HANSER


# Leseprobe

zu

## „Effektive Softwarearchitekturen“ (8. Auflage)

von Gernot Starke

ISBN (Buch): 978-3-446-45207-7
ISBN (E-Book): 978-3-446-45420-0

Weitere Informationen und Bestellungen unter
http://www.hanser-fachbuch.de/9783446452077
sowie im Buchhandel

# Inhalt

# Vorwort

*Haben Sie jemals einen dummen Fehler zweimal begangen?*
*– Willkommen in der realen Welt.*
*Haben Sie diesen Fehler hundertmal hintereinander gemacht?*
*– Willkommen in der Software-Entwicklung.*

Tom DeMarco,
in: „Warum ist Software so teuer?“

Wenn Sie sich für Baukunst interessieren, dann erkennen Sie sicherlich die „Handschrift“ berühmter Architekten wie Frank Lloyd Wright, Le Corbusier oder Mies van der Rohe immer wieder, egal, wo auf der Welt Sie auf Bauwerke dieser Meister stoßen. Die Funktionalität des Guggenheim Museums in New York oder des Opernhauses in Sydney gepaart mit deren Schönheit und Ästhetik sind unvergessliche Eindrücke. Das erwarten wir heute auch von unseren IT-Systemen: Funktionalität gepaart mit Stil!

Seit mehr als zwanzig Jahren versuche ich, Systementwicklern die Kunst des Architekturentwurfs nahezubringen. Die Erfahrung hat mich gelehrt, dass man jede Person, die mit gesundem Menschenverstand ausgestattet ist, zu einem guten Systemanalytiker ausbilden kann. Softwarearchitekten auszubilden, ist wesentlich schwieriger.

Früher waren viele unserer Systeme so einfach, dass einzelne Programmierer die Struktur leicht im Kopf behalten konnten. Heutzutage gehört mehr dazu, um die Struktur eines Systems zu beherrschen, die Auswirkungen von Technologieentscheidungen vorauszusehen und die Vielzahl von Hilfsmitteln wie Generatoren, Frameworks, Libraries und Entwicklungswerkzeuge kosteneffizient und zielführend einzusetzen.

Viele Jahre war ich davon überzeugt, dass nur Erfahrung in der Erstellung großer Systeme und selbst gemachte Fehler gute Architekten hervorbringen. Wir wussten einfach zu wenig über Wirkungen und Folgewirkungen von Designentscheidungen. In den letzten Jahren ist die Entwicklung von Architekturen mehr und mehr zur Ingenieursdisziplin herangereift.

Gernot Starke ist es gelungen, die Essenz dieser Disziplin auf den Punkt zu bringen. Die Tipps und Tricks, die er in diesem Buch zusammengetragen hat, vermitteln Ihnen eine Fülle von Praxiserfahrungen. Selbst wenn Sie zu den Veteranen der Branche gehören, werden Sie neben vielen Déjà-vu-Erlebnissen sicherlich noch die eine oder andere Perle entdecken. Wenn Sie gerade Ihre ersten Sporen als Architekt(in) verdienen, dann können Sie sich mit den Empfehlungen den einen oder anderen Holzweg ersparen.

Trotz aller Fortschritte in der IT bleiben Konstruktion und Ausgestaltung von Architekturen dauerhaft eine Domäne für kreative Gestaltungsarbeit von Menschen und Teams. Softwarearchitekt ist daher ein Beruf mit sicherer Zukunft!

Aachen, im September 2013

*Peter Hruschka*

# Vorwort zur siebten Auflage (2015)

*Softwarearchitektur: Das ist die Königsdisziplin des Software-Engineering.*

Prof. Ernst Denert
in [Siedersleben04]

Motiviert durch die Herausforderungen realer Projekte habe ich in der siebten Auflage dieses Buchs große Teile gründlich renoviert: Die Entwurfsmethodik und -prinzipien (Kapitel 4) sind komplett neu strukturiert und überarbeitet und ein ganz neues Kapitel über Modernisierung und Evolution ist (endlich) dazugekommen.

Die Architektursichten als Werkzeug der Kommunikation und Dokumentation orientiere ich an den von Peter Hruschka und mir gestalteten (und unter *www.arc42.de* frei verfügbaren) Vorschlägen und Begriffen.

Meiner Meinung nach ergänzen sich Agilität und Architektur wunderbar, darum habe ich viele Teile um Hinweise zur Architektur in agilen Projekten ergänzt.

Diese Auflage enthält eine Zusammenfassung des aktuellen iSAQB-Lehrplans zum *Certified Professional for Software Architecture* (CPSA-F), mit Hinweisen zur gezielten Prüfungsvorbereitung.

# Vorwort zur achten Auflage

Seit der ersten Auflage 2002 sind 15 Jahre vergangen – in denen sich die Disziplin „Softwarearchitektur" zum Glück erheblich weiterentwickelt hat. Unsere Branche hat viel gelernt, Agilität ist endlich im Mainstream angekommen, schnelles Feedback, beispielsweise durch automatisiertes Testen, Normalität statt Ausnahme.

In der achten Auflage habe ich an fast allen Teilen des Buchs korrigiert und poliert. Die Aktualisierungen des iSAQB-CPSA-F-Lehrplans sind komplett eingeflossen und das Kapitel über Microservices ist von den Altlasten der SOA befreit ☺.

Zusätzlich bekommt das Buch jetzt eine komplett neue Website, in *responsive design*, die mittels Github[1], Jekyll und Docker entwickelt wird (*http://esabuch.de*).

Möge dieses Buch helfen, bessere Software zu bauen!

Köln, August 2017

*Gernot Starke*

[1] Für Insider oder falls Sie Verbesserungsvorschläge haben: *https://github.com/gernotstarke/esabuch.de-site*

# 12.1 Beispiel: Datenmigration im Finanzwesen

## 1 Einführung und Ziele

### Zweck des Systems

Dieses Dokument beschreibt die Softwarearchitektur des M&M[1]-Systems zur Migration von ca. 20 Millionen Personen- und Kontodaten der Firma Fies Teuer AG, einer Organisation aus der Finanzdienstleistung.[2]

Der Auftraggeber von M&M betreibt seit etwa 1970 einige Mainframe-Anwendungen (Cobol, VSAM) zur Pflege von Personen-, Konto- und Bankdaten. Diese Systeme werden zurzeit durch eine homogene Java-Anwendung abgelöst, die durch ein anderes, parallel zu M&M laufendes Projekt entwickelt wird.

Fies und Teuer AG hat in Deutschland ca. 20 Millionen Kunden und pflegt für diese Kunden insgesamt mehr als 50 Millionen Konten. Kunden können natürliche oder juristische Personen sein, teilweise auch andere Organisationen (Verbände, Vereine etc.).

Konten enthalten neben reinen Buchungsinformationen auch statistische oder sonstige finanzbezogene Informationen zu den Kunden oder mit ihnen assoziierten Personen (z. B. Ehepartner) oder Organisationen (z. B. von ihnen geführte Unternehmen).

