 Workflow–Transaktionen **23**
5.1 Datenbank–Transaktionen 23
5.2 Datenbank–Transaktionen in Workflowsystemen 24
5.3 Erweiterte Transaktionsmodelle 27
5.4 Workflow–Transaktionen 29
5.4.1 Begriffe 29
5.4.2 Das Konzept der Workflow–Transaktionen 31
5.4.3 Anforderungen an ACID–Aktivitäten 32
5.5 Einsatzgebiete von Workflow–Transaktionen 34

6 Kompensations–Sphären **34**
6.1 Begriffe 35
6.2 Das Konzept der Kompensations–Sphären 35
6.3 Vergleich zwischen Transaktions–Sphären und Kompensations–Sphären 38

7 Ausblick und Zusammenfassung **39**
7.1 Ausblick auf Erweiterungen 39
7.2 Zusammenfassung 40

1 Einführung und Motivation

Ein *Workflow–Management–System* (WFMS) ist ein Softwaresystem zur Koordination und kooperativen Abwicklung von Geschäftsvorgängen in verteilten Rechnerumgebungen. Geschäftsvorgänge können formal in Geschäftsprozessen beschrieben und modelliert werden. Ein *Geschäftsprozeß* ist eine geordnete Menge von Vorgangsschritten in Verbindung mit dem organisatorischen Umfeld des Unternehmens. In einem Geschäftsprozeß wird in der Hauptsache spezifiziert, in welcher Reihenfolge die Vorgangsschritte ausgeführt werden, welche Arbeitsobjekte in den einzelnen Schritten bearbeitet werden und welche menschlichen, technischen und organisatorischen Ressourcen zur Ausführung benötigt werden. Das Ziel eines Geschäftsprozesses ist das Erreichen eines betrieblichen Ergebnisses.

Die Aufgabe eines Workflow–Management–Systems ist die Modellierung, Steuerung, Überwachung und Protokollierung von Geschäftsprozessen. Durch die Modellierung werden die Analyse und die anschließende Optimierung von Geschäftsprozessen unterstützt.

In vielen Workflow–Management–Systemen werden Geschäftsprozesse in Form eines *Aktivitätennetzes* modelliert. Eine *Aktivität* ist ein zusammenhängendes Stück Arbeit, das durch eine Person oder ein Programm ausgeführt wird und auf ein oder mehrere Arbeitsobjekte wirkt. Ein Arbeitsobjekt kann ein physisches Objekt (z. B. ein Werkstück) oder ein informationstechnisches Objekt (Daten oder ein Dokument) sein. In einer Aktivität können zur Bearbeitung des Arbeitsobjekts beliebige Anwenderprogramme eingesetzt werden. Der Zugriff dieser Programme auf Daten wird dabei nicht durch das Workflowsystem kontrolliert.

Durch die Identifikation und die anschließende Spezifikation von Geschäftsprozessen werden große Optimierungspotentiale in einem Unternehmen aufgedeckt, die durch die Neuordnung ganzer Wertschöpfungsketten ausgenutzt werden können. Dadurch werden nicht mehr nur lokale, sondern auch globale Optimierungen ermöglicht. Mit der Abkehr von abteilungsorientierten hin zu prozeßorientierten Organisationsstrukturen wird die Automatisierung von Prozessen und die Integration bestehender Computeranwendungen erleichtert. Die unternehmensweite Steuerung und die verbesserten Informationsfähigkeiten des Systems erlauben den Ablauf besser kontrollierter Vorgänge, die zu einer hohen Produktivitätssteigung führen können.

Die Einführung eines Workflow–Management–Systems in ein Unternehmen muß aber wohlüberlegt und sorgfältig geplant sein. Die wertschöpfenden Prozesse innerhalb des Unternehmens werden dadurch unter die Kontrolle des Workflow–Management–Systems gestellt. Der Erfolg des Unternehmens hängt somit direkt von der Funktionsfähigkeit des Systems ab. Wenn das System einmal nicht funktionsfähig sein sollte, kommen alle computerunterstützten Geschäftsprozesse zum Erliegen. Der mögliche Ausweg, die Prozesse kurzfristig ohne Computerunterstützung durchzuführen, ist meist nicht einfach gangbar, da es dadurch zu Inkonsistenzen zwischen den Daten im System und der Realität kommt. Nach dem Neustart des Systems kann es im allgemeinen nicht sofort wieder eingesetzt werden, da zuerst der veraltete

Datenzustand manuell auf den neuesten Stand gebracht werden muß. Durch eine schrittweise Einführung und durch den Einsatz eines fehlertoleranten und stabilen Systems kann man dieser Gefahr begegnen.

1.1 Allgemeine Anforderungen

Ein Workflow–Management–System muß als das „Rückgrat“ eines Unternehmens angesehen werden: Ein Bruch wäre tödlich.

Oberste Strategie beim Einsatz eines solchen Systems muß daher die Fehlervermeidung sein. Da in der realen Welt dieses Ziel aber nicht vollständig erreicht werden kann, benötigt man darüber hinaus Mechanismen, um auf Fehler reagieren zu können. Das ganze System muß daher folgende allgemeine Anforderungen erfüllen.

- **Korrektheit:**
  Ein System ist korrekt, wenn es die Aufgabe, für die es spezifiziert ist, erfüllt. Voraussetzung dafür ist unter anderem, daß die Integrität der Daten, die durch das System verwaltet werden, gewährleistet ist. Nur mit konsistenten Daten kann ein System korrekt arbeiten. Inkonsistente Daten können zu fehlerhaftem Verhalten führen.

  Auch beim Auftreten von Fehlern darf das System keine inkonsistenten Zustände erzeugen. Diese Forderung hat große Auswirkungen auf die Fehlerbehandlungsmechanismen des Workflowsystems.

  Diese Anforderung kann nicht alleine vom Workflowsystemen erfüllt werden, da es keine vollständige Kontrolle über die bearbeiteten Daten hat. In den Aktivitäten können Programme oder Menschen Daten außerhalb der Kontrolle des Workflowsystem inkonsistent verändern, ohne daß das System darauf Einfluß hat oder die Inkonsistenz überhaupt bemerkt.

- **Hohe Zuverlässigkeit:**
  Ein System ist hoch zuverlässig, wenn es über lange Zeiträume hinweg ohne Auftreten eines effektiven Fehlers funktioniert. Ein Fehler ist dann effektiv, wenn er die spezifizierte Funktion des Systems beeinträchtigt [GR93].

- **Hohe Verfügbarkeit:**
  Längerfristige Ausfallzeiten des Gesamtsystems verringern die Verfügbarkeit. Während dieser Zeit kann keiner der Mitarbeiter eines Unternehmens oder einer Behörde weiterarbeiten, da bei dem umfassenden Einsatz eines Workflow–Management–Systems nahezu alle Arbeiten über das System oder zumindest mit dessen Hilfe abgewickelt werden. Der Stillstand des Systems kann daher zu immensen Kosten führen. Die Verfügbarkeit kann durch hohe Zuverlässigkeit oder durch den Einsatz redundanter Komponenten erhöht werden. Durch den Einsatz von Mechanismen für einen schnellen, weitgehend automatischen Wiederanlauf des Systems (Recovery) kann auch die Verfügbarkeit erhöht werden.

- **Robustheit:**
  Ein robustes System verhält sich tolerant gegenüber unerwarteten Eingaben und bleibt auch in Ausnahmesituationen weiterhin funktionsfähig. Die Eigenschaft Robustheit trägt damit zur Erhöhung der Verfügbarkeit bei.

- **Hohe Flexibilität:**
  Ein flexibles System erlaubt auch nach dem Auftreten eines Fehlers oder einer Ausnahmesituation, die nicht automatisch durch Fehlerbehandlungsmechanismen des Systems beseitigt werden können, manuelle Eingriffe in die Kontrolle des Systems, um den Fortgang der Prozesse zu ermöglichen.

  In einem Workflowsystem werden immer auch Fehler auftreten, die sich nicht automatisch beheben lassen. Das sind zum einen Systemfehler, wie der Ausfall eines Rechnerknotens, und zum anderen Fehler, die das System nicht erkennen kann, da sie semantischer Natur sind. In einem solchen Fall ist der Benutzer auf die Flexibilität des Systems angewiesen: Der Eingriff eines Menschen ist nötig. Er muß die Kontrolle übernehmen und das System in einen Zustand überführen, in dem es die Kontrolle wieder selbst übernehmen kann. Dazu muß das System Methoden anbieten, die der Benutzer „manuell“ anwenden kann, um die Fehlersituation zu bereinigen und den Prozeß wieder in geordnete Bahnen zu lenken. Kompetente Benutzer können so unter Ausnützung ihres Fachwissens und dem Einsatz von Software–Werkzeugen die Auswirkungen von Fehlern beseitigen und so das System reparieren.

2 Fehlermodell

In diesem Abschnitt werden grundlegende Begriffe zu Fehlern und zur Fehlerbehandlung definiert und erläutert. Dabei wird besonders auf die Sichtweise des Benutzers und die Charakteristika von Workflow–Management–Systemen eingegangen.

2.1 Grundlegende Begriffe

Der Begriff des Fehlers wird in diesem Bericht sehr weit gefaßt: Alle Abweichungen vom Normalfall, d. h. vom vorspezifizierten Ablauf eines Geschäftsprozesses, fallen in den Bereich der Fehlerbehandlung. Der Begriff „Fehler“ umfaßt daher nicht nur Probleme des Workflowsystems, der beteiligten Softwaresysteme und des dazugehörigen Rechnersystems, sondern auch Probleme, die sich aus der Spezifikation des Geschäftsprozesses ableiten. Diese Probleme werden Ausnahmesituationen im Ablauf eines Workflows genannt. Im folgenden sind mit dem Begriff Fehler meist auch Ausnahmesituationen gemeint.

Nach dem Auftreten eines Fehlers ergeben sich zwei unterschiedliche Probleme: Der Fehler muß entdeckt werden (Detektion), und die Auswirkungen des Fehlers

müssen beseitigt werden (Fehlerbehandlung). Die Fehlerentdeckung ist Voraussetzung für die Fehlerbehandlung.

Fehler sind *transient*, wenn sie nur gelegentlich auftreten [GR93]. Sie sind daher nicht oder nur schwer reproduzierbar. Wenn das Programmsystem wiederholt in den Zustand gebracht wird, in dem vorher der Fehler aufgetreten war, muß der Fehler nicht zwangsläufig erneut auftreten. Das System kann daher z. B. mit einer Wiederholung der fehlgeschlagenen Funktion auf das Auftreten dieser Fehlerart reagieren. Ursachen für transiente Fehler können z. B. Timing–Probleme oder Überlastsituationen sein.

Fehler, die reproduzierbar in einem bestimmten Zustand des Programmsystems auftreten, werden *persistent* genannt. In diese Klasse gehören Programmierfehler wie die Übergabe falscher Parameter an Prozeduren oder Prozeduren, die von ihrem spezifizierten Verhalten abweichen.

Fehler können auch durch die Reichweite ihrer Auswirkungen unterschieden werden:

- Kein Verlust des flüchtigen Speichers eines Rechners
  Dazu gehören z. B. Fehler in den Anwendungsprogrammen, Prozeßabstürze oder Kommunikationsfehler. Das Betriebssystem des Rechners ist beim Auftreten eines solchen Fehlers nicht in Mitleidenschaft gezogen. Ein Neustart des Rechners ist nicht erforderlich.

- Verlust des flüchtigen Speichers
  Transiente Rechnerknotenausfälle gehören in diese Klasse. Ein Auftreten eines solchen Fehlers zieht auch das Betriebssystem in Mitleidenschaft. Ein Neustart des Rechners ist erforderlich.

- Verlust des persistenten Speichers
  Unwiederbringlicher Verlust von Daten durch Zerstörung eines Datenspeichers.

2.2 Fehlerbehandlungsmechanismen

Der Benutzer will so wenig wie möglich mit den Auswirkungen von Fehlern und dem Problem der Fehlerbehandlung konfrontiert werden. Ein Hauptanliegen der Fehlerbehandlung ist somit, daß auftretende Fehler möglichst schnell erkannt und ihre Auswirkungen beseitigt werden. Ein Eingreifen des Benutzers soll nicht notwendig werden. Dieses Ziel ist offensichtlich nicht immer zu erreichen. Wir definieren daher folgende drei Stufen von Fehlerbehandlungsmechanismen, die sich in der Art unterscheiden, wie das System aufgetretene Fehler behandelt und wie diese Behandlung durch den Systembenutzer wahrgenommen wird:

1. **Transparente Fehlerbehandlung**

   Die höchste Stufe ist die *transparente Fehlerbehandlung*. Diese liegt vor, wenn auftretende Fehler automatisch erkannt und behandelt werden, so daß sie ohne Auswirkungen auf die vom System zu leistende Arbeit bleiben und keinen

Benutzereingriff erfordern. Die Fehlerbehandlung des Systems ist durch den Benutzer nicht beobachtbar, sie ist maskiert.

Ein typisches Beispiel für eine transparente Fehlerbehandlung ist das Auftreten eines transienten Netzwerkfehlers, z. B. einen vorübergehenden Ausfall einer Kommunikationsverbindung. Der Benutzer bemerkt dabei höchstens eine Verzögerung des Fortgangs, aber keine Abweichung vom normalen Ablauf. Fehler dieser Art werden häufig schon auf einer tiefer liegenden Systemschicht, z. B. dem Betriebssystem, behandelt.

Vollständige Fehlertransparenz ist das Ideal des Benutzers. Das System erscheint ihm fehlerfrei.

2. **Automatische Fehlerbehandlung**

   Die nächste Stufe innerhalb dieser Klassifikation ist die *automatische Fehlerbehandlung*. Das System behandelt hier den Fehler ohne zusätzlichen Eingriff des Benutzers. Im Gegensatz zur transparenten Fehlerbehandlung wird aber das Auftreten eines Fehlers und die darauf folgende Fehlerbehandlung gegenüber dem Benutzer nicht maskiert. Die Fehlerbehandlung bleibt durch den Benutzer beobachtbar. Automatische Fehlerbehandlung bedeutet nicht, daß auch die Fehlerdetektion automatisch erfolgen muß. Der Benutzer kann die automatische Fehlerbehandlung auch manuell anstoßen.

   Das System besitzt Mechanismen, um die Auswirkungen des Fehlers so weit zu minimieren, daß nach Auftreten des Fehlers wieder ein korrekter Zustand erreicht werden kann, von dem aus die Bearbeitung sinnvoll fortgesetzt wird. Das Auftreten des Fehlers erzwingt somit einen Zustandswechsel, der vom Benutzer wahrgenommen werden kann.

