Diplomarbeit

# Ablaufkonfigurierung eines Kassensystems

vorgelegt von

Marlies Eschner
Kimbernstraße 15
22455 Hamburg
Matrikel-Nr. 400 928 0

März 1999

Erstbetreuer: Prof. Dr. Heinz Züllighoven

Zweitbetreuer: Dr. Volker Haarslev

Diese Diplomarbeit wurde am Fachbereich Informatik der Universität Hamburg zur teilweisen Erfüllung der Anforderungen zur Erlangung des Titels Diplom-Informatikerin eingereicht.

**Erklärung**

Hiermit versichere ich, diese Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Hamburg, den 03. März 1999

**Marlies Eschner**

Kimbernstraße 15
D-22455 Hamburg
Matrikel-Nr. 400 928 0

**Betreuung**

Prof. Dr. Heinz Züllighoven (Erstbetreuer)
Dr. Volker Haarslev(Zweitbetreuer)

**Prof. Dr. Heinz Züllighoven**

Fachbereich Informatik
Arbeitsbereich Softwaretechnik
Universität Hamburg
Vogt-Kölln-Straße 30
D-22527 Hamburg

**Dr. Volker Haarslev**

Fachbereich Informatik
Arbeitsbereich Kognitive Systeme
Universität Hamburg
Vogt-Kölln-Straße 30
D-22527 Hamburg

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Motivation

Diese Arbeit ist in der Firma c.a.r.u.s. Computer Service GmbH entstanden, in der ich seit 1995 als Systementwicklern tätig bin. Das Softwarehaus c.a.r.u.s entwickelt objektorientierte Anwendungssoftware. Diese wird für einzelne Auftraggeber individuell enstellt. Die Projekte sind hierbei im Allgemeinen Client/Server-Anwendungen unter Einbeziehung relationaler Datenbanken. Ein Schwerpunkt ist die Entwicklung von Kassen- und Warenwirtschaftssystemen für den Handel.

Diese Diplomarbeit beschäftigt sich mit dem Kassensystem. Es wurde ein Kassensystem entwickelt, dass sehr flexibel war, aber durch die Flexibilität wurde die Wartung sehr aufwendig. Die Festlegung der Steuerung der Kassenbedienvorgänge geschah durch eine händische Konfigurierung. Die Konfigurationsdaten wurden in einer Datei gespeichert und zur Laufzeit der Kasse eingelesen und ausgewertet. Zur Unterstützung der Konfigurierung sollte deshalb ein Werkzeug erstellt werden, das die Konfigurierung des Kassensystems erleichtert. Dies war zuerst der Auftrag für meine Diplomarbeit.

Bei der genaueren Analyse des Kassensystems stellte es sich heraus, dass es nicht ausreicht einen Konfigurator zu erstellen. Es war ein Redesign des Kassensystems notwendig. Aus diesem Grunde beschäftigt sich diese Arbeit nicht wie ursprünglich geplant mit der Konzeption eines Konfigurators, sondern mit der Analyse und dem Redesign der Kassensoftware.

## 1.1 Überblick über die Arbeit

Im zweiten Kapitel gebe ich eine kurze Begriffsklärung für die Begriffe Konfiguration und Konfigurierung.

Am Anfang des Kapitels 3 wird eine kurze Einführung in die Domäne Kassensysteme gegeben. Die Analyse des alten Kassensystems und Erstellung von Anforderungen für eine Konfigurierung erfolgt danach. In der Analyse wird aufgezeigt welche Ursachen die lange Einarbeitszeit hat. Es wird beschrieben, welches Wissen sich jemand aus welchen Quellen erarbeiten muß und mit welchen Mitteln die Konfigurationsdaten dargestellt werden. Am Ende des Kapitels stelle ich die Anforderungen für eine bessere Konfigurierung zusammen.

Der Lösungsansatz für die Anforderungen wird im Kapitel 4 erläutert. Die Grundlagen für die Lösung sind das Prinzip der Strukturähnlichkeit der fachlichen Begriffe und den Softwarekonstrukten durch die Vergegenständlichung der fachlichen Begriffe aus dem WAM-Ansatz und die Vergegenständlichung von Prozessen aus dem Konzept des Prozessmusters.

Ausdrucksmittel, um dies zu realisieren, sind zum einem statisch ein Kassen-Rahmenwerk, zum anderen dynamisch die Benutzung von Komponenten zum Konfigurieren.

Die technischen Ausführungen des Lösungsansatz werden in Kapitel 5 und 6 beschrieben.

Im fünften Kapitel geht es dabei zuerst um den Aufbau des Rahmenwerks. Es wird die Architektur des WAM-Schichtenmodells beschrieben. Außerdem wird der Aufbau der Klassen für die Bedienvorgänge dargestellt. Am Ende des Kapitels werden zwei Änderungsanforderungen und die dafür notwendigen Änderungen im Rahmenwerk erläutert.

Das sechste Kapitel gibt eine kurze Einführung in die Komponententechnologie. Es wird gezeigt, wie die Semantik einer Komponente dargestellt werden kann. Zum Schluß wird ein Einblick gegeben, wie die Konfigurierung mit Hilfe der Komponenten aussehen kann.

Im letztem Kapitel fasse ich die Ergebnisse der Diplomarbeit zusammen und gebe einen Ausblick auf weitere Themen, die sich bei der Thematik Konfigurierung ergeben.

# 2 Konfiguration, eine kurze Begriffsbestimmung

Bevor ich näher auf die Kassendomäne eingehen werde, gebe ich hier eine kurze Einführung in die Begriffe Konfiguration und Konfigurieren.

Zuerst eine Definition aus dem Meyer Lexikon

**Definition 1: Konfiguration**

> "1) [lat.], bildungssprachlich: bestimmte Art der Gestaltung.
>
> 2) Physik: die Gesamtheit der momentanen Orte der Teilchen eines Mehrteilchensystems. Bei Atomen gibt die Konfiguration an, welche Zustände einer Elektronenschale durch Elektronen besetzt sind.
>
> 3) Chemie: die räumliche Anordnung der Atome eines Moleküls."
>
> [Meyers Lexikon 98]

Konfiguration ist also eine Gestaltung oder räumliche Anordnung von Objekten. Übertragen auf eine Kassensystem, bedeutet dies die Anordnung der Softwareobjekte. Doch was ist Konfigurierung ?

Dazu ein Zitat aus dem Bereich der Kognitionswissenschaften, dem wissensbasiertem Konfigurieren. Dies ist eine Definition aus dem Projekt KONWERK.

**Definition 2: Konfigurierung**

> " Konfigurieren wird als Syntheseaufgabe verstanden, bei der Objekte aus einer Domäne zu einer Konfiguration zusammengefügt werden. Dabei sind gegeben:
>
> - Eine Spezifikation der Aufgabe (Konfigurationsziele), die insbesondere angibt, welche Anforderungen die zu erzeugende Konfiguration erfüllen soll
> - Eine Menge von Objekten der Anwendungsdomäne (Domänenobjekte) und deren Eigenschaften (Parameter).
> - Eine Menge von Relationen und Restriktionen zwischen den Objekten. Dabei sind für die Konfigurierung insbesondere die kompositionellen Beziehungen von Bedeutung.
> - Wissen über die Vorgehensweise bei der Konfigurierung (Kontrollwissen)."

Das Konfigurationsziel für den Kassenbereich ist ein Kassensystem für einen Kunden des Softwarehauses c.a.r.u.s. zu erstellen. Die Menge von Domänenobjekte sind die Geräte- und Softwarekomponenten. Eine Konfiguration ist dann ein konkretes zusammengestelltes Kassensystem oder auch Teile des Kassensystems. Weitergehend beschäftigt sich Wimmel in seiner Diplomarbeit [Wimmel 98] mit dem wissensbasiertem Konfigurieren von Kassensystemen. Im Kapitel 6 werde ich noch einmal auf den Konfigurator, den er erstellt hat, Bezug nehmen. Bevor ich jedoch darauf eingehe, wende ich mich dem Thema meiner Diplomarbeit zu, der Ablaufkonfiguration von Kassensystemen. Das nächste Kapitel beschäftigt sich mit der Analyse der alten Konfigurierungstechnik.

# 3 Analyse

Für die Anforderung ein Werkzeug zu schreiben, das die Konfigurierung einer Kasse unterstützt, ist es zunächst notwendig die Domäne PC-Kasse zu verstehen. Weiterhin muß die Struktur der bestehenden Kasse sowie die derzeitige Konfigurierungstechnik, untersucht werden. Dies wird in diesem Kapitel beschrieben. Es wird außerdem dargestellt, welche Probleme sich durch die derzeitige Konfigurierungstechnik ergeben und welche Ursachen dafür verantwortlich sind. Aus dieser Analyse ergeben sich dann die Anforderungen für eine zukünftige Konfigurierungstechnik.

## 3.1 Domäne PC-Kasse

PC-Kassen sind Registrierkassen, die als Basissystem einen Personalcomputer haben. Sie werden eingesetzt um Verkaufsvorgänge und Zahlungsvorgänge zu unterstützen. Die Einsatzbereiche können unterschiedlich sein. PC-Kassen werden z.B. in den folgenden Bereichen eingesetzt:

- Gastronomie: Großküche, Gaststätte, Restaurant
- Freizeitveranstaltungen, Kino, Theater
- Filialhandel, Lebensmittelmarkt, Buchhandel

Für diese Bereiche gibt es unterschiedliche Ausprägungen einer PC-Kasse. Zum einen können an ihr unterschiedliche Geräte angeschlossen sein, zum anderen gibt es unterschiedliche Abläufe beim Verkaufsvorgang.

Als Geräte können z.B. zur Eingabe Scanner, Tastatur, Touchscreen, Waage, Magnetkartenlesegeräte und zur Ausgabe Drucker, Zeilendisplay, Entwerter, Grafikbildschirm angeschlossen sein.

An einer Supermarktkasse gibt es z. B.:

- ein Zeilendisplay für den Kunden
- ein Zeilendisplay für den Kassierer
- eine Kassenschublade
- einen Magnetkartenleser
- einen Bondrucker
- eine Waage
- eine Tastatur mit Festtasten für Pfandflaschen

In einem Bekleidungsgeschäft gibt es dagegen

- ein Zeilendisplay für den Kunden
- einen Grafikbildschirm für den Kassierer
- eine Tastatur
- einen Magnetkartenleser
- eine Kassenschublade

Ein Kartenverkaufsautomat hingegen hat nur

- einen Magnetkartenleser
- einen Grafiktouchscreen

Die Kombinationsmöglichkeiten der Hardware einer PC-Kasse sind also recht vielfältig.

Genauso ist es mit den Abläufen bei der Kasse. Die Bezahlungsvorgänge können recht unterschiedlich sein, wie z.B.

- Bar
- Bar mit Fremdwährung
- Kreditkarte
  - hausinterne
  - externe
- EC-Scheck
- Bezahlung per Rechnung
- die ganze Summe wird mit einer Zahlungsart gezahlt
- die Summe wird mit mehreren Zahlungsarten bezahlt.

  (z.B. Euroscheck 400,- DM, Bar 22,50 DM)

Auch die Preisberechnung eines Artikels kann unterschiedlich sein, es gibt z.B. Rabatte beim Personalverkauf oder beim Gewerbeverkauf.

Für eine Grundbedienungsvorgang wie die Anmeldung eines Kassierers kann es verschiedene Möglichkeiten geben, z.B. :

- die Eingabe von einer Kennung und einem Paßwort
- einen Identifizierungsschlüssel
- eine Identifizierungsmagnetkarte.

Eine konkrete Kasse wird also aus wenigen Komponenten zusammengestellt, den Geräten und den Bedienungsvorgängen, doch es gibt eine Anzahl von Kombinationsmöglichkeiten. Aufgrund dieser Tatsache entstand die Idee bei c.a.r.u.s. die Komponenten objektorientiert abzubilden. Für eine konkrete Kasse soll dann die Konfigurierung mit einer Beschreibungsdatei geschehen, d.h. es wird in der Datei angegeben welche Geräte angeschlossen sind und welche Vorgänge in welcher Reihenfolge erlaubt sind. Diese Datei wird beim Start der Kasse eingelesen. Auf diese Weise sollte ein extrem flexibles Kassensystem entstehen, das an alle Kundenwünsche angepaßt werden kann.

Beim Arbeiten mit diesem Konzept hat sich jedoch ergeben, dass die Wartbarkeit der Konfigurationsdatei schlecht ist. Im folgenden wird genauer das Kassensystem, Eyecash, beschrieben, das nach diesem Konzept erstellt wurde. Es wird erläutert, wie es konfiguriert wird und welche Probleme sich durch diese Konfigurierung ergeben.

## 3.2 Beispiel zur Anschauung

Zum besseren Verständnis beschreibe ich ein kleines Szenario eines Bedienvorgangs.

Kassieren mit einem EC-Scheck ohne Scheckaufdruck

Die Ausgangssituation ist: Ein Kunde befindet sich in einem Supermarkt, er hat in einem Warenkorb die Artikel gesammelt, die er erwerben möchte. Er steht an der Kasse, seine Artikel sind jetzt schon in die Kasse eingegeben worden. Der Kassierer[1] drückt die Summentaste, auf dem Kundendisplay wird die Bonsumme von 156,34 DM angezeigt.

[1] Eigentlich müßte ich jetzt von Kassiererinnen sprechen, weil die meisten Benutzer einer Kasse Frauen sind. Doch der Einfachheit halber wähle ich die männliche Form

Der Kunde möchte die angezeigte Bonsumme mit einem EC-Scheck bezahlen. Der Kassierer drückt auf die Taste Scheck. Im Display erscheint der Text "156,34 DM Betrag bestätigen". Der Kassierer bestätigt mit der Returntaste den angezeigten Betrag. Der Kunde übergibt die Scheckkarte an den Kassierer. Dieser führt die Karte durch den Magnetkartenleser, die benötigten Kundendaten werden ausgelesen. Falls die Karte nicht korrekt ausgelesen wurde, müssen die Kartendaten per Hand eingegeben werden. Der Kunde stellt den Scheck auf den Betrag von 156,34 DM aus und unterschreibt den Scheck. Der Bon wird ausgedruckt und der Kassierer legt den Scheck in die geöffnete Kassenschublade und schließt sie wieder.

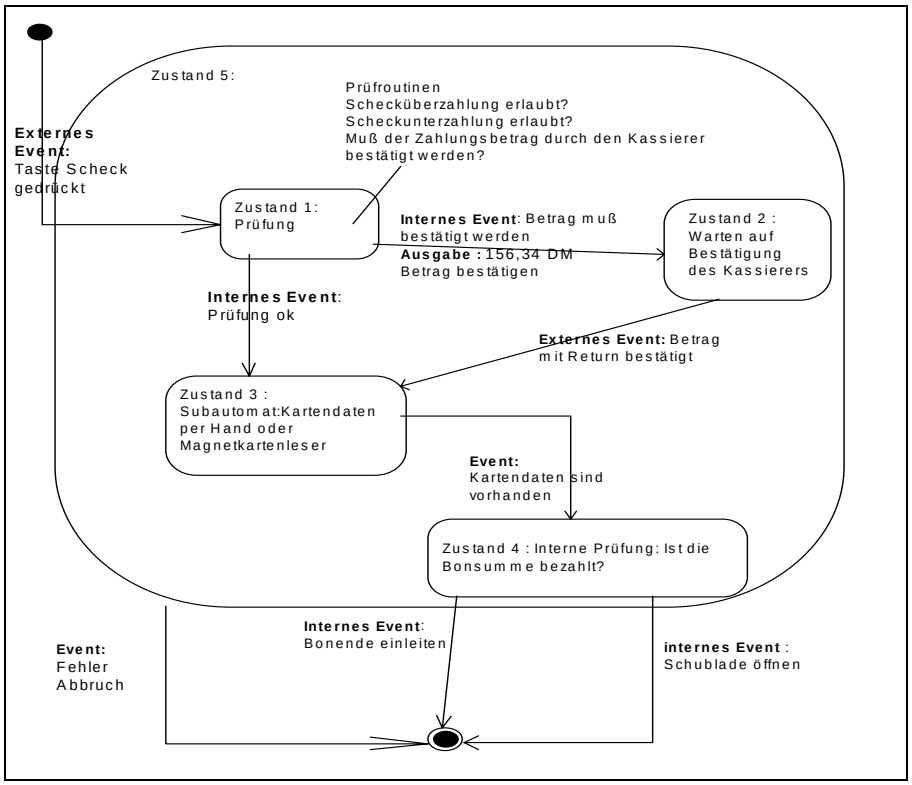


**Abbildung 3-1: Grafische Darstellung des Vorgangs "Kassieren mit einem EC-Scheck ohne Scheckaufdruck" Notation Statecharts [Harel 87]**

In dieser Grafik wird zwischen internen und externen Events unterschieden. Externe Events sind Events, die der Benutzer der Kasse, der Kassierer, durch eine Aktion an der Kasse erzeugt. Dies wäre z.B. das Drücken einer Taste oder das Schließen der Kassenschublade. Interne Events werden benutzt zur internen Steuerung des Kassensystems.

Dieser Teilvorgang so wie er in der Abbildung 3-1 abgebildet ist entspricht dem Teilvorgang in der Abbildung 3-7.

## 3.3 Kassensystem Eyecash

### 3.3.1 Probleme der Konfigurierung

Das Kassensystem Eyecash ist eine objektorientierte Applikation, das bedeutet unter anderem, dass die einzelnen Teilsysteme durch entsprechende Objekte gekapselt sind. Als große Teilsysteme gibt es die Eingabeobjekte, die Ausgabeobjekte, Actioncontroller und die Steuerung. Durch diesen Aufbau ist es möglich für eine konkrete Kasse einzelne Teile zu ändern ohne, dass andere Teilsysteme geändert werden müssen.

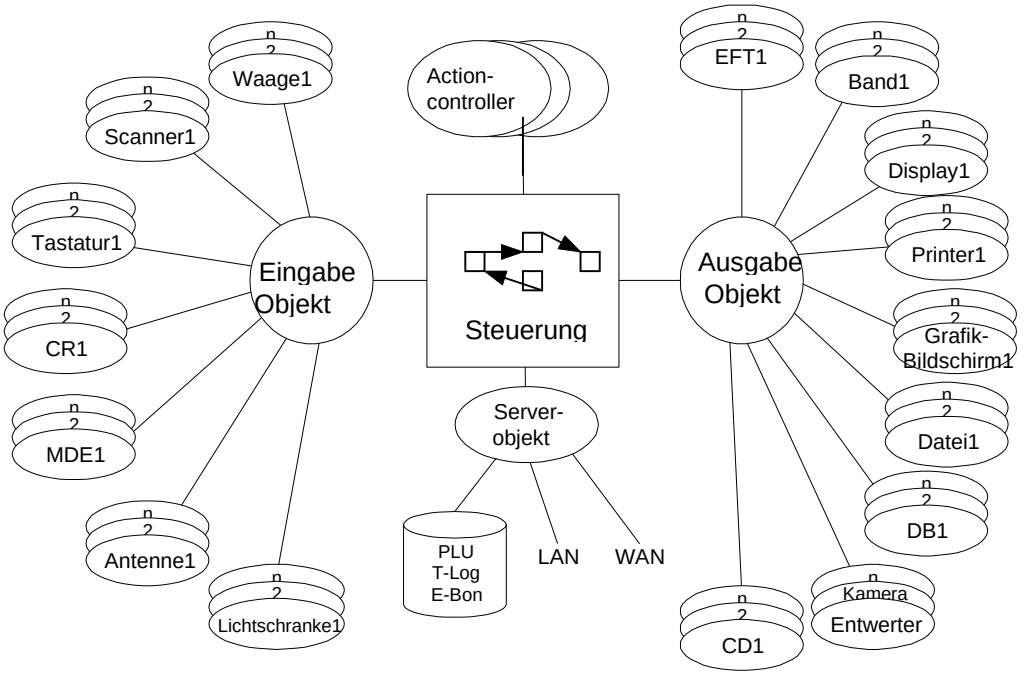


**Abbildung 3-1: Struktur der Eyecash-Kasse**

Die Flexibilität der Ablaufsteuerung der Kasse wird durch das Konzept eines Automaten erreicht.

Der Automat wird in der Konfigurationsdatei beschrieben, d.h. in der Datei steht, wieviel Zustände der Automat hat, welche Übergänge zwischen den Zuständen möglich sind und welche Aktionen bei den Zustandsübergängen ausgeführt werden sollen.

Diese Beschreibung wird beim Start des Kassenprogramms eingelesen und steuert dann die Kasse. Diese Konfigurationsdatei wird verändert, wenn ein anderer Ablauf für die Kasse gefordert wird.

Durch diese Art der Konfigurierung der Kasse sollte es möglich sein, schnell und flexibel die Abläufe der Kasse zu ändern, ohne den Sourcecode der Kasse selbst anzufassen. So sollte Entwicklungszeit gespart werden.

Bei den ersten Prototypen der Kasse war diese Art der Steuerung kein Problem. Die Anzahl der beteiligten Klassen und deren Funktionen war eher gering und noch gut überschaubar. Der Automat hatte noch wenige Zustände. Doch mit der wachsenden Anzahl von Klassen und somit dem komplexer werdenden Automaten wurde es immer schwieriger zu verstehen, wie ein Vorgang abläuft und welche Varianten möglich sind.

Die Erwartungen hinsichtlich der kurzen Veränderungszeiten sind nicht erfüllt worden. Der Einarbeitungsaufwand ist relativ hoch. Es dauert länger, einen Ablauf in der Konfigurationsdatei zu verändern oder neu zu erstellen, als ihn mit einer Programmiersprache, wie z.B. C++, auszuprogrammieren.

Der Hauptgrund dafür ist:

- Das Konzept, mit einen Automaten die ganze Kasse zu steuern, ist falsch. Die Steuerung des Ablaufs wurde dadurch programmiertechnisch auf ein sehr niedriges Niveau gebracht. Es wurde durch den Automaten eine Programmiersprache abgebildet, die keine strukturierten Programmierkonstrukte hat, wie Schleifen oder Verzweigungen.

Durch den gerade genannten Punkt ergibt sich auch der nächste Punkt. Weil keine struktuellen Sprachkonstrukte zu Verfügung standen, wurden die Events "mißbraucht".

- Events wurden für Sprachhilfskonstrukte und für fachliche Konstrukte genutzt. Bei dieser Art einen Ablauf zu programmieren, sind die fachlichen und die Sprachhilfskonstrukte schwer zu unterscheiden.

Durch die nicht objektorientierte Abbildung der Ablaufsteuerung wurde die Einarbeitung in einen Vorgang erschwert.

- Verteilte Informationen
  Die Informationen, die benötigt werden um einen Vorgang zu verstehen, befinden sich in der Konfigurationsdatei, im Sourcecode und in den dazu gehörigen Dokumentationen, mit den jeweils verschiedenen Darstellungsarten. Aus diesen Informationen muß ein Modell gebildet werden, welches aber so nicht gespeichert wird und damit beim nächsten Bearbeiten nicht mehr zu Verfügung steht.
- Eine kryptische Beschreibung des Automaten in der Konfigurationsdatei
  Alle Elemente des Automaten sind durch Zahlen verschlüsselt. Das sind die auslösenden Events für die Übergänge, die Zustände und die Aktionen.

Die fachlichen Informationen der Kasse wurden auf einen abstrakten Automaten abgebildet. Wenn der abstrakte Automat bearbeitet, d.h. verändert, erweitert oder Teile davon wiederverwendet werden sollen, muß der abstrakte Vorgang wieder, um ihn zu verstehen, auf einen fachlichen Vorgang abgebildet werden, dies kostet sehr viel Zeit.

Es wurden zwar, wie [Keil-Slawik 90] es beschreibt, externe Gedächtnisse (Artefakte) von Lernprozessen geschaffen, nämlich den Automaten und die Sourcen, doch in diesen Artefakten sind nicht alle notwendigen Informationen explizit vorhanden.

Wenn man sich die Definition des Artefakts endlicher Automat anschaut, wie sie z.B. Balzert in seinem Lehrbuch definiert [Balzert 96], erkennt man einen Punkt des Informationverlustes.

> "*Ein endlicher Automat oder Zustandsautomat ... besteht aus einer endlichen Anzahl von Zuständen. Der Zustand eines Systems beinhaltet* ***implizit*** *die Informationen, die sich aus den bisherigen Eingaben ergeben haben und die benötigt werden, um die Reaktion des Systems auf noch folgende Eingaben zu bestimmen."*

Das heißt für die Kassenzustandsautomaten, welche Information der Zustand fachlich für die Kasse hat, ist nicht explizit gespeichert worden. Der Zustand ist nur eine Nummer, sie bildet keine Semantik ab. Der Zustand hat keinen semantischen Namen und keinen Kontext.

Die implizite Information über den Zustand der Kasse muß wieder erarbeitet werden. Dieser Prozess ist sehr mühsam, weil die Verschlüsselung über mehrere Stufen geht und zudem noch frustrierend ist, weil die Informationen nach kurzer Zeit wieder verloren sind.

Ein Entwickler der oft die Kasse konfigurierte, drückte es so aus:

> " Nach einer Woche kann ich wieder von vorne anfangen, dass macht mir keinen Spaß."

Um diese Punkte zu begründen, werde ich zunächst die Grundkonstruktion der Kasse erklären.

### 3.3.2 MVC-Architektur

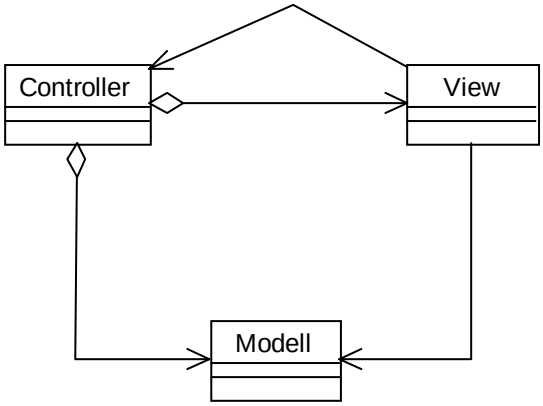


**Abbildung 3-2: MVC - Architektur** Notation nach UML

Die Kasse wurde nach der MVC (**M**odell, **V**iew, **C**ontroller) Architektur konstruiert. Es gibt Modelle, Controller und Views, die in einer bestimmten Beziehung stehen.