Die inhaltliche Bedeutung dieser Konto- und Buchungsinformationen sowie die darin enthaltenen Attribute haben über die Betriebszeit der bisherigen Anwendung stark gewechselt. Zur Umstellung auf das neue konsolidierte Objektmodell hat ein Team von ca. 20 Fachexperten eine Menge von mehreren Hundert fachlichen Regeln aufgestellt, nach denen die Migration ausgeführt werden muss.

Sämtliche bestehenden Daten aus dem bisherigen Format (VSAM-Dateien, EBCDIC-Codierung) müssen in dieses Java-Objektmodell migriert werden. Diese Migration ist Aufgabe des hier beschriebenen Systems.

### Ausgangssituation der bestehenden Daten

- Früher waren die Daten anhand der „Konten“ organisiert.
  - Für jegliche Operationen mussten Sachbearbeiter zuerst die betroffenen Konten identifizieren, erst danach konnten fachliche Operationen (Buchungen, Auskünfte) für die betroffenen Personen ausgeführt werden.
  - Im Zuge der steigenden Kundenorientierung der Fies und Teuer AG sollen die Daten künftig anhand von Personen organisiert werden.
- Die Ausgangsdaten liegen in Form verschiedener Dateien auf Bändern vor, deren Einträge nach fachlichen Kriterien einander zugeordnet werden müssen.

---

[1] M & M steht für **M**igration von **M**assendaten.

[2] Wie Sie leicht erraten können, habe ich den Namen dieser Organisation bewusst verändert ...

- Früher geltende Schlüssel- und Kennzahlensysteme müssen in neue Schlüssel überführt werden. Beispiel: Die alte Tarifklasse „T13“ wird zur neuen Tarifklasse „Jährliche Zahlung ohne Skonto“.

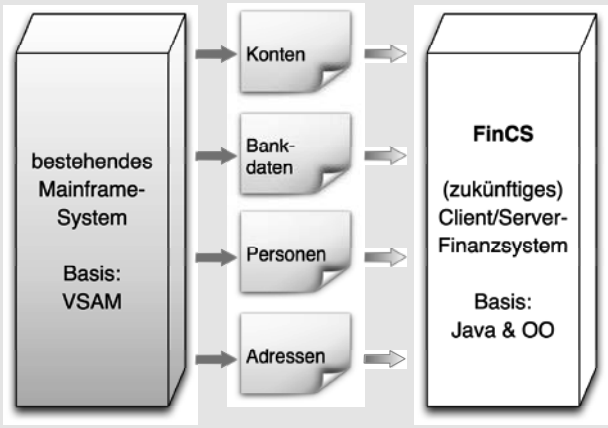


**Bild 12.1**
Zweck des Systems: Migration bestehender Daten

**Leserkreis**

- Alle in Sektion 1.3 dieser Dokumentation genannten Stakeholder von M&M.
- Softwareentwickler, Architekten und technische Projektleiter, die Beispiele für eine Architekturdokumentation (auf Basis des arc42-Templates) suchen.
- Mitarbeiter von IT-Projekten, die sich das Leben in der Softwarearchitektur vereinfachen, indem sie auf ein bewährtes Template zur Dokumentation zurückgreifen.

## 1.1 Fachliche Aufgabenstellung

Die wichtigsten funktionalen Anforderungen:

- Bestehende Kunden-, Konto- und Adressdaten sollen von den bisherigen VSAM[3]-basierten Mainframe[4]-Programmen aus dem EBCDIC-Format in das Objektmodell von FINCS, einer in Entwicklung befindlichen Java-Anwendung, migriert werden.
- Die Ausgangsdaten liegen in Form verschiedener Dateien oder „Bänder“ vor, deren Einträge nach fachlichen Kriterien einander zugeordnet werden müssen.
- Früher waren die Daten nach *Konten* organisiert, im neuen System sind *Personen* der Bezugspunkt.
- Teile der früheren Schlüssel- oder Kennzahlensysteme sollen in neue Schlüssel überführt werden.

---

[3] VSAM = Virtual Storage Access Method; eine Zugriffsmethode für von IBM-Großrechnern verwendete Dateien. VSAM-Dateien bestehen aus einem Metadatenkatalog sowie mindestens einer physischen Datei. Mehr zu VSAM in den Literaturhinweisen am Ende dieses Kapitels. Übrigens gibt es das Gerücht „VSAM ist grausam“.

[4] Eine aktuelle Einführung in OS/390 finden Sie unter *http://www.informatik.uni-leipzig.de/cs/esvorles/index.html*

## 1.2 Qualitätsziele

Die primären Qualitätsziele von M&M lauten:

- Effizienz (Performance): Migration von ca. 20 Millionen Personen- und Kontodaten innerhalb von maximal 24 Stunden.
- Korrektheit: Die Migration muss revisionssicher und juristisch einwandfrei erfolgen. Hierzu sind geeignete Maßnahmen zur Fehlervermeidung und -erkennung nötig.

### Nicht-Ziele

Was M&M nicht leisten soll:

- Änderbarkeit oder Flexibilität der fachlichen Transformationsregeln – die Migration ist einmalig.
- Es bestehen keine besonderen Anforderungen an Sicherheit – M&M wird nur ein einziges Mal produktiv betrieben, und das innerhalb eines gesicherten Rechenzentrums.

## 1.3 Stakeholder

| Rolle | Beschreibung | Ziel |
|---|---|---|
| Management der Firma Fies und Teuer AG | Projektmanagement, Lenkungsgremium | Reibungslose und fehlerfreie Migration.<br>Keine negative Presse. |
| Endkunden der Firma | 20 Millionen Kunden. Treten im Projektverlauf nicht in Erscheinung, sind nur indirekt beteiligt. | Korrekte Migration ihrer Daten. |
| Revision oder Buchprüfung | Interne Revisionsabteilung, die (inhaltliche und juristische) Korrektheit der Migration durch Stichproben validieren wird. | Einfacher Zugang zu Stichproben.<br>Korrekte Migration. |
| Projekt FINCS | Das Entwicklungs- und Architekturteam des künftigen Client/Server Finanzsystems. | Flexibilität beim geplanten Objekt- und Datenmodell. |
| Boulevard-presse | Legt besonderes Augenmerk auf Firma Fies und Teuer AG. Fehlverhalten wird gnadenlos publiziert. | Migrationsfehler, „Whistleblower“ im Projekt finden. |