   Ein Beispiel dazu ist ein System, das einen Fehler innerhalb einer bestimmten Aktion dadurch behandelt, daß die bisher durch diese Aktion bewirkten Zustandsänderungen rückgängig gemacht werden und danach eine alternative Vorgehensweise gewählt wird. Der Benutzer muß in diesem Fall nicht in die Fehlerbehandlung eingreifen, hat aber keine Transparenz, da eine Alternative eine Abweichung vom Normalablauf darstellt.

3. **Manuelle Fehlerbehandlung**

   In der dritten Stufe, der *manuellen Fehlerbehandlung*, kann oder soll das System die Fehlerbehandlung nicht selbst übernehmen. Das System stellt als einzige Unterstützung dem Benutzer Werkzeuge zur Verfügung, damit dieser die Fehlerbehandlung selbst durchführen kann. Es gibt hier keine automatisch anlaufende Fehlerbehandlung.

   In vielen Fällen muß auf den Ausfall von Hardware mit dieser Art der Fehlerbehandlung reagiert werden. Auch das Beseitigen von Programmierfehlern fällt in diese Kategorie. Manuelle Fehlerbehandlung sollte immer als letzter

Ausweg zur Beseitigung einer Fehlersituation verfügbar sein und auch durch entsprechende Werkzeuge unterstützt werden.

2.2.1 Fehlerdetektion

Transparente Fehlerbehandlung setzt die automatische Detektion des Fehlers durch das System voraus. Bei den beiden anderen Klassen wird dies nicht unbedingt vorausgesetzt. Hier kann man zwischen *automatischer* und *manueller Fehlererkennung* unterscheiden. Letzteres bedeutet, daß der Benutzer die Fehlerbehandlung selbst initiieren kann oder muß. Die Behandlung kann aber wieder vom System durchgeführt werden.

Notwendig für die Fehlererkennung ist die Definition eines korrekten Systemzustands. Eine Definition kann über eine Menge von Rahmenbedingungen (constraints) auf dem Systemzustand erfolgen. Die Verletzung einer solchen Bedingung, d. h. ein Abweichen vom korrekten Systemzustand, wird dann als Fehler bezeichnet.

2.3 Fehler in Workflow–Management–Systemen

Es gilt nun festzulegen, welche Arten von Fehlern aus Sicht des Workflowsystems relevant sind. Wir ordnen dazu die Fehler nach dem Ort ihres Auftretens in mehrere Fehlerebenen ein. Ein Fehler sollte möglichst von den Fehlerbehandlungsmechanismen in der Ebene behandelt werden, in der er aufgetreten ist. Nur wenn dies nicht möglich ist, kann und muß der Fehler eine Ebene höher gereicht werden.

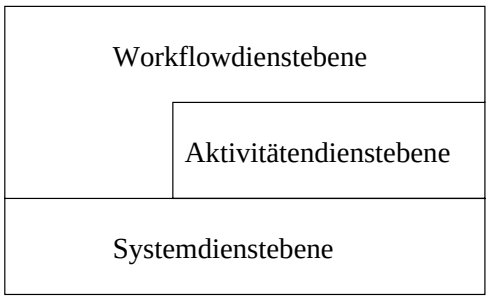


Abbildung 1: Die Fehlerebenen in einen Workflow–Management–System

2.3.1 Systemdienstebene

Diese Ebene stellt die grundlegende Funktionalität zum Ablauf eines Workflow–Management–Systems bereit [EL95]. Hier sind z. B. das Betriebssystem, das Kommunikationssystem (Netzwerksoftware) oder ein Datenbank–Management–System zu finden. Die hier auftretenden Fehler werden typischerweise entweder schon auf dieser Ebene behandelt und haben somit keinen Einfluß auf das Workflow–Management–System, oder führen zu einem (zumindest partiellen) Neuanlauf des Workflowsystems

mit anschließendem Recovery, wenn der Fehler auf die Workflowdienstebene weitergereicht wird.

Das Workflowsystem muß daher auf das Auftreten eines weitergereichten Systemfehlers vorbereitet sein. Alle zum Recovery des Workflowsystems notwendigen Daten müssen auf sicherem Speicher geschrieben werden.

Typische Beispiele für Fehler auf der Systemdienstebene sind Verbindungsabbrüche, das Auftreten von Verklemmungen (Deadlocks), Betriebssystemausfälle oder Hardwaredefekte. Die angewendeten Fehlerbehandlungsmechanismen basieren häufig auf bekannten Techniken, wie das Wiederholen fehlgeschlagener Funktionen, das Verwenden abgesicherter Übertragungsprotokolle oder das Prinzip des Rollbacks in Datenbank–Management–Systemen. Durch die Verwendung replizierter Software– und Hardwarekomponenten kann ein Teil der Fehler auf dieser Ebene abgefangen bzw. gemildert werden.

2.3.2 Workflowdienstebene

Auf dieser Ebene befindet sich die eigentliche Funktionalität des Workflowsystems. Fehler dieser Ebene haben einen direkten Einfluß auf das Workflowsystem und müssen vom Workflowsystem behandelt werden. Beispiele für Fehler aus dieser Ebene sind Fehler in der Modellierung oder Ausführung eines Geschäftsprozesses.

Die Fehler bzw. Ausnahmesituationen dieser Ebene können in zwei Klassen eingeteilt werden.

- *Vorhersehbare Fehler* sind Abweichungen vom normalen Ablauf des Workflows. Da der Ort des Auftretens dieser Fehler vorhersagbar ist, kann mit dem Mittel der Geschäftsprozeß–Modellierung auf die Fehler reagiert werden. Als Beispiel kann auf den Fehlschlag einer Aktivität eine alternative Aktivität gestartet werden oder die Aktivität wiederholt werden.

- *Unvorhergesehene Fehler* können aufgrund ihrer Eigenschaft nicht schon zur Spezifikationszeit im Prozeß modelliert werden. Sie treten dann auf, wenn veränderte Randbedingungen oder bis dahin unbekannte oder neue Anforderungen auf einen gestarteten Prozeß treffen. Ein Beispiel für einen unvorhergesehenen Fehler kann das Fehlen einer passenden Alternative sein.

Wenn Fehler in der Systemdienstebene oder in der Aktivitätenebene nicht behebbar waren, werden sie an die Workflowdienstebene weitergereicht. Als oberste Schicht muß das Workflowsystem auf diese Fehler reagieren. Weitergereichte Fehler können entweder vorhersagbar oder nicht vorhersagbar sein.

2.3.3 Aktivitätenebene

Hier finden sich die Fehler, die bei der Ausführung von Aktivitäten in deren Kontrollbereich auftreten. Aktivitäten bestehen aus manuell oder automatisch ausgeführten Tätigkeiten.

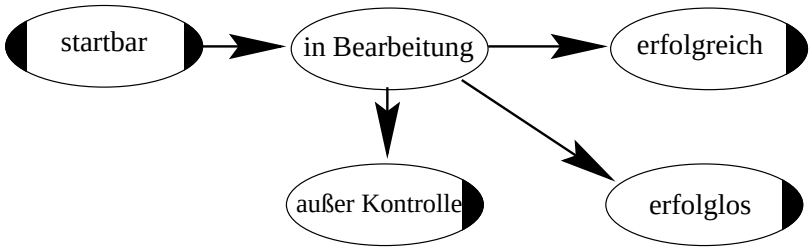


Abbildung 2: Das Zustandsübergangsdiagramm einer Aktivität

Die Bearbeitung einer Aktivität kann durch das folgende grundlegende Zustandsübergangsdiagramm modelliert werden (siehe Abbildung 2). Das Diagramm ist für diesen Bericht vereinfacht worden. Reale Implementierungen von Workflowsystemen haben meistens komplexere Übergangsdiagramme. Ein Startzustand ist am linken Rand schwarz markiert. Ein Endzustand ist rechten Rand markiert.

Eine Aktivität beginnt mit dem Initialzustand STARTBAR. Der Start durch den Bearbeiter überführt die Aktivität in den Zustand IN BEARBEITUNG. Dort können Fehler auftreten, die auf der Aktivitätenebene angesiedelt sind. Wenn ein solcher Fehler erkannt wird, wird die Aktivität in den Zustand ERFOLGLOS gebracht. Auf diese Art bekommt das Workflowsystem Kenntnis vom Auftreten eines Aktivitätenfehlers und ein Fehlerkontext kann von der Aktivität an das Workflowsystem übergeben werden. Kann eine Aktivität ohne Auftreten eines Fehlers beendet werden, kommt sie in den Zustand ERFOLGREICH. Eine Aktivität wird in den zusätzlichen Zustand AUSSER KONTROLLE gebracht, wenn die Aktivität weder eine erfolgreiche, noch eine erfolglose Bearbeitung melden kann. In diesem Fall ist die Aktivität außer Kontrolle geraten. Problematisch ist hierbei die Detektion des Übergangs IN BEARBEITUNG nach AUSSER KONTROLLE. In diesem Fall muß das Workflowsystem ohne Hilfe der Aktivität erkennen, daß ein Aktivitätenfehler aufgetreten ist. Voraussetzung für ein fehlertolerantes Workflowsystem ist, daß das System auch dann sinnvoll weiterarbeiten kann, wenn einmal dieser Zustand auftritt.

Folgende Fehlersituationen sind Beispiele für Fehler auf der Aktivitätenebene:

- Ein Anwendungsprogramm in einer Aktivität kann aufgrund einer mangelhaften Installation des Systems nicht gefunden werden.
- Eine Aktivität meldet fehlerhafte Ausführung aufgrund des Fehlschlagens einer wichtigen Operation (z. B. Datei existiert nicht) innerhalb des Anwendungsprogramms. Der Fehler kann möglicherweise transient sein.
- Eine Aktivität meldet sich nicht mehr, da sie aufgrund eines Programmierfehlers abgestürzt ist.
- Eine Aktivität braucht sehr lange — zu lange! Diese Fehlerart muß vom oben erwähnten Absturz unterschieden werden, da sich die Aktivität aus der Sicht des

Workflowsystems im ordnungsgemäßen Zustand IN BEARBEITUNG befindet. Es kann außerhalb der Aktivität nicht entscheidbar, ob in der Aktivität ein Fehler aufgetreten ist. Nur aufgrund der überlangen Bearbeitungszeit wird eine fehlerhafte Bearbeitung vermutet. Diese Fehlerart kann vom Workflowsystem z. B. über eine Zeitüberwachung der Aktivität detektiert werden. Das System muß deshalb darauf reagieren, da in der Aktivität Ressourcen belegt und damit eventuell andere Anwendungsprogramme blockiert sein können. Ebenso können nachfolgende Aktivitäten vor Beendigung dieser Aktivität nicht gestartet werden. Die Ursache für diese Fehler kann in einer programmierten Endlosschleife (Programmierfehler) oder einer nicht auslösbaren Wartesituation (Deadlock) liegen, oder einfach daran, daß die Bearbeitung sehr viel länger als erwartet dauert.

3 Funktionale Anforderungen an die Fehlerbehandlung

Im erstem Kapitel wurden bereits allgemein gehaltene Anforderungen an ein Workflowsystem vorgestellt. In diesem Kapitel werden diese Anforderungen in funktionale Anforderungen verfeinert und so spezialisiert, daß sie die Bedürfnisse eines Workflowsystems in Bezug auf die Fehlerbehandlung darstellen. Mit den funktionalen Anforderungen soll beschrieben werden, welche Funktionen der Benutzer eines Workflowsystems von der Fehlerbehandlung erwartet. Wie diese Anforderungen realisiert werden können, wird in dem Kapitel über Lösungsansätze beschrieben.

Die Anforderungen sind nach den im Kapitel Fehlermodell definierten Stufen: *transparent*, *automatisch* und *manuell* geordnet. Bei der Betrachtung der Fehlerbehandlung in Workflowsystemen sind zwei Benutzertypen zu unterscheiden. Zum einen gibt es den Workflowmodellierer, der sich zur Modellierungszeit mit der Fehlerbehandlung von Workflows auseinanderzusetzen hat. Zum anderen gibt es den normalen Benutzer, der die Aktivitäten bearbeitet, und der die Auswirkungen von Fehlern sieht.

3.1 Anforderungen an transparente Fehlerbehandlung

Folgende Anforderungen betreffen Fehlerbehandlungsfunktionen, die automatisch vom System beim Auftreten von Fehlern oder beim Versagen einer Workflow–Komponenten ausgeführt werden, ohne daß der Benutzer etwas davon mitbekommt. Die laufenden Workflows dürfen daher während der Fehlerbehandlungsphase und in der anschließenden Wiederanlaufphase nur blockiert sein und müssen danach wieder normal weiterlaufen. Diese Eigenschaft wird auch häufig mit dem Begriff *forward–recovery* bezeichnet. Die Arbeitsergebnisse bzw. auch die Ergebnisse von Zwischenschritten sollen Fehler des Systems dauerhaft überstehen können.

Anforderung: Persistenter Workflow–Zustand

Der Zustand des Workflowsystems darf auch beim Versagen einer Workflowsystem–Komponente oder beim Auftreten eines Fehlers in der Abarbeitung des Workflows nicht verloren gehen. Zum Zustand eines Workflowsystems gehören die Zustände aller Aktivitäten und die internen Verwaltungsdaten des Systems. Falls das Versagen einer Komponente zu einem Zusammenbruch des Systems führt, muß sich das System selbständig neu initialisieren und wiederanlaufen. Wenn sich das System in einem korrekten Zustand befand, muß es diesen Zustand ohne Datenverlust wiederherstellen. Wenn es sich gerade in einem Übergang zwischen zwei Zuständen befand, muß entweder ein alter, korrekter Zustand möglichst kurz vor dem Zusammenbruch wiederhergestellt werden (Backward Recovery) oder es muß ein neuer, korrekter Zustand erzeugt werden (Forward Recovery). Beidesmal sollen dabei möglichst wenig Daten verloren gehen.

Die Anforderung kann auch etwas abgeschwächt formuliert werden, indem man nicht mehr fordert, daß jeder Zustand des Systems persistent sein muß, sondern dies nur noch von bestimmten, ausgezeichneten Zuständen fordert. Je weniger dieser ausgezeichneten Zustände man innerhalb des Workflows hat, desto mehr Arbeit geht beim Backward Recovery verloren, da man nur auf diese ausgezeichneten Zustände zurücksetzen kann.