Ein Modell ist ein Klasse, welche die anwendungsfachlichen Daten enthält, die persistent gespeichert werden. In der Kasse ist dies z.B. der Bon. Er wird in der Klasse CMTransaktion abgebildet. Eine Transaktion besteht aus mehreren Transaktionsequenzen. Die Transaktionssequenzen sind z.B. Bonstart, Artikelregistrierung, Bonsumme, Bezahlung und Bonende. Der Bon, der dem Käufer beim Einkauf von z.B. Lebensmitteln gegeben wird, ist eine Sicht davon. Es gibt weiterhin Views (Sichten), die in der Kasse in Eingabe- und Ausgabeviews aufgeteilt werden. Die Views sind für die Eingabe und Darstellung von Daten des Modells zuständig. Das dritte Element in dieser Architektur ist der Controller. Der Controller ist für alle Aktionen zuständig, die unabhängig von der konkreten Darstellung sind. Die Controller besitzen das Modell und eins bis n Views. Die Controller, die die Vorgänge, wie z.B. den Bezahlungsvorgang steuern, werden in der Kasse Actioncontroller genannt.

Die Controller sind in MVC hierarchisch geordnet nach einer Baumstruktur. Es gibt einen Hauptcontroller in der Wurzel, der die Kontrolle über weitere Controller hat. Außerdem gibt es das Konzept eines aktuellen Controllers. Der aktuelle Controller ist derjenige, zudem der View gehört, auf dem sich der Mauszeiger befindet. Alle Events werden an den aktuellen Controller weitergeleitet. Wenn dieser das Event nicht verarbeiten kann, wird die Nachricht nach einer bestimmten Struktur weitergeleitet.

### 3.3.3 Architektur des Kassensystems

Es bestand die Vorstellung, eine Bausteinsammlung aus Kassenelementen zu haben, aus der dann durch "einfaches Zusammenstecken" eine konkrete Kasse wird. Durch dieses Bausteinprinzip sollte Entwicklungs- und Wartungszeit gespart werden.

Realisiert wurde dies, indem das Modell der Kasse objektorientiert in C++ abgebildet wurde. Diese Sourcen werden kompiliert und beim Starten der Applikation mit einer Konfigurationsdatei parametrisiert. In der Konfigurationsdatei wird festgelegt, welche Objekte erzeugt werden können, zum Teil, wie die Objekte zusammenarbeiten und welches Objekt aufgrund eines Ereignisses eine Funktion ausführt. Diese Konfigurationsdatei wird verändert, wenn ein anderer Ablauf für die Kasse gefordert wird.

In welche Abstraktionsschichten das Kassensystem aufgeteilt ist, zeigt die Darstellung und Beschreibung aus der Diplomarbeit von Schwanke und Albrecht.(Abbildung 3-3) Diese Diplomarbeit war die Grundlage für das heutige Kassensystem Eyecash.

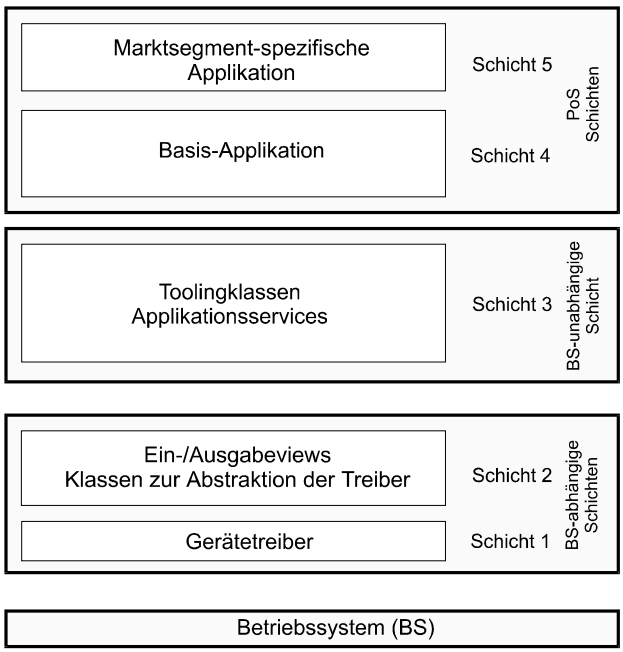


**Abbildung 3-3: Die Softwarestruktur von EYECASH als Schichtenmodell**

[Schwanke, Albrecht 95]

**"Schicht 1 - Gerätetreiber**

Schicht 1 setzt direkt auf dem Betriebssystem auf. Sie enthält die Gerätetreiber, die zur Kommunikation mit den Ein- / Ausgabegeräten, sowie zur Ansteuerung spezieller Hardware (z.B. Scanner, Magnetkartenleser, Kamera) erforderlich sind.

**Schicht 2 - Abstraktionsklassen**

Diese Schicht kapselt die Gerätetreiber und das Betriebssystem. Sie stellt den darüberliegenden Schichten eine plattformunabhängige und einheitliche Schnittstelle zur Verfügung, über die die Objekte der Schichten 3-5 mit den Geräten ohne Kenntnis der speziellen Hardware- oder Treibereigenschaften kommunizieren können.

**Schicht 3 - Toolingklassen**

Die Klassen in Schicht 3 bauen auf der abstrakten Schnittstelle auf, die durch Schicht 2 zur Verfügung gestellt wird. Sie sind somit unabhängig von spezieller Hardware oder bestimmten Eigenschaften eines Betriebssystems. Schicht 3 stellt der Kassenapplikation allgemeine Dienste und generelle Funktionalitäten zur Verfügung. Darunter fallen Mechanismen wie z.B.: Transaktionssicherung, Logging oder universelle Ein- und Ausgabe.

**Schicht 4 - Applikationsbasis**

Diese Schicht stellt die Basis für eine vollständige Kassenapplikation zur Verfügung. Es werden die Funktionen und Klassen implementiert, die unabhängig von einer bestimmten Marktausrichtung in jedem Kassensystem zur Verfügung gestellt werden können. Mit den Klassen dieser Schicht erhält man bereits eine vollständig nutzbare Kasse.

**Schicht 5 - Marktspezifische Applikation**

Schicht 5 wird durch die Klassen gebildet, die erforderlich sind, um die Kasse an die Bedürfnisse eines speziellen Marktsegmentes anzupassen."

[Schwanke, Albrecht 95]

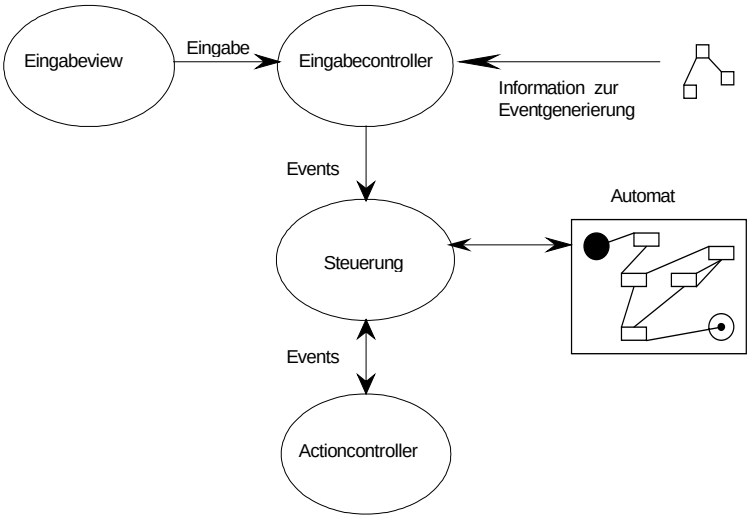


**Abbildung 3-4: Zusammenspiel bei einer Eingabe**

Ein Beispiel, wie ein Vorgang in der Kasse ausgeführt wird:

Die Eingabeviews aus der Schicht 2 und die Eingabecontroller aus der Schicht 3 verarbeiten die Eingabeinformationen zu eindeutigen Kassenereignissen.

Es wurde z.B. die Taste mit der Beschriftung "Scheck" gedrückt. Der Eingabeview erhält das Event, dass die Taste Nr. 3187 gedrückt wurde, dies ordnet er der Tastengruppe 22 zu und übergibt es dem Eingabecontroller. Dort wird das einmalige Drücken einer Taste der Gruppe 22 als Kassenereignis "Zahlung mit Scheck"( enEV_PAYMENTSCHECK) interpretiert. Der Eingabecontroller sendet das Event “enEV_PAYMENTSCHECK“ zu dem Kassenautomat. Der Kassenautomat hat die Information welche Aktion ausgeführt werden soll. In dem Beispiel wäre es die Funktion "Scheckzahlung einleiten" der Klasse CCPaymentScheck. Wenn die Aktion beendet ist, geht die Kontrolle wieder an den Eingabecontroller zurück, der wiederum auf die Eingaben der Eingabeviews wartet.

### 3.3.4 Kritik an der Konstruktion

Wenn man die zugrunde gelegte Architektur und den Aufbau der Kasse vergleicht, kann man erkennen, dass die Kasse nicht durchgängig nach der MVC - Architektur gebaut wurde. Bei MVC ist ein Controller für einen fachlichen Vorgang zuständig. In der Kasse gilt dies zwar auch, aber nur auf der logischen Ebene, technisch ist er nur durch die abstrakte Steuerung des Automaten verbunden. Diese Verbindung muß immer wieder hergestellt werden und zwar künstlich, weil diese Verbindung nicht in die objektorientierte Schicht der Kasse gezogen wurde. Dies geschieht in dem großen Automaten, der diese Steuerung übernimmt. Damit steuert nicht die objektorientierte Konstruktion die Kasse, sondern ein Automat der die Klassen als Funktionsbibliothek benutzt.

Allein in der Beschreibung der Konfigurationsdatei, also in dem großen Automaten, wird festgelegt, welche Funktionen in welchem Zustand der Kasse aufgerufen werden können.

Durch diese Art der Konfigurierung kann in jedem Zustand der Kasse jede Funktion der Actioncontroller aufgerufen werden. Das bedeutet, es kann ein Spaghetticode erzeugt werden. Die Übergänge zwischen den Zuständen können, so wie sie hier benutzt wird, als goto's bezeichnen. Wenn man die folgenden Punkte von Dijkstra betrachtet, wird deutlich, dass die Steuerung der Kasse durch diese Art der Programmierung nicht einer strukturierten Programmierung entspricht und damit schlecht wartbar ist.

"Das in älteren Programmiersprachen noch enthaltene Steuerkonstrukt "Sprung" bzw. "goto-Anweisung“ verstößt gegen die oben aufgestellten Forderungen (nach Lokalität und Linearität) :

- Eine Sprunganweisung verwischt völlig den semantischen Unterschied zwischen einer Auswahl und einer Wiederholung. ...
- Lokalität und Linearität sind nicht garantiert, da mit Sprüngen an beliebige Stellen des Algorithmus und damit auch in anderer Kontrollstrukturen hinein gesprungen werden kann.
- Die Terminierung kann nicht oder nur mit großem Aufwand sichergestellt werden, da bei einem Sprung nicht klar ist, ob er dazu dient, eine Anweisung zu wiederholen oder ob es sich um Sprünge zur Realisierung von Auswahlbedingungen handelt.
- Das strukturierte Denken in Sequenz, Auswahl und Wiederholung wird nicht unterstützt, da durch einen Sprung jederzeit die "Notbremse gezogen werden kann" d.h. wenn man einen Algorithmus formuliert hat und am Ende merkt, dass das Problem einen Rücksprung an den Anfang erfordert, dann wird man nicht gezwungen, den Algorithmus mit einer Wiederholung neu zu strukturieren.
- Syntax und Semantik einer Sprunganweisung sind nicht selbst erklärend.
- Die Fehleranfälligkeit steigt bei der Verwendung von Sprunganweisungen. "

**[Dijkstra 69/72]**

Es gibt in dem Automaten keine strukturierten Steuerkonstrukte, sondern nur die bedingten oder unbedingten Sprünge in einen anderen Zustand.

Wenn die Kasse besser und schneller konfigurierbar sein soll, muß zuerst die Steuerung der Kasse durch eine Programmiersprache abgebildet werden, die strukturierte Sprachkonstrukte hat.

Bevor ich auf die anderen Punkte eingehe werde ich zum besseren Verständnis die Beschreibung des Automaten und seine Funktionen erklären.

### 3.3.5 Kryptische Beschreibung des Kassenautomaten

In der Konfigurationsdatei stehen verschiedene Abschnitte, sogenannte Kapitel. Die meisten Kapitel sind für die Parametrisierung der Kasse bestimmt, wie z.B. die Zuordnung eines Actioncontrollers zu einer Nummer. Die Beschreibung des Steuerungsautomaten steht in einem eigenem Abschnitt.

Der Automat hat n verschiedene Zustände. In einer bestehenden Kasse sind es zur Zeit 351 Zustände. Alle Elemente des Zustandsautomaten sind durch Nummern verschlüsselt. Der Automat hat als Eingabealphabet Events. Die Ausgabe des Automaten kann entweder leer, ein Event oder der Aufruf einer Methode eines Actioncontrollers sein.

Der Automat hat das Konzept der Subautomaten, um den Überblick über die Steuerung des Kassensystems nicht zu verlieren. Jeder Subautomaten hat seinen eigenen Nummernkreis. Ein Subautomat ist für einen fachlichen Teilvorgang gedacht. Innerhalb eines Subautomaten werden nur zwei Actioncontroller aufgerufen, der Actioncontroller, der für diesen Vorgang vorgesehen ist und ein Actioncontroller, der die allgemeinen Fehler behandelt.

Wie das Konzept des Subautomaten technisch umgesetzt wird, beschreibe ich im Kapitel 3.3.7.

```
;------------------------------------------------------------  ------------------------------------------------------------
; Status           : 700
; Bedeutung       : Scheckzahlung einleiten
; Erwartete Events:
; enEV_FNNOTALLOWED                  = -6    // Funktion nicht erlaubt
; enEV_INPUTSEQERR                   = -7    // Fehlerhafte Eingabefolge
; enEV_DIRECTNOTALLOWED              = -8    // Direct Payment nicht erlaubt
; enEV_UNDER_TENDER_NOTALWD          = -2182 // Keine Unterzahlung erlaubt
; enEV_OVER_TENDER_NOTALWD           = -2183 // Keine Ueberbezahlung erlaubt
; enEV_ONLY_DIREKT_TENDER_ALWD       = -2184 // Nur Direktzahlung erlaubt.
; enEV_NOEVENT                       = 0     // ok
; enEV_DIRECTDIALOG                  = 2172  // Bestaetigung Direktzahlung anfordern
; enEV_PAYMENTSCHECK                 = 2180  // Scheckzahlung einleiten
;------------------------------------------------------------  ------------------------------------------------------------
state700=(-6,0,724,4)(-7,0,724,4)(-8,0,724,4)   (-2182,0,724,4)(-2183,0,724,4)(-2184,0,724,4)
(0,0,701,-53)(2172,0,703,0)(2180,0,700,15)
```

**Abbildung 3-5: Auszug aus der Konfigurationsdatei: Die Beschreibung eines Zustandes mit seinen Übergängen.**

Abbildung 3-5 ist der Zustand 700 aus der grafischen Darstellung in Abbildung 3-7, die vollständige textuelle Beschreibung befindet sich im Anhang A.

Die Zeilen, die mit einem Semikolon beginnen, sind Erläuterungen. Bei diesem Zustand ist es eine Erklärung welche Events den Nummern entsprechen und was es für den aktuellen Vorgang bedeutet.

Die Zustände des Automaten sind numeriert. Es bestehen Ziffernkreise für einzelne Vorgänge, wie z.B. die Zustände 1-50 sind für die Anmeldung reserviert.

Jedem Zustand sind eins bis n Zahlentupel für die Zustandsübergänge zugeordnet. Die Ziffern in dem Zahlentupel stehen für verschiedene Gruppen.

Erklärung der Notation in EBNF

| | |
|---|---|
| Zustand | ::= state<Nummer> = {<Zustandsübergang>} |
| Zustandsübergang | ::=(< EventNr >, <Sprungart>, <Follow>, <Action>) |
| EventNr | ::= [-]<Nummer> |
| Sprungart | ::= 0 \|1 \|2 |
| Follow | ::= <Nummer> |
| Action | ::= [-]<Nummer> |
| Nummer | ::= {num}[1] |
| num | ::= 0 \|1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 |

**EventNr**

EventNr steht für ein Event, das die Transition auslöst. Die Eventnummer muß innerhalb eines Zustands eindeutig sein.

**Sprungart**

Die Statusmaschine hat mehrere Subautomaten zur besseren Übersicht. Sprungart ist eine Nummer, mit der gesteuert wird, ob ein Zustandsübergang als normaler Sprung, als Sprung in einen Subautomaten oder als Rücksprung aus einem Subautomaten interpretiert wird.

Sprungart = 0 -> Normaler Sprung zum nächsten Status

Sprungart = 1 -> Sprung in einen Subautomaten.

Sprungart = 2 -> Rücksprung aus einem Subautomaten.

Bei einem Sprung in einen Subautomaten merkt sich der Zustandsautomat den Startzustand, beim Rücksprung erfolgt der Übergang in den gemerkten Zustand.

**Follow**

Follow ist die Nummer des Folgezustands. Die Statusmaschine wechselt in diesen Zustand, sobald das Event dieser Transition eintrifft. Wenn aus einer Subroutine gesprungen werden soll, wird Follow ignoriert. Deshalb wird Follow auf Null gesetzt.

**Action**

Action ist die Nummer eines Actioncontrollers. Actioncontroller ist eine Klasse, die Aktionen in einem anwendungsfachlichen Vorgang ausführt, z.B. die Klasse CCPaymentCash, sie steuert die Barzahlung. Die Zuordnung eines Controllers zu einer Nummer ist im Kapitel [ActionObjects] festgelegt. Sobald das Event dieser Transition eintrifft, wird es an diesen Actioncontroller weitergeleitet.

**Verschlüsselung der Elemente durch Nummern**

Alle Elemente des Automaten sind durch Nummern verschlüsselt. Der Konfigurierer muß im Kopf haben, welche Eventnummer welchem Event entspricht, welche Actioncontrollernummer welcher Actioncontrollerklasse entspricht, welche semantische Bedeutung der Zustand mit der Nummer xy hat. Weiterhin muß er wissen, welche Nummernkreise für welche Vorgänge im System reserviert sind. >>Bei Nummern besteht, die beim raschen Überfliegen über die Beschreibung nicht auffallen und somit.

Außerdem ist durch die Verschlüsselung kein rasches Überfliegen der Beschreibung nach einem gesuchten Begriff möglich, weil Zahlenfolgen nicht wie Wörter durch Menschen schnell einer Bedeutung zugeordnet werden. Somit entsteht die Gefahr von Zahlendrehern, weil sie als Fehlerquelle nicht schnell aufzufinden sind.

Dies ist die erste Art der Verschlüsselung von Informationen. Eine zweite Verschlüsselung der Informationen ist die Art wie Events benutzt werden. Dadurch, dass die Methode nur indirekt im Steuerungsautomaten durch das Event benannt wird, ist es nicht immer zu erkennen, welche Aktion ausgelöst wird.

Beispiel:

enEV_TOGGLESCHECKPAYMENT = 2186 // Schaltet von Karten auf Handeingabe um

Dieses Event beendet den Vorgang, Daten einer Scheckkarte mit den Magnetkartenlesegerät einzulesen. Es wird der Vorgang, Kartendaten per Tastatur einzugeben, eingeleitet.

### 3.3.6 Verteilte Informationen für die Vorgänge

Die Beschreibung in der Konfigurationsdatei ist die erste Hürde zum Verstehen eines Vorgangs. Um einen Ablauf verstehen zu können, muß man gleichzeitig wissen, welche Events ein Actioncontroller zurückgibt, nachdem er durch den Automaten aufgerufen wurde.

Um verstehen zu können wie der Vorgang weiterlaufen kann, muß man wissen, welche Ereignisse der Actioncontroller zurückgeben kann. Entweder befindet sich die Information in der Dokumentation des Actioncontrollers oder es muß der Sourcecode des Actioncontrollers durchsucht werden.

Jeder Actioncontroller besitzt die Methode Run. Diese Methode wird von dem Automaten mit dem Parameter Event aufgerufen. In der Methode Run wird das Event einer Behandlung zugeordnet. Wenn die Handlung nicht komplex ist, geschieht diese Behandlung innerhalb der Run-Methode, ansonsten in einer eigenen Methode, die von dort aufgerufen wird. Die Methode Run liefert ein Ereignis an die Statusmaschine zurück, welches eine neue Transaktion auslöst, falls es einen Übergang für dieses Ereignis gibt.

Für einen Subautomaten ist in der jetzigen Ausprägung des Kassensystems ein Actioncontroller zuständig. Ein Vorgang kann aus mehreren Subautomaten bestehen. Welche Wahlmöglichkeiten es für den Vorgang gibt, wird bis jetzt noch nicht dokumentiert. Die Informationen müssen aus den einzelnen Sourcen der Actioncontroller heraus gearbeitet werden.

In der Beschreibungsdatei ist die Reihenfolge von Aktionen schwer erkennbar. Um sich da schneller einen Überblick zu verschaffen, hilft die grafische Dokumentation des Automaten die mit Rational Rose© erstellt wurde, die ich im nächsten Abschnitt mit ihren Vor- und Nachteilen erläutern werde.

#### 3.3.6.1 Dokumentation grafische Darstellung des Automaten

Um einen besseren Überblick über die Abläufe zu haben, gibt es für den Automaten eine Dokumentation in Form einer grafischen Darstellung des Automaten.

Der Zustandsautomat wird in Rational Rose © mit Hilfe von Statecharts beschrieben. Diese Dokumentation wurde erstellt, um sich schneller einen Überblick über einen Ablauf zu verschaffen.

Die einzelnen Subautomaten werden in einzelnen Statecharts beschrieben.

Folgende Konventionen wurden benutzt:

| grafische Notation | Bedeutung | Beschreibungsdatei |
|---|---|---|
| Event 5^Event 8 | Event 5 erzeugt Event 8 im nächsten Zustand | (5,0,Follow,- 8) |
| Event 5/ Actioncontroller1 | Actioncontroller15 wird aktiviert mit dem Event 5 | (5,0,Follow,15) |
| Event 5/ wiederholen | Event 5 erzeugt Event 5 im nächsten Zustand | (5,0,Follow,- 5) |
| Event 5/ 0 | Event 5 löst keine Aktion aus, es erfolgt aber ein Zustandsübergang | (5,0,Follow,0) |

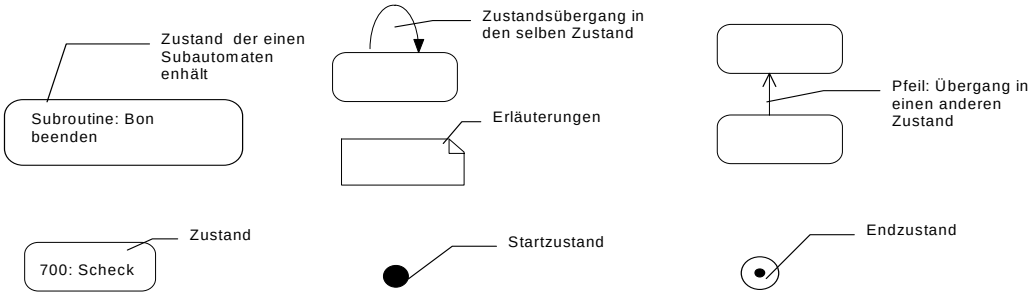


**Abbildung 3-6: Legende für die Dokumentation**

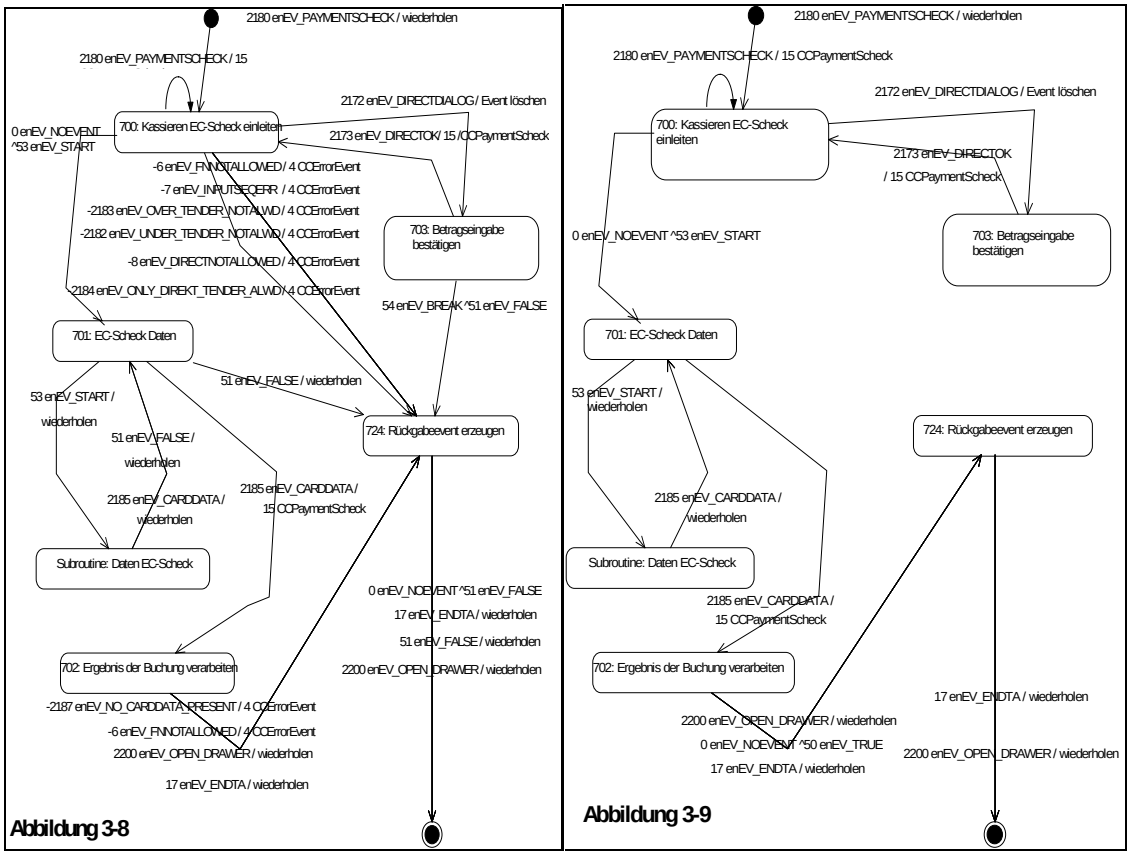


**Abbildung 3-7: Dokumentation zur Beschreibung des Kassenautomaten „Kassieren mit einem EC-Scheck ohne Scheckdruck“ mit Fehlermeldungen**

**Abbildung 3-8: grafische Dokumentation ohne Fehlermeldungen**

Die Dokumentation in Form einer grafischen Beschreibung verschafft einen geringfügig besseren Überblick. Es ist erkennbar, welche Vorgängerzustände und Nachfolgerzustände ein Zustand hat. Dies ist nützlich bei der Suche nach dem Erzeuger des Events.

