Transaktionskonzepte in der Fertigung

Ursula Schmidt
Universität Stuttgart
IPVR
Breitwiesenstr.20-22
7000 Stuttgart 80
email:ursula.schmidt@informatik.uni-stuttgart.de

Abstract: In diesem Artikel wird im Rahmen einer regulären Architektur zur Transaktionsverarbeitung ein erweiterter Typ von Resource Manager vorgestellt, ein Physical Resource Manager, der Fertigungsoperationen unter Transaktionsschutz anbietet. Dabei bildet eine Transaktion einen Bearbeitungsschritt, z.B. Bohren oder Fräsen, eines Arbeitsplans ab und enthält maximal eine physische Operation. Bei der Spezifikation eines dazu befähigten Physical Resource Managers (PRM) muß besonders berücksichtigt werden, daß dessen Operationen oft nicht rücksetzbar sind. Ein solcher PRM kann maximal nur ein *Reset* im Gegensatz zu einem regulären Rollback ausführen. Dies schließt den klassischen Ansatz der Backward Recovery zur Fehlerbehandlung aus und erfordert eine zusätzliche Komponente zur Fehlerbehandlung, den Physical Recovery Manager. Insgesamt wird eine Neufassung des Begriffes des Ununterbrechbarkeit notwendig.

1 Einleitung

In diesem Kapitel soll der Einsatz von Transaktionen in der Fertigung motiviert werden und eine Abgrenzung zu bisherigen Arbeiten der Fehlertoleranz in Informatik und Fertigung gezogen werden.

1.1 Einführung

Die Bearbeitung von Werkstücken und der Einsatz von automatisch gesteuerten Maschinen blikken auf eine lange Tradition zurück. Schon im Jahre 1808 wurden gelochte Blechkarten zur automatischen Steuerung von Webmaschinen benutzt. Der Sprung zur Technik der numerischen Steuerung erfolgte 1952, als am M.I.T. erstmalig binär kodierte Lochstreifen verwendet wurden, um eine Werkzeugmaschine mit Daten zu versorgen. Die Entwicklung und der technologische Fortschritt nahmen jedoch von Anfang an ihren Ausgangspunkt an der jeweiligen Maschine - Aspekte einer maschinenübergreifenden und systemeinheitlichen Steuerung wurden erst in den letzten Jahren mit dem Aufkommen flexibler Fertigungssysteme Gegenstand wissenschaftlicher Untersuchungen. Vor allem die Forderung nach einer "Just-in-time" Produktion macht jetzt eine Integration der betrieblichen Abläufe in ein allgemeines Systemkonzept notwendig. Während sich für die Kommunikation in der Fertigung inzwischen erste Standards durchsetzen konnten ([MAP, MMS]), fehlen allgemeine Mechanismen zur Steuerung und zur Fehlerbehandlung.

In: Tagungsband Datenbanken in Büro, Technik und Wissenschaft, 3. - 5. März 1993, Braunschweig, S. 104 - 123.

In der Informatik hingegen existieren vor allem im Bereich der verteilten Systeme, der Fehlertoleranz und der transaktionsverarbeitenden Systeme eine Vielfalt von Konzepten und Erfahrungen. Um damit ein Konzept zur einfachen und effektiven Fehlerbehandlung in einer Fertigungsumgebung definieren zu können, wollen wir ein definiertes failfast Verhalten der beteiligten Komponenten zugrundelegen. Dies heißt, wenn eine Komponente von einem Fehler betroffen ist, hält sie an, und zwar in einem wohldefinierten Zustand. Um dieses Verhalten zu gewährleisten, wollen wir Transaktionen verwenden, die ihre Vorteile und Nützlichkeit in transaktionsverarbeitenden Systemen und allgemein in verteilten Systemen zur Genüge unter Beweis gestellt haben.

Der Vorteil des klassischen "ACID"-Transaktionskonzeptes (siehe z.B. [Gr81]) liegt darin, daß zusammen mit einer sicheren, konsistenten Ausführung im logischen Einbenutzerbetrieb eine atomare Ausführung der Gesamtheit der zu einer Transaktion gehörenden Operationen garantiert wird. Daraus entwickelten sich Konzepte zur Behandlung langdauernder, komplexer Transaktionen. Hier ist zum Beispiel das Konzept der Sagas zu nennen [GS87], bei dem Einzeltransaktionen verkettet und explizite Kompensationstransaktionen für den Fehlerfall angegeben werden können. Daraus entstand das ConTract-Modell [WR91], das klassische Transaktionen als Einheiten für Bearbeitung und Kontrollfluß verwendet. Es ist keine Erweiterung des klassischen Transaktionskonzepts, sondern entspricht durch expliziten Kontrollfluß und die sichere Weitergabe von Information (Kontext) über Transaktionsgrenzen hinweg eher einem allgemeinen Programmiermodell.

Alle diese Ansätze legen klassische ACID-Transaktionen als Einheiten der Recovery zugrunde, und insbesondere das Prinzip der Ununterbrechbarkeit. Implizit wird dabei auch immer eine einheitliche Art der Recovery angenommen, nämlich Backward Recovery durch Zurücksetzen der betroffenen Daten auf ihren Ausgangszustand. Dies ist möglich, da die von einer Transaktion ausgeführten Operationen erst zum Zeitpunkt der Transaktion externalisiert werden und da die Operationen immer rücksetzbar und wiederholbar sind.

Erst in letzter Zeit gibt es Versuche, die Vorteile von Transaktionen in weitergehenden Anwendungsgebieten wie der Fertigung einzusetzen. Probleme ergeben sich durch die Andersartigkeit der zu schützenden Operationen, da die bisherigen Annahmen nicht mehr greifen. Die grundlegende Problematik bei der Einbindung von *real actions*, d.h. Operationen, die Auswirkungen auf die reale Welt haben, liegt einerseits darin, daß diese Auswirkungen unmittelbar externalisiert werden und andererseits, daß sie oft nicht rückgängig gemacht werden können. Selbst wenn man sich auf Operationen aus dem Fertigungsbereich beschränkt, findet man Operationen mit den verschiedensten Eigenschaften vor, die einer einheitlichen Recovery im Wege stehen. Dies erfordert eine Erweiterung der bisherigen Konzepte.

1.2 Abgrenzung zu bisherigen Arbeiten

Überlegungen zur Integration von real actions in Transaktionen reichen weit zurück. So wird in [Gr81] vorgeschlagen, real actions erst zum Zeitpunkt des Commit der umgebenden Transaktion durchzuführen. Dies setzt implizit voraus, daß real actions immer gelingen oder daß sie zumindest die Transaktion nicht entscheidend beeinflussen. In unserer Arbeit gehen wir jedoch davon aus, daß real actions fehlschlagen können und daß sich die umgebende Transaktion möglichst nach dem Ergebnis dieser real action auszurichten hat. In [Pa88] werden real actions in verschiedene semantische Klassen unterteilt, um davon ausgehend eine Fehlerbehandlung durchführen zu können[1]. Exemplarisch wurde ein System erstellt, das Eingaben und Ausgaben auf einem Bildschirm unter Transaktionsschutz stellt.

Zwei weitere Arbeiten untersuchen den Einsatz von Transaktionen im Bereich der Fertigungssysteme. In [WZ 87] werden in Anlehnung an den hierarchischen Aufbau eines Fertigungssystems Abstraktionsebenen einer Steuerungshierarchie eingeführt, darunter auch die Ebene der Fertigungstransaktionen. Hier wird die Eigenschaft, daß Transaktionen von einem definierten Zustand in einen anderen definierten Zustand überführen, benutzt, um bestimmte Eigenschaften eines Werkstücks zu beschreiben: eine Fertigungstransaktion überführt ein Teil mit genau festgelegten Eigenschaften in ein neues Teil mit genau festgelegten Eigenschaften. Transaktionen werden zur Modellierung von Vorgängen verwendet; auf Fehlerbehandlung wird nicht eingegangen. [JWZ88] beschreiben das Problem der Nichtrücksetzbarkeit physischer Operationen und motiviert die Notwendigkeit von Forward Recovery. Unter Einführung des Begriffs "point of no return" werden die Unterschiede zwischen Transaktionen in kaufmännischen und technischen Anwendungen herausgehoben.