Anforderung: Persistenter Aktivitäten–Zustand

Ebenso wie der Workflow–Zustand sollte auch der Zustand einer Aktivität bei einem Absturz nicht verloren gehen. Zum Zustand einer Aktivität gehört nicht nur der Status der Aktivität im Workflowsystem, sondern auch der Datenzustand der in den Aktivitäten aufgerufenen Anwendungsprogrammen. In diesem Datenzustand steckt die ganze bisher in dieser Aktivität geleistete Arbeit.

Diese Anforderung ist nicht nur durch das Workflowsystem zu realisieren, da ein Teil des Datenzustands von den Anwendungsprogrammen verwaltet wird. Beispielsweise kann in einer Aktivität mit dem Anwendungsprogramm Textverarbeitung ein Text erstellt werden. Beim Auftreten eines Fehlers, sei es im Workflowsystem oder in der Textverarbeitung, soll nicht die gesamte Arbeit verloren gehen. Der Benutzer kann zwar zur Vermeidung des Arbeitsverlusts beitragen, indem er den Text in regelmäßigen Intervallen absichert, aber da aber das Workflowsystem dies nicht erzwingen kann, ist der Nutzen nur begrenzt.

3.2 Anforderungen an automatische Fehlerbehandlung

In diesem Abschnitt werden Anforderungen an Funktionen zur Fehlerbehandlung beschrieben, die automatisch vom System beim Auftreten von Fehlern ausgeführt werden. Im Gegensatz zum vorherigen Abschnitt reagiert das System auf den Fehler, indem es vom normalen Ablauf abweicht und Maßnahmen zur Fehlerbeseitigung trifft, deren Auswirkungen vom Benutzer erkannt werden können.

Anforderung: Automatischer Neustart einer Aktivität

Eine Aktivität, die mit einem Fehler beendet wird, d. h. in dem Zustand ERFOLGLOS oder AUSSER KONTROLLE überführt wurde, soll automatisch neu gestartet werden, da bei einem wiederholten Lauf ein transienter Fehler nicht mehr unbedingt auftreten muß. Die Aktivität kann dann erfolgreich beendet werden.

Eine automatisch ausgeführte Aktivität, die an einen Informationsdienst eine Abfrage gesandt hat und die aufgrund einer Überlastung des Netzes nicht erfolgreich durchgeführt werden konnte, kann hier als Beispiel dienen. Die Überlast kann nach kurzer Zeit überwunden sein und eine erneute Anfrage kann zum Erfolg führen.

Anforderung: Alternative Aktivitäten

Eine weitere Methode zur Reaktion auf erfolglose oder außer Kontrolle geratene Aktivitäten ist der Start einer alternativen Aktivität. Alternative Aktivitäten müssen in der Prozeßspezifikation zu jeder normalen Aktivität definiert werden können. Die Aufgabe einer alternativen Aktivität ist, dasselbe Ziel wie eine normale Aktivität zu erreichen, aber auf eine unterschiedliche Art und Weise.

Ein Beispiel für eine sinnvolle Anwendung einer alternativen Aktivität ist die Benachrichtigung einer Person. Die normale Aktivität kann aus dem Verschicken einer Email an eine Person bestehen. Falls dies nicht möglich ist, kann als alternative Aktivität die Person telefonisch benachrichtigt werden.

Anforderung: Alternative Wege im Geschäftsprozeß einschlagen

Die Verallgemeinerung des Konzepts der alternativen Aktivität sind alternative Wege im Geschäftsprozeß. Wenn eine Aktivität oder ein Zweig mit mehreren Aktivitäten fehlschlägt, soll der Geschäftsprozeß einen anderen alternativen Ablauf verfolgen. Im Gegensatz zu den alternativen Aktivitäten wird hier die Alternative nicht auf eine Aktivität beschränkt, sondern auf eine ganze Reihe von Aktivitäten. Dazu muß im Geschäftsprozeß der alternative Weg modelliert sein, der eingeschlagen wird, wenn ein Fehler aufgetreten ist.

Als Beispiel kann folgende Situation herangezogen werden: Bei der Buchung einer Reise wird festgestellt, daß ein Flug nicht gebucht werden kann. Die Alternative — die zweite Wahl sozusagen — wird dann versucht, eine Bahnreise zu reservieren.

3.3 Anforderungen an manuelle Fehlerbehandlung

In diesem Abschnitt werden alle diejenigen Funktionen zur Fehlerbehandlung beschrieben, die manuell, d. h. durch eine explizite Aktion des Bearbeiters, ausgelöst werden müssen. Die Fehlererkennung kann dabei entweder durch das System oder durch den Benutzer geschehen. Es muß die Möglichkeit bestehen, daß der Benutzer, wenn er einen Fehler erkannt hat, manuell in den Geschäftsprozeß eingreift und entsprechende Maßnahmen ergreift. Welche Maßnahmen dankbar sind, werden in den

folgenden Anforderungen beschrieben. Diese Anforderungen führen zu Funktionen, die an der Bedienoberfläche des Workflowsystems angeboten werden müssen und dort auch ausgeführt werden können.

Anforderung: Abbruch einer Aktivität

Während der Bearbeitung einer Aktivität kann sich herausstellen, daß die weitere Bearbeitung sinnlos geworden ist. Der Benutzer kann auch feststellen, daß die bisherige Bearbeitung fehlerhaft war und daß die Arbeit von vorne begonnen werden muß. Er benötigt dazu die Möglichkeit, die Aktivität abzubrechen. Die Aktivität kann dabei noch auf der Arbeitsliste stehen oder schon in Bearbeitung gewesen sein. Je nach Situation wird sie durch das System in den Zustand 'startbar' oder 'erfolglos' versetzt. Nach dieser Maßnahme können dann weitere Fehlerbehandlungsmechanismen greifen.

Anforderung: Abbruch des gesamten Prozesses

Beim Abbruch eines Geschäftsprozesses werden alle laufenden Aktivitäten abgebrochen. Während der Bearbeitung eines Geschäftsprozesses kann sich herausstellen, daß der gesamte Geschäftsprozeß sinnlos geworden ist. Ursache kann entweder ein aufgetretener Fehler oder eine Änderung in der Auslösesituation der Umwelt sein. So kann z. B. ein Kunde anrufen und seine Bestellung stornieren. Der dazugehörige Bestellprozeß muß aufgrund der geänderten Situation dann manuell durch einen Bearbeiter abgebrochen werden.

Anforderung: Geordneter Abbruch

Die erwähnten Abbrüche einer Aktivität oder eines Prozesses müssen einen konsistenten Datenzustand hinterlassen. Alle aktuellen, aber noch nicht in den Datenzustand eingeflossenen Ergebnisse müssen verworfen werden oder zumindest entsprechend markiert werden. Denn es kann durchaus sinnvoll sein, diese halbfertigen Ergebnisse bei einer erneuten Bearbeitung weiterzuverwenden. Alle stabil gespeicherten Ergebnisse müssen konsistent aus dem Datenbestand beseitigt werden, indem der ursprüngliche Datenzustand zu Beginn des Geschäftsprozesses wiederhergestellt wird. Die Modifikationen des Systemzustands müssen aber nicht soweit gehen, daß alle Spuren des abgebrochenen Prozesses beseitigt werden müssen. Im Protokoll des Prozesses sollte noch erkennbar sein, daß dieser Prozeß bis an eine bestimmte Stelle bearbeitet und dann abgebrochen wurde. Beim Rückgängigmachen der Datenänderungen muß berücksichtigt werden, daß diese Ergebnisse eventuell schon von anderen Programmen weiterbearbeitet worden sein können. Hier müssen entsprechende Maßnahmen getroffen werden, um dieses Problem zu beseitigen oder nicht entstehen zu lassen.

Anforderung: Aktivitäten bzw. Workflowteile erneut bearbeiten

Wenn ein Bearbeiter erst einige Zeit nach dem erfolgreichen Ende einer Aktivität bemerkt, daß dort ein Fehler gemacht wurde, möchte er diese falsch bearbeitete

Aktivität wiederholen und erneut bearbeiten. Aufgrund dieser erneuten Bearbeitung kann auch die erneute Bearbeitung nachfolgender Workflowteile nötig werden. Bei der Wiederholung müssen die Datenabhängigkeiten der Aktivitäten untereinander berücksichtigt werden, da sonst ein inkonsistenter Datenzustand entstehen kann.

Ein Sonderfall dieser Anforderung kann als *selektives Wiederholen* bezeichnet werden. In diesem Fall möchte der Benutzer eine zurückliegende fehlerhafte Aktivität A_{Fehler} wiederholen. Dabei sollen die weniger weit zurückliegenden Aktivitäten, d. h. die Aktivitäten, die nach A_{Fehler} beendet worden sind, unverändert gelassen werden. Dies ist dann möglich, wenn keine Datenabhängigkeit zwischen A_{Fehler} und den nachfolgenden Aktivitäten besteht.

Anforderung: Zusätzliche Aktivitäten

Wenn ein Benutzer einen Fehler in der Ausführung des Geschäftsprozesses erkennt, muß er mit irgendeiner Methode in den Ablauf eingreifen können. Diese Methode kann das Einfügen zusätzlicher Aktivitäten in die aktuelle Geschäftsprozeßinstanz sein. In den zusätzlichen Aktivitäten kann der Benutzer beliebige Fehlerbehandlungen durchführen. Auch kann damit flexibel auf unerwartete Situationen reagiert werden. Z. B. kann auf diese Weise einfach auf spezielle Sonderwünsche eines wichtigen Kunden eingegangen werden.

Anforderung: Ändern ganzer Teile des gerade bearbeiteten Ablaufs

Wenn die Änderungswünsche zu komplex werden, um mit zusätzlichen, sequentiell ausgeführten Aktivitäten bearbeitet werden zu können, dann müssen auch ganze Geschäftsprozeßteile zur Laufzeit neu spezifiziert werden können. Der veränderte Prozeß muß dabei weiterhin konsistent bleiben.

4 Lösungsansätze

Für die oben erwähnten Anforderungen existieren in einigen Workflowsystemen bereits Ansätze, wie die Anforderungen erfüllt werden können. Teilweise sind die im folgenden beschriebenen Ansätze bisher nicht verwirklicht worden.

Die Systemdienstebene wird im folgenden nicht weiter berücksichtigt. Wir beschränken uns auf Beschreibung von Ansätze zur Fehlerbehandlung, die hauptsächlich auf der Workflowdienstebene, teilweise auch auf der Aktivitätendienstebene angewendet werden können. Die Ansätze sollen möglichst unabhängig von einen konkreten Workflowsystem und unabhängig von einer konkreten Betriebssystem–Plattform sein.

Die im folgenden beschriebenen Ansätze sollen nicht als alternative Methoden zur Fehlerbehandlung verstanden werden, sondern sollen in der Summe die im vorherigen Kapitel gestellten Anforderungen überdecken. Daher sind die Vorschläge nicht unbedingt überschneidungsfrei.

4.1 Modellierung im Workflow

Dieser Fehlerbehandlungsansatz ist der Klasse 'Automatische Fehlerbehandlung' zuzuordnen.

Bei der Modellierung eines Geschäftsprozesses werden die Eigenschaften der Aktivitäten und der Kontroll– und Datenfluß zwischen den Aktivitäten spezifiziert. Es gibt üblicherweise keine weiteren, speziellen Konstrukte zur Behandlung von Fehlern. In den meisten bisher üblichen Workflowsystemen kann aber trotzdem im begrenzten Maße eine Fehlerbehandlung durchgeführt werden. Über die Auswertung eines Rückgabewerts (bzw. eines Fehlerkontextes) einer Aktivität kann eine fehlerhafte Bearbeitung erkannt werden. In diesem Fall kann über den Kontrollfluß spezifiziert werden, daß anstelle des normalen Ablaufs eine Fehlerbehandlung in weiteren, zusätzlichen Aktivitäten stattfinden soll. In diesen Aktivitäten können Maßnahmen zur Beseitigung der Auswirkungen der fehlgeschlagenen Aktivitäten durchgeführt werden. In der Modellierung sind die Aktivitäten, die zur Fehlerbehandlung eingesetzt werden, nicht von den „normalen“ Aktivitäten zu unterschieden. Nach der Bearbeitung der Aktivitäten für die Fehlerbehandlung wird der Kontrollfluß zur normalen Aktivität weitergeleitet. In Abbildung 3 sind drei „normale“ Aktivitäten (A, B, C) zu sehen und zu jeder normalen Aktivität gibt es eine oder mehrere Fehlerbehandlungsaktivitäten (F_x).

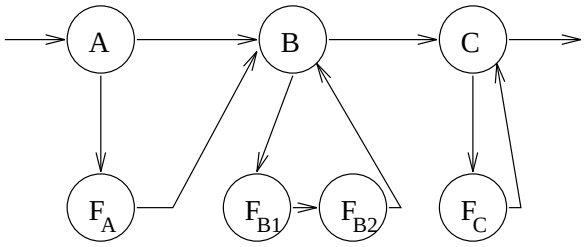


Abbildung 3: Die Modellierung von Fehlerbehandlung mit vorhandenen Sprachmitteln

Der Vorteil des Nutzens der vorhandenen Modellierungsmittel zur Modellierung der Fehlerbehandlung liegt darin, daß keine Erweiterung der bestehenden Modellierungssprache nötig ist. Die Nachteile werden im folgenden aufgelistet:

- Die Fehlerursache muß in den Fehlerbehandlungsaktivitäten beseitigt werden. Ansonsten erzeugt man auf diese Weise eine Endlosschleife.

- Die Modellierung von Geschäftsprozessen zusammen mit der nötigen Fehlerbehandlung ist sehr aufwendig. Zu jeder Aktivität muß im Prinzip eine Fehlerbehandlung modelliert werden. Die Erstellung von Geschäftsprozessen wird dadurch erschwert. Als Folge der aufwendigen Modellierung ist damit zu rechnen, daß daher wohl häufig unwichtige Sonderfälle zur Erleichterung der Modellierung einfach weggelassen werden. Die Prozesse werden dadurch wenig robust.

Die selten benutzten Fehlerbehandlungsaktivitäten müssen genauso intensiv entwickelt und getestet werden, wie die häufig benutzen normalen Aktivitäten, die den eigentlichen Prozeß ausmachen. Die bedeutend größere Zahl an Aktivitäten bei Geschäftsprozessen mit Fehlerbehandlung erhöht die Kosten für die Entwicklung und den Test der Prozesse.

