Objektorientierte Produktsysteme im Versicherungsbereich

Jens Coldewey
Coldewey Consulting
Uhdestr. 12
D-81477 München
Tel: +49-89-74995702
Fax: +49-89-74995703
email: jens_coldewey@acm.org
http://www.coldewey.com

© Coldewey Consulting, 1999

Die Fähigkeit, schnell neue Produkte einzuführen, zählt immer mehr zu den strategischen Wettbewerbsvorteilen von Versicherungen. Allerdings hinkt die Datenverarbeitung meist hinter den Anforderungen der Fachabteilungen hinterher. Die fachliche Komplexität des Problems sprengt die Grenzen herkömmlicher Hostsysteme, während die technischen Anforderungen aus dem täglichen Betrieb an die Grenzen des lehrbuchmäßigen Einsatzes objektorientierter Technik stoßen.

Jens Coldewey ist unabhängiger Berater mit Schwerpunkt Architektur und Objekttechnologie in großen Organisationen. In den letzten Jahren hat er mehrere Versicherungen bei Architekturfragen unterstützt. Herr Coldewey ist Kolumnist in der Zeitschrift ObjektSpektrum.

Der fachliche Hintergrund

Durch die Deregulierung im Versicherungsmarkt ist der Wettbewerbsdruck auf die einzelnen Unternehmen in den letzten Jahren spürbar gestiegen. Je nach der Ausgangsposition am Markt stehen dem Unternehmen drei verschiedene Strategietypen zur Auswahl: Kostenführerschaft, Eroberung von Nischenmärkten, oder eine Differenzierungsstrategie, bei der die Produkte auf besonders geeignete Art auf die Bedürfnisse der Zielkunden zugeschnitten sind [Por96]. Die meisten Unternehmen entscheiden sich für eine Differenzierungsstrategie: Die Kostenführerschaft kann in der Regel nur von einem einzelnen Unternehmen erfolgreich erlangt werden, während sie für Wettbewerber zu ruinösen Preiswettkämpfen führen kann. Eine Nischenstrategie eröffnet meist nicht das Marktvolumen, das ein großes Versicherungsunternehmen dauerhaft benötigt.

Bei den meisten Differenzierungsstrategien hingegen versuchen die Unternehmen sich durch innovative Produkte am Markt hervorzuheben. Dabei sind derzeit vier Trends zu beobachten:

- Die Verknüpfung von Lebensversicherungen mit allgemeinen Finanzdienstleistungen („Allfinanz“).

- Die Weiterentwicklung klassischer Versicherungsleistungen hin zum umfassenden Risiko Management mit umfassenden Dienstleistungen für den Kunden („Total Care“).

- Der Aufbau innovativer Vertriebswege, insbesondere über das Internet.

- Das Angebot neuer Deckungen, insbesondere der Aufbau spartenübergreifender Produkte.

Die meisten Unternehmen verfolgen keinen dieser Trends in Reinform, sondern versuchen, entsprechend ihrer jeweiligen Zielgruppe einen geeigneten Mix zu finden.

Zusätzlich zu den gesetzlichen Änderungen hat in den letzten Jahren die Markentreue der Kunden spürbar nachgelassen, viele Kunden sind also bereit, bei auch nur geringfügig besseren Angeboten das Unternehmen zu wechseln. Bei innovativen Deckungen ist es daher für das Unternehmen wichtig, schnell auf Änderungen am Markt reagieren zu können. So kann z.B. eine neue Form der Familienversicherung schnell zu empfindlichen Prämieneinbußen führen, wenn nicht schnell mit einem vergleichbaren Tarif nachgezogen wird.

Viele Unternehmen sind diesen Anforderungen noch nicht in vollem Umfang gewachsen. Die Entwicklungszeit für neue Produkte beträgt derzeit teilweise noch zwei Jahre. Ein großer Teil dieser Zeit vergeht mit fachlichen Abstimmungen, aber auch mit der Anpassung der verarbeitenden Systeme an die neuen Produkte. Prozeßexperten gehen davon aus, daß sich diese Zeit auf ein halbes Jahr reduzieren läßt, wenn die fachliche Entwicklung des Projektes richtig gemanagt wird und die Zeiten für die DV-Umsetzung entfallen. Ein so verkürzter Entwicklungsprozeß bedeutet eine erhöhte Beweglichkeit am Markt und damit ein direkter strategischer Wettbewerbsvorteil.