Wenn man die Abbildung 3-7 und Abbildung 3-8 vergleicht, erkennt man einen Grund dafür, warum es trotzdem noch nicht viel übersichtlicher ist. Die vielen Fehlerevents blähen den Automaten auf. Der normale Vorgang ist nicht auf einen Blick zu erkennen. Ein weiterer Grund ist die niedrige Programmierebene, es existieren viele Events nur, weil programmiersprachlich kein anderes Mittel vorhanden ist. In der Beschreibung des Automaten existieren eine Menge von Events, die aber logisch auf verschiedene Ebenen gehören. Im nächsten Abschnitt werde ich die Ebene betrachten, in der Events benutzt werden, um höhere Sprachkonzepte zu erhalten.

### 3.3.7 Sprachhilfskonstrukte erschweren die Lesbarkeit eines Vorgangs.

Der Automat arbeitet auf drei verschiedenen Ebenen. Auf der ersten steuert er, welche Aktionen durch externe Events angestoßen werden, auf der zweiten legt er die Aufrufreihenfolge der Funktionen fest. Als drittes gibt es Hilfskonstrukte, um programmiersprachlich mehr leisten zu können.

Es wurden in den Automaten Hilfskonstrukte eingebaut um mehr Ausdrucksmöglichkeiten zu haben. Das sind Subautomaten und Sequenzen von Methoden

**Subautomaten**

Um Teile der Beschreibung wieder zu verwenden und um sich in dem Automaten besser orientieren zu können, wurden sogenannte Subautomaten eingeführt. Die Zustände wurden numeriert und für einzelne Vorgänge wurden Nummernkreise vergeben.

Beispiele :

| | |
|---|---|
| Anmelden/Abmelden/Pause: | 1 - 50 |
| Preisabfrage: | 100 - 150 |
| Standardverkauf: | 150 - 200 |
| Gewerbeverkauf: | 200 - 250 |
| Personalverkauf: | 250 - 300 |
| Bon erfassen: | 300 - 350 |
| Kassieren EC-Scheck ohne Scheckdruck: | 700 - 725 |

In diese Subautomaten wird durch die Angabe eines Flags hinein gesprungen. Siehe die Erläuterung zur Abbildung 3-5.

Das Springen in den Subautomaten ist zur Orientierung als Konzept eine gute Idee, doch bei der Realisierung führt es zu einem Spaghetticode. Es kann jeder Zustand des Subautomaten angesprungen werden und es kann aus jedem Zustand zurückgesprungen werden. Beim Rücksprung braucht nicht angegeben zu werden, wohin zurückgesprungen werden soll, weil automatisch zu dem Zustand gesprungen wird, aus dem der Sprung in die Subroutine erfolgte.

Dies erhöht die Einarbeitungszeit in den Vorgang.

**Sequenzen von Methoden**

Die Grundidee des Automaten war zuerst ein ereignisgetriebener Automat. Er reagiert auf externe Ereignisse, indem er Methoden eines Actioncontrollers aufruft.

Aus verschiedenen Gründen wurde dieses Konzept aufgebrochen. Der Automat erzeugt auch als Ausgabe Events, diese nenne ich interne oder Steuerungs-Events.

Zwei Gründe dafür sind:

- Es soll nach einem externen Event nicht eine Methode, sondern eine Folge von Methoden aufgerufen werden.

In einer Programmiersprache, wie z.B. C++, würde die Anforderung, dass zwei Aktionen direkt hintereinander ausgeführt werden sollen, wie z.B. Scheckausdruck und Schublade öffnen, dadurch erfüllt, dass sie in einer Sequenz geschrieben werden.

Programmbeispiel

```
ScheckDrucker->Drucken(poECScheck);
KassenSchublade->Öffnen();
```

Dies ist übersichtlich und verständlich.

- Der Aufruf einer Methode erfolgt indirekt durch ein Event. Deshalb muß genau dieses Event vorhanden sein. Wenn jedoch ein anderes Event eintrifft, muß dieses Event auf das benötigte abgebildet werden.

Das Konzept des Steuerungsautomaten sieht vor, dass auf Ereignisse reagiert wird und dass bei den Transaktionen Aktivitäten geschehen. Um jedoch Sequenzen von Aktivitäten zu ermöglichen, wurde das Prinzip durchbrochen. Der Automat erzeugt ein Ereignis, auf das er dann selbst reagiert. Dies wird benötigt, um Teilvorgänge miteinander zu verbinden. Weil die Methoden der Controller nicht direkt aufgerufen werden können, muß genau das Ereignis generiert werden, das der Actioncontroller in seiner Run-Methode verarbeiten kann.

Die Stelle Action in dem Beschreibungstupel wird benutzt, um Ereignisse zu erzeugen.

state# =(Event, Sprungart, Follow, Action)

Ist die Zahl in der Konfigurationsdatei an der Stelle Action negativ, so wird ein Event erzeugt.

Die negative Zahl wird in eine entsprechende positive Zahl verwandelt (aus -4 wird 4 ) und als Event im nächsten Zustand ausgewertet. Durch dieses Eventmapping ist es möglich verschiedene Teilvorgänge direkt nacheinander auszuführen. Das Rückgabeevent eines Vorgangs wird auf ein anderes Event abgebildet.

Zum Beispiel soll nach jeder Scheckzahlung die Kassenschublade geöffnet werden, damit der Scheck in die Kassenschublade gelegt werden kann.

Die Methode des Actioncontroller CCPaymentScheck, Bezahlung mit Scheck, liefert bei vollständiger Zahlung das Event "enEV_NOEVENT " zurück.

***State250 =(0,0,250,-2200), (2200,0,251,24)***

enEV_NOEVENT erzeugt hier das Event -2200, welches dann das Event 2200,"Schublade öffnen", auslöst, das dem Actionkontroller CCOpenCashDrawer(24) übergeben wird.

Dadurch wird direkt nach dem Bezahlvorgang die Schublade geöffnet, damit der Scheck in die Schublade gelegt werden kann.

Zusammenfassend läßt sich feststellen, dass es vier Arten von Events gibt.

1. Es werden Events, die von der Benutzereingabe kommen, verarbeitet, wie z.B. das Drücken der Taste Scheckzahlung.
2. Der Automat erzeugt ein Event, um Sequenzen von Aktionen auszuführen.
3. Die Actioncontroller liefern ein Event zurück, mit dem eine neue Aktivität aufgerufen wird.
4. Die Events werden wiederholt, damit sie in den Subautomaten verwendet werden können.

Die erste Form sind "normale" Events einer Event gesteuerten Applikation. Es kommen Ereignisse von dem Benutzer. Bei der zweiten und dritten Art werden Events benutzt, um Sequenzen von Aktionen auszuführen, für die keine anderen sprachlichen Mittel zur Verfügung stehen. Die vierte Art wird benötigt, um das Konzept der Subautomaten durchzuführen. Diese Art, einen Vorgang zu programmieren, erschwert die Übersicht über einen Ablauf. Bei jedem Event muß überprüft werden, um welche Art es sich handelt.

Dies zeigt auch, dass die Events nur so benutzt werden, weil der Automat als Konstruktion nicht ausreicht, eine Applikation zu steuern. Wenn eine strukturierte Programmiersprache benutzt worden wäre, wären nur die Events der ersten Kategorie vorhanden. Die zweite Art würde durch Blöcke abgebildet. Bei der dritten Art würde ein Auswahlkonstrukt eingesetzt werden, um die richtige Funktionen aufzurufen. Die vierte würde dem Aufruf eines Unterprogramms entsprechen.

## 3.4 Auf welcher Stufe Konfigurierung?

In dem letzten Kapitel habe ich analysiert wie die Kasse konfiguriert wurde. Das Ergebnis war, dass die verwendete Konfigurierungstechnik nur zur einem Spaghetticode geführt hat. Dies lag auch an der Granularität der Konfigurierung. Die Wiederverwendung von Vorgängen wurde runter gebrochen auf die Wiederverwendung von einzelnen Funktionen und nicht auf die Wiederverwendung von Objekten, also eine funktionale Sicht der Programmierung.

Eine objektorientierte Sicht wäre eine Wiederverwendung von Teilvorgängen wie z.B. der Teilvorgang Artikelregistrierung. Dies lag auch daran, dass für die einzelnen Teilvorgänge keine Klassen erstellt wurden. Nach meinem Ermessen sind aber Klassen nötig, um das Domänenwissen über die Teilvorgänge darin abzubilden. Ein zweiter Vorteil ist, dass der Teilvorgang dann in einer objektorientierte Sprache abgebildet wird, es ist damit gesichert, dass strukturierte Programmierkonstrukte zu Verfügung stehen. Außerdem steht dieser Teilvorgang als ein softwaretechnischer Gegenstand zur Verfügung und kann als ein solcher mit seinen Metadaten verwaltet und wiederverwendet werden.

Meine These für die Konfigurierung eines Ablaufs ist, dass die kleinsten wiederverwendeten Teile so groß sein sollten, dass sie in sich abgeschlossene Tätigkeiten sind. Dies sind in der Kassendomäne, die Teilvorgänge wie Artikelregistrierung oder eine Bezahlung mit Kreditkarte. Ein Teilvorgang ist unteilbar von einer Person durchzuführen. Diese These begründe ich damit, dass es gerade diese Teile sind, aus denen im Sprachgebrauch ein Vorgang sich in Teilvorgängen aufgliedert. Ein Standardverkauf besteht aus genau solchen Teilvorgängen, wie ich es z.B. in dem Szenario Standardverkauf im Kapitel 4.1 aufgeführt habe. Die Kassiererin registriert einen Artikel, sie storniert einen Artikel, sie führt eine Preisüberschreibung durch. Dies ist die benötigte Ebene der Strukturierung der Vorgangssteuerung in einem Verkaufsvorgang. Dies entspricht auch dem Vorgehen wie Workflow-Management-Systeme (WFMS) erstellt werden. Es werden als kleinste Einheiten eine Tätigkeit eines Mitarbeiters benutzt, um einen Workflow zu erstellen. Auf diese Weise bleibt der Vorgang übersichtlich und intellektuell faßbar. Die einzelnen Tätigkeiten können dann für sich genommen noch einmal verändert werden, das muß aber in einem separaten Konfigurierungs-Prozess erfolgen.

Ein gesamter Bedienvorgang kann von verschiedenen Personen durchgeführt werden. Ein Vorgang wie der Standardverkauf kann z.B. von einer anderen Person unterbrochen werden, wenn eine Stornierung durchgeführt werden muß. Dafür muß sich dann z.B. der Filialleiter an der Kasse anmelden.

Aus mehreren Teilvorgänge kann wiederum eine größerer Bedienvorgang zusammensetzt werden, so dass auf dieser Weise eine Konfigurierung möglich ist.

## 3.5 Anforderungen

Im Eingang des Kapitels habe ich die Punkte aufgezählt, die meiner Meinung nach zu der schlechten Wartbarkeit der Kassenkonfiguration führen. Nach der genaueren Analyse der Details läßt sich sagen, dass als erstes eine strukturierte Programmiersprache zur Konfigurierung verwendet werden muß, so dass kein Spaghetticode mehr entstehen kann. Dadurch würden auch die Notwendigkeit entfallen, Events als Programmierhilfskonstrukte zu verwenden.

Überdies sollen die Teilbedienvorgänge als eigenständige Klassen vorhanden sein. Die Informationen über einen Ablauf sollen alle in diesen Klassen vorhanden sein, so dass sie schnell wieder zur Verfügung stehen, damit die Einarbeitungszeit in einen Teilvorgang kurz ist.

Als drittes muß die Steuerung der einzelnen Teilvorgänge durch eine Maschine unterstützt werden, deren Niveau höher liegt als bei der alten Steuerung.

Es muß möglich sein die Informationen, die der Konfigurierer für sein Modell des Teilvorgangs braucht, zu speichern, so dass er alle Informationen vor sich liegen hat, um sich schnell wieder in den Vorgang einzuarbeiten.

Keil-Slawik beschreibt es so:

" Nur wenn es möglich ist, das Geschaffene aus dem Prozess des Schaffens herauszulösen, beispielsweise indem man Artefakte schafft, die die invarianten Eigenschaften herauslösen und aufbewahren, muß man mit der Entwicklung nicht immer wieder von vorne anfangen, kann man Teillösungen erzielen, auf denen sich aufbauen läßt und die auch für sich, unabhängig von Gesamtzusammenhang, optimiert werden können."

[Keil-Slawik 92]

Für die Konfigurierung bedeutet dies, dass die verschiedenen Ebenen, die in dem Automaten abgebildet werden, so aufgeteilt werden, dass für jede Ebene Artefakte entstehen, die in der nächsten Ebene verwendet werden.

Eine Ebene ist programmiertechnisch, d.h. die Sequenzen von Aktivitäten werden dort zusammengestellt. Diese Ebene wird jetzt durch die Steuerungsevents abgebildet. Das Ergebnis dieser Ebene ist eine Methode, welche die Sequenzen von Methoden zusammenfaßt und durch einen sprechenden Namen repräsentiert.

Die nächste Ebene ist die Fachliche der Kasse. In dieser kann das Teilergebnis der letzten Ebene benutzt werden. Ich nenne sie die Teilvorgang-Ebene der Konfigurierung. In dieser wird nur auf externe Ereignisse reagiert. Es wird in dieser Ebene alle Methoden zu einem Teilvorgang zusammengefaßt. Dieser kann dann als Ergebnis für die Gesamtkonfigurierung benutzt werden. Dieser Teilvorgang soll als ein Artefakt abgebildet werden.

Für einen Teilvorgang müssen die folgenden Informationen bearbeitet und gespeichert werden:

Gibt es innerhalb des Teilvorgangs eine zwingende Reihenfolge oder logische Abhängigkeiten?

- Welche Funktionalität hat der Teilvorgang?
- Welche Eingabe ist möglich?
- Welche Ein- und Ausgabedaten müssen vorhanden sein?
- Welche Wahlmöglichkeiten gibt es?
- Welche Fehlerquellen gibt es?

Eine andere Ebene ist die Steuerung der Subautomaten. Auf dieser Ebene wird die Reihenfolge der Teilvorgänge gesteuert. Als Ergebnis dieser Ebene wird ein Vorgang als Artefakte angeboten.

Für diesen Vorgang müssen ähnliche Informationen, wie beim Teilvorgang angeboten werden:

Gibt es zwischen den Teilvorgängen eine zwingende Reihenfolge oder logische Abhängigkeiten?

- Welche Funktionalität hat der Vorgang?
- Welche Eingabe ist möglich?
- Welche Ein- und Ausgabedaten müssen vorhanden sein?
- Welche Wahlmöglichkeiten gibt es?
- Welche Fehlerquellen gibt es?

Als weitere Ebene müssen diese Vorgänge für eine Kasse zusammengefügt werden. Welche Vorgänge soll es in der konkreten Kasse geben und zu welchem Zeitpunkt kann ich die Vorgänge starten.

Auf diese Art und Weise sollte es möglich sein, sich schnell wieder in die Konfigurierung eines Teilvorgangs einzuarbeiten. Welche Konzepte nötig sind, um diese Anforderungen zu erfüllen, beschreibe ich im nächsten Kapitel.

# 4 Darstellung des Lösungsansatzes

Im letzen Kapitel habe ich die bisherige Ablaufkonfigurierung analysiert und Anforderungen für eine Konfigurierung erstellt. In diesem Kapitel beschreibe ich meinen Lösungsansatz.

Zuerst beschreibe ich, zum besseren Verständnis des fachlichen Hintergrunds, ein Szenario über einen Verkaufsvorgang. Im nächsten Teil setze ich mich mit der Granualität der Konfigurierung auseinander. Mein Ansatz baut auf zwei Ansätzen auf. Der erste ist die Vergegenständlichung aus dem Werkzeug- & Materialansatz, der zweite ist das Konzept des Prozessmuster als flexibles Muster aus der Dissertation von [Gryczan96]. Mit Hilfe von diesen Ansätzen werde ich in diesem Kapitel ein Konzept entwickeln, das die Anforderung aus Kapitel 3, die Konfigurierung von Kassen unter der Verwendung von Artefakten, erfüllt.

Meinen Lösungsansatz kann man wie folgt kurz zusammenfassen; die Ablauf-Konfigurierung wird auf eine Parameter-Konfigurierung umgestellt, oder anders formuliert, die Abläufe werden ausprogrammiert, erhalten dadurch ein "Skelett" und werden dann durch komplexe Parameter dynamisch an verschiedene Kunden angepaßt.

## 4.1 Szenario Verkaufsvorgang

Zur Illustration möchte ich zuerst einen normalen Verkaufsvorgang anhand eines Beispieles in einem Szenario verdeutlichen. In der Abbildung 4-1 habe ich den zugehörigem Kassenbon abgebildet, der ein paar Begriffe skizziert. In der Abbildung 4-2 ist die zugehörige Kassentastatur zu sehen.

**Standardverkauf**

Ein Kunde tritt mit seinem Warenkorb an die Kasse. Er legt die Waren auf das Förderband.

Die Kassiererin registriert den ersten Artikel, einen Blumenkohl, indem Sie die Taste Warengruppe "WGR" drückt, die "12" für (Gemüse/Obst) eintippt, den Preis "389" Pf eingibt und die Eingabetaste drückt. Es wird der Bonkopf und eine Bonzeile ausgedruckt. Die Artikelbezeichnung und der Preis werden in dieser Filialkette immer nach einer Artikelregistrierung auf dem Kundendisplay und auf dem Kassiererdisplay angezeigt.

Als nächstes registriert sie einen Joghurt, indem sie den Barcode scannt. Der Artikelname und der Preis 0,99 DM wird auf dem Bon in einer Bonzeile gedruckt und zusätzlich am Display angezeigt.

Der Kunde beschwert sich, weil der Joghurt im Angebot ist und nur 0,89 DM kostet. Die Kassiererin sieht in der Angebotsliste nach und findet die Aussage des Kunden bestätigt. Daraufhin storniert sie die letzte Bonzeile, indem sie die Taste Sofortstorno drückt. Der Drucker druckt die Stornozeile auf den Bon.

Weil sie nicht dazu berechtigt ist, einen Preis zu ändern, ruft sie ihre Kollegin. Die Kollegin ist Aufsichtskassiererin und besitzt daher das Recht einen Sonderpreisverkauf durchzuführen. Sie führt jetzt für den Joghurt eine Preisüberschreibung durch, indem Sie die Taste Sonderpreis drückt. Auf dem Display erscheint die Meldung "SONDERPREISVERKAUF". Weiterhin wird die Aufsichtskassiererin aufgefordert, ihre Personalnummer und Paßwort einzugeben. Sie gibt beides ein. Auf dem Display erscheint die Meldung "ARTIKEL/SONDERPREIS" . Sie scannt den Barcode des Joghurt ein und gibt den neuen Preis von 89 Pf ein, indem sie im Ziffernblock die Tasten "8" "9" und die Eingabetaste drückt. Die Aufsichtskassiererin wird automatisch wieder abgemeldet. Intern wurde für den Sonderpreisverkauf eine spezielle Bonsequenz erzeugt, in der gespeichert wird, wer sich angemeldet hat und die Artikelregistrierungsdaten. Auf dem Bon wird eine normale Artikelregistrierzeile gedruckt.

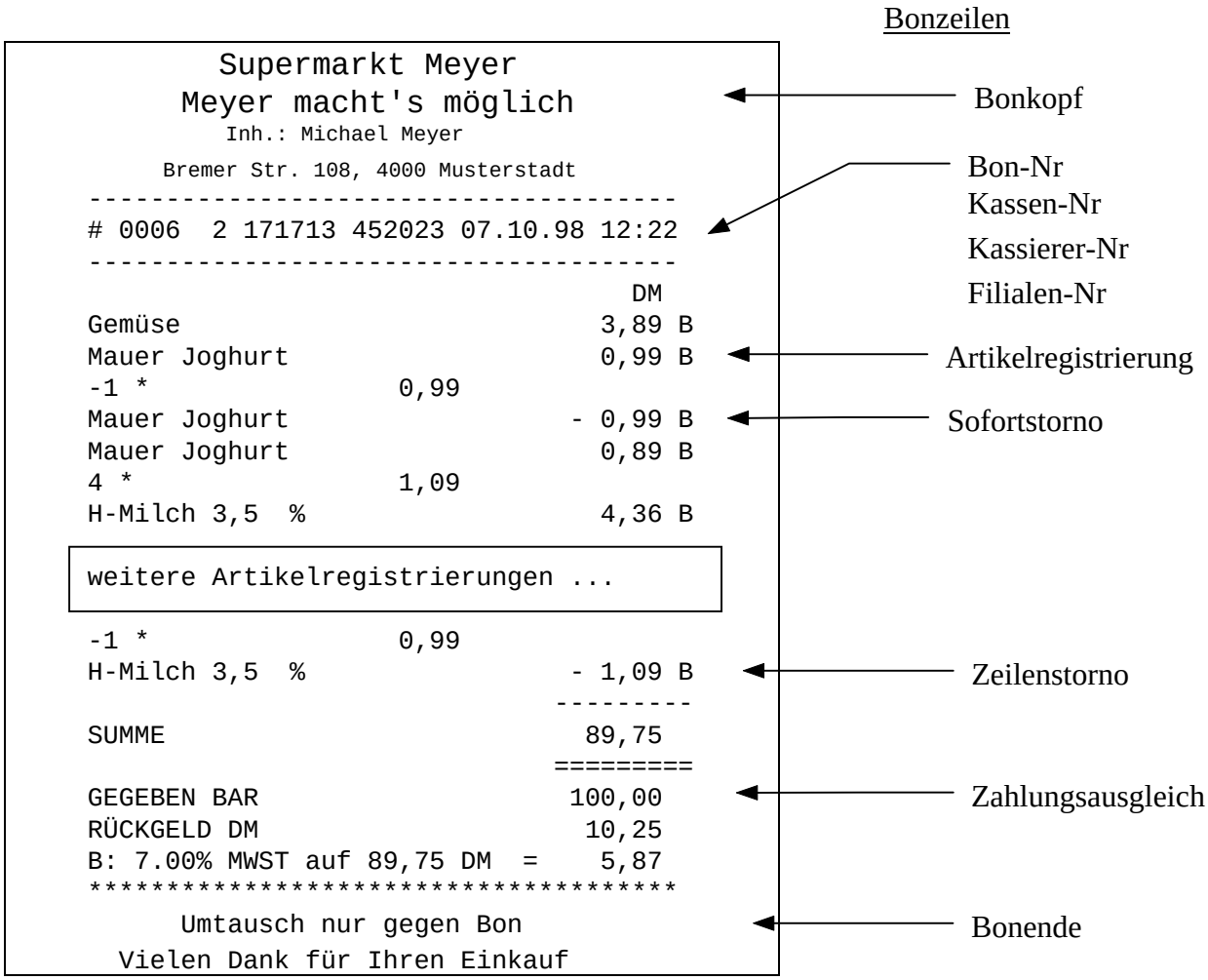


**Abbildung 4-1: Bon für das Szenario Standardverkauf**

Als nächstes registriert die Kassiererin vier Packungen Milch, indem sie zuerst die Zifferntaste "4" und dann die Taste Menge "*" drückt und den Barcode einer Milchpackung scannt.

Die nächsten Artikel werden alle über den Scanner eingelesen.

Der Kunde bemerkt, dass eine Milchpackung über dem Verfallsdatum ist, er möchte sie zurückgeben. Die Kassiererin bietet ihm an, sich eine neue zu holen, doch er lehnt ab, weil er es eilig hat. Also führt sie ein Zeilenstorno durch. Sie drückt die Zeilenstornotaste, auf dem Display erscheint Zeilenstorno, sie scannt eine Milchpackung. Die Stornozeile " -1 X H-Milch 3,5 % -1,09" wird auf dem Bon ausgedruckt.

Die Kassiererin drückt die Summentaste. Die Summe wird auf dem Bon gedruckt und der Betrag wird auf dem Display angezeigt. Die Summe beträgt 89,75 DM.

Der Kunde zahlt 100,- DM bar. Die Kassiererin drückt die "100 DM BAR" Taste, die Schublade öffnet sich. Sie gibt den Kassenbon und das auf dem Kunden- und Kassiererdisplay angezeigte Wechselgeld von 10,25 DM heraus und schließt die Schublade.

Zu diesem Zeitpunkt wird auf dem Journal-Bon-Drucker ein Duplikat des Bons gedruckt.


**Abbildung 4-2: Belegung einer Kassentastatur**

## 4.2 Lösungsansatz

Aus dem Kapitel 3 ergab sich die Anforderung, die Bedienvorgänge als Artefakte zu behandeln und eine strukturierte Programmiersprache anzubieten, gleichzeitig besteht aber immer noch die Anforderung, ein flexibles Kassensystem zu entwickeln, welches es ermöglicht, spezielle Kundenwünsche mit einem geringen Kostenaufwand zu verwirklichen.

Mein Lösungsansatz ist die Konfigurierung des Kassensystems in mehrere Stufen zu unterteilen. Zum einem wird der Ablauf vergegenständlicht. Die Bedienvorgänge mit ihrem Anwendungsmodell werden in Klassen abgebildet. Zwischen den Modell des Anwendungssystems und dem Anwendungsmodell wird versucht, eine möglichst hohe Strukturgleichheit zu erzielen. Die Bedienvorgänge werden objektorientiert abgebildet, um sie als Artefakt zu nutzen. Die Flexibilität wird durch die Konfigurierung der Bedienvorgängen erreicht. Es werden durch die objektorientierten Techniken wie Subklassen und Komposition Klassenfamilien erstellt, die bei der Konfigurierung genutzt werden können.

Die Semantik und die Parametrisierungs-Varianten dieser Bausteine werden durch eine Komponententechnologie explizit dargestellt, so dass die Konfigurierung der einzelnen Bedienvorgängen eine kassenfachliche Konfigurierung wird. Eine Komponente kann Informationen darüber liefern, welche Verbindungsmöglichkeiten existieren. Ein Bedienvorgang kann dann mit verschiedenen Parameter konfiguriert werden, wobei die Parameter recht komplex sein können, d.h. dass ganze Teilvorgänge ausgetauscht werden. Jetzt wird der Ablauf nicht mehr konfiguriert, sondern es werden Ablaufkomponenten durch die Konfiguration verbunden. Die reine Ablauf-Konfigurierung gibt es nicht mehr. Wenn wesentliche Änderungen in einem Ablauf vorgenommen werden müssen, wird dieser neuer Vorgang innerhalb von Kassen-Rahmenwerk ausprogrammiert.