# 2 Einflussfaktoren und Randbedingungen

## 2.1 Technische Einflussfaktoren und Randbedingungen

| Randbedingung | Erläuterung |
|---|---|
| Hardware-infrastruktur | IBM Mainframe als Plattform des Altsystems, Sun Solaris Cluster für das Zielsystem |
| Software-infrastruktur | ▪ Sun Solaris als Betriebssystem der Zielumgebung<br>▪ Oracle als neue Datenbank<br>▪ JEE-kompatibler Applikationsserver als mögliche Betriebsumgebung |
| Ausgangsdaten in EBCDIC | Ausgangsdaten liegen in EBCDIC-Kodierung auf vier getrennten Bändern vor. Eine Vorsortierung durch die Fies und Teuer AG ist nicht möglich. |
| Systembetrieb | Batch |
| Grafische Oberfläche | keine, Bedienung kann per Konsole erfolgen |
| Programmier-sprachen | Aufgrund des vorhandenen Know-hows des Entwicklungsteams soll in Java programmiert werden. |
| Analyse- und Ent-wurfsmethoden | objektorientiertes Vorgehen |
| Datenstrukturen | Objektmodell der Zielumgebung bekannt (Bestandteil eines anderen Projektes) |

## 2.2 Organisatorische Einflussfaktoren

Die Fies und Teuer AG ist als juristisch penibler Auftraggeber bekannt. Das Management versteht rein gar nichts von IT (und gerüchteweise auch nichts von Geld, aber das ist eine andere Geschichte).

Die verworrenen Eigentumsverhältnisse der Fies und Teuer AG lassen komplizierte Entscheidungsstrukturen befürchten.

Fies und Teuer AG hat langfristige Lieferverträge mit diversen IT-Dienstleistern abgeschlossen, die eine marktorientierte und freie Auswahl eventuell benötigter externer Mitarbeiter methodisch verhindern.

Eine hochgradig effektive externe Qualitätssicherung fordert eine umfangreiche Dokumentation (und prüft diese sogar auf inhaltliche Korrektheit – in der IT-Branche eher unüblich, doch ist das ebenfalls eine andere Geschichte).

# 3 Kontextabgrenzung

### Fachlicher Kontext

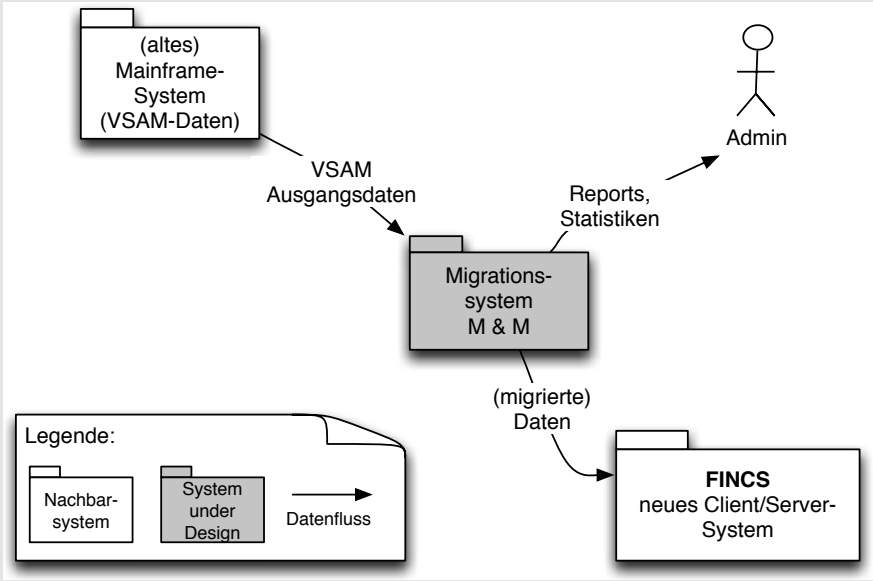


**Bild 12.2** Fachlicher Kontext

### Kurzbeschreibung der externen Schnittstellen

| Schnittstelle | Beschreibung | Technologie |
|---|---|---|
| VSAM-Daten | Vier Arten von Daten (Personendaten, Adressdaten, Bankdaten, Kontodaten) | Bänder, LTO-1, siehe Sektion 3.2.1 |
| Migrierte Daten | Gemäß des vom FINCS-Projekt erstellten Objektmodells | Übergabe des Objektgraphen über eine Session-Bean |
| Reports, Statistiken | Während der Migration: Kontinuierliche Ausgabe der Anzahl migrierter Personen. Nach der Migration: Ausgabe der Anzahl von Fehlersätzen | Konsole |

### Technischer oder Verteilungskontext

Die Migration wird auf zwei getrennten Servern ablaufen: ein Migrationsserver zur eigentlichen Datenmigration und Ausführung der technischen und fachlichen Transformationen, ein zweiter als Datenbankserver.

Das neue System (FINCS) wird direkt auf die von der Migration erstellte Datenbank zugreifen. Der Migrationsserver muss daher das künftige Klassen- und Tabellenmodell kennen.

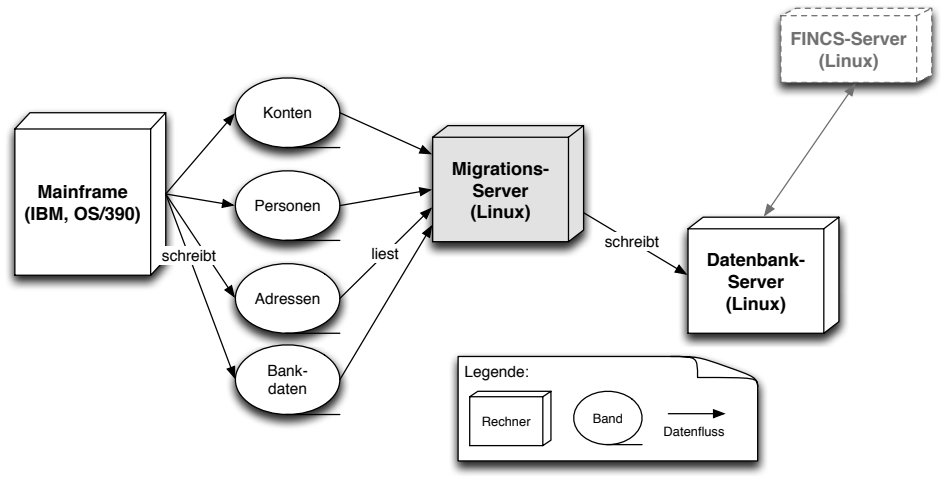


**Bild 12.3** Technischer-/Verteilungskontext

### Übersicht der Eingangsdaten(-schnittstellen)

Sämtliche Eingangsdaten werden auf jeweils einzelnen LTO-1-Bändern pro Datenart geliefert. Jedes dieser Bänder enthält bis zu 100 Gigabyte an Daten – die genaue Datenmenge kann aufgrund technischer Beschränkungen der Fies und Teuer AG vorab nicht spezifiziert werden.

