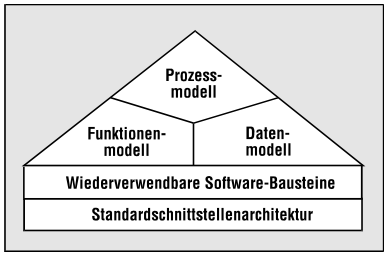


VAA

Handbuch
für die Spezifikation
fachlicher Anwendungsbausteine

Wolfgang Keller, sd&m

Version 1.0
1. Februar 1996

Management Summary

Mit diesem Dokument legt der AK-VAA ein einheitliches Handbuch zur Spezifikationsmethodik für die fachlichen Anwendungsbausteine von VAA vor. Dies geschieht mit dem Ziel, den fachlichen Arbeitskreisen wie *Partner* oder *Schaden/Leistung* die Erstellung von Spezifikationen in einheitlicher Qualität zu erleichtern. Ein Großprojekt wie VAA kommt nicht ohne „Methodenhandbuch“ aus.

Ziel sind verständliche, fachlich richtige und ausreichend präzise Spezifikationen, die sich einfach in robuste und fehlerarme Software für die beteiligten Versicherungsunternehmen umsetzen lassen. Die Methodik und die Konstrukte der Spezifikation orientieren sich dabei auch an der VAA-Architektur, was ihre spätere Umsetzung in Software erleichtert.

Spezifikationen bestehen aus fachlichen Übersichtsteilen und eher formalen Modellen, die mit der sog. Sachbearbeitermethode spezifiziert werden. Geschäftsprozesse, Funktionen- und Datenmodelle sind darin enthalten. Benutzerschnittstellen werden nicht spezifiziert. Geschäftsprozeßmodelle haben lediglich den Charakter von Referenzmodellen. Die Methode ist eine Form der Objektorientierung, die lediglich Datenkapselung benutzt und auf fortgeschrittene Konstrukte verzichtet. Dies erleichtert es den Projektbeteiligten die Methode zu erlernen, da nur wenige neue Konstrukte als Ergänzung zur konventionellen Daten- und Funktionenmodellierung erlernt werden müssen.

STOP

Dieses Dokument ist keine Richtlinie für die Beschreibung technischer Querschnittskomponenten wie zum Beispiel Datenmanager, Parametersystem oder Workflowsystem. Dafür empfehlen wir, wenn der Bedarf auftritt, objektorientierte Beschreibungen. Es wird hier diskutiert werden, warum die Schbearbeitermethode für technische Subsysteme nicht geeignet ist. Ein Methodenhandbuch für technische Subsysteme wird vom AK-VAA nicht erarbeitet werden. Es kann beim AK-VAA aber die Beschreibung einer „objektorientierten Notation“ in Form eines Handbuches angefordert werden. Es existieren auch ausreichend große Beispiele, die im Rahmen eines Coachings erläutert werden können.

Wolfgang Keller
software design & management
GmbH & Co. KG
Thomas-Dehler-Str. 27
81737 München
Tel. 089-63812-210
Fax. 089-63812-150
Internet-Mail: wk @sdm.de

Inhaltsverzeichnis

I. Einführung und Überblick 1
I.1. Anforderungen an VAA-Spezifikationen 1
I.2. Was ist eine Spezifikation? 3
I.3. Spezifikation mit Sachbearbeitern 4
I.4. Rolle der Softwarearchitektur 5
I.5. Methodische Voraussetzungen 7
I.6. Umfang und Allgemeingültigkeit von Spezifikationen bei VAA 8
I.7. Überblick über dieses Handbuch 9

II. Geschäftsprozesse 12
II.1. Spezifikation von Geschäftsprozessen 12
II.2. Beispiel 16
II.3. Darstellung des Beispiels 19
II.4. Beschreibungselemente 20
II.4.1. Graphische Beschreibungselemente 20
II.4.2. Textuelle Beschreibungselemente 21
II.5. Mögliche weitere Beschreibungsattribute 22
II.6. Exkurs: Objektorientierte Geschäftsprozeßmodelle 23

III. Sachbearbeiter 24
III.1. Spezifikation mit Sachbearbeitern 24
III.2. Beispiel 28
III.3. Bildung sinnvoller Anwendungsbausteine 30
III.4. Beschreibungselemente 32
III.4.1. Graphische Elemente für Zustandsdiagramme 32
III.4.2. Sachbearbeiter 32
III.4.3. Datensicht 33
III.4.4. Aufträge 33

IV. Batches 36
IV.1. Motivation 36

V. Datenmodell 37
V.1. Datenmodellierung bei VAA 37
V.2. Beispiel 37
V.2.1. Beispiel Entitäts- / Beziehungsdiagramm 37
V.2.2. Beispiele für Entitätsbeschreibungen 38
V.2.3. Diskussion verwendeter Konstrukte 39
V.3. Beschreibungselemente 40
V.3.1. Graphische Beschreibungselemente 40
V.3.2. Entitätsbeschreibungen 41

VI. Datentypen 43
VI.1. Ziele eines Datentypkonzeptes 43
VI.2. Beschreibungselemente 44
VI.2.1. Primitive Datentypen 44
VI.2.2. Aufzählungstypen 46
VI.2.3. Feldtypen (Arrays, Reihungen) 47
VI.2.4. Einschränkungstypen 47
VI.2.5. Verbundtypen (Records) 48

VII. Offene Schnittstellen 49
VII.1. Beispiel 49
VII.2. Formale Syntax 50

VIII. Mustergliederung 51
VIII.1. Ziele und Gültigkeit 51
VIII.2. Mustergliederung 51
VIII.3. Der „fachliche Beschreibungsteil“ 52

IX. Checkliste Spezifikationsfehler 54
IX.1. Bilder und Texte 54
IX.2. Generelle Fehler 54
IX.3. Geschäftsprozeßmodelle 55
IX.4. Sachbearbeiter 56
IX.5. Datenmodelle 57

X. Anhang 58
X.1. Auswahl der Methode 58
X.2. Offene Fragen VAA 59
X.2.1. Wiederverwendung 59
X.2.2. Begriff des Anwendungsbausteins 60
X.2.3. Referenzmodelle kontra „objektorientierte Wiederverwendung“ 61
X.2.4. Geschäftsprozeßmodelle 61
X.2.5. Benutzungsschnittstelle 61
X.2.6. Abgrenzung der Workflow- und der DV-Vorgangssteuerung 61
X.3. Literaturverzeichnis 63
X.4. Historie der Versionen dieses Papiers 65
X.4.1. Version 0.a 65
X.4.2. Version 0.b und 1.0 65

 © GDV, sd&m 1996

I. Einführung und Überblick

I.1. Anforderungen an VAA-Spezifikationen

VAA hat sich zum Ziel gesetzt, einen Markt von Anwendungssoftwarebausteinen für Versicherungsunternehmen zu schaffen, auf dem Softwarebausteine unterschiedlicher Hersteller von den Versicherungsunternehmen zu einem für ihr Geschäft und ihre Prozesse maßgeschneiderten Gesamtsystem zusammengestellt werden können.

Damit dieses Ziel umgesetzt werden kann, müssen entsprechende Softwarebausteine zunächst spezifiziert und dann realisiert werden. Den einzelnen Herstellern werden dabei von der VAA nicht nur die technischen Rahmenbedingungen in Form einer Softwarearchitektur vorgegeben, in die die Softwarebausteine integrierbar sein müssen. Es werden auch die fachlichen Leistungen einzelner Bausteine durch Arbeitsgruppen von VAA definiert.

Damit Softwarehersteller als Anbieter von Bausteinen für VAA auftreten können, müssen für die Bausteine also Spezifikationen vorliegen, die so präzise sind, daß zwei verschiedene Hersteller auf der Basis dieser Spezifikationen Software mit derselben Außenfunktionalität herstellen können.

Dazu dürfen Spezifikationen nicht nur Stoffsammlung und grobe Anforderungsdefinitionen sein, sondern müssen die Außenschnittstelle der jeweiligen Bausteine präzise beschreiben. Ist die Spezifikation nicht ausreichend genau, so treten zwei negative Effekte ein:

- Bausteine sind nicht problemlos in ein VAA-konformes System integrierbar.
- Softwarehersteller haben keinen Anreiz, in VAA zu investieren, da sie nicht sicher sein können, daß die Software, die sie realisieren, auch wirklich den Anforderungen entspricht.

Qualität

VAA steht und fällt also mit ***fachlich richtigen*** und präzisen Spezifikationen der Anwendungsbausteine. Die Qualität dieser Spezifikation muß dabei höher sein als die von Referenzmodellen. Referenzmodelle wie zum Beispiel IAA, FAA oder auch ARIS haben lediglich den Anspruch ein Softwaresystem oder Teile daraus für ein repräsentatives Unternehmen möglichst beispielhaft und allgemein zu spezifizieren. Sie haben nicht den Anspruch, daß die Modelle 1:1 ohne Anpassung an ein Unternehmen in Software umgesetzt werden können und somit in einer Vielzahl von Unternehmen lauffähig sind. Sie haben auch nicht den Anspruch, so präzise zu sein, daß daraus unzweideutig Software abgeleitet werden kann.

Um die Ziele von VAA zu erreichen, müssen die Spezifikationen also strengeren Qualitätsmaßstäben genügen, als Referenzmodelle. Um eine solche Qualität in einem Großprojekt wie VAA zu erreichen ist es notwendig aber noch nicht hinreichend, in allen Arbeitskreisen nach einer einheitlichen Spezifikationsmethode vorzugehen. Eine solche einheitliche Spezifikationsmethode wird in diesem Papier vorgeschlagen und mit dem Projektfortschritt von VAA noch weiter verfeinert werden.

Das primäre Ziel einer solchen Methode ist es nicht, einen Maßstab zu haben, an dem man prüfen kann, ob formale Regeln eingehalten wurden. Ziel ist es vielmehr, die Erstellung qualitativ hervorragender Spezifikationen zu unterstützen.

Verständlichkeit, Präzision, fachliche Richtigkeit

Dazu müssen die entstehenden Spezifikationen vor allem ***verständlich*** sein. Eine Überformalisierung dient diesem Ziel meist nicht und soll hier vermieden werden. Verständlichkeit heißt vielmehr, daß auch versicherungstechnische und DV-technische Nichtexperten in der Lage sein müssen, die nach der Methode entwickelten Spezifikationen zu verstehen. Es kann nicht erwartet werden, daß alle Leser von Spezifikationen DV-technisch gebildete Versicherungsexperten sind. Die Spezifikationen müssen also von Fachabteilungen *gerne gelesen* werden. Sie müssen ferner ausreichend ***präzise*** sein, damit Softwarehersteller bei der Umsetzung zu unzweideutigen Ergebnissen kommen. Sie müssen auch von solchen Entwicklern verstanden werden können, die keine Experten auf dem jeweiligen Fachgebiet sind, sondern lediglich über versicherungswirtschaftliches Grundwissen verfügen. Dazu kann die Methode nur indirekt durch geeignete Metaphern beitragen. Die eigentliche Arbeit ist hier von den Anwendern der Methode zu leisten. Eine Qualitätssicherung nach einheitlichen Maßstäben ist unbedingt erforderlich.

Eine Qualitätssicherung der Spezifikationen muß also zunächst die fachliche Richtigkeit prüfen. Dazu muß sie die Spezifikationen verstehen. Bei der Prüfung soll sich die Qualitätssicherung also nicht primär daran orientieren, ob auch die Mustergliederung akribisch genau eingehalten wurde. Wenn das der Fall ist, ist das zwar schön, aber eben meist nicht ausreichend. Eine Qualitätssicherung muß vielmehr prüfen, ob die Spezifikationen fachlich richtig, lesbar und ausreichend präzise sind.

Damit das möglich ist, muß das Erstellen der Spezifikationen allen Beteiligten ***Freude*** bereiten. Den Spezifikateuren muß das Schreiben der Spezifikationen leichtfallen. Die hier zu entwickelnde Methode muß also selbst leicht verständlich und einfach anwendbar sein. Der Qualitätssicherung und den Fachabteilungen muß das Lesen und die Prüfung der Spezifikationen leichtfallen. Dazu müssen Spezifikationen flüssig lesbar sein.

Integrierbarkeit

Eine weitere Anforderung ergibt sich aus der schieren Größe des Vorhabens VAA. VAA ist ein *Software-Großprojekt*. Damit sich später alle Bausteine zu einer Lösung integrieren lassen, muß schon die Spezifikation der Bausteine fachlich ***konsistent*** sein. Man kann dies durch zwei Maßnahmen fördern. Erstens kann man Spezifikationen maschinell auswertbar machen, so daß bestimmte Konsistenzbedingungen maschinell geprüft werden können. Dazu ist zwingend eine einheitliche Aufschreibung der Spezifikationen erforderlich.

Dies ist jedoch noch nicht der entscheidende Punkt. Zweitens kann man durch die Spezifikationsmethode eine sinnvolle ***Modularisierung schon der Spezifikation*** fördern. Das heißt insbesondere, daß fachliche Spezifikationen in überschaubaren Einheiten erstellt werden müssen, die durch klare und vor allem geistig ***prüfbare Schnittstellen*** voneinander abgegrenzt werden. Eine solche Clusterung der Spezifikation kann schon durch die Spezifikationsmethode gefördert werden.

Eine Bildung verständlicher und in sich selbst ***prüfbarer*** Einheiten wird durch die hier vorgelegte Methode in mehrfacher Hinsicht gefördert. Auf Ebene von sog. Sachbearbeitern durch Kapselung und Unterstützung von Datenabstraktion und auf der Ebene mehrerer kooperierender Softwarebausteine durch die Bildung von abgeschotteten Einheiten - sogenannten Clustern. Datenabstraktion und Clusterung kann wieder nicht durch die Methode allein vorgegeben werden sondern muß das besondere Augenmerk einer Qualitätssicherung genießen.

 © GDV, sd&m 1996

Robustheit und Wartbarkeit

Ein Qualitätskriterium für Software allgemein ist ihre Robustheit und Wartbarkeit, also das kontrollierte Verhalten im Fehlerfall und bei unvorhergesehenen Situationen sowie die Möglichkeit, Fehler schnell zu finden.

Zur Robustheit von Software kann man schon in der Spezifikation einen wesentlichen Beitrag leisten, indem man Vor- und Nachbedingungen sowie Fehlerzustände an den Schnittstellen von Bausteinen präzise spezifiziert. Wird dies vernachlässigt, bekommt man Wartungsprobleme und ein unbefriedigendes Verhalten im leider unvermeidlichen Fehlerfall.

Eine Spezifikationsmethode kann hier wieder nur durch die Bereitstellung entsprechender Konstrukte unterstützen. Genutzt werden müssen die Konstrukte von den Spezifikateuren und überprüft werden kann die Robustheit und Wartbarkeit nicht durch formale Kriterien, sondern nur durch eine intelligente Qualitätssicherung.

I.2. Was ist eine Spezifikation?

Wir haben oben unser Verständnis der wesentlichen Anforderungen an eine Spezifikationsmethode vorgestellt. Bevor in den folgenden Kapiteln die einzelnen Elemente der Spezifikationsmethode für VAA vorgestellt werden, soll der Begriff der Spezifikation jedoch noch gegen andere Dokumente im Software-Entwicklungsprozeß angegrenzt werden.

Unter einer (System-) ***Spezifikation*** wollen wir Dokumente verstehen, die die ***fachliche*** Funktionalität und die ***Schnittstellen*** eines ***Software***systems nach außen beschreiben.

Fachliches WAS und nicht technisches WIE

Fachliche Spezifikationen beschreiben also nur fachliche Inhalte und keine technischen Details. Solche werden in der Phase der Systemkonstruktion beschrieben. Spezifikationen beschreiben das WAS ein Softwaresystem tut und nicht, WIE es getan wird. Das WIE ist der Systemkonstruktion und der Realisierung vorbehalten. Das heißt insbesondere, daß eine Außen- und keine detaillierte Innenansicht beschrieben wird.

EDV und nicht die gesamte Versicherung

Eine weitere Betonung in unserer Definition liegt auf dem Begriff Software. Es macht keinen Sinn, im Rahmen von VAA Objekte des Versicherungsgeschäfts zu beschreiben, die keinerlei Bezug zur im Rahmen von VAA zu erstellenden Software haben. Eine exakte Grenzziehung ist hier notwendig, um eine Chance zu haben, das Ziel, nämlich ein Softwaresystem, zu verwirklichen. Man sollte sich bei der Beschreibung also auf die Softwaresysteme konzentrieren und nicht auf die Beschreibung von Geschäftsprozessen, die überhaupt nicht durch Software unterstützt werden.

Bei der Beschreibung der Geschäftsprozesse werden auch noch Abläufe beschrieben, die nur teilweise von DV unterstützt werden. Bei der Spezifikation der Anwendungsbausteine werden jedoch nur noch elektronische Sachbearbeiter beschrieben und keine menschlichen Tätigkeiten. Läßt man diesen Fokus fallen, läuft man Gefahr, sich stark zu verzetteln.

I.3. Spezifikation mit Sachbearbeitern

Dieses Handbuch wird als Methode für die Spezifikation von DV-Systemen die Sachbearbeitermethode von Denert [Den91] einführen. Diese Methode wird in Kapitel III noch ausführlich beschrieben. Damit man jedoch die Beziehung zwischen Softwarearchitektur und Spezifikation besser verstehen kann, sollen Sachbearbeiter und Geschäftsvorfälle hier schon vorab in Kurzform eingeführt werden.

Die fachliche Funktionalität wird bei VAA in Form einer Menge von elektronischen Sachbearbeitern beschrieben. Ein Sachbearbeiter ist ein elektronischer Agent, der fachliche Dienste erbringen kann. Wie er diese fachlichen Dienst erbringt, bleibt ihm *überlassen (Information Hiding*). Zur Erbringung seiner Dienste stehen ihm seine privaten „elektronischen Karteikästen“ zur Verfügung, in denen nur er allein schreiben darf. Andere Sachbearbeiter dürfen in seinen Karteikästen lediglich lesen, nie jedoch schreiben. Die Karteikästen (das Datenmodell) bilden das Gedächtnis der Sachbearbeiter. Die privaten Karteikästen eines Sachbearbeiters sind seine Datensicht.

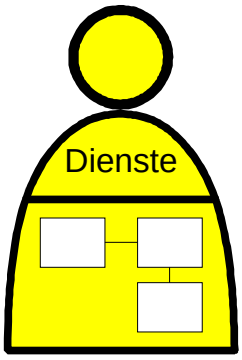


Abbildung 1: Sachbearbeiter mit Datensicht

Abbildung 1 zeigt symbolisch einen elektronischen Sachbearbeiter, der seinen Karteikasten gegen direkte Zugriffe nach außen versteckt[1] und Dienste anbieten kann.

Typische Dienste eines Sachbearbeiters sind zum Beispiel die Aufnahme eines neuen Risikos in eine Versicherungspolice, die Auszahlung eines Schadens oder die Erstellung der Jahresabrechnung für einen Kunden.

Es handelt sich dabei also um komplexe fachliche Vorgänge, die es auch erforderlich machen können, daß ein Sachbearbeiter sich der Dienste anderer Sachbearbeiter zur Aufgabenerfüllung bedient. Typische Sachbearbeiter wären zum Beispiel ein Partner-Sachbearbeiter, der alle Geschäftsvorfälle rund um das fachliche Objekt „Partner“ erledigen kann. Man beachte, daß dabei Aufträge wie Kunde anlegen, ändern oder löschen i.A. keine interessanten Fälle sind. Interessanter und spezifikationswürdig sind die oben aufgezählten komplexen Aufträge (Geschäftsvorfälle).

Bezug zur Objektorientierung

Es ist für Kenner objektorientierter Methoden gut ersichtlich, daß es sich bei der Sachbearbeitermethode um eine objektorientierte Spezifikation, allerdings ohne Vererbung, Polymorphie und generische Programmierung (Templates) handelt. Übernommen wurde die Datenkapselung, also die Idee abstrakter Datentypen. Abbildung 2 zeigt die methodische Einordnung im Überblick.

Sachbearbeiter als Metapher

[1] Information-Hiding, also das Verstecken der eigenen Karteikästen, ist eine Praxis, die auch im Büroalltag bei Sachbearbeitern aus Fleisch und Blut immer wieder beobachtet werden kann.

 © GDV, sd&m 1996

Die Sachbearbeitermethode ist eine Metapher, die sich - wie zu erwarten - gut für die Beschreibung der elektronischen Sachbearbeitung in administrativen DV-Systemen eignet. Sie trägt damit zur Verständlichkeit der Spezifikationen bei. Metaphern können das Leben stark erleichtern, wenn sie in Gebieten eingesetzt werden, zu denen sie passen und für die sie entwickelt wurden.

Sie können das Leben aber auch beliebig erschweren, wenn sie in Gebieten eingesetzt werden, zu denen sie nicht passen. Sachbearbeiter also zum Beispiel für die Spezifikation technischer Querschnittskomponenten in VAA einzusetzen, wäre ein Bruch der Metapher. Ein solches Vorgehen muß daher vermieden werden. Eine weiterführende Diskussion von Metaphern in der Anwendungsentwicklung ist in [Kil+93] enthalten.

Methode	SA/SD	Sachbearbeiter	Objektorientiert
Charakteristik	-Daten -Funktionen	+ Kapselung	+ Vererbung + Polymorphie
Einordnung		Soft-OO	Hard-OO
Verwendung		fachliche Spezifikationen (Versicherungs-logik)	Architektur-Spezifikationen (Technik)

Abbildung 2: Position der Sachbearbeitermethode

I.4. Rolle der Softwarearchitektur

Damit später eine einfache und bruchlose Umsetzung der Spezifikationen in Software erfolgen kann, ist es praktisch, wenn ein direkter und einfacher Bezug zwischen Spezifikationsmethode und der Umsetzung besteht. Wie die Realisierung der Software erfolgt, wird von der Softwarearchitektur maßgeblich bestimmt. Es ist also gut, wenn eine einfache Abbildung zwischen der Spezifikationsmethode und der Softwarearchitektur besteht.

Abbildung 3 zeigt die Softwarearchitektur von VAA [VAA95]. Die Sachbearbeiter bilden darin in Form der *Arbeitsebene* den Kern der fachlichen Funktionalität. Sachbearbeiter werden normalerweise durch Aktionen des Benutzers angestoßen, also aus der Benutzungsschnittstelle. Diese ist in Abbildung 3 aber nicht sichtbar - ein offener Punkt der VAA-Architektur (siehe auch Abschnitt X.X.2). Wir wollen also bis zur Klärung des Punktes als Arbeitshypothese annehmen, daß die Sachbearbeiter ihre Aufträge aus den elementaren Arbeitsschritten der Geschäftsprozesse erhalten. Diese Annahme ist brauchbar, da ein menschlicher Sachbearbeiter in einem Arbeitsschritt eines Geschäftsprozesses normalerweise EDV-Dialoge verwendet, um bestimmte fachliche Aktionen, also zum Beispiel die Aufnahme eines neuen Risikos in eine Police, auszulösen. Man kommt dann zu der Darstellung in Abbildung 4.