Durch die Aufteilung der Konfigurierung in, Erstellung eines Bedienvorgangs, Einstellung der Werkzeuge und Parametrisierung der Vorgänge mit den Werkzeugen, erfolgt eine mehrstufige Konfigurierung, die intellektuell einfacher zu handhaben ist, weil immer Artefakte erstellt werden, die in ihrer Ganzheit genutzt werden können.

Das Konzept, welches ich in diesem Kapitel beschreibe, hat eine andere Sichtweise von Flexibilität, bzw. Konfigurierung als es die Ersteller des Kassensystems hatten; das System ist nicht mehr vollkommen flexibel und offen. Die Flexibilität hängt jetzt von dem anwendungsfachlichem Kontext ab. Dadurch wird eine höhere Wartbarkeit erreicht. Die Bedienungsvorgänge werden in objektorientierte Klassen gekapselt, um sie als Artefakt zu benutzen. Diese Klassen werden noch mit einer weiteren Hülle umgeben, damit sie bei der Konfigurierung genutzt werden können, diese Hülle nenne ich Komponenten-Interface. Genauer gehe ich im Kapitel 6 auf die Komponententechnik ein. Mit diesem Komponenten-Interface kann dann das Baukastenprinzip weiter verfolgt werden. Sie entsprechen erst dann Bausteinen, wenn sie ein explizites Interface haben, mit dem etwas zusammengesteckt werden kann. Oft werden Komponenten-Systeme, z.B. in Werbebroschüren, mit den Bausteinsystem von der Firma Lego© verglichen. Dort haben einzelne Bausteine explizite "Schnittstellen", ein einfacher Baustein hat z.B. acht kleine Noppen auf der Oberseite und auf der Unterseite eine Lochstruktur, die es ermöglicht diesen Stein auf die Noppen von anderen Steinen zu setzen, so dass aus mehreren Bausteinen ein neuer Gegenstand erstellt werden kann. Bei der Weiterentwicklung des Produktes wurden spezifische Bausteine entwickelt. Es gibt je nach Ausrichtung Spezialbausteine für Autos, Weltraumflugobjekte oder Wasserfahrzeugen. Durch diese Spezialbausteine kann ein Kind z.B. schnell ein Weltraumobjekt erstellen, weil es die einzelnen Teile eines Objektes nicht mehr selbst erstellen muß. Eine Satellitenschüsselbaustein oder ein Sitzbaustein kann einfach benutzt werden. Eines ist aber gleich geblieben, jedes dieser Bausteine hat definierte Schnittstellen, an denen das Kind erkennen kann, wie es die Bausteine verbinden kann.

## 4.3 Vergegenständlichung aus dem Werkzeug- & Materialansatz

Die Modelle der Vorgangsklassen werden nach dem Werkzeug- & Materialansatz(kurz WAM-Ansatz) konzipiert. [WAM 98]

Der WAM-Ansatz hat als besonderes Merkmal die Anwendungsorientierung. Züllighoven beschreibt dies so:

> "Anwendungsorientierte Software zeichnet sich durch hohe Gebrauchsqualität aus, die wir so charakterisieren:

- Die Funktionalität des Systems orientiert sich an den Aufgaben aus dem Anwendungsbereich.
- Die Handhabung des Systems ist benutzergerecht.
- Die im System festgelegten Abläufe und Schritte lassen sich je nach Anwendungssituation problemlos an die tatsächlichen Anforderungen anpassen." [WAM 98 S.4]

Ein wesentliche Grundsatz, um dieses Ziel zu erreichen, ist die Strukturähnlichkeit. Die wird von Züllighoven folgenderweise definiert.

**Definition 3: Strukturähnlichkeit:**

> "Strukturähnlichkeit bezieht sich im WAM-Ansatz auf das Verhältnis von Software und Anwendungsbereich: Die softwaretechnischen Komponenten eines Softwaresystems modellieren die relevanten Konzepte und Gegenstände des Anwendungsbereichs. Die Architektur des Anwendungssystems spiegelt die wesentlichen Bezüge zwischen den Konzepten und Gegenständen des Anwendungsbereichs wider. Diese *Strukturähnlichkeit* hat zwei entscheidende Vorteile: Die Anwender finden die Gegenstände ihrer Arbeit und die Begriffe ihrer Fachsprache im Anwendungssystem repräsentiert. Sie können entsprechend ihre Arbeit in gewohnter Weise organisieren. Die Entwickler können Softwarekomponenten und Anwendungskonzepte bei fachlichen und softwaretechnischen Änderungen zueinander in Beziehung setzen und somit die wechselseitigen Abhängigkeiten erkennen."
>
> [WAM 98 S.126]

Diese Strukturähnlichkeit wurde im Rahmen eines Leitbildes durch Entwurfsmetapher erreicht. Die Gegenstände werden durch die Entwurfsmetaphern Werkzeug, Material und Automat abgebildet. Das häufigste Leitbild ist bei WAM der Arbeitsplatz für qualifizierte und eigenverantwortliche Tätigkeiten. In diesem Leitbild wird der Prozess eines Arbeitsvorgangs nicht von dem Anwendungssystem sondern eigenverantwortlich durch den hochqualifizierten Mitarbeiter bestimmt. Dies spiegelt sich in der Struktur der Werkzeuge wider. Werkzeug wird so definiert

**Definition 4: Werkzeug**

> "Ein *Werkzeug* unterstützt wiederkehrende Arbeitsabläufe und -handlungen. Es ist bei unterschiedlichen Aufgaben und Zielsetzungen nützlich. Ein Werkzeug wird von seinem Benutzer je nach den Erfordernissen einer Situation gehandhabt oder wieder zur Seite gelegt. Es schreibt keine festen Arbeitsabläufe vor. Als Softwarewerkzeug ermöglicht es den interaktiven Umgang mit den Arbeitsgegenständen."

Der Gegenstand, den das Werkzeug bearbeitet, wird im WAM-Ansatz Material genannt.

**Definition 5: Materialien**

> *"Materialien* sind die Arbeitsgegenstände, die schließlich zum Arbeitsergebnis werden. Materialien werden mit Werkzeugen entsprechend bearbeitet. Softwarematerialien verkörpern »reine« anwendungsfachliche Funktionalität. Sie werden niemals direkt benutzt und stellen sich auch nicht selbst dar. Ein Softwarematerial ist durch sein Verhalten, nicht durch seine Struktur charakterisiert."

Diese Strukturähnlichkeit wurde in WAM-Ansatz bisher nur auf Gegenstände und ihre Konzepte angewandt. Abläufe wurden nur im geringen Umfang in die Überlegungen mit einbezogen, weil das Leitbild von WAM sich meistens auf den Arbeitsplatz für qualifizierte und eigenverantwortliche Tätigkeiten bezieht. Ein Beispiel dafür ist ein Sachbearbeiter im Bankbereich. Für dieses Leitbild trägt das Bild, das er einen Schreibtisch hat, auf dem Werkzeuge zur seiner Unterstützung liegen. In welcher Reihenfolge er die Werkzeuge benutzt liegt in seiner Verantwortung.

Im Kassenbereich besteht ein anderes Leitbild vom Arbeitsplatz. Durch die Vorgänge an der Kasse werden die Geld- und Warenbewegungen zwischen Filiale und Kunden erfaßt, damit sie später ausgewertet und kontrolliert werden können. Dies ist der zentrale Platz des Warenabganges und des Geldzuflußes zwischen Geschäft und Kunden. Gleichzeitig sind an der Kasse keine hochqualifizierten Mitarbeiter tätig, sondern zum größten Teil Zeitkräfte, die oft kurzfristig eingearbeitet werden müssen. Weil der Kassenbereich ein hoch sensibler Bereich ist, sollen die Betrugsmöglichkeiten durch die Kassierer auf ein Minimum beschränkt werden. Die Vorgänge an der Kasse sollen daher sehr genau festgelegt sein. Die Prozessverkettung geht daher nicht von dem Menschen, sondern von der Maschine aus. Der Kassenprozess bestimmt welcher nächster Teilvorgang erlaubt bzw. zwingend dran ist.

Der Verkauf und die anderen Vorgänge sind nach dem WAM-Ansatz Automaten im oberen Handlungskontext. [vgl. Gryczan95] Sie steuern die Handlungen der Benutzer. Automaten im unteren Handlungskontext sollen nur lästige Routinearbeiten ersetzten, während Automaten im oberen Handlungskontext, die Handlungsfähigkeit des Benutzers einschränken.

**Definition 6: Automat im unteren Handlungskontext**

> "*Automaten* sind im Rahmen einer zur erledigenden Aufgabe ein Arbeitsmittel, um Material zu bearbeiten. Sie erledigen lästige Routinetätigkeiten als eine definierte Folge von Arbeitsschritten mit festem Ergebnis ohne weitere äußere Eingriffe.
> *Automaten* laufen unauffällig im Hintergrund, wenn sie einmal vom Benutzer oder von der Arbeitsumgebung gestartet sind.
> Sie können auf ihren Zustand überprüft und im vorgegebenen Rahmen eingestellt werden."

Der Automat im oberen Handlungskontext steuert die Handlungsfolge. Er bestimmt welche Handlungen als nächstes erlaubt sind. Der Automat sorgt dafür, dass sich das Anwendungssystem in einem konsistenten Zustand befindet. Es werden nur Handlungen zugelassen, die die Materialien in einem fachlichen konsistenten Zustand bringen. Automaten im oberen Handlungskontext stellen deshalb nur die Werkzeuge zur Verfügung, die im fachlichen Kontext einen Sinn ergeben. Damit dies für das System sicher gestellt ist, muss der Softwareentwickler das System sehr genau analysiert haben, weil im Falle eines inkonsistenten Zustandes der Benutzer des Systems nur wenig Einflussmöglichkeiten hat. Die *Prozeßverkettung* geht vom Automaten aus, dies bedeutet die einzelnen Arbeitsschritte werden durch die Software vorgegeben und in einen Gesamtvorgang eingebunden. Der Benutzer hat nur so eine große Handlungsfreiheit, wie der Automat es ihm vorgibt. Dies bedeutet, dass der Automat dem Benutzer Hilfen zur Seite geben muss, die es ihm ermöglichen, die nächste Handlung auszuführen.

In einer Definition halte ich die Ergebnisse fest:

**Definition 7: Automat im oberen Handlungskontext**

> Automaten im oberen Handlungskontext schränken die Handlungsfreiheit der Benutzer ein. Die Prozessverkettung geht vom Automaten aus. Der Automat legt fest welche Arbeitsschritte zu einem Zeitpunkt erlaubt sind. Der Automat sorgt dafür, dass sich das Material und das Softwaresystem in einem fachlich konsistentem Zustand befindet.

## 4.4 Prozessmuster für Kooperation

Die Vorgänge sollen aber nicht nur einfach vollständig ausprogrammiert werden, weil das Kassen-Rahmenwerk für verschiedene Filialketten einstellbar sein soll. Aus diesem Grund greife ich die strukturähnliche Abbildung von Gegenständen aus WAM auf und übertrage sie auf die Bedienvorgänge.

Bedienvorgänge werden vergegenständlicht. Es gibt im Einzelhandel eine Begriffsbildung für diese Gegenstände. Die Bedienvorgänge können benannt werden, es gibt z.B. den Standardverkauf, dass ist der normale bzw. die häufigste Verkaufsart, den Personalverkauf und den Gewerbeverkauf. Diese Vorgänge unterscheiden sich durch, den Ablauf, unterschiedliche Arten der Bezahlung und unterschiedliche Preisberechnungen. Der Gewerbe- und Personalkunde bekommt z.B. einen Rabatt auf die Artikel und der Gewerbekunde darf auf Rechnung einkaufen.

Bedienvorgänge legen einen gewissen Ablauf fest, es ist jedoch kein total festgelegter Ablauf in dem Sinne, dass erst immer ein Teilvorgang A erfolgt danach immer zwingend ein Teilvorgang B. Es bestehen zwar fachliche Abhängigkeiten d.h. ein Teilvorgang darf erst dann erfolgen, wenn vorher ein anderer abgeschlossen wurde. Dieser muß aber nicht direkt vor dem anderen ausgeführt werden, z.B. kann nur ein Artikel während eines Verkaufsvorgang storniert werden, wenn er vorher registriert wurde. Diese Abhängigkeiten erinnerten mich an die Art wie Vorgänge für Kooperation im WAM-Ansatz beschrieben werden. Für die Sachbearbeiter, z.B. einer Bank, die innerhalb eines kleinen Kreis zusammenarbeiten, wurde das Modell des Prozessmusters von Gryczan ausgearbeitet. Das Prozessmuster vergegenständlicht die Zusammenarbeit. Gryczan definiert Prozessmuster folgenderweise.

**Definition 8: Prozessmuster**

> "Prozessmuster vergegenständlichen das zur routinisierten Zusammenarbeit passende Kooperationsmodell. Mit ihnen können Verantwortlichkeiten innerhalb einer Kooperation beschrieben und verändert werden und sie machen die Koordination deutlich." [WAM 98,S.449 ]

Prozessmuster vergegenständlichen ein flexibles Muster von Zusammenarbeit.

**Definition 9: flexibles Muster**

> Ein flexibles Muster hat einen statischen und einen dynamischen Anteil. Der statische Anteil ist die immer wieder kehrende Invariante. Das dynamische ist die Anpassung an die jeweilige Situation. "Es ist eine Handlungseinheit aus Handlungsmöglichkeiten und Gegenstand mit einer bestimmten Struktur und einem 'um zu'. Ein flexibles Muster wird in einer Situation durch die Verbindung eines Gegenstandes mit einer Handlung instantiiert " [Gryczan 95 S. 63 ff]

### 4.4.1 Prozessmuster Laufzettel

Das Prozessmuster wurden in dem Kontext von qualifizierten Sachbearbeitern als Material Laufzettel ausgeprägt. [vgl. WAM 98,S.449 ff] Laufzettel lassen sich an Vorgangsmappen heften.

> "Ein Laufzettel ist ein Arbeitsgegenstand, der von einem Anwender für einen speziellen Vorgang erstellt wird. Dazu gehört die Festlegung, wer für welche Tätigkeiten zuständig ist, welche Reihenfolge zwischen den Arbeitsschritten besteht und welche Dokumente benötigt werden.
>
> Ein Laufzettel kann für Routinevorgänge aus einer Sammlung vorformulierter Laufzettel entnommen werden. Zeichnet sich ein neuer Routinevorgang ab, kann der Anwender diese Sammlung durch einen prototypischen Laufzettel (im Sinne einer Vorlage) erweitern.
>
> Laufzettel lassen sich entsprechend den Anforderungen der konkreten Situation von Benutzern verändern oder anpassen. Dies gilt natürlich nicht für bereits abgeschlossene Arbeitsschritte, die unveränderbar dokumentiert werden. Geändert werden können nur Zuständigkeiten (etwa wenn der vorgesehene Kollege erkrankt ist) und zu erledigende Tätigkeiten (etwa wenn ein Arbeitsschritt nur unzureichend erledigt wurde und wiederholt werden muß).
>
> Ein Laufzettel ist eine Anweisung für den Transport der damit verbundenen Vorgangsmappe. Auf Basis der im Laufzettel vereinbarten Zuständigkeiten versendet das Postversandsystem die Mappe zum jeweils zuständigen Arbeitsplatz.
>
> Anwender koordinieren ihre Zusammenarbeit anhand von Laufzetteln. Sie informieren sich über den Stand der Vorgangsbearbeitung. Sie sehen, wer bisher welche Tätigkeit erledigt hat und was noch zu erledigen ist und haken selbst einzelne Tätigkeiten als erledigt ab.
>
> Vorgangsmappen können nachverfolgt werden. Ein Anwender kann beim Postversandsystem anfragen, an welchem Arbeitsplatz eine Vorgangsmappe derzeit in Arbeit ist und wo sie bisher war. Diese Nachfrage sagt aber nichts darüber aus, ob und wie der Inhalt der Vorgangsmappe derzeit auf dem Arbeitsplatz bearbeitet wird."

Laufzettel lassen sich entsprechend den Anforderungen der konkreten Situation von Benutzern verändern oder anpassen. Dies bedeutet die Mitarbeiter können im laufenden Prozess den Laufzettel ändern und damit flexibel an die bestehenden Geschäftsbedingungen anpassen. Ein Laufzettel wird von einem Sachbearbeiter angelegt und an ihm können dann die verschiedenen Zustände markiert werden. Wie in der Abbildung 4-3 wurde durch einen Haken festgelegt, dass die Tätigkeit Kundenzuordnung abgeschlossen ist.

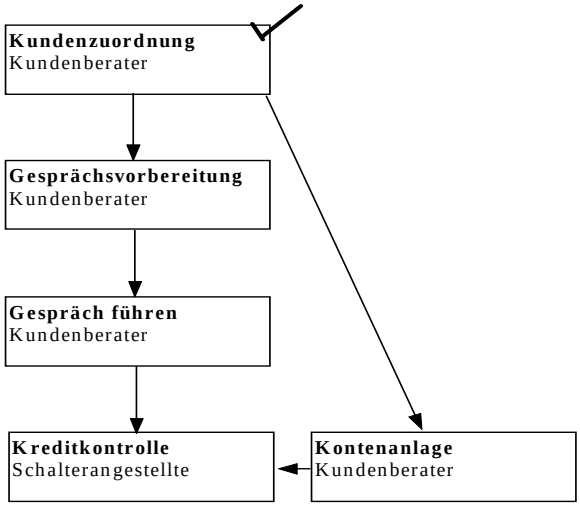


**Abbildung 4-3: Ein Prozessmuster für Kreditverarbeitung [WAM 98 S.450]**

Für die Vorgänge der Kasse muß dieses Prozessmuster angepaßt werden. Die Vorgänge der Kasse sind feiner strukturiert, sie werden im Regelfall von einer Person durchgeführt und der Prozess wird stärker von der Maschine bestimmt. Außerdem soll es bei den Kassenvorgängen nicht möglich sein, zur Laufzeit das Prozessmuster im Ablauf zu ändern d.h. die Kassierer können die noch zu erledigen Tätigkeiten nicht selbst verändern.

Das Prozessmuster Laufzettel spielt technisch in einem anderem Bereich. Der Nachrichtenfluß zwischen den einzelnen Prozessen ist synchron. In der Kasse existiert ein zum Teil asynchrones Verhalten. Einzelne Eingabegeräte laufen in eigenen Prozesse. Die Events der Eingabegeräte werden in einer Eventschleife serialisiert. Daher gibt es ein anderes Eventmodell als bei der Kooperation. Events bei der Kasse enthalten Daten, die weitergeleitet und verarbeitet werden.

Und doch kann hier das Prinzip des flexibles Musters angewandt werden.

Das flexible Muster wird jedoch nicht situativ an die momentane Situation angepaßt. Das flexible Muster wird hier auf einer anderen Ebene behandelt.

Die einzelnen Vorgänge einer Kasse können sehr verschieden aussehen und doch kann man Muster erkennen, die sich immer wieder wiederholen. Die flexible Anpassung erfolgt bei der Konfigurierung der Bedienvorgänge.

Die statische Abbildung der Prozesse findet durch eine Abbildung der Vorgänge in Klassen statt. Die dynamische Adaption an die jeweilige Filialkette geschieht durch die Parametrisierung der Kassenvorgänge.

## 4.5 Ablaufstrategie für Kassenbedienvorgänge

Für die Kassenvorgänge kann man ein Menge von Vorgängen angeben, die zwar immer wieder leicht abgeändert werden, aber im Prinzip sehr ähnlich in sogenannte Muster von Vorgängen abgebildet werden können. Verdeutlichen möchte ich dies im Bereich Verkauf. Im Kapitel 4.1 habe ich in einem Szenario einen Standardverkauf beschrieben. In diesem Verkaufsvorgang kann man verschiedene Phasen eines Verkaufsvorganges erkennen. Die erste ist der Start eines Verkaufs. In dieser Phase wird der Bon erzeugt. Beim Verkauf wird der Bon erst dann erzeugt, wenn die erste Artikelregistrierung vollständig durchgeführt wurde. Danach steht der Verkauf im Bon. In Anlehnung an die alte Registrierkasse, bei der es klingelte wenn ein Artikel registriert wurde, wird dieser Zustand "after the first ring" genannt. In diesem Zustand kann der Verkaufsvorgang nur durch eine explizite Stornierung abgebrochen werden. Es werden im Normalfall weitere Artikel registriert und eventuell einzelne Bonzeilen storniert. Es können auch Zwischensummen über den Bon gebildet werden. In die nächste Phase bzw. Zustand kommt der Bon, wenn die Endsumme über den Bon gebildet wurde, der Zahlungsausgleich. In dieser Phase dürfen keine Artikel mehr erfaßt werden. Die Endsumme unterscheidet sich von der Zwischensumme dadurch, dass eventuell ein Rabatt für spezielle Kunden ausgegeben wird. Die Phase nach der Endsumme wird Zahlungsausgleich genannt, das bedeutet im Normalfall, dass der Kunde den Summenbetrag in ein oder mehreren Teilbeträgen bezahlt. Nach dem Zahlungsausgleich wird der Bon abgeschlossen. Das Bonende wird erzeugt.

Dieser Vorgang habe ich abstrakt in der Abbildung 4-4 beschrieben. Die Pfeile zwischen den Kästen haben die fachliche Bedeutung, dass eine Tätigkeit vor der anderen Tätigkeit erfolgen muß. Der Kasten um die Tätigkeiten Artikelregistrierung, Kundenidentifizierung, Storno und Rabatt, hat die Bedeutung, dass die Tätigkeiten innerhalb des Kastens optional ausgeführt werden können.

In der Abbildung 4-4: Ablaufstrategie Standardverkauf wird dieser Vorgang abgebildet.

Ein Kassenvorgang hat also ein Muster nach dem der Vorgang gesteuert wird. Ich nenne dieses Muster Ablaufstrategie. Nach welcher Ablaufstrategie der Vorgang gesteuert wird, hängt von verschiedenen Faktoren ab. Zuerst von der *Verkaufsart*. Ist es ein "normaler" Verkaufsvorgang oder ist es ein spezieller, bei dem eine besonderer Ablaufstrategie gewählt wird. Dies ist z.B. beim Happy Hour Verkauf der Fall. Für diesen Verkauf wird auf alle Artikel einer Warengruppe 50% Rabatt gegeben.

Ein weiteres Kriterium für eine Ablaufstrategie ist der *Kunde*. Wenn ein Kunde bei einem Verkauf, durch eine irgendwie geartete Anmeldung, identifiziert wird, kann es z.B. sein, dass der Kunde bestimmte Artikel nicht kaufen darf, weil er z.B. nur mit einem Gutschein bezahlen darf oder für den Kunden gibt es besondere *Preisberechnungsstrategien*, wie z.B. Personalrabatt.

Ein weiteres Kriterium für eine Ablaufstrategie ist der Artikel. Bestimmte Artikel verlangen eine besondere Eingabe von Maßen oder Farben. Dies verändert auch die Ablaufstrategie. Es muß z.B. für Stoff die Länge und Breite angegeben werden.

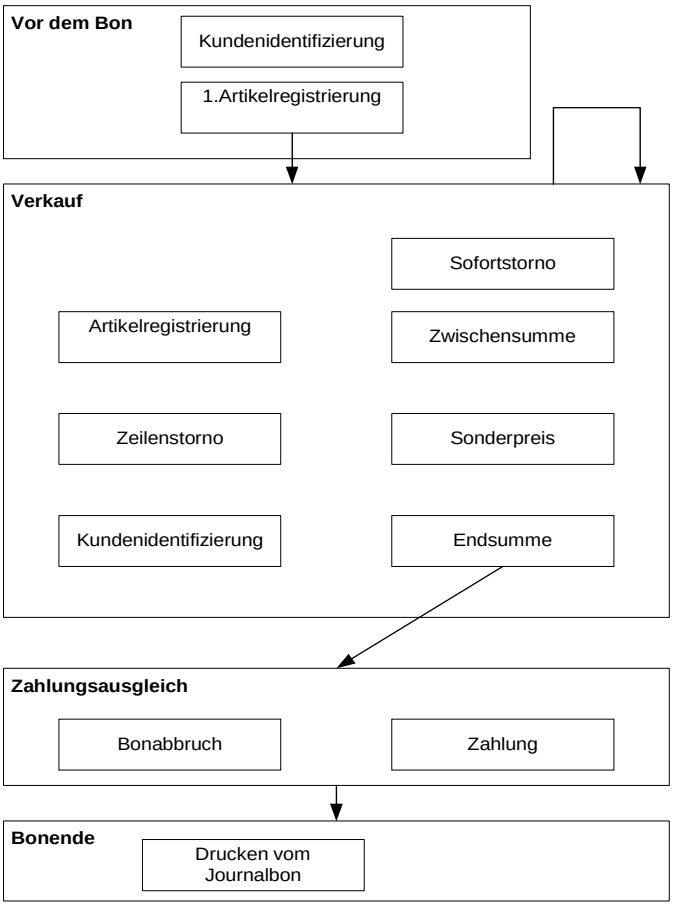


**Abbildung 4-4: Ablaufstrategie Standardverkauf**

In der Abbildung 4-4 verwende ich als Grundlage die grafische Notation vom Prozessmusterkonzept. Martina Wulf hat sie als Grundlage für die grafische Beschreibung von Tätigkeitsnetzen genommen [vgl. Wulf 95 S.71 ] Ich verwende hier Erweiterungen, die noch nicht formal begründet sind. Auf die grafische Darstellung von flexiblen Abläufen gehe ich in dieser Arbeit nicht weiter ein.

### 4.5.1 Vorgang und Tätigkeit

In der Kasse gibt es Muster für die Vorgänge. Es gibt Vorgänge in der Kasse, die sich so stark ähneln, dass man ein statisches Gerüst für sie erstellen kann. Dieses Gerüst hat Punkte, bei denen unterschiedliche Variationen möglich sind. Es kann eine Sammlung von Bedienvorgängen erstellt werden. Wenn für eine Handelskette eine neue Kasse erstellt werden soll, kann aus den Bedienvorgängen ausgewählt werden Der Bedienvorgang kann für die jeweilige Handelskette durch Konfigurierung angepaßt werden. Falls eine ganz neue Art von Bedienvorgang gewünscht wird, kann auch ein neues Muster erstellt werden.

