Kapitel 15: Objektorientierter Entwurf von Informationssystemen

15.1 Darstellung des konzeptionellen Datenbankentwurfs als Klassendiagramm mit dem Industriestandard UML

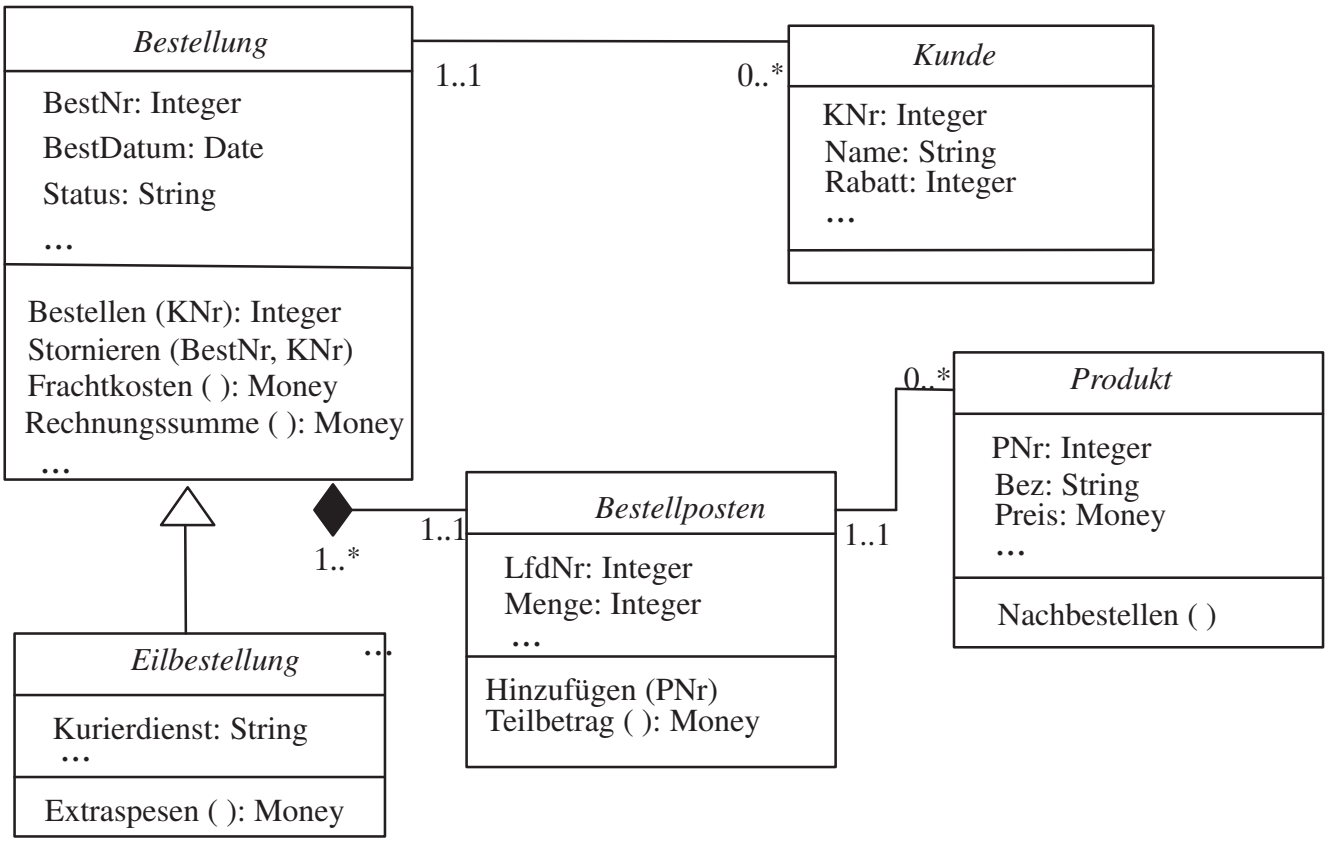
Datenbankentwürfe, die mit einer semantischen Datenmodellierungsmethode, z.B. als Entity−Relationship−Diagramm, erstellt worden sind, lassen sich auch in Form von *Objektklassendiagrammen* darstellen (vgl. auch Kapitel 6 über ODMG). Für die Darstellung hat sich der Industriestandard *UML (Unified Modeling Language)* durchgesetzt. Vorteile gegenüber ERM−ähnlichen Darstellungen liegen angeblich darin, daß

1) Objektaggregationen, d.h. die Bildung komplexer Objekte, zu gekapselten Objekten und
2) anwendungsspezifische Funktionen, d.h. Methoden auf Objekten,

in einfacher Weise direkt im Klassendiagramm dargestellt werden können.