Die DV-Umsetzung auf ein Minimum zu verkürzen ist Aufgabe eines *Produktsystems*. Der Einsatz eines solchen Systems bedeutet eine radikale Änderung in der Anwendungslandschaft der meisten Versicherungen: Statt für jedes Produkt ein eigenes Bestands- und Schadenssystem zu erstellen, kommt ein einziges, entsprechend flexibles System für alle Produkte zum Einsatz. Dies stellt höchste Ansprüche an die DV-Technik.[1]

Einige Definitionen

Zunächst ist es notwendig, drei Phasen im Leben eines Produktes zu unterscheiden:

1. Während der *Produktfindung* werden die existierenden Produkte und weitere Marktdaten analysiert und statistisch ausgewertet, um das fachliche Design des neu-

[1] Betriebswirtschaftlich entspricht dies genau genommen einer Strategie des „Mass Customization“. Dabei wird versucht, möglichst kundenspezifische Produkte mit den niedrigen Kosten der Massenproduktion zu verbinden. Siehe [APi97] für eine allgemeine Diskussion, sowie [Kel98] für deren Bedeutung im Rahmen von Produktsystemen.

en Produkts zu entwickeln. In dieser Phase spielen auch Marketingaspekte eine Rolle sowie die Unternehmensstrategie. Auch wenn hier durchaus elektronische Unterstützung denkbar ist, gibt es meines Wissens derzeit noch keine speziellen Systeme hierfür, wenn man von allgemeinen Statistik- und Datawarehousing Werkzeugen absieht.

2. Bei der *Produktdefinition* werden die einzelnen Komponenten des Produktes im Detail definiert und getestet. In der bisherigen DV-Landschaft war dies die Analysephase der Verarbeitungssysteme. Bei Einsatz eines Produktsystems wird das neue Produkt mit Hilfe spezieller Software am Bildschirm entworfen und getestet.

3. In der *Produktion* wird das Produkt schließlich verkauft. Abgeschlossene Policen werden im Bestandssystem verwaltet, auftretende Schäden oder sonstige Leistungen mit Hilfe des Schadenssystems geprüft und, falls die entsprechenden Forderungen berechtigt sind, beglichen. Zur Unterstützung des Verkaufspersonals kommen Angebotssysteme zum Einsatz und die Rentabilität eines Produkts wird schließlich mit statistischen Programmen ermittelt.

Unterstützung der Produktionssysteme

Ein Produktsystem, das die modellierten Produkte nicht in Produktion unterstützt, ist wertlos. Bei der Beurteilung eines Produktsystems ist es daher von großer Bedeutung, welchen Teil dieser Wertschöpfungskette von dem System gesteuert werden kann. So ist die Steuerung der Angebotssoftware technisch nicht allzu schwierig, bringt aber auch nur geringen wirtschaftlichen Nutzen. Der Anschluß eines Bestandssystems ist deutlich schwieriger, ist aber auch die Voraussetzung, um den Entwicklungszyklus effektiv zu verkürzen. Produktgetriebene Schadenssysteme sind derzeit noch im Experimentierstadium, während der Anschluß von Statistiksystemen eher eine Anforderung an Bestands- und Schadenssysteme darstellt, denn an das Produktsystem.

Die unterschiedlichen Produktionssysteme stellen auch unterschiedliche Anforderungen an das Produktmodell. So ist für Angebot und Bestand vor allem die Tarifierung von Bedeutung, während der Bestand zusätzlich noch in der Lage sein muß, die variierenden Vertragsdaten in der Datenbank zu speichern. Für die Unterstützung des Schadenssystems sind wiederum genaue Aufschlüsselungen der versprochenen Leistungen notwendig, sowie Regeln zur Identifikation von Glattläufern usw. Soll ein Produktsystem die gesamte Wertschöpfungskette unterstützen, also Angebot, Bestand und Schaden/Leistung, so muß es ein entsprechend flexibles Produktmodell anbieten.