Ein Vorgang hat Werkzeuge, die für die Tätigkeiten genutzt werden können. Für den Verkauf gibt es z.B. Werkzeuge zur Artikelregistrierung und Zahlungswerkzeuge. Für den Ablauf hat der Vorgangs eine Ablaufstrategie. Die Ablaufstrategien enthalten Tätigkeiten die zueinander in Beziehung stehen. Es wird für den Vorgang festgelegt welche fachlichen Abhängigkeiten bestehen. Für jede Tätigkeit wird angegeben welche Werkzeuge benutzt werden sollen. Die Werkzeuge in der Kasse sind stark an Abläufe gebunden, deswegen können diese separat eingestellt werden. Es können Werkzeuge zu einem Kombi-Werkzeug zusammengestellt werden. Das Zahlwerkzeug kann z.B. ein Kombiwerkzeug sein, welches Bar- und Scheckzahlung ermöglicht. Für eine Tätigkeit kann eine Vorbedingung angegeben werden. Dies wäre z.B. für das Artikelregistrierung; "die tatsächliche Anzahl von Artikelregistrierung ist kleiner, als die maximale Anzahl von Artikelregistrierungen". Die Bedingungen beziehen sich dann auf das Material. Dies Material ist der Bon, in dem alle notwendigen Daten über die durchgeführten Tätigkeiten festgehalten werden.

Ein Vorgang hat verschiedene Tätigkeiten, die durch eine Ablaufstrategie gesteuert werden. Der Vorgang hat Werkzeuge, die für die Tätigkeiten benutzt werden können. Die Tätigkeiten werden im Normalfall von einem Rollenträger ausgeführt. Eine Rolle verwende ich in dem Sinne von [Floyd 97] als funktionelle Rolle. Dies bedeutet, dass eine Rolle die Beschreibung von Funktionen und Zuständigkeiten zusammenfaßt. Eine Person kann mehrere, wechselnde Rollen haben. Eine Angestellte in einer Handelskette kann z.B. die Rolle einer Kassiererin und die Rolle der Aufsichtskassiererin besitzen. Die Aufsichtskassiererin ist dazu berechtigt Preisüberschreibungen durchzuführen. Siehe Szenario Verkaufsvorgang.

Im Sonderfall soll eine Tätigkeit auch vom dem Kassenautomat selbst gestartet werden können. In dem Fall kann man aber nicht mehr von Tätigkeit sprechen, sondern von einer Operation. Denn zu einer Tätigkeit gehört nach [Floyd 97] Handlungsspielraum und Verantwortung. An der Stelle wird dann als Auszuführender kein Rollenträger sondern die Kasse eingetragen.

Die Ablaufstrategie legt fest welche Werkzeuge *startbereit* sind, d.h. welche Werkzeuge der Kassierer benutzen kann. Wenn ein Werkzeug nicht startbereit ist, muss das dem Benutzer gemeldet werden. Für ein Kassensystem gibt es oftmals keine komplexen Anzeigemöglichkeiten, so dass der Kassierer nicht immer erkennen kann welche Vorgänge erlaubt sind.

**Bedienvorgang am Kassensystem**

Ein Vorgang hat eine Ablaufstrategie und Tätigkeiten. Die Tätigkeiten stehen durch die Ablaufstrategie in Beziehung zueinander. Für den Vorgang stehen Werkzeuge in einem Behälter, die Werkzeugkiste, für die Erledigung der Tätigkeiten, zur Verfügung. Der Vorgang wird durch einen Automaten gesteuert. Der Automat steuert den Ablauf unter Benutzung der Ablaufstrategie. Er stellt dem Benutzer nur die Werkzeuge zur Verfügung, die nach der Ablaufstrategie benutzt werden dürfen. Ein Vorgang kann abstrakt formuliert werden. Er gibt an, welchen Typ die Werkzeuge für den Vorgang haben dürfen. Für jede Tätigkeit ist ein Ausführender festgelegt, der die Tätigkeit durchführen darf. Eine Tätigkeit kann in kleinere Teiltätigkeiten aufgeteilt werden. Jeder dieser Teiltätigkeiten hat dann wieder eine Ablaufstrategie. Die Daten des Vorganges werden in einem Bon protokolliert. Ein Vorgang kann Eigenschaften haben, die den Ablauf beeinflussen, wie z.B. die maximalen Anzahl von Artikelregistrierungen. Die Strategie eines Vorgangs kann während des laufenden Vorganges geändert werden, wenn sich wesentliche Strategieparameter geändert haben.

Die Tätigkeit wird mit einem Namen versehen. Eine Tätigkeit kann aus Teiltätigkeiten oder aus Beschreibung der Tätigkeit bestehen. Bei der Tätigkeit kann zusätzlich der Nachfolger angegeben werden, wenn es einen gibt. Für die Tätigkeit wird eine Rolle angegeben. Diese Rolle kann z.B. ein Kassierer, ein Aufsichtskassierer oder die Kasse sein. Wenn die Kasse eingetragen wurde, hat es die Bedeutung, dass das Kassensystem die Tätigkeit automatisch selbst startet. Ein Beispiel dafür ist "Journal-Bon drucken", diese Funktion wird automatisch am Ende des Vorgangs aufgerufen.

Für den Verkauf aus der Abbildung 4-4 habe ich die Beschreibung von verschiedenen Tätigkeiten hier aufgeführt:

**Tätigkeit**: Verkauf

Hat Teiltätigkeiten

- Artikelregistrierung
- Storno
- Rabatt
- Zwischensumme
- Endsumme

**Tätigkeit**: Zahlungsausgleich

Hat Teiltätigkeiten

- Zahlung
- Bonabbruch

**Tätigkeit**: Bonende

Hat Teiltätigkeiten

- JournalBon drucken

**Tätigkeit**: Artikelregistrierung

Werkzeug: Artikelregistrierer

Artikelfilter: kein

Ausführender :Kassierer

**Tätigkeit**: Endsumme bilden

Werkzeug: Endsummenerzeuger

Ausführender: Kassierer

Nachfolger: Zahlungsausgleich

**Tätigkeit**: Zahlung

Werkzeug: Zahlwerkzeug mit Zahlungsart: Bar, EC-Scheck

Ausführender: Kassierer

**Tätigkeit**: Journal-Bon drucken

Werkzeug: Formular-Drucker Journal-Bon

Ausführender: Automat

## 4.6 Technische Umsetzung mit Rahmenwerk und Componentware

Die statische Seite des flexiblen Musters, Bedienvorgänge, wird in einem Rahmenwerk abgebildet. Es entsteht daraus ein Skelett, dass verschiedene Schnittstellen anbietet, um es mit "Fleisch" zu füllen und "lebendig" zu machen. Diese Dynamik wird durch die Werkzeugkomponenten erreicht. Im Rahmenwerk werden die Bedienvorgänge als Klassen abgebildet, mit sogenannten flexiblen Punkten. Flexible Punkte bedeutet, dass genau an diesen Punkten Flexibilität vorgesehen wird. Pree nennt sie Hot Spots. [Pree 97] Bei diesen Hot Spots wird mit Hilfe von Entwurfsmustern oder Vererbung die gewünschte Flexibilität erreicht. Dabei wird für jeden Hot Spot mit Hilfe einer abstrakten Klasse eine Schnittstelle definiert. Diese Schnittstelle wird dann in einzelnen Klassen implementiert, diese Klassen sind jeweils für bestimmte Konfigurationen angepaßt. Damit können bei der Konfigurierung die geeignetsten Klassen ausgewählt werden und so eine speziell auf einen bestimmte Kundenkreis angepaßte Steuerung erstellt werden.

Die Werkzeuge werden technisch durch Komponenten realisiert. Diese Komponenten sind Halbfertigprodukte wie sie im Kapitel 6 definiert werden. Die Komponenten habe ich deswegen gewählt, weil sie die Möglichkeit bieten das Metawissen über das Werkzeug bzw. Automaten selbst zu repräsentieren. Ein Softwaremodule ist dadurch für die Konfigurierung und für den Einsatz in einem konkreten Kassensystem geeignet. Das Anwendungsmodell wird strukturerhaltend mit Hilfe des WAM-Ansatzes einmal abgebildet und kann durch die Komponententechnik doppelt genutzt werden.

Mit Hilfe dieser Werkzeugkomponenten und den Mustern von Ablaufstrategie kann für die Handelskette schnell eine Standardkasse erstellt werden. Wenn das vorhandene Ablaufmuster oder die vorhandenen Werkzeugkomponenten nicht ausreichen, können für den Kunden spezielle Ablaufstrategien bzw. Komponenten programmiert werden. Durch diese Art der Konfigurierung und Programmierung ist man flexibel und schnell. Es können vorhandene Halbfertigprodukte genutzt werden und sehr spezielle Kundenwünsche können durch die Mächtigkeit der benutzten Programmiersprache immer realisiert werden. Wenn es sich herausstellt, dass der spezielle Kundenwunsch doch eine Anforderung ist, die immer wieder in leicht abgewandelter Form von anderen Kunden kommt, kann eine neue Ablaufstrategie mit neuen Komponenten erstellt werden.

Wie das realisiert werden kann, erläutere ich in den nächsten beiden Kapiteln.

# 5 Konzeption eines Objektorientiertes Kassen-Rahmenwerks

In diesen Kapitel wird der Aufbau eines Kassen-Rahmenwerks dargestellt, der im Gegensatz zum alten Kassen-Rahmenwerk die Abläufe modelliert.

Bevor ich die Aufteilung des Rahmenwerks näher erläutere, werde ich die Modellarchitektur aus dem WAM-Ansatz darstellen, um das Kassen-Vorgangs-Rahmenwerk in einem größeren Kontext zu stellen.

## 5.1 Wie spielt das Kassen-Rahmenwerk zusammen

### 5.1.1 WAM-Modellarchitektur

Im WAM-Ansatz wird eine Modellarchitektur dargestellt, die für verschiedene Domänen einsetzbar ist. [WAM98 S.324ff]

**Definition 10: Modellarchitektur:**

> „Eine Modellarchitektur beschreibt die allgemeinen Prinzipien hinter einer Softwarearchitektur. Sie umfaßt die grundlegenden Elemente, deren Verknüpfungen und die Regeln, die für eine Softwarearchitektur gelten. Eine Modellarchitektur gibt Anleitung bei der softwaretechnischen Realisierung eines Softwaresystems."
>
> [WAM 98 S.344ff]
>
> Schicht in der WAM-Modellarchitektur:
>
> Eine Schicht organisiert die softwaretechnischen Komponenten einer Modellarchitektur zu einer fachlich und technisch motivierten Entwurfs- und Kontruktionseinheit. Eine Schicht hat selbst keine Schnittstelle und keine Beziehung zu anderen Schichten; Schnittstellen und Beziehungen über Verbindungsstücke haben nur die enthaltenen Komponenten. Die verschiedenen Schichten einer Modellarchitektur sind hierarchisch aufgebaut. Je nach Sichtbarkeit und Verwendungszusammenhang sprechen wir von einer protokollbasierten oder objektorientierten Schicht. Als Komponenten innerhalb einer Schicht verwenden wir Klassenbibliotheken und Rahmenwerke. Die Verbindungsstücke sind durch Konstruktionsmuster, Benutzt-Beziehung oder Vererbung realisiert.

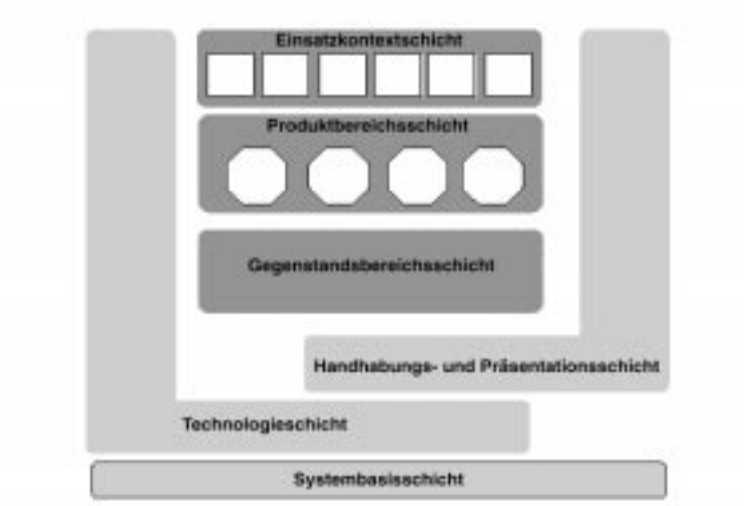


**Abbildung 5-1: Die Schichten der WAM-Modellarchitektur**

Die Schichten repräsentieren verschiedene Partitionen der Modellarchitektur.

Ich gebe einmal die fachliche Definition aus [WAM 98] wieder und fasse die technische Umsetzung aus [WAM 98] zusammen, die mir für meine Umsetzung wichtig erscheinen. Ich erkläre die Schichten von oben nach unten.

**Einsatzkontextschicht:**

**fachlich:**

> "Ein Einsatzkontext umfaßt den konkreten Arbeitszusammenhang, in dem ein Anwendungssystem in einem Unternehmen eingesetzt wird. Dieser Arbeitszusammenhang definiert die Menge der Aufgaben, die mit Hilfe des Anwendungssystems im Einsatzkontext erledigt werden. Dabei können Produkte und Dienstleistungen aus unterschiedlichen Produktbereichen angeboten und bearbeitet werden. Je nach Arbeitszusammenhang kann auch ein Produkt unterschiedlich bereitgestellt werden. Die Loslösung des Einsatzkontextes vom Produktbereich vereinfacht eine kundenorientierte Unternehmensorganisation. Der Einsatzkontext ist eine Modellierungseinheit. „

**technisch:**

Der Einsatzkontext benutzt ausschließlich Komponenten aus der Produktschicht. Sie kann die Komponenten aus den verschiedenen Produktbereichen benutzen, dadurch ist sie sehr flexibel, weil die wesentlichen Werkzeuge und Gegenstände schon vorher realisiert worden sind.

**Produktbereich:**

**fachlich:**

> "Ein Produktbereich ist eine Organisationsform, die nach dem Objektprinzip oder nach einem an Produkten ausgerichteten Prozessprinzip aufgebaut ist. Er ist ein Teil des Anwendungsbereichs, der weitgehend unabhängig vom Rest analysiert, modelliert und konstruiert werden kann. Ein Produktbereich ist eine für die Softwarearchitektur relevante Makrostruktur, die als Modellierungseinheit verwendet werden kann."

**technisch:**

Die Produktbereiche werden technisch durch eine objektorientierte Produktbereichsschicht abgebildet. Sie werden durch ein Black-box-Rahmenwerke abgebildet. Es benutzt die tiefer liegende anwendungsfachliche Schicht Gegenstandsbereich und die technischen Schichten Handhabungs-, Präsentations- und Technologieschicht. Für ein Produktbereich werden aber nicht Komponenten aus den anderen Produktbereichen benutzt. Wenn verschiedene Produktbereiche dieselben Konzepte benutzen, geschieht diese Verbindung über den Gegenstandsbereich.

**Gegenstandsbereich:**

**fachlich:**

> "Der Gegenstandsbereich umfaßt die fachlichen Konzepte, die grundlegend für die unterschiedlichen Produktbereiche sind. Er charakterisiert das Geschäft eines Unternehmens. Der Gegenstandsbereich bildet die fachliche und konstruktive Basis für alle Produktbereiche. Er ist eine Modellierungseinheit."

**technisch:**

Der Gegenstandsbereich bildet die Konzepte der Produktschicht ab, aber sie stellen oftmals nur die abstrakten Klassen der Konzepte dar. Deshalb besteht diese Schicht aus einem White-box-Rahmenwerk, welches dann in der Produktschicht durch Ableitungen realisiert wird. Einzelne Gegenstände können in den Produktbereichen so universell eingesetzt werden, so dass sie bereits in dieser Schicht realisiert werden können.

Die Erläuterungen von Handhabungs-, System- und Technikschicht ist hier nicht weiter relevant, deswegen verweise ich auf die Literatur [WAM 98].

## 5.2 Die fachliche Klassen der Bedienvorgänge

### 5.2.1 konkrete Implementation Attribute und Umgangsformen eines Bedienvorgangs

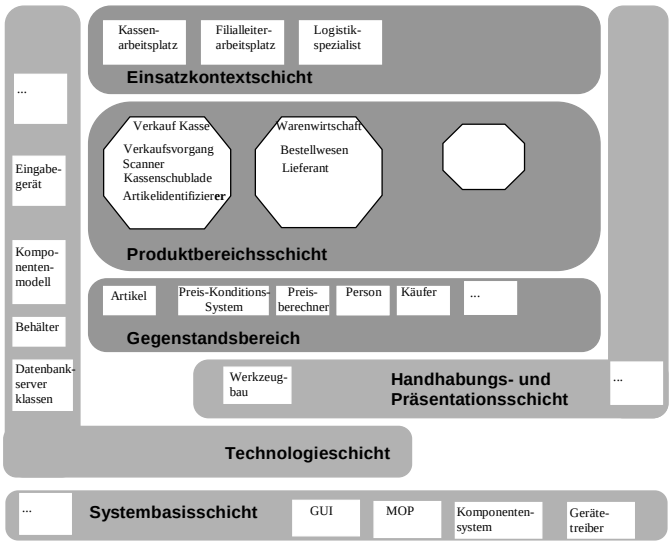


**Abbildung 5-2: Die Kasse Warenwirtschaft Schichtenarchitektur**

In der Abbildung 5-2 habe ich einige Beispiele für die Klassen aus dem Warenwirtschaft- und Kassenbereich eingezeichnet nach der WAM-Modellarchitektur. Für den Produktbereich *Kasse Verkauf* werden aus dem Gegenstandsbereich die Klassen `Artikel` und `Preis-Konditionssystem` und `Preisberechner` benutzt. Diese Klassen werden auch in der Warenwirtschaft benutzt zur Planung und Controlling im Produktbereich *Warenwirtschaft*. Im Produktbereich *Verkauf Kasse* liegen die Klassen `Bedienvorgänge` und die Werkzeuge, die bei den Bedienvorgängen benutzt werden, wie z.B. der `Artikelregistrierer`. In der Einsatzkontextschicht sind die Klassen, die für den konkreten Arbeitsplatz des Kassierers benötigt werden. Das WAM-Schichtenmodell ist für die Architektur einer Firmensoftware konzipiert, für das Kassensystem wird aber angestrebt Kassen für verschiedene Domänen und Firmen zu erstellen, daher gibt es in der Einsatzkontextschicht später verschiedene Kassensysteme, die für die jeweilige Einsatz angepaßt werden. Dafür müssen aus den verschiedenen unteren Schichten die passenden Komponenten zusammengestellt werden. Dies bedeutet, z.B. dass für Kassen unterschiedliche GUI-Builder genommen werden können. Die GUI-Builder werden dann nur in der Systemschicht ausgetauscht. Wenn ein anderes Preiskonditionssystem genommen werden soll, geschieht hier der Austausch in der Gegenstandsschicht. Wenn neue Abläufe für die Kasse dazukommen, werden diese in der Produktschicht erstellt. Wenn ein neues Komponentenmodell genommen werden soll, wird der Austausch in der Systembasis vollzogen, durch die Kapselung des Komponentenmodells in der Technologieschicht bleibt diese Änderung bei den anderen Schichten ohne Auswirkung.

### 5.2.2 Aufbau der Kassenablaufsteuerung

Im Kassensystem gibt es für jede Vorgangsart eine eigene Klasse, siehe Abbildung 5-3 , z.B. für den `Verkauf` oder eine für die `Wechselgeldeinlage`. Für die Steuerung welcher Vorgang wann gestartet wird, gibt es eine eigene Klasse, die Klasse `Kassenvorgänge.` Jeder Vorgang hat ein Muster, mit dem der Vorgang und seine Werkzeuge gesteuert werden. Dieses Muster ist die Ablaufstrategie. Es wird eine andere Ablaufstrategie benutzt, wenn für einen Vorgang der Ablauf anders sein soll.

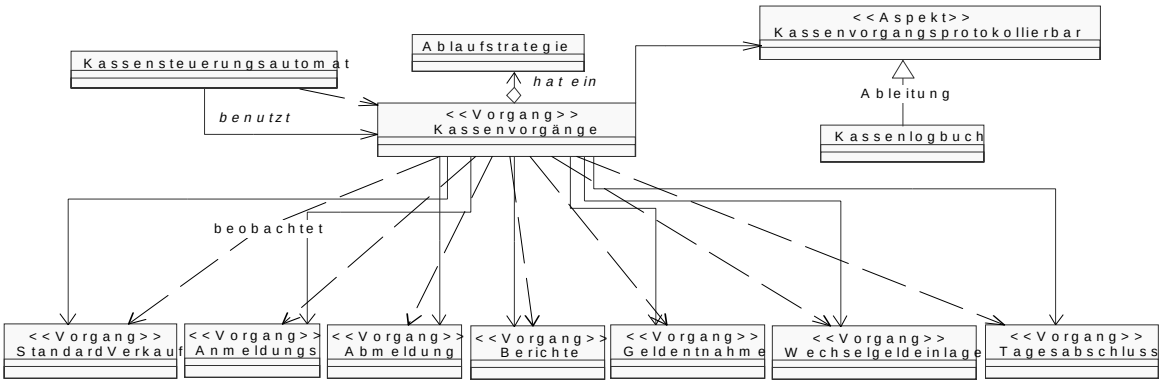


**Abbildung 5-3: Kassenvorgänge**

Die gestrichelte Linie bedeutet, dass die Objekte der jeweiligen Klassen nach dem Beobachter-Entwurfsmuster verbunden sind. Die Klasse auf die der Pfeil der gestrichelten Linie nicht zeigt, ist die Beobachter Klasse, die andere Klasse wird beobachtet und meldet seinen Beobachtern Änderungen. Die anderen Klassenbeziehungen habe ich jeweils einmal exemplarisch in der oberen Abbildung beschriftet.

**Ablaufstrategie für eine Kassenablaufsteuerung**

**fachlich:**

Eine Ablaufstrategie legt die Steuerung eines Bedienvorganges an der Kasse fest. Einzelne Tätigkeiten werden in fachliche Abhängigkeiten gestellt. Für eine Tätigkeit wird festgelegt:

- Welche Werkzeuge benutzt werden sollen
- Wer die Tätigkeit ausführen darf, d.h. welche Rolle der Ausführende haben muss. z.B. Kassierer, Aufsichtskassierer, aber auch der Automat Kasse.
- Vorbedingungen, die erfüllt sein müssen

**technisch:**

Technisch ist die Ablaufstrategie eine Klasse, die die Vorgangsklasse nach dem Strategie-Entwurfsmuster benutzt. Die Vorgangsklasse muss so nicht immer wieder, über Fallunterscheidungen, nach der richtigen Strategie suchen. Außerdem greift die Ablaufstrategie auf die Werkzeugkiste, die kassenspezifischen Werkzeuge enthält, des Vorganges zu, siehe Abbildung 5-3, und ändert den Zustand der Werkzeuge. Die Ablaufstrategie aktiviert und deaktiviert die Werkzeuge des Vorgangs. Wenn ein anderes Ablaufverhalten gewünscht wird, muß eine neue Strategieklasse für den Vorgang gesetzt werden. Die Werkzeuge werden von der Vorgangsklasse beobachtet. Wenn ein Werkzeug eine Tätigkeit beendet hat, meldet es der Vorgangsklasse. Die Vorgangsklasse bearbeitet die Meldung mit Hilfe der Ablaufstrategie.

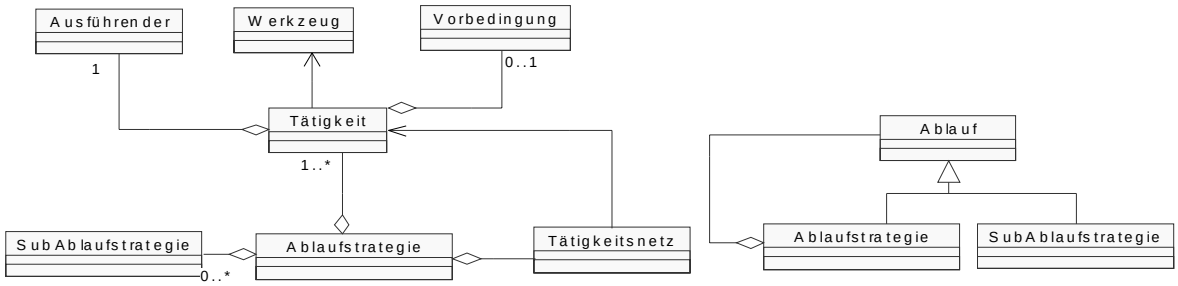


**Abbildung 5-4: Tätigkeit und Ablaufstrategie**

Das statische Klassendiagramm der `Ablaufstrategie` ist in Abbildung 5-4 zu sehen. Die `Ablaufstrategie` enthält `Tätigkeiten` und ein `Tätigkeitsnetz`, das die `Tätigkeiten` miteinander in Beziehung setzt. Außerdem kann eine `Ablaufstrategie` `SubAblaufstrategien` enthalten, die die Subprozesse eines Vorgangs steuern.

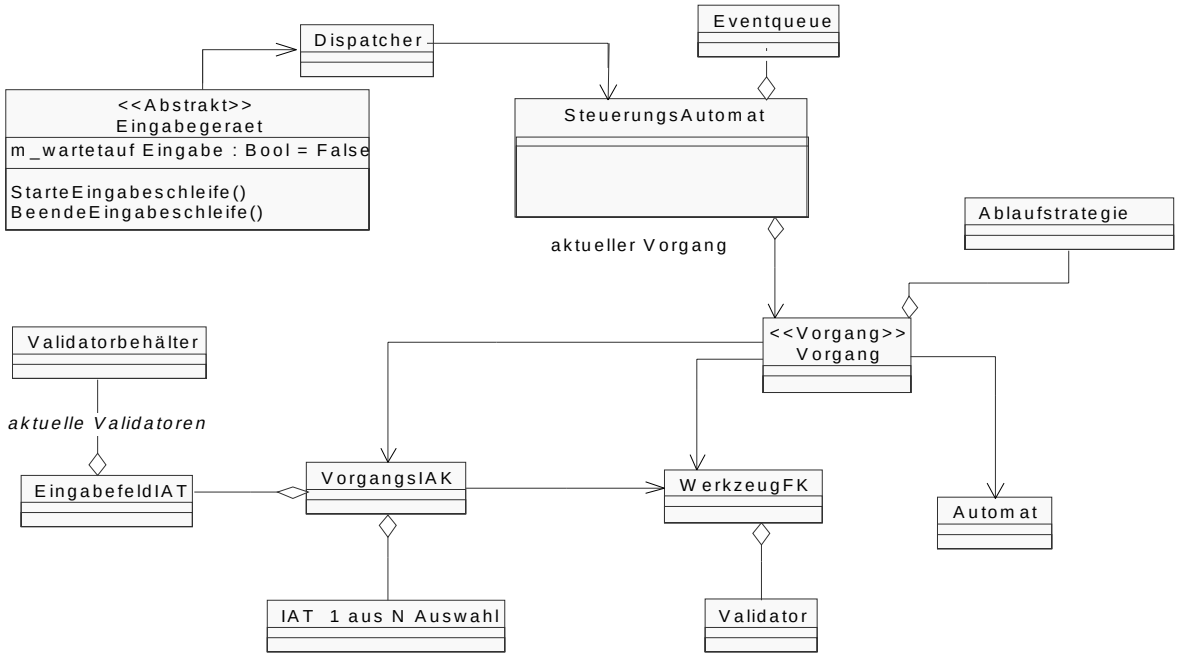


**Abbildung 5-5: Eingabegerät Steuerungsautomat Vorgangswerkzeug**

Steuerung zwischen den Vorgängen.