- Das Geschäftsprozeß–Modell wird durch die vielen Fehlerbehandlungsaktivitäten schnell sehr unübersichtlich. Pro Aktivität können mehrere Fehlerbehandlungsaktivitäten hinzukommen. Durch die hohe Anzahl an Aktivitäten läßt nicht mehr erkennen, aus welchen Aktivitäten der Prozeß eigentlich besteht, da keine Unterscheidung zwischen Fehlerbehandlungsaktivitäten und normalen Aktivitäten besteht.

- Die Modellierung der Fehlerbehandlung kann auch wieder selbst Fehler enthalten. Es wäre daher eine Fehlerbehandlung für die Fehlerbehandlung nötig. Irgendwann muß willkürlich angenommen werden, daß eine Aktivität die Auswirkungen des Fehlers beseitigt und immer erfolgreich beendet wird. Diese Annahme ist in der Realität nicht haltbar. Dies ist allerdings ein grundlegendes Problem der Fehlerbehandlung, das auch bei anderen Methoden nur schwer umgangen werden kann.

- Die einzelnen Fehlerabfragen sind oftmals in ihrem Aufbau identisch. Sie müssen aber für jede Aktivität immer wieder neu modelliert werden. Nur wenn das Workflowsystem entsprechende Methoden anbietet (z. B. parametrisierbare Blöcke von Aktivitäten), kann eine Fehlerbehandlung wiederverwendet werden.

- Die Modellierung aller im voraus bekannten Fehlerfälle fängt nicht alle möglichen Fehler ab. Es können immer noch unerwartete Fehler auftreten, für die keine Behandlung definiert ist. Es ist praktisch unmöglich, alle Fehlerfälle vorherzusagen und mit einer entsprechenden Reaktion abzudecken. Bei der Modellierung muß immer vorausgesehen werden, wo im Geschäftsprozeß Fehler und Ausnahmesituationen auftreten können.

Wenn das Aktivitätenprogramm abstürzt und damit keinen Rückgabewert mehr an das Workflowsystem übermitteln kann, d. h. wenn die Aktivität in den Zustand AUSSER KONTROLLE überführt werden muß, dann funktioniert diese Methode der Fehlerbehandlung nicht mehr.

4.2 Alternative Aktivitäten und Pfade

Dieser Fehlerbehandlungsansatz ist auch der Klasse 'Automatische Fehlerbehandlung' zuzuordnen. Im Gegensatz zum vorherigen Ansatz wird hier vorgeschlagen, die Modellierungssprache um ein weiteres Konstrukt zu ergänzen, das speziell für die Fehlerbehandlung geeignet ist.

Wenn eine Aktivität oder eine gruppierte Menge von Aktivitäten fehlschlägt, dann sollte die Möglichkeit bestehen, eine alternative Vorgehensweise zu versuchen. Diese kann in der Ausführung einer alternativen Aktivität oder eines alternativen Pfads im Geschäftsprozeß geschehen. Die Umschaltung auf die alternative Vorgehensweise wird automatisch durch das Workflowsystem beim Auftreten des Auslösers vollzogen. Dieses Verhalten kann bei der Modellierung durch ein *Präferenzkonstrukt* erreicht werden: Wenn während der Ausführung eines Workflow–Pfads ein Fehler auftritt, wird ein alternativer Pfad ausgeführt. Ein Pfad besteht dabei aus einer oder mehrere Aktivitäten. In Abbildung 4 ist ein ordentlicher Pfad mit drei Aktivitäten zu sehen. Wenn bei der Ausführung ein Fehler auftritt, wird auf den alternativen Pfad mit zwei Aktivitäten umgeschaltet.

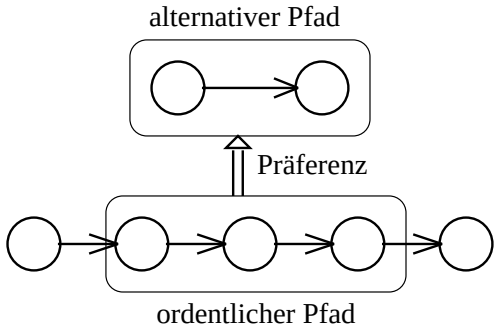


Abbildung 4: Ein im Geschäftsprozeß spezifizierter alternativer Pfad

Als Auslöser für die alternative Ausführung kommen zwei Arten in Frage: Entweder schlug die Ausführung einer Aktivität auf dem ordentlichen Pfad fehl oder eine Aktivität wurde innerhalb einer vorher bestimmten Zeit nicht bearbeitet. Die Umschaltung kann also aufgrund eines aufgetretenen Fehlers oder aufgrund des Eintretens eines Ereignisses wie das Ablaufen einer Zeitüberwachung erfolgen.

Der Start einer alternativen Aktivität bzw. das Umschalten auf einen alternativen Pfad kann grundsätzlich auf zwei Arten erfolgen: Entweder werden die alternativen Aktivitäten zusätzlich oder ersetzend gestartet. Bei dem zusätzlichen Start werden die ordentlichen Aktivitäten wiederholt und parallel zu den alternativen Aktivitäten gestartet. Diese Vorgehensweise ist dann sinnvoll, wenn z.B. eine säumige Bearbeitung einer Aktivität angemahnt werden soll. Die Bearbeitung der normalen Aktivität hat sich aber durch die Mahnung noch nicht erledigt. Der ersetzende Start kann durch eine Situation motiviert werden, in der etwas fehlgeschlagen ist und daher etwas anderes probiert werden soll.

Mit alternativen Aktivitäten kann eine ganze Hierarchie aufgebaut werden, da die einzelnen alternativen Aktivitäten auch wieder fehlschlagen können. Das Konzept der alternativen Aktivität löst damit das Problem der Fehlerbehandlung in Workflowsystemen nicht, da immer noch die letzte alternative Aktivität fehlschlagen kann. Es mindert nur das Problem.

Alternative Aktivitäten sind dann sinnvoll, wenn schon zur Modellierungszeit bekannt ist, wo Fehler oder timeouts innerhalb eines Workflows auftreten können.

4.3 Ad–hoc–Modifikationen

Dieser Fehlerbehandlungsansatz ist der Klasse 'Manuelle Fehlerbehandlung' zuzuordnen.

Unter dem Begriff Ad–hoc–Modifikationen werden alle Maßnahmen zusammengefaßt, die ein Benutzer ergreifen kann, um *manuell* in die Kontrolle der Workflow–Engine einzugreifen. Mit solchen Modifikationen können zur Laufzeit Änderungen an einem Workflow vorgenommen werden, die im allgemeinen nur für diese eine spezielle Instanz des Prozesses gelten. Änderungen können entweder an den Kontrolldaten oder an der Struktur des Workflows vorgenommen werden.

Im folgenden werden einige typische Ad–hoc–Modifikationen aufgezählt:

- Einzelne Aktivitäten werden in Anschluß an eine Aktivität in den Workflow eingefügt (Umleitung).
- Einzelne in Workflow spezifizierte Aktivitäten werden ausgelassen (Abkürzung).
- Einzelne Aktivitäten werden zusätzlich parallel zu einer Aktivität in den Workflow eingefügt (Zusatz).

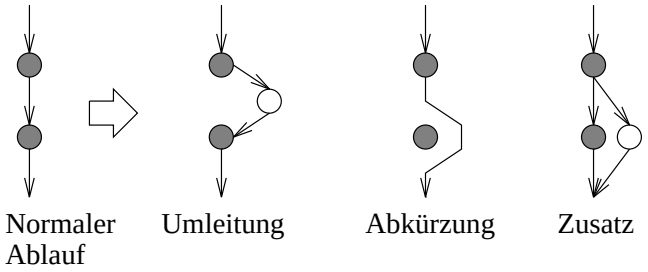


Abbildung 5: Die Ad–hoc–Modifikationen Umleitung, Abkürzung und Zusatz

- An einer Verzweigung des Kontrollflusses wird die Entscheidung der Workflow–Engine einen bestimmten Zweig zu verfolgen überstimmt. Dazu wird in die Evaluierung des Entscheidungsprädikats eingegriffen, indem bestimmte Wertebelegungen manuell geändert werden.
- Entscheidungen der Workflow–Engine (z. B. bezüglich der Verteilung der Aktivitäten an bestimmte Personen) können überstimmt werden.
- Der Abbruch eines Prozesses oder einer Aktivität kann auch als Ad–hoc–Modifikationen verstanden werden.

- Zur Modellierungszeit kann ein Teil des Workflows unspezifiziert gelassen werden. Nach der Instanziierung des Workflows kann durch das zusätzliche Einfügen von Aktivitäten der Workflow zuerst inkrementell spezifiziert und dann ausgeführt werden. In Abbildung 6 ist diese Art der Feinspezifikation veranschaulicht.

- In einer weiteren Ausbaustufe kann man eventuell erreichen, daß der Bearbeiter im unspezifizierten Teil des Workflows seine Tätigkeiten ausführt, ohne sie vorher zu spezifizieren. Das Workflowsystem protokolliert diese Tätigkeiten (z. B. Aufruf von Anwendungsprogrammen) mit, generalisiert sie und erzeugt daraus automatisch eine Feinspezifikation. Dieser mitprotokollierte Workflowteil kann dann als Vorlage für ein erweitertes Modell des Geschäftsprozesses dienen.

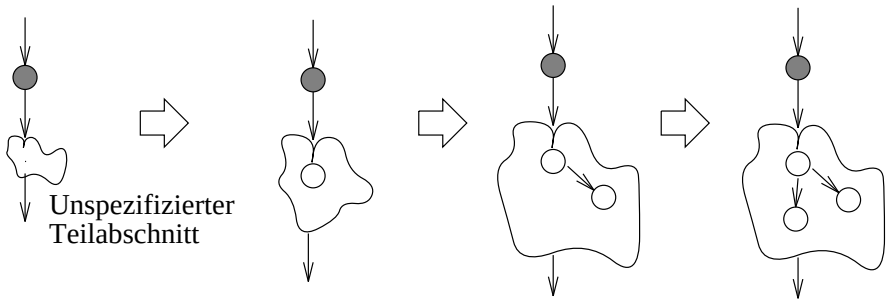


Abbildung 6: Ein unspezifizierter Teilabschnitt wird schrittweise verfeinert

Ad–hoc–Modifikationen sollen dem Benutzer die Möglichkeit geben, vom starren Ablauf des modellierten Workflows abzuweichen und so flexibel Ausnahmesituationen behandeln zu können. Er kann damit auch auf solche Situation reagiert, die so ungewöhnlich sind, daß daran während der Modellierung nicht gedacht worden ist. Im Gegensatz zu den alternativen Aktivitäten muß man nicht schon zur Modellierungszeit den genauen Punkt kennen, an dem ein Fehler auftreten kann.

Ein Geschäftsprozeß wird mit dem Ziel geschaffen, eine bestimmte Aufgabe zu erfüllen. Sinn des Einsatzes von Workflowsystemen ist die Überwachung und Kontrolle, daß dieses Ziel auch erreicht wird. Wenn nun jeder Bearbeiter den Prozeß umdefinieren kann, dann kann nicht mehr gewährleistet werden, daß das betriebliche Ziel des Geschäftsprozesses auch erreicht wird. Durch das Einfügen beliebiger Aktivitäten entsteht die Gefahr, daß eine Tätigkeit ausgeführt wird, die die Datenintegrität des Prozesses beeinträchtigt. Eine Garantie des Workflowsystem, daß die Daten nach Ausführung korrekter Workflows konsistent sind, kann nicht mehr gegeben werden, da korrekte Workflows durch Ad–hoc–Modifikationen zu nicht–korrekten Workflows verändert werden können.

Beim Einsatz von Ad–hoc–Modifikationen muß daher gefordert werden, daß die Rahmenbedingungen des Prozesses (das Erreichen des Ziels) und die Datenkonsistenz durch die vorgenommenen Modifikationen nicht verletzt werden. Dazu müssen

während der Modellierung Integritätsbedingungen angegeben werden, die Teil des Workflow–Modells werden. Ad–hoc–Modifikationen werden nur dann zur Laufzeit erlaubt, wenn sie keine der Integritätsbedingungen des Workflows verletzen.

Die Ausführung der Funktionen, die zur Ad–hoc–Modifikationen dienen, muß über die Vergabe von Modifikationsrechten abgesichert werden. Falls dies nicht geschieht, kann die Kontrollfunktion der Workflow–Engine durch jeden Bearbeiter beliebig umgangen werden. Als Träger solcher Rechte eignen sich die Organisationsrollen: Der Abteilungschef darf einen Prozeß verändern, während seine Mitarbeiter dies nicht ohne seine Erlaubnis dürfen. Einzelne Prozeßelemente im Workflow–Modell, etwa ein Kontrollflußpfeil oder ein Workflowdatenobjekt, können auch durch den Einsatz von Rechten vor Veränderung geschützt werden.

Ad–hoc–Modifikationen sind gerade ein aktuelles Forschungsthema im Workflowbereich. Entsprechend gibt es noch keine einheitlichen Vorstellungen über die zu realisierenden Funktionen und nur selten Realisierungen.

5 Workflow–Transaktionen

In Datenbankensystemen wird das Konzept der ACID–Transaktionen zur robusten und fehlertoleranten Abwicklung von Datenbankoperationen eingesetzt. Die Übertragung dieses Konzepts auf Workflowsysteme birgt einige Schwierigkeiten. Nach einer kurzen Vorstellung des Konzepts werden in diesem Abschnitt einige Lösungsansätze aus der Literatur kurz beschrieben und dann wird ein eingeschränktes Einsatzgebiet dieses Konzepts in Workflowsystemen vorgestellt.

5.1 Datenbank–Transaktionen

Bei der Entwicklung von Datenbankanwendungssystemen sah man sich schon vor einiger Zeit mit ähnlichen Problemen zur Fehlerbehandlung konfrontiert, wie sie jetzt auch bei Workflowsystemen auftreten: Man wollte unter allen Umständen die Konsistenz der Daten sichern, ohne daß der Entwickler von Datenbankanwendungssystemen durch komplexe Fehlerbehandlungen belastet oder gar überlastet wird. Das Problem wird bei Datenbanken üblicherweise mit dem Konzept der *ACID–Transaktionen* gelöst, das in [GR93] ausführlich beschrieben ist.