| Eingangsdaten | Beschreibung |
|---|---|
| Kontodaten | Trotz der Bezeichnung enthält dieses Band die Buchungs-/Bewegungsdaten für sämtliche Konten.<br>Variables Satzformat mit 5–25 Feldern, gemäß FT-Satzart 43. |
| Personendaten | Stammdaten der von der Fies und Teuer AG betreuten Personen und Organisationen (Unternehmen, Vereine, Stiftungen, Behörden).<br>Variables Satzformat mit 15–50 Feldern, gemäß FT-Satzart 27. |
| Adressdaten | Adress- und Anschriftsdaten, inklusive Informationen über Empfangsberechtigte, Orts- und Straßenangaben, Angaben über Haupt- und Nebenadressen (Zentralen, Filialen) für Organisationen und Unternehmen.<br>Variables Satzformat mit 5–40 Feldern, gemäß FT-Satzart 33.<br>Mehrere Sätze pro Person möglich. |
| Bankdaten | Daten externer Banken (Referenz- und Gegenkonten der Personen und Organisationen).<br>Festes Satzformat, aber mehrere Sätze pro Person möglich. |

## 4 Lösungsstrategie

Migration im Batch mit folgenden Kernkonzepten:

- Datenflussarchitekturstil mit Pipe-und-Filter-Elementen, mit relationaler Datenbank als Pipe (Qualitätsziel: Performance).
- Parallelisierung der rechenintensiven fachlichen Migration (Qualitätsziel: Performance).
- Fehlersensoren in sämtlichen Verarbeitungsschritten, Sammlung sämtlicher erkannter Fehler in der Fehlerdatenbank (Qualitätsziel: Korrektheit).

## 5 Bausteinsicht

Ausgehend von einer Menge von VSAM-Dateien (geliefert als Bänder), konvertiert („migriert“) die M&M-Anwendung sämtliche Datensätze in das neue Objektmodell.

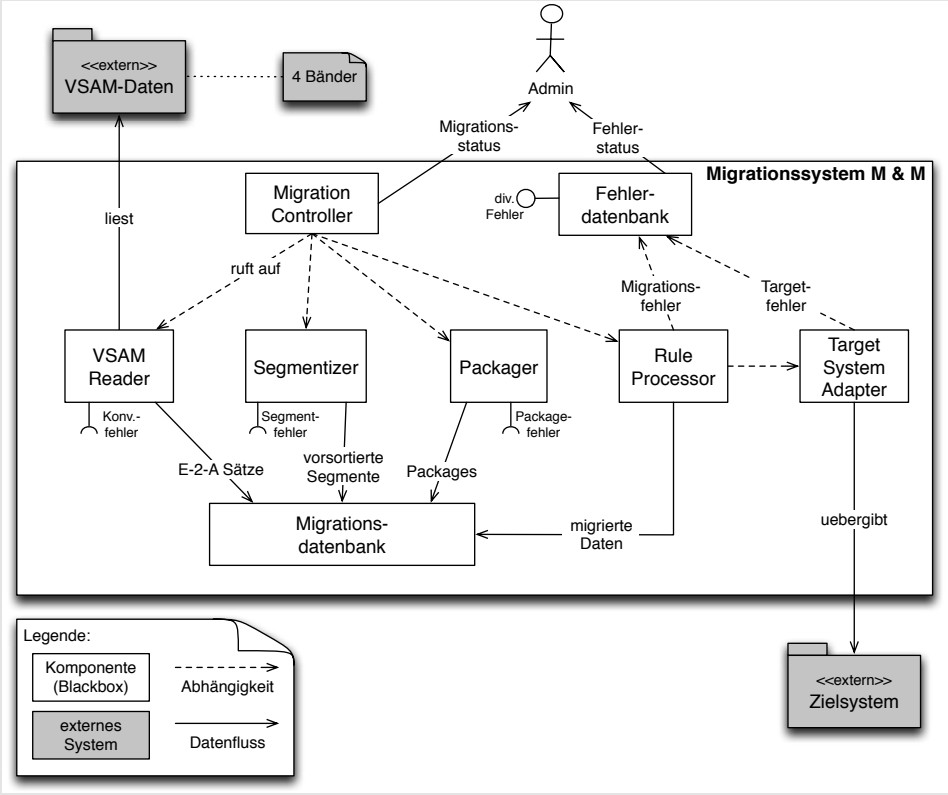


**Bild 12.4** Whitebox-Darstellung des Migrationssystems, Level-1

## 5.1 M&M Bausteinsicht Level 1

**Grundlegende Strukturentscheidung: Datenfluss mit Pipes-und-Filters**

Zur Lösung des M&M-Migrationsproblems wenden wir ein modifiziertes Pipes-und-Filter-Architekturmuster an, ausführlich beschrieben im Buch „Pattern-Oriented Software Architecture“ von Buschmann et al. ([Buschmann+96]).

Ein wesentliches Entscheidungskriterium für diese Architektur war die Möglichkeit, Teile dieser Bausteine im Bedarfsfall (Performance-Bedarf) zu parallelisieren.

Entgegen dem klassischen Pipe-und-Filter-Muster arbeitet der Eingangsbaustein (VSAM Reader) seine Eingangsdaten erst vollständig ab, bevor der nächste Filter aufgerufen wird.

| Baustein | Bedeutung |
|---|---|
| Migrations-Controller | Koordiniert die übrigen Bausteine und meldet den aktuellen Migrationsstatus an den Admin. Es handelt sich um eine einzelne Java-Klasse mit ausführlicher Ausnahme- und Fehlerbehandlung sowie einer Schnittstelle für Reporting und Auswertungen. Keine weitere Doku. |
| VSAM-Reader | Führt Formatkonvertierung von EBCDIC nach ASCII/UNICODE durch. Details siehe Sektion 5.1.1. |
| Segmentizer | Vorbereitung der Parallelisierung des „Rule Processors“: Zuordnung der einzelnen Datensätze der unterschiedlichen Datenquellen (Personen-, Konto-, Bank- und Adressdaten) zu zusammengehörigen Datengruppen oder Segmenten. Elemente eines Segments können unabhängig von Inhalten anderer Segmente bearbeitet werden. Notwendig ist die Aufteilung der Eingangsdaten auf mindestens drei bis fünf verschiedene Segmente. Es geht um eine möglichst *schnelle* Einteilung in Segmente, nicht um Optimierung oder Balancierung der Segmente. Daher liegt das Hauptaugenmerk auf Performance. Keine weitere Doku. |
| Rule-Processor | Erzeugt aus den formatkonvertierten Alt-Daten einen Objektgraphen gemäß den vom Neusystem FINCS erwarteten Klassenstrukturen. Siehe Sektion 5.1.2. |
| Migrations-datenbank | Speichert die konvertierten und segmentierten Migrationsdaten zwischen. Dient im Sinne des Pipe&Filter-Architekturmusters als Pipe, d. h. enthält keinerlei eigene Logik. Die Migrationsdatenbank ist eine Menge von Tabellen einer relationalen Datenbank (Person, Konto, Bank, Adresse plus einige Schlüsseltabellen). |
| Target-System-Adapter | Anbindung an das Zielsystem FINCS. Siehe Sektion 5.1.3. |

