Transaction Management in Web und E-Commerce

Problemseminar

Leitung: Prof. Dr. Erhard Rahm

Betreuer: Dipl.-Math. Robert Müller

ausgearbeitet durch:

Christian Blümel,
Matr.-Nr. 7754001

Inhaltsverzeichnis

Inhaltsverzeichnis 2
1. Einleitung 3
1.1 Definition E-Commerce 3
1.2 Entwicklung E-Commerce im Internet 3
1.3 Transaktionen im E-Commerce 5
2. ACID-Transaktionen 7
2.1 ACID-Transaktionen in Datenbanken 7
2.2 Nachteile von ACID-Transaktionen 8
3. Erweiterungen des klassischen Transaktionskonzeptes 8
3.1 Geschachtelte Transaktionen 9
3.2 ConTracts 9
3.3 Atomaritäts- und Isolationssphären 13
3.3.1 Das WISE-Projekt 13
3.3.2 Atomaritäts- und Isolationssphären in WISE 15
4. Atomarität in E-Commerce-Protokollen 16
4.1 Atomaritätsstufen 16
4.2 Nicht-atomare Protokolle 18
4.2.1 Digicash 18
4.2.2 SSL 19
4.2.3 SET 19
4.3 Beispiel für ein atomares Protokoll: NetBill 20
4.3.1 Vorbemerkung 20
4.3.2 Das NetBill-Transaktionsmodell 20
4.3.3 Das NetBill-Transaktionsprotokoll 21
5. Zusammenfassung 23
Literaturverzeichnis 24

1. Einleitung

1.1 Definition E-Commerce

Der Begriff E-Commerce wird unterschiedlich definiert. Einige Beispiele:

- E-Commerce bedeutet: Kaufen und Verkaufen über ein Netzwerk [...][1]
- E-Commerce ermöglicht die umfassende, digitale Abwicklung der Geschäftsprozesse zwischen Unternehmen und deren Kunden über öffentliche und private Netze [DA99].
- E-Commerce bezeichnet die Ausnutzung der technischen Mittel elektronischer Datennetze, um die Wirtschafts- und Absatzprozesse einer Unternehmung zu fördern und neue Absatzwege zu erschließen. Die Einzelziele erstrecken sich dabei von der Unternehmenskommunikation über die Wertschöpfungsprozesse bis zum Verkaufsvorgang über alle Marktphasen [FP97].

Der Begriff wird demnach unterschiedlich weit gefaßt: Vom reinen Kauf- und Verkaufsprozeß, über alle Geschäftsprozesse zwischen Unternehmen und Kunden bis hin zur Einbeziehung unternehmensinterner Prozesse. Im Rahmen des Problemseminars soll die Bedeutung und Anwendung von Transaktionen für E-Commerce-Anwendungen, die die Abwicklung von Prozessen zwischen Unternehmen und deren Kunden ermöglichen, im Mittelpunkt stehen.

1.2 Entwicklung E-Commerce im Internet

Ein Grund für die derzeitige Euphorie um E-Commerce ist das Wachstum der Zahl der Internetnutzer. So schätzen verschiedene Marktforschungsunternehmen wie zum Beispiel die *International Data Corporation*[2] die Anzahl der Internetnutzer für das Jahr 2000 in den USA auf 60-80 Millionen, in Europa auf 45-60 Millionen. Verglichen mit den Zahlen von 1995 (USA: rund 25 Millionen, Europa: rund 10 Millionen) stellt dies ein Wachstum von durchschnittlich 22% (USA) und sogar 40% (Europa) dar (siehe Abb. 1).

---

[1] Amjad, Umar: E-Commerce Bedrock, DBMS Magazin, http://www.dbmsmag.com, 1997

[2] International Data Corporation, http://www.idc.com

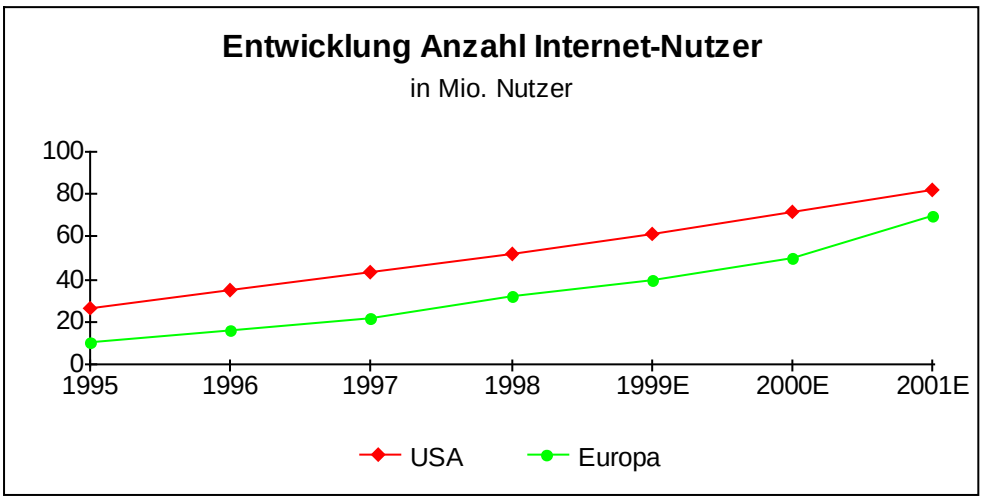


Abb. 1 Erwartete Entwicklung der Anzahl der Internetnutzer in Millionen Quelle: IDC

Das bedeutet, daß sich für die Unternehmen ein immer größer werdender Markt für den Absatz von Produkten entwickelt. Dementsprechend optimistisch – und auch sehr unterschiedlich - sind die Prognosen für die Entwicklung der mittels E-Commerce erzielbaren Umsätze. So sehen *Forrester Research*[3] u.a. den Umsatz im *Business-to-Business-Bereich* (B2B-Bereich) in den USA von 9 Mrd. USD im Jahr 1995 auf ca. 120 Mrd. USD im Jahr 2002 anwachsen. Im *Business-to-Consumer-Bereich* (B2C-Bereich) wird erwartet, daß sich der Umsatz, der 1995 noch bei 5 Mrd. USD lag, auf ca. 25 Mrd. USD im Jahre 2002 steigern wird (siehe Abb. 2).

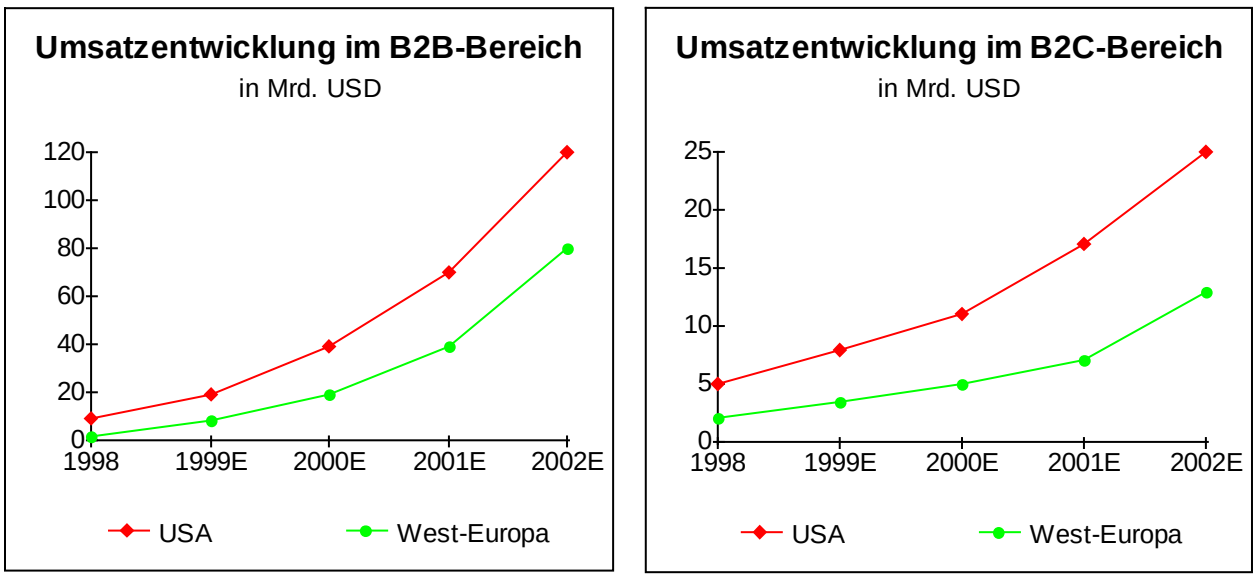


Abb.2 Erwartete Umsatzentwicklung im B2B- und B2C-Bereich Quelle: Forrester Research