Eine ACID–Transaktion ist eine Menge von Operationen auf Daten mit folgenden Eigenschaften:

- Atomicity (Atomizität)
  Eine Transaktion wird entweder vollständig durchgeführt oder es hat den Anschein, als sei sie überhaupt nicht ausgeführt worden. Sie ermöglicht einen Übergang von einem Zustand der Datenbank in den nächsten Zustand, ohne daß von außen Teilschritte dieses Übergangs sichtbar wären. Falls bei dem Übergang ein Fehler auftritt, befindet sich das Datenbanksystem wieder im ur-

sprünglichen Zustand und es hat den Anschein als hätte es diesen Zustand nie verlassen.

- Consistency (Konsistenz)
  Eine Transaktion produziert nur konsistente Ergebnisse, ansonsten wird sie abgebrochen. Ein Ergebnis ist konsistent, wenn der neue Datenbankzustand alle Konsistenzbedingungen der Anwendung erfüllt, die durch die Spezifikation der Datenbankanwendung gegeben sind. Die Transaktion überführt somit einen konsistenten Ausgangszustand in einen neuen, ebenso konsistenten Zustand.

- Isolation
  Eine Transaktion kann Operationen auf der Datenbank so ausführen als wäre sie alleine im Datenbanksystem. Das System sorgt durch den Einsatz von Datensynchronisationsprotokollen dafür, daß die parallele Ausführung von Transaktionen möglich wird, ohne das es zu Konsistenzverletzungen kommt. Die Auswirkungen einer Transaktion werden damit erst nach ihrer erfolgreichen Beendigung für andere Transaktionen sichtbar.

- Durability (Dauerhaftigkeit)
  Die Auswirkungen einer Transaktion sind nach dem erfolgreichen Abschluß der Transaktion dauerhaft gespeichert. Das System garantiert, daß es auch nach dem Auftreten eines Systemfehlers die Daten wieder herstellen kann.

Die Datenbankanwendungsprogrammierung wird durch diese garantierten Eigenschaften der Transaktion erheblich vereinfacht, da die Auswirkungen eines Fehlers (z.B. inkonsistente Datenzustände) nicht mehr durch das Anwendungsprogramm zu beseitigen sind. Das Datenbanksystem garantiert, daß sich die bearbeiteten Daten nach dem Fehlschlag einer Transaktion exakt in dem Zustand befinden, in dem sie zu Beginn der Transaktion waren. Es müssen daher keine Funktionen geschrieben werden, die den Zustand der Daten nach einem Fehler bestimmen und die entsprechenden Maßnahmen zur Wiederherstellung des Ursprungszustands durchführen. Die anwendungsabhängige Reaktion auf die fehlgeschlagene Transaktion muß allerdings weiterhin durch den Anwendungsprogrammierer implementiert werden. Die vorher in allen Anwendungsprogrammen notwendigen Fehlerbehandlungsroutinen sind durch die Transaktionen in das Datenbanksystem übernommen worden.

5.2 Datenbank–Transaktionen in Workflowsystemen

Das Konzept der ACID–Transaktionen kann auch bei der Bearbeitung von Workflows eingesetzt werden. Transaktionen mit der ACID–Eigenschaft, die in Workflowsystemen eingesetzt werden, nennen wir *Workflow–Transaktionen*. Ein gesamter Workflow oder nur Abschnitte eines Workflows können zu einer Workflow–Transaktion zusammengefaßt werden. Eine Workflow–Transaktion besitzt dieselben Eigenschaften wie

eine Datenbank–Transaktion. Die folgenden Punkte müssen bei der Übertragung des Transaktionen–Konzepts auf Workflowsysteme berücksichtigt werden:

Innerhalb von Datenbank–Transaktionen werden nur einfache Operationen wie das Lesen oder Schreiben eines Datums verwendet. Falls z. B. SQL–Befehle in der Transaktion verwendet werden, werden diese in Lese– und Schreiboperationen umgewandelt. Diese Operationen arbeiten nur auf Daten und nicht auf physischen Objekten. Daher ist die inverse Operation einfach zu bestimmen. Leseoperationen ändern nichts und brauchen daher keine inverse Operation. Das Schreiben eines Wertes kann durch ein erneutes Schreiben mit dem ursprünglichen Wert rückgängig gemacht werden. Die einfachen Operationen werden durch in Workflow–Transaktionen durch erheblich komplexere Aufrufe von Workflow–Aktivitäten ersetzt.

In einer Datenbank–Transaktion sind alle Operationen streng sequentiell angeordnet. In Workflow–Transaktion dagegen kann der Kontrollfluß zwischen den Aktivitäten frei definiert werden. Insbesondere können auch parallele Aktivitäten stattfinden.

Dieser Ansatz der uneingeschränkten Übertragung von Datenbank–Transaktionen in Workflowsysteme weist aber einige schwerwiegende Probleme auf, die den alleinigen Einsatz der Workflow–Transaktionen als Mittel zur fehlertoleranten Abwicklung von Workflowsystem verhindern:

- Datenbanken bieten Resource Manager an, die die Lese– und Schreiboperationen im Rahmen von Transaktionen implementieren.

  Ein Resource Manager stellt transaktionsgeschützte Zugriffsmethoden auf die von ihm verwalteten Ressourcen bereit. Ressourcen können gemeinsam benutzte Daten oder auch allgemeine Betriebsmittel, wie Maschinen sein. Ein Resource Manager synchronisiert gleichzeitige Zugriffe auf seine Ressourcen (concurreny control) und hält dadurch die Ressourcen konsistent. Zur Synchronisation werden üblicherweise Sperrverfahren verwendet, es können aber auch andere Protokolle, wie z. B. optimistische Synchronisationsverfahren eingesetzt werden. Der Resource Manager speichert im Fall eines ordnungsgemäßen Endes der Transaktion ('commit') alle in dieser Transaktion gemachten Änderungen an den Ressourcen stabil ab. Bei einem Abbruch der Transaktion ('rollback') muß der Resource Manager den Zustand der Ressource vor Beginn der Transaktion wieder hergestellen. Dazu protokolliert er auf dem Log des Transaktionssystems alle Operationen. Dieser Log wird auch dazu benutzt, um nach einem Systemabsturz selbständig den aktuellen Zustand wieder herstellen zu können ('recovery').

  Workflow–Aktivitäten besitzen im allgemeinen keine solchen Vorkehrungen, um an einer Transaktionen teilnehmen zu können. Sie dürfen ihrer Definition nach beliebiger komplexer Natur sein. Selbst manuelle Tätigkeiten sollen als Aktivitäten modelliert werden können. In vielen Fällen werden in Aktivitäten bereits bestehende Software eingesetzt, die keine der ACID–Eigenschaften unterstützen.

Ein Workflowsystem hat daher keine Kontrolle darüber, welche Daten durch eine Aktivität verändert werden. Bei dem Abbruch einer Transaktion kann daher der ursprüngliche Zustand weder durch das Workflowsystem noch durch die Aktivität wieder hergestellt werden. Die entsprechende inverse Aktivität kann das System nicht automatisch ermitteln. Eine solche Operation müßte von der Aktivität oder dem Workflow–Modellierer bereitgestellt werden.

Nicht speziell vorbereitete Aktivitäten können wegen der fehlenden Recovery–Fähigkeit und der unsynchronisierten Datenzugriffe nicht an einer Workflow–Transaktion teilnehmen.

- ACID–Transaktionen in Datenbanken haben im allgemeinen nur eine sehr kurze Lebensdauer (Größenordnung: Sekunden bis Minuten). Das Datenbanksystem ist darauf ausgelegt, möglichst viele Transaktionen pro Zeiteinheit durchzuführen. Wenn eine Transaktion fehlschlägt, kann sie nochmals gestartet werden. Workflows dauern im allgemeinen sehr lange (Größenordnung: Stunden bis Wochen). In dieser Zeit kann sehr viel Arbeit innerhalb eines Workflows geleistet werden. Diese Arbeit ginge verloren, wenn der Workflow auf den Anfangszustand zurückgesetzt würde.

- Wie in Datenbanksystemen müssen auch in Workflowsystemen Transaktionen parallel ablaufen können. Dabei kann es zu Konflikten kommen, wenn auf dasselbe Datenobjekt zugegriffen wird. Zur Verhinderung von gleichzeitigen Zugriffen und daß vorläufige Daten vor dem Ende einer Transaktionen von anderen Transaktionen weiterverarbeitet werden, benutzt man Datensynchronisationsprotokolle. Damit realisiert man die Isolationseigenschaft der Transaktionen.

  ACID–Transaktionen benutzen häufig ein Sperrverfahren zur Synchronisation der Datenzugriffe. Die Sperren werden in einer Wachstumsphase der Transaktionen erworben und in einer Schrumpfungsphase wieder abgegeben. Wenn eine Sperre wieder freigegeben worden ist, dürfen keine neue Sperren erworben werden. Zusammen mit den langen Laufzeiten der Workflows bedeutet das, daß die Sperren in Workflow–Transaktionen sehr lange gehalten werden müssen. Es steigt somit die Wahrscheinlichkeit, daß Workflows auf gesperrte Daten zugreifen wollen und deshalb warten müssen. Die Wartezeit kann dabei unter Umständen sehr lange dauern. Dadurch wird die geforderte Parallelität der Workflows stark eingeschränkt. Zudem steigt mit der Dauer einer Transaktion die Verklemmungsgefahr (Deadlock).

- Eine Aufgabe von Sperren ist es, daß die Zwischenergebnisse einer Transaktion bis zur deren Bestätigung (Commit) nicht anderen Transaktionen zur Verfügung stehen. Damit soll verhindert werden, daß andere Transaktionen mit vorläufigen Daten arbeiten, die beim Abbruch einer Transaktion ungültig würden. Die Transaktionen sind voneinander isoliert.

Die Isolationseigenschaft ist aber bei Workflowsystemen häufig unerwünscht. Dort existiert die Situation, daß man kooperativ auf einem gemeinsamen Datenbestand (z.B. ein Dokument) arbeiten will, in dem auch vorläufige Ergebnisse anderer Aktivitäten erwünscht sind. Man möchte nicht abwarten, bis alle vorherigen Ergebnisse fertig und bestätigt sind. In Geschäftsprozessen müssen auch vorläufige Zwischenergebnisse benutzt werden können.

Das Konzept der Kooperation ist viel wichtiger als das der Isolation. Der gemeinsame Zugriff auf Daten darf dabei aber trotzdem nicht zu kritischen Inkonsistenzen führen.

Aus diesen Gründen erweisen sich ACID–Transaktionen als alleiniges Mittel zur Abwicklung von fehlertoleranten Workflows als nicht geeignet. Daher wurden in der Literatur verschiedene erweitere Ansätze beschrieben, die die angesprochenen Probleme stärker berücksichtigt.

5.3 Erweiterte Transaktionsmodelle

In diesem Abschnitt werden einige bekannte erweiterte Transaktionskonzepte (siehe auch [GR93]) vorgestellt. Die ACID–Eigenschaften der Transaktionen werden dabei teilweise aufgegeben und teilweise erweitert.

Sicherungspunkte

Eine Transaktion kann durch Sicherungspunkte in mehrere Abschnitte unterteilt werden. Beim Auftreten eines Fehlers kann die Transaktion statt an den Anfang auf einen Sicherungspunkt zurückgesetzt werden. Von dort aus kann die Arbeit der Transaktionen wiederaufgenommen werden. Dadurch geht weniger Arbeit als bei einem vollständigen Rücksetzen verloren. Die ACID–Eigenschaften der Transaktionen bleiben dabei erhalten.

Geschachtelte Transaktionen

Eine geschachtelte Transaktion (nested transaction) [Mos81] besteht aus einer Hierarchie von Transaktionen. Die oberste Transaktion besitzt als einzige alle ACID–Eigenschaften. Alle untergeordneten Transaktionen entbehren die Dauerhaftigkeit. Beim Rücksetzen einer übergeordneten Transaktion müssen alle bereits bestätigten Untertransaktionen mit zurückgesetzt werden. Die Ergebnisse einer Untertransaktion sind nur für die unmittelbar übergeordnete Transaktion sichtbar. Die Ergebnisse der geschachtelten Transaktionen werden für die Außenwelt erst mit der Bestätigung der obersten Transaktion sichtbar. Untertransaktionen können parallel ablaufen, dürfen dann aber nur auf disjunkte Datenobjekte zugreifen.

Geschachtelte Transaktionen sind damit als eine Generalisierung von Sicherungspunkten zu verstehen. Die Granularität der Bearbeitung kann wie dort beliebig verfeinert werden. Zudem wird aber die parallele Ausführung von Untertransaktionen ermöglicht und die Vergabe der Sperren an Untertransaktionen kann flexibler gesteuert werden. Untertransaktionen besitzen nur die Sperren, die sich von der überge-

ordneten Transaktion bekommen haben und geben neu erworbene Sperren an die übergeordnete Transaktion zurück.

Flexible Transaktionen

Das Modell der flexiblen Transaktionen [ZNBB94] basiert auf übergeordneten Transaktionen, die für die Steuerung der untergeordneten Transaktionen verantwortlich sind. Dieser Ansatz wird bei Transaktionen eingesetzt, die sich über mehrere heterogene Datenbanken erstrecken. Die verteilte flexible Transaktion ist damit nicht mehr atomar, während es die untergeordneten Transaktionen auf einem Rechnerknoten weithin sind.

Der Grundgedanke ist die Bereitstellung alternativer Pfade mit unterschiedlicher Präferenz innerhalb der verteilten Transaktion. Wenn eine Subtransaktion abbricht, wird auf einen alternativen Pfad mit niedriger Präferenz umgeschaltet. Eine flexible Transaktion wird bestätigt, wenn einer der alternativen Pfade zum Erfolg geführt hat.

Es werden drei Arten von Untertransaktion unterschieden. Eine Subtransaktion ist kompensierbar, wenn es eine weitere Transaktion gibt, mit der die Ergebnisse semantisch rückgängig gemacht werden können. Eine wiederholbare Subtransaktion garantiert nach endlich vielen Versuchen einen erfolgreichen Abschluß. Eine Pivot–Transaktion (Angelpunkttransaktion) ist weder kompensierbar noch wiederholbar.

Für jeden Pfad im Ausführungsgraphen markiert eine Pivot–Transaktion den kritischen Punkt dieses Pfades. Vor diesem Punkt kann die Transaktion rückgesetzt werden, nach diesem Punkt muß sie zu einem erfolgreichen Ende geführt werden.