Produktmodelle

Für Sachversicherungen hat sich eine Stücklistenmetapher mit einer baumartigen Modellierung als vorteilhaft erwiesen [SLe96] [Kel98]. So können versicherbare Objekte festgelegt werden und für diese vertragsrelevante Merkmale definiert werden. Für ein Gebäude könnten dies z.B. die Fläche, die Nutzungsart und die baulichen Vorkehrungen gegen Brand und Diebstahl sein. Zu diesen Objekten werden nun bestimmte Risiken

versichert, wie z.B. Brand, Einbruch, Vandalismus, Sturmschäden, usw. Für jedes Risiko garantiert der Vertrag eine bestimmte Deckung[2]. Für komplexere Produkte kann es auch sinnvoll sein, die Risiken nochmals nach Verkaufstarifen zu gruppieren, aus denen der Kunde auswählen kann. Klassisch sind hier in der Kraftfahrzeugversicherung die Haftpflicht als obligatorischer Bestandteil und Voll- oder Teilkasko als optionale Verkaufstarife.

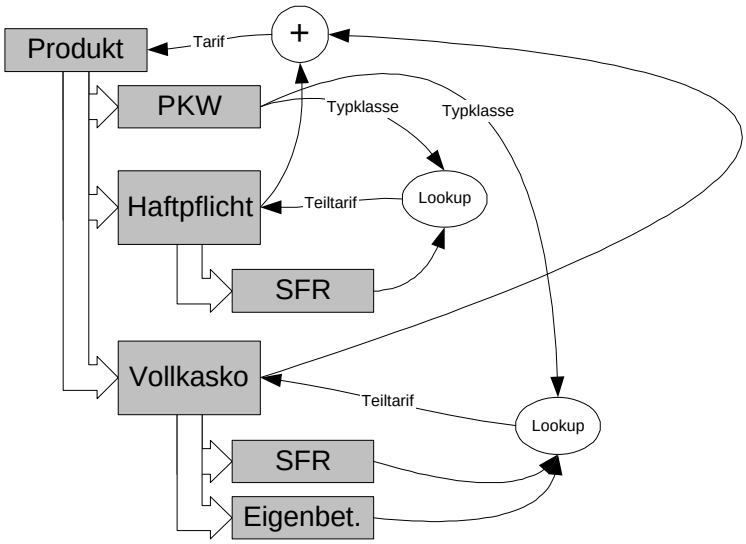

Abbildung 1: Stark vereinfachtes Modell eines Kraftfahrzeugtarifs.

Um nun Tarife, maximale Deckungen und Plausibiltätsregeln für solche Produkte festzulegen, können an diese Knoten Formeln angehängt werden, welche die gewünschten Werte aus den jeweiligen Eingabewerten berechnen (siehe Abbildung 1). Diese Formeln können ähnlich gestaltet sein, wie in Tabellenkalkulationen und dadurch auch für Experten der Fachabteilungen leicht erstellbar sein. Eine solche Modellierung ähnelt den attributierten Bäumen, die im Compilerbau zum Einsatz kommen [ASU86].

Im Detail ergeben sich noch weitere Anforderungen an ein Produktmodell, wie z.B. die Vergleichbarkeit verschiedener Produkte, deren Diskussion hier aber den Rahmen sprengen würden.

Dieser Ansatz trägt allerdings nur sehr begrenzt für Versicherungen mit einer starken Kapitalkomponente, also Lebens- und Krankenversicherungen. In diesen Produktarten differenzieren sich die Anbieter weniger über den Deckungsumfang, sondern eher über das Wirtschaften mit Altersrücklagen und Kapitaleinlagen. Wesentlicher Bestandteil dieser Produkte sind die Formelwerke, mit denen Prämien, Auszahlungsbeträge usw. aus der Vetragshistorie berechnet werden. Solche Berechnungen werden durch das beschriebene Modell nicht in übersichtlicher Form darstellen. Für die Steuerung eines Bestandssystems ist das Modell daher nicht ausreichend. Allerdings lassen sich Angebots-

[2] Einige Modellierungsansätze verwenden statt Risiken den Begriff „Ereignis“ und statt der Deckung den Begriff „Leistung“. Dies sind aber vor allem Unterschiede in der Terminologie.

systeme durchaus so steuern, wenn die dafür notwendigen mathematischen Formeln zur Verfügung stehen.