[3] Forrester Research, http://www.forrester.com

Für Unternehmen bedeutet E-Commerce andererseits nicht nur die Möglichkeit, neue Märkte zu erschließen, sondern auch, Kosten zu senken: Im Einkauf (Ausschreibung auf Web-Site, Auktionen, Automatisches Ordern), Marketing (Onlinewerbung, Produktkataloge im Web) und Vertrieb (Direktvertrieb mittels E-mail, Verkauf über Web-Site, Ausnutzen von kurzfristigen Marktentwicklungen durch Anpassung von Preisen).

1.3 Transaktionen im E-Commerce

E-Commerce-Prozesse stellen bestimmte Anforderungen an Anwendungen:

- Ausführung von E-Commerce-Prozessen

Ein E-Commerce-Prozeß besteht in der Regel aus mehreren Einzeloperationen, für deren Ausführung folgende Forderungen gelten können:

1. „Alles-oder-nichts-Prinzip“

   Beispiel: Beim Drücken des Bestell-Buttons in einem Web-Store soll das Warenlager angewiesen werden, den Versand vorzunehmen, in der Buchhaltung wird der Verkauf vermerkt und die Versendung einer Bestätigung an den Kunden veranlaßt. Konnte die Benachrichtigung des Warenlagers zum Beispiel aufgrund eines Kommunikationsfehlers nicht vorgenommen werden, wird es nicht erwünscht sein, daß der Kunde eine Bestätigung und die Rechnung bekommt, obwohl die Ware (bis zur Beschwerde des Kunden) nicht ausgeliefert wird. In diesem Fall sollte der Vermerk in der Buchhaltung zurückgesetzt und das Bestätigungsschreiben nicht abgeschickt werden.
2. Partielle Ausführung soll möglich sein

   Beispiel: Ein Kunde bestellt eine Reise bestehend aus Flug, Hotel, Mietwagen. Bei der Buchung können Flug und Hotel gebucht werden, der gewünschte Mietwagen ist aber nicht erhältlich. In diesem Fall ist es nicht wünschenswert, daß die Flugreise und das Hotel storniert werden, obwohl die Einzeloperation „Auto mieten“ (vorerst) fehlgeschlagen ist.

- Bewältigung einer potentiell großen Anzahl gleichzeitiger Nutzer

E-Commerce-Anwendungen müssen eine potentiell große Anzahl gleichzeitiger Nutzer bewältigen können, ohne daß es dadurch zu Störungen der Konsistenz des Gesamtsystems kommt.

Beispiel: Ein Online-Händler registriert pro Sekunde hunderte gleichzeitiger Bestellungen für einen amerikanischen Bestseller, da er dieses Buch zu 10% des Listenpreises anbietet. Jede einzelne Bestellung muß vermerkt werden, um einen Überverkauf des Lagers zu vermeiden.

- Wiederherstellung eines konsistenten Prozeßzustandes nach einem Fehlerfall

E-Commerce-Anwendungen müssen sich nach einem Hardwareausfall (schnell) wieder in den vor der Unterbrechung zuletzt als konsistent vermerkten Zustand zurückversetzen lassen. Die Ergebnisse von vor der Unterbrechung ausgeführten Transaktionen dürfen dabei nicht verlorengehen.
Beispiel: Eine Finanztransaktion nimmt einen Guthabentransfer vor. Nachdem der Betrag von einem Konto abgebucht wurde, fällt der Rechner, noch bevor die Gutschrift vorgenommen werden konnte, aus. Nach Ersetzung des Rechners muß der Guthabentransfer entweder fortgesetzt oder aber die Abbuchung zurückgesetzt und der Transfer noch einmal vorgenommen werden.

- Ablauffähigkeit auf verteilten, heterogenen Systemen

Die Abwicklung von E-Commerce-Prozessen über Unternehmensgrenzen hinweg macht es notwendig, daß E-Commerce-Anwendungen auf heterogenen Systemen lauffähig sind, die geografisch verteilt und zu unterschiedlichen Organisationen gehören können.
Beispiel: Eine Rechtsanwaltskanzelei in Leipzig nimmt über das WWW Verträge zur Prüfung und Übersetzung in beliebigem Format (zum Beispiel *Word*) entgegen, konvertiert diese auf einem *WindowsNT*-System in eine *Framemaker*-Datei und reicht diese an einen Subunternehmer (Übersetzungsagentur) in München als Framemaker-Datei weiter, der Framemaker unter *HP-UX* nutzt.

- Unterstützung manueller Tätigkeiten

Die Abwicklung von E-Commerce-Prozessen kann sowohl automatisierte Operationen als auch manuelle Tätigkeiten beinhalten.
Beispiel: Verpackung einer bestellten Vase durch einen Lagermitarbeiter. Falls dieser die Vase (die letzte aus der bereits ausgelaufenen Produktion) beim Verpacken zerbricht, muß zum Beispiel die Buchung des Verkaufspreises storniert und der Kunde benachrichtigt werden können.

In DBMS befreien *ACID-Transaktionen* den Programmierer weitgehend von allen Aspekten des Fehlerfalls und der Nebenläufigkeit. Sie werden in Kapitel 2 vorgestellt. ACID-Transaktionen besitzen jedoch Nachteile und sind für den Einsatz in E-Commerce-Anwendungen nicht in jedem Fall geeignet. In der Literatur werden eine Reihe von Erweiterungen des klassischen ACID-Transaktionskonzeptes vorgeschlagen, von denen g*eschachtelten Transaktionen* und *Contracts* sowie ein Beispiel für die Anwendung von *Atomaritäts- und Isolationssphären* in Kapitel 3 beschrieben werden. In Kapitel 4 wird schließlich auf ein speziell für den Verkauf von digitalen Gütern entwickeltes, hochatomares Transaktionsprotokoll eingegangen.

2. ACID-Transaktionen

2.1 ACID-Transaktionen in Datenbanken

Eine Transaktion ist eine ununterbrechbare Folge von DML-Befehlen, welche die Datenbank von einem logisch konsistenten Zustand in einen neuen logisch konsistenten Zustand überführt [RA97]. Dabei gelten die sogenannten ACID-Eigenschaften:

A	=	Atomicity:	„Alles-oder-nichts-Eigenschaft“.
C	=	Consistency:	Eine erfolgreiche Transaktion erhält die DB-Konsistenz.
I	**=**	Isolation:	Die Transaktion wird im logischen Einbenutzerbetrieb durchgeführt.
D	=	Durability:	Die Ergebnisse einer erfolgreich beendeten Transaktion werden dauerhaft gesichert, selbst wenn es zu beliebigen, erwarteten Fehlern kommt.