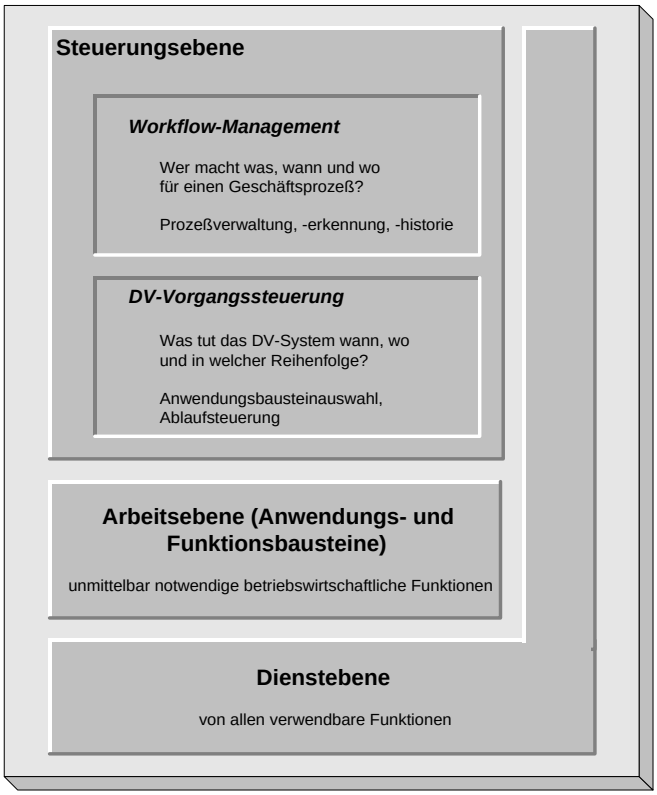


Abbildung 3: VAA-Softwarearchitektur

In Abbildung 3 ist ferner keine Datenbank-Schicht dargestellt. Diese wird jedoch bei VAA als Bestandteil der Dienstebene angesehen. Logische Datenmodelle werden direkt vom Datenmanager unterstützt.

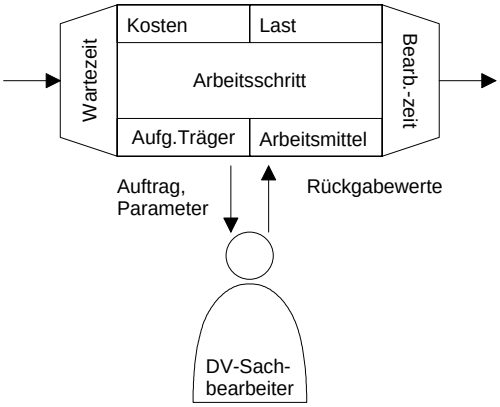


Abbildung 4: Beauftragung von Sachbearbeitern aus den Geschäftsprozessen

Zusammenfassend kann also festgestellt werden, daß die Sachbearbeitermethode zur bisher vorhandenen VAA-Architektur kompatibel ist. Sachbearbeiter können damit später leicht in dem durch die VAA-Architektur vorgegebenen technischen Rahmen realisiert werden. Eine Klärung der Abgrenzung zwischen Workflow-Steuerung und DV-Vorgangssteuerung wäre zwar hilfreich, ist aber momentan bei der fachlichen Spezifikation noch nicht zwingend erforderlich.

 © GDV, sd&m 1996

I.5. Methodische Voraussetzungen

Dieses Handbuch kann und soll keine Einführung in das Software-Engineering oder in einzelne Spezifikationstechniken, wie Daten- und Funktionenmodellierung, geben.

In diesem Handbuch wurde Wert darauf gelegt, die eingeführten Methoden anhand der vorhandenen Literatur zu fundieren, um Kennern eingeführter Methoden die Einarbeitung zu erleichtern. In den folgenden Kapiteln wird von dem „Handwerkszeug“ von DV-Fachleuten für die Fundierung von Spezifikationsmethoden immer wieder Gebrauch gemacht werden.

Die einzelnen dort vorkommenden Begriffe zur Datenmodellierung, Funktionenmodellierung und zur objektorientierten Analyse werden nicht weiter erläutert, da wir davon ausgehen, daß es sich bei den Nutzern dieses Handbuches, also den „Schreibern“ von Spezifikationen, vor allem um DV-Fachkräfte handelt. Die Resultate, nämlich die fertigen Spezifikationen, sollten aber auf jeden Fall auch ohne umfassendes Grundwissen der Informatik verständlich sein, da wir nicht davon ausgehen können, daß alle Leser von VAA-Spezifikationen ebenfalls DV-Fachkräfte sind.

Weiterführende Literatur

Sollten Begriffe oder die Herkunft von Konzepten unklar sein, so sei an dieser Stelle zentral auf die entsprechende Literatur verwiesen.

Zur *Datenmodellierung* bieten die Bücher von Date [Dat86] oder Vetter [Vet82] eine gute Einführung. Auch Sommerville [Som89] gibt einen guten, kompakten Überblick und verweist auf weitere Quellen von Date und Chen zum Thema *relationale Datenbanken*.

Funktionenmodellierung und *Strukturierte Analyse* werden überblicksartig von Sommerville [Som89] behandelt. Dort findet man Verweise auf weiterführende „Klassiker“ der Strukturierten Analyse [DeM78], der SA/SD-Methode und der Methoden von Jackson [Jac75, Jac83].

Objektorientierte Methoden werden in diesem Handbuch nur am Rande gestreift. Als Einführungen sei auf [Gol+95, Gra95, Tay92, Jac92] verwiesen. Besonders Taylor [Tay92] bietet eine kompakte und leicht verständliche deutsche Einführung in das Thema. Mit dynamischen Aspekten beschäftigt sich besonders Jacobson [Jac92], auf den die Modellierung mit *Use Cases* zurückgeht.

Das Thema *Robuste Software*, das in den Anforderungen an Spezifikationen eine wichtige Rolle gespielt hat wird von Meyer [Mey88] und Walden und Nerson [Wal+95] besonders in den Vordergrund gerückt. Das letztere Werk baut eine komplette Methode um das Thema „*Clusterung*“ auf. Der Einfluß von Meyer auf das Buch ist unverkennbar.

Fehler in Projekten und besonders bei Spezifikationen behandelt das Buch von Webster [Web95] - auch wenn es sich *Pitfalls of Object Oriented Development* nennt - sind viele der „Löcher“, in die man fallen kann, nicht auf die objektorientierte Systementwicklung beschränkt. Besonders die *Management Pitfalls* nicht.

Mit dem Thema gut leserlicher Spezifikationen und Spezifikationen speziell für umfangreiche Systeme befaßt sich Denert. In seinem Buch *Software-Engineering* [Den91] wird auch die später noch zu beschreibende Sachbearbeitermethode eingeführt. In einem Artikel über dokumentenorientierte Spezifikation [Den93] werden die Ideen vorgestellt, die auch diesem Handbuch zugrunde liegen.

I.6. Umfang und Allgemeingültigkeit von Spezifikationen bei VAA

Spezifikationen im Rahmen von VAA müssen nicht nur für **ein** Versicherungsunternehmen Gültigkeit besitzen, sondern für eine Menge von Versicherungsunternehmen. Es ist daher nach dem Kern zu suchen, der für alle Versicherungsunternehmen als stabil und gleich angenommen werden kann. Dieser Teil muß in wiederverwendbarer Form spezifiziert werden und als Baustein für mehr als ein Unternehmen realisierbar sein. Nach aller Erfahrung werden dafür vor allem fachliche Kernobjekte wie Partner, Risiko und Vertrag in Frage kommen. Solche Objekte sind über lange Zeit stabil. Ihre Logik hat oft schon die Lebensdauer von Anwendungssystemen überlebt.

Es gibt allerdings auch Bereiche, in denen sich die Unternehmen aus Wettbewerbsgründen differenzieren möchten. Dies werden besonders die Geschäftsprozesse sein, die individuell je VU gestaltet werden. Sie können daher nicht allgemeinverbindlich spezifiziert werden. Die Dialoge zur Manipulation von fachlichen Objekten können stark durch die Geschäftsprozesse beeinflußt sein. Triviale Pflegedialoge mögen noch allgemein spezifizierbar sein. Komplexe Dialogketten, die aus Geschäftsprozesse resultieren, sind nicht mehr allgemeingültig. Erschwerend kommt noch hinzu, daß viele Geschäftsprozesse einer starken Änderungsrate unterworfen sind.

Was muß also spezifiziert werden?

Nach Denert [Den91] besteht eine Spezifikation aus der Spezifikation der Benutzungsschnittstelle (Dialoge) und der Spezifikation des Anwendungskernes, bestehend aus Funktionen- und Datenmodell, die zu einem Sachbearbeitermodell vereinigt werden. Dazu werden noch Datentypen spezifiziert. 1989, als die Methode [Den91] dokumetiert wurde, waren Workflow-Systeme noch nicht verbreitet. Heute, da Workflow-Systeme üblich sind, sollte man auch die Geschäftsprozesse modellieren. Will ein Versicherungsunternehmen ein Anwendungssystem für sich alleine bauen, wird es also all diese Bereiche eines Anwendungssystems spezifizieren.

Im Rahmen von VAA können davon aber nur die Teile allgemeingültig spezifiziert werden, die auch für alle VU Gültigkeit haben können und die einer moderaten Änderungshäufigkeit unterworfen sind.

Spezifikationsteil	Zu spezifizieren in VAA
Geschäftsprozesse	Ja, aber als Referenzmodell zur Erläuterung. Es besteht nicht die Absicht, Geschäftsprozesse identisch nach der Spezifikation in allen VU zu implementieren
Dialoge, Benutzungsschnittstelle	Nein, Zu viele offene Punkte in der VAA-Architektur und abhängig von VUs
Anwendungskern Sachbearbeiter Datenmodell Datentypen	Ja, in realisierbarer Form nach der Sachbearbeitermethode

Tabelle 1: Umfang und Verbindlichkeit der VAA-Spezifikationen

Tabelle 1 zeigt, welche Elemente einer Spezifikation dieses Handbuch abdecken muß und welchen Grad der Verbindlichkeit diese Spezifikationen jeweils haben. Wirklich verbindlich sind also die Spezifikationen des Anwendungskernes (alias Arbeitsebene, alias Anwendungsbausteine, alias Funktionsbausteine).

 © GDV, sd&m 1996

Dynamisches Verhalten und Use Cases

Methodikern mag bei der obigen Aufstellung auffallen, daß es sich nur um statische Spezifikationselemente handelt und dynamische Darstellungen, wie zum Beispiel Use Cases [Jac92] oder Anwendungsszenarien ausgespart bleiben.

Solche Darstellungen können das Verständnis für komplexe Zusammenhänge extrem fördern. Sie sind aber auch extrem aufwendig zu erstellen und noch aufwendiger zu warten. Sie sollen daher hier nicht vorgeschrieben werden, können aber in beliebiger Form in den Freitext-Teilen jederzeit verwendet werden.

I.7. Überblick über dieses Handbuch

Spezifikationen bestehen nach der Mustergliederung, die in Kapitel VIII beschrieben wird aus drei wesentlichen Teilen (siehe auch Abbildung 5). In einer fachlichen Beschreibung wird in Form von freiem Text beschrieben, was man mit dem spezifizierten System fachlich machen kann und wie man es verwenden und organisatorisch einbinden sollte.

Systemdefinition und Benutzerhandbuch

Dieser erste Teil einer zu erstellenden Spezifikation ist als fachliche Abgrenzung und Benutzerhandbuch zu verstehen und wird typischerweise bis zu 50% des Umfanges der Spezifikation ausmachen. Der Teil dient der fachlichen Verständlichkeit und der Lesbarkeit der Spezifikation. Viele Fachabteilungen werden nur diesen Teil einer Spezifikation lesen, und den Rest übergehen.

Formale Spezifikation

Der zweite Hauptteil einer zu erstellenden Spezifikation besteht aus einem „formalen“ Modell, so wie man es aus Spezifikationen, zum Beispiel nach der strukturierten oder der objektorientierten Analyse gewohnt ist. Die einzelnen Teile hierzu werden in Kapitel II bis VI beschrieben. Dieser Teil dient dazu, ein ausreichend formales Modell des Spezifikationsgegenstandes für Softwareentwickler zur Verfügung zu stellen. Entwickler werden also beide Teile lesen. Den informellen ersten Teil einer Spezifikation, um sich über die fachlichen Zusammenhänge zu informieren und den formalen Teil, um Software nach der Spezifikation präzise konstruieren zu können.

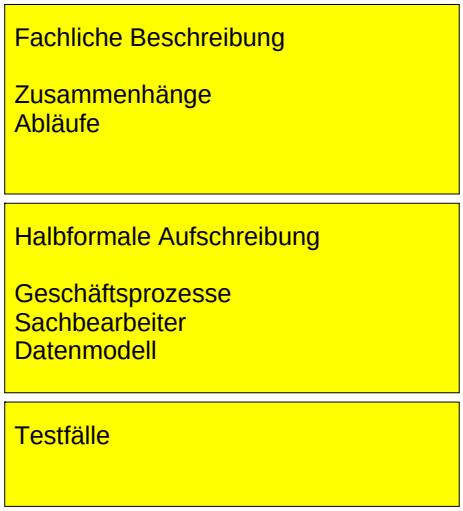


Abbildung 5: Haupteile der Spezifikation

Dieses mehrgleisige Vorgehen ist aufwendiger als manche Spezifikationen, die wir heute in Unternehmen finden. Oft werden Softwareentwickler dazu tendieren, nur den formalen Teil zu schreiben. Die Folge kann dann sein, daß die entstehenden Spezifikationen von Fachabteilungen

nicht mehr verstanden werden. Fachabteilungen werden oft dazu neigen, lediglich ihre fachlichen Anforderungen grob als Freitext, also in beliebiger Form, zu dokumentieren. Die Folge ist dann oft, daß Softwareentwickler nicht die Präzision vorfinden, die sie benötigen, um unzweideutig aus der Spezifikation Software zu entwickeln. Es bleibt also nur, sowohl die formale als auch die informelle Spezifikation zu schreiben. Dann sind sicher beide Lesergruppen ausreichend berücksichtigt. Der Preis dafür ist ein erhöhter Aufwand gegenüber qualitativ schlechteren Spezifikationen.

Testfälle

Als dritter, optionaler Teil können auch Testfälle spezifiziert werden. Dazu sollen beim derzeitigen Stand von VAA aber noch keine „Richtlinien“ vereinbart werden. Eine Beschreibung für ein solches, optionales Kapitel kann später ergänzt werden. Die Testfälle sind deshalb auch in der Mustergleiderung nicht weiter erwähnt. In manchen Fällen, zum Beispiel bei Modulen mit viel Berechnungsfunktionalität ist die Angabe einiger Testfälle und Proberechnungen jedoch auf jeden Fall sinnvoll.

Aufbau des formalen Teils

Dieses Handbuch stellt zunächst die einzelnen Modellierungselemente für den formalen Teil der fachlichen Spezifikationen nacheinander vor. Es sind dies:

- Geschäftsprozeßmodellierung - siehe Kapitel II
- Das Sachbearbeitermodell und Geschäftsvorfälle für die Funktionenmodellierung - siehe Kapitel III.
- Datenmodellierung, so wie sie im Rahmen von VAA verwendet wird - siehe Kapitel IV. Dazu gehört auch noch das Konzept und die Modellierung von Datentypen in Kapitel VI.

Abbildung 6 gibt einen Überblick über die wesentlichen Beziehungen zwischen den einzelnen Teilen des formalen Spezifikationsteils.

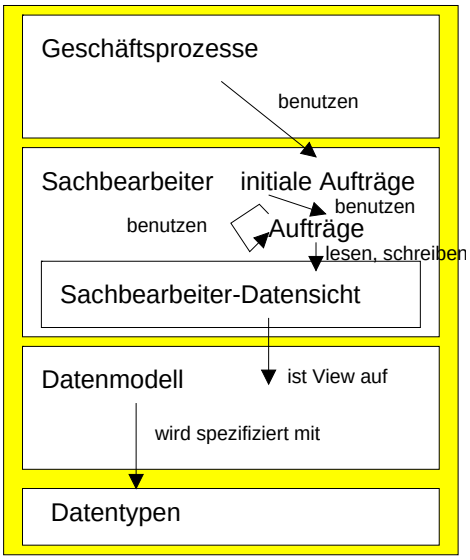


Abbildung 6: Beziehungen zwischen den Teilen des formalen Spezifikationsteils

 © GDV, sd&m 1996

Mustergliederung

Fachliche VAA-Spezifikationen werden nach einer Mustergliederung aufgebaut. Diese wird in Kapitel VIII beschrieben. In diesem Kapitel wird auch beschrieben, wie man den fachlichen Freitext-Teil, also das Benutzerhandbuch, aufbauen sollte.

Gewichtung des Umfanges

Die Tatsache, daß die formalen Teile der Spezifikation hier ausführlicher beschrieben werden, als die informellen und daß ihre Beschreibung an erster Stelle steht ist keine Aussage über die Wichtigkeit des formalen Teils. Der formale Teil ist wichtig - ohne eine ausreichende und leicht verständliche fachliche Beschreibung in Form von freiem Text - also ohne den ersten Spezifikationsteil ist er jedoch total wertlos.

II. Geschäftsprozesse

II.1. Spezifikation von Geschäftsprozessen

Das abteilungsübergreifende Denken in Geschäftsprozessen sollte eigentlich immer eine Selbstverständlichkeit gewesen sein. Mit der teilweise starken Taylorisierung von Arbeitsabläufen ist es jedoch lange Zeit in den Hintergrund getreten.

Seit etwa 15 Jahren bemerken wir eine starke Gegenbewegung zum Taylorismus. Business Process Reengineering [Ham+94] wurde als Mode- und Wunderwaffe für die Rationalisierung von Abläufen entdeckt. Gleichzeitig bieten sich mit dem Aufkommen von Vorgangssteuerungs- und Workflow-Systemen neue Möglichkeiten, die Bearbeitung von komplexen Vorgängen besser als bisher zu unterstützen und auch zu kontrollieren. Positive Effekte der neuen Softwaregeneration sind der Wegfall von Transportzeiten, die Reduktion von Liegezeiten und zum Beispiel die Möglichkeit, Durchlaufzeiten problemlos zu überwachen, um nur einige zu nennen.

Ferner bieten Workflow- und Dokumentenmanagement-Systeme Möglichkeiten, die bei aktengestützter Bearbeitung schlicht nicht denkbar waren. Elektronische Akten können quasi kostenlos kopiert werden. Sie stehen parallel überall zur Verfügung. Sie kennen ihren eigenen Weg durch die Organisation. Sie können selbst Termine überwachen.

Natürlich sollen diese Vorteile auch im Rahmen von VAA genutzt werden. Es sollte also im Rahmen von fachlichen VAA-Spezifikationen möglich sein, Geschäftsprozesse darzustellen und workflowbasierte Systeme zu spezifizieren. Die Geschäftsprozeßmodellierung dient bei VAA also mehreren Zwecken:

- Gerade bei komplexen Abläufen, die durch EDV unterstützt werden sollen, können Geschäftsprozeßmodelle dem Verständnis der Spezifikation dienen. Es wird klar, wozu und wann Funktionalität benötigt wird.
- Die Geschäftsprozeßmodelle sollen als Modell der Geschäftsprozesse eines repräsentativen VU dienen können. Nicht jedes VU wird sie identisch implementieren. Sie müssen aber dem Anspruch genügen, daß man auf der „grünen Wiese“ ein VU mit diesen Geschäftsprozessen eröffnen und betreiben könnte. Die spezifizierten Geschäftsprozesse müssen das Arbeitsgebiet also ausreichend abdecken und in sich schlüssig sein.
- Geschäftsprozeßmodelle dienen dazu, festzustellen, ob die VAA Spezifikationen die Prozesse auch im ausreichenden Maße mit DV-Funktionalität unterstützen. Will man einen Arbeitsschritt durch DV unterstützt wissen, so muß im Sachbearbeitermodell entsprechende Funktionalität vorhanden sein.

Geschäftsprozeßmodellierung als Grundlage für Business Process Reengineering oder als Grundlage für die Entscheidung der Frage, mit wieviel DV ein Prozeß in einem VU unterstützt werden soll, tritt bei VAA in den Hintergrund. Die im folgenden dargestellten Methoden können jedoch von jedem VU auch für solche Zwecke eingesetzt werden.

Bevor jedoch die Modellierungsmethode geschildert wird, sollen hier noch einige Anmerkungen folgen, die die VAA-Spezifikationsteams vor teuren Fehlern bei der Geschäftsprozeßmodellierung bewahren sollen.

VAA bietet Referenzmodelle, ist aber keine Vorschrift

Die Geschäftsprozeßmodelle, die im Rahmen von VAA-Spezifikationen entstehen, sollen Referenzmodelle sein und nicht etwa Modelle, die für jedes Versicherungsunternehmen unverändert einsetzbar sind. Sollten verbindliche, einheitliche Modell entwickelt werden, wären endlose Diskussionen zu erwarten. Der Ausgang wäre fast sicher negativ. Statt dessen sollen Ge-

 © GDV, sd&m 1996

schäftsprozeßmodelle in VAA-Spezifikationen für ein sinnvolles, einigermaßen repräsentatives VU spezifiziert werden. Einheitliche Geschäftsprozesse kann man nur für Hilfsfunktionen erwarten, die nicht wettbewerbsrelevant sind.

Würde sich VAA zum Beispiel als Ziel setzen, auch die Geschäftsprozesse für die Buchhaltung von Versicherungsunternehmen zu spezifizieren, dann ist zu erwarten, daß man sich auf ein einheitliches Vorgehen einigen kann. In der Praxis werden solche Aufgaben aber schon zum Beispiel von SAP oder anderer Standardsoftware unterstützt. Man kann dort beobachten, daß sich Unternehmen auch an die Geschäftsprozesse anpassen, die von Standardsoftware diktiert werden. Man kann jedoch auch beobachten, daß die Bereitschaft dazu in wettbewerbsentscheidenden Betriebsfunktionen nicht vorhanden sein wird.

Wann Geschäftsprozeßmodellierung sinnlos ist

Die Tatsache, daß in der VAA-Spezifikationsmethode für fachliche Bausteine ein Kapitel Geschäftsprozeßmodelle vorhanden ist, kann dazu verleiten, überflüssige und triviale Geschäftsprozeßmodelle zu spezifizieren. Es ist keinesfalls für jeden Anwendungsbaustein nötig, ein zugehöriges Geschäftsprozeßmodell zu spezifizieren.

Nur aus der Tatsache, daß Geschäftsprozeßmodellierung heute Mode ist, kann man nicht folgern, daß Geschäftsprozeßmodellierung immer nötig und sinnvoll ist. Es gibt durchaus EDV-Anwendungen, denen keine sinnvollen Geschäftsprozesse zugrunde liegen. Betrachtet man zum Beispiel ein Partnersystem, so wäre es ineffizient mit der Spezifikation des Partner-Sachbearbeiters gleichzeitig alle Geschäftsprozesse zu spezifizieren, in denen irgendwann der Partner-Sachbearbeiter benutzt wird. Man würde dafür zu lange brauchen. Außerdem handelt es sich um „atomare Buchungen“ auf einer Datenbasis, die keine längere Vorgangsverfolgung erfordern.