### 5.1.1 VSAM Reader

- *Zweck/Verantwortlichkeit:* Konvertiert Datensätze aus VSAM-Format (EBCDIC-Codierung) in ein unter Unix direkt verarbeitbares ASCII- oder UNICODE-Format. Gleichzeitig werden die Bestandteile der VSAM-Daten (teilweise besitzen einzelne Bits- oder Bitgruppen in der EBCDIC-Darstellung besondere Bedeutung) in einzeln identifizierbare Datenelemente aufgelöst. Der VSAM Reader führt keinerlei fachliche Prüfungen durch, sondern lediglich Formatkonvertierungen.
- *Schnittstelle(n):* Eingabe von VSAM-Dateien. Für jede Dateiart (Personen, Kontodaten, Adressdaten, Bankdaten) existieren jeweils einzelne Reader-Komponenten, die den Satzaufbau ihrer jeweiligen Eingabedateien kennen.
- *Variabilität:* keine, weil das Gesamtsystem nur einmalig ablaufen soll
- *Offene Punkte:* Für eine spätere Nutzung bei anderen Unternehmen könnten die einzelnen Konverter „pluggable“ gemacht werden. Dies ist zurzeit nicht vorgesehen.

### 5.1.2 Rule Processor (und Packager)

Zweck/Verantwortlichkeit: erzeugt einen Objektgraphen mit der jeweiligen Person (natürliche oder nichtnatürliche Person) als Wurzelknoten sowie allen zugehörigen fachlichen Objekten als Zweigen (Adressen, Bankverbindungen, Konten, Buchungen).

Der Rule Processor führt damit den eigentlichen fachlichen Migrationsschritt aus der alten (datenartorientierten) Darstellung in die neue (personenorientierte) Darstellung durch. Eventuell sind hierzu mehrere Iterationen über die Regelbasis notwendig.

Beispiele (sind Bestandteil der Anforderungsspezifikation; hier zur Illustration des notwendigen Vorgehens erneut dargestellt):

- Bei der Migration einer verheirateten Frau müssen vorher der Ehepartner sowie gegebenenfalls sämtliche früheren (geschiedenen, verschwundenen oder gestorbenen) Ehepartner migriert werden.
- Bei der Migration von Minderjährigen müssen vorher die natürlichen Eltern, Vormunde und Pflegeeltern migriert werden.
- Bei der Migration von Firmen müssen vorher sämtliche Vorgängerunternehmen, aus denen die aktuelle Firma hervorgegangen ist, migriert werden. Gleiches gilt analog für Vereine oder Verbände.

Damit diese fachlich zusammengehörigen Daten oder Datengruppen durch parallel ablaufende Prozesse migriert werden können, müssen in einem Vorverarbeitungsschritt sämtliche zusammengehörigen Daten als „Paket“ zusammengestellt werden. Diese Aufgabe übernimmt der Packager.

### 5.1.3 Target-System Adapter

*Zweck/Verantwortlichkeit:* übernimmt den vom Rule Processor erzeugten Objektgraphen in das Objektmodell des (neuen) Zielsystems. Hierfür sind eventuell kleinere Modifikationen notwendig, je nach endgültig entschiedener Laufzeitumgebung auch nur die abschließende Persistierung.

Nach erfolgreicher Übernahme in das neue System (Zielsystem) liefert der Target-System Adapter eine Meldung zurück – im Fehlerfall wird der gesamte Objektgraph zusammen mit den Fehlermeldungen des Zielsystems in die Fehlerdatenbank geschrieben.

## 5.2 Bausteinsicht Level 2

Dieser Teil verfeinert die Blackbox-Bausteine von Level 1. Aufgrund des Beispielcharakters dieser Dokumentation sind nur wenige Bausteine detailliert dargestellt.

*Anmerkung:* Auch in einer echten Dokumentation können Sie sich auf die besonders wichtigen Komponenten konzentrieren. Geben Sie jedoch für sämtliche Komponenten eine Begründung, warum Sie auf eine detaillierte Darstellung verzichten.

### 5.2.1 VSAM-Reader Whitebox

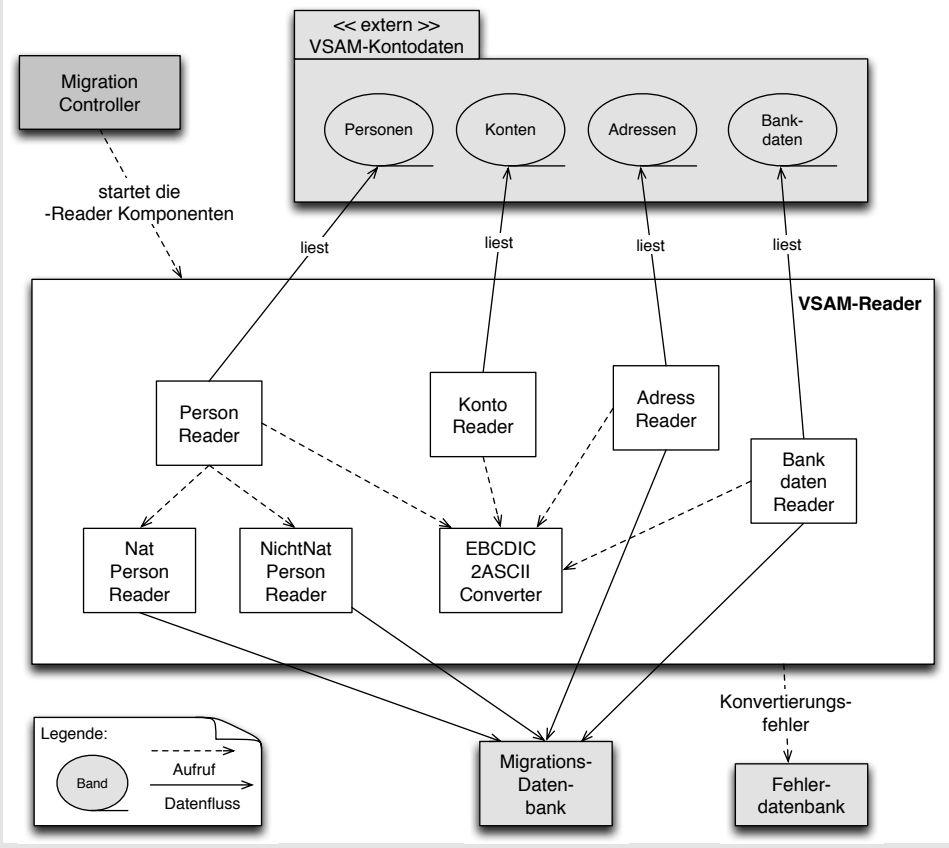


**Bild 12.5** Interner Aufbau des VSAM-Reader

| Baustein | Bedeutung |
|---|---|
| EBCDIC2ASCII Converter | Konvertiert eine Zeichenkette von EBCDIC nach ASCII. Als einziger Baustein des Systems in ANSI-C implementiert.<br>Behandelt nur Zeichenketten bis zur Länge von max. 256 Zeichen.<br>Siehe: *http://en.wikipedia.org/wiki/EBCDIC* |
| Reader-Komponenten | Die Reader-Komponenten lesen jeweils eine bestimmte Datenart aus „ihrem“ Eingabeband. Sie überführen sämtliche darin enthaltenen Daten in eigenständige Attribute.<br>Die einzelnen Reader kapseln die Syntax der „alten“ VSAM-Repräsentation sowie die technischen Details der Datencodierung.<br>Die einzelnen Reader schreiben ihre Ergebnisse in die Migrationsdatenbank, im Fehlerfall rufen sie die Fehlerdatenbank. |