Eine ACID-Transaktion befreit den Datenbank-Anwendungsprogrammierer somit weitgehend von Aspekten des Fehlerfalls und der Nebenläufigkeit.

Besteht eine E-Commerce-Anwendung zum Beispiel nur aus einem Anwendungsprogramm und einer Datenbank, so unterscheidet sich der Einsatz von Transaktionen nur unwesentlich von herkömmlichen Datenbankanwendungen: Als Beispiel könnte hier eine Anwendung genannt werden, die über ein Web-Interface die Bestellung eines Produktes ermöglicht.

Während des Bestellvorganges gibt der Kunde dabei sowohl seine Rechnungsanschrift als auch die gewünschten Produktparamter ein. Vor Ausgabe der Bestätigung sollte gewährleistet sein, daß alle Daten in den (verschiedenen) Tabellen der Datenbank gespeichert werden konnten. Dies könnte durch eine ACID-Transaktion sichergestellt werden.

2.2 Nachteile von ACID-Transaktionen

Sobald jedoch eine vollständige Zurücksetzung der Transaktion im Fehlerfall unerwünscht ist, Systeme unterschiedlicher Unternehmen oder auch Personen in eine Transaktion involviert sind, stößt eine ACID-Transaktion auf ihre Grenzen.
Im folgenden werden nach [OH96] Bereiche aufgezeigt, in denen die Verwendung von ACID-Transaktionen in E-Commerce-Anwendungen zu Problemen führen kann:

- **Transaktionen, die im Fehlerfall nur teilweise zurückgesetzt werden sollen**
  Das Problem: Beispielsweise kann man sich hier eine Transaktion vorstellen, die eine Hotelbuchung, den Kartenkauf für ein Konzert und die Buchung einer Zugfahrt umfaßt. Hotelbuchung und Kartenkauf wurden erfolgreich durchgeführt, aber der gewünschte Zug ist überfüllt. Die Folge einer ACID-Transaktion wäre die Rücksetzung der bereits erfolgreich durchgeführten Buchungen, was durchaus nicht immer wünschenswert ist.

- **Transaktionen, die manuelle Operationen einschließen**
  Das Problem: Ein Mitarbeiter eines Reisebüros läßt sich die Last-Minute-Angebote anzeigen und wird durch das System zu einer Entscheidung aufgefordert, während das System einen Lock auf diese Angebote gesetzt hat. Der Mitarbeiter beschließt aber, erst einmal eine Pause zu machen. Der Lock auf die Angebote blockiert weitere Buchungen.

- **Transaktionen, die über Firmengrenzen hinweg verlaufen**
  Das Problem: Kaum ein Unternehmen wird es zulassen, daß seine Systeme von einem fremden System aus gesteuert werden, wie es bei einer nach dem *Zwei-Phasen-Commit-Protokoll* durchgeführten ACID-Transaktion der Fall wäre.

3. Erweiterungen des klassischen Transaktionskonzeptes

In der Literatur finden sich zahlreiche Erweiterungen des klassischen Transaktionskonzeptes, von denen geschachtelte Transaktionen, das ConTract-Modell und ein Beispiel für Atomaritäts- und Isolationssphären im folgenden vorgestellt werden. Die Erläuterung der Konzepte geschachtelte Transaktion und ConTract-Modell erfolgt auf der Grundlage von [RM99].

3.1 Geschachtelte Transaktionen

Geschachtelte Transaktionen ermöglichen eine Einschränkung des Rücksetzbereiches einer Transaktion. Eine geschachtelte Transaktion T (hier als *Top-Level-Transaktion* bezeichnet) besteht aus mehreren Subtransaktionen S_1, S_2, ..., S_n, von denen jede wiederum aus Subtransaktionen bestehen kann (siehe Abb. 3)

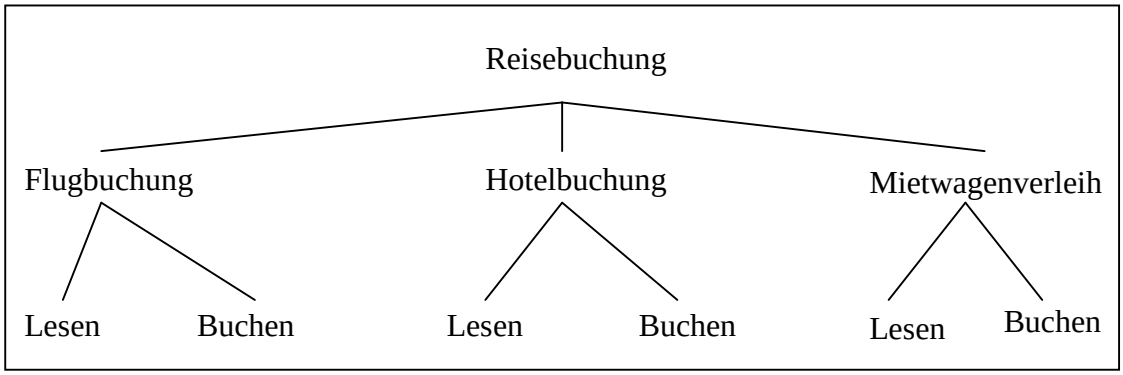


Abb. 3 Beispiel für eine geschachtelte Transaktion Quelle: nach [RM99]

Jede Subtransaktion S ist dabei atomar und separat rücksetzbar. Das bedeutet, daß im Fehlerfall nicht die gesamte Arbeit verlorengeht. Beim Scheitern einer Subtransaktion kann diese wiederholt werden oder eine anderweitige Korrektur (zum Beispiel eine alternative Subtransaktion) durchgeführt werden.
Die Änderungen einer Subtransaktion werden bei ihrem Commit nach außen hin (für andere Top-Level-Transaktionen) noch nicht sichtbar. Erst wenn die Top-Level-Transaktion ihr Commit durchgeführt hat, werden auch die Änderungen der Subtransaktionen für andere Transaktionen sichtbar. Das bedeutet, daß zwar eine Aufweichung des Atomaritätskriterium stattfindet, aber weiterhin strikte Isolation angewendet wird: Ein Zugriff auf die durch T bzw. S gesperrten Datenobjekte vor dem Commit von T ist nicht möglich.

Geschachtelte Transaktionen sind für die Anwendungsentwicklung durch kommerzielle Produkte wie den *TP Monitor Encina* von *Transarc* oder den *Microsoft Transaction Server* verfügbar [BN97].

3.2 ConTracts

Das ConTract-Modell wurde von *Wächter* und *Reuter* an der Universität Stuttgart entwickelt und soll ein Ausführungsmodell für die zuverlässige Abwicklung langlebiger und verteilter Workflows bereitstellen. Für die Realisierung von E-Commerce-Anwendungen ist das

ConTract-Modell interessant, da es nicht nur eine Einschränkung des Rücksetzbereiches, sondern auch ein Forward Recovery bietet. Ein *ConTract* ist die konsistente und fehlertolerante Ausführung einer Anzahl von Aktionen (*Steps*), die in einer separat spezifizierten Kontrollflußbeschreibung (*Script*) sequentiell oder parallel miteinander verknüpft sind. Das bereits besprochene Problem einer Reisebuchung, bestehend aus einer Flugreservierung, einer Hotelbuchung und einer Reservierung bei einem Autoverleih, ist in Form eines ConTracts in Abb. 4 dargestellt.