Abgrenzung des Systems

Ebenso wie die Frage, *was* ein Produktsystem steuern soll, ist auch die Frage, was *nicht* mit ihm kontrolliert wird, für den Projekterfolg entscheidend. So wäre es zum Beispiel denkbar, Aspekte des Workflows mit dem Produktsystem zu steuern, Oberflächen des Bestandssystems zu definieren, Dokumente zu verfassen, Richtlinien für Rückversicherungen aufzunehmen, usw. Zwar lassen sich für jede dieser Verbindungen gute Argumente anführen, die Realisierung der Basisfunktionen ist jedoch schon so schwierig, daß hier die Gefahr besteht, das Projekt zu überfrachten. Weniger ist hier oft mehr.

Die technische Einbettung

Wie ich bereits ausgeführt hatte, ist ein isoliertes Produktsystem ohne Nutzersystem nicht sinnvoll. Die Frage, wie die definierten Produkte in das Nutzersystem kommen, gehört daher zu den wesentlichen Entscheidungen im Entwurfsprozeß. Hier sind verschiedene Ansätze erprobt worden.

Der datenorientierte Ansatz

Insbesondere wenn Organisationen ihre ersten Versuche mit Produktsystemen machen, trifft man immer wieder auf den Ansatz, der in Abbildung 2 dargestellt ist: Das Produktsystem legt seine Daten in einer Produktdatenbank ab, aus der sich die Nutzersysteme dann die benötigten Informationen holen.