Dies kann dann garantiert werden, wenn die flexible Transaktionen wohlgeformt ist, d. h. auf dem Pfad mit der niedrigsten Präferenz sind alle Transaktionen vor dem kritischen Punkt kompensierbar und alle Transaktionen nach dem Punkt wiederholbar.

Sagas

Sagas [GMS87] bestehen aus einer linearen Kette von Transaktionen, die durch eine höhere Kontrollschicht gesteuert werden. Die Untertransaktionen geben nach ihrer Bestätigung ihre Datenänderungen frei. Sagas sind daher nicht voneinander isoliert. Zu jeder Untertransaktion gibt es eine Kompensations–Transaktion, die die Ergebnisse der Untertransaktion semantisch rückgängig machen. Eine Kompensations–Transaktion darf nicht scheitern. Wenn eine Transaktion innerhalb der Saga abbricht, werden durch die höhere Kontrollebene alle schon bestätigten Transaktionen durch die Ausführung der Kompensations–Transaktionen in umgekehrter Reihenfolge kompensiert.

ConTracts

Der ConTracts–Ansatz [WR92] geht über die reine Erweiterung des ACID–Transaktionskonzepts hinaus und bietet eine Reihe von Konzepten an, die eine robuste und fehlertolerante Abwicklung von langandauernden Abläufen erlauben.

In einem ConTract werden die elementaren Berechnungsschritte von der Beschrei-

bung der Ablaufstruktur getrennt. Die eigentliche Anwendungsprogrammierung findet in rein sequentiellen Code in den elementaren Arbeitseinheiten (Steps) statt, die als klassische ACID–Transaktionen ausgeführt werden. In einem Skript wird der Kontrollfluß zwischen den Steps beschrieben. Steps können dabei zu weiteren ACID–Transaktionen geschachtelt werden, zwischen denen Abhängigkeitsregeln angegeben werden können. Zu jedem Step muß es einen Kompensations–Step geben, der die Auswirkungen des Steps wieder rückgängig machen kann.

Als Korrektheitskriterium wird die Eigenschaft benutzt, daß Anfangs– und Endzustände eines ConTracts korrekt sind. Bei einem Systemfehler wird die Bearbeitung, wenn möglich, an derselben Stelle vorgesetzt oder der ConTract wird über die Kompensations–Steps in den Anfangszustand versetzt.

Die in einem ConTract verwendeten Daten werden in einer Kontext–Datenbank stabil verwaltet. Der Zugriff der Steps auf die gemeinsamen Daten wird über Eingangs– und Ausgangsinvarianten synchronisiert. Spezifisch für die Anwendung können unterschiedliche Synchronisationsverfahren eingesetzt werden.

5.4 Workflow–Transaktionen

Die Probleme bei der Verwendung von ACID–Transaktionen in langandauerenden Vorgängen sind bereits angesprochen worden. Das Konzept der Workflow–Transaktionen eignet sich also nicht als alleiniges Konzept zur Implementierung von fehlertoleranten Workflows. Wenn man diese Probleme berücksichtigt, zeigt sich aber, daß dieses Konzept für einen begrenzten Einsatz in Workflowsystemen durchaus geeignet sein kann. Das Einsatzgebiet ergibt sich aus den folgenden Randbedingungen:

- Die Transaktionen müssen von kurzer Dauer sein. Ansonsten greifen die bereits oben beschriebenen Probleme bei langandauerenden Transaktionen.

- Alle Aktivitäten der Workflow–Transaktionen müssen als Resource Manager an der Transaktionen teilnehmen können und entsprechende Schnittstellen zur Steuerung des Recovery anbieten. Die Schnittstellen müssen zu dem im Workflowsystem verwendeten Transaktions–Service passen. Es können damit keine beliebigen Aktivitäten an einer Workflow–Transaktion teilnehmen!

5.4.1 Begriffe

Eine **Sphäre** ist eine Menge von Aktivitäten in einem Workflow. Wenn zwischen Aktivitäten Abhängigkeiten in der Art existieren, daß nie eine der Aktivitäten erfolgreich und die andere erfolglos beendet werden darf, können die Aktivitäten zu einer Sphäre zusammengefaßt werden. Eine Sphäre wird zur Modellierungszeit des Workflows spezifiziert. In der Abbildung 7 ist eine Sphäre in einem Aktivitätennetz eingezeichnet.

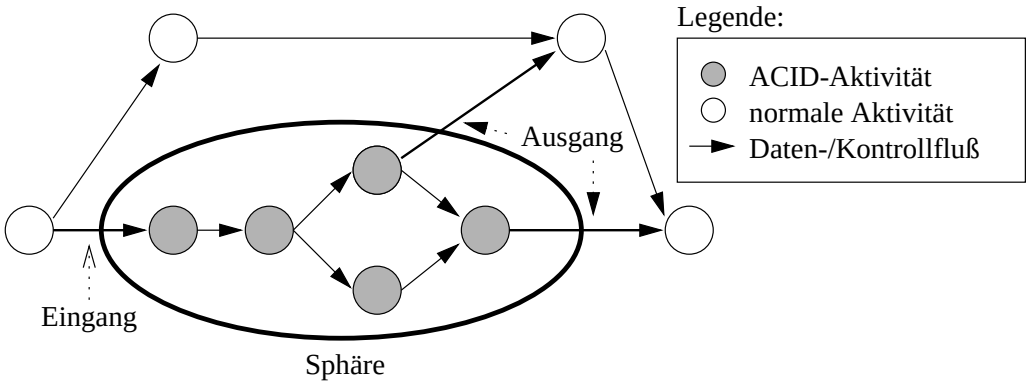


Abbildung 7: Eine Sphäre in einem Aktivitätennetz

Eine Sphäre muß dabei keine Zusammenhangskomponente im Aktivitätennetz bilden. Es müssen also nicht alle Aktivitäten in einer Kontrollflußbeziehung stehen, wie in Abbildung 8 gezeigt.

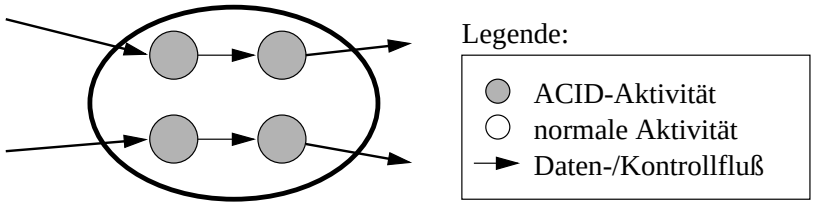


Abbildung 8: Eine Menge von Aktivitäten in einer Sphäre, die keine Zusammenhangskomponente im Aktivitätennetz bilden

Als **Eingang** einer Sphäre wird der Kontroll– bzw. Datenfluß bezeichnet, der von einer Aktivität außerhalb der Sphäre zu einer Aktivität innerhalb der Sphäre führt. Entsprechend wird als **Ausgang** der Kontroll– bzw. Datenfluß definiert, der von einer Aktivität innerhalb der Sphäre zu einer Aktivität außerhalb der Sphäre führt. Eine Sphäre kann mehrere Ein– und Ausgänge besitzen. In der Abbildung 7 hat die Sphäre einen Eingang und zwei Ausgänge.

Als **ACID-Aktivitäten** werden Aktivitäten bezeichnet, die die Eigenschaft haben, daß sie entweder selbst Resourcen–Manager sind oder nur auf Daten über Resourcen–Manager zugreifen. Es muß gewährleistet sein, daß die in den Aktivitäten verwendeten Daten nur über Resourcen–Manager im Rahmen einer Transaktionen gelesen oder verändert werden (siehe Abschnitt 5.4.3).

Definition: *Workflow–Transaktion*

> *Eine Workflow–Transaktion ist eine Menge von Aktivitäten (eine Sphäre im Workflowmodell), die im Kontext einer ACID–Transaktion ausgeführt werden.*

5.4.2 Das Konzept der Workflow–Transaktionen

Ein Workflowsystem, das Workflow–Transaktionen anbietet, tritt als Starter und als normaler Teilnehmer der Transaktion auf. Beim Betreten einer Sphäre muß die Workflow–Transaktionen durch das Workflowsystem bei einem Transaktions–Service initiiert werden. Dann registriert sich das Workflowsystem selbst als Teilnehmer. Dazu muß das Workflowsystem als Resource Manager für die Workflow–Daten auftreten können.

Falls das Workflowsystem Datenflüsse verwaltet, über die Aktivitäten mit Daten versorgt werden, muß das System dafür sorgen, daß die Daten die Sphäre nicht verlassen. Dasselbe gilt für den Kontrollfluß. Erst mit dem erfolgreichen Ende der Transaktion dürfen Aktivitäten außerhalb der Sphäre angestoßen werden. Das Workflowsystem muß als Resource Manager die Isolationseigenschaft der Transaktion bereitstellen, indem die Daten– und Kontrollflüsse der Sphäre nach außen bis zum erfolgreichen Ende verzögert werden.

Durch die Teilnahme an der Transaktion kann das Workflowsystem den Bearbeitern eine Funktion an ihrer Bedienoberfläche anbieten, mit der sie eine Workflow–Transaktion interaktiv abbrechen können.

Falls die Workflow–Transaktion rückgesetzt werden soll, muß das Workflowsystem den Zustand des Workflows wieder in den Anfangszustand der Sphäre bringen. Alle bisherigen Änderungen innerhalb der Sphäre müssen durch das Workflowsystem rückgängig gemacht werden. Nach dem Abbruch und Rücksetzen der Sphäre wird die Workflow–Transaktion durch das Workflowsystem neu gestartet. Anstatt eines Neustarts sind auch andere Aktionen denkbar. So könnte man z. B. den Neustart x–mal versuchen und nach dem x–ten Fehlschlag eine alternativen Workflow starten.

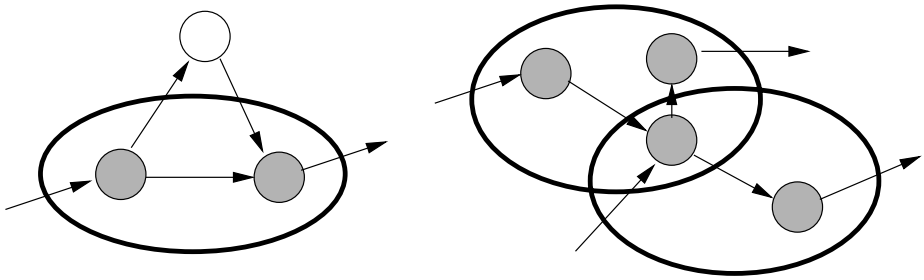
Eine Workflow–Transaktion muß zusammen mit dem Workflow in der Modellierungskomponente des Workflowsystems spezifiziert werden. Eine Workflow–Transaktionen wird durch eine Sphäre modelliert. Alle Aktivitäten, die an Transaktion teilnehmen sollen, müssen in eine Sphäre aufgenommen werden. Die Modellierungskomponente muß auch dafür sorgen, daß folgende strukturellen Bedingungen für die Sphäre eingehalten werden:

- Alle Aktivitäten der Sphäre sind Teilnehmer an der Workflow–Transaktion. Die Modellierungskomponente muß daher prüfen, ob die an der Sphäre teilnehmenden Aktivitäten die entsprechenden Randbedingungen erfüllen, wie sie in Abschnitt 5.4.3 beschrieben werden.

- Eine Schachtelung von Sphären ist erlaubt und dient zur Verkleinerung des Bereichs, der zurückgesetzt werden soll. So kann ein feineres Recovery–Granularität unterstützt werden. Da einmal spezifizierte Workflows in Form von Subprozessen wiederverwendet werden können und in diesen Prozessen auch Sphären definiert sein können, benötigt man auch aus diesem Grund die Möglichkeit geschachtelter Workflow–Transaktionen.

Innerhalb der Aktivitäten können durch Anwendungsprogramme neue Transaktionen begonnen und wieder beendet werden. Diese Transaktionen sind dann als in die Workflow–Transaktion geschachtelte Transaktionen zu realisieren.

- Die partielle Überlappung von Sphären ist nicht möglich. Eine Aktivität darf damit immer nur an höchstens einer Sphäre teilnehmen. Partiell überlappende Sphären können durch eine Vereinigungsoperation in eine einzige Sphäre überführt werden. Partiell überlappende Sphären erweisen sich somit als unnötig. Der Effekt der feineren Transaktionsgranulat kann durch geschaltete Sphären ebenso erreicht werden.



(a) (b)

Abbildung 9: Verlassen und Wiedereintritt des Kontrollflusses (a) und partiell überlappende Sphären (b) sind nicht erlaubt.

- Aus der Isolationseigenschaft der Sphäre ergibt sich, daß kein Pfad von einem Ausgang auf einen Eingang derselben Sphäre existieren darf.

  Angenommen, es gäbe einen solchen Pfad. Aufgrund der Isolation wird der Kontrollflußausgang erst nach Beendigung der Sphäre aktiv. Die Sphäre kann aber noch nicht beendet sein, da der Kontrollflußeingang auf diesem Pfad noch nicht aktiv sein kann, d. h. es gibt eine nicht beendete Aktivität in der Sphäre. Die Sphäre kann noch nicht beendet sein. Es gibt einen Widerspruch, daher darf kein solcher Pfad existieren.

5.4.3 Anforderungen an ACID–Aktivitäten

Damit Aktivitäten an einer Workflow–Transaktion teilnehmen können, müssen sie bestimmten Voraussetzungen genügen. Wir unterscheiden deshalb zwischen normalen Aktivitäten und sogenannten *ACID–Aktivitäten*, die diese Voraussetzungen erfüllen.

- Die Aktivitäten dürfen nur über Resource Manager auf Daten zugreifen. Wenn sie Daten selbst verwalten, müssen die Aktivitäten selbst als Resource Manager auftreten. Eine Aktivität, die als Resource Manager agieren möchte, muß alle notwendigen Funktionen implementiert haben, um selbständig ein Recovery ausführen zu können.

- Die Aktivitäten müssen eine geeignete Schnittstelle aufweisen, über die bestimmte Funktionen der Aktivitäten ausgelöst werden können (z.B. das Recovery, das Commit, das Rollback). Ebenso müssen sie Schnittstellen für die Teilnahme an einem 2–Phasen–Commit–Protokoll zu ermöglichen. Damit erreicht der Transaktions–Service eine gemeinsame Übereinkunft aller Beteiligten Resource Manager über den Erfolg oder Mißerfolg der Transaktion. Die angebotene Schnittstelle muß zu dem im Workflowsystem verwendeten Transaktions–Service passen.