Im Fall der Schadensbearbeitung ist ein Geschäftsprozeßmodell dagegen sinnvoll. Ein „Schaden“ kann selbst als ein Vorgang betrachtet werden, der von Sachbearbeiter zu Sachbearbeiter wandert und dabei von der Schadensmeldung bis zur Zahlung oder Ablehnung eine komplizierte Wegstrecke zurücklegt. Es wäre nicht verkehrt sich einen Schaden als elektronischen Ordner vorzustellen, der nach bestimmten, in einem Geschäftsprozeßmodell festzulegenden Regeln, weitergereicht wird.

Man kann also positiv formulieren, daß Geschäftsprozeßmodellierung vor allem dann sinnvoll ist, wenn folgende Kriterien erfüllt sind:

- Es werden Arbeitsabläufe unterstützt, die sich über einen längeren Zeitraum erstrecken.
- Die Vorgänge haben mehrere Arbeitsschritte
- Die Arbeitsschritte werden von mehreren Bearbeitern durchgeführt
- Man kann sich den Vorgang als eine Akte vorstellen, die durch eine Organisation wandert.

Sind diese Kriterien ganz oder teilweise für eine VAA-Komponente erfüllt, so ist dies ein Anhaltspunkt dafür, daß ein Geschäftsprozeßmodell erstellt werden sollte.

Waren wir ohne Geschäftsprozeßmodellierung früher rückständig?

Was unterscheidet nun ein modernes Schadensystem auf der Basis einer Vorgangssteuerung von einem Partnersystem? Ein Partnersystem ist, wie zum Beispiel auch eine Vertragsverwaltung, ein bestandsführendes System. Es erfolgen Buchungen bzw. Änderungen - danach liegt der Bestand wieder solange, bis ihn irgend jemand wieder anfaßt. Die frühen DV-Systeme, mit denen in den frühen 60er und 70er Jahren gerade auch in Versicherungen erste Rationalisierungserfolge erzielt wurden, waren vor allem Bestandsverwaltungssysteme. Sie entsprachen dem damaligen Stand der DV-Technik.

Durch den Fortschritt der Technik ist es heute nun auch möglich Vorgangsbearbeitungssysteme mit elektronischen Akten zu spezifizieren und zu implementieren. Diese haben jedoch eine andere Qualität als Bestandsverwaltungssysteme.

Es ist also keineswegs nur Rückständigkeit, daß früher keine Geschäftsprozeßmodelle spezifiziert wurden - weil Anwendungen spezifiziert wurden, für die großteils keine Geschäftsprozeßanalysen benötigt wurden. Genauso ist es nicht rückständig, kein Geschäftsprozeßmodell zu spezifizieren, wenn man eine bestandsführende Anwendung spezifiziert, die in die Klasse der frühen, einfachen Anwendungen gehört, die aber im Rahmen von VAA spezifiziert werden muß, weil VAA Spezifikationen für komplette VUs bieten muß.

Konzentration auf DV-gestützte Systeme

Oft findet man in Geschäftsprozeßmodellen Prozesse wie „*Strategische Planung*“, „*Liquidität sichern*“ oder „*Unternehmensplanung*“. Bei solchen Prozessen handelt es sich typischerweise um Vorgänge, die nicht mit EDV unterstützt werden sollen. Ein Geschäftsprozeßmodell für solche Prozesse sollte auf keinen Fall im Rahmen von VAA erstellt werden - es wäre Ressourcenverschwendung.

Aufwand für die Geschäftsprozeßmodellierung

Man kann beobachten, daß der Aufwand, der in Geschäftsprozeßanalysen investiert wird, erheblich ist. Wenn für die Analyse und Dokumentation eines umfangreicheren Geschäftsprozesses ein bis zwei Bearbeiterjahre verwendet werden, ist dies nicht ungewöhnlich. Beispiele für umfangreichere Geschäftsprozeßmodelle sind öffentlich zugänglich und sollten auch genutzt werden, wenn VAA-Spezifikationsteams gute Beispiele für Geschäftsprozeßmodellierung suchen. Als exemplarische Quellen für frei verfügbare Geschäftsprozeßmodelle erheblichen Umfanges seien hier genannt:

- *Das V-Modell, Der Standard für die Softwareentwicklung mit Praxisleitfaden* [VMo93].
  Es handelt sich hierbei um ein Modell für den Geschäftsprozeß Softwareentwicklung, das als Beispiel für einen komplexen, tief geschachtelten Prozeß mit vielen Beteiligten dienen kann. Auch wenn der Inhalt nicht direkt betriebswirtschaftlich/fachlich ist, fällt die Übertragung gerade Softwareentwicklern leicht

- *Geschäftsprozeßmodell BMW/SM-15* der FAST e.V. - siehe [Fas95a, Fas95b]. Beschrieben wird ein kompletter Geschäftsprozeß der Teilefertigung in einem BMW-Werk. Der Bericht ist mit ca. 200 Seiten recht umfangreich und kann ebenfalls als Referenzbeispiel herangezogen werden.

- *ARIS-Referenzmodell*: Für den industriellen Bereich gibt es Daten- und Geschäftsprozeßmodelle. Das ARIS-Geschäftsprozeßmodell kann als Beispiel im Buchhandel auf CD erworben werden. In dem Modell kann man mit einer read-only Version des ARIS-Toolset navigieren.

Bei der Geschäftsprozeßmodellierung im Rahmen von VAA wird es darauf ankommen, nicht zuviel Aufwand in die Geschäftsprozeßmodellierung zu investieren. Detaillierte Bücher von 200-300 Seiten sind nicht nur zu aufwendig - sie sind auch kaum noch zu überblicken.

 © GDV, sd&m 1996

Richtige Granularität

Man findet oft Geschäftsprozesse die zu fein oder zu grob spezifiziert sind. Einen Anhaltspunkt für die unterste Ebene, also einen Arbeitsschritt, bietet die allgemeine, betriebswirtschaftliche Organisationslehre. Die Organisationslehre (siehe z.B. [Wit93]) kennt Kriterien für die Modellierung und vor allem für die Abgrenzung von *Arbeitsschritten*:

- Ein Wechsel des *bearbeiteten Objektes* ist ein Indikator für einen neuen *Arbeitsschritt*
- Ein Wechsel des *Bearbeiters* ist ein Indikator für einen neuen *Arbeitsschritt*
- Ein Wechsel des *Arbeitsmittels* ist ein Indikator für einen neuen *Arbeitsschritt*
- Ein Wechsel der *Arbeitsphase* (Planung, Ausführung, Kontrolle) ist ein Indikator für einen neuen *Arbeitsschritt*

Als Zerlegungsstufen werden in der Organisationslehre Verantwortungsbereiche und Teilaufgaben angeboten. Ein Blick in die klassische, betriebswirtschaftliche Organisationslehre kann also lange Modellierungsdiskussionen ersparen.

Es macht insbesondere keinen Sinn, Vorgänge, die ununterbrochen an einem Arbeitsplatz stattfinden, in winzige, atomare Teilschritte zu zerlegen. Die resultierenden Modelle wären unverständlich, der Aufwand wäre immens hoch. Umgekehrt macht es keinen Sinn einen kompletten Schadensvorgang, der mehrfach unterbrochen werden kann und von mehreren Bearbeitern erledigt wird, als einen Arbeitsschritt zu spezifizieren.

Teilprozesse versus Arbeitsschritte

Über die Grenzen von Geschäftsprozessen, Teilprozessen und Arbeitsschritten wird oft endlos diskutiert. Die Erfahrung zeigt, daß solche Diskussionen völlig sinnlos sind, weil sie keine Werte schaffen. Die Grenzziehung bei der hierarchischen Zerlegung von Geschäftsprozessen kann völlig willkürlich erfolgen. Eckpfeiler ist dabei der Gesamtprozeß. Über seinen Umfang muß meist nicht lange diskutiert werden. Zweiter Eckpfeiler sind die atomaren Arbeitsschritte für die oben brauchbare Abgrenzungskriterien gegeben wurden.

Die Zerlegung in hierarchische Ebenen sollte so erfolgen, daß die Regeln menschlicher Wahrnehmung nicht verletzt werden. Psychologen haben herausgefunden, daß Menschen auf einer Ebene 7 +/- 2 Objekte gut verstehen können.

Tapetenartige Geschäftprozeßmodelle mit 30 oder 40 Arbeitsschritten auf einer Ebene sollten also sicher in Hierarchiestufen zerlegt werden. Geschäftsprozesse, die aus jeweils genau einem Arbeitsschritt bestehen, sind aller Wahrscheinlichkeit nach keine und sollten besser nicht modelliert werden. Wo die Hierarchiestufen gezogen werden ist ziemlich belanglos, solange die Grundregeln der menschlichen Wahrnehmung eingehalten werden.

Wieviele Attribute benötigt man für die Beschreibung von Geschäftsprozessen?

Für die Beschreibung von Geschäftsprozessen kann man eine Unmenge von Beschreibungsattributen angeben. Es gibt Untersuchungen [Fas95b], die eine ziemlich vollständige Liste der Möglichkeiten enthalten. Eine Methode, die eine möglichst große Anzahl solcher Beschreibungsattribute enthält, mag die genaueste sein - aber nicht unbedingt die beste.

Praktische Projekterfahrung mit Geschäftsprozeßmodellen hat gezeigt, daß vor allem solche Geschäftsprozeßmodelle sich als nützlich in Projekten bewährt haben, die einfach und gut verständlich waren. Je komplexer und formaler die Geschäftsprozeßmodellierung war, desto weni-

ger wurde sie von Fachabteilungen akzeptiert und war in das Reich der Organisationsabteilungen verbannt. Das sollte mit den VAA - Geschäftsprozeßmodellen nicht passieren.

Statt die Beschreibungselemente theoretisch einzuführen wird im folgenden ein Beispielprozeß beschrieben. Die einzelnen Elemente zum Nachschlagen werden dann im Abschnitt II.II.3 genauer beschrieben.

II.2. Beispiel

Als Beispiel soll eine Kreditbearbeitung, zum Beispiel in einer Bank, gewählt werden. Ein versicherungsfremdes Beispiel wird hier verwendet, um Diskussionen über die versicherungsfachliche Richtigkeit zu vermeiden. Das Beispiel dient lediglich zur Demonstration der Methode.

3.1 **Geschäftsprozeß** Baukreditbearbeitung

Auslöser Kundenanfrage Immobilienfinanzierung

Der Geschäftsprozeß wird durch eine Anfrage des Kunden in einer Geschäftsstelle initiiert.

Beschreibung

Die Baukreditbearbeitung ist einer der wesentlichen Geschäftsprozesse des Filialgeschäftes der XY-Bank. Um im Wettbewerb bestehen zu können, darf eine Kreditbearbeitung für die Immobilienfinanzierung in einfachen Fällen nicht länger als 3-5 Tage in Anspruch nehmen. In besonders einfachen Fällen bis 200.000 DM soll auch eine Genehmigung an nur einem Tag möglich sein, wenn alle Unterlagen vorliegen.

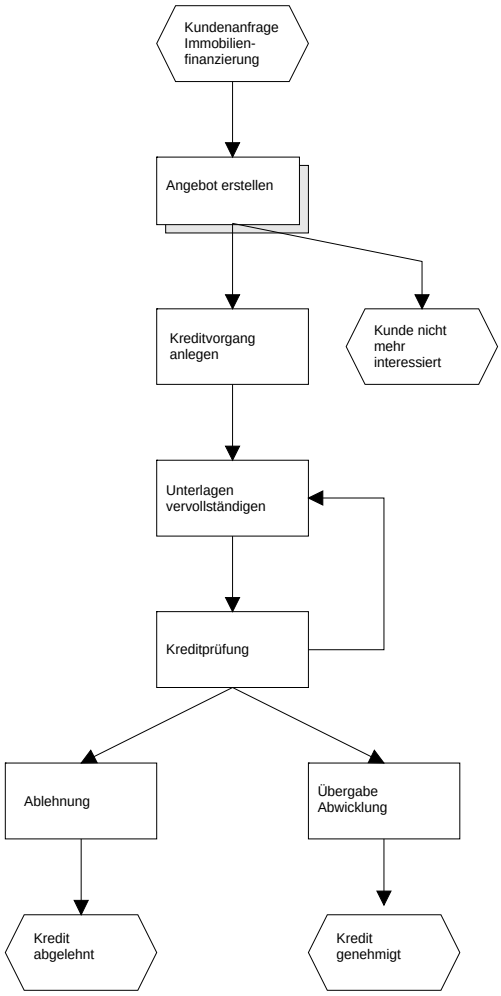


Abbildung 7: Geschäftsprozeß Baukreditbearbeitung

 © GDV, sd&m 1996

Abbildung 7 zeigt den Gesamtprozeß der Kreditbearbeitung. Zunächst wird ein Angebot erstellt - Dieser Teilprozeß unterteilt sich in mehrere Stufen.

Ist der Kunde mit dem Angebot grundsätzlich einverstanden, wird der Kreditsachbearbeiter einen Kreditvorgang anlegen. Er wird im nächsten Schritt den Kunden um Unterlagen bitten (Arbeitsschritt Unterlagen vervollständigen). Meint der Kreditsachbearbeiter, daß die Unterlagen ausreichend vollständig sind, wird der Vorgang an die zentrale Kreditprüfung (Arbeitssschritt Kreditprüfung) weitergeleitet.

Hier kann der Antrag abgelehnt werden. Der Kreditsachbearbeiter erhält den Vorgang dann zurück und muß die Ablehnung dem Kunden kommunizieren (Arbeitsschritt Ablehnung).

Wenn der Antrag genehmigt wurde, geht er zurück an den Kreditsachbearbeiter, der die Abwicklung einleitet (Übergabe Abwicklung).

Die zentrale Kreditprüfung kann auch Unterlagen nachfordern. In diesem Fall geht der Vorgang zurück an den Kreditsachbearbeiter in den Arbeitsschritt Unterlagen vervollständigen.

Häufigkeit

Jährlich werden ca. 20.000 Kreditbearbeitungen durchgeführt. Die dabei vergebenen Kredite haben ein Volumen von ca. 25 Mrd. DM.

Teilprozess Angebot erstellen

Beschreibung siehe 3.1.1

Arbeitsschritt Kreditvorgang anlegen

Auslöser

Bemerkung: Auslöser ist hier der Abschluß des vorhergehenden Arbeitsschrittes - ein solch „trivialer Auslöser“ sollte besser gar nicht erst beschrieben werden.

Bearbeiter Kreditsachbearbeiter

Arbeitsmittel EDV-System Kreditbearbeitung

Beschreibung

Ein Angebot ist erstellt - der Kunde möchte das Angebot annehmen. Die Eckdaten des Angebotes werden im Kreditbearbeitungssystem erfaßt (**Sachbearbeiterauftrag** Kreditvorgang.erfassen).

Arbeitsschritt Unterlagen vervollständigen

Bearbeiter Kreditsachbearbeiter

Arbeitsmittel Scanner, EDV-System Kreditbearbeitung

Beschreibung

Abhängig von den Eckdaten ermittelt das Kreditbearbeitungssystem die benötigten Unterlagen, die von Wertgrenzen abhängen. (**Sachbearbeiterauftrag** Kreditvorgang.UnterlagenErmitteln).

Unterlagen werden vom Kunden eingereicht und mit dem **Sachbearbeiterauftrag** Kreditvorgang.UnterlageZuordnen dem Vorgang zugeschrieben. Sind alle Unterlagen vorhanden, so erfolgt eine Freigabe mit dem **Sachbearbeiterauftrag** Kreditvorgang.UnterlageZurPrüfungWeiterreichen.

Arbeitsschritt Kreditprüfung

Bearbeiter zentrale Kreditprüfung

Arbeitsmittel EDV-System Kreditbearbeitung, Gutachten, weiters

Beschreibung

Der Sachbearbeiter in der zentralen Kreditprüfungsstelle erhält den Vorgang zur Prüfung. Abhängig von den Eckdaten erhält er Prüfaufforderungen vom System. Unter anderem auch die Frage, ob Grundbucheinsicht genommen werden muß, Wertgutachten erstellt werden müssen und weiteres.

Der Prüfer kann die Krediterteilung befürworten (mit dem **Sachbearbeiterauftrag** Kreditvorgang.befürworten) oder ablehnen (mit dem **Sachbearbeiterauftrag** Kreditvorgang.ablehnen). Der Kreditvorgang geht an die Filiale zurück, wo die weitere Bearbeitung erfolgt.

Die Beschreibung der zwei fehlenden Arbeitsschritte ist jetzt einfach und muß hier im Beispiel nicht mehr erfolgen.

Wichtig für das Beispiel ist es noch, einen Teilprozeß zu beschreiben, also die Zerlegung einer „größeren Aktion“. Dabei kann man die Konsistenzbedingung bestätigt finden, daß alle Ein- und Ausgänge in den Teilprozeß „Angebot erstellen“ auch innerhalb dieser Box behandelt werden müssen. Sonst wäre die Prozeßbeschreibung nicht korrekt verfeinert.

3.1.1 **Teilprozess** Angebot erstellen

Auslöser Kundenanfrage Immobilienfinanzierung

Beschreibung

Als erster Schritt erfolgt ein Beratungsgespräch. Ist der Kunde nicht mehr interessiert, so ist der Kreditvorgang leider beendet. Nach der Beratung erfolgt eine Angebotserstellung mit dem PC. Auch nach dem Angebot kann es passieren, daß der Kunde das Angebot ablehnt.

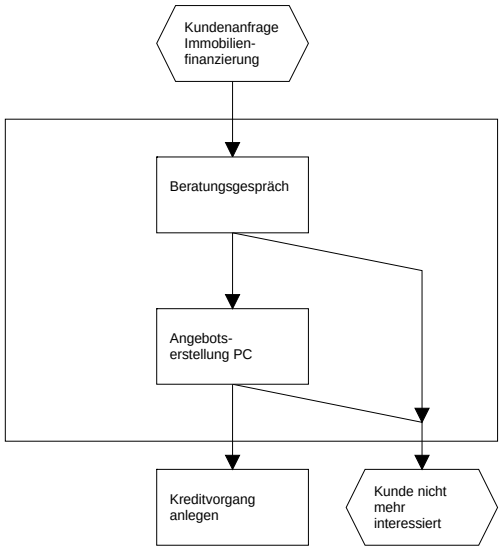


Abbildung 8: Teilprozeß Angebot erstellen

 © GDV, sd&m 1996

Arbeitsschritt Beratungsgespräch

Bearbeiter Kreditsachbearbeiter

Arbeitsmittel Werbemittel des Zentralvertriebs Baukredite

Beschreibung

Der Finanzierungsbedarf, die Einkommenssituation und die steuerliche Situation des Kunden werden ermittelt. Es werden dem Kunden verschiedene Finanzierungsmodelle unserer Bank vorgestellt. Der Kunde wird über seine Präferenzen (risikobewußt, risikoavers) befragt.

Arbeitsschritt Angebotserstellung PC

Bearbeiter Kreditsachbearbeiter

Arbeitsmittel Programm PC-Baufinanz der AllFinanz

Beschreibung

Die Präferenzen und Eckdaten des Kunden werden erfaßt. Es wird ein rechtsverbindliches Angebot erstellt, das der Kunde zum Mitnehmen erhält.

II.3. Darstellung des Beispiels

Bei der Beschreibung wurde Wert auf Einfachheit, Verständlichkeit und Lesbarkeit gelegt. Ein Geschäftsprozeß hat nur vier Beschreibungsmittel. Seinen Namen, eine Beschreibung in Form eines Überblicks, eine Aussage über die Häufigkeit und seine Teilprozesse und Arbeitsschritte.

Arbeitsschritte können nicht weiter zerlegt werden - siehe dazu die Kriterien, die schon oben diskutiert wurden. Teilprozesse werden selbst wieder beschrieben wie ein Geschäftsprozeß - sie unterscheiden sich von Arbeitsschritten durch ihre Zerlegbarkeit. Darüber hinaus gibt es keine Begriffe. Sollen weitere Attribute dringend betrachtet werden, so kann das in der Beschreibung in Form von Freitext erfolgen.

Text statt Tabellen

Durch diesen „minimalistischen“ Ansatz wird das Ausfüllen von Formularen vermieden. In ersten Ansätzen für ein VAA-Methodenhandbuch wurden stets Tabellen angegeben, um Geschäftsprozesse oder Arbeitsschritte zu beschreiben. Das definieren von Tabellen ist einfach für den Methodiker. Das Ausfüllen von Tabellen ist aber aufwendig für den Spezifikateur. Tabellen haben konkret folgende Nachteile:

- Das Ausfüllen von Tabellen mit vielen Attributen ist aufwendig - Tabellen sind in Textprogrammen meist schwieriger zu handhaben, als normaler Fließtext.
- Die Lesbarkeit leidet darunter, daß es viele Tabelleneinträge der Art: „kein Eintrag“, „Eintrag sinnlos“, „keine Angaben“ gibt.
- Textprogramme werden durch Tabellen übermäßig belastet. Bei den meisten handelsüblichen PC-Textprogrammen leidet ab einer Dokumentgröße von mehr als 20-30 Seiten die Handhabbarkeit der Dokumente, weil der Seitenumbruch im Hintergrund und das Layout von Tabellen das Textverarbeitungsprogramm stark belasten.
- Für Freitexte kann man zur Not auch einen simplen ASCII-Editor verwenden. Bei Tabellen ist das schlechter möglich.
- Tabellen sind schlechter maschinell durch Makros oder Textverarbeitungswerkzeuge (AWK, SED, Perl) auswertbar als Freitext mit Schlüsselworten.

Aus diesen Gründen verwenden wir Schlüsselworte und Freitext statt Tabellen. Nicht benötigte Abschnitte werden gar nicht erst geschrieben oder aus einem Textbaustein bei der Bearbeitung

gelöscht. Das Ergebnis sind besser lesbare Texte, die sich leichter bearbeiten und auswerten lassen.

Wenige Bilder plus Text

Ein Bild sagt mehr als 1000 Worte - aber was sagen 1000 Bilder? Es ist sinnvoll, die Prozeßbilder als Illustration für die Texte zu verwenden. Das Mischen von Bildern und Texten ist daher unbedingt erforderlich, da es Leser gibt, die primär durch Bilder verstehen und andere die primär durch Lesen verstehen. Auch ergänzen sich beide Darstellungen. Man findet häufig aus technischen Gründen Darstellungen, bei denen Bilder und Texte getrennt sind - solche Darstellungen sind abzulehnen.