Im Bereich der flexiblen Fertigungssysteme existieren zahlreiche Arbeiten zum Thema Fehlerdiagnose, jedoch kaum Arbeiten im Hinblick auf die Fehlerbehebung. Falls in Leitsystemen Fehlerbehandlung durchgeführt wird, liegt die Verantwortung für das Ergreifen korrekter Maßnahmen beim Maschinen- und Leitsystembediener. Nach [DIN] ist Qualitätssicherung die "Gesamtheit der Tätigkeiten des Qualitätsmanagements, der Qualitätsplanung, der Qualitätslenkung und der Qualitätsprüfungen". Die Qualitätssicherung in Leitsystemen befaßt sich damit, Fehler im Vorfeld oder während der Produktion zu vermeiden oder fehlerhafte Teile zu erkennen und auszusortieren. Hier haben sich folgende Vorgehensweisen und Begriffe etabliert: Bei der Fehlermöglichkeits- und Einflußanalyse (FMEA) wird versucht, schon vor der Produktion nach möglichen Fehlerquellen zu suchen und diese zu eliminieren. Dies ist sowohl für das Produkt als auch für den Prozeß möglich. Ein ähnlicher Ansatz findet sich beim Quality Function Deployment (QFD oder auch "House of Quality"): Ausgehend von Kundenanforderungen wird das Produkt auf Schwachstellen analysiert. Dagegen soll Statistical Process Control (SPC) im laufenden Produktionsbetrieb systematische Abweichungen von Prozeßsollwerten feststellen und diesen gegensteuern. Dieses Prinzip schließt auch die Überwachung der Werkzeugabnutzung und das rechtzeitige Ersetzen verbrauchter Werkzeuge ein. Total Quality Management (TQM) schließlich hat zum Ziel, die Gesamtheit aller unternehmerischen Aktivitäten unter Qualitätsgesichtspunkten zu optimieren. Unser Ansatz hingegen geht von einem schon eingetretenen und erkannten Fehler aus; im Verlauf dieses Artikels wird beschrieben, wie unter Verwendung des Transaktionskonzeptes die Auswirkungen des Fehlers transparent für die Anwendung umgangen werden und wie weitergearbeitet werden kann.

---

[1] Wir haben diesen Ansatz um eine generische Klasse und orthogonale Unterscheidungskriterien erweitert; dieses dient als logisches Grundgerüst für den in diesem Artikel nicht näher beschriebenen Physical Recovery Manager.

Auf dem Gebiet der Roboteranwendungen existieren zahlreiche Arbeiten zum Thema Fehlererkennung, -diagnose und -behebung. Es wird von autonomen Robotern ausgegangen und es werden hauptsächlich Methoden der Künstlichen Intelligenz eingesetzt. [Do87] untersucht, wie mit Fehlern im Weltmodell eines Roboters, d.h. mit Planungsfehlern, verfahren werden kann. Andere Arbeiten befassen sich mit der Ausnahmebehandlung für Roboter, wobei fest vorgegebene Ersatzpläne vorgesehen sind: In [MH88] werden Roboterausfälle entweder durch Neuplanung für die verbleibenden Aufträge abgefangen oder durch das (dort nicht näher spezifizierte) Anbieten von Ersatzoperationen. Die Probleme der nicht unmittelbar entdeckbaren Fehler und der unvorhergesehenen Ereignisse werden ausführlich in [Gi90] behandelt. Alle diese Arbeiten definieren und benutzen ihre eigene Entwicklungsumgebung. Unser Ansatz konzentriert sich auf Vorgänge in allgemeinen Fertigungssystemen, speziell auf Werkzeugmaschinen. Dort wird eine flexible, dynamische Fehlerbehandlung eigesetzt und gleichzeitig eine automatisch gesicherte Datenkonsistenz geboten.

Dieser Artikel ist wie folgt aufgebaut: Kapitel 2 beschäftigt sich mit den Schnittstellen, die ein Physical Resource Manager (PRM) dem transaktionsverarbeitenden System anbieten muß, ebenso wie mit den relevanten Eigenschaften der Anwendung und deren Auswirkungen auf die Operationen, die eine Transaktion umfassen kann. Im dritten Kapitel wird die Frage behandelt, ob ein PRM Dienste oder reale Maschinen repräsentiert. Daraus werden erste Folgerungen für die Recovery und die Architektur eines PRM vorgestellt. Im vierten Kapitel wird dessen Recovery ausgehend vom physischen Aufbau beschrieben. Kapitel 5 schließt mit einigen zusammenfassenden Bemerkungen und einem Ausblick auf mögliche, folgende Forschungsaktivitäten.

2 Einbettung des PRM in eine transaktionsverarbeitende Architektur

Ein PRM als Teil eines transaktionsverarbeitenden Systems (TPS) muß sowohl die definierten Schnittstellen anderer Komponenten nutzen als auch eigene, definierte Schnittstellen nach außen anbieten. Als Teilnehmer einer globalen Transaktion muß er sowohl mit dem Transaction Manager, der die Koordination aller an einer Transaktion beteiligten Komponenten übernimmt, kooperieren als auch Anwendungen transaktionsgeschützte Dienste anbieten. Hier soll auf die wichtigsten dieser Aspekte eingegangen werden.

2.1 Eigenschaften eines PRM

Zunächst muß geklärt werden, worin sich ein PRM, der Fertigungsoperationen auf einem Werkstück unter Transaktionsschutz anbietet, von einem regulären Resource Manager unterscheidet. Real Actions im allgemeinen, und Fertigungsoperationen im speziellen, besitzen im Vergleich zu Datenbankoperationen die Eigenschaft, daß ihre Auswirkungen sofort sichtbar werden und daß diese Auswirkungen oft nicht rücksetzbar sind. Es kann also nicht vorausgesetzt werden, daß ein Zurücksetzen oder ein Kompensieren möglich sind. Um dennoch Transaktionsschutz zu erreichen, wird im Fehlerfall für eine Transaktion mit Fertigungsoperationen nicht der letzte konsistente Zustand, der Zustand vor Beginn der Transaktion, sondern der nächste konsistente Zustand angestrebt. Mit Hilfe von Forward Recovery wird versucht, die unterbrochene Transaktion erfolgreic zu Ende zu bringen. Es wird semantische Information über die physische Operation ausgewertet, um geeignete Fehlerbehebungsmaßnahmen zu bestimmen. Es muß aber klargestellt werden, daß es Fälle gibt, in denen auch Forward Recovery nicht hilft. Dann muß die Transaktion geregelt zurückgesetzt werden.

2.2 Schnittstellen zum Transaction Manager

Hier stellt sich als erstes die Frage, ob ein PRM seine im Vergleich zu regulären Resource Managern unterschiedliche Recovery externalisiert oder ob er dem (globalen) Transaction Manager eine Schnittstelle bietet wie jeder normale Resource Manager auch. Beim *externalisierten Recoveryverhalten* könnte ein PRM nach einem nicht behebbaren Fehler z.B. ein "physical abort" anstelle eines einfachen "abort" zurückgeben. Hier muß das TPS darauf eingerichtet sein, PRMs mit unterschiedlichen Verhalten zu koordinieren. Die Anwendung hingegen, d.h. der Anwendungsprogrammierer, muß nicht mit diesen Aspekten belastet werden. Die zweite Möglichkeit ist, daß der PRM zum TPS hin ein *reguläres Verhalten* wie ein klassischer RM an den Tag legt. Es ist dann Sache allein des PRMs, Befehle vom Transaction Manager intern in die entsprechenden, semantisch korrekten Aufrufe umzusetzen. Hier muß der Benutzer des TPS wissen, daß er auf Resource Managern arbeitet, die physische Operationen durchführen und ein anderes Verhalten beim Rücksetzen besitzen.

Im Endeffekt läuft dies darauf hinaus, daß entweder das TPS oder der Benutzer des TPS die Unterschiede in den Resource Managern kennen muß. Wir haben uns aus folgenden Gründen für die zweite Möglichkeit entschieden:

- Ein Benutzer einer Fertigungsumgebung weiß ohnehin, wie seine Umgebung beschaffen ist. Er geht nicht davon aus, daß nach dem Fehlschlagen eines Bearbeitungsvorgangs das Werkstück sich wieder im unbearbeiteten Zustand befindet. Er weiß, daß im Fehlerfall Fertigungsoperationen nicht einfach zurückgesetzt werden, da dies nicht generell möglich und oft auch nicht sinnvoll ist. Ist zum Beispiel das aktuelle Werkstück bei einem Transportvorgang beschädigt worden, macht es wenig Sinn, es zunächst zurückzutransportieren, um es dann erst wegzuwerfen.

- Ist jeder Resource Manager (regulärer RM und PRM) für die korrekte Ausführung der empfangenen Aufrufe eigenverantwortlich, können einheitliche Schnittstellen geboten werden, und damit können die unterschiedlichsten Resource Manager in ein (existierendes) TPS eingebunden werden. Andernfalls könnten schon geringfügige Änderungen in einem Resource Manager oder das Einbinden neuer Resource Manager einen Neuentwurf des gesamten Systems notwendig machen.

Ein PRM muß also am Zwei-Phasen-Commit-Protokoll für verteilte Transaktionen teilnehmen können.

2.3 Schnittstellen zur Anwendung

Die Schnittstellen zur Anwendung werden von der Wahl des TPS entscheidend beeinflußt. Wir wollen eine Architektur zur verteilten Transaktionsverarbeitung auf ihre Unterstützung für den Einsatz von PRMs untersuchen und davon ausgehend Anforderungen an geeignete Anwendungsschnittstellen formulieren. Die Anwendung ist dabei jene nicht näher spezifizierte Instanz, die Befehle an RMs ausschickt.