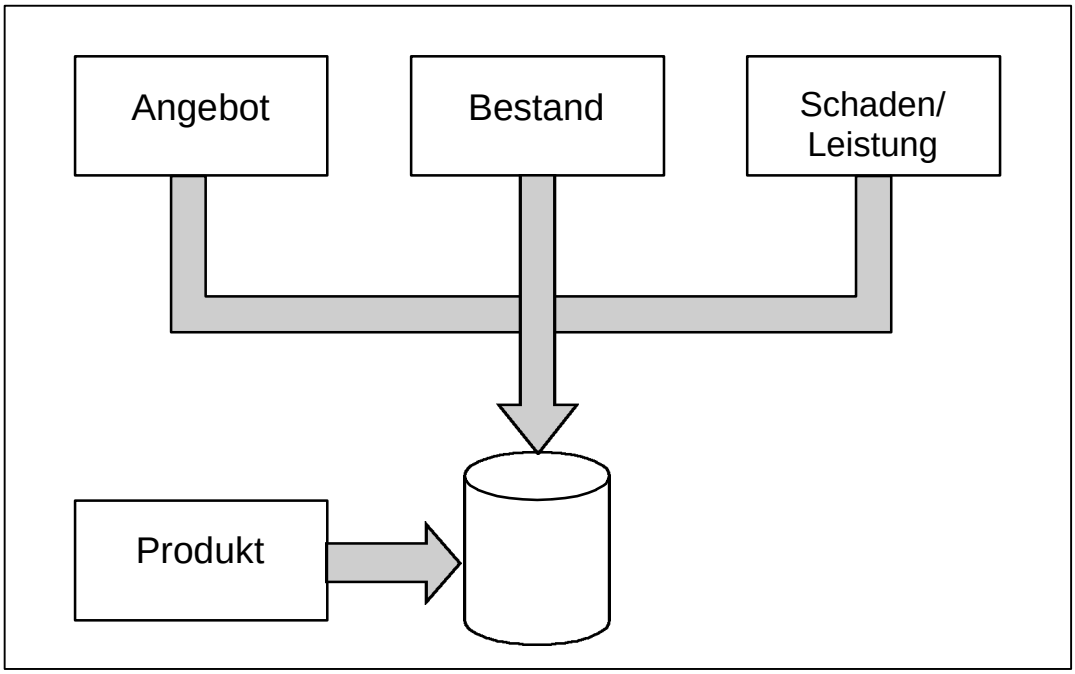

Abbildung 2: Datenorientierter Anschluß der Laufzeitsysteme an das Produktsystem

Auch wenn dieser Ansatz zunächst einleuchtend erscheint, hat er mehrere Nachteile. Neben allgemeinen methodischen Überlegungen, wie zum Beispiel verletzter Datenkapselung, trägt er nicht dem Umstand Rechnung, daß die Definition von Produkten ganz andere Anforderungen an die Datenbank stellt, als die Nutzersysteme. Während der Definition arbeiten die Anwender eher wie Entwurfsingenieure. Sie beschäftigen sich längere Zeit mit einem Teil des Systems, an dem sie immer wieder Änderungen machen. Ihre Arbeit folgt nicht einem definierten Ablauf, sondern ist eher intuitiv geprägt. Auch läßt sich nicht im voraus festlegen, welche Informationen zu welchem Zeitpunkt benötigt werden. Dies entspricht eher einem projektorientierten Ansatz, als einem transaktionsorientierten.

Im Gegensatz dazu unterstützen die Nutzersysteme meist klar definierte Arbeitsschritte, in denen eine bestimmte Datenmenge, zum Beispiel ein Vertrag geändert wird, bevor ein anderer Arbeitsschritt angegangen wird. Dies entspricht eher dem üblichen Verhalten von Transaktionssystemen.

Nachdem diese nicht-funktionalen Charakteristika wesentlichen Einfluß haben auf das Paradigma und das Design der Datenbank [Col98], resultiert dieser Ansatz in der Regeln in schlechter Performance und unhandlicher Bedienung des Systems.

Trennung zwischen Definition und Runtime

Um den jeweiligen Anforderungen der einzelnen Systeme optimal gerecht zu werden, empfiehlt es sich, die Systeme je nach Anforderungen zu trennen. Abbildung 3 zeigt ein Beispiel für eine solche Trennung.

Das Produktsystem selbst ist wegen der komplexen Daten und des großen Vererbungspotentials objektorientiert gebaut und verwendet eine Objektdatenbank. Diese Datenbank enthält alle Produktinformationen, wie sie vom Produktentwickler benötigt werden. Dies unterstützt auch die navigationsorientierte Arbeitsweise der Produktentwickler optimal.

Das Bestandssystem und das Schadenssystem sind typische Transaktionssysteme, die mit relationaler Technik erstellt sind. Das kommt neben der großen Verfügbarkeit und den meist schon vorhandenen Lizenzen auch dem Betriebsteam entgegen, das in der Regel über langjährige Erfahrung beim Betrieb großer relationaler Datenbanken verfügt. Diese Systeme verwenden speziell auf sie zugeschnittene Produktdatenbanken, die auf hohe Performance im Massenbetrieb optimiert sind. Da diese Datenbanken nicht die gesamte Produktinformation enthalten müssen, kann zum Beispiel auf die Darstellung der Produkthierarchie verzichtet werden, die in relationalen Datenbanken nur schwer modelliert werden kann.