### 5.2.2 Rule Processor Whitebox

Der Rule Processor führt die eigentliche fachliche Migration von Datensätzen durch. Er kombiniert die unterschiedlichen Kategorien von Eingangsdaten (Personen, Konten und Buchungen, Adressen, Bankverbindungen) miteinander und bearbeitet alle fachlichen Sonderfälle. Insbesondere besorgt der Rule Processor die benötigte Reorganisation der Daten, sodass künftig die Personen als Einstiegspunkte in die Navigation dienen, statt, wie früher, die Konten.

#### Strukturrelevante Entwurfsentscheidungen

- Kein kommerzieller Regelinterpreter: Nach einigen Versuchen mit kommerziellen Regelmaschinen entschied sich das Projektteam gegen den Einsatz einer solchen Komponente. Letztendlich war der Grund, dass die Umsetzung der fachlichen Regeln in eine Regelsprache keine wesentliche Vereinfachung gegenüber der Programmierung in Java gebracht hätte.

  *Anmerkung:* Aus heutiger Sicht (also posthum ...) würde ich diese Entscheidung nochmals überdenken und insbesondere Open-Source-Regelframeworks[5] auf ihre Tauglichkeit hin überprüfen.
- Keine explizite Regelsprache: Der Ansatz, die Regeln von den Fachexperten formulieren und nur durch einen Interpreter ausführen zu lassen, scheiterte an der Komplexität des gesamten Datenumfelds. Die effiziente und performante Navigation innerhalb des jeweils benötigten Objektraumes war mit einer Regelsprache nicht zu bewerkstelligen. Siehe dazu die Anmerkung zum Regelinterpreter.

[5] Wie beispielsweise JBoss-Drools.

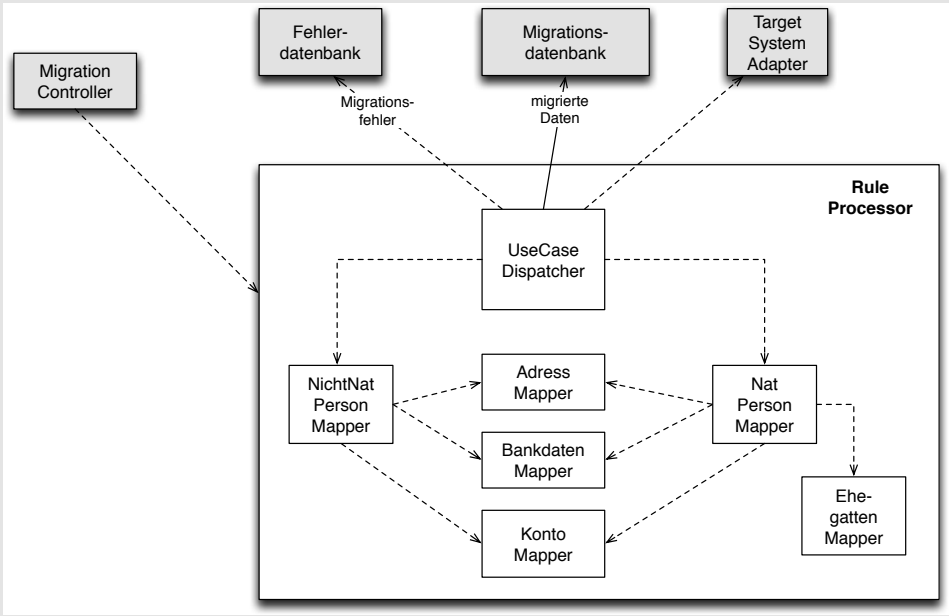


**Bild 12.6** Interner Aufbau des Rule Processor

| Baustein | Bedeutung |
|---|---|
| UseCase-Dispatcher | Ruft die eigentlichen fachlichen Migrationskomponenten (Mapper) auf. Siehe Sektion 5.2.2.1. |
| Person-Mapper (NichtNat sowie Nat) | Erzeugt aus den Eingabedaten eine natürliche (NatPerson) oder Nicht-Natürliche.* (NichtNat-)Person im neuen Objektmodell und füllt die Adressdaten, Bankdaten, Konto- und Buchungsdaten entsprechend auf. Handelt es sich um einen Verheirateten, wird die zugehörige Ehefrau (sowie eventuell Ehefrauen aus früheren Ehen) passend migriert.<br>Der Person Mapper benutzt die anderen Mapper. |

* Im Sprachgebrauch dieses Systems sind nicht-natürliche Personen sämtliche Organisationen, beispielsweise Firmen, Vereine, Verbände, Genossenschaften u. Ä.

*Anmerkung:* Aufgrund des Beispielcharakters dieser Dokumentation sind nachfolgend nur der UseCaseDispatcher sowie der PersonMapper ansatzweise beschrieben. In einer echten Dokumentation sollten Sie alle Bausteine so weit wie nötig detaillieren.

### 5.2.2.1 UseCase-Dispatcher

*Zweck/Verantwortlichkeit:* Der UseCase-Dispatcher entscheidet aufgrund der Datenkonstellation, welche der fachlichen Mapper-Komponenten angesprochen werden muss. Diese Entscheidung ist nicht trivial und nicht durch die Mapper selbst zu treffen. Teilweise müssen neben den aktuellen Daten weitere Sätze gelesen werden, um die Entscheidung abschließend treffen zu können.

*Beispiele:*

- Einpersonengesellschaften als nichtnatürliche Personen können teilweise durch den Personen-Mapper abgebildet werden, abhängig vom Gründungsdatum der Gesellschaft und dem Verlauf der Firmengeschichte (muss immer eine Einpersonengesellschaft gewesen sein und darf nicht aus einer Mehrpersonengesellschaft hervorgegangen sein).
- Natürliche Personen, die in der Vergangenheit Mitinhaber von Mehrpersonengesellschaften gewesen sind und darüber hinaus heute noch als Unternehmer tätig sind, müssen in Abhängigkeit von der Rechtsform der früheren Gesellschaft sowohl als natürliche als auch als nichtnatürliche Person geführt werden. Die Konto- und Buchungsinformationen sind entsprechend aufzuteilen.