Darüber hinaus wird häufig proklamiert, daß die Fokusierung auf Objekte (und das "Zurückdrängen" von Relationships) den Entwurfsprozeß selbst besser strukturiert und zu einer übersichtlicheren Darstellung führt.

Beispiel eines Klassendiagramms a la UML:



15.2 Weitere Aspekte des Anwendungsentwurfs mit dem Industriestandard UML

UML beinhaltet eine Reihe weiterer Diagrammarten zur Darstellung des gesamten Anwendungsentwurfs (also nicht nur der Datenaspekte). Dazu zählen:
- *Paket−Diagramme* als vergröberte Sicht auf große Klassendiagramme
- *Interaktionsdiagramme* zur Darstellung der Methodenaufrufe zwischen Objektklassen
- *Anwendungsfall−Diagramme (Use−Case Diagrams)* zur Grobdarstellung des Verhaltens von Objekten in bestimmten Anwendungsabläufen
- *Zustandsübergangsdiagramme (State−Transition Diagrams)* zur feineren Darstellung des Verhaltens von Objekten in bestimmten Anwendungsabläufen

Insbesondere Zustandsübergangsdiagramme sind für die Spezifikation der "Dynamik" eines Informationssystems, also des Verhaltens der Datenobjekte im Kontext von *Geschäftsprozessen (Business Processes)* von Bedeutung.

Workflow−Management für Geschäftsprozesse

Für die computergestützte Abwicklung geschäftlicher oder verwaltungstechnischer Abläufe hat sich der Begriff *Workflow−Management* eingebürgert. Beispielanwendungen sind etwa die Bearbeitung von Kreditanträgen in einer Bank oder Schadensfällen in einer Versicherung, die elektronische Einreichung, Begutachtung und Publikation wissenschaftlicher Zeitschriftenartikel oder − heute noch fiktiv − die elektronische Einreichung und automatisierte Verarbeitung von Steuererklärungen oder die Abwicklung aller bei Immobilienkäufen notariell, behördlich sowie finanz− und versicherungstechnisch erforderlichen Schritte. Derartige Workflows bestehen aus mehreren, automatisierten oder intellektuellen Arbeitsschritten − in der Literatur häufig *Aktivitäten* genannt −, die verschiedenen Ausführungsorganen − Sachbearbeitern oder Computerapplikationen − zugeordnet werden. Die von Aktivitäten aufgerufenen Applikationen können als Aufrufe von Methoden auf Objekten, sogenannten *Business Objects* wie zum Beispiel einem Konto− oder Bestellungs−Objekt, interpretiert werden. Insofern ergibt sich aus der objektorientierten Datenmodellierung und der funktionsorientierten Modellierung von Geschäftsprozessen eine integrierte Darstellung der Anwendung.

Heute verfügbare *Workflow−Management−Systeme (WFMS)* unterstützen dieses Grundmodell auf drei Ebenen:

- durch eine Methode zur *Spezifikation des Kontroll− und Datenflusses* zwischen den Aktivitäten eines Workflows,
- durch eine *Workflow−Engine zur Ausführung von Workflows* (im wesentlichen durch Interpretation der Workflow−Spezifikation) und damit systemunterstützten Koordination der Aktivitäten sowie
- durch eine Palette von *Administrationswerkzeugen* zur Simulation, Analyse und Visualisierung von Workflows sowie zur Festlegung und Überwachung von Regeln für die Zuteilung von Arbeitsschritten zu den Ausführungsorganen (dem sog. Worklist−Management).

Prozeßmodellierung mit State−Charts (bzw. State- und Activity-Charts)

Der State- und Activity-Chart-Formalismus wurde in Arbeiten von David Harel (ursprünglich für "reaktive", eingebettete Steuerungssysteme) entwickelt und inzwischen in den Industrie-Standard UML übernommen. State- und Activity-Charts stellen duale Sichtweisen einer Spezifikation dar.