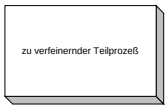
Keine redundanten Auflistungen

In vielen Spezifikationsdokumenten, vor allem auch bei Geschäftsprozessen, findet man lange Listen von benutzten Entitäten, Funktionen, Dialogen etc. Solche Darstellungen sollten entweder aus den Texten generiert werden, da man die Benutzung von Objekten im Text kennzeichnen kann, oder aber man sollte sie weglassen. Kompakte Übersichten - also Inhaltsverzeichnisse - sind immer dann gut, wenn sie automatisch aktualisiert werden. Wenn sie manuell gepflegt werden müssen, kostet das zuviel Zeit und es ist i.A. nicht damit zu rechnen, daß die Verzeichnisse stimmen. Also wird von solchen redundanten Auflistungen Abstand genommen.

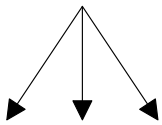
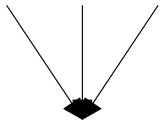
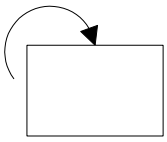
II.4. Beschreibungselemente

Geschäftsprozesse werden durch Graphiken und textuelle Beschreibungen der einzelnen Teilprozesse beschrieben. Diese können selbst wieder eine Graphik enthalten, wenn sie zusammengesetzt sind. Die verwendbaren graphischen und textuellen Elemente werden im folgenden beschrieben.

II.4.1. Graphische Beschreibungselemente

Bezeichnung	Symbol	Beschreibung
Ereignis, Auslöser Zustand, Ergebnis		Dieses Symbol beschreibt sowohl die einen Geschäftsprozeß bzw. Teilprozeß auslösenden Ereignisse, als auch die Zustände bzw. Ergebnisse, die vom Geschäfts-/Teilprozeß erreicht werden. Es wurde ein einheitlichen Symbol gewählt, da Ergebniszustände wiederum als Auslöser fungieren können. Die sechseckige Form wurde gewählt, um einfacher Text in diesem Symbol einstellen zu können.
zu zerlegender Teil- oder Geschäftsprozeß		Der in diesem Symbol beschriebene Teilprozeß wird in einer weiteren Beschreibung detaillierter dargestellt. Dies wird durch die Schattierung kenntlich gemacht.
Arbeitsschritt		Dieses Symbol beschreibt einen Arbeitsschritt als kleinste darstellbare Einheit in der vorgeschlagenen grafischen Beschreibungsnotation eines

 © GDV, sd&m 1996

Bezeichnung	Symbol	Beschreibung
		Geschäftsprozesses.
Folgt auf, setzt voraus		Ein einzelner Pfeil beschreibt die zeitliche und logische Folge von einzelnen Arbeitsschritten und Teilprozessender grafischen Beschreibung.
Parallelität		Auf einen Arbeitsschritt können parallel mehrere andere folgen.
Synchronisation		Es kann erforderlich sein, daß mehrere Schritte erledigt sein müssen, damit der nächste beginnen kann. Komplexere Bedingungen (und, oder, Zeitbedingungen etc.) werden in der textuellen Erläuterung zum Bild beschrieben.
Iteration, Interaktion		Der Geschäfts- oder Teilprozeß wird wiederholt ausgeführt. Dies ist z.B. dann der Fall, wenn aufgrund unzureichender Daten eine Angabe von Zusatzdaten zur Eingrenzung z.B. eines einzelnen Kunden aus einer Kundenliste erforderlich ist.

II.4.2. Textuelle Beschreibungselemente

II.4.2.1. Geschäftsprozeß, Teilprozeß

Ein Geschäftsprozeßmodell besteht aus der Beschreibung mindestens eines Geschäftsprozesses **(Regel 1.)**

Ein Geschäftsprozeß wird genauso beschrieben wie ein Teilprozeß **(Regel 2.)**.

Zu jedem Teilprozeß muß ein Name (Identifikator) vergeben werden. Es kann ein Auslöser angegeben werden - ist dieser klar, so muß er nicht angegeben werden. Eine Beschreibung in Form von Freitext ist verpflichtend. Die Beschreibung soll eine Graphik enthalten. Zu einem Teilprozeß kann eine Häufigkeit angegeben werden, um Mengenschätzungen möglich zu machen. Nach dem Teilprozeß werden alle diejenigen Teilprozesse genannt, die in ihm enthalten sind. Wenn es nicht mindestens einer ist, kann es sich logischerweise nicht um einen Teilprozeß handeln. Es muß ein (unteilbarer) Arbeitsschritt vorliegen **(Regel 3.)**.

Formale Syntax

(Regel 1.) *Geschäftsprozeßmodell ::= Geschäftsprozeß +*

(Regel 2.) *Geschäftsprozeß ::= Teilprozeß.*

(Regel 3.) *Teilprozeß* ::=

Geschäftsprozeß *Identifikator*

[**Auslöser** *Identifikator*

Freitext]

Beschreibung

Freitext

[**Häufigkeit**

Freitext]

(Teilprozeß | Arbeitsschritt) +.

II.4.2.2. Arbeitsschritt

Für einen Arbeitsschritt muß ein Name festgelegt werden. Die Angabe eines Bearbeiters, einer bearbeitenden Instanz oder einer sonstigen Beschreibung für den Bearbeiter ist oft sinnvoll. Wenn sich der Bearbeiter aus dem Kontext ergibt, kann er weggelassen werden. Die Angabe der verwendeten Arbeitsmittel, wie Dokumente, EDV-Unterstützung und anderes ist oft ebenfalls sinnvoll. Wenn sie sich aus dem Kontext ergibt, kann sie ebenfalls weggelassen werden. Eine textuelle Beschreibung des Arbeitsschrittes ist allerdings verpflichtend. **(Regel 4.)**

Formale Syntax

(Regel 4.) *Arbeitsschritt* ::=

Arbeitsschritt *Identifikator*

[**Auslöser** *Identifikator*

Freitext]

[**Bearbeiter** *Freitext*]

[**Arbeitsmittel**

Freitext]

Beschreibung

Freitext

II.5. Mögliche weitere Beschreibungsattribute

Die Liste der weiteren Attribute, die man zu einer Prozeßbeschreibung angeben kann, ist schlicht unendlich. So kann man die komplette Aufbauorganisation seines Unternehmens beschreiben - was für ein Referenz-Versicherungsunternehmen natürlich schwierig ist.

Man kann Kosten- und Zeitattribute angeben mit dem Ziel, Kosten- und Durchlaufzeitaussagen zu gewinnen - was im Rahmen von VAA ebenfalls keinen Sinn macht, weil jedes VU seine Geschäftsprozesse kosten- und zeitmäßig selbst optimieren muß.

 © GDV, sd&m 1996

Man kann weitere Steuerungsinformationen, wie Ereignisse, Asynchronität, Nebenläufigkeit, Rendezvouskonzepte spezifizieren. Das geschieht typischerweise mit dem Ziel eines direkten Überganges zu einem Vorgangssteuerungssystem - ist also für VAA ebenfalls nicht relevant.

Man kann Dokumente und Arbeitsmittel wesentlich genauer spezifizieren - das kann auch im Rahmen von VAA sinnvoll sein. Schlüsselworte werden dazu jedoch nicht unbedingt benötigt.

In der Literatur [Fas95b] ist eine große Auswahl weiterer Attribute enthalten, die bei Geschmack in den Freitexten verwendet werden können. Hier sollte jedoch eine möglichst handliche, einfache und minimale Methode spezifiziert werden.

II.6. Exkurs: Objektorientierte Geschäftsprozeßmodelle

Kenner der Objektorientierung werden sich bei der Lektüre des oben angegebenen Beispiels gefragt haben, warum man denn den Kreditvorgang nicht als Vorgangsobjekt spezifizieren sollte. Man würde dann einfach ein Objekt Kreditvorgang spezifizieren, das von einer oder mehreren Protokollklassen (gebräuchlich ist auch die Bezeichnung Aspektklasse [Kil+93]) für Vorgänge abgeleitet wird. In der Protokollklasse Vorgang sind allgemeine Eigensschaften von Vorgängen, wie Wiedervorlagen, Weitergabe, Delegation etc. modelliert. In der abgeleiteten Klasse werden die speziellen Eigenschaften von Kreditvorgängen modelliert.

Dies ist in der Tat eine praktische und natürliche Methode. Die angegebenen Sachbearbeitermethoden wären großteils mit leicht modifiziertem Namen auch als Methoden eines Vorgangsobjektes plausibel.

In der Literatur findet man eine Darstellung dieser Idee [Mül+93]. In vielen objektorientierten Vorgangssteuerungssystemen ist eine Vorgangsmappen-Metapher implementiert. Mit der Sachbearbeiter-Methode bedeutet dies einen Bruch der Sachbearbeitermetapher, da man kein Objekt Kreditvorgang beschreiben kann sondern nur den Verwalter (Sachbearbeiter), der für alle Kreditvorgänge zuständig ist. Auch sind Protokollklassen nicht modellierbar. Dies ist als Argument zu sehen, später noch auf Objektorientierung umzusteigen. Dazu kann jedoch die hier dargestellte Methodik auch als vernünftiger Zwischenschritt hinführen.

III. Sachbearbeiter

III.1. Spezifikation mit Sachbearbeitern

Dieses Handbuch beschäftigt sich mit der Spezifikation von Anwendungsbausteinen von VAA, also damit wie man die sogenannte „fachliche Funktionalität“ beschreiben soll.

Der Zweck „fachlicher Anwendungsbausteine“ ist es, den Anwendern der Anwendungsbausteine zu helfen. Der Anwender, der im Rahmen von Geschäftsprozessen Aufgaben erledigt, soll durch das DV-System unterstützt werden. Dazu beauftragt der Anwender das DV-System, ***Geschäftsvorfälle*** für ihn zu erledigen. Beispiele für solche Geschäftsvorfälle sind das Stornieren eines Vertrages oder die Aufnahme eines neuen Risikos in eine Police.

Man kann sich nun vorstellen, daß das DV-System aus einer Menge von elektronischen ***Sachbearbeitern*** besteht, die in der Lage sind, Geschäftsvorfälle zu erledigen. Der menschliche Sachbearbeiter delegiert also Arbeit an elektronische Sachbearbeiter. Ein VAA-Anwendungsbaustein kann dann als eine sinnvolle Menge solcher Sachbearbeiter aufgefaßt und spezifiziert werden.

Aufgabe der fachlichen Spezifikationsteams von VAA ist es, die Aufgabenbeschreibungen der elektronischen Sachbearbeiter zu formulieren und ihre Zusammenarbeit sinnvoll zu organisieren.

Geschäftsvorfälle und sonstige Aufträge

Sachbearbeiter können Aufträge der Anwender bearbeiten. Aufträge, die direkt vom Anwender aus einem Geschäftsprozeß oder aus einer Stapelverarbeitung (Batch) heraus erteilt werden, nennen wir Geschäftsvorfälle oder auch initiale Sachbearbeiteraufträge. Ein solcher Geschäftsvorfall stellt zugleich meist auch eine sinnvolle, fachliche Transaktion dar. Er ist also i.A. nicht unterbrechbar und wird entweder vollständig oder überhaupt nicht ausgeführt.

Sachbearbeiter können zur Erledigung ihrer Aufträge weitere Sachbearbeiter mit Teilaufgaben betrauen. Es handelt sich dabei um alle übrigen Sachbearbeiteraufträge.

Damit ein Sachbearbeiter weiß was er tun soll, benötigt er Informationen - also Eingabedaten. Er kann nach Abschluß des Auftrages Informationen zurückliefern - Ausgabedaten. Für jeden Sachbearbeiterauftrag kann man also eine genaue Schnittstelle spezifizieren, die angibt, welche Daten zur Ausführung benötigt werden, und welche durch die Ausführung ermittelt werden.

Gedächtnis der Sachbearbeiter - Datensichten

Damit zum Beispiel der Vertragssachbearbeiter einer Police ein neues Risiko hinzufügen kann, muß er Policen und Risiken kennen und sie sich permanent, auch über den Ablauf eines Programmes oder Auftrages hinaus merken können. DV-Experten ist klar, daß er sich dafür einer Datenbank bedienen wird. Für unsere modellhafte Vorstellung wollen wir annehmen, daß er dazu Karteikästen benutzen wird. In diesen Karteikästen hält er alle für ihn relevanten Informationen fest.

Die Summe aller Karteikästen aller Sachbearbeiter bildet das Datenmodell des Unternehmens.

Wie auch bei der menschlichen Sachbearbeitung macht es keinen Sinn, wenn jeder beliebige Sachbearbeiter in den Vertragsdaten beliebige Änderungen vornehmen kann. Unter menschlichen Sachbearbeitern würde dies aus guten Gründen Widerstand hervorrufen.

 © GDV, sd&m 1996

Wir wollen deshalb als Regel definieren, daß nur der Sachbearbeiter, dem ein Karteikasten (eine Datensicht) gehört, in diesen auch schreiben darf. Alle anderen Sachbearbeiter dürfen ihn nur lesen.

Dies macht eine zweite Regel erforderlich, daß nämlich die Karteikästen der Sachbearbeiter disjunkt sein müssen - daß es also keinen Karteikasten gibt, der mehr als einem Sachbearbeiter gehört. Bei menschlicher Sachbearbeitung ist das automatisch sichergestellt. Bei elektronischer Sachbearbeitung stellt es eine bewußte Einschränkung dar.

Datenabstraktion

Durch die Tatsache, daß ein Sachbearbeiter seine Karteikästen (Daten) vor unbefugter Benutzung durch andere Sachbearbeiter abschirmt, erhalten wir ein Form von Datenabstraktion. Es gibt Daten (Karteikästen) und Funktionen. Die Funktionen eines Sachbearbeiters können aber nicht auf alle Daten fremder Sachbearbeiter zugreifen, sondern nur auf ihre eigenen. Ein Sachbearbeiter kann also als ein Verwalter angesehen werden, der seine Objekte vor unerlaubten Zugriffen schützt. Zugleich wird durch die Sachbearbeiter eine vernünftige Modularisierung eingeführt.

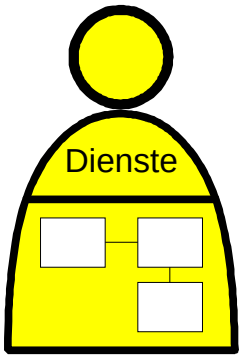


Abbildung 9: Sachbearbeiter mit Datensicht

Diese Vorstellung führt zu der Darstellungsform in Abbildung 9. Der Sachbearbeiter bietet Dienste an und schirmt seine Karteikästen nach außen ab.

Zustandsmodelle

Ein Sachbearbeiter wird typischerweise für eine bestimmte Klasse betrieblicher Objekte zuständig sein - zum Beispiel für Auftragspositionen.[2] Solche betrieblichen Objekte haben typischerweise einen Lebenslauf.

Abbildung 10 zeigt den Lebenslauf einer Auftragsposition für eine Auftragsabwicklung. Das Beispiel ist fachlich nicht sehr spezifisch und soll nur die Methode demonstrieren. Die Auftragsposition durchläuft in ihrem Leben die Zustände *angelegt*, *angenommen*, *freigegeben für Produktion*, *erledigt* und *archiviert*. Der Anfangszustand, der durch ein auf dem Kopf stehendes Dreieck symbolisiert wird, ist der Zustand, in dem das Objekt noch nicht existiert. Der Endzustand wird dann erreicht, wenn das Objekt aufgehört hat zu existieren.

In einem Sachbearbeiter muß jedes nicht-triviale Zustandsmodell der von dem Sachbearbeiter verwalteten Objekte spezifiziert werden. Gibt es zum Beispiel einen Auftragssachbearbeiter, der sowohl Aufträge als auch Positionen verwaltet, so muß man die Zustandsmodelle der Aufträge und der Positionen in dem Sachbearbeiter angeben. Falls aber sowohl Aufträge als auch die Positionen selbst komplexe Zustandsmodelle haben, sollte man lieber getrennte Sachbearbeiter spezifizieren.

---

[2] Hinweis für OO-Kenner: Man kann sagen, daß der Sachbearbeiter eine Meta-Klasse (ein Verwalter) für eine oder mehrere Klassen ist. Dementsprechend besitzt er das Zustandsmodell aller Klassen die er verwaltet. Objektorientiert kann man vieles einfacher formulieren, wenn die Terminiologie vorhanden ist. Davon kann hier aber i.A. nicht ausgegangen werden.

Zusammenhang Sachbearbeiteraufträge - Zustandsmodelle

Eine wichtige Frage ist noch, wie die Zustandsübergänge der Objekte geschehen, die der Sachbearbeiter verwaltet. Man kann beobachten, daß Zustände zum Beispiel der Auftragsposition immer dann wechseln, wenn eine fachlich sinnvolle Aktion an ihnen vorgenommen wird - also wenn ein Sachbearbeiterauftrag ausgeführt wird.

Durch das *Erstellen* der Auftragsposition wird sie *angelegt*. Durch ein *Storno* geht sie in den Endzustand über. Durch das *Prüfen* wird eine Auftragsposition *angenommen.* Man kann als Anhaltspunkt also feststellen, daß für jeden Zustandsübergang ein Sachbearbeiterauftrag vorhanden sein muß.

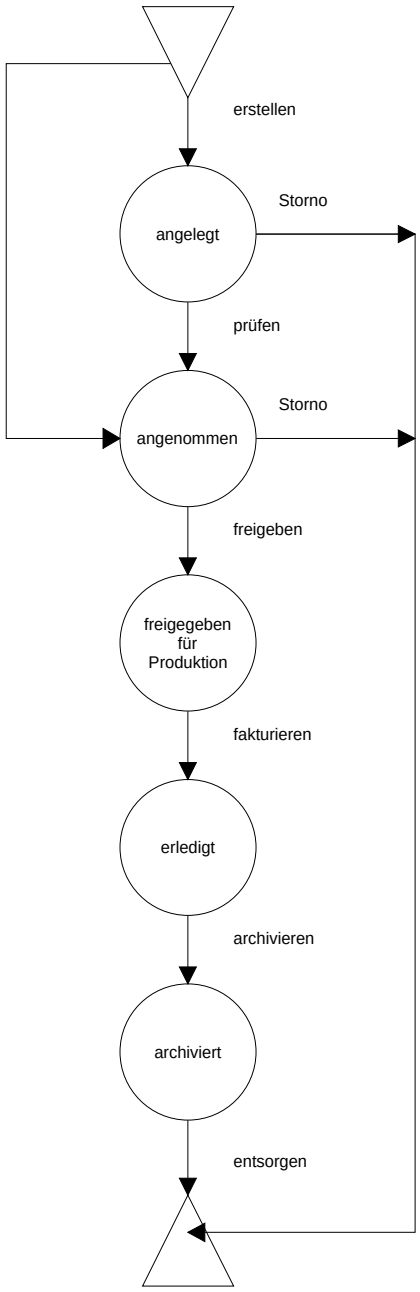


Abbildung 10: Zustandsmodell Auftragsposition - ohne Aufträge

 © GDV, sd&m 1996

Triviale Zustandmodelle

Nicht selten wird man Zustandsmodelle der in Abbildung 11 dargestellten Art finden. Bevor man sich der Mühe unterzieht, ein solches Zustandsmodell zu zeichnen, sollte man es lieber weglassen und durch den Hinweis „trivial“ ersetzen.

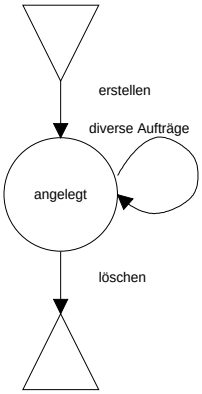


Abbildung 11: Triviales Zustandsmodell

Zusammenfassung: Elemente eines Sachbearbeiters

Bevor im folgenden Abschnitt ein Beispiel für einen vollständigen Sachbearbeiter angegeben wird, können wir hier noch die Elemente einer Sachbearbeiterspezifikation zusammenfassen. Abbildung 12 enthält dazu eine bildliche Darstellung der Inhalte der Spezifikation eines Sachbearbeiters.

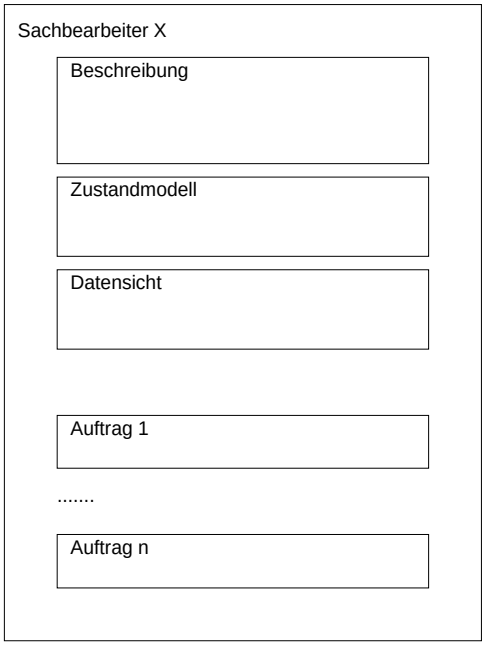


Abbildung 12: Struktur einer Sachbearbeiterspezifikation

III.2. Beispiel

Als Beispiel wollen wir wieder einen Sachbearbeiter verwenden, der mit Versicherungen nichts zu tun hat. Wir wollen den Sachbearbeiter für Auftragspositionen weiter verfeinern, der oben schon als Beispiel gedient hat. Die Funktionalität ist natürlich einfacher, als dies in realen Versicherungsanwendungen der Fall wäre.

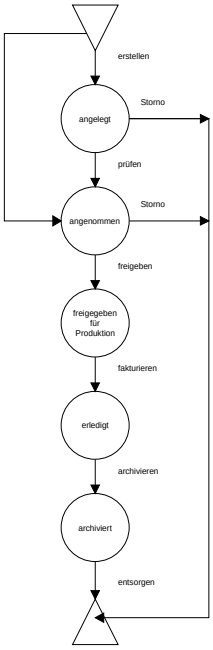
Sachbearbeiter Auftragspositionen

Beschreibung

Der Sachbearbeiter Auftragspositionen verwaltet alle Positionen von Kundenaufträgen der XY-GmbH. Er wurde von der Verwaltung kompletter Aufträge getrennt, da Positionen einzeln bearbeitet werden können. Aufträge machen vor allem Angaben über die Auslieferung und die Konditionen, die in den Positionen nicht redundant gespeichert werden sollen.

Zustandsmodell

Siehe schon Abbildung 10 - Deshalb hier nur noch verkleinert wiedergegeben.



Datensicht

Entität Auftragspositon

Bemerkung: Hier könnten auch mehrere Entitäten, Einschränkungen auf einzelne Attribute oder hierarchische Jackson-Datensichten angegeben werden.

 © GDV, sd&m 1996

Auftrag[3] erstellen

Braucht	Auftrag	:	Auftragsnummer
Braucht	Artikel	:	Artikelnummer
Braucht	Menge	:	Stückzahl
Liefert	Positonsnummer	:	Laufnummer

Vorbedingung

Der Auftrag ist vorhanden

Effekt

Eine neue Auftragsposition mit den angegebenen Eingabeparametern wird angelegt. Sie befindet sich im Zustand angelegt.

Auftrag prüfen

Braucht	Auftrag	:	Auftragsnummer
Braucht	Positionsnummer	:	Laufnummer
Liefert	Ausführbarkeit	:	**BOOL**

Effekt

Für den Artikel der Auftragsposition wird eine Verfügbarkeitsprüfung durchgeführt. Dies geschieht durch Aufruf des Sachbearbeiters Artikel.reservieren.