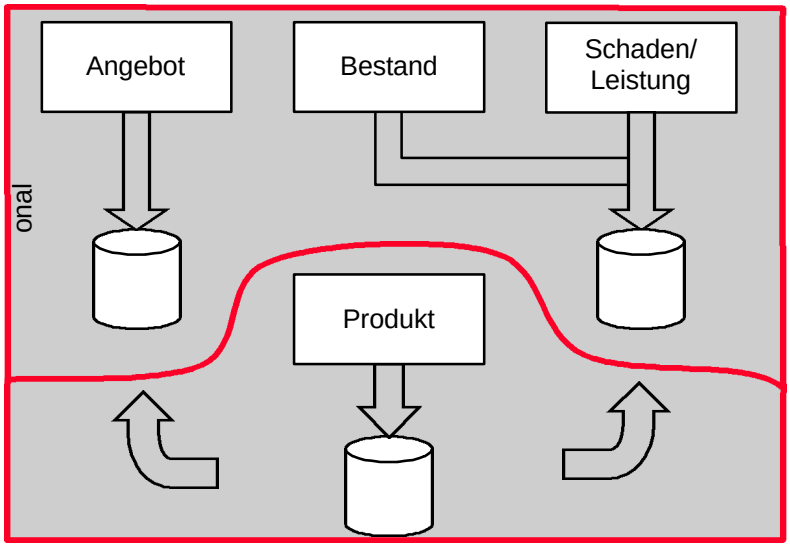

Abbildung 3: Trennung zwischen Runtime und Defintionssystem

Das Angebotssystem ist schließlich als typisches Außendienstsystem für Einzelbetrieb optimiert. Es verzichtet möglicherweise vollständig auf eine Datenbank für die Produktinformationen und verwendet statt dessen einfache Dateien. Dies erleichtert auch das Herunterladen neuer Produkte über das Internet.

Spezielle Subsysteme sorgen für den Export der Produktdaten aus dem Objektgeflecht in die jeweilige Darstellung des Zielsystems. Dies ist vergleichbar mit einem Compiler, der den gleichen Code auf verschiedene Zielplattformen übersetzen kann. In der Tat sind die dabei verwendeten Techniken durchaus mit Techniken des Compilerbaus vergleichbar.

Natürlich birgt dieser Ansatz auch Nachteile. Die Produktinformationen liegen redundant vor und die verschiedenen Darstellungen müssen konsistent gehalten werden. Allerdings scheint dies derzeit der einzige Ansatz zu sein, der auf der einen Seite eine intuitive Modellierung ermöglicht, auf der anderen Seite aber die Anforderungen das Betriebs an Performance und Verteilbarkeit erfüllt.

Design des Definitionsssystems

Das Definitionssystem stellt eine Reihe recht spezieller Anforderungen:

- Die Bedienung sollte auch für „normale“ Anwender leicht zu erlernen sein. Das erfordert eine gute Konsistenz der Oberfläche, die mit möglichst wenig unterschiedlichen Mechanismen auskommen sollte.

- Ziel des Systems ist die Modellierung innovativer Produkte, also solcher Produkte, die heute noch gar nicht erfunden sind. Das erfordert hohe Flexibilität und auch schnelle Anpassbarkeit.

- Die Bedienung nutzt vor allem Navigation: Der Anwender navigiert sich durch den Produktbaum und verschafft sich dabei einen Überblick über das Modell. Die einzelnen Arbeitsschritte sind keiner natürlichen Reihenfolge unterworfen.

- Das System ist nicht Online mit der sonstigen Landschaft verbunden, sondern bedient stand-alone einzelne Mitarbeiter oder Teams. Es ist also eher ein Einzelplatzsystem, keinesfalls aber in die Transaktionssysteme eingebunden.

Diese Anforderungen sprechen dafür, das Definitionssystem in objektorientierter Technik zu entwerfen. Die Erfahrung zeigt, daß nicht objektorientierte Systeme entweder kaum bedienbar sind, oder aber so schwer wartbar, daß an eine flexible Weiterentwicklung kaum zu denken ist.

Abbildung 4 zeigt das grundlegende Objektmodell für den attributierten Produktbaum. Es besteht im wesentlichen aus dem Produktbaum auf der linken Seite und dem Syntaxbaum für die angehängten Funktionen auf der rechten Seite. Beide sind mit einem Composite [GHJ+95] modelliert. Abseits von diesem Grundschema trifft man eine Vielzahl verschiedener Designentscheidungen für Detailprobleme.