- Die Menge der von mehreren konkurrierenden Aktivitäten benutzten Daten müssen relativ klein sein, damit sich Sperren oder andere Synchronisationsprotokolle nicht auf die die Kooperationsfähigkeit der Workflows auswirkt.

- Wenn in den Aktivitäten physische Operationen aufgeführt werden, bedarf es der Verwendung eines erweiterten Resource Manager, der *Physical–Resource–Manager* (PRM) genannt wird [Sch93]. "Real actions" haben im Gegensatz zu Datenbankoperationen die Eigenschaft, daß ihre Auswirkungen sofort sichtbar werden und daß diese Auswirkungen oft nicht mehr rücksetzbar sind. Das klassische Beispiel für eine solche Operation ist das Bohren eines Loches. Unter der Annahme, daß nur eine physische Operation in der Workflow–Transaktion stattfindet, kann das Recovery eines PRM so aussehen: Wenn der Abbruch der Transaktion vor der physischen Operation stattfindet, dann muß der PRM wie ein regulärer RM reagieren. Es wird ein Rollback durchgeführt. Wenn der Abbruch nach der Ausführung der physischen Operation stattfindet, wird wiederum ein normales Rollback durchgeführt. Die physische Operation wird dabei nicht rückgesetzt. Beim wiederholten Starten der Transaktionen wird dann die bereits in der vorherigen Transaktion ausgeführte physische Operation ausgelassen. Wenn der Abbruch während der physischen Operation stattfindet, dann muß eine anwendungsspezifische Fehlerbehebungsmaßnahme durch den PRM getroffen werden.

  Wenn man mehrere physische Operation innerhalb einer Workflow–Transaktion benutzen will, wird die Komplexität des Recovery deutlich höher.

  Für den Fall, daß die Auswirkungen physischer Operation zurückgehalten werden können, z.B. das Verschicken einer Email oder eines Briefes, ist es Aufgabe des Resource Managers, dafür zu sorgen, daß die Operation erst in der Propagierungs–Phase des Zwei–Phasen–Commit–Protokolls am Ende der Workflow–Transaktion ausgeführt wird. So wird die Email solange verzögert, bis die gesamte Transaktion erfolgreich beendet wird. Die Operation darf dann allerdings nicht mehr fehlschlagen.

5.5 Einsatzgebiete von Workflow–Transaktionen

Durch den alleinigen Einsatz von Workflow–Transaktionen kann man das Ziel eines fehlertoleranten Ablaufs von Geschäftsprozessen nicht erreichen. Nur in einem eng beschränkten Einsatzfeld erweist sich das Konzept der Workflow–Transaktionen als hilfreich. Ein solches Einsatzfeld können z. B. stark datenbankorientierte Geschäftsprozesse sein. Dort sind im allgemeinen schon die entsprechenden Resource Manager mit den standardisierten Schnittstellen vorhanden.

Wichtig erweist sich dieses Konzept auch bei der Verwendung von sogenannten *business objects* als Aktivitäten. Business–Objekte sind Repräsentanten für alle in einem Geschäftsprozeß vorkommenden Objekte. Dies können Programme, Personen oder Daten in der traditionellen Sichtweise sein. Business–Objekte werden zur Zeit in der BOMSIG special interest group der OMG (Object Management Group) standardisiert. Diese Objekte bieten Methodenaufrufe an, um Operationen auf Daten durchzuführen. Die Methoden sind oftmals von kurzer Dauer und werden automatisch ausgeführt, d. h. es gibt kaum manuelle Interaktion. Diese Methodenaufrufe können in einer Workflow–Transaktion als Operationen eingebunden werden.

Als Realisierungsansätze für Workflow–Transaktionen bieten sich die Standards für verteilte Transaktionen an: Es kommt die X/Open Spezifikation for Distributed Transaction Processing (DTP) XA und der Object Transaction Service (OTS) [OTS94] der Object Management Group (OMG) in Frage. Insbesondere im Verbund mit dem Einsatz der Business–Objekte kann sich OTS als sinnvoll erweisen. Die Aktivitäten müssen die in den Standards spezifizierten Funktionen als API anbieten.

Bei Einsatz von OTS müssen die Anwendungsprogramme in den Aktivitäten ein Objektinterface besitzen. Das Workflowsystem muß sich der Dienste eines Corba–kompatiblen Objekt Request Brokers (ORB) bedienen, um die Anwendungsprogramme in den Aktivitäten aufzurufen.

Eine prototypische Implementierung dieses Ansatzes wurde im Rahmen des Projekts Workflow–Management im Software–Labor der Universität Stuttgart durchgeführt. Genauere Beschreibungen dieses Projekts finden sich in [BS96] und [SB96].

6 Kompensations–Sphären

Das Konzept der Workflow–Transaktionen stellt hohe Anforderungen an die Funktionalität der Aktivitäten, die an einer Workflow–Transaktionen teilnehmen. In vielen Fällen wird aber ein Workflowsystem mit Aktivitäten eingesetzt, die nicht diesen Anforderungen entsprechen. Oftmals haben die Anwendungsprogramme, die in den Aktivitäten aufgerufen werden, kein Wissen darüber, daß sie im Rahmen eines Geschäftsprozesses eingesetzt werden. Sie können daher auch nicht auf die Bedürfnisse der Workflow–Transaktionen abgestimmt werden.

Aus diesem Grund ist ein weiteres Konzept bei der Bearbeitung von Workflows nötig, das den Ablauf von Workflows fehlertoleranter macht, ohne diese hohen Anfor-

derungen zu besitzen. Der Ansatz der *Kompensations–Sphären* [Ley95] stellt kaum noch Anforderungen an die Aktivitäten. Im Gegenzug dazu muß man aber auf die Isolationseigenschaft und die garantierte Konsistenz der Anwenderdaten bei der Ausführung eines Workflow verzichten. Die Eigenschaft der Atomizität und der Dauerhaftigkeit bleiben erhalten. Das Mittel zur Erreichen dieses Ziels sind Kompensationsaktivitäten.

6.1 Begriffe

Eine **Kompensationsaktivität** unterscheidet sich nur durch ihre Verwendung von einer normalen Aktivität. Jede Kompensationsaktivität muß einer normalen Aktivität bzw. einer Sphäre zugeordnet sein und soll alle Auswirkungen der normalen Aktivität bzw. der gesamten Sphäre beseitigen. Das Workflowsystem bietet außer dem Aufruf der Kompensationsaktivität keine weitere Unterstützung, um dieses Ziel zu erreichen. Wegen der fehlenden Isolationseigenschaft der Kompensations–Sphären muß die Kompensationsaktivität auch dafür sorgen, daß die Datenänderungen der normalen Aktivität eventuell schon von anderen Aktivitäten gelesen und zur Weiterverarbeitung benutzt worden sind. Die Kompensationsaktivität muß auch in diesen Fällen geeignete Maßnahmen treffen.

Der Begriff **Sphäre** [1] bezeichnet auch hier eine nicht unbedingt zusammenhängende Menge von Aktivitäten in einem Workflow.

Definition: *Kompensations–Sphäre* (engl.: compensation sphere)

> *Eine Kompensations–Sphäre ist eine Menge von Aktivitäten, die entweder alle im Zustand 'erfolgreich' oder alle im Zustand 'kompensiert' sind, wenn der Kontrollfluß die Sphäre verlassen will.*

Das Zustandsdiagramm für die Aktivitäten muß daher wie in Abbildung 10 modifiziert werden. Es wird zusätzlich ein Zustand KOMPENSIERT eingeführt. Dieser Zustand ist weitgehend äquivalent zum Zustand STARTBAR mit dem Unterschied, daß mindestens eine Bearbeitung und eine Kompensation der Aktivität stattgefunden hat. Der Endzustand ERFOLGREICH wird dann verlassen, wenn andere Aktivitäten der Sphäre kompensiert werden müssen. Eine Sphäre ist dann kompensiert, wenn alle Aktivitäten der Sphäre kompensiert sind.

6.2 Das Konzept der Kompensations–Sphären

Zur Modellierungszeit werden Aktivitäten zu einer Sphäre zusammengefaßt. Aus der Sicht der Aktivitäten außerhalb der Sphäre werden die Aktivitäten zu einer atomaren Ausführungseinheit. Zum Ausführungszeitpunkt des Workflows sorgt das Workflowsystem dafür, daß die Sphäre von den Aktivitäten außerhalb der Sphäre isoliert wird,

[1] siehe Definition Seite 29

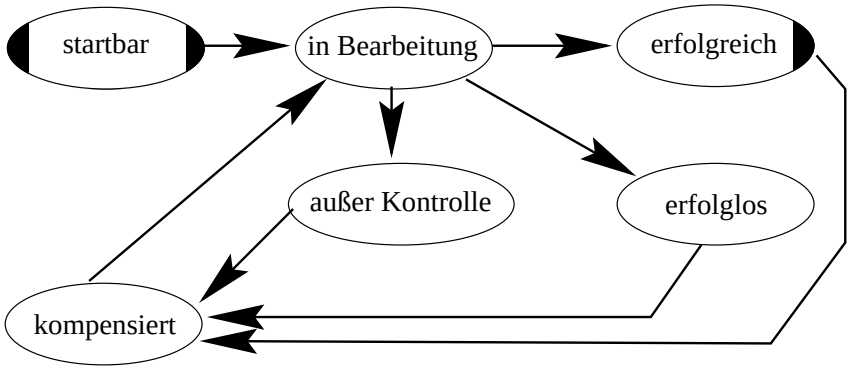


Abbildung 10: Das Zustandsdiagramm für Aktivitäten bei Kompensations–Sphären

indem der Kontroll– und Datenfluß bis zum Ende der Sphäre verzögert wird. Aktivitäten außerhalb der Sphäre können somit keine Zwischenergebnisse von Aktivitäten innerhalb der Sphäre über das Workflowsystem bekommen. Da die Anwenderprogramme aber weiterhin auf beliebigen Datenbeständen arbeiten können, die nicht im Kontrollbereich des Workflowsystem liegen müssen, können Zwischenergebnisse durchaus von anderen Programmen verarbeitet werden. Die Isolation kann daher nicht durch das Workflowsystem garantiert werden.

Wenn ein Fehler auftritt, werden alle bereits beendeten Aktivitäten kompensiert und alle Aktivitäten abgebrochen, die noch in Bearbeitung sind. Danach kann die Sphäre entweder neu gestartet werden oder es wird ein alternativer Weg im Workflow eingeschlagen, wie im vorherigen Kapitel beschrieben.

Dieses Konzept fordert schwächere Voraussetzungen an die teilnehmenden Aktivitäten als die Workflow–Transaktionen. Die Aktivitäten müssen kompensierbar sein, d. h. zu jeder Aktivität A in der Sphäre muß eine Kompensationsaktivität A^{-1} existieren, die die Auswirkungen der Aktivität A rückgängig macht. Wie diese Kompensierbarkeit erreicht wird, liegt ganz in der Verantwortung des Erstellers der Aktivität. Das Konzept der Kompensations–Sphären bietet dazu außer dem Aufruf der Kompensationsaktivität keine weitere Unterstützung an. Ergänzend kann die Anforderung aufgestellt werden, daß jede Aktivität (bzw. das Anwenderprogramm innerhalb der Aktivität) an ihrer Schnittstelle eine Funktion anbieten muß, mit der die Aktivität vorzeitig abgebrochen werden kann, ohne daß dadurch der Anwenderdatenbestand in einem inkonsistenten Zustand hinterlassen wird. Mit dieser Anforderung kann eine Optimierung angewendet werden, die mit Hilfe des vorzeitigen Abbruchs der Aktivität die Rücksetzzeit der Sphäre verkürzt, indem unnötige Arbeit nach Auftreten eines Fehlers in der Sphäre verhindert wird. Die Aktivität A muß nach ihrem Ende oder nach einem Abbruch erneut gestartet werden können, ohne daß dadurch ein Fehler auftritt.

Die Anforderungen an die Kompensationsaktivität sind dafür aber um so ausgeprägter. Neben der Existenz die Aktivität muß auch gefordert werden, daß die Aktivität niemals fehlschlägt. Die Aufgabe der Kompensationsaktivität, das Beseitigen

der Auswirkungen der Aktivität A, wird dabei in keiner Weise durch das Workflowsystem unterstützt. Der Kompensationsaktivität muß diese Aufgabe völlig selbständig und korrekt durchführen.

Eine Sphäre wird dann zurückgesetzt, wenn eine der Aktivitäten in einen Fehlerzustand (siehe Abbbildung 10) überführt wird. Das Zurücksetzen einer Sphäre sollte auch über den interaktiven Aufruf einer Funktion möglich sein, die dem Benutzer des Workflowsystem an der Bedienoberfläche angeboten wird. Über diese Funktion kann manuell ein Zurücksetzen ausgelöst werden, das eventuell durch spezielle Rechte abgesichert werden kann.

Nach der Auslösung der Rücksetzvorgangs ist eine Entscheidung möglich, ob die Sphäre bis zu ihrem Beginn oder zu einem weniger weit zurückliegenden Punkt zwischen den Aktivitäten zurückgesetzt werden soll. Wenn diese Auswahl, wohin zurückgesetzt werden soll, dem Benutzer interaktiv überlassen wird, dann hat man eine Art "Undo"–Funktion im Workflow realisiert. Man kann die letzten Vorgangsschritte (Aktivitäten) innerhalb der Grenzen der Sphäre rückgängig machen und dann an dem gewünschten Punkt weiterarbeiten.

Für die Ausführungsreihenfolge der Kompensationsaktivitäten gibt es verschiedene Möglichkeiten. Die Kompensationsaktivitäten können alle parallel ausgeführt werden, da alle dazu notwendigen Daten schon während der Ausführung der normalen Aktivitäten gespeichert werden. Falls ein solches Verhalten nicht gewünscht ist, können die Kompensationsaktivitäten in der umgekehrter Reihenfolge wie die normalen Aktivitäten ausgeführt werden. Diese umgekehrte Reihenfolge kann durch Umdrehen der Kontrollflußbeziehung bestimmt werden oder durch Auswerten der Startzeitpunkte aller normalen Aktivitäten. Eine dritte denkbare Methode besteht in einer frei spezifizierbaren Reihenfolge, die während der Modellierung des Prozesses festgelegt werden muß.