Wenn der Artikel nicht innerhalb der im Auftrag definierten Frist lieferbar ist, bleibt die Position im Zustand angelegt. Ansonsten geht sie in den Zustand angenommen über - der Artikel wurde für die Position reserviert.

Liefert als Ausführbarkeit TRUE (WAHR), wenn eine Reservierung vorgenommen werden konnte. Liefert FALSE (FALSCH), wenn keine Reservierung vorgenommen werden konnte.

Auftrag Storno

Braucht	Auftrag	:	Auftragsnummer
Braucht	Positionsnummer	:	Laufnummer

Vorbedingung

Die Auftragsposition ist vorhanden

Effekt

Eine eventuell vorhandene Warenreservierung für die Auftragsposition wird rückgängig gemacht (Sachbearbeiter Artikel.aufhebenReservierung). Die Position wird gelöscht.

Die übrigen Aufträge wie *freigeben*, *fakturieren*, *archivieren* und *entsorgen* werden hier nicht mehr spezifiziert. Das Beispiel sollte in dieser Form ausreichen. Für weitere Beispiele sei auf die Musterspezifikation verwiesen.

Hinweise zum Vorgehen bei der Spezifikation

Zustandsübergänge, die durch einen Auftrag hervorgerufen werden, müssen nicht explizit spezifiziert werden, solange sie eindeutig sind. Genauso wäre es denkbar bei jedem Auftrag eine Vorbedingung anzugeben, mit der geprüft wird, daß sich die Auftragsposition in einem Zustand befindet, der für das Erteilen des Auftrages vorausgesetzt werden muß. Solche Vor- und Nach-

[3] Bei **Auftrag** handelt es sich um ein Schlüsselwort für einen Sachbearbeiterauftrag. Nicht um einen Auftrag als fachliches Objekt. Das könnte hier zu Verwirrung führen.

bedingungen werden als implizit vorhanden vorausgesetzt, da sie aus dem Zustandsmodell abgelesen werden können. Eine redundante Spezifikation erschwert das Schreiben der Spezifikation und ermüdet den Leser durch endlose Zitate von Zustandsübergängen.

Die Effektbeschreibungen sollten möglichst fachlich sein und den Zustand beschreiben, der durch den Auftrag erreicht wird und weniger den Weg, wie man diesen Zustand erreicht. Es kann natürlich nicht verboten werden, in seltenen Fällen auch Pseudocode anzugeben, also das WIE, wie der Effekt erreicht wird. Spezifikationen sollten sich allerdings primär auf das WAS konzentrieren.

Die Beauftragung anderer Sachbearbeiter erfolgt über Angabe des Sachbearbeiters, des Auftrages und wenn nötig der Parameter. Man kann dafür auch eine formale Syntax einführen. Dann sind mittels Tools auch die Aufrufbeziehungen nachvollziehbar. Dies soll hier zunächst jedoch nicht vorgeschrieben werden.

Vergleich mit objektorientierten Spezifikationen

Wer objektorientierte Spezifikationen kennt, wird die obige Spezifikation als umständlich empfinden. Wenn man sich fragt, was eigentlich methodisch anders ist, stößt man vor allem darauf, daß man Objekte nicht direkt über Referenzen ansprechen kann, sondern alle Aufträge an den Sachbearbeiter (=Verwalter, Metaklasse) erteilt. Dadurch besteht der Zwang, mit jedem Auftrag den vollen Schlüssel zu übergeben. Man hat es hier also mit Objekten zu tun, auf die es keine direkten Referenzen gibt. Man muß dem Sachbearbeiter über den fachlichen Schlüssel immer sagen, welches Objekt man manipulieren möchte und kann dann die Parameter für die Operation mitgeben.

Das ist umständlicher als reine Objektorientierung, aber immer noch wesentlich sicherer, als rein „frei fliegende“ Funktionen. Es gilt also leider, daß wer nicht objektorientiert arbeiten möchte, schreiben muß.

Ein weiterer störender Punkt ist, daß man bei der Spezifikation von Parametern ebenfalls keine Objektreferenzen oder kompletten Objekte angeben kann (die gibt es in der Metapher ja nicht) sondern nur primitive Datentypen, also Fremdschlüssel, übergeben kann, wenn man auf ein Objekt Bezug nehmen will. Damit wird die Entscheidung über den Schlüssel schon zur Zeit der Spezifikation gefällt. Spezifikationen, die nur aus einem Daten- und einem Funktionsmodell bestehen, sind aber auch nicht besser.

III.3. Bildung sinnvoller Anwendungsbausteine

Im Rahmen von VAA ist noch die Frage offen, wie Anwendungsbausteine abgegrenzt werden sollen. Bei Anwendung der Sachbearbeitermethode und Anwendung von schlechten Erfahrungen, die mit schlecht modularisierten, objektorientierten Spezifikationen gemacht wurden, bietet sich dafür eine einsichtige Lösung an.

Man kann beobachten, daß ein Sachbearbeiter meist für eine (Police) oder nur wenige Entitäten (Partnersystem) des Datenmodells zuständig ist. Oft gibt es aber die Situation, daß mehrere Sachbearbeiter zusammen sinnvolle Arbeit erbringen. Sie bilden Teams - zum Beispiel bei der Schadensbearbeitung. Die initialen Aufträge (Geschäftsvorfälle) werden dann meist von einem Sachbearbeiter entgegengenommen und es werden Unteraufträge an andere Sachbearbeiter vergeben.

Beispiel sei hier ein Sachbearbeiter für Reisebuchungen. Eine Pauschalreise besteht aus mehreren Leistungen, typischerweise einer Flugbuchung, einer Hotelbuchung und Transfers sowie eventuell Zusatzleistungen. Die Verwaltung der Reisebuchungen wäre für ein Unternehmen der Touristikbranche sicherlich ein sinnvoller Anwendungsbaustein. Dieser Anwendungsbaustein enthält mehrere Sachbearbeiter und einen Teamchef, den Sachbearbeiter „Reisebuchungen“, der sie koordiniert. Eine solche Struktur ist in Abbildung 13 dargestellt. Die Aufträge des Sachbearbeiters Reisebuchungen sind also wohl vor allem initial - der Sachbearbeiter bietet nach außen

 © GDV, sd&m 1996

die Geschäftsvorfälle an und versteckt zugleich die interne Struktur des Bausteins - für den Benutzer der Geschäftsvorfälle kann es egal sein, wie die Leistungen erbracht werden.

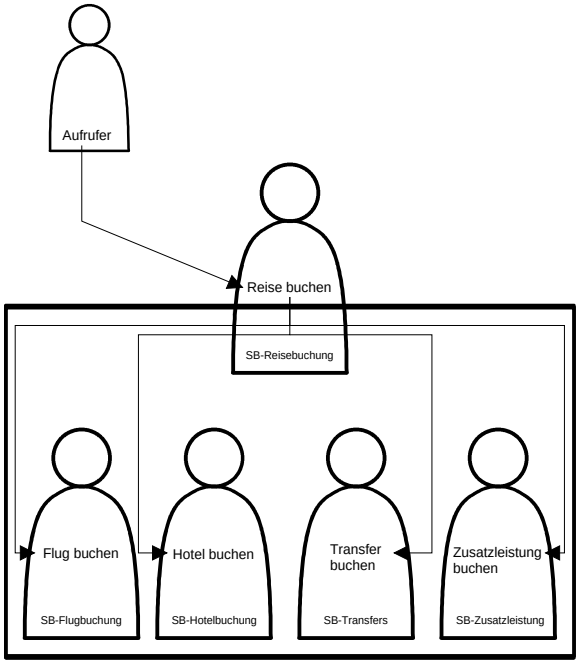


Abbildung 13: Anwendungsbaustein mit gut definierter Schnittstelle

Dieses Vorgehen kommt der Forderung nach der Bildung größerer, entkoppelter Einheiten nach. Man hat die Anforderung, daß gerade große Systeme auch oberhalb der Ebene einzelner Objekte wie Flugbuchung oder Transferbuchung noch geclustert werden sollten, um den Überblick zu erleichtern und die Anwendungsbausteine besser zu entkoppeln.

Dies erfordert zwar einen etwas höheren Aufwand in der Spezifikation, da viele eigentlich initiale Aufträge der Teammitglieder des Reisebuchungsbausteins in der Schnittstellenklasse, also dem Teamchef Reisebuchungen einfach identisch weitergereicht werden. Es schafft aber die Möglichkeit, den Cluster später intern anders zu organisieren, wenn dies sinnvoll erscheint.

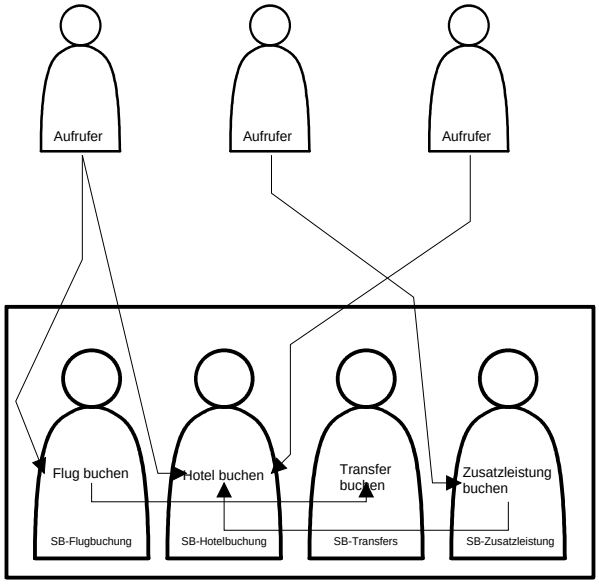


Abbildung 14: Anwendungsbaustein mit schlecht strukturierter Schnittstelle

Verzichtet man auf eine Clusterung, also die Bildung von Anwendungsbausteinen mit Schnittstellensachbearbeitern (Schnittstellenklassen), dann hat man es mit dem in Abbildung 14 dargestellten Effekt zu tun. Aufrufer wenden sich an beliebige Stellen des Teams - eine interne Um-

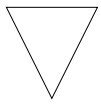
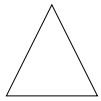
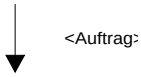
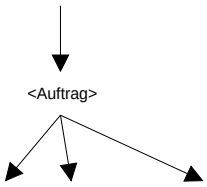
strukturierung des Teams bedeutet, daß man auch die Struktur der Benutzer der Dienste ändern muß. Eine Umstrukturierung bricht fast immer die Abläufe der Dienstbenutzer. Das kann vermieden werden. Solchermaßen durch Cluster entflochtene Systeme sind besser wartbar und schneller änderbar, außerdem besser und einfacher zu testen.

Die Einführung von Schnittstellensachbearbeitern oder Teamchefs, ist also unbedingt anzuraten. Es liegt dann nahe, ein Sachbearbeiterteam als Anwendungsbaustein zu sehen.

III.4. Beschreibungselemente

In diesem Abschnitt werden die graphischen Elemente für Zustandsmodelle dargestellt. Es wird die formale Syntax für Sachbearbeiterspezifikationen angegeben.

III.4.1. Graphische Elemente für Zustandsdiagramme

Bezeichnung	Symbol	Beschreibung
Anfangszustand		Zustand, in dem ein Objekt noch nicht existent ist. Es kann immer nur genau einen Anfangszustand geben.
Endzustand		Zustand, in dem ein Objekt aufhört zu existieren. Es kann beliebig viele Endzustände geben.
Zustand		Zustand in dem sich ein Objekt zwischen einzelnen Bearbeitungsschritten (Aufträgen) befindet.
Zustandsübergang mit Aktion		Jeder Zustandsübergang wird durch eine Aktion bewirkt.
Zustandsübergang mit Verzweigung		Falls nach einer Aktion mehrere Zustände eingenommen werden können, wird dies durch mehrere von der Aktion abgehende Pfeile symbolisiert. Es handelt sich also um Zustandsübergänge, die je nach Inhalt des Auftrages zu einem anderen Folgezustand führen.

III.4.2. Sachbearbeiter

Ein Sachbearbeiter wird beschrieben durch seinen Namen (Identifikator) und ist mit einer fachlichen Beschreibung ausreichend zu definieren.

Die Angabe eines (nicht trivialen) Zustandsmodells ist verpflichtend. Das Zustandsmodell ist durch eine eingebettete Graphik zu beschreiben. Ist dies nicht möglich, weil zu viele Zustandsübergänge vorhanden sind und die Graphik unübersichtlich werden würde, so ist es in Form einer Zustandsübergangsmatrix zu beschreiben. Ist das Zustandsmodell lediglich trivial, so reicht die Angabe „trivial“ für das Kapitel Zustandsmodell.

Die Angabe einer Datensicht erfolgt durch die Angabe einer Datensichtbeschreibung. Momentan ist das nur eine Aufzählung von Entitäten oder von Attributen und Entitäten.

Die Aufträge des Sachbearbeiters (Funktionen) werden danach einzeln angegeben. Mindestens ein Auftrag sollte vorhanden sein. **(Regel 5.)**

 © GDV, sd&m 1996

Formale Syntax

(Regel 5.) *Sachbearbeiter* ::=

Sachbearbeiter *Identifikator*

Beschreibung

Freitext

Zustandsmodell

Freitext

Datensicht

DatensichtBeschreibung

SachbearbeiterAuftrag+

III.4.3. Datensicht

Eine Datensicht ist eine Menge von kompletten Entitäten und Attributen. **(Regel 6.)**

Anmerkung: Hier wird in einer späteren Version die Syntax der Datensichtsprache AK-Datenmanager übernommen werden, die allerdings dem Autor bei der Erstellung des Dokumentes nicht ausreichend dokumentiert zur Verfügung stand, da eine Kurzzusammenfassung für VAA erst im AK-Datenmanager in Arbeit ist. Es handelt sich dabei um die Beschreibung von Jackson-Bäumen über Datenmodellen mit integrierten Konstrukten für die Zeitnavigation. Dies ist besonders für Auswertungen wichtig.

Beispiel

Das folgende Beispiel beschreibt einen komplexeren Fall einer Datensicht. Die Datensicht besteht aus zwei Entitäten und bei diesen jeweils auch wieder nur aus Ausschnitten, die durch Bedingungen angegeben werden.

Die Datensicht ist also rein informell beschrieben. Mit der Einführung einer Datensichtsprache können hier gerade für komplexe Datensichten, wie man sie für die Tarifierung oder Policierung benötigt werden, noch wesentliche Verbesserungen erzielt werden.

Datensicht

Entität Forderung, allerdings nur die Instanzen mit der *Forderung.Art* = *„Anspruch“*

Entität Korrektur, allerdings nur die Instanzen, die von einem Anspruch abhängig sind (also zu einem Anspruch gehören).

Formale Syntax

(Regel 6.) *DatensichtBeschreibung* ::=

*(Entitätsname | Entitätsname.Attributname | Freitext)**

(Regel 7.) *Entitätsname* ::= *Identifikator*

(Regel 8.) *Attributname* ::= *Identifikator*

III.4.4. Aufträge

Ein Sachbearbeiterauftrag wird beschrieben durch seinen Namen (Identifikator), durch eine Menge von beliebig vielen Parametern, beliebig viele Vor- und Nachbedingungen und eine Effektbeschreibung. **(Regel 9.)**

Ist eine Vorbedingung oder Nachbedingung verletzt, so ist dies als Systemfehler zu betrachten. Hier soll als Beschreibung der Semantik ausreichen, daß dann die Bearbeitung komplett abgebrochen werden muß, weil ein schwerwiegender Softwarefehler vorliegt.

Parameterbeschreibung

Je Parameter muß beschrieben werden, ob es sich um einen Ein-, Ausgabe- oder Ein-/Ausgabeparameter handelt. Es muß der Name des Parameters angegeben werden, ein Datentyp sowie eine Kardinalität. Damit ist auch die Übergabe von variablen Arrays über Typen möglich. **(Regel 10.)**

Für die Angabe, ob es sich um einen Ein-, Ausgabe- oder Ein-/Ausgabeparameter handelt, wird die folgende Notation verwendet **(Regel 11.)**:

Schlüsselwort	Bedeutung
Braucht	Eingabeparameter
Liefert	Ausgabeparameter
Ändert	Ein-/Ausgabeparameter

Es ist möglich, nicht nur ein einfaches Argument zu übergeben, sondern zum Beispiel auch ein optionales Argument oder eine Menge von Argumenten (Array von Zahlen). Dafür gelten die folgenden Kardinalitätsoperatoren **(Regel 13.)**.

Operator	Bedeutung
:	Genau ein Argument des beschriebenen Typs.
o:	Kein oder ein Argument - optionaler Parameter.
::	Mindestens ein Element, beliebig viele.
o::	Beliebig viele, also auch Null Elemente des angegebenen Datentyps werden übergeben.

Datentypen werden näher in Kapitel VI beschrieben - sie haben eine eigene Syntax. **(Regel 14.)**

Effektbeschreibung

Die Effektbeschreibung erfolgt als Freitext. Dabei sollte man sich darauf konzentrieren, den Effekt eines Auftrags zu beschreiben, also WAS durch den Auftrag bewirkt wird und nicht unbedingt detailliert ausführen, WIE es bewirkt wird.

Wenn ein Spezifikateur über das WIE detaillierte Vorstellungen hat, kann er das natürlich ebenfalls niederlegen. Er sollte es nur nach eine Effektbeschreibung tun und vor allem zusätzlich zur Effektbeschreibung. Langer Pseudocode kann sehr präzise das WIE beschreiben. Das WAS kann dadruch aber perfekt vernebelt werden. Pseudocode und das WAS sind also im Text deutlich zu trennen. Erst die Effektbeschreibung und dann, wenn es unbedingt sein muß, auch noch Pseudocode.

In der Effektbeschreibung muß man oft auch noch angeben, welche Dienste anderer Sachbearbeiter benutzt werden. Dies wird mit dem Schlüsselwort **Sachbearbeiterauftrag** gekennzeichnet, gefolgt von *Sachbearbeitername*.*Methodenname*(*Parameter*). Die Angabe von Parametern ist optional.

 © GDV, sd&m 1996

Beispiel

Effekt

Der neue Zustand des Anspruchs ist abgelehnt. Es wird ein Standardschreiben erstellt mit **Sachbearbeiterauftrag** Korrespondenz.Brief(Anspruchsteller, Textbaustein, Grund, ...)

Auf diese Weise könnte man auch maschinell die Konsistenz von Aufrufen innerhalb der Spezifikationen prüfen.

Formale Syntax für Sachbearbeiteraufträge

(Regel 9.) *SachbearbeiterAuftrag* ::=

Auftrag *Identifikator*

*ParameterBeschreibung**

[**Vorbedingung**

Freitext]*

[**Nachbedingung**

Freitext]*

Effekt

Freitext

(Regel 10.) *ParameterBeschreibung* ::=

InOutOprator *ParameterName* *Kardinalität* *Datentyp*

(Regel 11.) *InOutOperator* ::=

(**Braucht** | **Liefert** | **Ändert**)

(Regel 12.) *Parametername* ::= *Identifikator*

(Regel 13.) Kardinalität ::=

(**:** | **::** | **o:** | **o::**)

(Regel 14.) Datentyp ::=

siehe Kapitel VI

IV. Batches

IV.1. Motivation

Batches sind ein Systemteil, der leider bei der Systemspezifikation oft vernachlässigt wird - so zunächst auch hier[4]. Das potentielle Ergebnis ist schlechtes Batchdesign oder ad-hoc Programmierung von Batches. Beides Effekte, die es zu vermeiden gilt.

Es wird hier eine Beschreibungstechnik eingeführt werden, die Batches in einzelne Steps unterteilt, die selbständig ausgeführt werden können und definierte Zwischenergebnisse (Checkpoints) liefern. Dies ist für ein Batchdesign besonders wichtig, da man gerade in große Batches immer wieder Checkpoints einfügen muß, um nach einem Abbruch eines Steps nicht den kompletten Batch nachfahren zu müssen.

Die Steps verwenden die Funktionalität der Sachbearbeiter - auf diese Weise wird vermieden, daß Anwendungskernlogik verstreut oder mehrfach spezifiziert wird.

Die Steps werden graphisch in einem Batch-Ablaufplan dargestellt. Zwischenergebnisse können in temporären Datenmodellen oder Datensichten abgelegt werden.

Eine Syntax könnte hier noch dargestellt werden - diese sollte aber nach einem umfangreichen Beispiel entwickelt werden und nicht umgekehrt.

[4] Der Teil über die Beschreibung von Batches wird in der nächsten Version ergänzt.

 © GDV, sd&m 1996

V. Datenmodell

V.1. Datenmodellierung bei VAA

Zur Datenmodellierung gibt es eine große Zahl von Methoden, die von ihren jeweiligen Anhängern teilweise dogmatisch vertreten werden. Es wird oft strikt zwischen physischer und logischer Datenmodellierung unterschieden. Es gibt auch in der Datenmodellierung Vererbungskonstrukte, verschiedene Arten von Beziehungen, existentielle und nicht-existentielle Abhängigkeiten und viele weitere Konstrukte, mit denen sich die Hersteller verschiedener Tools voneinander differenzieren.

Dieser Methodensupermarkt ist unübersichtlich und zudem mit oft endlosen, dogmatischen und deshalb oft wenig zielführenden Diskussionen über Vor- und Nachteile einzelner Methoden belastet. Um bei VAA schnell und unkompliziert zu brauchbaren Ergebnissen zu kommen, wird im Rahmen von VAA eine einfache und bewährte Methode einer mitarbeitenden Versicherung eingesetzt, der Württembergischen.

Diese Methode liegt relativ nahe am späteren physischen Datenmodell und hat sich nach Aussagen der Württembergischen gerade deshalb bewährt. Zudem gibt es eine dazu passende Datensichtsprache und entsprechende Konstrukte für die Historisierung und Zeitnavigation. Diese werden ebenfalls verwendet werden.

Da wir hier davon ausgehen, daß dem Leser die Methode der logischen Datenmodellierung grundsätzlich vertraut ist, soll hier die Aufschreibung für VAA erst an Beispielen und dann in Form von formalen Syntaxbeschreibungen wiedergegeben werden.

V.2. Beispiel

Im folgenden wird ein beispielhafter Ausschnitt aus einem Datenmodell gezeigt, das mit der VAA-Methode spezifiziert wurde. Interessant ist nicht der (beliebig triviale) fachliche Inhalt, sondern die Art der Modellierung im Vergleich zu anderen Methoden.

Die dargestellte Numerierung entspricht der Mustergliederung.

V.2.1. Beispiel Entitäts- / Beziehungsdiagramm

3.4 Datenmodell

3.4.1 Überblick

Wir betrachten einen Ausschnitt aus einem System für die Kontenverwaltung institutioneller Anleger (siehe Abbildung 15).

Institutionelle Anleger können als verschachtelte Gesellschaften auftreten. Dabei kann ein institutioneller Anleger Muttergesellschaft eines anderen institutionellen Anlegers sein. Betrachtet werden nur Beteiligungen mit bestimmendem Einfluß nach KWH §14.