Ein *Activity-Chart* spezifiziert den Datenfluß zwischen den Aktivitäten in Form eines gerichteten Graphen mit Aktivitäten als Knoten und Datenelementen als Kantenbeschriftung.

Ein *State-Chart* beschreibt das Verhalten einer Spezifikation, indem es den Kontrollfluß zwischen Aktivitäten als Transitionen zwischen Zuständen spezifizieren. Ein State-Chart ist im wesentlichen ein endlicher Automat mit einem ausgewiesenen Anfangszustand und Transitionen, die mit *Event-Condition-Action Regeln (ECA-Regeln)* beschriftet sind. Eine Transition von Zustand *X* nach Zustand *Y*, die mit der ECA-Regel *E[C]/A* beschriftet ist, feuert genau dann, wenn sich der Automat in Zustand *X* befindet, das Ereignis *E* eintritt und die Bedingung C erfüllt ist. Der Automat wechselt dann in den Zustand *Y* und die Aktion *A* wird ausgeführt. Bedingungen sind aussagenlogische Formeln über elementaren Bedingungen auf Zuständen (z.B. *in(S)*, die wahr ist, wenn Zustand S betreten ist) und − kontrollflußrelevanten − globalen Variablen (u.a. denjenigen Variablen, die für die Datenflußspezifikation im Activity−Chart verwendet werden). Eine Aktion kann sowohl explizit eine Aktivität starten (durch den Ausdruck *st!(aktivität))*, als auch Ereignisse erzeugen oder Bedingungen setzen (z.B. *fs!(C)* setzt die Bedingung *C* auf den Wert falsch). Jeder der drei Teile des *E[C]/A*-Tripels kann auch leer sein.

Zwei weitere Besonderheiten unterscheiden State-Charts von herkömmlichen Zustandsautomaten. Zum einen ist dies die im Hinblick auf schrittweise Verfeinerung und Abstraktion wichtige Möglichkeit zur *Schachtelung von Zuständen*. Ein Zustand kann ein komplettes State-Chart beinhalten, wobei beim Betreten eines übergeordneten Zustands implizit der jeweilige initiale Unterzustand betreten wird und beim Verlassen eines Elternzustandes alle betretenen Kinderzustände verlassen werden. Zum anderen erlauben State-Charts *orthogonale Zustandskomponenten*, mit denen Parallelität ausgedrückt wird, so daß Transitionen in verschiedenen orthogonalen Komponenten gleichzeitig feuern können.

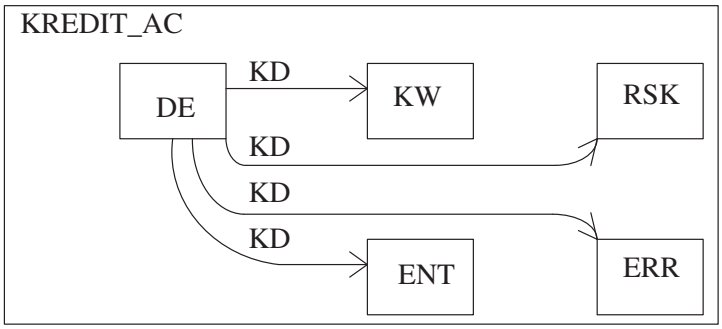
Einfaches Beispiel: Simplifizierte Kreditantragsprüfung

Eine simplifizierte Kreditantragsprüfung bestehe aus den Aktivitäten
- Dateneingabe (DE) zur Erfassung der Kundendaten (KD),
- Kreditwürdigkeitsprüfung (KW) des Kunden,
- Risikoabschätzung (RSK) aufgrund des Betrags, der Währung und Zahlungsart des Kredits,
- Entscheidung (ENT) über den Kreditantrag und einer
- generischen Fehlerbehandlungsaktivität (ERROR).