<<Weitere Architekturbausteine werden in diesem Beispiel nicht dokumentiert>>

# 6 Laufzeitsicht

In der ersten Phase lesen die einzelnen Reader des VSAM-Readers alle angelieferten Dateien („Bänder“) und befüllen die Migrationsdatenbank. Danach teilt der Segmentizer die Daten in parallel bearbeitbare Segmente auf.

Der RuleProcessor lässt durch einen Packager (in Bild 12.7 durch den Aufruf „getPackage“ gekapselt) alle für die Migration einer Person oder eines Kontos notwendigen Daten aus der Migrationsdatenbank selektieren und führt anschließend die fachlichen Regeln auf diesem Datenpaket („Package“) aus.

Der Target System Adapter schreibt die migrierten Daten in die neue Datenbank.

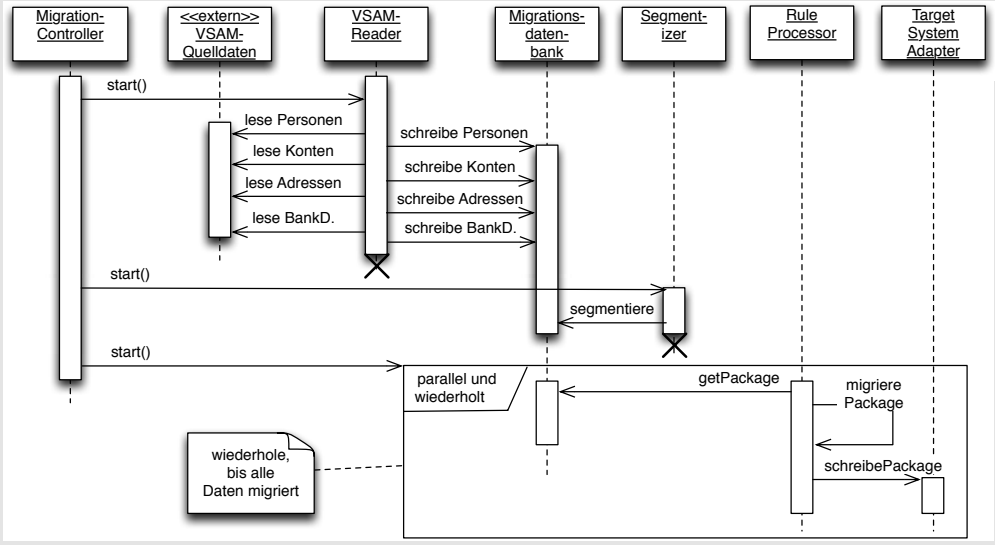


**Bild 12.7** Laufzeitsicht der M&M-Migration

# 7 Verteilungssicht

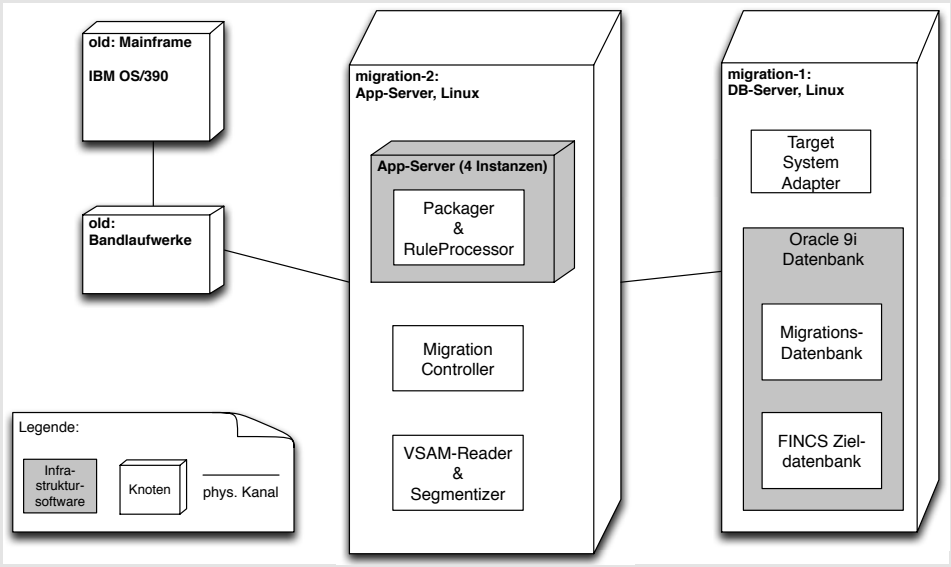


**Bild 12.8** Verteilungssicht: Migration auf zwei getrennten Servern

Die gesamte Migration findet auf einem eigenständigen Linux-Server („migration-2“) statt, auf dem in einem EJB-Container mehrere parallele Instanzen der eigentlichen Migrationskomponenten (Packer & RuleProcessor) ablaufen.

Über einen schnellen Bus ist dieser Server mit dem DB-Server („migration-1“) verbunden, auf dem der VSAM-Reader und der Segmentizer die initiale Befüllung der Migrationsdatenbank vornehmen.

*Anmerkung:* In einer echten Architekturdokumentation sollten Sie hier weitere Leistungsdaten der Hardware (Rechner, Netze, Bus-Systeme etc.) beschreiben.

# 8 Technische Konzepte

## 8.1 Persistenz

Die Datenspeicherung im neuen System übernimmt ein Java-Persistenz-Framework (Hibernate). Das Mapping der Objektstruktur auf die relationale Datenbank wird vom Framework komplett übernommen.

Für die eigentlichen Migrationskomponenten wird Persistenz damit nahezu transparent: Ist ein Objektgraph im Speicher aufgebaut, kann er auf einmal gespeichert werden.

**Persistenz der Zwischenformate und Fehlerfälle**

Die Eingangsdaten in sequenziellem VSAM-Format werden über SQL direkt in die Migrationsdatenbank geschrieben. Ebenso gelangen sämtliche Fehlersätze per SQL in die Fehlerdatenbank.

## 8.2 Ablaufsteuerung

Die Koordination und Ablaufsteuerung aller Migrationsprozesse wird durch einen zentralen Baustein (MigrationController) durchgeführt.

Die gesamte Verarbeitung erfolgt satzweise. Die Personendatei wird satzweise verarbeitet, sämtliche zugehörigen Kontoinformationen werden diesen Sätzen zugeordnet und die resultierenden Datenpakete anschließend gemeinsam migriert.

Aus Kostengründen hat der Auftraggeber eine kommerzielle Workflow- oder Prozess-Engine abgelehnt.

## 8.3 Ausnahme- und Fehlerbehandlung

Aufgrund der hohen Korrektheitsanforderungen der Auftraggeber werden sämtliche Ausnahmen beziehungsweise nicht migrierbaren Datensätze in einer Fehlertabelle gespeichert.