Ein institutioneller Anleger wird von mehreren Mitarbeitern betreut. Jeder Mitarbeiter kann gleichzeitig mehrere institutionelle Anleger betreuen.

Zu jedem institutionellen Anleger gibt es mehrere Konten. Diese sind existentiell von dem Anleger abhängig.

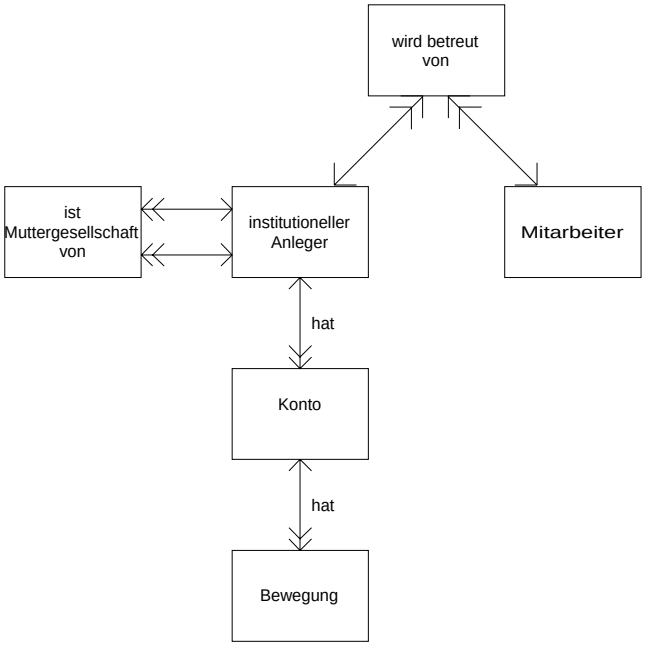


Abbildung 15: Beispiel Datenmodell

Jedes Konto hat mehrere Bewegungen. Diese sind existentiell von ihrem Konto abhängig.

Man beachte, daß in der Erläuterung jede Beziehung und jede Entität erläutert wird und auftaucht. Ein Übersichtsbild für sich alleine ist oft nicht ausreichend. Besser ist es, den Inhalt auch noch verbal zu beschreiben.

V.2.2. Beispiele für Entitätsbeschreibungen

3.3.2 Entität institutionellerAnleger

Beschreibung

Institutionelle Anleger sind die wichtigen Kunden der XY Brokerage GmbH & Co. KG. Sie unterscheiden sich von normalen Anlegern durch ihr Anlagevolumen, ihre Betreuung und durch die Betreuung.

Primärschlüssel

Kundennummer

Schlüsselkandidat

Firma, Firmensitz

Attribute

hat	Kundennummer	:	KdNummer

Ordnungsmerkmal für den Kunden - künstlicher Schlüssel.

hat	Firma	:	Name

Die Firma ist der Name, unter dem der institutionelle Anleger firmiert.

hat	Firmensitz	:	Adresse
hat	Deckungsbeitrag	:	TausendDMBetrag

Deckungsbeitrag, der im letzten Geschäftsjahr mit dem Kunden erwirtschaftet wurde.

Weitere Attribute nach Belieben.

 © GDV, sd&m 1996

3.3.3 **Entität** Konto

Beschreibung

Wertpapierkonto, das nach der firmeninternen Methode XY geführt wird. Der Bestand ergibt sich jeweils aus der Summe der aktuellen Buchungen. Diese werden zum Geschäftsjahresschluß bereinigt und archiviert. Saldierungen sind verboten.

Primärschlüssel

Kontonummer

Attribute

hat Kontonummer : KtoNummer

Ordnungsmerkmal für das Konto - künstlicher Schlüssel.

ist abhängig von Kunde

Ein Kunde kann viele Konten besitzen. Die Konten müssen vollständig gelöscht sein, damit ein Kunde gelöscht werden kann.

hat Deckungsbeitrag : TausendDMBetrag

Deckungsbeitrag, der im letzten Geschäftsjahr über das Konto erwirtschaftet wurde

Weitere Attribute nach Belieben.

V.2.3. Diskussion verwendeter Konstrukte

Die Methodik ist, was die semantische Ausdrucksstärke anbelangt, einfacher als andere Datenmodellierungsmethoden. Dies ist ein Vorteil, weil sie schneller zu erlernen ist. Es ist aber auch ein Nachteil, weil semantische Einschränkungen dann in den Sachbearbeitern definiert werden müssen.

Integritätsbedigungen, wie zum Beispiel „Referential Integrity“ müssen in den Sachbearbeitern oder im Freitextteil der Datenmodelle spezifiziert werden. Damit ist es nicht möglich, eine kanonische Abbildung zum Beispiel in automatisch generierten Code zur Sicherung von Integritäten anzugeben. Für ein Großprojekt wie VAA handelt man sich hier ein unnötiges Wartungsproblem ein. Es ist anzunehmen, daß dies zu Mehrdeutigkeiten in der Realisierung von Bausteinen führen wird.

Kardinalitäten von Beziehungen

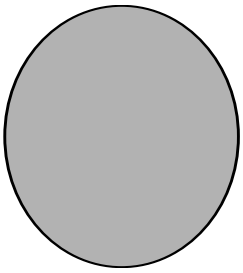
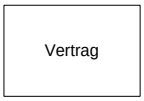
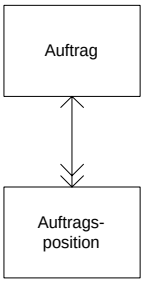
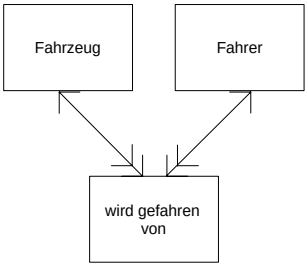
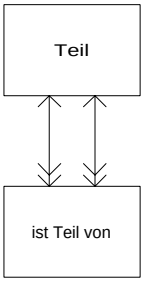
Generell wird zum Beispiel nicht zwischen einer 1:n und einer 0:n Beziehung unterschieden. Damit muß zum Beispiel dieFrage, ob ein Auftrag mit keinen Positionen erlaubt ist, im Sachbearbeiter diskutiert werden.

Beziehungen immer als Entität

Bei vielen Datenmodellierungsmethoden wird zwischen Beziehungsentitäten und anderen Entitäten unterschieden. Dies ist hier nicht der Fall. Eine n:m Beziehung kann keine Attribute tragen. Eine Beziehung wird im Falle einer n:m-Beziehung oder einer Selbstbeziehung immer als Entität dargestellt. Damit liegt die Methode nahe an der physischen Modellierung, in der spätestens in der physischen Umsetzung eine Beziehungstabelle aus Fremdschlüsseln entstehen muß. Diese Vereinfachung ist vertretbar.

V.3. Beschreibungselemente

V.3.1. Graphische Beschreibungselemente

Bezeichnung	Grafik	Beschreibung
Informationsobjekt		❑ Alle Entitäten, die innerhalb eines Informationsobjekts liegen, unterliegen derselben Zeitlogik (Versionsführung) ❑ Jedes Informationsobjekt enthält genau eine Führungsentität ❑ Die Führungsentität ist von keiner innerhalb des Informationsobjekts liegenden Entität abhängig.
unabhängige Entität		❑ Menge/Klasse von Objekten (Verträgen) der realen Welt mit gleichen Attributen.
abhängige Entität (1:n-Beziehung)		❑ Zwischen Entität 1 (Auftrag) und Entität 2 (Auftragsposition) besteht eine 1:n-Beziehung. ❑ Entität 1 (Auftrag) ist Parent von Entität 2 (Auftragsposition). Dadurch wird eine hierarchische Abhängigkeit ausgedrückt (Existenzvoraussetzung) ❑ 1:1-Beziehungen werden ebenfalls mit dieser Symbolik ausgedrückt. Sie sind implizit enthalten.
abhängige Entität mit zwei Parents		❑ Spezialfall einer von 2 verschiedenen Parents abhängigen Entität mittels jeweils einer 1:n-Beziehung. ❑ Dient zum Ausdruck einer n:m Beziehung
abhängige Entität mit doppelter Parent-Beziehung n:m Beziehung		❑ Spezialfall einer von einem Parent abhängigen Entität mittels zweier 1:n-Beziehungen. ❑ Dient als Beschreibung für Selbstbeziehungen.

 © GDV, sd&m 1996

V.3.2. Entitätsbeschreibungen

Die Beschreibung einer Entität enthält die Angabe eines Namens (Identifikator), einer betriebswirtschaftlichen Beschreibung in Form von Freitext, eine Primärschlüsseldeklaration, die Angabe beliebig vieler weiterer Schlüsselkandidaten und danach eine Folge von beliebig vielen Attributen in beliebiger Reihenfolge. **(Regel 15.)**

Eine Attributbeschreibung besteht aus einem Attributnamen, einer Kardinalität (vergleiche Funktionsparameter, **(Regel 13.)**, Seite 35) und der Angabe eines Datentyps. Die Angabe eines präzisen fachlichen Datentyps ist verpflichtend und von extremer Wichtigkeit (siehe Kapitel VI). Attribute werden zusätzlich durch einen aussagefähigen Freitext beschrieben. Dieser ist optional für den Fall, daß der Name schon selbstbeschreibend ist. **(Regel 16.)**

Beziehungen werden nur bei der abhängigen Entität (die, auf die doppelte Pfeilspitze zeigt) formal beschrieben. Redundante Darstellungen sind zu vermeiden. Die Erläuterung der Beziehungen zum Datenmodellübersichtsbild ist also um so wichtiger. Die Angabe einer Beziehung erfolgt durch die Angabe einer Entität, von der die beschriebene Entität in einer 1:n Beziehung abhängig ist. Erläuterungen können in einem optionalen Beschreibungstext erfolgen. **(Regel 17.)**

Angaben über Mengenrahmen, Verzweigungsgrad von Beziehungen und weitere Angaben, die man in Datenmodellierungstools noch findet, können nach Bedarf im Freitext beschrieben werden.

Bemerkungen zur Vollständigkeit der Beschreibung

Eine Primärschlüsseldeklaration ist in der logischen Datenmodellierung manchmal explizit ausgeklammert. Wir haben sie aufgenommen, weil sie nützlich ist und nicht schadet.

Momentan sind noch keine syntaktischen Konstrukte für Zeitlogik und Informationsobjekte enthalten. Diese müssen noch abgestimmt und eingefügt werden. Bis dahin werden die Informationsobjekte durch Angabe in den Übersichtsbildern und textuelle Beschreibung spezifiziert.

Momentan ist die Syntax für Beziehungen recht spärlich. Da es in der VAA Methode nur abhängige Entitäten gibt, reicht es aus, dies durch die **(Regel 17.)** zu formulieren. Man könnte für die Beziehungsentitäten von n:m Beziehungen statt des Schlüsselwortes *Entität* noch das Schlüsselwort *Beziehung* einführen und außerdem statt *ist abhängig von* noch *ist Teil von*, um zum Beispiel die Beziehung zwischen Auftragsposition und Auftrag besser lesbar zu machen. Dies kann für spätere Versionen des Handbuches noch diskutiert und berücksichtigt werden.

Formale Syntax

(Regel 15.) *Entitätsbeschreibung* ::=

Entität *Identifikator*

Beschreibung

Freitext

Primärschlüssel

*Attributname**

{**Schlüsselkandidat**

*Attributname** } *

Attribute

(*Attributdeklaration* | *Beziehungsdeklaration*) +

(Regel 16.) *Attributdeklaration* ::=

hat *AttributName* *Kardinalität* *DatentypName*

Beschreibungstext

(Regel 17.) Beziehungsdeklaration ::=

ist abhängig von *EntitätsName*

Beschreibungstext

Formale Syntax, die man überlesen kann

(Regel 18.) *Bescheibungstext ::= Freitext*

(Regel 19.) *AttributName ::= Identifikator*

(Regel 20.) *EntitätsName ::= Identifikator*

(Regel 21.) *DatentypName ::= Identifikator*

 © GDV, sd&m 1996

VI. Datentypen

VI.1. Ziele eines Datentypkonzeptes

Das Konzept von Datentypen ist Informatikern aus höheren Programmiersprachen wie Modula, Pascal oder ADA wohlvertraut. Diese Sprachen erlauben es dem Programmierer eigene, primitive Datentypen zu definieren.

In der Welt der Datenbanken und in der Welt von kommerziellen Programmiersprachen, wie zum Beispiel COBOL, finden wir keine komplexeren, selbstdefinierbaren Datentypen. Folge ist, daß als Attributtypen in der Datenmodellierung oft Deklarationen der Art

Attribute

hat Kontonummer : PIC 9(5)

Ordnungsmerkmal für das Konto - künstlicher Schlüssel.

angegeben werden. Dies hat gravierende Nachteile. Stellt sich heraus, daß das Format für die Kontonummer geändert werden muß, so gibt es keine Möglichkeit, eine solche Änderung leicht und konsistent in einem kompletten System durchzuführen.

Es ist also dringend anzuraten für Attribute und Parameter außer in Ausnahmefällen NIE primitive Datentypen zu verwenden, sondern immer fachliche Datentypen. Definiert man also das Attribut als:

Attribute

hat Kontonummer : KtoNummer

Ordnungsmerkmal für das Konto - künstlicher Schlüssel.

Und zusätzlich den Datentyp KtoNummer

Datentypen

DIE KtoNummer
IST EIN NAT(5)

Dumme Nummer für die Numerierung von Konten.

dann hat man das Wissen an genau einer Stelle konzentriert und kann später Änderungen einfach durchführen. Einen entsprechenden Generierungsmechanismus vorausgesetzt, verläuft ein solcher Wartungsfall großteils automatisch. Noch einfacher ist es, wenn man nur Wertgrenzen ändert. In einem solchen Fall können automatisch die Wertprüfungen in der Benutzeroberfläche und im Anwendungskern angepaßt werden. Wenn man auf ein Datentypkonzept verzichtet, kann man solche Vorteile später nicht nutzen. Um das Konzept voll nutzen zu können, muß auch noch die folgende Bedingung erfüllt sein:

Der Name eines Datentyps muß VAA-weit einheitlich sein. Diese Forderung ist organisatorisch sicherzustellen. Diese Anforderung muß leider gemacht werden, um die Spezifikation großer Mengen von Datentypen zu vermeiden.

Wird sie aufgehoben, dann muß man leider auf typsichere Schnittstellen zwischen den Anwendungsbausteinen verzichten. Dies erleichtert die Wartung, verringert aber die Sicherheit zur Laufzeit.

Im folgenden wird also ein System von Datentypdeklarationen beschrieben, das es ermöglicht, das Wissen um Datentypen an genau einer Stelle zu spezifizieren und so die Wartung zu erleichtern. Davon abgesehen sind fachliche Namen für Datentypen auch fachlich aussagekräftiger und machen Spezifikationen leichter lesbar.

Ist das Datentypkonzept zu komplex?

Das hier vorgestellte Datentypkonzept ist nicht ganz einfach. Es entspricht in seinen Möglichkeiten einer mäßig komplexen Programmiersprache der dritten Generation (Pascal, C). Zeiger wurden jedoch weggelassen.

Widerspricht ein Konzept mit 17 syntaktischen Regeln also nicht den in der Einleitung gemachten Forderungen nach Einfachheit und Verständlichkeit? Es wird sichtbar, daß bei der Auswahl der Syntax Wert auf Lesbarkeit und nicht auf kompakte Schreibweise gelegt wurde.

Die meisten Konstrukte, wie Einschränkungstypen, Verbundtypen, Aufzählungstypen findet man auch in Datenmodellierungswerkzeugen wie ADW oder ähnlichen wieder. Nur sind sie dort immer fest und ohne Indirektionsstufe an ein Attribut oder einen Parameter gekoppelt, was zur Folge hat, daß man für Änderungen ein komplettes Datenmodell an vielen Stellen anpassen muß.

Bleibt dann noch die Frage, ob sich solche Typen auch direkt auf die (primitiveren) Typkonstrukte einer Datenbank abbilden lassen? Unserer Erfahrung nach ist das durch Generatoren, die Typdeklarationen rekursiv zum Beispiel in COBOL-Copy-Strecken umsetzen können, möglich. Solange sich jeder elementare Datentyp in der Zielsprache abbilden läßt und sich zu jedem Konstrukt eine Abbildungsregel angeben läßt, kann man sozusagen per mathematischer Induktion zeigen, daß sich auch alle zusammengesetzten Typen abbilden lassen.

VI.2. Beschreibungselemente

VI.2.1. Primitive Datentypen

Ein System von Datentypen muß auf primitiven Datentypen basieren, von denen komplexere Datentypen abgeleitet werden. Für die Spezifikationen der fachlichen Bausteine von VAA werden die primitiven Datentypen vereinbart, die in der folgenden Tabelle genannt werden

Datentyp	Bedeutung
BOOL	Wahrheitswerte, boolesche Werte mit zweiwertiger Logik. Konstanten **TRUE** und **FALSE**
INT(<n>)	Ganzzahl der Länge <n> mit Vorzeichen, zum Beispiel **INT(5)**
NAT(<n>)	Natürliche Zahl der Länge <n> ohne Vorzeichen, zum Beispiel **NAT(3)**
FIX(<n>,<m>)	BCD-Arithmetik mit <n> Vorkommastellen und <m> Nachkommastellen.
FLOAT(<n>)	Gleitkommazahl mit n-stelliger Genauigkeit.

 © GDV, sd&m 1996

Datentyp	Bedeutung
STRING(<n>)	Zeichenkette mit <n> Zeichen Länge. Zum Beispiel **STRING(10)** für eine Zeichenkette der Länge 10. Konstante **EMPTY** für die leere Zeichenkette.
DATE	Datum der Genauigkeit TT/MM/JJJJ
TIME	Zeit in der Genauigkeit HH:MM:SS

(Regel 22.) *PrimitiverTyp* ::=

(**BOOL** | **INT(**<*n*>**)** | **NAT(**<*n*>**)** | **FIX(**<*n*>,<*m*>**)** | **FLOAT(**<*n*>**)** | **STRING(**<*n*>**)**
DATE | **TIME**).

Jede Datentypdeklaration wird durch einen Kopf **DER** | **DIE** | **DAS** *Datentypname* eingeleitet. Es folgt dann eine entsprechende Beschreibung des Typs. Die einzelnen Arten von Typdeklarationen werden in den folgenden Abschnitten erläutert.

Die syntaktische Überblicksregel wird hier noch angegeben **(Regel 23.)**

Formale Syntax

(Regel 23.) *Datentypdeklaration* ::=

der Datentypname
(**IST EIN** *PrimitiverTyp* |
Aufzählungstypdeklaration |
Feldtypdeklaration |
Unterbereichstypdeklaration |
Verbundtypdeklaration |
)
[**Beschreibung** *Freitext*]

(Regel 24.) *der* ::= (**DER** | **DIE** | **DAS**).

Beispiele primitiver Typen

Primitive Typen sollten nur zurückhaltend direkt eingesetzt werden. Wo mit einer Änderung zu rechnen ist, oder wo der Typ zum Beispiel für einen Fremdschlüssel verwendet werden soll, ist es besser, wenn man einen Anwendungstyp basierend auf einem primitiven Typ verwendet. Dies wird durch das folgende Beispiel illustrieren:

DIE Kontonummer

IST EIN **NAT(**5**)**

Beschreibung Typ für den Primärschlüssel der Entität Konto.

DER DMBetrag

IST EIN **FIX (**10,2**)**

Im ersten Fall kann es sein, daß man später für die Kontonummer noch fachliche Regeln vereinbaren möchte oder ihre Länge ändern muß. Der DMBetrag ist zwar momentan ziemlich sicher, aber kann so auch im Falle einer fortschreitenden Inflation später erweitert werden.

Direkt kann man dagegen meist Typen wie **BOOL, DATE** oder **TIME** einsetzen.

VI.2.2. Aufzählungstypen

Aufzählungstypen werden für Typen verwendet, bei denen man wenige Werte typischerweise als Konstante zur näheren Beschreibung eines Gegenstandes einsetzen und später im Rahmen von Fallunterscheidungen einsetzen möchte.

Beispiele für Aufzählungstypen

DIE RGBFarbe

HAT WERTE

rot

gelb

blau

DAS Geschlecht

HAT WERTE

männlich

weiblich

Formale Syntax

(Regel 25.) *Aufzählungstypdeklaration* ::=

HAT WERTE

WertName *

(Regel 26.) *WertName* ::= *Identifikator*

 © GDV, sd&m 1996

VI.2.3. Feldtypen (Arrays, Reihungen)

Feldtypen werden auch als Reihungen oder Arrays bezeichnet. Sie bilden ein Feld über einem festen Trägertyp. Als Trägertyp ist jeder andere Datentyp zugelassen.

Beispiele für Feldtypen

DER Bildschirmpunkt

IST LISTE VON INT(5**) [** RGBFarbe **]**

Beschreibung ein Farbpunkt eines Bildschirms setzt sich aus jeweils einem Wert für rot, gelb und blau zusammen

DAS Dreieck

IST LISTE VON Punkt **[** 3 **]**

Beschreibung Ein Dreieck wird durch drei Punkte bestimmt.

Formale Syntax

(Regel 27.) *Feldtypdeklaration* ::=

(**IST LISTE VON** *Trägertyp* **[** *NatürlicheZahl* [, *NatürlicheZahl*] **]** |
IST LISTE VON *Trägertyp* **[** *Aufzählungstyp* **]** |
IST LISTE VON *Trägertyp* **[** *Unterbereichstyp* **]**)

Formale Syntax, die man überlesen kann

(Regel 28.) *Trägertyp* ::= *Identifikator*

(Regel 29.) *NatürlicheZahl* ::= (**0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**)*

(Regel 30.) *Aufzählungstyp* ::= *Identifikator*

(Regel 31.) *Unterbereichstyp* ::= *Identifikator*

VI.2.4. Einschränkungstypen

Oft möchte man nur eine bestimmte Teilmenge eines skalaren, primitiven Datentyps zulassen. Man bildet dann Einschränkungstypen. Man kann dieser Einschränkungstypen durch Bildung eines Intervalls auf einem skalaren Datentypen (also einem geordneten, nicht zusammengesetzten Datentyp) definieren.

Beispiele für Einschränkungstypen

DER Kleinbetrag

IST EIN DMBetrag

IM BEREICH 0,00 .. 20,00

DER Blauton

IST EINE Farbe

IM BEREICH hellblau .. dunkelblau

Im ersten Fall ist der Trägertyp ein Datentyp, der selbst als Fixkommazahl definiert war. Im zweiten Fall ist der Trägertyp ein Aufzählungstyp.

Formale Syntax

(Regel 32.) *Unterbereichstypdeklaration* ::=

(**IST EIN** | **IST EINE**) *Trägertyp*
IM BEREICH *UntererWert* .. *ObererWert*

Formale Syntax, die man überlesen kann

(Regel 33.) Trägertyp ::= *Identifikator*

(Regel 34.) *UntererWert* ::= *Konstante*

(Regel 35.) *ObererWert* ::= *Konstante*

VI.2.5. Verbundtypen (Records)

Verbundtypen fassen mehrere Datenelemente zu einem einzigen Begriff zusammen. Unten sind als Beispiel eine Adresse und ein Punkt angegeben. Viele Datenmodellierungswerkzeuge (zum Beispiel ADW) bieten heute ebenfalls dieses Konstrukt an.