Mit dem Distributed Transaction Processing Model (DTP-Modell) [X/O91] wurde vom X/Open Ausschuß eine allgemeine Systemarchitektur zur verteilten Transaktionsverarbeitung entwickelt (Abb. 1). Es beschreibt, wie verschiedene Resource Manager zu einer Anwendung dazugebunden werden können:

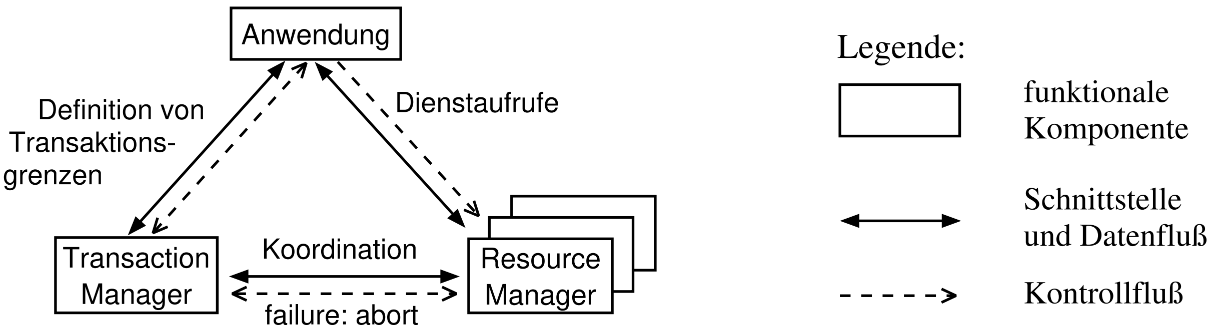


Abb. 1 Architektur nach X/Open

Hier werden autonome Resource Manager vorausgesetzt; der Transaction Manager (TM) übergibt lediglich zu Beginn einer Transaktion einen Transaktionsbezeichner an die beteiligten Resource Manager und koordiniert deren Abschluß. Die Anwendung übergibt beim Aufruf eines Resource Managers keinen Transaktionsbezeichner. Vielmehr muß der Resource Manager die Transaktion anhand des *threads*[1] erkennen, von dem aus er sowohl vom Transaction Manager als auch von der Anwendung aufgerufen wird. Wird z.B. der thread anhand seiner Prozeßkennung identifiziert, so muß diese auf den Transaktionsbezeichner abgebildet werden, um die durch den Prozeß geleistete Arbeit der richtigen Transaktion zuzuordnen. Dadurch läßt sich die natürliche Anwendungsschnittstelle eines Resource Managers ohne Modifikation nutzen.

Zentralisierte Recovery: Es gibt ein ähnliches Modell zur Beschreibung einer Systemarchitektur, das in [GR92] ausführlich beschrieben ist. Hier wird von einem zentralen Transaction Manager ausgegangen, und es wird auch ein zentraler Log für alle Resource Manager angenommen, mit Hilfe dessen der Transaction Manager Recovery für die Resource Manager durchführt. Durch die Zentralisierung der Recovery können fortgeschrittene Recoverytechniken realisiert werden.

[1] Ein thread ist hier ein Betriebssystemprozeß.

Unser Ansatz nimmt unabhängige Resource Manager mit eigenem Log und selbständiger Recovery an. Bei Aufrufen eines RM wird immer der dazugehörige Transaktionsbezeichner mit übergeben. Dies ist uns vor allem bei der Fehlerbehandlung wichtig. Für die Recovery werden die Grundtechniken des Modells des zentralen Logs übernommen, speziell die im folgenden verwendete Undo- und Redo-Logik.

2.4 Die Anwendung - Integration des PRM in ein Fertigungssystem

Die Frage, wie ein TPS mit PRMs in der Fertigung eingesetzt werden kann, wirft die Frage nach der Art der Anwendung, die die PRMs aufruft, auf.

Nach dem allgemeinen Modell zur verteilten Transaktionsverarbeitung ist die Anwendung die Instanz, die Dienstaufrufe an die RMs sendet. Unsere Betrachtungen beziehen sich auf die Fertigung, genauer auf die Möglichkeiten der automatischen Fehlerbehebung in Fertigungs- oder CAM (Computer Aided Manufacturing) - Systemen, und als Anwendung kommt hier nur ein Leitsystem in Betracht. Von den vielfältigen Funktionen innerhalb eines Leitsystems wie Auftragseinplanung, Werkzeugüberwachung oder Betriebsdatenüberwachung interessiert uns besonders die sogenannte Auftragsdurchsetzung. Sie ist dafür zuständig, die für die Bearbeitung von Werkstücken notwendigen Operationen zu initiieren und durchführen zu lassen und dabei einen von anderen Funktionen erstellten Zeitplan einzuhalten. Die Aufrufe für die Durchführung der Operationen sollen jetzt nicht mehr direkt an die jeweiligen Maschinen gesendet werden, sondern an die Physical Resource Manager gehen, die die jeweilige Maschine oder Zelle repräsentieren. Erst nach dem Ausführen der notwendigen Maßnahmen, um die Operation unter Transaktionsschutz zu stellen, werden die einzelnen PRM-Aufrufe an die zugrundeliegenden Maschinen weitergeleitet.

Hier stellt sich die Frage, wie und von wem die Transaktionsgrenzen gesetzt werden. Hierzu ist es nötig, sich die Abläufe im Leitsystem von der Auftragserteilung bis zur Auftragsdurchsetzung anzusehen (es soll hier aber nicht auf den prinzipiellen Aufbau von Leitsystemen eingegangen werden). Nach der Auftragserteilung wird für das Werkstück[1] der Arbeitsplan[2] abgearbeitet, und alle Operationen werden zur Bearbeitung an verschiedenen Maschinenklassen eingeplant. Die nötigen Transportvorgänge zwischen den Bearbeitungsmaschinen werden erst bei Bedarf bestimmt, da die Instanzen des geforderten Maschinentyps und damit deren Positionen nicht von vornherein feststehen.

Hier bietet es sich geradezu an, jeden Schritt des Arbeitsplans durch eine Transaktion als Einheit der Recovery zu modellieren; in diesem Fall können die Transaktionsgrenzen automatisch beim Umsetzen des Arbeitplans in Operationen auf Maschinen gesetzt werden. Aus der sequentiellen Struktur eines Arbeitsplans für einen Werkstücktyp ergibt sich eine sequentielle Folge der einzel-

[1] Der Einfachheit halber wird von Losgröße eins ausgegangen.

[2] Ein Arbeitsplan ist eine Sequenz von Operationen (ohne Transportvorgänge), die an einem Werkstück durchgeführt werden sollen, unter Angabe von Parametern wie Maschinentyp oder Bearbeitungsdauer.

nen Transaktionen, Transaktionskette genannt. Jedes zu bearbeitende Werkstück wird durch eine Instanz seines Transaktionskettentyps beschrieben; diese Instanz wird dann abhängig von der tatsächlichen Ausführungsfolge modifiziert.

2.5 Annahmen über Transaktionen

Transaktionen auf PRMs enthalten physische und nichtphysische Operationen. Physische Operationen sind Operationen, die das Werkstück in seiner Form verändern, beispielsweise Fräsen, Bohren oder Drehen, und Transportoperationen, die das Werkstück bewegen. Als nichtphysische Operationen bieten PRMs typischerweise Operationen der Programmverwaltung (z.B. Laden, Binden oder Starten) oder auch das Lesen und Schreiben von Statusvariablen an[1].

Physische Operationen haben keine "prepare"-Phase. Entweder schlagen sie fehl oder sie gelingen - ihr Ergebnis wird aber in jedem Fall sofort sichtbar. Nichtphysische Operationen in einer Transaktionen sind oft vom Ergebnis der physischen Operation semantisch abhängig, und daher muß man versuchen, das Ergebnis der Gesamttransaktion vom Ergebnis der physischen Operationen abhängig zu machen. Beim parallelen Ablauf zweier physischer Operationen in einer Transaktion jedoch gibt es, wenn die Operationen nicht entweder beide erfolgreich beendet werden oder beide scheitern, kein gemeinsames Ergebnis, nach dem sich der Ausgang der Gesamttrans-aktion richten kann. Wr können zwar für die fehlgeschlagene Operation Forward Recovery veranlassen, aber es gibt Fälle, in denen diese nicht gelingt. Um die Konsistenz der Daten zu erhalten, müßten dann die semantischen Abhängigkeiten der nichtphysischen von den physischen Operationen bekannt sein, was wir nicht voraussetzen können. Der Fall zweier paralleler physischer Operationen, und damit der mehrerer paralleller physischer Operationen, innerhalb einer Transaktion muß daher ausgeschlossen werden. Die parallele Ausführung mehrerer Operationen, z.B. Doppellochbohren, kann jedoch als eine einzige Operation aufgefaßt werden, solange sie von einem einzigen Dienstaufruf verursacht wird und somit auch nur ein Ergebnis liefert.