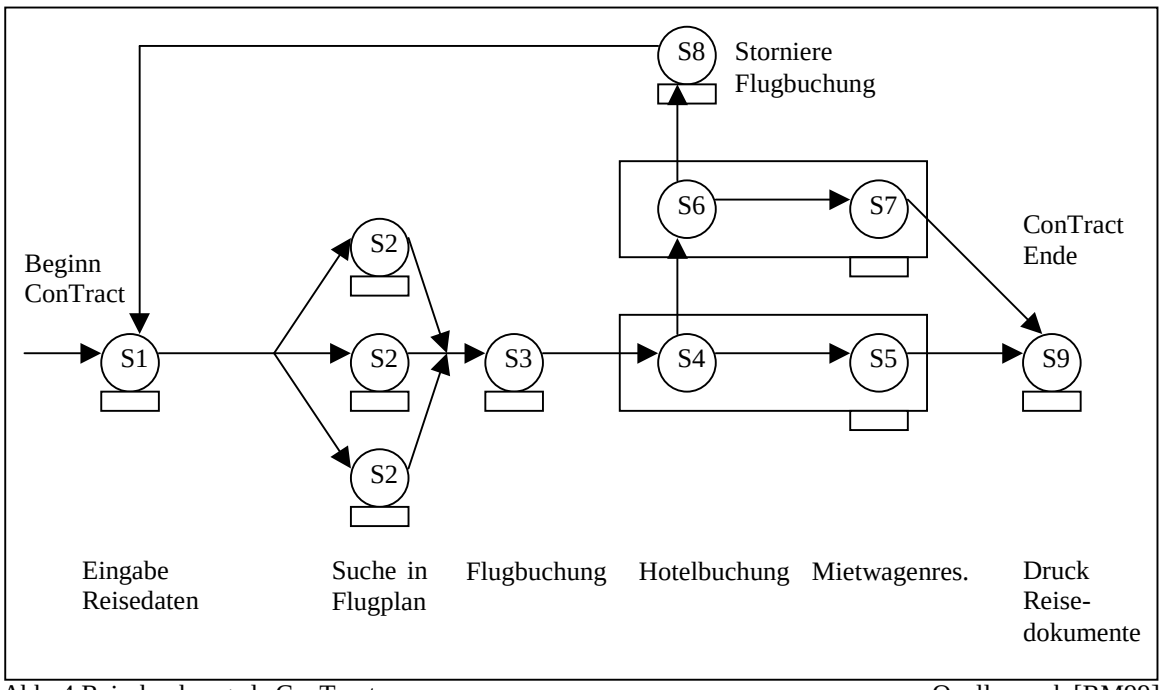


Abb. 4 Reisebuchung als ConTract Quelle: nach [RM99]

Der Step Flugbuchung würde zum Beispiel folgendermaßen definiert werden können:

```
STEP Flugbuchung
      DESCIPTION: Reserviere n Sitze eines Fluges und bezahle sie
      IN:               fluggesellschaft:       STRING;
                        flugnummer:             STRING;
                        datum:                  DATE;
                        anz_sitze:              INTEGER;
                        ticketpreis:            DOLLAR;
      OUT:              status                  INTEGER;

flugbuchung()
{     char* flugnummer;
      long  datum;
      int   anz_sitze;
      ...
      EXEC SQL
            UPDATE      Reservierungen
            SET         besetzte_Sitze = besetzte_Sitze+ :anz_sitze
            WHERE flug = :flugnummer      AND
                        datum = :datum          ...
      END SQL
      ...
}
```

Der Kontrollfluß der Reisebuchung wiederum kann mittels folgender Kontrollflußkonstrukte definiert werden:

- Sequenz bzw. Block:

  `BEGIN <block> END`

  (<block>: Sequenz von Programmschritten oder Kontrollflußanweisung):
- Verzweigung:

  `IF <condition> <block>  [ELSE <block>]`

  `CASE <contextelement>:<block> [...[<contextelement>: <block>]] ENDCASE`
- Schleife:

  `WHILE <condition> <block> ENDWHILE`

  `FOR (<value1> TO <value2>) <block> ENDFOR`
- Fork and Join paralleler Abschnitte:

  `PARBEGIN <block> PAREND`
- Parallele Schleife:

  `PARFOREACH (<contextelement>: <type> IN <set>) <block> ENDPARFOREACH`

Damit ergibt sich folgende Ablaufsteuerung für das Beispiel:

```
CONTRACT Geschaeftsreise
CONTEXT-DECLARATION
      ticketpreis:      dollar;
      von, nach:        city;
      datum:            date_type;
      ok:               boolean;
      ...
CONTROL-FLOW-SCRIPT
      S1:   dateneingabe  (in_context:  ;  out_context:  datum,  von,  nach,
anz_sitze, ...);
      PARFOREACH(  fluggesellschaft:  EXECSQL  select  fluggesellschaft  from
... ENDSQL)
            S2:   suche_flug_heraus  (in_context:  fluggesellschaft,  datum,
von, nach, anz_sitze; out_context: flugnummer, ticketpreis);
      ENDPARFOREACH
      S3:   flugbuchung(in_context:  fluggesellschaft,  flugnummer,  datum,
...)
      S4:   hotelbuchung(in_context:     „Regent     Beverly     Hilshire“;
out_context: ok, reservierungsnr);
      IF (ok[4]) THEN S5: autoreservierung(..., „Sixt“, ...);
      ELSE BEGIN
            S6: hotelbuchung(..., „Royal Plaza“, ...);
            IF (ok[6]) THEN S7: autoreservierung (..., „Europcar“, ...);
            ELSE S8: storniere_flug(...);
      END
      S9: drucke_reisedokumente(...);
END_CONTROL-FLOW-SCRIPT
...
...
END-CONTRACT Geschaeftsreise
```

Jeder Step wird in Form einer ACID-Transaktion ausgeführt. Mehrere Steps können zu einer Atomaritätssphäre zusammengefaßt werden, für die im Fehlerfall ein ROLLBACK durchgeführt werden kann. Das bedeutet aber, daß ein ConTract im allgemeinen keine ACID-Transaktion ist. Ein (semantisches) ROLLBACK eines ConTracts kann allerdings (explizit durch Administrator oder Benutzer veranlaßt) durchgeführt werden, indem Kompensations-Steps ausgeführt werden. Dies kann zum Beispiel beim endgültigen (semantischen) Scheitern eines ConTracts (Reisender bricht sich vor der Abreise ein Bein) notwendig sein.
Die Kompensationsanweisungen für das Beispiel könnten lauten:

```
COMPENSATIONS
      C1: mache_nichts();
      C2: mache_nichts();
      C3: storniere_flug(...);
      C4: storniere_hotel(...);
      C5: storniere_auto(...);
      C6: storniere_hotel(...);
      C7: storniere_auto(...);
      C8: mache_nichts();
      C9: entwerte_reiseunterlagen();
END_COMPENSATIONS
```

Das Forward Recovery bei ConTracts wird durch die Speicherung des Verarbeitungs-zustandes (Kontext) eines ConTracts in einer transaktionsgeschützten Kontext-Datenbank erreicht. Ein Kontext besteht dabei u.a. aus:
- Benutzereingaben
- Ein- und Ausgabevariablen von Steps

Dabei wird der alte Kontext nicht überschrieben, sondern eine chronologische Speicherung vorgenommen. Nachdem eine Unterbrechung zum Beispiel durch einen Hardwarefehler aufgetreten ist, kann nach Beseitigung der Unterbrechungsursache der Kontext zum Unterbrechungszeitpunkt ausgelesen und die Abarbeitung wieder aufgenommen werden.
Abb. 5 zeigt ein Beispiel für eine Kontext-Datenbank.

Für einen ConTract ist weiterhin die Vereinbarung von anwendungsspezifischen Isolationsprädikaten möglich. Die Isolationsprädikate werden eingeführt, da strikte Isolation bei langlebigen Prozessen in der Regel zu restriktiv ist. Isolationsprädikate spezifizieren zulässige Änderungsoperationen.
Isolationsprädikate im Beispiel Reisebuchung:

S1 überprüft, ob Budget >= Reisekostenlimit

Falls JA, wird S1-Ausgangs-Isolationsprädikat P_{1A}= (budget >= reisekostenlimit) generiert und bekannt gemacht.