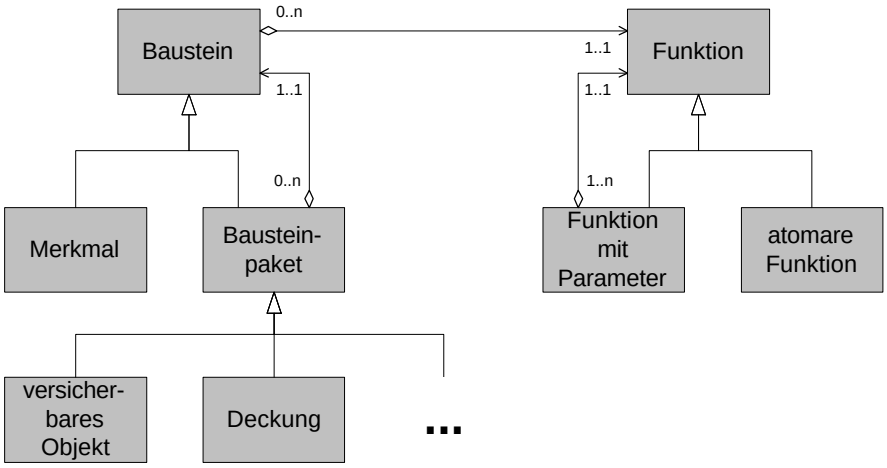

Abbildung 4: Grundsätzliches Objektmodell des Definitionssystems

Design der Runtime

Wie ich bereits zuvor geschildert hatte, haben Angebotssystem und Backoffice-Systeme grundsätzlich unterschiedliche Anforderungen an die Runtime. Während für Angebotssysteme vor allem die Berechnung von Tarifen wichtig ist, wobei die Produktstruktur auch in aufwendig zu verarbeitenden Formaten abgelegt sein kann, muß für die Backoffice-Systeme eine performante Datenhaltung sichergestellt sein. Es erscheint daher sinnvoll, die Anforderungen an dieses Systeme getrennt zu analysieren.

Tarifrechner

Der Kern der Laufzeitsysteme ist ein Tarifrechner, der in der Lage ist, die Funktionen des Produktbaumes auszuwerten. Dieser Tarifrechner ermöglicht es, auf allen Plattfor-

men die gleichen Berechnungen anzustellen. Wegen der zum Teil wichtigen Rundungen ist es dabei sinnvoll, diesen Tarifrechner codeidentisch auf allen Plattformen einzusetzen[3]. Im Kern ist der Tarifrechner eine einfache Stackmaschine [ASU86]. Die Schnittstelle beschränkt sich auf wenige Funktionen, wie z.B.:

- `sessionHnd startSession ()`
- `void setVar (sessionHnd, id, wert)`
- `wert compute (sessionHnd, byteCode)`
- `void closeSession (sessionHnd)`

Ein solcher Tarifrechner wird den Anforderungen an den Tagesbetrieb gerecht. Eine Laufzeit von ca. 5µs pro Befehl ist bereits ohne besonderes Tuning erreichbar und mit entsprechendem Aufwand nochmals deutlich zu reduzieren. Die Schnittstelle ist sehr klein und damit gut zu bedienen. Lediglich der Betrieb innerhalb von Transaktionssystemen stellt erhöhte Anforderungen an die Realisierung.

Meta-Runtime

Während zur Unterstützung von Angebotssystemen der Tarifrechner im wesentlichen ausreicht, muß für die Backoffice-Systeme erheblich höherer Aufwand getrieben werden. Verträge zu den neuen Produkte müssen ja ohne Änderung des physikalischen Datenmodells in der Bestandsdatenbank gespeichert und verarbeitet werden. Dafür muß das Datenmodell flexibel genug sein, um die verschiedenen Datenstrukturen aufnehmen zu können. Eine bewährte Technik dazu ist die sogenannte „Vertikalisierung“ der Daten (siehe Abbildung 5).

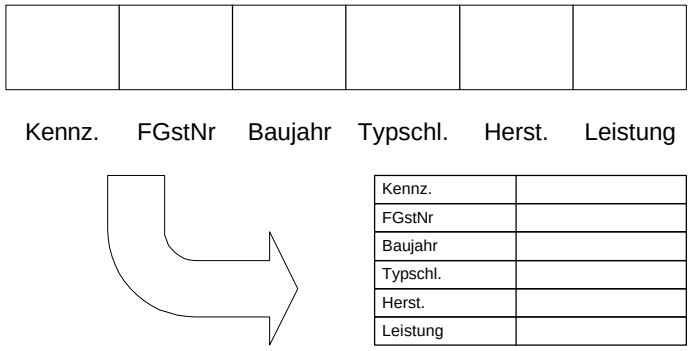

Abbildung 5: Bei der Vertikalisierung von Daten werden die einzelnen Attribute eines Vertrages in eigenen Zeilen gespeichert.