Es werden also nur Transaktionen betrachtet, die genau eine physische und beliebig viele nichtphysische Operationen umfassen. Prinzipiell ist es zwar möglich, Transaktionen mit mehreren, sequentiellen physischen Operationen zu behandeln; dies bedingt jedoch eine höhere Komplexität der Recoverymechanismen. Um die grundlegenden Mechanismen zu erklären, beschränken wir uns auf den einfacheren Fall einer physischen Operation je Transaktion.

Des weiteren nehmen wir an, daß die Aufrufe einer noch nicht abgeschlossenen Transaktion über jede Art von Fehler hinaus erhalten bleiben. Dies könnte durch eine stabile Warteschlange für Aufrufe realisiert werden, aus der diese Aufrufe nur entfernt werden, wenn sie erfolgreich bearbeitet worden sind (siehe dazu [BHM90]). Damit wird das Fortsetzen einer Transaktion nach erfolgreicher Fehlerbehandlung ermöglicht.

---

[1] Typische Dienste eines Fertigungsgerätes sind z.B. im Kommunikationsprotokoll Manufacturing Message Specification (MMS) beschrieben [MMS].

2.6 Einhalten der ACID - Eigenschaften

Insgesamt stellt sich die Frage, inwieweit die von uns auf RMs und PRMs definierten Transaktionen noch die Eigenschaften klassischer Transaktionen bewahren.

Nichtphysische Operationen auf PRMs sollen genauso geschützt werden wie nichtphysische Operationen auf regulären RMs[1]. (Diese Thematik allerdings war und ist Gegenstand zahlreicher Untersuchungen, und es soll in diesem Artikel nicht weiter darauf eingegangen werden).

Es wurde schon erklärt, daß Ununterbrechbarkeit nur noch für die PRMs, nicht mehr aber für das Werkstück garantiert werden kann. Konsistenz ist ebenfalls gewährleistet, zumindest nach unseren Anforderungen: im Falle eines Fehlers werden alle Daten zurückgesetzt und mit Abbruch der Transaktion verschwindet auch das Werkstück, logisch gesehen, aus dem System[2]. Wir können diese Eigenschaften garantieren, da wir nur (P)RMs betrachten, die reguläre Resource Manager und Fertigungsgeräte im weiteren Sinne darstellen. Für das Werkstück selbst steht aber im TPS kein Repräsentant zur Verfügung, außer der Instanz der Transaktionskette und den das Werkstück beschreibenden Variablen, für die jedoch wieder die klassischen Transaktionseigenschaften geboten werden. Isolation und Dauerhaftigkeit gelten unverändert.

3 Resource Manager und Transaktionen

Vor der eigentlichen Beschreibung der Recovery für einen Physical Resource Manager müssen noch zwei grundlegende Aspekte der Modellierung geklärt werden: Zum einen stellt sich die Frage, welche Objekte oder Dienste ein PRM denn nun darstellen soll - eine Maschine, eine Fertigungszelle[3] oder eine Klasse von Operationen, und es muß festgelegt werden, welcher Mechanismus bei Abbruch einer Transaktion an die Stelle des Rollbacks tritt. Zum anderen muß geklärt werden, wie ein PRM aufgebaut ist.

3.1 Abbildung von Werkzeugmaschinen auf Resource Manager

Es gilt also zu klären, wie Dienste auf einen Physical Resource Manager abgebildet werden können und um welche Dienste es sich dabei handelt. In einer Fertigungsumgebung konkretisiert sich dieses Problem zu der Frage, ob ein PRM Klassen von Diensten oder Klassen von Maschinen zum TPS hin vertritt. Davon hängt ab, ob sich Konzepte klassischer Resource Manager auch auf PRMs übertragen lassen.

---

[1] Siehe dazu jede relationale Datenbank oder transaktionsgeschützte Dateisystems wie in [SW91] oder [EMS89].

[2] Die Daten über das Werkstück können natürlich weiterverwendet werden, z.B. bei einer manuellen Weiterbearbeitung außerhalb von Transaktionen.

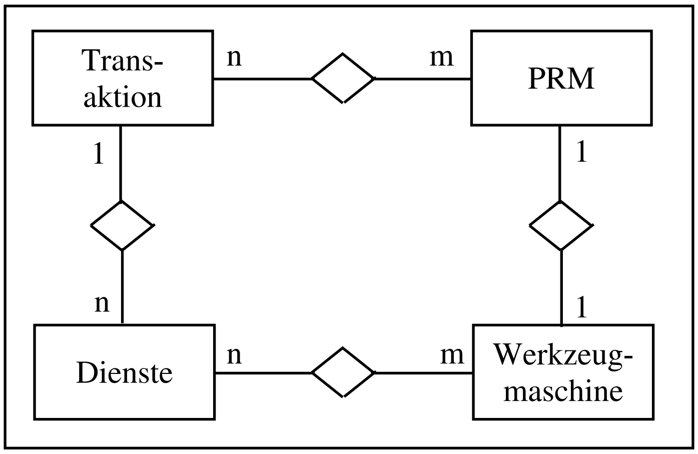
[3] Eine Fertigungszelle ist eine Gruppe gleichartiger Werkzeugmaschinen, die nach außen, d.h. zum Leitsystem wie eine einzige Maschine wirkt und auch als eine einzige Maschine angesprochen wird.

Repräsentation von Werkzeugmaschinen: Viele Werkzeugmaschinen bieten verschiedene Klassen von Operationen an - auf manchen Bearbeitungszentren kann sowohl gebohrt als auch gefräst werden, Drehmaschinen erlauben oft auch Fräsarbeiten. Durch das direkte Aufeinanderfolgen der Operationen lassen sich Be- und Entladezeiten ebenso einsparen wie die Zeit, die möglicherweise für eine neue Einspannung gebraucht wird. Andererseits können in einem Fertigungssystem verschiedene Maschinen auch den gleichen Typ von Operation, zum Beispiel Fräsen, anbieten. Wir bevorzugen die direkte Abbildung einer Werkzeugmaschine auf einen PRM aus zwei Gründen:

- Jede Werkzeugmaschine verhält sich anders, wenn sie eine Operation durchführt. Sie kann z.B. unterschiedliche Genauigkeitsklassen anbieten oder nach bestimmten Fehlern die Fortsetzung der unterbrochenen Operation (d.h. des Bearbeitungsprogramms) unterstützen. Es erscheint nicht angebracht, Operationen mit derart verschiedenem Verhalten von einer einzigen Instanz anbieten zu lassen.

- Bei der direkten Abbildung wird die Integration eines Resource Managers in ein Leitsystem vereinfacht. Anstelle einer realen Werkzeugmaschine wird der stellvertretende PRM beauftragt; Planungs und Scheduling müssen nicht von der Leitsystemebene in die der Resource Manager verlagert werden.

Konzept der Serverprozesse: In der klassischen Transaktionsverarbeitung existiert das Konzept, identische Serverprozesse einer Serverklasse und damit einem Resource Manager zuzuordnen und bei Bedarf neue Serverprozesse erzeugen zu lassen. Man könnte nun versucht sein, in der Fertigungsumgebung das Analogon in einem PRM, der eine Fertigungszelle darstellt, zu sehen. Prozesse jedoch sind billig, schnell und leicht zu replizieren - was auf Fertigungsmaschinen nicht zutrifft und auch nicht auf die von ihnen angebotenen Dienste. Die Beziehungen zwischen diesen Elementen werden in Abb.2 mit Hilfe eines Entity-Relationship-Diagramms dargestellt:



Jede Transaktion kann verschiedene PRMs aufrufen; dabei ist jedoch zu beachten, daß nur einer dieser PRMs eine physische Operation ausführen darf. Jeder PRM kann von verschiedenen Transaktionen aufgerufen werden.

Jeder Dienst wird zu einem bestimmten Zeitpunkt für genau eine Transaktion ausgeführt; eine Transaktion kann beliebig viele Dienste aufrufen, aber nur einer davon darf eine physische Operation sein.

Abb. 2 Beziehungen zwischen Transaktionen, Resource Managern, Maschinen und Diensten

Dennoch ist es möglich, eine Fertigungszelle von einem einzigen PRM darstellen zu lassen. Die Fehlerbehandlung innerhalb einer solchen Fertigungszelle, beispielsweise das Wiederholen eines durch Maschinenausfall unterbrochenen Bearbeitungsvorganges auf einer funktionsfähigen Maschine, wird nach außen hin nicht sichtbar und ist dann auch nicht Sache des PRM's. Die Vorgehensweise enspricht derjenigen auf Inter-PRM-Ebene[1], und es wird an dieser Stelle nicht weiter darauf eingegangen.