Kontext-Datenbank						
ConTract-ID	Erzeuger-Step	Kontext-Variable			Zeitstempel	
		Name	Typ	Wert	Uhrzeit	Datum
1317	S1	von	string	Leipzig	16:18	29.6.99
1317	S1	nach	string	Hawaii	16:18	29.6.99
...	...	...	...	...	...	...
1317	S2	flugnummer	string	UA171	16:19	29.6.99
...	...	...	...	...	...	...
1317	S4	hotelname	string	Holiday Inn	16:25	29.6.99

Abb.5 Kontext-Datenbank Quelle: nach [RM99]

Andere ConTracts dürfen nun Budget-Einträge unter Beachtung der aktuellen Isolationsprädikate ändern. (Zum Beispiel ist eine Erhöhung des Budgets möglich). Ausgangsprädikate wie P_{1A} dienen zudem nachfolgenden Steps als Eingangsprädikate P_{1E} zur Konsistenzüberprüfung. Zum Vergleich: Bei ACID-Transaktionen besagt das Isolationsprädikat, daß Werte von Objekten während der ganzen Ausführung der Transaktion den Wert beibehalten, den die Transaktion zu Beginn „gesehen“ hat.

ConTract ist bisher nur als Prototyp-Implementierung in Form des *Apricots-Systems* an der Universität Stuttgart verfügbar.

3.3 Atomaritäts- und Isolationssphären

3.3.1 Das WISE-Projekt

Das *WISE*- (Workflow based Internet Services, [AF98]) Projekt verfolgt das Ziel, eine Software-Infrastruktur zu erschaffen, die es Unternehmen ermöglicht, E-Commerce in Form eines virtuellen Unternehmens durchzuführen.

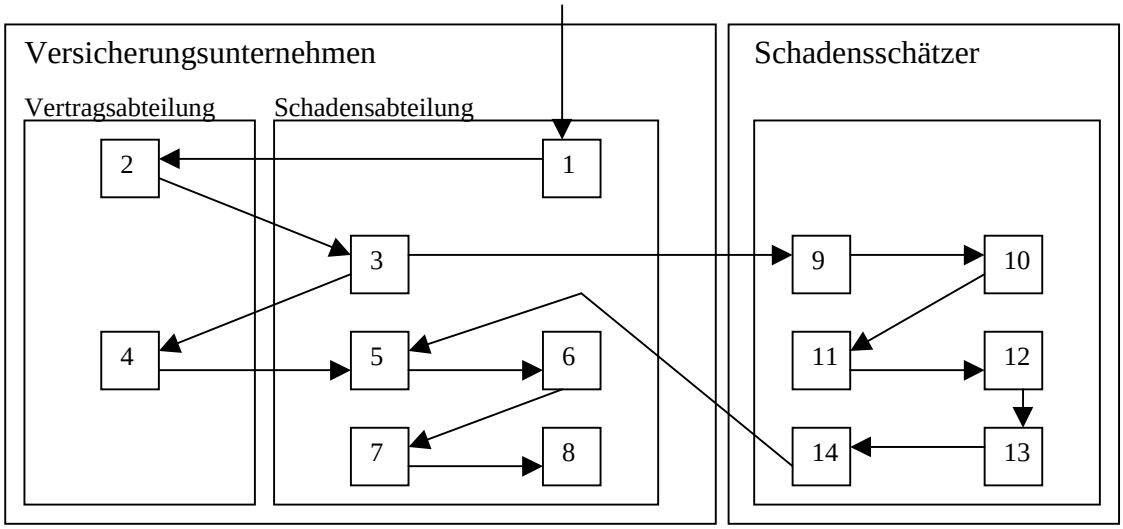
Dabei soll ein System entstehen, das es ermöglicht, virtuelle Geschäftsprozesse zu definieren, zur Ausführung zu bringen und zu überwachen. Ein virtueller Geschäftsprozeß ist ein Geschäftsprozeß, der unter Ausnutzung der Kommunikationstechnologie über Unternehmensgrenzen hinweg durchgeführt wird. Ein virtuelles Unternehmen wird in diesem Zusammenhang als ein Unternehmen mit virtuellen Geschäftsprozessen definiert.

Das WISE-Projekt beeinhaltet eine *Internet Workflow Engine*, die als eine Art zugrundeliegendes, verteiltes „Betriebssystem“ die Ausführung von Geschäftsprozessen kontrolliert, ein Prozeß-Modellierungstool, um die Geschäftsprozesse zu definieren und zu überwachen, einen Katalog, um die bereitgestellten Unternehmens-Services anderen bekannt

machen zu können und eine multimediale Kommunikationsumgebung für die Angestellten. Dabei implementiert das Projekt Transaktionsmechanismen als eine Möglichkeit, um Ausführungsgarantien bereitzustellen.

3.3.1.1 Beispielanwendung

Die folgende, ebenfalls aus [AF98] entnommene Beispielanwendung, die nach Angaben der Autoren tatsächlich implementiert wurde, soll das Einsatzspektrum von WISE verdeutlichen. Das vorzustellende Szenario ist in Abb. 6 dargestellt.



Legende:

1 Erhalt der Schadensmeldung	2 Überprüfung der Kundendaten	3 Schadensklassifikation
4 Raub: Berechnung Obergrenze	5 Endgültige Empfehlung für Zahlung	6 Auszahlung
7 Update Kundenstatus	8 Fall abgeschlossen	9 Überprüfung der erhalt. Daten
10 Plausibilitätscheck	11 Vereinbarung Temin mit Kunden	12 Besichtigung
13 Vergleich mit ähnlichen Fällen	14 Empfehlung für Zahlung	

Abb.6 Beispiel für ein virtuelles Unternehmen Quelle: nach [AF98]

Die Handelsgemeinschaft wird in diesem Beispiel aus 2 Abteilungen eines Versicherungsunternehmens und einem selbständigen Schadensschätzer gebildet. Das Versicherungsunternehmen definiert nun einen virtuellen Prozeß, um Schadensersatzansprüche bearbeiten zu können.

Im ersten Schritt erhält die Schadensabteilung des Versicherungsunternehmens die Schadensmeldung und sammelt alle benötigten Informationen über den Schaden, den Kunden und eventuell andere beteiligte Parteien (1). Diese Daten werden in die Vertragsabteilung übermittelt, die mittels eines SAP-Systems die Daten des Kunden überprüft (2). Nachdem festgestellt wurde, ob überhaupt ein Versicherungsvertrag für diese Art von Schaden existiert,

geht der Fall wieder zurück an die Schadensabteilung, die den Schaden genau klassifiziert (3). Im Beispiel sollen dabei nur zwei Schadensklassen Beachtung finden: Raub und Feuer. Im Falle eines Raubes wird der Fall wieder an die Vertragsabteilung übergeben, die anhand des Polizeireportes die gestohlene Gesamtsumme berechnet (4). Danach wird wieder in der Schadensabteilung die Zahlungsobergrenze aus dem Vertrag ermittelt und eine endgültige Empfehlung abgeben, wieviel die Versicherung bezahlen sollte (5). Die Auszahlung erfolgt (6), wird vermerkt (7) und der Fall ist abgeschlossen (8).
Im Fall des Feuerschadens arbeitet die Versicherung dagegen mit einem Schadensschätzer zusammen, der allein eine Empfehlung über die Höhe des zu zahlenden Betrages abgibt.
Dies geschieht durch eine Analyse der übermittelten Daten (9), einem Check des Sachverhaltes auf Plausibilität (10), der Vereinbarung eines Besichtigungstermins (11), der Besichtigung des „Tatortes“ (12), dem Studium ähnlicher Fälle (13) und der Abgabe einer Schätzung über den zu zahlenden Betrag (14). Interessant ist hierbei die Einbindung des Schadensschätzer in den Gesamtprozeß „Schadensbearbeitung“, wobei der Schadensschätzer ein vollkommen eigenständiges Unternehmen ist. Dabei stellt der Schadensschätzer einen „Einstiegspunkt“ bereit, über den er die notwendigen Daten erhalten und seine eigenen Prozesse triggern kann.