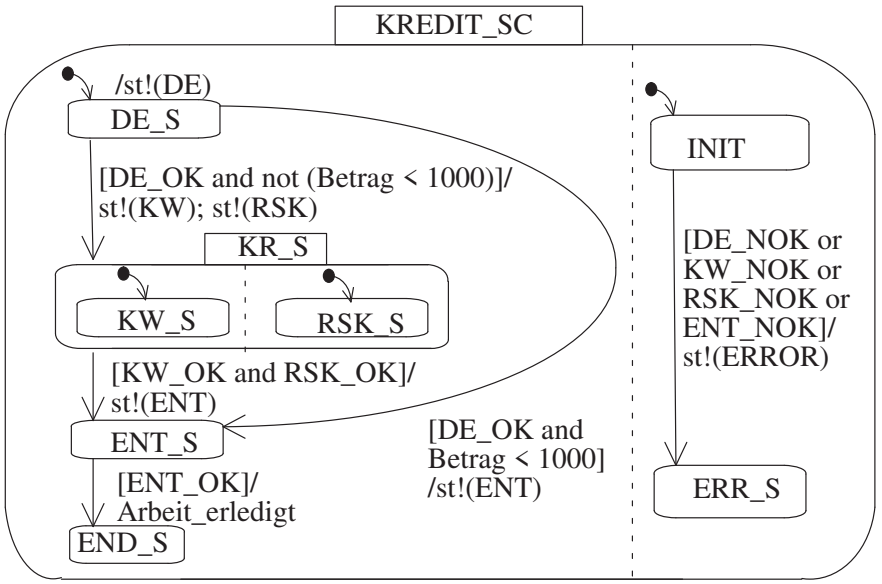
Die Aktivitäten KW und RSK sind nur bei Krediten über einer bestimmten Höhe notwendig; sie sind außerdem parallel ausführbar.

Die Aktivitäten können als Methoden eines Objekts “Kreditantrag” angesehen werden. Insofern beschreiben State− und Activity−Charts das dynamische Verhalten eines solchen Objekts.

Activity−Chart



State–Chart



Komplexeres Beispiel: Bestellungen via Internet (Electronic Commerce)

Kontrollfluß. Der Workflow beginnt mit einer Aktivität, die eine *neue Bestellung* in einer Datenbank erfaßt. Das Versandhaus stellt zwei Arten von Zahlungsmodalitäten zur Verfügung. Der Kunde kann entweder bei der Bestellung seine Kreditkartennummer angeben oder eine Rechnung anfordern, nach deren Erhalt er die angefallene Rechnungssumme überweisen muß. Der Kontrollfluß muß an dieser Stelle aufgespalten werden. Wurde eine *Kreditkartenzahlung* gewünscht, wird zunächst eine Aktivität zur Bonitätsprüfung der Kreditkarte gestartet. Nach der Prüfung der Kreditkarte werden die beiden Kontrollfluß-Pfade wieder zusammengeführt.

Nach der Registrierung der Bestellung werden der *Versand* sowie das Senden einer *Bestätigungs-Mail* an den Kunden initiiert. Dazu werden parallel zwei Sub-Workflows gestartet. Einer der beiden Sub-Workflows startet eine Aktivität, die eine Bestätigung an den Kunden versendet. Das Versenden von Waren erfolgt extern aus einem oder mehreren Lagern, die u. U. sogar von separaten Großhändlern autonom verwaltet werden. Daher startet der zweite Sub-Workflow zunächst eine Aktivität, die für jedes Lager die auszuliefernden Artikel bestimmt. Die Zuordnung wird in Artikellisten festgehalten. Innerhalb eines jeden Lagers, aus dem Artikel für die aktuelle Bestellung ausgeliefert werden müssen, wird eine Anwendung gestartet, die als Parameter die Artikelliste erhält und eine Versendegarantie zurückliefert. In einer Schleife werden so oft einzelne Lager angesprochen, bis alle Artikel auf der Bestellung bearbeitet wurden und die Versendung garantiert wurde. Wenn beide Subworkflows beendet sind, wird der Kontrollfluß erneut aufgespalten, abhängig davon, welche Zahlungsmodalität gewünscht wurde. Wurde Zahlung per Kreditkarte gewählt, wird die Kreditkarte nun belastet und der Workflow beendet. Auf diesem Kontrollfluß-Pfad werden keinerlei menschliche Eingriffe benötigt.