Beispiel für Verbundtypen

DER Punkt

BESTEHT AUS

X_Koordinate	:	Koordinate
Y_Koordinate	:	Koordinate

DIE Adresse

BESTEHT AUS

Straße	**:**	**STRING(**40**)**
Hausnummer	**:**	**NAT(**3**)**
Postleitzahl	**:**	PLZTyp
Stadt	**:**	**STRING(**40**)**

Formale Syntax

(Regel 36.) *Verbundtypdeklaration* ::=

BESTEHT AUS
(*SelektorName* *Kardinalität* *DatentypName*) +

Formale Syntax, die man überlesen kann

(Regel 37.) *SelektorName* ::= *Identifikator*
zur *Kardinalität* siehe (Regel 13.) auf Seite 35.
(Regel 38.) *DatentypName* ::= *Identifikator*

 © GDV, sd&m 1996

VII. Offene Schnittstellen

In VAA werden fachliche Spezifikationen immer wieder auf fachliche Funktionalität verweisen müssen, die in anderen Systemen noch nicht spezifiziert ist. Es ist wichtig, daß solche (fachlichen) Funktionen zentral gesammelt, konsolidiert und spezifiziert werden.

In den Freitxtbeschreibungen von Geschäftsprozessen und Sachbearbeiteraufträgen wird die Beauftragung von Sachbearbeitern stets mit dem Schlüsselwort **Sachbearbeiterauftrag** gekennzeichnet.

Es muß nun gelten, daß alle verwendeten (aufgerufenen) Sachbearbeiteraufträge in einer Spezifikation auch definiert sein müssen. Wenn es sich um Aufträge handelt, die in der Spezifikation soweiso spezifiziert werden, ist dies der Fall.

Wenn es sich um Funktionen handelt, die in der Spezifikation nicht definiert sind, sollen solche Aufträge im Abschnitt „Offene Schnittstellen“ genannt werden.

Das fachliche Chefdesign (ein Ansprechpartner des AK-VAA) muß von Zeit zu Zeit alle offenen Schnittstellenspezifikationen sammeln und mit den Teams abstimmen. Es kann dann entweder ein Sachbearbeiter sprezifiziert werden oder aber es wird zumindest eine einheitliche Schnittstelle (Name, Parameter, Effekt) für Gesamt-VAA festgelegt.

VII.1. Beispiel

Im einem Baustein Schaden/Leistung muß für die Bearbeitung von Schadensfällen stark auf die Texterstellung mittels eines beliebigen Korrespondenzsystems zurückgegriffen werden. In einem Sachbearbeiter Anspruch wird ein Sachbearbeiterauftrag Anspruch.ablehnen spezifiziert. Dieser Sachbearbeiterauftrag muß unter anderem auch einen Standardbrief erstellen lassen:

Auftrag ablehnen

Braucht Anspruchsnummer : AnspruchNr

Braucht Grund : Text

Vorbedingung

Der Anspruch ist vorhanden

Effekt

Der neue Zustand des Anspruchs ist abgelehnt. Es wird ein Standardschreiben erstellt mit **Sachbearbeiterauftrag** Korrespondenz.Brief(Anspruchsteller, Textbaustein, Grund, ...)

In einem Abschnitt „Offene Schnittstellen“ muß die benutzte Funktion beschrieben sein.

Auftrag Korrespondenz.Brief

Effekt

Erstelen eines Standardbriefes mit Übergabe eines Adressaten, eines Textbausteins und entsprechender Parameter

Die Aufträge sollten nach fiktiven Sachbearbeiternamen geordnet werden. Man erhält so besseren Überblick. Die Beschreibung von Parametern bleibt freigestellt.

VII.2. Formale Syntax

(Regel 39.) *OffeneSchniistellen* ::=

*OffenerAuftrag**

(Regel 40.) *OffenerAuftrag* ::=

Auftrag *SachbearbeiterName***.***AuftragsName*

*ParameterBeschreibung**

[**Vorbedingung**

Freitext]*

[**Nachbedingung**

Freitext]*

Effekt

Freitext

Formale Syntax, die man überlesen kann

(Regel 41.) *SachbearbeiterName* ::= *Identifikator*

(Regel 42.) *AuftragsName* ::= *Identifikator*

 © GDV, sd&m 1996

VIII. Mustergliederung

VIII.1. Ziele und Gültigkeit

Die folgende Mustergliederung ist ein Anhaltspunkt für den Aufbau der Spezifikation eines Anwendungsbausteines. Eine gute Spezifikation sollte nicht nur formal sein - sie sollte auch einen fachlichen Überblick bieten, der bis zu demselben Umfang haben kann, wie die formale Spezifikation selbst.

Das wesentliche Ziel der Mustergliederung ist es, zu erreichen, daß alle fachlichen Spezifikationen im Rahmen von VAA in etwa gleich aussehen, so daß man sich nach einer gewissen Einarbeitung in VAA-Spezifikationen gut zurechtfindet. Die Mustergliederung soll nicht dazu zwingen, jeden Punkt sklavisch auszufüllen. Oberstes Ziel für die Dokumentation ist Lesbarkeit und nicht das Befolgen von Dokumentationsrichtlinien.

Kapitel, die nicht benötigt werden, sollen also nicht mit trivialem Inhalt gefüllt, sondern lieber weggelassen werden.

Qualität der informellen Darstellungen

Eine detaillierte Untergliederung der fachlichen, informellen Teile der Spezifikation von Anwendungsbausteinen kann nicht durch eine stärkere Untergliederung sichergestellt werden. Ein einheitlicher und guter Stil wird vor allem durch gute Beispiele und eine konstruktive Qualitätssicherungsarbeit sichergestellt.

Einheitliche Methode ungleich einheitliche Qualität

Hier sei nochmals erwähnt, daß alleine das formale Befolgen der Spezifikationsmethode noch nicht sicherstellt, daß man Spezifikationen in guter oder im Rahmen von VAA ähnlicher Qualität erhält.

Dies kann nur durch gutes, sorgfältiges Spezifizieren und durch eine Qualitätssicherung erreicht werden, die die Dokumente mehrerer Teams vergleicht, um eine einheitliche Qualität zu erhalten. Diese Qualität muß vor allem inhaltliche Fragen stellen und keine formalen Fragen. Herumreiten auf formalen Kleinigkeiten zeugt leider zu oft von inhaltlicher Inkompetenz der Qualitätssicherung.

VIII.2. Mustergliederung

Die folgende Gliederung in Abbildung 16 zeigt ein Muster für die Gliederung fachlicher Spezifikationen für VAA. Die einzelnen Kapitel der formalen Spezifikation sind nach der jeweiligen Syntax weiter zu untergliedern. Für den Umfang der informellen Teile orientiere man sich an guten Musterspezifikationen. Als erstes Beispiel wird deshalb eine auszugsweise Spezifikation für das Arbeitsgebiet Schaden/Leistung zusammen mit dieser Mustergliederung zur Verfügung gestellt werden.

Da im Rahmen von VAA Anwendungsbausteine spezifiziert werden sollen, gehen wir davon aus, daß jeweils nur ein wesentlicher Geschäftsprozeß und damit auch nur ein Cluster in einem Dokument spezifiziert wird. Sollen mehrere Cluster (5-10 Sachbearbeiter, maximal 20-30 Entitäten) in einem Spezifikationsdokument beschrieben werden, wäre eine andere Gliederung vorzuziehen. Man kann dann auch gleich besser ein Dokument pro Anwendungsbaustein schreiben.

1	**Überblick und Abgrenzung des Anwendungsbausteins**
1.1	**Fachlicher Überblick**
1.2	**Abgrenzung**
2	**Fachliche Beschreibung des Anwendungsbausteins**
2.1	**Wesentliche Anforderungen**
2.2	**Lösungsalternativen**
2.3	**Beschreibung des Bausteins**
3	**Szenarien**
4	**Formale Beschreibung**
4.1	**Geschäftsprozesse**
4.1.1	**Geschäftsprozeß <Beispiel 1>**
4.1.n	**Geschäftsprozeß <Beispiel n>**
4.2	**Sachbearbeitermodell**
4.2.1	**Sachbearbeiter <Beispiel 1>**
4.2.n	**Sachbearbeiter <Beispiel n>**
4.3	**Batches**
4.3.1	**Batch <Batch 1>**
4.3.n	**Batch <Batch n>**
4.4	**Datenmodell**
4.4.1	**Überblick**
4.4.2	**Entität <Entität 1>**
4.4.n	**Entität <Entität n>**
4.5	**Datentypen**
4.6	**Offene Schnittstellen**

Abbildung 16: Mustergliederung für VAA-Anwendungsbausteine

VIII.3. Der „fachliche Beschreibungsteil“

Der fachliche Beschreibungsteil, der aus den Kapiteln 1 und 2 der Mustergliederung besteht, sollte vor allem aus gut lesbaren Texten, erläuternden Abbildungen und sonstigen Diagrammen und Darstellungen bestehen, die dem Leser einen Überblick über das von einem Anwendungsbaustein beschriebene Fachgebiet geben, ohne daß sich der Leser mit dem Formalismus des Kapitels 3 befassen muß.

Aussagen dazu, wie man die Kapitel 1 bis 2 schreibt sind leider schwierig. Man kann sich hier nur an gelungenen Beispielen orientieren und die in Kapitel I dieses Dokumentes dargestellten Regeln beachten. Deshalb wird ergänzend zu diesem Dokument eine Musterspezifikation er-

 © GDV, sd&m 1996

stellt. Ein Blick in diese Musterspezifikation ist allemal besser als die Lektüre eines seitenlangen Kapitels über „praktischen EDV-Journalismus“

Kapitel 1 - Überblick und Abgrenzung des Anwendungsbausteins

Das Kapitel soll eine, auch für Nichtexperten verständliche, Beschreibung des Inhaltes geben. Es sollen die wesentlichen Bestandteile des Anwendungsbausteins genannt werden, vor allem der wesentliche Nutzen für einen Anwender. Das Kapitel soll vor allem kurz (bis ca. 5 Seiten) sein.

Wichtig ist besonders der Abschnitt *„1.2 Abgrenzung“*, da hier Aussagen darüber gemacht werden, was der Anwendungsbaustein **nicht** leisten soll oder kann.

Kapitel 2 - Fachliche Beschreibung des Anwendungsbausteins

Das Kapitel enthält eine verständliche, fachliche und ausführliche Beschreibung des spezifizierten VAA-Bausteins. Dafür ist es immer hilfreich, die wesentlichen Anforderungen aufzulisten, denen man Rechnung getragen hat. Dies geschieht in Abschnitt *„2.1 Wesentliche Anforderungen“*.

Falls es Lösungsalternativen gibt, die die Anforderungen erfüllen können, sind diese im Abschnitt *„2.2 Lösungsalternativen“* zu beschreiben. Es ist stets wichtig, daß Leser nachvollziehen können, wie das Team zu der beschriebenen Lösung gekommen ist, und welche Anforderungen das fachliche Design, also die Auswahl einer Lösung, beeinflußt haben.

Im Abschnitt *„2.3 Beschreibung des Bausteins“* wird dann die Lösung beschrieben.

Dazu sollte man die wesentlichen Geschäftsprozesse und ihre EDV-Unterstützung als fachliche Übersicht, am besten in Form von Bearbeitungsbeispielen, beschreiben. Falls eine Geschäftsprozeßbeschreibung nicht sinnvoll ist (Stichwort Partnersystem), muß man den Abschnitt anders gliedern. Wir wollen und können hier keine Vorschriften machen.

Mit der Zahl guter Spezifikationen in VAA wird die Sicherheit wachsen.

Kapitel 3 - Szenarien

Das Kapitel enthält einige praktische Geschäftsvorfälle, die mit dem System bearbeitet werden können und gibt an, wie diese mit dem System abgewickelt werden. Dabei sollten Bezüge zum Sachbearbeiter- und zum Datenmodell hergestellt werden.

Beschreibungsmittel sind freier Text und Bilder, wobei zum Beispiel punktuell auch die Methode von Jacobson (Use Cases [Jac92]) sinngemäß eingesetzt werden kann.

IX. Checkliste Spezifikationsfehler

Dieses Kapitel enthält eine Liste typischer Spezifikationsfehler. Die Liste dient der Selbstprüfung und der Qualitätssicherung von Spezifikationen. Die vorliegende Liste soll den VAA-Spezifikationsteams dazu dienen, sich selbst und sein Projekt zu überprüfen. Einige Fehler sind recht drastisch formuliert. Dies soll nicht arrogant wirken, sondern dazu dienen, deja-vu-Erlebnisse beim Leser auszulösen. Die Fehlermöglichkeiten prägen sich dann besser ein, als das mit politisch korrekten Formulierungen erreicht werden kann.

Die einzelnen, beschriebenen Schwachstellen sind in Spezifikationen immer wieder zu finden. Die Checkliste stammt aus längerer Erfahrung mit der Qualitätssicherung von Spezifikationen und ist teilweise auch dem Buch „*Pitfalls ob Object Oriented Development*“ [Web95] entnommen. Die meisten der dort beschriebenen Fehler beziehen sich nicht ausschließlich auf objektorientierte Softwareprojekte sondern können auch in jedem anderen Softwareprojekt gemacht werden. Die zusätzliche Lektüre des Buches empfiehlt sich auf jeden Fall noch. Die dort noch wesentlich umfangreicher beschriebenen Management- und sonstigen Fehler werden leider auch in VAA-Projekten gemacht werden.

IX.1. Bilder und Texte

Fehlende Beschreibungstexte zu Geschäftsprozeß und Datenmodellbildern

Oft werden Geschäftsprozeßbilder oder Datenmodellbilder nicht beschrieben. Ein Übersichtsbild über ein Datenmodell ist nutzlos, wenn besonders die Beziehungen zwischen den Entitäten nicht ausreichend beschrieben werden. Ein Geschäftsprozeßbild sollte aus den selben Gründen beschrieben werden. Es ist nutzlos, wenn nicht mindestens der normale Ablauf des Prozesses beschrieben ist.

Zu viele Objekte in einem Bild

Menschen sind leider nicht besonders gut in der Lage, besonders komplexe Bilder und Zusammenhänge gut zu verarbeiten. Datenmodell-Übersichten mit 30 oder mehr Entitäten haben daher außer für den Zeichner selten einen verständnisfördernden Wert. 7 +/- 2 Elemente auf jeder Darstellungsebene sollten ausreichen. Gibt es mehr Elemente, so sollte man hierarchische Darstellungen mit einer hierarchischen Zerlegung wählen.

Inkonsistenz zwischen Bildern und Beschreibungen

Bilder und beschreibende Texte müssen konsistent sein. Dazu gehört insbesondere, daß genau die Identifikatoren in den beschreibenden Freitexten wiederverwendet werden, die auch in den Bildern verwendet werden. Bilder, auf die nicht im Text Bezug genommen wird oder Bilder, die nicht ausreichend erläutert werden, erschweren das Verständnis einer Spezifikation.

IX.2. Generelle Fehler

Verwechslung von Methoden mit gutem Design

Nur weil ein Projekt nach einer bestimmten Methode abgewickelt wird, die man für gut halten mag, sind die Projektergebnisse noch lange nicht gut. Dieser Punkt wurde bereits häufiger erwähnt, soll aber in der Checkliste nicht fehlen. Die hier verwendete Sachbearbeitermethode für sich genommen garantiert keine gute Spezifikation und kein gutes Design. Dafür sind gute Abstraktionen, konsequente Modularisierung und vor allem Orientierung auf den Leser erforderlich, um nur einige wichtige Punkte zu nennen, die sich nicht in einer Methode formalisieren lassen.

 © GDV, sd&m 1996

Zu hoher Anspruch der Spezifikation

Man sollte sich periodisch immer wieder fragen, ob der spezifizierte Systemumfang überhaupt notwendig ist. Hilfreich ist die alte 80/20 Regel, auch bekannt als Pareto-Prinzip. Oft erreicht man mit 20% des Aufwandes 80% Abdeckung des Gesamtsystems. Man muß sich fragen, ob das nicht ausreicht.

Wenn Spezifikationen des öfteren funktional erweitert werden, wenn man immer wieder über seltene Sonderfälle diskutiert, wenn dieselben Spezifikationsdiskussionen sich wiederholen, kann man überlegen, ob man nicht besser versucht, das Erreichte zu konsolidieren und die Spezifikation abzuschließen.

Design by Enumeration

Wenn es länger dauert Sachbearbeiter zu implementieren, als sie zu spezifizieren und wenn in einer Realisierung viele Änderungen auftreten, hat man vermutlich Fehler bei der Wahl der Abstraktion und der Begriffe gemacht. Oft kommt es vor, daß Spezifikation durch Aufzählung mit einem guten fachlichen Design verwechselt wird. Damit das nicht passiert, muß nach der ersten Aufzählung von Objekten und Funktionen ein gründliches Nachdenken über Abstraktionen und Abläufe erfolgen.

Im Rahmen der VAA-Spezifikationen fehlt hier das Feedback aus der Realisierung. Wenn man aber nicht das Gefühl hat, daß eine Realisierung mehr als eine triviale Transformation wäre, hat man vermutlich die Spezifikation noch nicht gut genug ausformuliert.

IX.3. Geschäftsprozeßmodelle

Geschäftsprozesse, die nicht DV-unterstützt sind werden beschrieben

Es verursacht viel Aufwand, solche Geschäftsprozesse zu beschreiben, die nicht vernünftig formalisierbar sind (Geschäft sichern, Wettbewerbsfähigkeit erhalten) oder die sowieso nicht von DV unterstützt werden. VAA beschäftigt sich mit der DV-Unterstützung von Versicherungen. Dem sollten die Geschäftsprozeßbeschreibungen Rechnung tragen.

Fehler bei der Granularität

Die betriebswirtschaftliche Organisationslehre ist viel älter als Geschäftsprozeßmodellierung - aber deshalb nicht weniger aktuell

Arbeitsschritte werden unnötig als Teilprozesse dargestellt. Arbeitsschritte werden nicht aufgebrochen, obwohl sie eigentlich Teilprozesse sind - man beachte die oben vorgestellten Kriterien und die einschlägige Literatur zur Organisationslehre.

Konsistente Zerlegung von Teilprozessen

Die Ein- und Ausgänge eines Teilprozesses müssen in diesem vollständig beschrieben sein. Man muß dies leider manuell prüfen, wenn man kein Tool besitzt.

IX.4. Sachbearbeiter

Triviale Sachbearbeiteraufträge

Häufig findet man Sachbearbeiteraufträge der Art

Vertrag.anlegen, Vertrag.ändern, Vertrag.löschen

Entweder passiert hier wirklich nichts anderes, als daß eine „Karteikarte“ angelegt, geändert oder vernichtet wird. In diesem Fall sollte man solche Aufträge nicht beschreiben. Trivialitäten ermüden den Leser. Oder aber es gibt fachliche Dinge, die bei der Änderung eines Vertrages passieren. Dann ist vermutlich der Name des Auftrages falsch gewählt. Statt *Vertrag.ändern* kann man dann zum Beispiel Aufträge definieren, wie *Vertrag.RisikoHinzufügen* oder *Vertrag.PrämieÄndern*.

Keine überkomplexen Sachbearbeiter

Don’t create swiss army knife objects [Web95]

Schweizer Armeemesser sind faszinierende Gegenstände. Sie bieten Werkzeuge für einfach alles. Nur ist der Korkenzieher eines Schweizer Armeemessers einem spezialisierten Korkenzieher meist unterlegen. Man sollte also darauf achten, daß Sachbearbeiter nicht zu komplex werden. Sachbearbeiter mit mehr als 15-20 Methoden sollten zum Nachdenken veranlassen, ob ein spezialisierter Sachbearbeiter die Aufgabe nicht besser wahrnehmen kann. Dies gilt eingeschräkt auch für Schnittstellensachbearbeiter (Teamchefs). Zuviele Methoden deuten auf Mängel in der Abstraktion und unnötige Komplexität.

Vermeiden sie gläserne Sachbearbeiter

Don’t break user code

Wenn ihre Sachbearbeiter viele Aufträge kennen, wie GebenVertragsdatum, SetzeVertragsdatum werden sie gläsern. Der Kunde des Sachbearbeiters kennt alle Interna. Der Sachbearbeiter wird durchsichtig. Das Prinzip der Datenkapselung wird aufgeweicht.

Konkretes Problem ist, daß Änderungen der Implementierung des Sachbearbeiters den Code der Anwender des Sachbearbeiters brechen werden. Genau das soll durch Datenabstraktion vermieden werden.

Vermeiden sie den Bruch von Bildern und Metaphern

Write user code

Wenn Sie Sachbearbeiteraufträge spezifizieren, sollten Sie eine sinnvolle Folge von Aufträgen aufschreiben. Wenn sich dieser Code (*user code*) gut lesen läßt, haben sie gut gearbeitet. Sie haben mit den Sachbearbeitern eine „neue Sprache“ definiert, die sich gut verstehen läßt. Die Metaphern und Begriffe, die sie verwenden, scheinen gut zu sein.

Wenn sich eine solche Folge nicht flüssig lesen läßt und nicht klar ist, was eigentlich gemeint ist, dann sollte man über die Namensgebung für die Aufträge, die Metaphern und die Aufteilung der Sachbearbeiter noch mal nachdenken. Es ist wahrscheinlich, daß man noch eine bessere Lösung finden kann.

 © GDV, sd&m 1996

Divide et impera

Spaghetti are nice stuff for lunch
but avoid spaghetti objects

Spätestens seit der Nato-Konferenz 1968 über Software-Engineering ist „Teilen und Herrschen“ eine bekannte Regel bei der Gestaltung von Softwaresystemen. Verschlungene, komplexe Systeme können funktionieren. Es ist aber wahrscheinlicher, daß sich daraus Probleme bei der Wartung, beim Verständnis und beim Test ergeben. Man sollte also, wie auch bei Datenmodellen, darauf achten, daß Sachbearbeiter in „überschaubaren Teams“, sogenannten Clustern mit einer klar definierten Außenschnittstelle arbeiten. Dann sind sie besser zu testen, die Systeme sind leichter verständlich und Änderungen können besser vor den Benutzern versteckt werden. Als Faustregel gilt wieder, daß man versuchen sollte, aus 7 +/- 2 Sachbearbeitern einen Cluster zu bilden. Hat man eine Spezifikation mit 20 oder mehr Sachbearbeitern ohne weitere Struktur vor sich, dann wurde meist ein Fehler gemacht.

IX.5. Datenmodelle

Unzureichende Erläuterung von Übersichtsbildern

Für jedes Datenmodell muß ein Übersichtsbild vorhanden sein. In einer Erläuterung zu Übersichtsbildern muß jede Beziehung und jede Entität genannt und kurz erläutert werden. Insbesondere sind die Beziehungen auch zu benennen. Fehlt eine solche Beschreibung, so ist eine Übersicht für Außenstehende oft schwer nachvollziehbar.