3.3.2 Atomaritäts- und Isolationssphären in WISE

Die WISE-Engine garantiert die Ausführung eines Prozesses in dem Sinne, daß ein Prozeß immer in einem konsistenten Zustand beendet wird. Dies wird erreicht, indem im Fehlerfall entweder alle Änderungen, die der Prozeß ausgeführt hat, rückgängig gemacht werden oder indem er gezwungen wird, mit einer Folge von alternativen Aktionen zu terminieren, deren Ergebnisse bekannt sind.

Die Realisierung basiert dabei auf der Einführung von *Atomaritäts- und Isolationssphären.*
Leider wird die Verwendung eines Sphärenmodells in [AF98] nur erwähnt und auch in anderen Publikationen wie [SS98] nur oberflächlich behandelt.
Demnach schlagen die Autoren vor, Atomarität und Isolation vollständig zu entkoppeln. Grundlage hierfür sind Sphären für Atomarität und Isolation. Voraussetzung der Definition von Sphären ist zunächst eine workflow-ähnliche Beschreibung von Aktivitäten und eine partielle Ordnung auf diesen Aktivitäten. Es entsteht das Modell einer sogenannten *Transactional Coordination* (TC), das eine Menge erlaubter Ausführungspfade (Multi-Pfade)

von der Start-Aktivität eines Prozesses zu einer gültigen Teilmenge von Terminal-Aktivitäten beschreibt.

Eine Atomaritätssphäre SA (*sphere of atomicity*) ist ein zusammenhängender Teilgraph (gerichtet, azyklisch) einer transaktionellen Koordination TC, für den immer eine gültige Ausführung garantiert werden kann. Die Ausführung erfolgt, indem entweder ein Terminal-Knoten der SA erreicht wird oder aber SA durch eine semantische Kompensation zurückgefahren wird.

Durch Isolationssphären SI (*sphere of isolation*) ist es möglich, bestimmte Aktivitäten einer transaktionellen Koordination TC vor den konkurrierenden, parallelen Aktivitäten anderer TCs zu schützen (bzgl. Zugriff auf gemeinsame Datenobjekte). Dabei ist hervorzuheben, daß zur Bestimmung der Konfliktrelation neben den Aktivitäten selbst auch ihr Kontext beachtet wird. So können zwei Aktivitäten in bestimmten Sphären in Konflikt stehen. Dieselben Aktivitäten können jedoch ebenso in anderen Sphären nicht in Konflikt stehen. Eine Isolationssphäre verfügt über Informationen, zu welchen Aktivitäten außerhalb der Isolationssphäre ein Konflikt besteht. Diese Informationen liefert die Konfliktfunktion $Con_{SI}:\{Aktivitäten_{SI}\} \rightarrow \{Aktivitäten\}$. Angenommen, $SI \ni a_1$ soll ausgeführt werden. $Con_{SI}(a_1)$ liefert z.B. $\{a_2\}$. Falls nun die kontextabhängige Überprüfung feststellt, daß ein Konflikt besteht, kann die Ausführung serialisiert werden, um den Konflikt zu lösen.
Falls eine TC in eine Atomaritäts- und eine Isolationssphäre eingebettet ist, besitzt sie, falls die Konfliktpaare jeder Aktivität nicht vom Kontext der Sphären abhängig sind, die vom ACID-Modell her bekannte Semantik von Atomarität und Isolation.

Das WISE-Projekt befindet sich wie das Apricots-System im Prototyp-Stadium.

4. Atomarität in E-Commerce-Protokollen

4.1 Atomaritätsstufen

In Zusammenhang mit Datenbanken fordert die Atomaritätseigenschaft bei Transaktionen, daß eine Folge von Operationen so ausgeführt wird, daß entweder alle Operationen erfolgreich ausgeführt werden oder aber, falls eine der Operationen fehlschlägt, das System sich in einem Zustand befindet, als ob keine der Operationen ausgeführt worden wäre.

Protokollen, die im E-Commerce Anwendung finden, können nach [TY98] folgende Atomaritätsstufen zugeordnet werden:

Def.: Geldatomarität
Protokolle mit Geldatomarität schließen die Erschaffung oder Vernichtung von Geld bei Guthabenübertragungen aus.
Beispiel: Ausführung einer elektronischen Überweisung von Konto K1 zu Konto K2. Der Betrag, der von K1 abgebucht wird, muß in gleicher Höhe Fall K2 gutgeschrieben werden.

Def.: Warenatomarität

Protokolle mit Warenatomarität sind geldatomar und stellen sicher, daß eine Ware genau dann geliefert wird, wenn der Guthabentransfer stattgefunden hat.
„Beispiel“: Eine Warenlieferung per Nachnahme kommt als Nicht-E-Commerce-Protokoll der geforderten Warenatomarität sehr nahe: Ein Paket wird nur dann übergeben, wenn es bezahlt wird.

Def.: Zertifizierte Lieferung

Protokolle mit zertifizierter Lieferung sind geldatomar, warenatomar und ermöglichen es Kunden und Lieferanten, genau zu beweisen, welche Ware geliefert wurde.
Beispiel: Es soll möglich sein, zum Beispiel nachzuweisen, daß ein geliefertes Dokument mit dem Titel „Transaction Management in E-Commerce“ in Wahrheit nur eine Abhandlung über Nordamerika enthält. Anhand der Lieferung per Nachnahme läßt sich dies wieder verdeutlichen: Wenn das Paket bei der Bezahlung geöffnet und vom Postbeamten der genaue Inhalt protokolliert werden kann, so handelt es sich um eine zertifizierte Lieferung.
Insbesondere, wenn sich Kunden und Händler mißtrauen, sind Protokolle mit zertifizierter Lieferung notwendig. Im WWW gibt es keine Möglichkeit, einen vertrauenswürdigen Händler von einem betrügerischen Unternehmen anhand des Online-Stores zu unterscheiden.

Bemerkenswert ist, daß eine Vielzahl von Protokollen, die heute für E-Commerce-Applikationen verwendet werden, nicht oder nur teilweise über die obengenannten Eigenschaften verfügen.

Als *atomare Protokolle* sollen im folgenden Protokolle verstanden werden, die über zertifizierte Lieferung verfügen.

4.2 Nicht-atomare Protokolle

Folgende in E-Commerce-Anwendungen verwendeten Protokolle erfüllen die geforderten Atomaritätseigenschaften nicht oder nur teilweise:

4.2.1 Digicash

Digicash ermöglicht die anonyme Bezahlung im Internet. Folgende Schritte illustrieren die Arbeitsweise von Digicash:

a) Ein Kunde hebt Geld bei der Bank ab und erhält ein verschlüsseltes Token, das wie Geld benutzt werden kann.
b) Der Kunde fügt eine Verschlüsselung hinzu, die es dem Verkäufer ermöglicht, die Echtheit zu überprüfen, aber nicht mehr die Identität des Kunden herauszufinden.
c) Der Kunde kauft bei Verkäufer, wobei wiederum das Token kryptografisch verändert wird, indem nun auch die Daten des Verkäufers einfließen.
d) Der Verkäufer überprüft, ob er das Token nicht eventuell schon vorher erhalten hat.
e) Der Verkäufer schickt die Ware an den Kunden.
f) Der Verkäufer schickt die erhaltenen Token zur Bank.
g) Die Bank überprüft, ob das Token wirklich einzigartig ist, wobei die Identität des Kunden weiterhin nicht sichtbar bleibt. Eine Ausnahme bildet der Fall, wenn der Kunde das Token mehrfach verwendet hat. Dann wird seine Identität aufgedeckt und der Fall als Straftat verfolgt, da die Mehrfachverwendung eines Tokens ein Betrugsversuch ist.