Soll eine Rechnung gestellt werden, wird eine Aktivität gestartet, die auf den *Eingang der Zahlung* wartet. Diese Aktivität ist intellektuell-interaktiver Natur, da die Eingabe von Daten, z.B. ob der Rechnungsbetrag vollständig eingegangen ist, durch einen Mitarbeiter des Versandhauses erfolgt. Ist die Zahlung erfolgt, ist das Szenario beendet. Sollte die Rechnung nicht innerhalb einer Frist von 14 Tagen beglichen sein, wird dem Kunden eine Mahnung geschickt. Dieser Vorgang wiederholt sich maximal dreimal im Abstand von 14 Tagen, bis die Rechnung beglichen ist. Wurde die Zahlung 14 Tage nach der dritten Mahnung noch immer nicht geleistet, so wird die Rechtsabteilung des Versandhauses informiert. Das Mahnverfahren wird in einer Schleife spezifiziert. Innerhalb der Schleife wird eine Aktivität gestartet, die eine *Mahnung* versendet, der Stichtag für das Versenden der nächsten Mahnung ermittelt und der Zähler für die bereits versendeten Mahnungen inkrementiert.

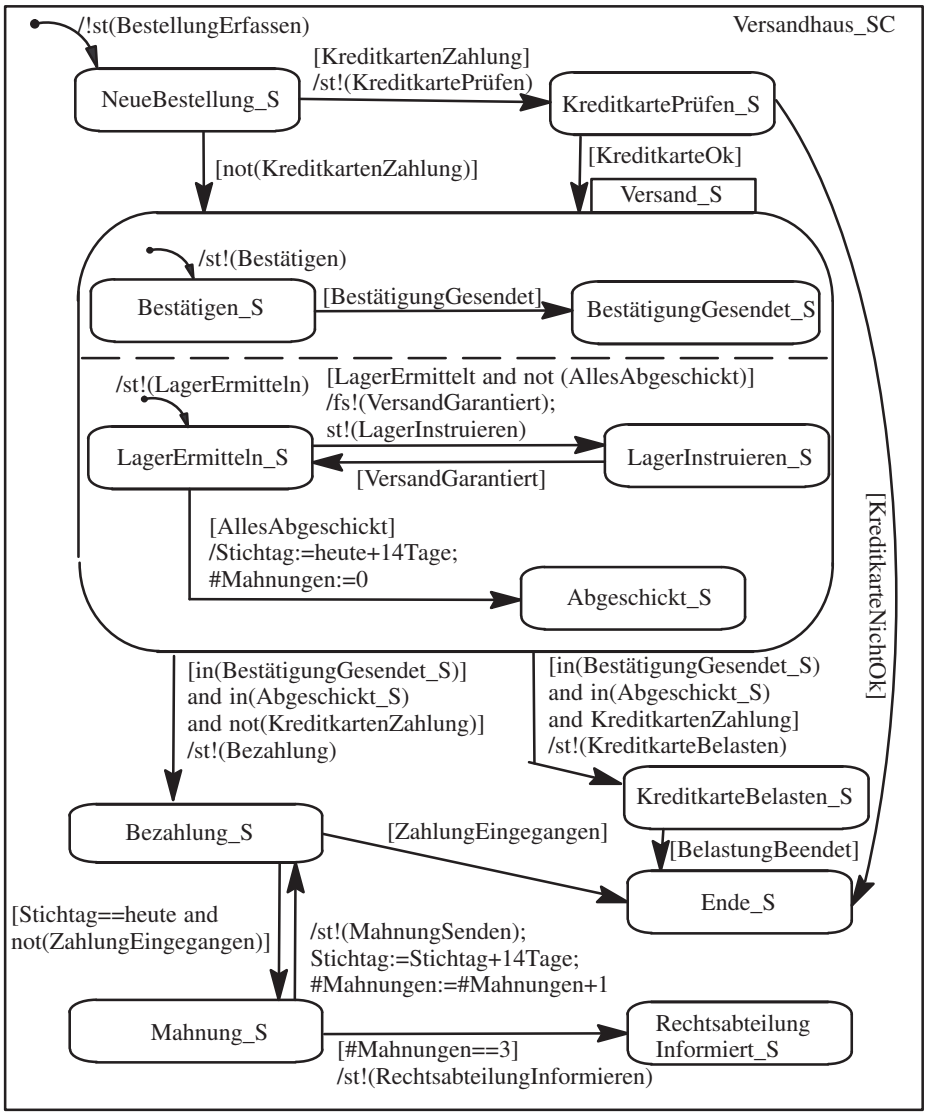
Datenfluß. Alle Eingabedaten werden in der Aktivität gesammelt, die die Bestellung erfaßt, und von ihr (bedarfsweise) an die übrigen Aktivitäten verteilt. Es ergibt sich ein sternförmiger Datenfluß, ausgehend von dieser Aktivität. Lediglich innerhalb der Schleife, die die Artikel auf die Lager verteilt, gibt es zusätzlichen Datenfluß, nämlich die jeweiligen Artikellisten in die eine und die Versandgarantien in die andere Richtung.