Prozeßmigration: Ähnliches trifft auch für die Prozeßmigration zu. In der regulären Transaktionsverarbeitung kann ein Prozeß auf einen anderen Knoten migrieren, indem dessen Daten und Kontext übertragen werden. Dort kann er fortgesetzt werden, als ob er sich auf dem ursprünglichen Knoten befände und nicht unterbrochen worden wäre. Wie jedoch ist der Kontext der Bearbeitung eines Werkstücks auf einer Werkzeugmaschine zu sehen? Wir beschränken uns vereinfachend auf die Achspositionen der Maschine, die mit Sicherheit zum Kontext zählen. Nun könnte eine identische Werkzeugmaschine, falls verfügbar, die gleichen Achspositionen einnehmen, was durch ein Wiederholen das Programms bis zum Unterbrechungszeitpunkt erfolgen kann. Zudem muß das Werkstück geladen werden, unserem Gegenstück zu den Daten, und genau hier treten die Probleme auf. Das Werkstück muß geladen werden, bevor die Achsen die "Kontext"-positionen einnehmen, da diese sich beim Ladevorgang in sicheren Positionen befinden müssen, d.h. in Positionen, in denen keine Möglichkeit einer Kollision mit dem Werkstück besteht. Üblicherweise müssen dazu die Achsen sogenannte Referenzpunkte anfahren. Erst nach dem Laden des Werkstücks kann der Kontext geladen, d.h. das Programm neu begonnen werden. Selbst wenn wir annehmen, daß das Werkstück die Programmunterbrechung unbeschadet überstanden hat, können wir doch nicht annehmen, daß die Operation wiederholbar ist, d.h. daß das Werkstück eine Wiederholung des Programmes bis zur Einnahme der Achspositionen zum Unterbrechungszeitpunkt im Sinne der Anwendung unbeschadet übersteht.

Eine Migration im Sinne eines physischen Ortswechsels des Werkstücks und Neustart des Programmes kann, wenn überhaupt, nur vom Physical Recovery Manager veranlaßt werden, der explizit den Transport eines Werkstückes und die folgende Zwischen- und Weiterbearbeitung auf einer anderen Maschine veranlaßt. Zur Unterstützung einer solchen Migration sollte die Maschinenadressierung nicht direkt, sondern relativ über einen Zeiger auf die aktuelle Maschine erfolgen. Beispielsweise führt ein maschinenbezogener Befehl der Art "Schreibe Ergebnis der vorigen Operation auf PRM A" nach einer Migration von Maschine A auf Maschine B zu einer Inkonsistenz, der Befehl "Schreibe Ergebnis auf aktuelle Maschine" liefert jedoch das gewünschte Resultat.

3.2 Reset, Rollback und Abort

In herkömmlichen TPS wird ein Abort einer Transaktion mit Hilfe eines Rollback, also des Zurücksetzens der Effekte der bisher durchgeführten Operationen, auf den beteiligten Resource Managern durchgeführt.

---

[1] Hier zeigt sich, wie sich analog zum hierarchischen Aufbau eines Fertigungssystems eine ebenfalls hierarchische Fehlerbehandlung entwickelt.

Ein Physical Resource Manager kann kein Rollback durchführen, wenn er schon eine physische Operation ausgeführt hatte. Ihm bleibt nur ein *Reset*: Ein *Reset* ist das Gegenstück eines PRMs zum Rollback eines regulären RMs. Es setzt alle Operationen einer Transaktion zurück, wobei die Rücksetzoperationen für nichtphysische Operationen aus dem Log bestimmt werden. Physische Operationen werden insofern zurückgesetzt, als die Maschine in ihre Referenzposition gefahren und das Werkstück herausgeladen wird. Aus Sicht der Werkzeugmaschine ist damit der Zustand, der vor Beginn der Transaktion herrschte, wiederhergestellt worden und somit wird auch eine eingeschränkte Art von Ununterbrechbarkeit erreicht. Die Sicht, genauer: der Zustand, des Werkstücks kann beim Reset nicht berücksichtigt werden.

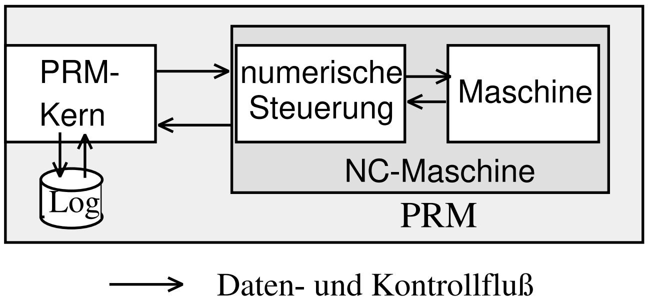
3.3 Laden und Entladen eines Werkstückes

Allgemein ist eine generelle Strategie notwendig, die das Laden und Entladen von Werkstücken im Rahmen der Fehlerbehandlung abdeckt. Ein Reset eines PRM wird dann, und nur dann, durchgeführt, wenn die Transaktion zumindest auf diesem PRM nicht mehr aus- oder fortgeführt werden kann. Wir fordern, daß beim Auftreten eines Resets die Maschine in einem Zustand hinterlassen wird, in dem prinzipiell noch andere Transaktionen darauf ablaufen können. Dies bedeutet auch, daß das aktuelle Werkstück entladen werden muß. Dabei kann vorausgesetzt werden, daß das Entladen eines Werkstückes immer mit dem Einnehmen der Referenzpositionen durch die Werkzeugmaschine (Referenzfahrt) gekoppelt ist. Diese Aktion kann nur vom Physical Recovery Manager durch den Aufruf des Reset initiiert werden[1]. Damit wird auch das zu Beginn erwähnte failfast-Verhalten erreicht - nach erfolgloser Recovery befindet sich der PRM in einem wohldefinierten Ausgangszustand.

3.4 Komponenten eines Physical Resource Managers

Nach den bisherigen Ausführungen hat ein PRM die Aufgabe, eine Werkzeugmaschine zum transaktionsverarbeitenden System hin zu vertreten. Nun muß der realen Aufbau eines solchen PRMs beschrieben werden. Prinzipiell besteht ein PRM aus zwei Teilen, der eigentlichen NC-Maschine und dem PRM-Kern mit dem Log, der die NC-Maschine nach außen mit Transaktionssemantik anbietet. Die Maschine selbst wird über eine numerische Steuerung (NC) mit Daten und Programmen versorgt (Abb. 3). Jede dieser drei Komponenten kann sich auf einem separatem Knoten befinden. Es gibt sogar Implementierungen, bei denen die Steuerung auf mehrere Knoten verteilt ist. Vom funktionalen Gesichtspunkt jedoch kann und muß die Steuerung als eine Komponente betrachtet werden. Es ist deren Aufgabe, nach außen ein korrektes Verhalten zu zeigen.

[1] Der Physical Recovery Manager hat eine Zwischenstellung zwischen Anwendung und Transaktionsmanager.



Aufgabenverteilung:
Maschine: physische Operationen
Steuerung: Maschinensteuerung und nichtphysische Operationen
PRM-Kern: Transaktionsschutz für alle Operationen

Abb. 3 Komponenten eines Physical Resource Managers

PRM-Kern: Der PRM-Kern ist die Schnittstelle des PRM zum TPS und dafür verantwortlich, eintreffende Dienstaufrufe unter Transaktionsschutz durchzuführen. Im Normalbetrieb beschränken sich seine Aktivitäten darauf, diese Dienstaufrufe in einen Log zu schreiben (dazu mehr in den folgenden Abschnitten) und die gewünschten Operationen durchzuführen bzw. von der NC-Maschine durchführen zu lassen. Der PRM-Kern nutzt die NC-Maschine als Server, um deren Dienste der Außenwelt anzubieten.
Der PRM-Kern und die NC-Maschine müssen möglicherweise über ein Netzwerk miteinander kommunizieren - dies wird bei der Erweiterung existierender NC-Maschinen zu PRMs oft unumgänglich. Im Vergleich zu regulären RMs vergrößert dies die Verantwortung des PRM-Kerns, doch nur auf diese Weise lassen sich existierende Maschinen in ein TPS integrieren[1].

Nichtphysische Operationen auf einer Werkzeugmaschine: Die Verwaltung von Programmen und Daten einer Werkzeugmaschine zählt zu den Aufgaben einer NC. Wir nehmen daher an, daß der Datenserver für die nichtphysischen Operationen sich auf dem gleichen Knoten wie die NC befindet. Bei sehr intensivem Datenzugriff mag eine separate Komponente, die auch dem PRM-Kern zugeordnet sein kann, notwendig werden. Da jedoch physische Operationen die typischen und auch langwierigsten Operationen in der Fertigung sind und Operationen auf Daten normalerweise nicht der Engpaß sind, gehen wir auf diesen Fall nicht weiter ein.

Der Log: Der Log dient dem sicheren und dauerhaften Speichern von Einträgen zur Unterstützung der Recovery. Der PRM benutzt ihn, um für jeden empfangenen Aufruf einen Logeintrag, bestehend aus dem Dienstaufruf mit seinen Parametern und dem Transaktionsbezeichner sowie den dazugehörigen Redo- und Undo-Einträgen, sicher zu speichern. Dieser Eintrag wird vom PRM vor der eigentlichen Ausführung des Dienstes gemacht. Dann ist es für die Recovery unwesentlich, ob der Fehler vor oder nach der Ausführung des Dienstes geschehen ist: Da wir Idempotenz für nichtphysische Operationen annehmen, hinterlassen sowohl das Undo einer noch nicht ausgeführten Operation als auch das Redo einer schon ausgeführten Operation keine unerwünschten Effekte. Physische Operationen werden gesondert vom Physical Recovery Manager behandelt, da sie im allgemeinen nicht einfach zurückgesetzt werden können.