In der Ablaufstrategie der Kasse wird die Ablaufsteuerung für die Kasse festgelegt. Zum Beispiel, dass nach dem Start der Kasse nur das Werkzeug `Anmeldung` aktiviert wird. Alle anderen Versuche andere Vorgänge zu starten, werden ignoriert, bzw. es wird mit einer Fehlermeldung angezeigt, dass erst eine Bedieneranmeldung erfolgen muss. Ein Vorgang kann nur gestartet werden, wenn er aktiviert worden ist.

Das Kassenvorgangswerkzeug beobachtet die Vorgangsautomaten. Wenn ein Automat mit der Bearbeitung fertig ist, meldet es seinem Beobachter, dass der Vorgang beendet ist. Das Kassenvorgangswerkzeug prüft welche Vorgänge jetzt erlaubt sind, bzw. welche Vorgänge jetzt durchgeführt werden müssen. Wenn z.B. die Bedieneranmeldung nach dem Kassenstart am Tagesanfang erfolgreich durchgeführt wurde, soll automatisch die Wechselgeldeinlage gestartet werden. Erst wenn dieser meldet, dass der Vorgang erfolgreich durchgeführt wurde, werden andere Automaten, bzw. Werkzeuge aktiv geschaltet. Aktiv geschaltet bedeutet, dass der Automat gestartet werden darf.

Der Automat Kassensteuerung steuert die Benutzung der Werkzeuge während des Vorgänge.

Für den Start der Verkaufsautomaten gibt es verschiedene Möglichkeiten. Zum einen gibt es die Events von den Eingabegeräten mit Daten (Datenevents), zum anderen die Events(Funktionsevents), die explizit einen Vorgang starten und nur eventuell Daten enthalten. Es kann noch zwischen verschiedenen Eingabegeräten unterschieden werden. Es gibt Eingabegeräte, die immer eindeutige Events versenden, so z.B. der Scanner, der immer nur das Event "EAN-Code" liefert, das als Datum den EAN-Code enthält; zudem gibt es Eingabegeräte, bei denen erst durch den Kontext klar ist, welche semantische Bedeutung die Daten des Datenevents haben, dafür ist die Tastatur ein Beispiel.

Wenn an der Kasse alle nötigen Vorgänge durchführt worden sind, ist sie bereit zum Verkauf. Das bedeutet der Automat Verkauf wird gestartet. Der Verkaufsautomat wird mit seinen Parametern initialisiert. Er erhält einen Laufzettel und die Werkzeuge, die beim Start des Vorganges erlaubt sind, werden aktiviert. Wenn kein Verkaufsvorgang in Arbeit ist, dürfen auch andere Vorgänge gestartet werden. Diese Vorgänge werden durch entsprechende Funktionsevents gestartet. Der Verkaufsautomat erhält diese Events und kann sie nicht verarbeiten. Das Event ist z.B. die Taste "Code" wurde gedrückt. Für dieses Event hat kein aktives Werkzeug des Verkaufs eine Behandlung. Deshalb wird dem Kassenvorgangswerkzeug gemeldet, dass ein Event unbekannt ist. Dieser holt sich das Event vom Vorgangsautomaten und prüft ob er eine Behandlung für das Event hat. Wenn er eine Behandlung hat, wie z.B. ein Codeprogramm Geldentnahme zu starten, führt er die Behandlung durch, andernfalls wird das Event als Fehler behandelt.

Als Beispiel, wie ein Bedienvorgang gesteuert wird, beschreibe ich den Vorgang Standardverkauf.

**Standardverkauf**

Der Bedienvorgang Standardverkauf hat vier verschiedene Phasen.

**Die Startphase:**

In dieser Phase darf eine Artikelregistrierung durchgeführt werden. Nach einer Artikelregistrierung wird dieser Zustand verlassen. Wenn die Artikelregistrierung erfolgreich durchgeführt wurde, wird der Bonkopf und die Artikelregistrierungszeile gedruckt. Diese Phase ist damit beendet.

Der Standardverkauf ist dann in der Phase Verkauf.

**Die Verkaufsphase:**

In dieser Phase können, je nach Konfiguration, folgenden Bedienvorgänge ausgeführt werden:

Artikelregistrierung, Sofortstorno, Zeilenstorno, Kundenidentifizierung, Zeilenrabatt und Zwischensumme. Dies bedeutet,dass für alle diese Tätigkeiten Werkzeuge zur Verfügung stehen. Alle diese Werkzeuge sind aktiv. Das Werkzeug Zeilenstorno darf nur vom Aufsichtskassierer benutzt werden. Alle anderen Werkzeuge dürfen vom Kassierer benutzt werden. Der Verkaufszustand wird verlassen wenn die Endsumme berechnet wurde, danach steht die Kasse im Zahlungsausgleich.

**Die Zahlungsausgleichsphase:**

Es können jetzt keine Artikelregistrierung mehr vorgenommen werden. Je nach Konfiguration des Zahlungsausgleich kann mit verschiedenen Geldmitteln bezahlt werden. Es stehen die für die Verkaufsart und Kunden zugelassene Zahlungswerkzeuge zur Verfügung. Der Bon kann in dieser Phase ganz abgebrochen werden. Wenn die Zahlungssumme ausgeglichen ist, wird die Phase Zahlungsausgleich verlassen.

**Die Phase Bonende:**

Der Verkauf ist jetzt abgeschlossen und je nach Konfiguration können verschiedene Funktionen ausgeführt werden. Es kann z.B. der ganze Kassenbon ausgedruckt werden oder nur das Bonende. Zusätzlich kann der Journalbon gedruckt werden. Die Bondaten werden persistent gespeichert. Wenn alle Funktionen durchgeführt worden sind, ist der Vorgang ganz abgeschlossen. Die Tätigkeiten in der Bonendenphase werden alle automatisch vom Automaten gestartet.

Nach der Phase Bonende steht der Verkaufsautomat wieder in der Startphase.

Der Standardverkauf hat als Werkzeugkomponenten Artikelregistrierung, Sofortstorno, Zeilenstorno, Personenidentifizierung, Zeilenrabatt, Zwischensumme, Bonabbruch und Zahlung.

Der Standardverkauf liegt als Black-box-Rahmenwerk in der Produktschicht. Für die Einsatzkontextschicht werden die Black Boxen durch Komponenten gefüllt.

Für den Standardverkauf hat die Ablaufstrategie vier verschiedene Subablaufstrategien für die Phasen Start, Verkauf, Zahlung und Bonende.

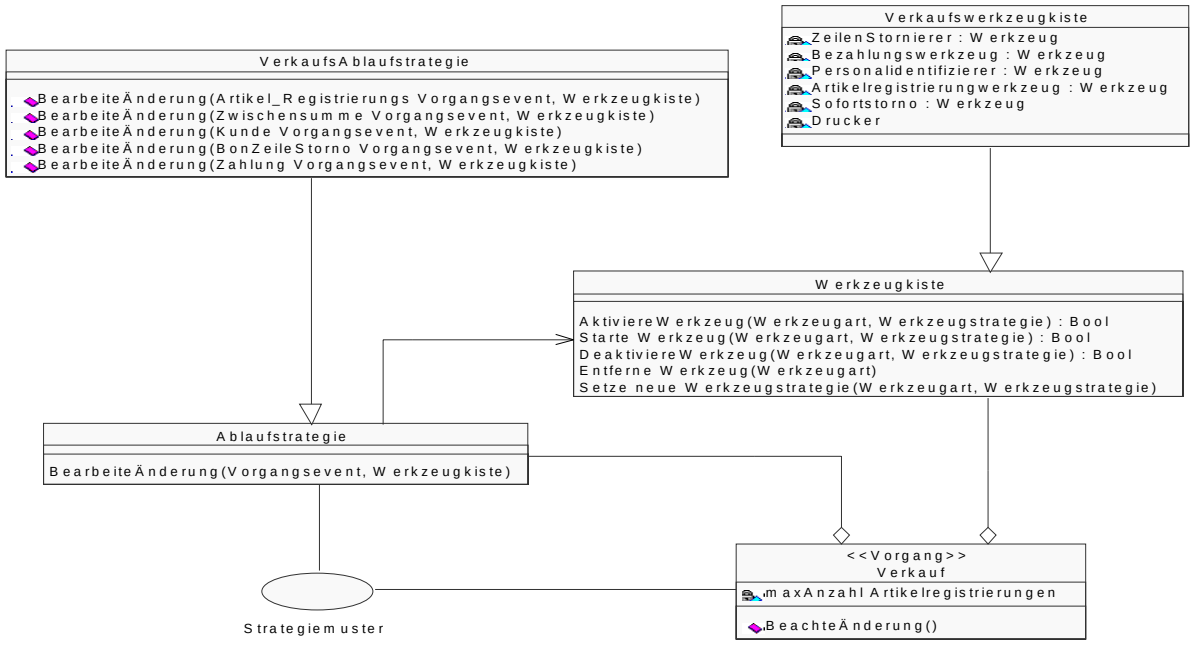


**Abbildung 5-6: Klassendiagramm** `VerkaufslaufzettelStrategie`

Die Steuerung des Verkaufsvorgang erfolgt nach dem gleichen Muster wie die Steuerung der Kassenbedienvorgänge. Der Verkaufsvorgang benutzt die Klasse `Verkaufslaufzettel`, um seine Werkzeuge zu steuern. Die Schnittstelle wird durch Events erweitert, die die Werkzeuge der Verkaufsklasse senden. Als Beispiel dient das Artikelregistrierungswerkzeug, wenn das Werkzeug einen Artikel registriert hat, meldet es dem Verkaufsvorgang dies über die Methode: `BeachteÄnderung(Vorgangsevent)`, wobei als Parameter die Klasse `Artikel_Registrierungs_Vorgangsevent` mit übergeben wurde.

Die Klasse Verkauf verwaltet alle Daten, die beim `Verkauf` anfallen, indem alle Daten im Bon protokolliert werden. Wenn ein anderes Ablaufverhalten gewünscht wird, muß eine neue Strategieklasse für den Verkauf gewählt werden. Die Klasse `Verkaufslaufzettel` legt fest welche Werkzeuge aktiv sind.

Im nächsten Abschnitt erläutere ich für die Artikelregistrierung, wie ein Ablaufverhalten während der Laufzeit geändert werden kann.

### 5.2.3 Artikelregistrierung mit Prozessänderung

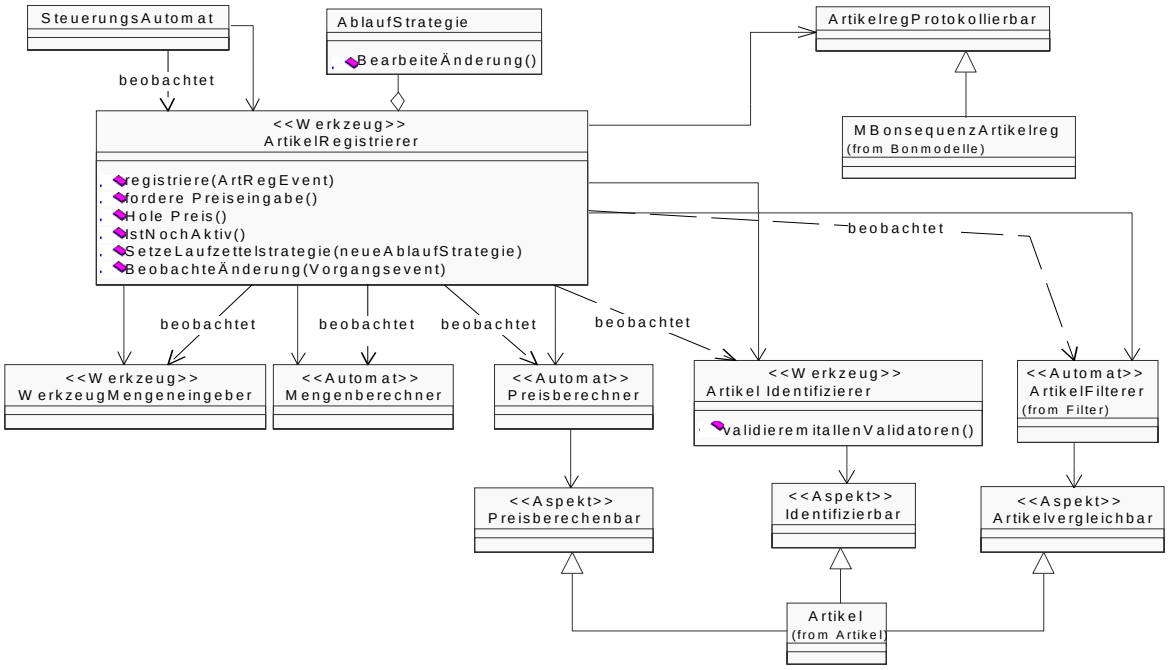


**Abbildung 5-7: Artikelregistrierer**

Das Werkzeug Artikelregistrierer benutzt verschiedene Subwerkzeuge, die über die Ablaufstrategie koordiniert werden. In der Abbildung 5-7 sind ein paar Werkzeuge aufgezeichnet. Zur Artikelidentifizierung benutzt es z.B. das Werkzeug Artikelidentifizierer. Die Werkzeuge arbeiten, wie es im WAM-Ansatz üblich ist, über einen Aspekt auf dem Material. Der Artikelregistrierer beobachtet seine Subwerkzeuge. Dadurch ist das Werkzeug lose mit den Subwerkzeugen gekoppelt.

Die Artikelregistrierung in einer Kasse ist so eingestellt, dass die Mengeneingabe optional vor der Artikelidentifizierung erfolgt. Wenn das Artikelidentifizierungswerkzeug einen Artikel identifiziert hat, wird automatisch der Preisberechner gestartet. Siehe Abbildung 5-8.

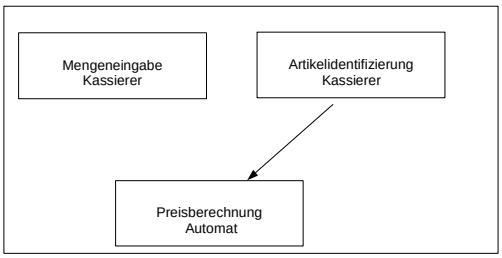


**Abbildung 5-8: Ablaufstrategie Artikelregistrierung Normalfall**

Es gibt jedoch Sonderfälle bei denen dieser Weg nicht ausreicht, dies ist der Fall, wenn nach der Artikelregistrierung noch Mengen oder Maße eingegeben werden müssen.

**Beispiel:**

Die Kassiererin will eine Melone registrieren. Weil sie denkt, dass die Melone nach Stückpreis verkauft wird, gibt sie nur die Artikelkurzbezeichnung ein. Jetzt wird in der Kasse nach dieser Ablaufstrategie der Preisberechner gestartet. Der Preis kann jedoch nicht berechnet werden, weil das Gewicht fehlt.

Für diesen Fall muss eine Lösung gefunden werden. Wenn ein Mensch diesen Vorgang steuern würde, würde er in diesem Fall angeben, dass die Mengeneingabe zwingend vor der Preiseingabe erfolgen muss. Er würde also eine neue Strategie benutzen, um das Problem zu lösen. Er hat jetzt mehr Parameter, um eine passende zu finden. Er wählt aus einem Repertoire von Möglichkeiten aus.

Hier wird eine Änderung des Musters durchgeführt. Beim Laufzettel im Kapitel 4.4.1 hat der Benutzer diese Änderung durchgeführt. In der Kasse wird diese Änderung durch die Verkaufsklasse durchgeführt.

Der `Artikelregistrierer` hat die Strategie, wie in Abbildung 5-8 beschrieben, nach der Artikelidentifizierung die Preisberechnung durchzuführen. Die wird gestartet. Der `Preisberechner` kann jedoch den Preis nicht berechnen, weil ihm dazu das Gewicht des Artikels fehlt. Er meldet das seinem Beobachter `Artikelregistrierer`, dass der Preis nicht berechenbar ist. Der `Artikelregistrierer` verarbeitet die Nachricht und sucht eine neue Strategie die zum Problem besser paßt. Er hat jetzt für die neue Strategie eine genauere Spezifikation, er weiß den genauen Typ des Artikels. Er benutzt eine Artikelregistrier-Strategie-Fabrik um eine neue Strategie zu suchen. Er übergibt der Fabrik den Artikel als Parameter. Die Fabrik liefert ein Strategie für einen Gewichtsartikel. Der Artikelregistrierer prüft anhand seiner Bonsequenz, welche Tätigkeiten von der neuen Strategie schon "abgehakt" werden können. Ein Artikel wurde schon identifiziert, jetzt muss vor der Preisberechnung die Mengeneingabe mit der Strategie Gewicht erfolgen. Siehe Abbildung 5-9. Der `Artikelregistrierer` startet das Werkzeug `Gewichtseingabe`. Das Gewicht wurde errechnet, siehe Abbildung 5-10. Der Preisberechner startet jetzt und kann den Preis für die Melone berechnen. Die Artikelregistrierung für diesen Artikel ist abgeschlossen.

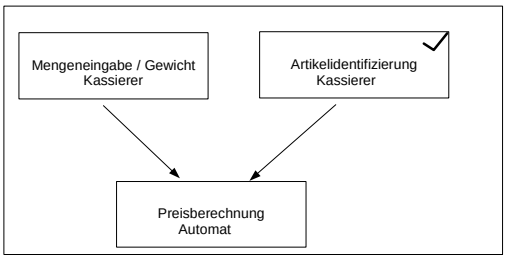


**Abbildung 5-9: Ablaufstrategie Artikelregistrierung: Artikel mit Maßeingabe**

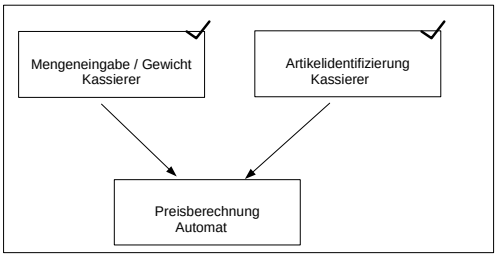


**Abbildung 5-10: Ablaufstrategie Artikelregistrierung nach der Gewichtseingabe**

## 5.3 Neue fachliche Anforderungen

Um zu zeigen welche Änderungen im Kassensystem notwendig sind, wenn eine neue Anforderung vom Kunden kommt, werde ich zwei Anforderungen aufzeigen und die Änderungen beschreiben, die im Kassensystem dafür vorgenommen werden müssen.

### Kaffeescheck-Verkauf

Die erste Änderung betrifft den Prozess. Eine Handelskette möchte einen neuen Verkaufsvorgang für ihr Kassensystem aufnehmen, den Kaffeescheck-Verkauf. Die Handelskette gibt ihren Mitarbeitern bargeldlos Gutscheine aus, mit denen sie Artikel aus der Warengruppe Röstkaffee kaufen können, ein 500 g Paket Kaffee einer beliebigen Sorte. Der Einkauf mit diesem Gutschein soll so erfolgen, dass während des Verkaufsvorgangs nur ein Artikel aus der Warengruppe gekauft werden kann. Während des Verkaufsvorgangs darf kein Zeilenstorno durchgeführt werden, als Abbruchmöglichkeit des Vorganges gibt es nur den Bonabbruch nach der Endsumme. Die Bezahlung für diesen Verkaufsvorgang ist nur mit dem Kaffeescheck erlaubt. Der Kaffeescheck wird bedruckt und damit entwertet.

### Chipleser

Eine Handelskette, die zur Zeiterfassung einen berührungslosen Chipleser benutzt, möchte jetzt ein Lesegerät für die Chipkarte an der Kasse zur Personalidentifizierung für den Verkauf an das Personal benutzen. Zusätzlich soll es zur Anmeldung der Kassierer an der Kasse benutzt werden.

### 5.3.1 Anpassungen für den Kaffeescheck-Verkauf

Für den Kaffeescheck-Verkauf müssen die folgenden Änderungen gegenüber dem normalen Verkauf durchgeführt werden. Der Menge der registrierten Artikel pro Bon wird auf eins beschränkt. Das Sofortstorno-Werkzeug und Zeilenstorno-Werkzeug dürfen nicht aktiv sein und als Bezahlungswerkzeug ist nur die Bezahlung mit einem Warengutschein zulässig. Die Standardbezahlungsarten wie Bar- und Scheckzahlung dürfen nicht benutzt werden. Außerdem muß bei der Artikelregistrierung geprüft werden, ob der Artikel aus der erlaubten Warengruppe ist.

**Artikelregistrierung**

Bei der Artikelregistrierung muss ein Filter gesetzt werden. Der Filter wird von der allgemeinen Klasse Filterstrategie abgeleitet. Er vergleicht ob ein Artikel in eine Warengruppe paßt. Der normale Standardverkauf hat einen Filter der nichts tut, d.h. er läßt alle Artikel durch. Die Filterklasse ist nach dem Strategiemuster von [Gamma et al. 96] konstruiert.

**Bezahlung**

Die Bezahlung geschieht über ein Zahlungswerkzeug. Es gibt schon ein Warengutschein-zahlungswerkzeug, das als Zahlungsmittel den Warengutschein verarbeitet. Die Bezahlung mit Warengutschein stellt eine Besonderheit gegenüber der Bezahlung mit Geldmitteln dar, weil nicht Geldmittel, sondern Waren miteinander verrechnet werden. Die Sicht auf das Material ist eine andere. Zur Berechnung der neuen Ausgleichssumme muss das Werkzeug nicht nur vom Verkaufsvorgang die Endsumme wissen, sondern außerdem die Artikel, die verkauft wurden. Der Geldwert des Warengutscheins ergibt sich aus dem Artikel, der verkauft wurde. Bei der Berechnung muss also nicht erst die Geldsumme, sondern der Warenwert ermittelt werden. Daraus kann die Berechnung der Zahlungssumme des Gutscheins erfolgen.

Es muss im Bon der Artikel herausgesucht werden, der auf dem Gutschein angegeben wird. Vorher soll geprüft werden, ob dieser Artikel im Verkaufsvorgang verkauft wurde. Dieses Werkzeug benötigt also neue Umgangsformen des Materials Bon. "Ist Artikel im Bon(Artikelfilter)" und "Gib Preis vom Artikel(Artikel)".

Die Parametrisierung erfolgt über den Warengutschein. Der Warengutschein wird in der Warenwirtschaft angelegt und hat den Warenwert von einem Artikel aus der Warengruppe Röstkaffee. Der Geldwert des Gutscheins kann erst errechnet werden, wenn für diesen Gutschein ein Artikel gekauft wurde. Wenn die Handelskette möchte, dass für ein Kaffeescheckgutschein nur ein halbes Paket Kaffee gekauft werden kann, so kann dies über die Warengutscheinpflege geändert werden. Für den Fall wären dann zwei Kaffeegutscheine nötig, um ein 500g Paket Kaffee zu kaufen. Die Klassenstruktur muß dafür nicht geändert werden.

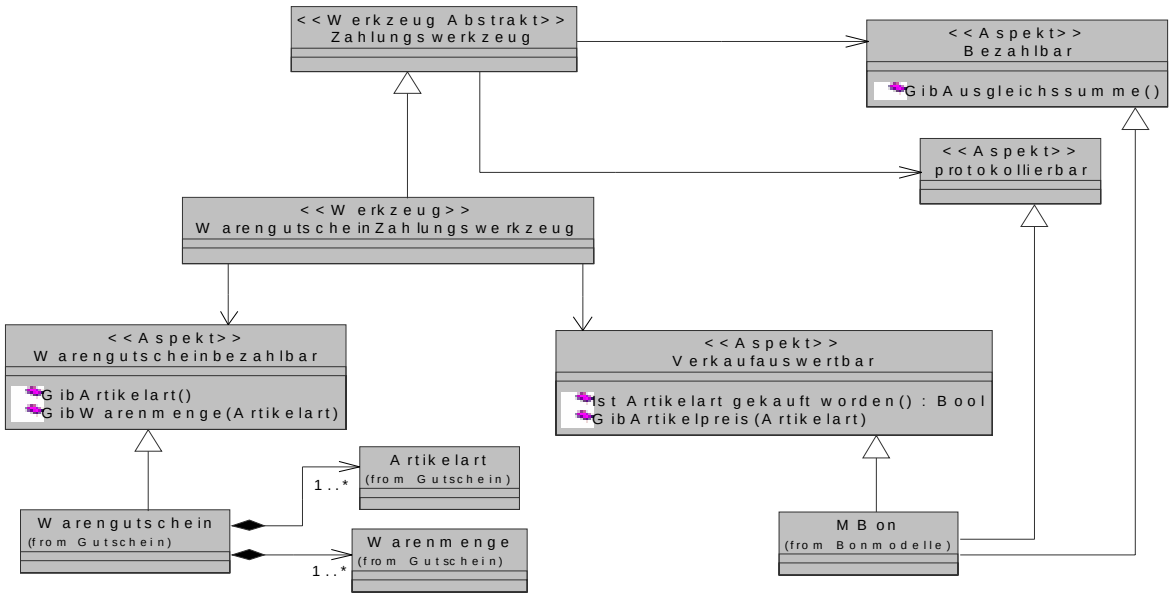


**Abbildung 5-11: Werkzeug "`Warengutschein-Zahlung`"**

**Phase Start:**

- Werkzeug Artikelregistrierung aktivieren
- Standard-Artikelregistrierungwerkzeug eduScho mit
    - Parameter Filterkomponente "Artikel aus Warengruppe Röstkaffee"

Nach erfolgreicher Artikelregistrierung

- Werkzeug Drucker starten
    - Parameter Druckformular Bonkopf Standard eduScho
    - Parameter Druckformular Bonsequenz Verkaufsart "Kaffeescheck"
    - Parameter Druckformular Bonsequenz Artikelregistrierung 1. Artikelregistrierung vom diesem Bon

Phase beenden, wenn Artikelregistrierung erfolgreich beendet ist.