Spezifikation als State- und Activity−Chart. Im State−Chart wird der Übersichtlichkeit halber auf die Darstellung von Ausnahmebehandlungen verzichtet (z.B. Nichtlieferbarkeit bestellter Artikel). Initial wird die Aktivität *BestellungErfassen* gestartet und der Workflow geht in den Zustand *NeueBestellung_S* über. Die Bedingung *KreditkartenZahlung* spezifiziert die gewünschte Zahlungsmodalität und wird von der Aktivität *BestellungErfassen* gesetzt. Ist die Bedingung wahr, wird die Aktivität *KreditkartePrüfen* gestartet, die bei ihrer Terminierung die Bedingung *KreditkarteOk* auf wahr oder falsch setzt. Schlägt der Test fehl, wird der Workflow beendet, ansonsten geht er in den geschachtelten Zustand *Versand_S* über. Innerhalb dieses geschachtelten Zustandes werden zwei Sub-Workflows ausgeführt. Die Parallelität der Ausführung wird durch die gestrichelte Linie spezifiziert. In den Sub-Workflows werden die Aktivitäten *Bestätigen*, *LagerErmitteln* und *LagerInstruieren* gestartet. Sie setzen jeweils die Bedingungen *BestätigungGesendet*, *LagerErmittelt* und *Versand-Garantiert* auf wahr. Dabei durchlaufen die Sub-Workflows die Zustände *Bestätigen_S*, *LagerErmitteln_S* und *LagerInstruieren_S* und terminieren in den Zuständen *Bestätigung-Gesendet_S* beziehungsweise *Abgeschickt_S*. Vor Betreten des Zustandes *Abgeschickt_S* werden der Zähler für verschickte Mahnungen *#Mahnungen* und der *Stichtag* des Versendens der ersten Mahnung initialisiert.

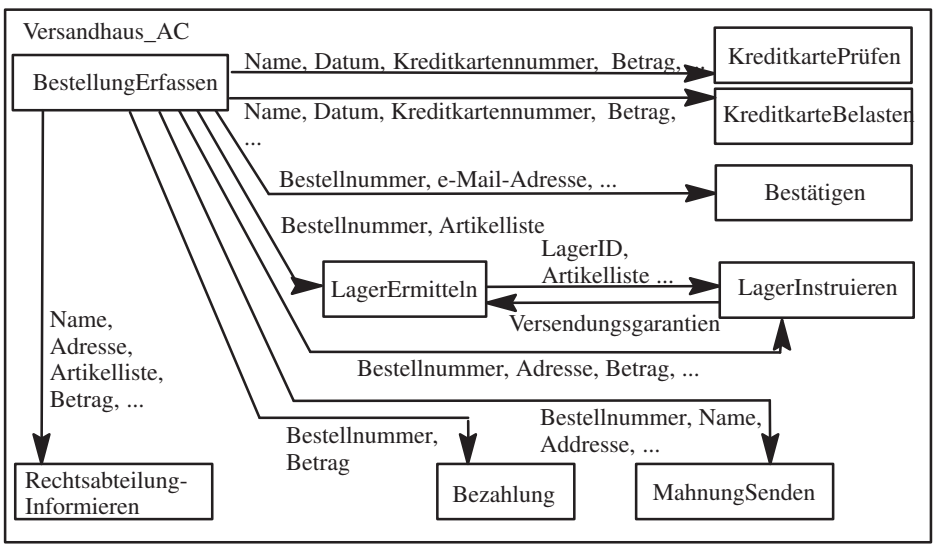
Haben beide Sub-Workflows ihren Endzustand erreicht, angezeigt durch die Bedingungen *in(Abgeschickt_S)* und *in(BestätigungGesendet_S)*, wird der geschachtelte Zustand *Versand_S* verlassen. Abhängig vom Wert der Bedingung *KreditkartenZahlung* wird die Kreditkarte durch Starten der Aktivität *KreditkarteBelasten* belastet und der Workflow endet im Zustand *Ende_S,* oder die Aktivität *Bezahlung* wird gestartet und der Workflow geht in den Zustand *Bezahlung_S* über. Bei der Aktivität *Bezahlung* wird die Bedingung *ZahlungEingegangen* auf wahr gesetzt, sobald die Rechnung beglichen wurde. Dadurch geht der Workflow in den Zustand *Ende_S* über. Die Aktivitäten *MahnungSenden* und *RechtsabteilungInformieren* mit den korrespondierenden Zuständen *Mahnung_S* und *RechtsabteilungInformiert_S* implementieren das Mahnverfahren.