[1] Bei der Entwicklung neuer Maschinen könnten PRM-Kern und NC-Steuerung integriert werden, und beim Entwurf der NC - Steuerung sollte auch auf die Belange des PRM-Kerns eingegangen werden.

Bestimmung der Log-Einträge: Um die Undo- und Redo-Einträge zu einer Operation bestimmen zu können, muß ein PRM zu jeder Operation (außer einer physischen) die entsprechenden Undo- und Redo-Operationen kennen. Für Leseoperationen beispielsweise ist die Undo-Operation die leere Operation, für Schreiboperationen ist es die Wiederherstellung des ursprünglichen Werts. Geladene Segmente oder Dateien werden durch Löschen zurückgesetzt. Nimmt man als Anwendungsschnittstelle MMS[1] an, so kann man von ungefähr achtzig Diensten ausgehen, von denen eine reale Werkzeugmaschine in der Regel nur eine Untermenge realisiert. Die Zuordnung von Aufruf zu Undo-Aufruf kann daher mit Hilfe einer einfachen Tabelle im PRM erfolgen. Der Redo-Aufruf ist aufgrund der Idempotenz im allgemeinen der Aufruf selbst.

Implementierung des Log: Wir nehmen für PRMs einen Log an, der direkt auf stabilen Speicher zugreift. Im Vergleich zu Höchstleistungsdatenbanksystemen tritt bei der Kommunikation mit Fertigungsgeräten nur ein geringer Datenverkehr auf. Da zudem jeder PRM einen eigenen Log besitzt, sollte dieser nicht die Ursache für Durchsatzminderungen sein. In hochentwickelten Transaktionsverarbeitungs-systemen hingegen kann ein Log, der direkt auf stabilen Speicher zugreift, leicht zum Engpaß werden. Dort ist man zum Teil dazu übergegangen, zur Durchsatzerhöhung Einträge im flüchtigen Hauptspeicher zwischenzuspeichern. Dies führt zu einer erhöhten Komplexität der Recoveryalgorithmen.

4 Recovery

In diesem Kapitel wird beschrieben, wie ein PRM Fehler behandelt. Unter Fehlern verstehen wir hier nicht konzeptuelle Fehler oder Programmfehler, sondern ausschließlich Fehler, die durch Ausfälle einzelner Komponenten bedingt sind. Weiter wird angenommen, daß Fehler immer entdeckt oder mitgeteilt werden. Als erstes werden die Maßnahmen bei isolierten Fehlern der beteiligten Komponenten beschrieben. Daraus lassen sich die Recoveryalgorithmen bei Fehlern mehrerer Komponenten ableiten. In diesem Zusammenhang wird auch auf die Reaktion auf einen Rücksetz-Befehl des Transaktionsmanagers eingegangen.

Das Vorgehen besteht darin, bei Auftreten eines Fehlers die Transaktion wieder in den Zustand zum Fehlerzeitpunkt zu bringen und danach weiterzuarbeiten. Mit dem erneuten Versuch der Durchführung soll ein Abbruch der Transaktion möglichst verhindert werden. Prinzipiell kann dies durch Abarbeiten der Undo- und Redo- Einträge aus dem Log erreicht werden: müssen Operationen rückgängig gemacht werden, wird der Log sequentiell abgearbeitet, beginnend mit der Undo-Operation des letzten Aufrufes und endend mit der der ersten Aufrufes. Beim Wiederholen schon durchgeführter Operationen hingegen wird mit der als erstes ausgeführten Operation begonnen. Eine effektive Implementierung eines Logs zur Unterstützung dieses Vorgehens ist in [GR92] ausführlich beschrieben. Die Sonderstellung der physischen Operationen wird im folgenden noch besprochen.

[1] MMS (Manufacturing Message Specification) ist ein Anwendungsprotokoll des auf dem ISO/OSI - Referenzmodell basierenden Profil MAP (Manufacturing Automation Protocol) und ist für die offene Kommunikation in Fertigungsumgebungen entwickelt worden.

Motiviert wird diese Art der Forward Recovery durch den Unterschied zu regulären Resource Managern. Bei ihnen führt ein Fehler unter Einsatz von Backward Recovery zum Rücksetzen der globalen Transaktion. Dort ist im Normalfall ein Wiederholen der Transaktion möglich, in der Fertigung jedoch nicht. Ein PRM versucht zunächst auf lokaler Ebene, d.h. auf dem betroffenen PRM, den Fehler zu beheben und die Transaktion fortzuführen. Schlägt die lokale Recovery fehl, wird versucht, die Transaktion auch unter Einbezug anderer PRMs zu Ende zu bringen. Nur wenn keine Möglichkeit zur Weiterführung mehr besteht, darf zurückgesetzt werden.

Wir wollen vergleichend das Recoveryprotokoll eines PRMs dem für einen regulären RM gegenüberstellen. Abb. 4 zeigt, wie prinzipiell mit einer Transaktion verfahren wird, in deren Verlauf ein Fehler auftritt.

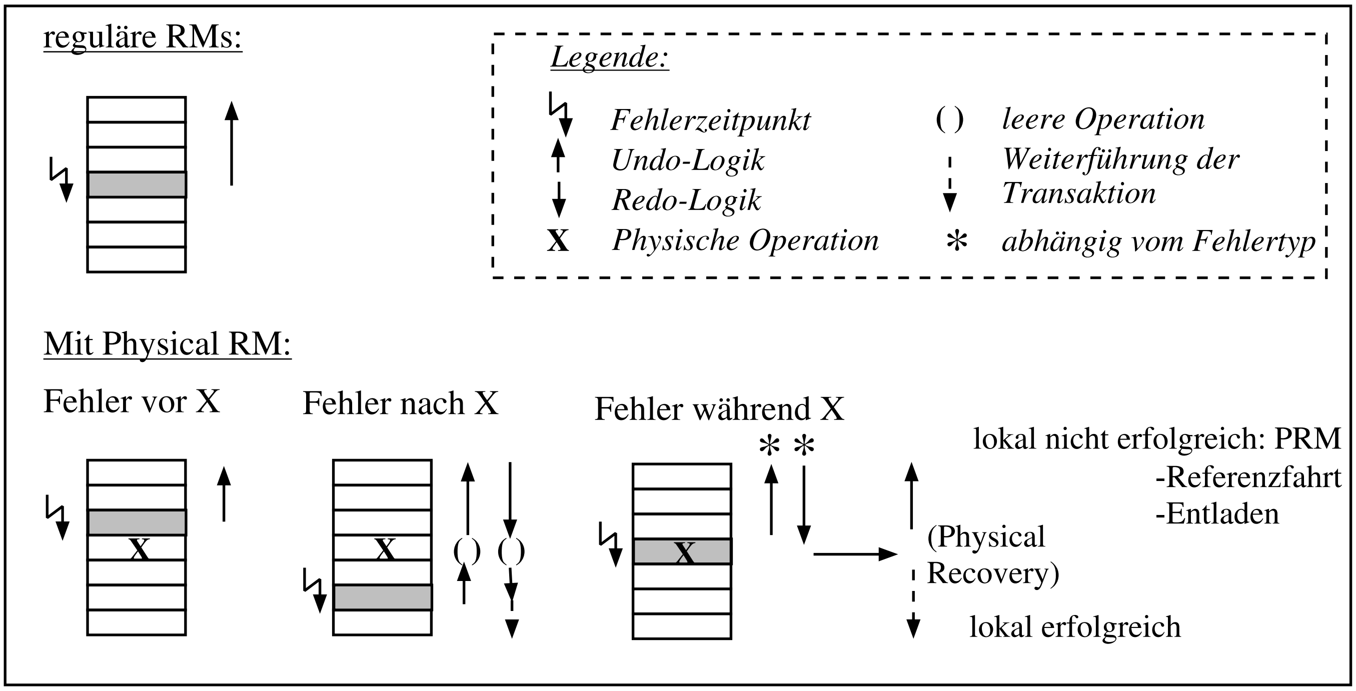


Abb. 4 Recovery für einen PRM

Die folgenden Abschnitten gehen auf die verschiedenen Fehler- und Rücksetzursachen und ihre Auswirkungen auf diese Vorgehensweise ein.

4.1 Rücksetzen

Ein PRM kann von seinem Transaction Manager aus Gründen, die außerhalb seiner Kontrolle liegen, eine Aufforderung zum Rücksetzen erhalten. Hier bleibt nichts anderes übrig, als ein "Reset" durchzuführen. Die Maschine wird also in die Grundposition gebracht, das Werkstück gegebenenfalls entladen und für die nichtphysischen Befehle die Undo-Logik angewendet.