**Phase Verkauf:**

- Summierer automatisch starten
    - Vorbedingung Maximale Anzahl von Artikelregistrierung erreicht
    - Summierer Standard eduScho
- Standard-Artikelregistrierungwerkzeug eduScho mit
    - Vorbedingung Maximale Anzahl von Artikelregistrierung noch nicht erreicht
    - Parameter Filterkomponente "Artikel aus Warengruppe Röstkaffee"

Phase beenden, wenn Summe auf dem Bon vorhanden

**Phase Zahlungsausgleich:**

Werkzeug Bonabbruch

Zahlungswerkzeug "Zahlung mit Warengutschein (1 Artikel aus Warengruppe Röstkaffee)"

Phase beendet, wenn Zahlungssumme ausgeglichen oder Bonabbruch

**Phase Bonende:**

- Werkzeug Drucker starten
    - Parameter Druckformular "Bonende Standard"
- Werkzeug JournalDrucker starten

Parameter Druckformular "Kaffeescheckbon"

Bei dem Vorgang `Kaffeescheck-Verkauf` wird die Anzahl der Artikelregistrierungen auf eins gesetzt. Beim `Kaffeescheck-Verkauf` wird zur Aufnahme einer neuen Strategie die Klasse `KaffeescheckAblaufstrategie` von der Klasse `VerkaufsAblaufstrategie` abgeleitet. In der Klasse werden die Methoden der Klasse überschrieben. Wie z.B. die Methode `BearbeiteÄnderung( ArtikelRegistrierungsVorgangsevent, Werkzeugkiste)`

Die Ablaufstrategie für die Startphase bleibt im wesentlichen gleich, nur das Layout des Bonausdrucks wird gegenüber dem Standardverkauf verändert. Es kommt zwischen dem Bonkopf und den Artikelregistrierungen die Zeile Kaffeescheckverkauf.

In der Phase Verkauf wird die Artikelregistrierung zwar zugelassen, aber dadurch anders als beim Standardverkauf wird die Summenbildung automatisch gestartet. Die Summenbildung wird gestartet, wenn die maximale Anzahl von Artikelregistrierungen erreicht wurde. Auf diese Weise kann der Ablauf einfach geändert werden, wenn die Handelskette einen Gutschein herausgibt, mit dem man zwei Artikel kaufen darf.

Die Zahlungssumme ist ausgeglichen wenn der Gutschein akzeptiert wird.

**Bonlayout**

Der Bon hat bei den meisten Zeilen das Standardlayout der Filialkette, nur nach dem Bonkopf wird KAFFEESCHECK gedruckt. Der Bezahlungstypkürzel ist KS , dies ist die Bezahlungs-artkurzbezeichnung für die eine Feld auf der Bonzeilenformular Gutscheinzahlung vorgesehen ist.

```
              E D U S C H O
        Bremer Str. 108, 4000 Musterstadt
          Frischer Kaffee , Frische Ideen
-------------------------------------
# 0150  1 171713 452023 21.01.99 17:29
-------------------------------------
          KAFFEESCHECK
                                    DM
1 *   7,49
Wiener Gold  500 g              7,49 B
                               ---------
SUMME                           7,49
                               =========
GEGEBEN KS                      7,49
RÜCKGELD DM                     0,00
B: 7.00% MWST auf 7,00 DM  =    0,49

*************************************
        Umtausch nur gegen Bon
   Vielen Dank für Ihren Einkauf
```

**Abbildung 5-12: Bon von einem Kaffeescheck-Verkauf**

### 5.3.2 Anpassung für ein neues Gerät


**Abbildung 5-13: Grafische Darstellung von einem KABA Benzig Chipleser**

Wie schon in der Anforderung beschrieben, soll ein neuer Gerätetyp an der Kasse angeschlossen werden. Der Gegenstand "Terminal berührungsloser Chipleser" (im folgendem Chipleser genannt ) wird in einem softwaretechnischen Gegenstand abgebildet (Klasse "`Personenkartenlesegerät`"). Diese Klasse definiert eine allgemeine Schnittstelle für Chipleser. Die gerätespezifischen Eigenschaften werden in abgeleiteten Klassen gekapselt. Der Chipleser kann verschiedene Funktionen durchführen. Es kann z.B. prüfen, ob die Person berechtigt ist, sich zu Zeiterfassungszwecken anzumelden. Das Gerät verfügt über fünf verschiedene Berechtigungsprüfungen, die durch die fünf verschiedenen Buttons ausgewählt werden können. Siehe die grafische Darstellung des Gerätes in Abbildung 5-13. Das Gerät erhält über eine Datei die benötigen Parameter für den Ablauf. In der Datei sind Personendaten und Berechtigungsstufen enthalten. Dadurch kann das Gerät selbständig prüfen, ob ein Benutzer berechtigt ist die ausgewählte Funktion durchzuführen. Wenn ein Benutzer sich z.B. angemelden will, drückt er den entsprechenden Funktionsbutton und hält seine Chipkarte vor das Gerät. Wenn die Anmeldung erfolgreich war, ertönt ein Piepton. Nach dieser Funktion wird ein Datensatz in eine Datei geschrieben. In diesem Datensatz wird die ausgeführte Funktion, die Uhrzeit und die Daten des Chips festgehalten.

Ein Beispiel:

Hier ist ein Datensatz wie er aus dem Shared Memory ausgelesen worden ist:

’AA 0 B2 3 99 01 27 215134 0 20000008 000024’

- AA ist die Terminalidentifikation
- 0 ist der momentane Betriebszustand (off-Line)
- B2 ist die Kennung welche Aktion am Terminal vorlag, in diesem Fall Anmeldung für Personalverkauf
- 3 ist die Kennung, dass Sekunden genau protokolliert wird
- 990127 ist das Datum 27.01.99
- 211513 ist die Zeit(mit Sekunden) entspricht 21:15:13
- 0 ist ein Fehlercode (alles OK)
- 20000008 ist die Firmennummer
- 0 wird nicht benutzt
- 00024 ist die Nummer des Chips der Angestellten Margot Meier

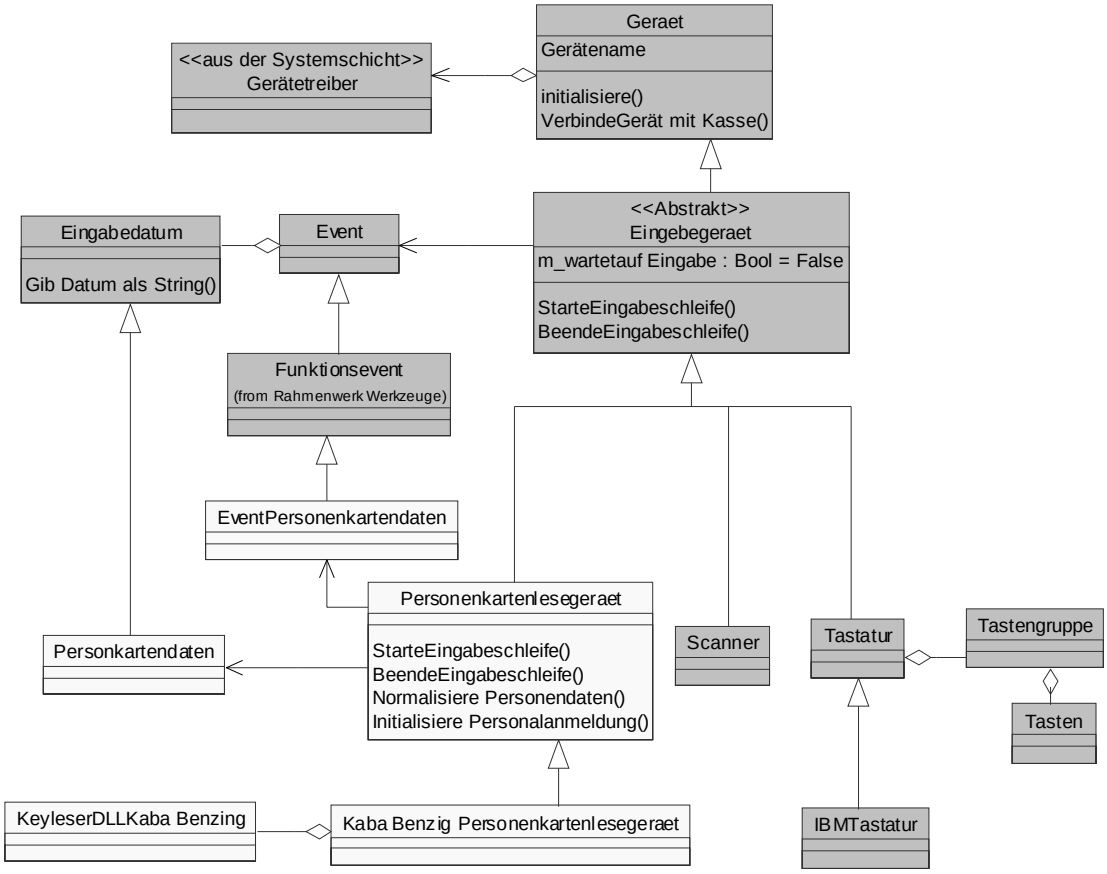


**Abbildung 5-14: Ableitungshierarchie für einen Chipleser**

In der Abbildung 5-15 werden die Klassen, die auf der Werkzeugseite verändert oder neu sind, aufgelistet. Die unveränderten Klassen haben einen grauen Hintergrund.

Für die Anbindung des Geräts an das Kassensystem, ist es nötig, dass das Kassensystem immer wieder prüft, ob Daten vom Gerät vorliegen. Für Eingabegeräte, bei denen gepollt werden muss, ob das Gerät Daten geliefert hat, gibt es schon eine Klasse im Kassensystem, die Klasse `Eingabeview`. Ich ändere die Bezeichnung dieser Klasse in `Eingabegerät`, weil sie als Gegenstand einem Eingabegerät entspricht. Daher wird die Klasse `Personen-kartenlesegeraet` von der Klasse `Eingabegerät` abgeleitet. Für das konkrete Gerät von Kaba-Benzig wird eine Klasse konstruiert, die von der Klasse `Personenkarten-lesegeraet` abgeleitet wird. Im laufenden Kassensystem pollt ein Objekt der Klasse `Kaba_Benzig_Personenkartenlesegeraet` in einer Endlosschleife, ob Daten in die Datei geschrieben wurden. Wenn Daten vorliegen, wird eine Methode dieser Klasse aufgerufen, die die Daten des Kartenlesers in ein kassenspezifisches Personendatenformat überführt. Abhängig von der gewählten Funktion des Gerätes wird ein Event erzeugt, das die Eingabedaten enthält. Dies wäre wenn z.B. wenn sich Personal zum Einkauf identifiziert hat, das Event `geprüfte Personalanmeldung erfolgt`. Dies Event wird an den Steuerungsautomaten weitergeleitet. Der Steuerungsautomat fügt das Event in die Eventschlange des Kassensystems ein. Wenn das Event dann verarbeitet werden soll, wird es vom aktuellen Werkzeug verarbeitet. Hier muss für den Chipleser eine Erweiterung des Kassensystems erfolgen. Auf der Werkzeugseite muß es eine Klasse geben, die das Event verstehen und verarbeiten kann. Dies ist in diesem Fall die Klasse Personalidentifizierung. Diese Klasse kann die Daten vom Event verarbeiten. Sie prüft über die Metadateninformation der Eingabedaten-Klasse, ob sie die Klasse Personaldaten enthält und downcastet die Eventklasse. Das Klassendiagramm dazu wird in Abbildung 5-14 dargestellt. Die vorhandenen Klassen sind grau gefüllt, die neuen Klassen sind hell gefüllt.

Die Klasse "`Kaba_Benzig_Personenkartenlesegerät`" kapselt die Kaba Benzig typischen Eigenschaften und Umgangsformen. Das bedeutet in der Klasse werden die Methoden der Oberklassen überschrieben und die gerätespezifischen Methoden aufgerufen, um auf die Daten zuzugreifen. Die Personenkartendaten werden in ein Kassensystem einheitliches Format gebracht. Für die verschiedenen Funktionen des Gerätes werden entsprechende neue Funktionseventtypen neu angelegt. Wenn jemand z.B. aus dem Personal etwas kaufen möchte und sich über den Chipleser über den ersten Button  identifiziert hat, wird das Event `geprüfte_Personalanmeldung_erfolgt` erzeugt. Das Event enthält dann die Daten der Chipidentifizierung. Bei einer Anmeldung der Kassiererin an das System wählt sie den Button  , für diese Funktion wird das Event `geprüfte_An-meldung_Kassiererin_`erfolgt in der Klasse `Personenkartenlesegerät` erzeugt.

Zur weiteren Verarbeitung des Events, muß auf der Werkzeugseite eine Erweiterung erfolgen. Diese wird in der Klasse `Personenidentifizierer-Funktionskomponente` vorgenommen. Die Funktionskomponente erhält einen weiteren Validator, um die Daten zu prüfen. Die Daten der Personenidentifizierung sind ja schon durch den Chipleser geprüft worden, daher muss hier eine andere Prüfung erfolgen als bei der Anmeldung der Kassiererin über die Tastatur.

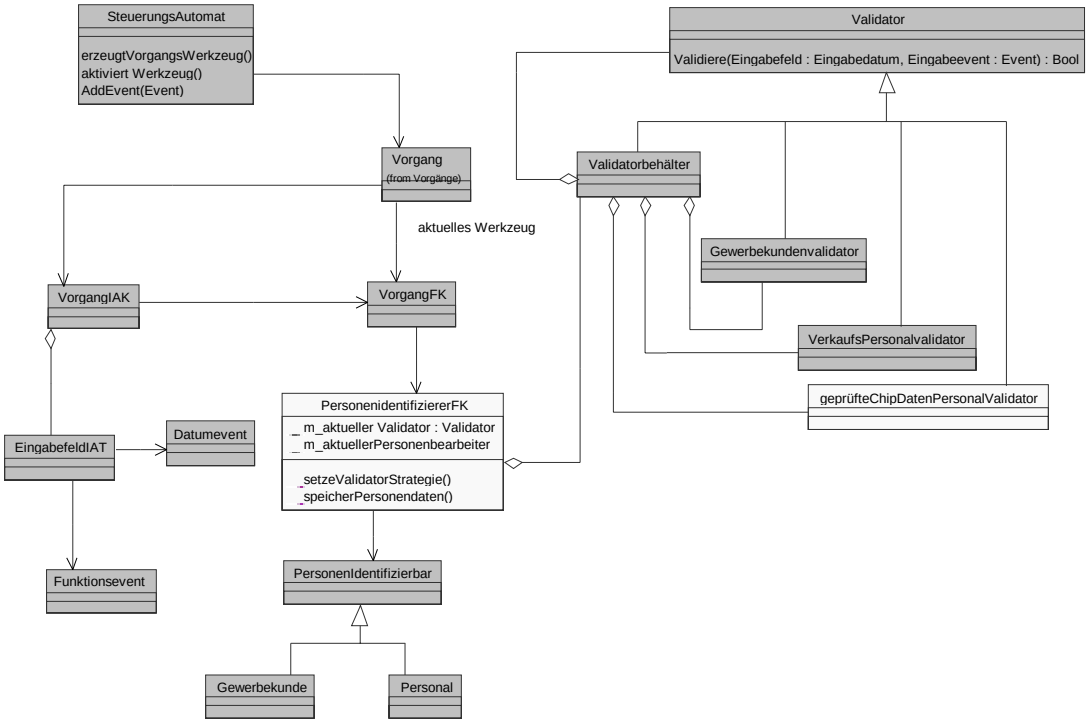


**Abbildung 5-15: Personalidentifizierer mit neuer Strategie Personaldaten**

Die Eingabe der Daten bei den meisten Vorgängen der Kasse erfolgt auf folgende Weise. Die **I**ntera**k**tions**k**omponente (IAK) hat im Normalfall für die Eingabe ein Eingabefeld. Es ist für das Werkzeug bei Eingabe noch nicht bekannt, welcher Fachwert in diesem Eingabefeld eingegeben wird. Es ist nur bekannt, dass es eine Zahl ist. Die Eingabeevents kann man in zwei Arten unterscheiden, einmal in Datenevents und in Funktionsevents. Die Datenevents kennzeichnen, dass Daten eingegeben wurden, es ist aber noch keine Funktion damit verbunden. Das Funktionsevent kann Daten enthalten, es ist aber nicht zwingend erforderlich. Die Daten des Datenevents werden in dem Eingabefeld der IAK gesammelt bis ein Funktionsevent kommt. Wenn das Funktionsevent kommt, werden die Daten des Eingabefeldes an die Funktionskomponente übergeben. Die **F**unktions**k**omponente (FK) des Werkzeuges prüft dann anhand seiner Sub-Werkzeuge welches die Eingaben verarbeitet kann. Zur Prüfung ob das Event von dem Werkzeug verarbeitet werden kann, verwendet das Sub-Werkzeug die Methode `Validiere`. Die Methode `Validiere` ist nach dem Strategie-Muster von [Gamma et al. 96] konstruiert. Dadurch das dies Muster benutzt wird, können unterschiedliche Strategien in einem Werkzeug benutzt werden, um die Eingabedaten zu prüfen. Für die Anmeldung des Personals über die Chipkarte wird ein neuer Validator benötigt. Dieser Validator kennt das Event "geprüfte Anmeldung Kassiererin erfolgt" und kann die Daten aus dem Datenobjekt des Funktionsevents verarbeiten.

### 5.3.3 Eine fachliche Änderung verändert nicht den gesamten Ablauf im Rahmenwerk

Der Ablauf wird in den Oberklassen festgelegt, der Kontrollfluß wird durch sie bestimmt. Einen anderen Ablauf wird durch das Überschreiben von Einschubmethoden und durch die Ausnutzung von Polymorphismus erreicht.

Die Benutzung von anderen Eventklassen beim Methodenaufruf, wie z.B. bei der Bezahlung ändert zudem nicht den Ablauf. Die Änderung des Ablaufs wird durch die Ausnutzung des Strategiemusters bewirkt. Die Ereignisse, auf die reagiert wird, sind dieselben wie beim Standardverkauf, der Unterschied zwischen den beiden Vorgängen liegt in Behandlung der Werkzeuge, die der Verkauf zur Verfügung hat. Beim Standardverkauf werden andere Subklassen von Werkzeuge aktiviert als beim Kaffeescheck-Verkauf. Die Beziehung im Rahmenwerk ändern sich nur auf der Subklassenebene. Die Klasse `Verkauf` kennt nur Werkzeuge auf der Oberklassenebene, d.h. sie kennt nur die Oberklasse `Zahlungswerkzeug`, aber nicht die abgeleitete Klasse `Warengutscheinzahlungs-werkzeug`. Welche konkreten Werkzeuge erzeugt bzw. benutzt werden, bleibt vor ihr gekapselt.

Das nach einem Teilvorgang immer geprüft wird, welche Vorgänge jetzt dran sind, bleibt gleich. Die Verkauf ruft die Ablaufstrategieklasse auf, wenn er eine Meldung von seinen Werkzeugklassen erhält. Er holt die Daten von den Werkzeugen und prüft, anhand der Ablaufstrategieklasse des Vorganges, welche Werkzeuge jetzt aktiviert werden müssen, bzw. welche Funktion vom Werkzeuge gestartet werden soll. Wenn z.B. beim Kaffeescheck der Verkauf die Meldung "Artikelregistrierung beendet" erhält, startet sie die Funktion Bonkopf drucken.

Auch beim Konstruieren eines neues Eingabegerätes werden nur Veränderungen durch Ableitungen vorgenommen. Die Änderungen wirken sich dadurch nicht in den Oberklassen aus.

An den Klassen, die nichts direkt mit der Personalidentifizierung zu tun haben, wie z.B. der Steuerungsautomat, muss keine Änderung vorgenommen werden. Diese Klassen arbeiten mit der Oberklasse des Events zusammen, sie benötigen kein Wissen über das abgeleitete Objekt.

# 6 Konfigurierung mit Hilfe von Componentware

Im letzten Kapitel habe ich die Konstruktion vom Rahmenwerk erklärt. Das Rahmenwerk stellt eine Struktur zur Verfügung, die es ermöglicht Komponenten einzubinden. Welche Bedeutung Komponenten haben und wie Sie definiert werden, behandle ich in diesem Kapitel.

**Komponenten**

Komponenten habe ich als Artefakte für die Bedienvorgänge ausgewählt, weil sie verschiedene Möglichkeiten bieten. Ein ausgezeichnetes Merkmal von Komponenten ist das externe Interface, welches Informationen über die Komponente anbietet. Es liefert zum einem die syntaktische Schnittstelle, zum anderen die semantische Bedeutung der Komponente.

Diese Eigenschaft ist nützlich für den Konfigurator der Kasse. Die Bedienkomponenten können z.B. gefragt werden, welche Subkomponenten sie unterstützen. Eine Eigenschaft von Komponenten ist es außerdem, dass sie in ein objektorientiertes Rahmenwerk eingebaut werden können.

## 6.1 Was ist eine Komponente

Was ist jetzt eigentlich eine Komponente? Auf der Konferenz "European Conference on Object-Oriented Programming 1996" (ECOOP96) wurde die folgende Definition festgelegt.

> "A component is a unit of composition with contractually specified interfaces and explicit context dependencies only. Components can be deployed independently and are subject to composition by third Parties."
>
> [ECOOP96-1]

Eine Komponente ist also eine Einheit der Komposition mit einem vereinbarten Interfaces und expliziten Kontextabhängigkeiten. Komponenten können unabhängig entwickelt werden und sollen durch Dritte benutzt werden. Als Erste sind hier die Ersteller der Komponenten und als Zweite die Benutzer gemeint, die die fertige Applikation benutzen.

Nach dieser Definition würden Komponenten dem Bild von Bausteinen entsprechen. Damit sind Komponenten Zwischenprodukte oder Halbfertigprodukte nach [Balzert95] auf dem Weg zur Endfertigung von Software.

Damit ist nichts über den internen Aufbau der Komponenten gesagt. Es wird nicht festgelegt nach welchem Paradigma die Komponente aufgebaut wird. Sie können mit verschiedenen Programmiersprachen erstellt werden. Über ein Protokoll können sie mit der Applikation kommunizieren. Die bekanntesten sind COM, CORBA und JavaBeans. Der Umgang mit Komponenten erinnert stark an eine objektorientierte Sichtweite.

## 6.2 Unterschied Komponente und Objekte

Es stellt sich die Frage, wenn man Komponenten und Objekte anschaut, was sie unterscheidet. Beide haben Methoden, die aufgerufen werden und kapseln einen inneren Zustand, durch ihre privaten Attribute.

Ein Unterschied ist das Interface. Es haben zwar beide ein Interface, jedoch die Komponente kann über sich selbst Informationen geben, sowohl Syntax als auch Semantik. Die Komponente ist nicht für den Endbenutzer sondern für eine Zwischenebene gedacht, nämlich für denjenigen, der mit Hilfe der Komponenten ein System für den Endbenutzer erstellt. Für diese Ebene brauchen sie die Informationen über sich selber, ein Metamodell. Über die Metaschnittstelle können dann Informationen über die Komponente abgefragt werden.

Im Kontext der verteilten Systeme wird dies Metamodell ein Dienstmodell der Komponenten genannt[CHRMÜ96]. Es stellt die benötigten Informationen zur Verfügung, um zu entscheiden, ob der Dienst der Komponente derjenige ist, der benötigt wird.

Ein Dienstmodell wird durch eine Dienstbeschreibung konkretisiert und in der Dienstrepräsentation wird die Syntax der Dienstbeschreibung festgelegt.

Komponenten haben im Gegensatz zur Objektorientierung keine Vererbungsstruktur. Sie können aber Schnittstellen vererben. Dies wird auch als Vorteil gesehen. [WeckSzyp96 ] Die Wiederverwendung wird durch Aggregation und nicht durch Vererbung erreicht. Lose Kopplung wird bei COM und JavaBeans z.B. durch die Interfaces erreicht, sie sichern eine Schnittstelle zu.

**Komponente vs. Objekte**

Ähnlichkeiten

Attribute

Methoden

Verschiedene Aspekte eines Objektes werden bei der Objektorientierung durch Aspektklassen und bei Komponenten durch die Interfaces erreicht.

**Unterschiede von Komponenten zu Objekten**

explizites Interface

Sprachunabhängigkeit

Halbfertigprodukt

keine Vererbung

Komponente ist grobkörniger als eine Objekt, vergleichbar mit Fassadenmuster

[vgl. Bäumer 98]

**Black- Grey- Glass- oder White-box-Komponente**

Verschiedene Sichten der Komponenten Benutzung

- Black-box : Die Komponente versteckt ihre Innenansicht. Die einzigen sichtbaren Elemente sind die bereitgestellten Methoden. Es kann dann konsequenterweise keine Manipulation der Komponente erfolgen.
- White-box : Alle Details der Komponenten können vom Benutzer eingesehen werden. Außerdem kann die Komponente verändert werden.
- Glass-box : Der Benutzer der Komponente hat eine vollständige Sicht auf die Komponente kann aber keine Details ändern.
- Grey-box : Diese Sicht gewährt eine eingeschränkte Sicht auf die Komponente. Es können explizite Einstellungen vorgenommen werden. Es kann aber nicht die Komponente im Wesentlichen geändert werden, sowie es bei der white-box der Fall ist.

[Griffel 98,S.419]

## 6.3 Semantik von Komponenten

Für die Benutzung der Bedienungskomponenten ist es wichtig, die Semantik und die Syntax der Komponente abzubilden.

Bei der Forschung für verteilte Objekte wurden verschiedene Möglichkeiten herausgearbeitet, die Semantik einer Komponente darzustellen. Zusammenfassend läßt sich sagen, dass es kein rein formales Modell gibt, die Semantik darzustellen.

**Dienstmodelle und Dienstbeschreibungen**

Um zu beschreiben, was eine Komponente leistet, kann man die Dienste betrachten die die Komponente erbringt und sie in einem Dienstmodell darstellen. Durch die Dienstbeschreibungen werden Sie zur Verfügung gestellt,. In dem Dienstmodell werden nur die für die Wiederverwendung wesentlichen Aspekte betrachtet. Zu den relevanten Merkmalen gehören die erbrachte Leistung, das dynamische Verhalten des Dienstes und die Art und Weise, wie der Dienst genutzt werden kann bzw. muß.