Das Activity-Chart enthält alle an dem Workflow beteiligten Aktivitäten. Alle Eingabedaten werden in der Aktivität *BestellungErfassen* gesammelt und von ihr bedarfsweise an die übrigen Aktivitäten verteilt. Zusätzlich übermittelt die Aktivität *LagerErmitteln* der Aktivität *LagerInstruieren* eine *LagerID* mit der zugehörigen Artikelliste. Die Aktivität *LagerInstruieren* schickt an ihrem Ende der Aktivität *LagerErmitteln* die erhaltenen Versandgarantien zurück.

State–Chart



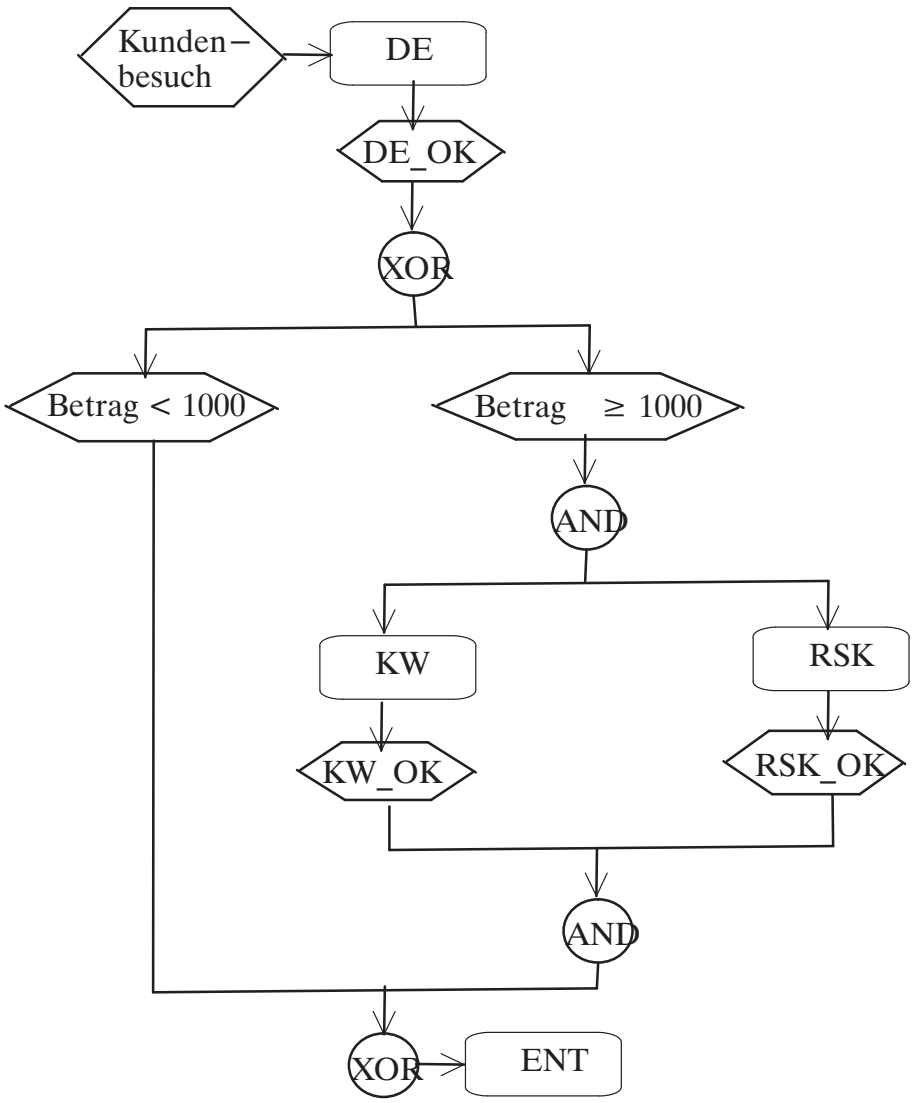
Activity-Chart



Alternative Methoden zur Prozeßmodellierung

Zur Modellierung von Geschäftsprozessen gibt es zahlreiche weitere “semiformale” oder auch stärker formalisierte Methoden, z.B. solche, die auf Varianten von Petrinetzen beruhen. Eine in der Praxis weit etablierte Methode ist die der *Ereignis−Prozeß−Ketten (Event−Process Chains), kurz: EPKs*. Dabei werden Aktivitäten mit Kontrollfluß−Konnektoren der Typen XOR (Verzweigung), AND (Parallelausführung) und OR (Nichtdeterminismus) verknüpft. Sofern die logischen Bedingungen für den Kontrollfluß, z.B. Verzweigungsbedingungen, hinreichend präzise und formal spezifiziert werden, lassen sich EPKs automatisch in State−Charts übersetzen. Dies kann nützlich sein, wenn Anwender beispielsweise mit EPKs zunächst grob modellieren, dann schrittweise verfeinern und präzisieren, bis sie schließlich eine übersetzbare Spezifikation erhalten, die sie schlußendlich dem StateChart−Interpreter eines Workflow−Management−Systems zur Ausführung übergeben können. EPKs können über die Beschreibung des Kontroll− und Datenflusses hinaus auch organisatorische Zuordnungen der Aktivitäten (Funktionen) zu Abteilungen, Sachbearbeiterrollen, etc. in Form von Annotationen darstellen.

Einfaches Beispiel eines EPK−Modells (Kreditantragsprüfung):



15.3 Objektorientierte Analyse– und Entwurfsmethodologie