4.2 Isolierter Fehler der Maschine

Die Recoverymaßnahmen für physische Operationen hängen von zahlreichen Faktoren ab, die oft erst zum Fehlerzeitpunkt bestimmt werden können. Da eine Einplanung aller Fehlerfälle, Faktoren und Recoverymöglichkeiten in den Arbeitsplan diesen soweit aufblähen würde, daß eine effektive Bearbeitung nicht mehr gesichert scheint, wird eine eigene Komponente, der Physical Recovery Manager, zur Verfügung gestellt. Er bestimmt bei Bedarf, und nur dann, die angemessenen Recoverymaßnahmen. Hier fließen Faktoren wie Art der Operation, Art des Fehlers, Zustand des Werkstückes, Zustand der Maschine oder auch allgemeine Strategien ein. Wir unterscheiden zwei Arten von Operationen: unterbrechbare, die weiter in wiederholbar, weiterführbar oder kompensierbar eingeteilt werden, und nicht unterbrechbare. Der Physical Recovery Manager bestimmt as all diesen Faktoren die jeweiligen Maßnahmen und fügt sie in den Arbeitsplan ein, der von der Anwendung, die nichts von dem Fehler bemerkt, abgearbeitet wird (siehe auch Abb. 5). Eine ausführlichere Beschreibung der Arbeitsweise des Physical Recovery Managers findet sich in [Sc92].

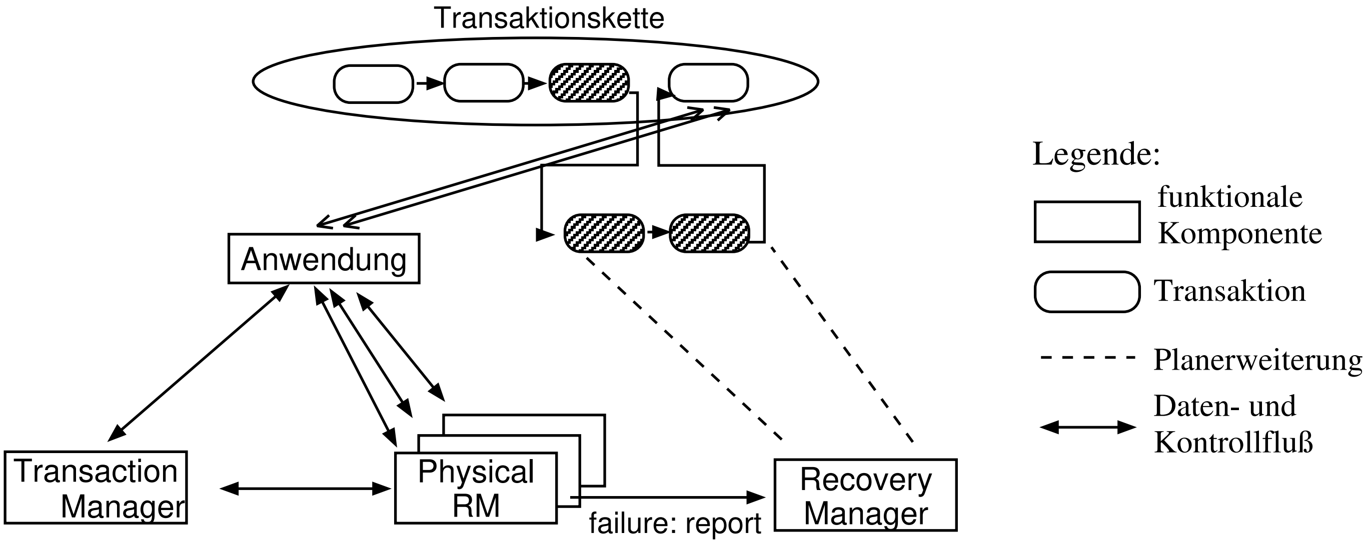


Abb. 5 Fehlerbehandlung bei physischen Operationen

Wichtig hierbei ist, daß die Recovery innerhalb der gleichen Transaktion stattfindet. Dies wird bewerkstelligt, indem der Transaktionsbezeichner der aktuellen Transaktion allen zur Recovery gehörenden Aufrufen mitgegeben wird. Bei lokaler Recovery, d.h. bei einer Fehlerbehebung ohne Einbezug anderer PRMs, werden die aktuellen Sperren der Transaktion gehalten und die Transaktion wird auf diesem PRM nach erfolgreicher Recovery weitergeführt. Im Prinzip geschieht hier nichts anderes, als daß die Transaktion durch Einfügen von Reparaturschritten erweitert wird. Sind Schritte auf anderern PRMs notwendig, wird auf dem aktuellen PRM ein Reset durchgeführt, damit dort zumindest noch andere Transaktionen ausgeführt werden können. Nun kann es durchaus geschehen, daß mehrere physische Reparaturschritte im Rahmen einer Fehlerbehandlung durchgeführt werden. Im Falle des endgültigen Scheiterns müssen dann alle beteiligten PRMs einen Reset durchführen. Falls ein PRM mehrere physische Operationen durchgeführt hatte, wird nur eine Referenzfahrt der Maschine durchgeführt.

4.3 Isolierter Fehler der NC-Steuerung

Aus Sicherheitsgründen führt dieser Fehler unmittelbar zu einem Nothalt der angesteuerten Maschine. Da in der Steuerung aufgrund des Fehlers mit großer Wahrscheinlichkeit Information über den realen Zustand der Maschine verlorengegangen sein kann, müssen das Bild, das die Steuerung von der Maschine besitzt, und der reale Zustand der Maschine wieder konsistent gemacht werden. Andernfalls entstünde ein beträchtliches Gefahrenpotential, das unter allen Umständen verhindert werden muß. Die einzige Möglichkeit, wieder Übereinstimmung zu erreichen, ist, sowohl die Maschine als auch die Steuerung zurückzusetzen, z.B. durch eine Referenzfahrt und einen Warmstart.

Da wir vorausgesetzt hatten, daß die Verwaltung von nichtphysischen Operationen einer Transaktion der Steuerung zugeordnet sind, entspricht der Warmstart der Steuerung dem Rücksetzen des nichtphysischen Teils der Transaktion. Zusammen mit der Referenzfahrt der Werkzeugmaschine erhalten wir also einen Zustand ähnlich dem nach einem Reset, mit dem Unterschied, daß das Werkstück nicht ausgeladen wurde.

Als nächstes wird die Transaktion wiederholt. Dazu holt der PRM-Kern die bisher ausgeführten Aufrufe, genauer gesagt: die Redo-Einträge, aus dem Log und führt diese aus, bis die physische Operation erreicht wird oder bis keine Redo-Einträge mehr im Log vorhanden sind. Falls jetzt keine Redo-Operationen mehr im Log vorhanden sind, wird mit der Transaktion an der unterbrochenen Stelle fortgefahren. Falls sich noch Redo-Einträge im Log befinden, gibt es zwei Möglichkeiten:

1.) Im einfachen Fall erfolgte der Nothalt, nachdem die physische Operation schon beendet war (in diesem Fall hatte er keine unmittelbaren Auswirkungen auf das Werkstück). Der Log-Eintrag für die physische Operation wird dann einfach übersprungen und die Redo-Einträge werden weiter abgearbeitet. Vor der Abarbeitung des letzten Log-Eintrages besteht eine gewisse Unsicherheit, ob diese Operation schon begonnen oder beendet wurde. Hier kann der Zustand der Transaktionswarteschlange weiterhelfen. Da wir idempotente nichtphysische Operationen annehmen, richtet es aber ohnehin keinen Schaden an, die Operation möglicherweise zu wiederholen.

2.) Im komplexeren Fall war die unterbrochene Operation eine physische. Hier muß der Physical Recovery Manager zu Rate gezogen werden, wie weiter verfahren werden soll. Nach erfolgreicher Recovery wird dann mit dem ursprünglichen Arbeitsplan fortgefahren[1].

Dieser Ansatz berücksichtigt noch nicht, daß eine Steuerung endgültig ausfallen kann, z.B. durch elektromechanische Fehler oder durch Fehler im Programmspeicher. Der Physical Recovery Manager kann damit zwar umgehen, für die interne Fehlerbehandlung des PRM ist dann aber, anstelle eines Wiederholens der Transaktion auf dem betroffenen PRM, folgendes Vorgehen nötig:

---

[1] Es gibt Maschinen, die parallel zur Bearbeitung von Werkstücken noch Dienste der Programm- oder Variablenverwaltung ausführen können. Da wir innerhalb einer Transaktion sequentielle, synchrone Aufrufe voraussetzen, müssen beide Operationen verschiedenen Transaktionen angehören, und in diesem Fall muß für beide beteiligten Transaktionen Recovery durchgeführt werden.