[3] Leider reicht das auch nicht, um gleiches Verhalten sicher zu stellen. Das liegt daran, daß sich die meisten Fließkommaprozessoren auf Serverhardware an die IEEE Normen zur Fließkommabearbeitung halten, während auf Mainframes manchmal auch andere Verfahren zum Einsatz kommen. Das kann zu Unterschieden auf dem letzten Bit führen, die bei entsprechend instabilen numerischen Verfahren das Endergebnis verfälschen können. Lösen läßt sich dieses Dilemma nur mit Dezimalarithmetik, die jedoch entsprechend langsam ist.

Allerdings steigt durch solche Bestandssysteme die Datenbank- und Prozessorlast der beteiligten Systeme. Zwischen 50 und mehrere 100 Datenbankzugriffe pro Vertragstransaktion stellen eine deutliche Herausforderung an die Administration der Systeme dar. Um überhaupt annehmbare Ergebnisse zu erhalten, ist es notwendig, die Funktionen nicht als Bäume in der Datenbank zu halten, sondern auch der Meta-Runtime den oben beschriebenen Tarifrechner zugrunde zu legen. Entsprechend intelligente Optimierung beim Export der Produktdaten können die Last auch nochmals senken.

Alles in allem stellen Produktsysteme, die den vollen Lebenszyklus eines Versicherungsproduktes unterstützen, noch immer eine erhebliche Herausforderung an Entwicklung und Betrieb dar. Die damit verbundenen Kosten dürften aber durch den strategischen Vorteil wett gemacht werden, der aus diesen Systemen erwächst.

Danksagung

Mein Dank gilt Wolfgang Keller für seine Unterstützung sowie den Teams der Projekte, die zu diesem Papier beigetragen haben: Ruth Leuzinger, Rudolf Donko, Michael Augustin, Stefan Blüml, Alwine Brem, Elke Knall, Peter Krömer, Andrea Meier, Michael Niebler, Robert Sigl, Heinz Weber, Rudolf Lewandowski, Klaus Kudjakov, Gernot Schretzlmeier, Marie-Luise Roters, Ralph Johnson, Alistair Cockburn, Jutta Eckstein und Alexander Rudyj.

Referenzen

[APi97] David M. Anderson, B. Joseph Pine: *Agile Product Development for Mass Customization - How to Develop and Deliver Products for Mass Customization, Niche Markets, JIT, Build-to-Order, and Flexible Manufacturing;* Irwin Professional Publishing, 1997; ISBN 0-78631175-4.

[ASU86] Alfred Aho, Ravi Sethi, Jeffrey Ullman*: Compilers – Principles, Techniques, and Tools;* Addison-Wesley, Reading, Massachusetts, 1986; ISBN 0-201-10088-6

[Col98] Jens Coldewey: *Choosing Database Technology;* Vortrag auf der Object Expo 98, Frankfurt; erhältlich über http://www.coldewey.com/publikationen

[GHJ+95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Design Patterns - Elements of Reusable Object-Oriented Software;* Addison-Wesley, Reading, Massachusetts, 1995; ISBN 0-201-63361-2

[Kel98] Wolfgang Keller: *Some Patterns for Insurance Systems;* in: *Proceedings of the 6th Conference on Pattern Languages and Programming;* erhältlich über http://www.objectarchitects.de

[Por96] Michael E. Porter: *Wettbewerbsvorteile - Spitzenleistungen erreichen und behaupten;* Campus Verlag GmbH, Frankfurt/Main, 1996; ISBN 3-593-34144-1

[SLe96] Paul Schönsleben, Ruth Leuzinger: *Innovative Gestaltung von Versicherungsprodukten - Flexible Industriekonzepte in der Assekuranz;* Gabler Verlag, Wiesbaden, 1996; ISBN 3-409-19470-3