Verschiedene Ansätze aus dem Forschungsbereich der verteilten Systeme.

In dem Bereich gibt es ein allgemeines Modell für die verteilten Objekte, das Dienstmodell [CHRMÜ96]. Zur Beschreibung des Dienstmodells gibt es verschiedene Ansätze.

Der namensbasierte , struktuelle und erweiterter Ansatz.

**Der namensbasierte Ansatz**

Die einfachste Form ist der namensbasierte Ansatz, er basiert auf Typnamen, d.h. der Diensttyp wird durch seinen Namen beschrieben. Diese Art enthält weder eine Information über die Schnittstelle noch Informationen über die Semantik des Dienstes.

Bei geeigneter Benennung kann die Semantik eines Dienstes von menschlichen Benutzern aus seinem Namen erschlossen werden. Dies gilt jedoch nicht für die Schnittstelle. Wenn es eine Namenskonvention gibt, die direkt auf die Schnittstelle schließen läßt, bedeutet dies, dass noch weitere Informationen, außer der eigentlichen Diensttyp-Beschreibung, notwendig sind.

Bei Diensten innerhalb einer Domäne kann man von einem gewissen Vorverständnis über die Namensgebung der Diensttypen ausgehen.

Bei namensbasierten Systemen ist es schlecht möglich Subtyp-Beziehungen darzustellen.

Der Vorteil vom namensbasierten ist die effiziente und einfache Realisierung.

**Struktuelle Dienstbeschreibungen**

Bei der strukturellen Beschreibung wird die Signatur der Schnittstelle in die Beschreibung einbezogen. Eine Signatur beschreibt eindeutig die an einer Schnittstelle möglichen Interaktionen. Im Fall einer operationalen Schnittstelle besteht sie aus den Namen, Parametern und Resultaten der Operationen. Dadurch ist es möglich die Schnittstellen von Dienstbringer und Dienstnutzer auf Konformität zu prüfen. Zusätzlich kann zur Laufzeit eine Überprüfung, der bei der Interaktion verwendeten Parameter, erfolgen.

Als Beschreibung wird meistens eine Schnittstellenbeschreibungssprache (engl. interface definition language, IDL ) benutzt.

Die Ausdrucksmächtigkeit von strukturellen Dienstbeschreibungen ist aber hinsichtlich der Semantik begrenzt, weil das dynamische Verhalten des Dienstes nicht dargestellt werden kann. Die Bedeutung des Dienstes kann nur durch die Wahl geeigneter Diensttyp- Operations- und Parameternamen ausgedrückt werden. Hier treten wieder dieselben Probleme wie bei den namensbasierten Dienstbeschreibungen auf. Hier können dann Dienste verwechselt werden, die die gleiche struktuelle Beschreibung haben, jedoch ein anderes Verhalten implizieren. Ein Beispiel, das oft dafür genannt wird, sind die Dienste STACK und QUEUE, deren Schnittstelle beide aus get und put bestehen. Trotz struktureller Äquivalenz besitzen diese Dienste eine unterschiedliche Semantik.

**Erweiterter Ansatz - Konzeptgraphen**

Ein anderer, aus dem Bereich der Wissensrepräsentation stammender Ansatz, verwendet sogenannte Konzeptgraphen zur Beschreibung von Diensten[PudBurg96]. Konzeptgraphen wurden ursprünglich zur Modellierung der Semantik der natürlichen Sprache entwickelt und sind aufgrund der ähnlichen Repräsentation und Organisation von Wissen für menschliche Benutzer intuitiv verständlich.

Ein Konzeptgraph ist ein endlicher, verbundener, gerichteter, bipartiter Graph. Die Knoten sind entweder Konzept oder Relationsknoten, wobei zwei Konzeptknoten nur über einen Relationsknoten verbunden sein können. Ein Konzept repräsentiert einen konkreten oder abstrakten Bestandteil des zu beschreibenden Systems. Beispielsweise könnte die Beschreibung eines Druckdienstes die konkreten Konzepte `DRUCKER`, `COMPUTER` und `PAPIER` sowie das abstrakte Konzept `Information` enthalten. Eine Relation beschreibt eine spezifische Beziehung zwischen Konzepten. Beispiele für Relationen sind die aus der objektorientierten Programmentwicklung bekannten Beziehungen `ist-ein (is-a)` und `benutzt(uses).`

Abbildung 6-1 zeigt als Beispiel für einen Konzeptgraphen die Beschreibung eines Druckers [PudBurg96]. Konzepte werden durch eckige, Relationen durch runde Klammern, dargestellt. Informell lautet die Aussage dieser Beschreibung: "Ein Drucker ist ein Gerät, das an einen Computer angeschlossen ist, es visualisiert Informationen, die textuell oder graphisch sein können, und druckt auf Papier oder Folien."

```
[PRINTER] -
      ->    (IS-A) -> [HARDWARE_DEVICE] ,
      ->    (CONNECTED) -> [COMPUTER] ,
      ->    (VISUALIZES) - > [INFORMATION] -> (CAN_BE) -
                                                  -> [TEXTUAL] ,
                                                  -> [GRAPHICAL] ,
      ->    (PRINTS-ON) -
                  -> [PAPER],
                  -> [SLIDES]...
```

**Abbildung 6-1:** Konzeptgraph zur Beschreibung eines Druckers

Auf Basis der semantischen Distanz zweier Konzeptgraphen kann ein Ähnlichkeitsmaß definiert werden, das einen Vergleich von Konzeptgraphen auf Ähnlichkeit und damit eine Klassifikation ermöglicht.

Die Erweiterung von strukturellen Beschreibungen um Schlüsselwörter und Dienstattribute sowie die Verwendung von Konzeptgraphen ermöglichen die Integration von semantischen Aspekten in Dienstbeschreibungen. Um jedoch das Verhalten von Diensten zu beschreiben, sind noch weitere Mechanismen notwendig.

**Erweiterter Ansatz - Protokollspezifikation**

Das Protokoll eines Dienstes legt u.a. die möglichen Reihenfolgen von Operationen an seiner Schnittstelle fest. Die Menge, der zu einem bestimmten Zeitpunkt zulässigen Operationsaufrufe hängt dabei vom jeweiligem Zustand des dienstbringenden Objektes ab, d.h. die Dienstverfügbarkeit muß nicht einheitlich sein.

Das Protokoll eines Diensterbringers kann durch ein Kanten beschriftetes Transitionssystem beschrieben werden. Dabei repräsentiert eine Stelle einen stabilen Zustand des Diensterbringers, in dem nur eine bestimmte Menge von Operationen zulässig ist.

Eine Möglichkeit für ein solches Netz ist ein Statechart [HAREL96]. Statechart bietet die Möglichkeit Hierarchien durch Subzustände darzustellen.

Diese Methoden beruhen darauf, dass die Transitionen nur eine semantische Bedeutung haben, wenn die Bedeutung der Namen den Beteiligten klar ist. Probleme bereitet dies für Systeme, bei denen die beteiligten Personen keinen Kontakt miteinander haben und daher keinen gemeinsamen Sprachgebrauch haben. In dem Kontext der Domäne, die ich in dieser Arbeit betrachte, stellt das aber kein so großes Problem dar, weil die Personenzahl, die mit dem System arbeiten, begrenzt ist, und durch ein Glossar und durch die Projektarbeit ein einheitliches Verständnis über die Begriffe gebildet werden kann bzw. vorhanden ist.

**Semantik der Kassenkomponenten**

Für das Kassensystem wurde der erweiterte Ansatz gewählt. Jede Komponente für das Kassensystem hat eine wissensbasierte Beschreibung. Die Beschreibung erfolgt mit NeoClassic.

NeoClassic wurde im AI Principles Research Department bei AT & T Bell Laboratories entwickelt. Es basiert auf einer Beschreibungslogik. NeoClassic ist eine hybride Wissensrepräsentationssprache.

Alle diese Ansätze beschreiben eine Komponente nicht vollständig. Aus diesem Grunde wurde in der Firma c.a.r.u.s. ein Konfigurierungswerkzeug entwickelt, dass das Wisssensrepräsentationssystem NeoClassic ausnutzt. Mit dem Konfigurierungswerkzeug kann der Benutzer, interaktiv die Komponenten für das Kassensystem zusammenstellen.

Jede Komponente für das Kassensystem hat eine wissensbasierte Beschreibung, die vom Konfigurator ausgelesen wird.[Wimmel 98]

## 6.4 Konfigurierung mit Hilfe des Konfigurators

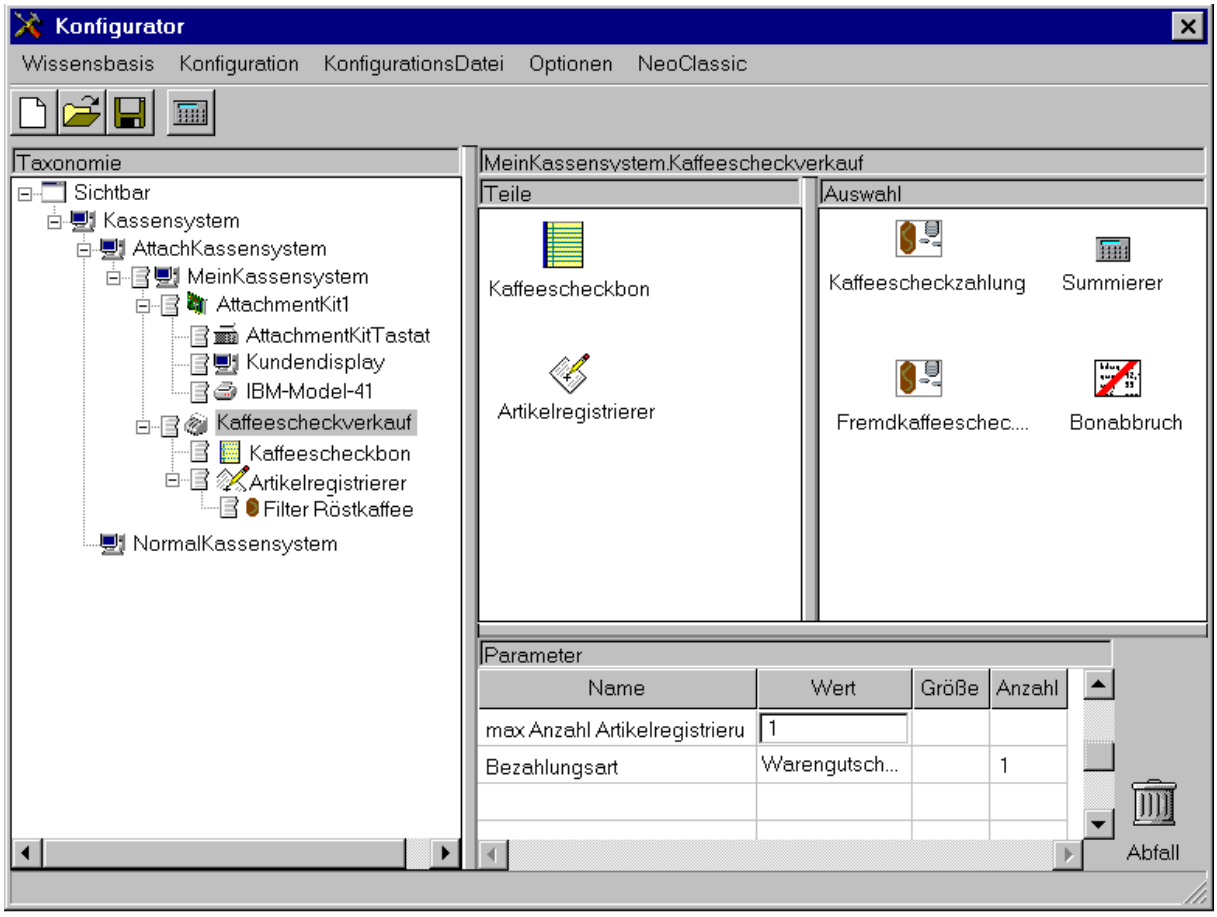


**Abbildung 6-2: Konfigurator**

Der Konfigurator, der in der Firma c.a.r.u.s. von [Wimmel 98] entwickelt wurde, kann für Konfigurierung der Bedienvorgänge der Kasse benutzt werden. Die einzelnen Komponenten können im Konfigurator geladen werden und ihm die notwendigen Informationen liefern, um die Kasse zu erstellen. Jede Komponente enthält eine wissensbasierte Beschreibung.

In der Abbildung 6-2 ist ein Prototyp abgebildet. Im linken Fenster "Taxonomie" werden die zu konfigurierenden Teile angegeben. Ein Vorgang wird konfiguriert, indem die für den Ablauf benötigten Komponenten ausgewählt werden. Die Daten, welche Komponenten ausgewählt wurden, werden gespeichert und zur Laufzeit ausgewertet. In der Abbildung 6-2 ist der Vorgang Kaffeescheckverkauf ausgewählt. Die Teile, die schon ausgewählt wurden, sind darunter in einer baumartigen Struktur (Tree-View) abgebildet, im mittleren Fenster werden sie zusätzlich abgebildet. Im rechten Fenster "Auswahl" sind diejenigen Teile abgebildet, die noch ausgewählt werden können. Im unteren Fenster "Parameter" können Parameter für eine Komponente eingestellt werden. Die Auswahl erfolgt in dieser Weise:

Um jetzt noch die Bezahlungswerkzeug Kaffeescheck auszuwählen, wird das Icon für Kaffeescheckzahlung per Drag und Drop vom Auswahlfenster in das Teilefenster gezogen. Wenn das geschehen ist, ist im Auswahlfenster nur noch der Summierer und der Bonabbruch abgebildet. Die Bezahlungsart Fremdkaffeescheck fällt jetzt weg, weil für diesen Verkauf nur eine Bezahlungsart zugelassen ist. Wenn der Summierer und der Bonabbruch ausgewählt wurde, sind alle Möglichkeiten der Auswahl ausgeschöpft. Jetzt müssen nur noch die Parameter der einzelnen Komponenten eingestellt werden, falls dies noch nicht geschehen ist.

# 7 Zusammenfassung und Ausblick

In dieser Diplomarbeit geht es um die Ablaufkonfigurierung eines Kassensystems. Das Ziel dieser Arbeit war es die bestehende Ablaufkonfiguration des bestehendes Kassensystems Eyecash so zu verändern, dass neue Kassensysteme für neue Kunden schnell und effizient erstellt werden können.

Dieses Kapitel reflektiert die Arbeit als Ganzes, indem es rückblickend ihre wesentlichen Stationen und Ergebnisse zusammenfaßt sowie Themen für weiterführende Arbeiten aufzeigt.

Um dieses Ziel erreichen zu können, habe ich zunächst die bestehende Konfigurierungstechnik des Kassensystems untersucht. Dabei habe ich festgestellt, dass die Objekte zur Konfigurierung zu kleinkörnig waren und dass die Elemente einer strukturierten Programmiersprache fehlten, um eine effiziente Konfigurierung durchzuführen. Zudem ging bei der benutzen Konfigurierungstechnik die anwendungsfachliche Ebene verloren, so dass sich der Konfigurierer diese Ebene immer wieder aneignen mußte. Daraus erfolgte die Anforderung für die Konfigurierungstechnik, einen Ansatz zu wählen, der die fachliche Struktur erhält, so dass es weniger Zeit kostet sich in die Anwendung einzuarbeiten. Zudem sollen Artefakte als Elemente zur Verfügung stehen, die als Ganzes gut wiederverwendet werden können.

Für die Lösung habe ich den WAM-Ansatz gewählt, weil er es als Zentrum hat, die Struktur der Anwendungsmodells in dem Modell des Anwendungssystem abzubilden und damit zu erhalten. In dem WAM-Ansatz werden Gegenstände abgebildet. Es werden keine Abläufe abgebildet, weil sie ein unterstützendes und kein ablaufsteuerndes Leitbild hat. Abläufe wurden nur in dem Prozeßmuster für situierte Koordination als Pläne und zwar als Laufzettel vergegenständlicht.

Der Laufzettel hat bei dem Ansatz von [Gryczan 96] einen Materialcharakter. Es diente den beteiligten Personen dazu ihre Zusammenarbeit bei arbeitsteiligen Vorgängen abzustimmen. Die Steuerung der Zusammenarbeit wurde von den beteiligten Personen selbst durchgeführt, weil es ja gerade darum ging die Zusammenarbeit zu unterstützen und nicht zu steuern.

In der Modellierung für ein Kassenablaufsystem reicht diese Technik jedoch nicht aus. In dem Kassensystem wird keine unterstützendes, sondern eine ablaufsteuernde Sichtweise eingenommen. Die Prozeßmuster werden aber als Grundlage genommen, um die Vorgänge zu vergegenständlichen.

Der Vorgang verwendet die Ablaufstrategie, um den Ablauf zu strukturieren. Die Ablaufstrategie benutzt Tätigkeiten und ein Tätigkeitsnetz um den Ablauf darzustellen. Die Tätigkeiten wurde für die Ablaufstrategie erweitert. Zu den Beschreibungen der Tätigkeiten wurden die Werkzeuge hinzugefügt, mit denen die Tätigkeit ausgeführt werden soll. Im Rahmenwerk der Kasse wurden die Verkaufsvorgänge auf ein flexibles Muster gebracht, das durch Ableitungen und durch Verwendung von verschiedenen Werkzeugen flexibel verändert werden kann. Im Kapitel 5 habe ich für einen Ablauf beschrieben, welche Klassen für eine Änderung angepaßt werden müssen.

Weiterhin habe ich das Kassenrahmenwerk in die Modellschichtenarchitektur des WAM-Ansatzes eingeordnet. Weil in dieser Architektur die letzte Schicht, die Einsatzkontextschicht, Black-box-Rahmenwerke enthält, können Komponenten für die Konfigurierung der Abläufe benutzt werden.

Die Werkzeuge, die in den Vorgängen benutzt werden, werden als Komponenten konstruiert, dadurch können sie zur Konfigurierung der Vorgänge benutzt werden.

Für die Flexibilität der Vorgänge der Kasse sind zwei Konzepte angewandt worden. Zum einem werden objektorientierte Rahmenwerke benutzt, um die Struktur der Vorgänge abzubilden und zum anderen werden Komponenten zur Konfigurierung der Black-box-Rahmenwerke benutzt. Im Rahmenwerk wurden die Werkzeuge als HotSpots eingebracht. Für den Vorgang wird festgelegt welche Arten für Werkzeuge benötigt werden und bei der Konfigurierung werden dann die konkreten Werkzeuge, die der geforderten Schnittstelle entsprechen, hinzugefügt. Auf diese Weise ist es möglich die Vorgänge zu verändern. Es kann dann, wie ich es bei der neuen Anforderung Kaffeescheckverkauf gezeigt habe, z.B. eine Artikelregistrierung mit einem Filter für die Warengruppe Röstkaffee versehen werden.

In Kapitel 6 habe ich dargestellt, welche Möglichkeiten es gibt die Semantik von Komponenten auszudrücken. Denn eine Komponente kann nur dann sinnvoll wiederverwendet werden, wenn es gelingt, die Semantik so gut wie möglich zu beschreiben, damit für einen Anwendungszweck schnell die richtige Komponente gefunden werden kann. Am Ende des Kapitels 6 habe ich den Prototypen eines Konfigurators gezeigt, der in dem Softwarehaus c.a.r.u.s. entwickelt wurde. Mit dem Konfigurator ist es möglich die Vorgänge für das Kassensystem zu konfigurieren.

Ein offener Punkt bleibt noch die grafische Darstellung von den Abläufen. Für meine Darstellungen habe ich die grafische Notation von den Tätigkeitsnetzen von [Wulf 95] benutzt. Ich habe jedoch nicht nachgewiesen, ob sie formal ausreichen würden und ob meine Erweiterungen formal zulässig sind.

# 8 Literatur


[Balzert96] **Balzert,** Helmut:
*Lehrbuch der Software - Technik*;
Heidelberg: Spektrum, Akademischer Verlag, 1996

[Bäumer 98] **Bäumer,** Dirk :
*Softwarearchitekturen für die rahmenwerkbasierte Konstruktion großer Anwendungssysteme;*
Disserationsschrift zur Vorlage am Fachbereich Informatik der Universität Hamburg, Januar 1998.

[Brachmann et al. 91] **Brachman**, R. J., **McGuiness**, D.L., **Patel-Schneider**, P. F., **Resnick**, L. A., **Borgida**. A., ;
*Living with CLASSIC: When and how to use a KL-ONE-like language.*
In John Sowa, Herausgeber, Principles of Semantic Networks: Explorations in the representation for knowledge, S 401-456, Morgan-Kaufmann, San Mateo, Kalifornien, 1991

[CHRMÜ96] **Christiansen**, Bernd; **Münke,** Malte;
*Typmanagement in offenen verteilten Systemen* ;
Diplomarbeit, Universität Hamburg, Fachbereich Informatik 1996

[Dijkstra 69/72] **Dijkstra** ;
"*GOTO's considered harmful* "
deutsch aus dem Lehrbuch [Balzert 96]

[ECOOP96-1] **Szyperski,** Clemens A.; **Pfister,** Cuno;
*Component-oriented programming:*
*WCOP'96 Report* vom "Workshop on Component-Oriented Programming" auf der
"European Conference on Object-Oriented Programming "
In M. Mühlhaeuser, editor, Special Issues in Object-Oriented Programming, Seiten 127–130. Dpunkt Verlag Heidelberg, 1997.

[Floyd 97] **Floyd,** Christiane;
*Einführung in die Softwaretechnik.*
Scriptum zur gleichnamigen Vorlesung,
Universität Hamburg, Fachbereich Informatik,
Arbeitsbereich Softwaretechnik, Hamburg, 1997

[Gamma et al. 96] **Gamma**, E. ; **Helm,** R.; **Johnson**, R. ; **Vlissides,** J.:
*Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software*.
Übersetzung von D. Riehle, Bonn: Addison Wesley, 1996.

[Griffel98] **Griffel,** Frank;
*Componentware: Konzepte und Techniken eines Softwareparadigmas*;
dpunkt Verlag Heidelberg, 1998;

[Gryczan96] **Gryczan,** Guido*;*
*Prozeßmuster zur Unterstützung kooperativer Tätigkeit.*
Wiesbaden: Deutscher Universitätsverlag, 1996 (DUV: Informatik).

[Balzert96] **Balzert,** Helmut:
*Lehrbuch der Software - Technik*;
Heidelberg: Spektrum, Akademischer Verlag, 1996

[Bäumer 98] **Bäumer,** Dirk :
*Softwarearchitekturen für die rahmenwerkbasierte Konstruktion großer Anwendungssysteme;*
Disserationsschrift zur Vorlage am Fachbereich Informatik der Universität Hamburg, Januar 1998.

[Günter 95] **Günter,** Andreas (Hrsg.) ;
*Wissensbasiertes Konfigurieren; Ergebnisse aus dem Projekt PROKON* ;
infix Verlag, Sankt Augustin, 1995

[Harel87] **Harel**, D. :
*A Visual Formalism for Complex Systems*,
in Science of Computer Programming Elsevier Publishers (North Holland) 1987

[KOBAS97] **Ivanov** E.:
*Methodische Entwicklung und Anwendung von Frameworks*,
Forschungsprojekt;
KOBAS, TU Ilmenau / ZFE der Siemens AG, Juni 1997,
www.theoinf.tu-ilmenau.de/~ivanov/scripts/konzept.ps

[Keil-Slawik 92] **Floyd**, Christiane. ; **Züllighoven**, Heinz; **Bude** Reinhard; **Keil-Slawik**, Reinhard:
*Software Development and Reality Construction*; Berlin, Heidelberg New York; Springer-Verlag1992

[Keil-Slawik 90] **Keil-Slawik**, Reinhard:
*Konstruktives Design*
Habilitationsschrift an der Technischen Universität Berlin März 1990

[Meyers Lexikon 98] **Meyer Lexikon**, 1998
elektronisches Lexikon im Internet vom
Institut für Informationsverarbeitung und Computergestützte neue Medien und © Bibliographisches Institut & F. A. Brockhaus AG
"http://www.meyer.bifab.de/meyer_frame.html"

[MFC97] Microsoft Visual C++ MFC Library Reference, Part 1 und 2 Volume 1 und 2, Visual C++ Version 5.0 Documentation Library Microsoft Press 1997

[PudBurg96] **Puder,** Arno**; Burger,** Cora;
*New Concepts for Qualitative Trader Cooperation.*
In Proceedings of the International Conference on Distributed Processing (ICDP ' 96), Dresden, 1996

[Raab 97] **Raab,** Jan;
*Entwicklung und Implementation eines Rollenwerkzeugkastens.*
Studienarbeit, Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, 1997.

[Schwanke,Albrecht95] **Schwanke,** Armin; **Albrecht**, Ralf;
Diplomarbeit an der Fachhochschule Hamburg 1995

[Balzert96] **Balzert,** Helmut:
*Lehrbuch der Software - Technik*;
Heidelberg: Spektrum, Akademischer Verlag, 1996

[Bäumer 98] **Bäumer,** Dirk :
*Softwarearchitekturen für die rahmenwerkbasierte Konstruktion großer Anwendungssysteme;*
Disserationsschrift zur Vorlage am Fachbereich Informatik der Universität Hamburg, Januar 1998.

[Udel94] **Udell,** John;
*Componentware*
Mai 1994; BYTE http://www.byte.com/

[WAM 98] **Züllighoven,** Heinz (Hrsg.):
*Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Materialansatz*;
Heidelberg dpunkt.verlag 1998

[WECK97] **Weck,** Wolfgang;
*Inheritance Using Contracts & Object Composition*;
Proceedings of the Second International Workshop on Component-Oriented Programming (WCOP '97); The TUCS General Publications Series, Vol. 5, ISBN 952-12-0047-2"; 1997; Jyväskylä, Finnland;
ftp://ftp.abo.fi/pub/cs/papers/wolfgang/WCOP97.ps.gz

[WeckSzyp96] **Weck,** Wolfgang; **Szyperski,** Clemens;
*"Do we need Inheritiance?"*
Postionpaper des Workshops CIOO96 auf der ECOOP 96
July 1996 Österreich, Linz
ftp://ftp.cs.utwente.nl/pub/doc/TRESE/cioo96/
weck.szyperski.ps.Z

[Wimmel 98] **Wimmel,** Stefan*:*
*Die wissensbasierte Konfigurierung eines Kassensystemes mit einer Beschreibungslogik*
Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Kognitive Systeme, Oktober 1998.

[Wulf 95] **Wulf,** Martina*:*
*Konzeption und Realisierung einer Umgebung zur Koordination rechnergestützter Tätigkeiten in kooperativen Arbeitsprozessen.*
Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, September 1995.