Fällt die Steuerung aus, bevor die physische Operation initiiert wurde, wird die gesamte Transaktion wiederholt, wobei der PRM durch eine funktionsfähigen, gleichartigen PRM ersetzt wird. Ist ein solcher nicht verfügbar, muß für die Gesamttransaktion ein Abort durchgeführt werden.

Bei einem Fehler nach Abschluß der physischen Operation ist ein Wiederholen der Transaktion auf einem anderen PRM nicht mehr möglich: Da wir keine Informationen über den Datenfluß innerhalb einer Transaktion besitzen, müssen wir die Möglichkeit berücksichtigen, daß orts- oder zeitabhängige Daten der physischen Operation gespeichert werden sollen. Diese dann auf einem anderen PRM abzubilden, kann die semantische Konsistenz verletzen. Auch hier ist ein Rücksetzen unumgänglich.

4.4 Isolierter Fehler des PRM-Kerns

Aufgabe des PRM-Kerns ist es, für eintreffende Aufrufe die für eine Fehlerbehandlung notwendige Information sicher zu speichern, die Ausführung der gewünschten Dienste zu initiieren und gegebenenfalls Recovery durchzuführen. Die Initiierung von Diensten bedeutet, den Aufruf in geeigneter Form an die NC-Maschine weiterzuleiten. Fällt der PRM-Kern aus, kann die NC-Maschine mit der Bearbeitung des momentanen Dienstes fortfahren, da sie unabhängig vom PRM-Kern arbeitet. Auch hier gibt es wieder zwei Varianten:

1.) Nach dem Restart könnte der PRM-Kern mit dem nächsten Request der Transaktion fortfahren - dies wird neben dem Vorhandensein einer stabilen Warteschlange auch durch einen Network Recovery Manager [Sc92] unterstützt, der die sichere Übertragung von Nachrichten, und daher auch der Antwort der Maschine, garantiert. Diese Methode kann angewendet werden, wenn zur Fortsetzung der Transaktion keine Information aus dem flüchtigen Speicher des PRM-Kerns benötigt wird. Sie ist insofern effektiv, als daß keine Operation zweimal ausgeführt wird. Der Nachteil ist aber, daß sie nicht in jeder Umgebung realisiert werden kann. Wir ziehen daher einen anderen Ansatz vor:

2.) Die Transaktion wird, wieder mit Ausnahme der physischen Operation, wiederholt. Die physische Operation war nicht vom Fehler des Kerns betroffen und darf nicht wiederholt werden, da Idempotenz nicht vorausgesetzt werden kann. Auch hier müssen die undo-Operationen nicht ausgeführt werden.

4.5 Kombinierte Fehler

Fehler der einzelnen Komponenten können unabhängig voneinander auftreten, aber sie müssen es nicht. Besonders, wenn sich die Komponenten eines PRMs auf dem gleichen Knoten befinden oder bei einem Stromausfall, können sie in jeder beliebigen Kombination auftreten.

- Falls alle Komponenten ausfallen, muß die Transaktion erst zurückgesetzt und dann wiederholt werden. Physische Operationen werden, wie oben beschrieben, gesondert behandelt.

- Fällt die gesamte NC-Maschine aus, d.h. sind sowohl die Steuerung als auch die eigentliche Maschine betroffen, der PRM-Kern arbeitet jedoch weiter, so kann nicht mehr von einem konsistenten Maschinenabbild in der Steuerung ausgegangen werden. In diesem Fall muß die gleiche Undo-Redo-Logik wie bei einem isolierten Fehler der Steuerung angewendet werden.

- Der Fall, daß Kern und Steuerung ausgefallen sind, die Maschine an sich jedoch noch operabel ist, tritt nicht auf. Bei einem Fehler der Steuerung wird automatisch der Nothalt ausgelöst und die Maschine muß zurückgesetzt werden; dies entspricht einem totalen Stromausfall.

- Im letzten verbleibenden Fall - der Kern und die Maschine sind ausgefallen, die Steuerung jedoch noch operabel - müssen analog zu einem totalen Stromausfall alle beteiligten Komponenten zurückgesetzt werden, um wieder Konsistenz zu erreichen.

5 Zusammenfassung und Ausblick

Ein PRM verwendet Transaktionstechnologie sowohl für physische als auch für nichtphysische Operationen. Für nichtphysische Operationen kann weiterhin Backward Recovery eingesetzt werden, Fehler bei physischen Operationen werden mit Hilfe von Forward Recovery behoben. Zum Transaktionsmanager und zur Anwendung hin verhält sich ein Physical Resource Manager wie ein regulärer Resource Manager und dies erlaubt ein einfaches Einbinden eines PRMs in ein bestehendes TPS.

Durch den Neubeginn oder die Erweiterung von Transaktionen im Fehlerfall wird versucht, den Abbruch von Transaktionen, und somit der gesamten Bearbeitung, zu vermeiden. Nach einem Ausfall von Einzelmaschinen oder des Gesamtsystems werden wohldefinierte Zustände erreicht, was die Wiederanlaufzeiten erheblich verkürzt. Die im Physical Recovery Manager komprimierte Fehlerbehandlung für Bearbeitungsoperationen erlaubt ein übersichtliches und effizientes Programmieren von Arbeitsplänen und steigert die Systemzuverlässigkeit.

Zur Zeit wird eine Simulationsumgebung für fertigungstechnische Abläufe erstellt, die zeigen wird, daß automatische Fehlerbehandlung durch Einsatz des Transaktionskonzepts auch für real actions möglich ist. Eine Erprobung mit echten Geräten muß zeigen, ob und inwieweit durch Anpassung an existierende, nichtideale Geräte Einschränkungen der Funktionalität eines PRMs auftreten und in welchem Umfang diese einer sinnvollen Integration eines PRM in ein TPS entgegenstehen. Mit dem hier vorgestellten Konzept ist jedoch der erste Schritt in Richtung einer Integration fertigungstechnischer Abläufe in transaktionsverarbeitende Systeme getan.

6 Literatur

[BHM90] Bernstein, Phil & Hsu, Meichun & Mann, Bruce: Implementing Recoverable Requests Using Queues. In: Proc. ACM SIGMOD, 1990.

[DIN] DIN 55350, Teil 11. Begriffe der Qualitätssicherung und Statistik. Grundbegriffe der Qualitätssicherung. Beuth Verlag, 1987.

[Do87] Donald, Bruce R.: Error Detection and Recovery in Robotics. Lecture Notes in Computer Science, 336, Springer, 1987.

[EMS91] Eppinger, J.L. & Mummert, L.B. & Spector, A.Z.: Camelot and Avalon: A Distributed Transaction Facility. San Mateo, CA, Morgan Kaufmann.

[Gi90] Gini, Maria: Automatic Error Detection and Recovery. In: Rembold, Ulrich (Ed.): Robot Technology and Applications, Marcel Dekker, 1990, pp. 445-483.

[GS87] Garcia-Molina, Hector & Salem, Kenneth: Sagas.
In: Proc. ACM SIGMOD, San Francisco, 1987,

[Gr81] Gray, Jim: The Transaction Concept: Virtues and Limitations.
In: Proc. 7th Int. Conf. on VLDB, Cannes, 1981.

[GR92] Gray, Jim & Reuter, Andreas: Transaction Processing - Concepts and Techniques.
Morgan Kauffman Publishers, 1992.

[JWZ88] Jablonski, S. & Wedekind, H. & Zörntlein, G.: Fehlerbehandlung in Flexiblen Fertigungssystemen (FFS). In: Informatik Forsch. Entw. (1988) 3:53-63.

[MAP] Manufacturing Automation Protocol Specification Version 3.0, 1988.

[MH88] Meijer, G.R. & Hertzberger, L.O.: Exception handling for robot manufacturing process control. In: Proc. 4th CIM Europe Conf., 18-20 May 1988, pp. 213-222.

[MMS] ISO 9506: Manufacturing Message Specification. Part I: Service Definition.

[Pa88] Pausch, Randy: Adding Input and Output to the Transactional Model.
CMU, Ph.D.Thesis, 1988.

[Sc92] Schmidt, Ursula: A Framework for Automated Error Recovery in FMS.
In: Proceedings of the 2nd Int. Conf. on Automation, Robotics and Computer Vision, Sept. 16 - 18, 1992; Singapore.

[SW91] Schmuck, Frank & Wyllie, Jim: Experience with Transactions in Quicksilver.
In: Proc. 13th ACM Symp. on Operation System Principles, Oct.13-16,1991, Pacific Grove, Ca.

[X/O91] X/Open: Distributed Transaction Processing: the XA-Specification.
Reading: X/Open Company, 1991.

[WR91] Wächter, Helmut & Reuter, Andreas: "The ConTract Model".
In: A.K. Elmagarmid (Ed.): Transaction Models for Advanced Database Applications. Morgan Kauffmann Publishers, 1991.

[WZ87] Wedekind, H. & Zörntlein, G.: Eine konzeptuelle Basis für den Einsatz von Datenbanken in Flexiblen Fertigungssystemen.
In: Informatik Forsch. Entw. (1987) 2:83-96.