Nun ist folgendes Scenario denkbar: In Punkt c) kommt es zu einer Unterbrechung der Kommunikationsverbindung. Der Kunde hat keine Möglichkeit herauszufinden, ob der Verkäufer das Token erhalten hat oder nicht. Der Kunde hat nun zwei Optionen:

1. Der Kunde kann das Token zur Bank bringen oder es ein weiteres Mal „ausgeben“. Hat nun aber der Verkäufer das Token erhalten und will es bei der Bank einlösen, wird der Kunde des Betruges verdächtigt und verfolgt!
2. Der Kunde unternimmt nichts. Hat nun der Verkäufer das Token aber nicht erhalten, wird der Kunde nie seine Ware sehen. Da der Kunde aber nicht mit Sicherheit sagen kann, ob Verkäufer nur vorgibt, das Token nicht erhalten zu haben, gibt er sein Geld nicht aus – aus Angst, des Betruges verdächtigt zu werden.

Die Möglichkeit des Händlers, den Erhalt des Tokens abzustreiten, verletzt die geforderten Eigenschaft der Warenatomarität.

4.2.2 SSL

SSL wurde von *Netscape* entwickelt und nutzt asymmetrische Verschlüsselungsverfahren, um einen gesicherten Kommunikationskanal aufzubauen. D.h., die übertragenen Daten (z. B. Kreditkarteninformationen) sind selbst nicht verschlüsselt, aber die Verbindung zwischen Client (Browser) und WWW-Server. Werden Kreditkarteninformationen übertragen (was dem Verlesen der Kreditkartennummer über eine gesicherte Telefonleitung entspricht), so kann man Geldatomarität in dem Grad erreichen, wie sie von Kreditkarten unterstützt wird.

Damit ist klar, daß SSL keine Warenatomarität unterstützen kann, da es die Verschlüsselung des Übertragskanals, nicht aber die übertragenen Daten selbst betrifft.

4.2.3 SET

Secure Electronic Transaction (SET) wurde von Visa und Mastercard aus einer Vielzahl von bereits veröffentlichten Protokollen wie zum Beispiel STT (Visa/Microsoft) und SEPP (MasterCard) entwickelt. Auch SET nutzt die in der realen Welt vorhandenen Kreditkarten, um Zahlungen zu ermöglichen. Dabei signiert der Kunde eine Bestellung und den dazugehörigen Preis und verschlüsselt seine Kreditkartennummer mit dem Public Key der Bank. Der Verkäufer nimmt den Auftrag an und leitet die Bestellung mit den verschlüsselten Zahlungsinformationen an die Bank weiter. Die Bank kann nun die Preise vergleichen und im Falle der Übereinstimmung kann sie mit ihrem Private Key die Kreditkartennummer entschlüsseln, den Betrag abbuchen und den Verkäufer informieren, daß er die Ware ausliefern kann.

Damit ist klar, daß auch hier Warenatomarität nicht unterstützt wird, da SET nur den Bezahlvorgang, nicht aber die Warenlieferung addressiert.

4.3 Beispiel für ein atomares Protokoll: NetBill

4.3.1 Vorbemerkung

Das *NetBill-Protokoll* wurde von Tygar u.a. entwickelt [TY95] und verfügt über alle 3 Stufen der Atomarität: Geldatomarität, Warenatomarität und zertifizierte Lieferung. Es ist auf den Austausch von digital übertragbaren Waren, d.h. Informationen, spezialisiert. Informationslieferung und Bezahlung werden dabei in einer atomaren Transaktion zusammengefaßt. Die folgenden Ausführungen sind [TY95] entnommen.

4.3.2 Das NetBill-Transaktionsmodell

Das NetBill-Protokoll vermittelt zwischen 3 Parteien: Dem Händler, dem Kunden und dem NetBill-Server (siehe Abb. 7). Zwischen Händler und NetBill-Server erfolgt eine Kommunikation dabei nur im Fehlerfall oder für administrative Funktionen.

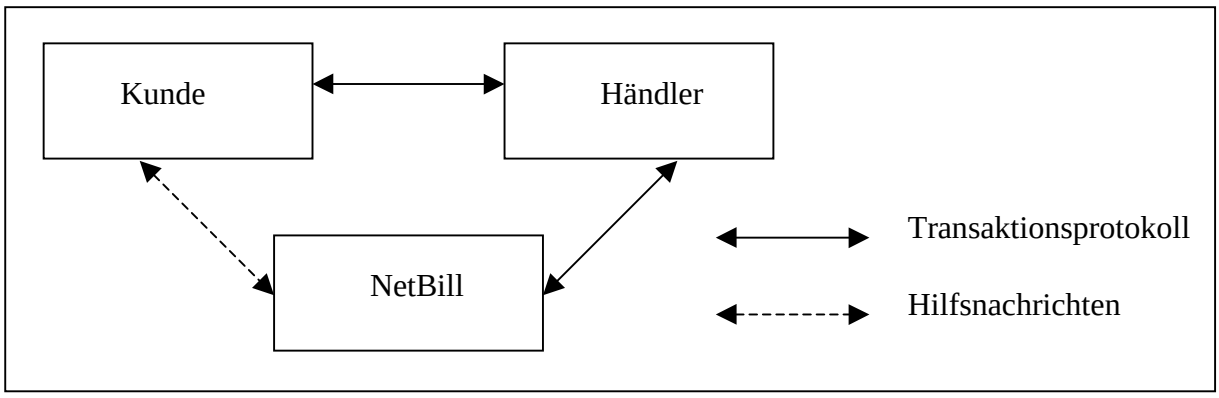


Abb. 7 Parteien einer Netbill-Transaktion Quelle: [TY95]

4.3.2.1 Eigenschaften

Eine Netbill-Transaktion verfügt über folgende Eigenschaften:

a) Nur autorisierte Kunden können ein Netbill-Konto belasten.

b) Kunde und Händler müssen in der Ware, die gekauft werden soll, und im Preis, der bezahlt werden soll, übereinstimmen.

c) Kunde und Händler werden vom Netbill-Server mit einen Nachweis über das Transaktionsergebnis benachrichtigt.

Zusätzliche Eigenschaften für Preisabstimmung und Lieferung der Ware sind:

d) Es existiert eine Phase der Angebotserstellung und Preisabstimmung zwischen Händler und Kunde

e) Ein Kunde erhält die bestellte Ware gdw. wenn sein Konto belastet wurde und der gleiche Betrag dem Händler gutgeschrieben wurde.

allgemein gilt:

f) Die Kommunikation soll in allen Phasen der Kaufs gegen Ausspähung und Veränderung geschützt werden.

4.3.3 Das NetBill-Transaktionsprotokoll

Im folgenden sollen die einzelnen Schritte des Transaktionsprotokolls dargestellt werden. Dabei bedeutet die Notation „X → Y“, daß X an Y eine Nachricht schickt. C = Kunde (Costumer), M = Händler (Merchant), N = NetBill-Server, K = Schlüssel (Key):

1) C → M

   Der Kunde erfragt den Preis für eine Ware beim Händler. Dieser Schritt ist eingebaut, um es dem Händler zu ermöglichen, eine Preisdiskriminierung durchzuführen. So kann er zum Beispiel den Preis für einen Studenten um 20% senken.