Für die Anforderungsanalyse und den Anwendungsentwurf wird – basierend auf der objektorientierten Darstellungsmethode (insbesondere den Klassendiagrammen) – häufig auch eine bestimmte *Vorgehensmethodologie* verfochten. Diese stellt – mehr oder weniger grobe – Richtlinien zusammen, in welchen Schritten die Analyse bzw. der Entwurf vorangetrieben werden sollte. Die wohl bekannteste unter diesen Schulen ist die *Object Modeling Technique (OMT)* von Rumbaugh, deren diagrammbasierte Notation zusammen mit den Notationen von Booch und Jacobson zum Industriestandard UML geführt hat.

OMT propagiert die folgenden Analyse– bzw. Entwurfsschritte, die zur schrittweisen Verfeinerung ineinander verzahnt iterierbar sind:

1) *strukturelle Modellierung* der Datenobjekte:
   Objektklassen mit ihren Assoziationen, insbesondere Aggregationen
2) *Dynamikmodellierung*:
   Zustandsübergangsdiagramme zur Darstellung des dynamischen Verhaltens von Objekten bzw. Geschäftsprozessen
3) *Funktionsmodellierung*:
   Beschreibung der Objektmethoden, ihrer gegenseitigen Aufrufe und des Datenflusses zwischen ihnen

Diese Grobstruktur kann weiter detailliert werden, bleibt jedoch letzten Endes nur ein Leitfaden und darf keinesfalls als Patentlösung für die Praxis angesehen werden.

Ergänzende Literatur zu Kapitel 15:

Object Management Group (OMG), Unified Modeling Language (UML) Documentation, http://www.omg.org/ oder http://www.rational.com/uml/

S.M. Lang, P.C. Lockemann, Datenbankeinsatz, Springer–Verlag, 1995

G. Weikum, D. Wodtke, A. Kotz–Dittrich, P. Muth, J. Weißenfels, Spezifikation, Verifikation und verteilte Ausführung von Workflows in MENTOR, in: T. Härder (Hrsg.), Informatik Forschung und Entwicklung Band 12 Heft 2, Springer–Verlag, 1997, Themenheft über Workflow–Management

G. Vossen, J. Becker (Hrsg.), Geschäftsprozeßmodelle und Workflow–Management – Modelle, Methoden, Werkzeuge, International Thomson Publishing, 1995

S. Jablonski, C. Bussler, Workflow Management – Modeling Concepts, Architecture, and Implementation, International Thomson Publishing, 1996

Scheer, A.–W., ARIS – Business Process Modeling, 2nd Edition, Springer–Verlag, 1999

Balzert, H., Wie erstellt man ein objektorientiertes Analysemodell?, Informatik Spektrum Band 20 Heft 1, Springer–Verlag, 1997