Unzureichende fachliche Beschreibungen von Entitäten

Entitäten werden zwar rein mathematisch gesehen durch die Menge ihrer Attribute beschrieben - dies hilft dem Leser einer Spezifikation aber wenig. Eine saubere fachliche Definition jeder Entität muß also erfolgen.

Unzureichende fachliche Beschreibungen von Attributen

Oft werden Attribute nicht ausreichend beschrieben. Maßstab muß immer ein Leser sein, der kein fachlicher oder DV-technischer Experte ist. Nehmen wir an, wir wollen das Attribut *Rückkaufswert* der Entität *Lebensversicherung* beschreiben, dann muß fachlich genau definiert werden, wie sich dieser berechnet. Eine pure Beschreibung durch den Datentyp *DM* reicht nicht aus.

Übertriebene fachliche Beschreibungen von Attributen

Beschreibungen der Art

Geburtsdatum - Datum, an dem der Vertragspartner geboren wurde

sollten vermieden werden, weil sie den Leser ermüden.

Fehler bei der Datenmodellierung

Datenmodelle sollten eine Qualitätssicherung durch erfahrene Datenmodellierer unterzogen werden. Die Fehler, die man bei der Datenmodellierung machen kann sind zahlreich - dazu sei auf die Literatur und auf Datenmodellierungshandbücher der Mitgliedsunternehmen von VAA verwiesen.

X. Anhang

X.1. Auswahl der Methode

Sucht man nach einer geeigneten Spezifikationsmethode für die fachliche Funktionalität von VAA, so hat man die Qual der Wahl aus einem Berg strukturierter und objektorientierter Methoden. Bei der Auswahl einer geeigneten Methode mußten folgende Faktoren berücksichtigt werden:

- Nur ca. 20% der Mitglieder der diversen VAA-Arbeitskreise sind mit objektorientierten Methoden vertraut. Die überwiegende Mehrheit hat in der Praxis mit Daten- und Funktionenmodellierung gearbeitet. Dies verbietet den sofortigen Einsatz „harter“ Objektorientierung für die fachlichen Teile von VAA. Die Sachbearbeiter-Methode fügt „klassischen“ Methoden nur wenig, aber ausreichend viel hinzu.

- Angesichts der Zukunftsorientierung von VAA soll der Weg zur Objektorientierung nicht verbaut, sondern erleichtert werden. Es wurden oben immer wieder Bemerkungen gemacht, daß man zum Beispiel Vorgangsobjekte und andere Objekte oft eleganter als mit Sachbearbeitern beschreiben kann. Auch wurde bemerkt, daß man sich, wenn man mit Objekten immer über eine Metaklasse kommuniziert, die Finger mit Primärschlüsseln als Funktionsargumenten wundschreibt. Dies ist leider nicht zu vermeiden gewesen.

- Geschäftsprozesse und deren enge Kopplung an das fachliche Modell der DV-Systeme mußten berücksichtigt werden. Geschäftsvorfälle passen sich in die Metaphern der prozeßorientierten Bearbeitung nahtlos ein.

- Für administrative kommerzielle Systeme werden nicht unbedingt objektorientierte Konzepte [Tay92] wie Vererbung, Polymorphie und Templates benötigt. Inwieweit Vererbung und generische Programmierung im Rahmen der fachlichen Bausteine von VAA eingesetzt werden können, ist noch fraglich. Wirksame Datenkapselung, wie durch die Sachbearbeiter-Methode bewirkt, hat sich in der Praxis als ausreichend erwiesen.

- Die Sachbearbeiter-Metapher [Den91] hat sich in zahlreichen, großen kommerziellen DV-Projekten bewährt. Für die Spezifikation technischer Basissoftware ist sie allerdings weniger geeignet.

- Nach der Sachbearbeiter-Methode spezifizierte Systeme wurden im großen Umfang problemlos in nicht-objektorientierten Programmierumgebungen realisiert. Eine Realisierung in objektorientierten Umgebungen ist ebenfalls mit einem einfachen, kanonischen Übergang möglich und wurde auch praktiziert. Der Weg zu reiner Objektorientierung, von der Metaklasse (Sachbearbeiter) zur Klasse ist ebenfalls noch gangbar.

Diese Gründe lassen uns annehmen, daß die hier dargestellte Sachbearbeiter-Methode [Den91] für die fachlichen Teile von VAA vorerst angemessen und geeignet ist.

 © GDV, sd&m 1996

X.2. Offene Fragen VAA

Die VAA-Architektur zielt in zwei Richtungen. Es wird ein technisches Rahmenwerk (Framework) spezifiziert, das bei jeder teilnehmenden Versicherung installiert sein muß, damit die eigentlichen fachlichen Anwendungen lauffähig sein können. Es wird weiter eine Menge versicherungsfachlicher Bausteine spezifiziert.

Wie man ein einheitliches, wiederverwendbares, technisches Framework spezifiziert, ist hinreichend bekannt. Wie man ein fachliches Rahmenwerk spezifiziert, das für verschiedene Unternehmen mit unterschiedlichen DV-Infrastrukturen, Geschäftsprozessen und Verfahren verwendbar ist, ist ein noch weitgehend offenes Thema.

Zu diesen Punkt gibt es in dem VAA-Dokument [VAA95] noch zahlreiche offene Punkte, die vom AK-VAA noch geklärt werden müssen. Die Lösung dieser Punkte wird auch Einfluß auf die endgültige Form der Spezifikation haben. Man kann aber nicht solange mit der Fortführung der fachlichen Spezifikationen warten, bis diese Punkte im Detail ausdiskutiert sind. Im folgenden sind die wesentlichen, derzeit noch offenen Punkte des AK-VAA genannt. Dies geschieht mit dem Ziel, Einschränkungen der Spezifikation, die in den vorhergehenden Kapiteln gemacht wurden, verständlich und nachvollziehbar zu machen.

X.2.1. Wiederverwendung

Bei Wiederverwendung von Software kann man zwischen

- Wiederverwendung von Konzepten,
- Wiederverwendung von Spezifikationen,
- Wiederverwendung von Sourcecode und
- Wiederverwendung von Binärcode (Libraries, Bibliotheken)

unterscheiden. Orthogonal dazu kann man noch zwischen

- fachlicher Wiederverwendung und
- technischer Wiederverwendung,

unterscheiden. Technische Wiederverwendung bezieht sich auf die technische Softwarebasis von Softwaresystemen, also zum Beispiel Datentypen, Fensterbibliotheken, Programmrahmen und ähnliches. Fachliche Wiederverwendung bezieht sich auf komplette Systeme (Standardsoftware), einzelne betriebliche Bereiche (Partnersystem) oder Algorithmen fachlicher Natur (Produktdesign für Lebensversicherungen, Lebensversicherungs-Tarifberechnung).

X.2.1.1. Technische Wiederverwendung

Es gibt in der Praxis der Softwareentwicklung ausreichend Beispiele dafür, daß Wiederverwendung technischer Softwarebausteine funktioniert - und zwar lange vor dem Auftauchen der Objektorientierung.

Funktionsbibliotheken und Programmrahmen gab es schon unter Assembler, Fortran und COBOL. In der objektorientierten Softwareentwicklung gibt es durch die Technik der Vererbung und der generischen Programmierung (Templates in C++ und ADA) weitergehende Möglichkeiten der Wiederverwendung. Dies führt heute zum Beispiel zum Einsatz von objektorientierten Klassenbibliotheken für GUI-Programmierung und zum Einsatz von Anwendungs-Frameworks. Diese gehen in ihrem Komfort über Programmrahmen (cut & paste) und Funktionsbibliotheken (nur Aufrufe möglich) hinaus.

Technische Wiederverwendung bis zur Wiederverwendung von Binärobjekten (zum Beispiel OO-Frameworks, Funktionsbibliotheken) ist stark verbreitet. Sie wird in dem im Rahmen von VAA zu spezifizierenden, technischen Framework sicher ihre Verwendung finden.

X.2.1.2. Fachliche Wiederverwendung

Problematischer ist die Wiederverwendung von Softwarebausteinen im Bereich betriebswirtschaftlicher Software, vor allem dann, wenn die einsetzenden Unternehmen nicht bereit sind, ihre Geschäftsprozesse an die gekauften Softwarebausteine anzupassen. Beispiele von „binärer Wiederverwendung“ finden sich im Bereich von Standardsoftware, so zum Beispiel bei SAP.

Am anderen Ende der Skala findet man Wiederverwendung von Konzepten bei den sog. Referenzmodellen, wie ARIS, FAA, BKDM, IAA, RAA um nur einige Beispiele für Referenzmodelle zu nennen.

Es finden sich ausreichend Beispiele für erfolgreiche fachliche Wiederverwendung, wenn sich die Unternehmen an die Systeme anpassen (SAP) oder wenn die Schnittstellen schmal sind und gleichzeitig der Programmieraufwand hoch ist (versicherungsmathematische Funktionen).

Umgekehrt finden sich viele gescheiterte Versuche von Unternehmen, die versucht haben zusammen gemeinsam Bausteine für ihre Kernkompetenzen zu entwickeln, wie zum Beispiel BKDM - eine Kooperation deutscher Banken und die zugehörigen Folgeaktivitäten.

VAA hat den Anspruch, einen Markt für Softwarebausteine zu schaffen, die wiederverwendbar sind, ohne daß sich die Unternehmen an einen genormten Prozeß anpassen. Über die Lösung der dabei auftretenden Probleme, wie die

- Anpassung an Geschäftsprozesse,
- das Einpassen in eine bestehende DV-Landschaft,
- die Anpassung an bestehende Datenmodelle,
- die Interoperabilität von Bausteinen verschiedener Hersteller und
- den Austausch und die Fortschreibung von Versionen,

sind aber in [VAA95] noch keine ausreichend präzisen Aussagen gemacht. Dies wird noch Aufgabe des AK-VAA sein. Bis dies geschehen ist, sollte man davon ausgehen, daß in den fachlichen Arbeitskreisen Referenzmodelle entworfen werden und keine monolithische Standardsoftware spezifiziert wird.

Damit liegen die fachlichen Arbeitskreise, wie Partner und Schaden/Leistung auf der sicheren Seite, da sich ein Referenzmodell allemal zu einem Standardprodukt verfeinern läßt.

X.2.2. Begriff des Anwendungsbausteins

Der Begriff des Anwendungsbausteins wird in der VAA-Architektur stark strapaziert. Er ist allerdings nicht sauber definiert. So gibt es in [VAA95] keine Aussagen darüber, welche Größe ein solcher Anwendungsbaustein haben soll, wieviele Funktionen in ihm zusammengefaßt werden sollen oder ob es sich bei einem Anwendungsbaustein um das handeln soll, was andere als „Business Object“ bezeichnen, wobei auch dieser Begriff in der Literatur nicht ausreichend definiert wird.

 © GDV, sd&m 1996

Als Beispiel für einen Anwendungsbaustein wurde informell zum Beispiel eine „Tarifmaschine“ genannt, aber nicht durch Beispiele präzisiert.

Es gibt derzeit bei VAA noch keine präzisen, schriftlichen Aussagen darüber, wie ein fachlicher Anwendungsbaustein in einem VAA-konformen Unternehmen installiert werden kann, wie Anwendungsbausteine verschiedener Hersteller kombiniert werden können etc.. Diese Diskussion muß und wird im AK-VAA noch geführt werden.

X.2.3. Referenzmodelle kontra „objektorientierte Wiederverwendung“

Inwieweit bei der fachlichen Wiederverwendung die Konzepte der Vererbung und der generischen Programmierung eingesetzt werden können, ist noch unklar. Was technische Subsysteme anlangt, gibt es dafür erfolgreiche Beispiele (GUI-Frameworks). Bei fachlicher Wiederverwendung kennen wir bisher nur gescheiterte Versuche.

Man liegt daher also auch hier auf der sicheren Seite, wenn man zunächst nur Referenzmodell spezifiziert.

X.2.4. Geschäftsprozeßmodelle

Gerade durch die Gestaltung der Geschäftsprozesse haben die VU die Möglichkeit, sich im Wettbewerb zu differenzieren. Wird einem Unternehmen in einer Hilfsfunktion, wie zum Beispiel dem Finanz- und Rechnungwesen durch SAP, ein genormter Geschäftsprozeß aufgezwungen, so hat dies i.A. keine Auswirkungen auf die Wettbewerbsposition.

Wird einem Unternehmen aber in seinem Kerngeschäftsfeld ein genormter Geschäftsprozeß aufgezwungen, so verliert es seine Gestaltungsspielräume im Wettbewerb. Die VAA kann also nicht akzeptiert werden und somit erfolgreich sein, wenn sie ihren Anwendern genormte Geschäftsprozesse aufzwingt.

Damit kam im AK-VAA die Frage auf, inwieweit es im Rahmen der fachlichen Arbeitskreise überhaupt sinnvoll ist, Geschäftsprozesse zu spezifizieren. Die Frage ist noch nicht abschließend diskutiert.

Bis dahin sollten die zu spezifizierenden Geschäftsprozesse also als Referenzmodelle aufgefaßt werden.

X.2.5. Benutzungsschnittstelle

Durch EDV automatisierte Geschäftsprozesse manifestieren sich besonders auch in der Benutzungsschnittstelle eines Softwaresystems. Damit wäre eine Benutzungsschnittstelle ebenfalls allenfalls als Referenzmodell spezifizierbar. Die Frage ob und in welcher Tiefe in VAA Benutzungsschnittstellen spezifiziert werden sollen, muß im AK-VAA ebenfalls noch abschließend diskutiert werden.

Bis dahin macht es auch wegen des hohen, damit verbundenen Aufwandes keinen Sinn, Benutzungsschnittstellen im Rahmen der fachlichen Arbeitskreise von VAA zu spezifizieren.

X.2.6. Abgrenzung der Workflow- und der DV-Vorgangssteuerung

Die Workflow- und die DV-Vorgangssteuerung werden zwar in der VAA Softwarearchitektur (siehe auch Abbildung 3) genannt, nicht jedoch funktional gegeneinander abgegrenzt.

Es gibt dazu Aussagen, die noch der Diskussion durch ein Chefdesign bedürfen. Zum Beispiel die Aussage, daß sich Anwendungsbausteine nicht gegenseitig aufrufen dürfen, sondern ausschließlich durch die DV-Vorgangssteuerung aufgerufen werden. In dem Bild fehlt ferner die Präsentationsschicht, so daß nicht klar ist, wie eine Eingabe in einem Dialog verarbeitet wird.

Angesichts der Tatsache, daß auch der Begriff Anwendungsbaustein nicht eindeutig definiert ist, kann in der Spezifikation der Teil der DV-Vorgangssteuerung schlecht beschrieben werden. Was diese tut, ist völlig offen.

Folglich kann bei der Spezifikationsmethode auf diesen Aspekt keine Rücksicht genommen werden. Eventuelle spätere Umgruppierungen der Spezifikation können erforderlich werden.

Zum Vergleich mit dem jetzigen Stand der VAA-Architektur (Abbildung 3, Seite 4) wird hier noch die Standardarchitektur von sd&m gezeigt (siehe Abbildung 17). Hier stellt die Workflow-Steuerung ein normales Anwendungssystem dar (Kasten Geschäftsprozesse). Aus diesem Anwendungssystem werden andere Anwendungen, die betrieblichen Informationssysteme aufgerufen. Die Kommunikation mit ihnen erfolgt über die Schnittstellen-Objekte. Dem Benutzer wird also zum Beispiel ein Dialog präsentiert. Aus diesem Dialog erteilt der Benutzer Aufträge an den Anwendungskern.

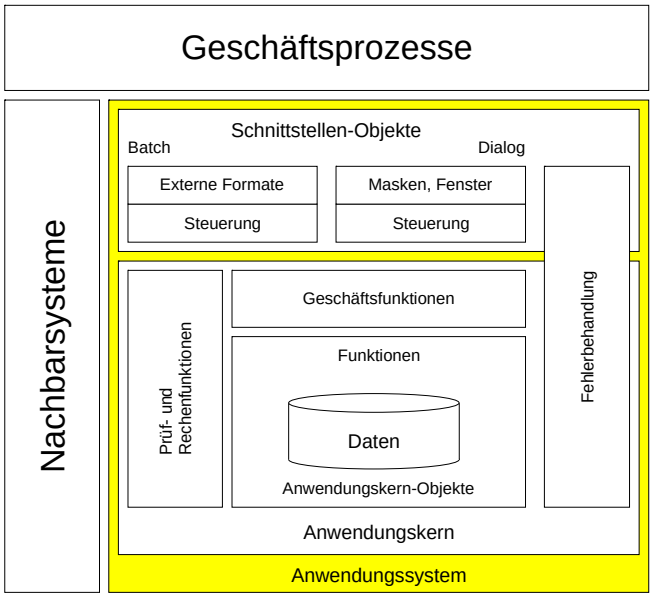


Abbildung 17: Eine andere Standardarchitektur (Quelle sd&m)

Hier gibt es eine klare Trennung zwischen der Workflow-Steuerung (Geschäftsprozesse), der Dialogsteuerung (Schnittstellen-Objekte) und der DV-Vorgangssteuerung (Geschäftsfunktionen, Anwendungskern-Objekt = Sachbearbeiter). Das Modell von VAA hat sicher auch Vorteile, die sich allerdings erst ermitteln lassen, wenn es präzise spezifiziert ist.

 © GDV, sd&m 1996

X.3. Literaturverzeichnis

[Cur+92] Curtis, B.; Kellner, M. I.: *Process Modelling*, CACM (35), September 1992, pp 75 - 90.

[Dat86] C. J. Date, *„An Introduction to Database Systems“*, Addison-Wesley 1986 - Inzwischen in zahlreichen überarbeiteten Auflagen erschienen.

[Dav94] Davenport, T., *'Process Innovation: Reengineering Work Through Information-Technology'*, Harvard Business School Press, 1993

[DeM78] Tom DeMarco: *„Structured Analysis and System Specification“*, Yourdon Press, New York 1978.

[Den91] Ernst Denert: *„Software Engineering“*, Springer Verlag 1991.

[Den93] Ernst Denert, E.: *Dokumentenorientierte Software-Entwicklung*; Informatik Spektrum (1993) 16: S. 159 - 164.

[Fas95a] *„Geschäftsprozeßmodellierung BMW/SM-15“*, FAST e.V., München, 1995, erhältlich über Internet-Email: info@fast.de

[Fas95b] *„Modellierung von Geschäftsprozesse, Projektergebnis Teil 2 - Metamodell“*, FAST e.V., München, 1995, erhältlich über Internet-Email: info@fast.de

[Fer+93] Ferstl, O. K.; Sinz, E. J.: *Geschäftsprozeßmodellierung*; Wirtschaftsinformatik, 35 (1993) 6, S. 589 - 592.

[Gol+95] Adele Goldberg, Kennth Rubin: *„Succeeding with Objects“*, Addison-Wesley 1995.

[Gra95] Ian Graham: *„Migrating to Object Technology“*, Addison-Wesley 1995.

[Ham+94] Hammer, M.; Champy, J.: *Business Rengineering*; Campus Verlag 1994.

[Jac75] Michael A. Jackson: *„Principles of Program Design“*, London. Academic Press 1975.

[Jac83] Michael A. Jackson: *„System Development“*, Prentice Hall 1983.

[Jac92] Ivar Jacobson: *„Object Oriented Software Engineering“*, Addison-Wesley 1995.

[Kil+93] Kilbert, K.; Gryczan, G.; Züllighoven, H.: *Objektorientierte Anwendungsentwicklung*; Vieweg Verlag 1993.

[Mey88] Bertrand Meyer: *„Object Oriented Software Construction“*, Prentice Hall 1988.

[Mül+93] Müller-Luschnat, G.; Hesse, W.; Heydenreich, N.: *Objektorientierte Analyse und Geschäftvorfallmodellierung*; Proceedings Emisa, Klagenfurt 1993.

[Pro93] *Unterlagen über das Vorgangssteuerungssystem Prominand*, IABG, München 1993.

[Som89] Ian Sommerville: *„Software Engineering“*, Addison-Wesley 1989.

[Tay92] David A. Taylor: *„Objektorientierte Technologien - Ein Leitfaden für Manager“*, Addison-Wesley 1992.

[VAA95] Gesamtverband Deutscher Versicherer (GDV), *„VAA - Die Anwendungsarchitektur der Versicherungswirtschaft“*, Bonn 1995.

[Vet82] Max Vetter, *„Aufbau betrieblicher Informationssysteme (Datenmodellierung)“*, Teubner Verlag 1982.

[VMo93] Adolf-Peter Bröhl, Wolfgang Dröschel, *„Das V-Modell - Der Standard für die Softwareentwicklung mit Praxisleitfaden“*, Verlag Oldenbourg, 1993.

[Vor93] Tagung: *"Marktübersicht Workflow Management, Vorgangssteuerung, Servicesysteme"*; Mummert und Partner, Frankfurt am Main 1993. Wird jährlich aktualisiert.

[Wal+95] Kim Walden, Jean-Marc Nerson: *„Seamless Object-Oriented Software Architecture“*, Prentice Hall 1995.

[Web95] Bruce F. Webster: *„Pitfalls of Object Oriented Development“*, M&T Books, 1995.

[Wit93] Wittlage, H.: *Unternehmensorganisation*, 5 Auflage, Verlag nwt 1993.

 © GDV, sd&m 1996

X.4. Historie der Versionen dieses Papiers

X.4.1. Version 0.a

Bearbeiter: Wolfgang Keller, sd&m

Entstand zwischen dem 8.1.96 und dem 22.1.96. Review am 22.1.96 durch den AK-VAA. Dabei wurden als wesentliche Änderungen gefordert:

- Kapitel für Benutzungsszenarien (Use Cases) in der Mustergliederung.
- Abschnitt für „offene Enden“ einführen, also Abschnitt für nicht spezifizierte Funktionalität von Nachbarsystemen.
- Klarer Ausschluß der Verwendung des Handbuches für technische Subsysteme.
- Begrifflichkeit muß teilweise noch an VAA angepaßt werden.

X.4.2. Version 0.b und 1.0

Bearbeiter: Wolfgang Keller, sd&m

Version 0.b entstand für das Meeting des AK-Schaden/Leistung am 25.1.96 in Wuppertal als Zwischenstand. Einige der zu Version 0.a geforderten Änderungen konnten berücksichtigt werden. Die Version 1.0 ist gegenüber der Version 0.b bis auf minimale Änderungen nicht mehr verändert worden, da ein detailliertes Feedback der AK-VAA-Mitglieder bis zum 31.1.96 nicht vorlag. Die unter der Version 0.a geforderten Änderungen sind bis auf den letzten Punkt (Begrifflichkeit) eingearbeitet und konsistent mit der Musterspezifikation.

Da bei der Vorstellung im AK Schaden/Leistung am 25.1.96 ebenfalls keine wesentlichen Einwände aufkamen, sind auch keine wesentlichen Änderungen für Version 1.0 nötig gewesen. Das Einarbeiten der Batches kann später erfolgen.