2) M → C

   Der Händler macht dem Kunden ein Angebot.

3) C → M

   Der Kunde teilt dem Händler mit, daß er das Angebot akzeptiert.

4) M → C

   Der Händler schickt die Ware (Information), verschlüsselt mit K.

5) C → M

   Der Kunde bereitet eine sogenannte *Electronic Purchase Order* (EPO) vor, die folgende Angaben signiert enthält:

   - Preis,
   - Checksumme der verschlüsselten Ware,
   - ein Timeout.

   Die EPO sendet der Kunde zum Händler.

6) M → N

   Der Händler fügt den Schlüssel K zur EPO hinzu, signiert seinerseits die EPO, die er dann an den NetBill-Server schickt.

7) N → M

   Der NetBill-Server überprüft nun die Signatur und Gegensignatur der EPO. Er überprüft weiterhin, ob der Kunde über ein genügend großes Guthaben verfügt, um die

Ware zu bezahlen, und ob der Timeout noch nicht erreicht wurde. Ist alles o.k., transferiert der NetBill-Server den Betrag vom Kundenkonto zum Händlerkonto. Der NetBill-Server speichert den Schlüssel K und die Checksumme der verschlüsselten Ware. Er erstellt danach eine signierte Quittung, die den Schlüssel K enthält und schickt sie an den Händler.

8) $M \rightarrow C$

Der Händler schickt die Quittung weiter an den Kunden, der nun seine Ware mit dem enthaltenen Schlüssel entschlüsseln kann.

Diskussion der Eigenschaften:

Eigenschaft (a) ist erfüllt, da der Kunde erst von NetBill authentifiziert werden muß, bevor seine EPO in Schritt (6) akzeptiert wird.

Eigenschaft (b) wird erfüllt, da die EPO alle relevanten Angaben enthält und sowohl vom Kunden als auch vom Händler signiert wird.

Eigenschaft (c) wird durch die vom NetBill-Server verschickte, signierte Quittung erfüllt.

Eigenschaft (d) wird durch Schritt (1) und (2) erfüllt.

Eigenschaft (e) wird durch die Schritte (4)-(8) erfüllt: Alle Guthabentransfer geschehen auf dem NetBill-Server. Da der Transfer lokal auf einer transaktionsbasierten Datenbank durchgeführt wird, ist er insbesondere atomar. So kann Geld weder entstehen noch verschwinden. NetBill ist daher *geldatomar*.

Wenn ein Fehler infolge eines Verbindungsabbruchs oder eines Hardwareausfalls vor Schritt (7) auftritt, so wird kein Guthaben übertragen und der Kunde erhält nicht den Schlüssel K, den er braucht, um die Ware zu entschlüsseln. Falls Schritt (7) erfolgreich durchgeführt wird, ist der Schlüssel sowohl beim Händler als auch beim NetBill-Server gespeichert. Wenn nun ein Fehler auftritt und so zum Beispiel Schritt (8) nicht ausgeführt werden kann, kann der Kunde den Schlüssel jederzeit beim NetBill-Server oder beim Händler jederzeit nachträglich erhalten. NetBill ist daher *warenatomar*.

Da NetBill über Warenatomarität verfügt, ist sicher, daß ein Austausch von Ware gegen Geld stattgefunden hat. Wenn jetzt aber der Kunde reklamiert, daß er nicht die gewünschte Ware erhalten hat, so kann er die verschlüsselte Version einem Richter präsentieren. Da der Kunde die Checksumme der verschlüsselten Ware signiert hat und der NetBill-Server dies gespeichert hat, ist eindeutig feststellbar und vom Kunden beweisbar, daß die verschlüsselte Ware die war, die der Kunde ursprünglich erhalten hat und er keine Veränderungen daran vorgenommen hat. Der Richter kann nun mittels der Schlüssels K die Ware entschlüsseln und

überprüfen, ob sie der beworbenen Ware entspricht. NetBill besitzt daher die Atomaritätsstufe *zertifizierte Lieferung*.
Eigenschaft (f) wird durch eine Verschlüsselung und einen Integritätscheck aller übertragenen Nachrichten erreicht.

Das Netbill-System befindet sich im Prototyp-Stadium.[4]

5. Zusammenfassung

An E-Commerce-Anwendungen werden bestimmte Anforderungen hinsichtlich Korrektheit und Flexibilität der Programmausführung, Wiederherstellbarkeit eines konsistenten Programmzustandes nach einem Fehlerfall und der Bearbeitung vieler gleichzeitig ablaufender E-Commerce-Prozesse gestellt.

Transaktionen mit ACID-Eigenschaften sind aufgrund von bestimmten Nachteilen (keine Einschränkung des Rücksetzbereiches möglich, strikte Isolation bei langlaufenden E-Commerce-Prozessen oft zu restriktiv, keine manuellen Operationen vorgesehen) nicht immer für den Einsatz in E-Commerce-Anwendungen geeignet.

Erweiterte Transaktionskonzepte bieten Lösungsmöglichkeiten für bestimmte Problemstellungen im E-Commerce-Bereich wie die Möglichkeit zur Beschränkung des Rücksetzbereiches (geschachtelte Transaktionen) oder Forward Recovery und Lockerung der strikten Isolation (ConTracts), sind aber bis auf geschachtelte Transaktionen nicht in kommerziell anwendbaren Tools implementiert.

Die konsequente Übertragung der Atomaritätseigenschaft auf den E-Commerce-Bereich hat zur Entwicklung des NetBill-Transaktionsmodells geführt, das auf den Austausch von digital übertragbaren Waren, spezialisiert ist und das zertifizierte Lieferung bietet.

[4] NetBill: http://www.netbill.com

Literaturverzeichnis

[AF98] G. Alonso, U. Fiedler, C. Hagen, A. Lazcano, H. Schuldt, and N. Weiler: *WISE: Business to Business E-Commerce.* In: Proceedings of the 1st European Workshop on Workflow and Process Management (WPM´98) „Workflow Management Research Projects“, Zürich, October 1998.

[BN97] Bernstein, P.; Newcomer, E.: *Principals of Transaction Processing*, Morgan Kaufmann Publishers, Inc., San Francisco, S.160, 1997

[CT95] Cox, B; Tygar, J.D.; Sirbu, M.: *Netbill Security and Transaction Protocol,* Proceedings of the First USENIX Workshop on Electronic Commerce, S. 77-88, Juli 1995

[DA99] Dangelmaier, Wilhelm: *Electronic Commerce,* Vorlesungsfolien, S.11, Universität-GH Paderborn, 1999

[FP97] Fochler, K.; Perc, P.; Ungermann, J.: *Electronic Commerce mit Lotus Domino*, Addison-Wesley, S.7, 1997

[OH96] Orfali, R.; Harkey, D.; Edwards, J.: *The Essential Client/Server Survival Guide*, John Wiley & Sons, Inc., New York, Toronto, Singapore, u.a., S.266ff., 1996

[RA97] Rahm, E.: *Datenbanksysteme I*, Vorlesungsscript WS 1997/1998, S. 7-19, 1997

[RM99] Rahm, E.; Müller, R.: *Workflow-Management-Systeme*, Vorlesungsscript SS 1999, S. 4-11ff., 1999

[SS98] Schuldt, H.; Schek, H.-J.: *Transaktionelle Koordination von Subsystemen: Sphären zur Entkopplung von Transaktionseigenschaften*, In: Proceedings of the 9th Workshop „Transaktionen und ihre Anwendung“, Warnemünde, Germany, 1998

[TY98] Tygar, J.D.: *Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce*, In: Proceedings of the 24th VLDB Conference, New York, 1998