Die Schachtelung von Sphären muß, wie bei den Workflow–Transaktionen auch, aufgrund der Wiederverwendung von Workflowteilen erlaubt sein. Daraus ergibt sich die Notwendigkeit, auch ganze Sphären rückgängig machen zu müssen. Zusätzlich zu dem Kompensieren aller Einzelaktivitäten einer Sphäre kann man auch die Möglichkeit schaffen, mit einer einzigen Kompensationsaktivität eine ganze Sphäre auf einmal zu kompensieren. Dazu müssen dann zu Sphären auch Kompensationsaktivitäten einführt werden.

Ein Überlappen von Sphären bedeutet, daß eine Aktivität an mehr als an einer einzigen Kompensations–Sphäre teilnimmt. Wenn man dies zuläßt, handelt man sich das Problem der kaskadierenden Kompensation weiterer Sphären ein. Da Überlappung keinen weiteren Vorteil als eine feinere Abstufung der Sphären bringt und dieser Vorteil auch über die Schachtelung von Sphären erlangt werden kann, kann das Überlappen ohne Verlust an Funktionalität verboten werden.

6.3 Vergleich zwischen Transaktions–Sphären und Kompensations–Sphären

Das Konzept der Kompensations–Sphären unterscheidet sich in einigen wesentlichen Punkten von dem Konzept der Workflow–Transaktionen. Kompensations–Sphären stellen hauptsächlich die Eigenschaft der Atomizität bereit. Dabei werden keine besonderen Anforderungen an die Aktivitäten gestellt. Das Workflowsystem führt keine undurchdringbare Isolation der Sphären durch. Zwischenergebnisse aus den Aktivitäten in der Sphäre können von allen Aktivitäten auf Kosten der Konsistenz genutzt werden. Es wird keine standardisierte Schnittstelle zur Einbindung in einen transaktionalen Kontext gefordert. Die Aktivitäten müssen nicht an einem Zwei–Phasen–Commit–Protokoll teilnehmen können. Die Aktivitäten müssen keine Resource Manager sein und ihre Änderungen auf den Daten rückgängig machen können. Sie müssen kein Recovery implementiert haben.

Ein Schwachpunkt der Kompensations–Sphären ist die Tatsache, daß für eine Garantie der Rücksetzbarkeit einer Sphäre gefordert werden muß, daß Kompensationsaktivitäten nicht fehlschlagen dürfen. Diese Forderung ist aber nur schwer verwirklichbar.

Die Dauerhaftigkeit der Ergebnisse der Aktivitäten kann durch das Workflowsystem nur dann gewährleistet werden, wenn es auch die Kontrolle über die in den Aktivitäten bearbeiteten Daten hat. Das ist aber nur vor und nach der Bearbeitung einer Aktivität der Fall. Wenn das System aber während der Bearbeitung einer Aktivität abstürzt, gehen alle Änderungen verloren, die in den laufenden Aktivitäten gemacht wurden. Auf diese Weise kann ein inkonsistenter Datenzustand entstehen, der eventuell zur Folge hat, daß die anschließende Wiederholung der Aktivität fehlschlagen kann.

Kompensations–Sphären können entweder über die vorhandenen Mittel der Workflow–Spezifikationssprache modelliert werden oder durch eine direkte Unterstützung in der Workflow–Engine realisiert werden. Beim Modellierungsansatz werden die Kompensationsaktivitäten wie normale Aktivitäten behandelt. Der Kontrollfluß zwischen den Kompensationsaktivitäten muß explizit festgelegt werden. Im zweiten Ansatz wird der Zustand einer Sphäre durch die Engine verwaltet. Die Engine stößt bei Bedarf die Kompensationsaktivitäten an. Es ist keine weitere Spezifikationen des Kontrollflusses nötig, es sei denn, man möchte die Reihenfolge der Kompensationsaktivitäten explizit festlegen.

Der Vorteil der Kompensations–Sphären liegt darin, daß bei einer Realisierung nur Änderungen im Workflowsystem nötig sind. Auf die Realisierung der Aktivitäten hat dieses Konzept keine Auswirkungen.

7 Ausblick und Zusammenfassung

Workflow–Transaktionen überdecken nur einen kleinen Teil der in Kapitel 3 aufgestellten Anforderungen. Nur durch den gemischten Einsatz der in den darauffolgenden Kapiteln vorgestellten Lösungsansätze können fehlertolerante Workflows geschaffen werden. Insbesondere die Kombination von Workflow–Transaktionen und dem Konzept der Kompensationsaktivitäten erhöht die Fehlertoleranz:

7.1 Ausblick auf Erweiterungen

Workflow–Transaktionen verwenden das aus den Datenbanken entliehene Konzept der ACID–Transaktionen, um die Abwicklung von Workflows stabiler zu machen. Das Konzept kann aber nicht auf ganze Workflows angewandt werden, da dann die Probleme langandauerender Abläufe auftreten. Das Anwendungsgebiet der Workflow–Transaktionen muß daher auf kleine Workflowabschnitte beschränkt werden. Es bleibt aber weiterhin die Anforderung offen, ganze Workflows oder zumindest große Teile des Workflows fehlertolerant ablaufen zu lassen. Beim Auftreten eines Fehlers möchte man gewisse Teile oder sogar den ganzen Workflow rückgängig machen können, um den Workflow von einem konsistenten Zustand aus fortsetzen zu können. Die Auswirkungen des Workflows sollten dabei rückgängig gemacht oder kompensiert werden.

Diese Anforderung kann durch eine Verbindung von Workflow–Transaktionen mit dem Konzept der Kompensationsaktivitäten erreicht werden. Workflow–Transaktionen werden dabei als Grundbausteine für stabile Workflows einsetzt. Auf Workflowebene können sie damit als atomare Schritte angesehen werden. Mit dem erfolgreichen Abschluß einer Workflow–Transaktion werden die Ergebnisse externalisiert, d. h. andere Aktivitäten können mit den Ergebnissen weiterarbeiten. Falls nach dieser Freigabe ein Fehler auftreten sollte, der ein Rückgängigmachen der bisherigen Ergebnisse des Workflows erforderlich machen würde, kann man das Konzept der Workflow–Transaktionen nicht mehr nutzen. Eine Lösung dieses Problems erhält man, wenn man jeder Sphäre eine Kompensationsaktivität zuordnet, die semantisch die Ergebnisse rückgängig macht. Ein Workflow kann dann als fehlertolerant angesehen werden, wenn es möglich ist, konsistent aus jeden Zwischenzustand des Workflows wieder auf den Anfangszustand zu kommen[2]. Dabei sollte es auch korrekte Zwischenzustände geben, auf die man alternativ zurücksetzen kann, um nicht zuviel Arbeit zu verlieren, wenn der Workflow weiterbearbeitet wird.

Robustheit kann erreicht werden, indem alle Aktivitäten eines Workflows in Sphären eingebunden werden und zu jeder Sphäre eine Möglichkeit bereitgestellt wird, die Auswirkungen der Sphäre zu kompensieren. Wenn eine Aktivität nicht in einer Sphäre eingebunden ist, darf sie nach ihrer Ausführung keine inkonsistenten Auswirkungen hinterlassen.

Im realen Einsatz von Workflowsystemen darf diese scharfe Anforderung an fehler-

[2] siehe dazu auch das ConTracts Modell in [WR92]

tolerante Workflows abgeschwächt formuliert werden. Hier erweist es sich als sinnvoll, nur noch zu fordern, daß die wesentlichen Teile eines Workflows fehlertolerant ablaufen müssen. Es wird in vielen Geschäftsprozessen Aktivitäten geben, die nicht abgesichert werden müssen, da sie keine kritischen Tätigkeiten enthalten, die inkonsistente Daten zurückliessen, wenn sie fehlschlügen. Diese Aktivitäten werden unwesentlich genannt. Durch eine Kennzeichnung der Aktivitäten, welche für den Erfolg des Geschäftsprozesses wesentlich sind und welche nicht, kann beim Auftreten eines Fehlers entschieden werden, ob und welche Maßnahmen ergriffen werden müssen [EL95]. Unwesentliche Aktivitäten müssen dann nicht an Workflow–Transaktionen teilnehmen und benötigten keine Kompensationsaktivität. Der Vorteil dieser Methode ist der, daß bei dem Auftreten von Fehlern in einem unwesentlichen Teil des Workflows keine Fehlerbehandlung stattfindet muß, die zum Verlust von Arbeit führen könnte. Allerdings müssen damit auch die Garantien für eine fehlertolerante Ausführung auf die wesentlichen Teile des Workflows beschränkt werden.

Flankierend zu der Realisierung dieses Konzeptes sollten auch Konzepte wie Ad–hoc–Modifikationen und alternativer Pfade in das Workflowsystem integriert werden. Weiterhin müssen auf Systemebene die notwendigen Maßnahmen zur Erlangung von Fehlertoleranz realisiert sein. Eine verläßliche Datenübertragung, die durch Transaktionen abgesichert ist, sowie die redundante Auslegung der einzelnen Softwarekomponenten sind hier denkbar.

Durch den hier vorgeschlagenen Weg wird die Bearbeitung von Workflows zuverlässiger und fehlertoleranter. Das Workflowsystem kann flexibler eingesetzt werden und erschließt sich damit ein größeres Anwendungsfeld.

7.2 Zusammenfassung

Zu Beginn dieses Berichts wurden allgemeine Anforderungen an ein robustes und fehlertolerantes Workflowsystem aufgestellt. In einem Fehlermodell für Workflow–Management–Systeme wurden verschiedene Klassen von Fehlerbehandlungen eingeführt und Fehlerarten nach ihrem Ort des Auftretens klassifiziert. Geordnet nach den Fehlerbehandlungsarten transparent, automatisch und manuell wurden anschließend die allgemeinen Anforderungen in funktionale Anforderungen verfeinert. Teile diese Anforderungen können durch die vorgeschlagenen Lösungskonzepte erfüllt werden: Die Modellierung der Fehlerbehandlung mit vorhandenen Sprachmitteln ist einfach, führt aber zu aufwendigen Workflows. Wenn Aktivitäten wegen eines Absturzes keinen Rückgabewert mehr liefern können, versagt diese Methode. Alternative Aktivitäten und alternative Pfade im Workflow erhöhen die Flexibilität der modellierten Fehlerbehandlung im Workflow. Mit dem Konzept der Ad–hoc–Modifikationen können hochflexible Workflow realisiert werden. Insbesondere kann man mit diesem Konzept auf unerwartete Gegebenheiten reagiert werden.

Dem Konzept der Workflow–Transaktionen wurde ein eigenes Kapitel gewidmet. Darin wurde zuerst das aus dem Datenbankbereich übernommene Konzept der ACID–Transaktionen erläutert. Die direkte Übernahme dieses Konzepts in Workflowsysteme

bereitet allerdings Probleme, die insbesondere mit der langen Laufzeit von Workflows zusammenhängen. Dieses Problem tritt auch bei speziellen Datenbankenanwendungen auf und wurde dort mit Hilfe erweiterter Transaktionsmodelle angegangen. Einige der in der Literatur beschriebenen Modelle wurden in diesem Bericht mit ihren wesentlichen Merkmalen umrissen.

Durch die Einschränkung des Einsatzbereichs von ACID–Transaktionen auf eine kleine Menge spezieller, sogenannter ACID–Aktivitäten erhält man das Konzept der Workflow–Transaktionen. Eine Menge von ACID–Aktivitäten kann zu einer Sphäre zusammengefaßt werden. In einer Sphäre läuft genau eine Workflow–Transaktion ab, die dieselben Eigenschaften wie eine ACID–Transaktion in Datenbanken hat.

Da die Anforderungen an ACID–Aktivitäten sehr hoch sind und diese üblicherweise von existierenden Aktivitäten nicht erfüllt werden, wurde ein weiteres Konzept vorgestellt. Kompensationssphären erlauben das Zurücksetzen der Sphäre auf den Anfangszustand durch die Ausführung von Kompensationsaktivitäten. Die Reihenfolge der auszuführenden Kompensationsaktivitäten kann dabei auf mehrere Arten festgelegt werden.

Abschließend wurde kurz skizziert, wie ein Verbund der Konzepte Workflow–Transaktionen und Kompensationsaktivitäten die Abwicklung eines Workflows fehlertoleranter machen kann.

Literatur

[BS96] Bildstein, Hubert ; Schreyjak, Stefan: *Der Einsatz von Workflow–Transaktionen in FlowMark* Universität Stuttgart, Software–Labor. 1996. – Fakultätsbericht Nr. 1996/18, Software–Labor Bericht SL–4/96

[EL95] Eder, J. ; Liebhart, W.: The Workflow Activity Model WAMO. **In:** *Proc. 3. Int. Conference on Cooperative Information Systems (CoopIS)*. Wien, 1995, S. 87–98

[GMS87] Garcia-Molina, Hector ; Salem, Kenneth: SAGAS. **In:** *Proceedings ACM SIGMOD*. San Francisco, 1987, S. 249–259

[GR93] Gray, Jim ; Reuter, Andreas: *Transaction Processing*. Morgan Kaufmann, 1993

[Ley95] Leymann, F.: Supporting Business Transactions via Partial Backward Recovery in Workflow Management Systems. **In:** Lausen, G. (Hrsg.): *Proc. Datenbanksysteme in Büro, Technik und Wissenschaft*. Berlin : Springer, Maerz 1995, S. 51–70

[Mos81] Moss, J. E. B. *Nested Transactions: An Approach to Reliable Distributed Computing*. MIT Laboratory for Computer Science, Cambridge, Massachusetts 1981 – PhD Thesis

[OTS94] Object Management Group (OMG): *Object Transaction Service*. August 1994. – Document No. 94.8.4

[SB96] Schreyjak, Stefan ; Bildstein, Hubert: *Beschreibung des prototypisch implementieren Workflowsystems Surro* Universität Stuttgart, Software–Labor. 1996. – Fakultätsbericht Nr. 1996/19, Software–Labor Bericht SL–5/96

[Sch93] Schmidt, Ursula: Transaktionskonzepte in der Fertigung. **In:** *Proc. Datenbanksysteme in Büro, Technik, Wissenschaft*. Braunschweig, März 1993

[WR92] Wächter, Helmut ; Reuter, Andreas: The ConTract Model. **In:** Elmagarmid, A. K. (Hrsg.): *Database Transaction Models for Advanced Applications*. San Mateo : Morgan Kaufmann, 1992, Kapitel 7, S. 219–263

[ZNBB94] Zhang, Aidong ; Nodine, Marian ; Bhargava, Bharat ; Bukhres, Omran: Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems. **In:** *ACM SIGMOD*, Association of Computing Machinery, 1994, S. 67–78