Sämtliche Fehlersätze müssen nach der Migration von Sachbearbeitern manuell migriert werden, wofür lediglich ca. 200 Personentage Aufwand zur Verfügung stehen. Jeden Tag kann ein Sachbearbeiter im Mittel 25 Personen migrieren, daher dürfen in der Fehlertabelle höchstens 5000 (= 200 * 25) Datensätze enthalten sein.

## 8.4 Transaktionsbehandlung

Die einzelnen Migrationsschritte speichern ihren Zustand jeweils satzweise ab. Dadurch ist zu jedem Zeitpunkt ein Wiederaufsetzen durch Neustart der gesamten Anwendung möglich.

## 8.5 Geschäftsregel und Validierung

Aus Kosten- und Lizenzgründen hat der Auftraggeber die Verwendung einer kommerziellen Regelmaschine (wie ILOG oder VisualRules o. Ä.) a priori ausgeschlossen.

Die Geschäftsregeln (hier: Migrationsregeln) wurden vollständig in Java codiert.

### 8.6 Kommunikation und Integration

Die Integration mit dem Quellsystem erfolgt über den Austausch von VSAM-Bändern.

Integration mit dem Zielsystem über Java Session-Beans und den Java Application Server. Der Objektgraph der migrierten Daten wird als POJO-Parameter übermittelt (*plain old java object*).

## 9 Entwurfsentscheidungen

| Entscheidung | Datum & Entscheider | Begründung, Konsequenz, Alternativen |
|---|---|---|
| Beschreibung der Migrationslogik in Java, NICHT in einer vereinfachten natürlichen Sprache | PL | Nach einigen Versuchen, unter anderem mit eigens formulierten Regelsprachen (mit Lex und Yacc) befand das Entwicklungsteam, dass eine Umsetzung fachlicher Regeln in eine solche Regelsprache keine wesentliche Vereinfachung gegenüber der Programmierung in Java gebracht hätte. |
| Kein kommerzieller Regelinterpreter | PL | Nach einigen Versuchen mit kommerziellen Regelmaschinen entschied sich das Projektteam gegen den Einsatz einer solchen Komponente.<br>Siehe Entscheidung zu Regelsprache.<br>*Anmerkung:* Aus heutiger Sicht (also posthum ...) würde ich diese Entscheidung überdenken und insbesondere Open-Source-Regelframeworks wie JBoss-Drools auf ihre Tauglichkeit hin überprüfen. |
| Pipes-und-Filter-Architektur | Architekt, Entwickler | Die Parallelisierbarkeit der eigentlichen Migrationsregeln (im RuleProcessor) wird hierdurch vereinfacht.<br>Es wird keinerlei GUI benötigt. |
| Dual-Server | Auftraggeber | Aufgrund der hohen Performance-Anforderungen werden Datenbank und Migrationslogik auf zwei getrennten Servermaschinen ausgeführt. Als DB-Server kommt die später vom neuen Kontosystem genutzte Maschine zum Einsatz. |
| Die Verarbeitung der fachlichen Regeln durch den Rule Processor (siehe Sektion 5.1.6 bzw. 5.2.2) wird durch mehrere parallele Prozesse (innerhalb des J2EE Application Servers) durchgeführt. | PL, Kunde | Anforderung seitens des Kunden. Eine rein sequenzielle Bearbeitung aller vorhandenen Personen wäre in der verfügbaren Laufzeit von 24 Stunden nicht möglich gewesen. |

## 10 Qualitätsszenarien

<Entfällt im Beispiel>

## 11 Risiken

<Entfällt im Beispiel>

## 12 Glossar und Referenzen

Das Glossar entfällt im Beispiel.

Mehr zu VSAM im frei verfügbaren IBM-Redbook:
*http://www.redbooks.ibm.com/abstracts/sg246105.html*

## Anmerkungen zum System

Einige Worte zur Klärung: Dieses Projekt hat wirklich stattgefunden, und das M&M-System ist wirklich entwickelt worden. Der Auftraggeber, die Fies und Teuer AG, hieß in Wirklichkeit natürlich anders, auch ist der fachliche Inhalt ein wenig vereinfacht worden.

Das echte Migrationsprojekt fand um 2002/2003 irgendwo in Deutschland statt. Über die Laufzeit von gut zwölf Monaten analysierten mehr als zehn Fachexperten die Logik und Geschäftsregeln der bestehenden Mainframe-Systeme und erstellten auf dieser Basis das fachliche Migrationskonzept. Hierbei galt es, die besonderen Anforderungen an Nachweispflichten zu berücksichtigen, ebenso die über die Zeit veränderten gesetzlichen Rahmenbedingungen. Verzahnt damit erstellte ein Team von mehr als zehn Entwicklern und Architekten die passende Software.

Einige der Stolpersteine und Herausforderungen des „echten“ M&M-Projektes:

- Die Datenqualität der Ausgangsdaten schwankte stark: So musste das Projekt die ursprünglich eingeführte Unterscheidung zwischen „Muss“ und „Optional“ für Datenfelder stark einschränken, weil 2–3 % der mehr als 20 Millionen Personendatensätze nicht alle (heute benötigten!) Muss-Attribute enthielten.
- Logische oder fachliche Attribute waren durch extreme Speicherplatzoptimierungen stark verklausuliert und immens schwer identifizierbar. Beispiel: „Wenn es sich um einen Fall des Typs 42 handelt, dann gibt Bit 4 (im EBCDIC-Format) des Datenrecords an, ob es sich um eine Hin- oder Rückbuchung handelt.“
- Die Logik der bestehenden Programme war teilweise älter als 30 Jahre – und für manche der Altprogramme gab es keine Ansprechpartner mehr.

# 13 Werkzeuge für Softwarearchitekten

*Wer als Werkzeug nur einen Hammer hat,*
*sieht in jedem Problem einen Nagel!*

Paul Watzlawik

**Fragen, die dieses Kapitel beantwortet:**

- Welche Kategorien von Werkzeugen können helfen?
- Nach welchen Kriterien könnte ich solche Werkzeuge auswählen?

Eine kleine Warnung vorweg: Sie werden in diesem Kapitel nichts über konkrete Werkzeuge lernen, keine Bedien- oder Einsatzhinweise finden, keine Tipps, wie Sie mit Tool X Ihr Problem Y lösen.

Kategorien von Werkzeugen

Vielmehr möchte ich versuchen, etwas Systematik in die Vielfalt der Werkzeuglandschaft zu bringen – und Ihnen das Spektrum aller möglicherweise hilfreichen Werkzeugkategorien vorstellen. Dazu erfahren Sie ein wenig über mögliche Auswahlkriterien.

## 13.1 Kategorien von Werkzeugen

### Werkzeuge zum Anforderungsmanagement

Anforderungsmanagement sollte über den gesamten Lebenszyklus von Systemen praktiziert werden – in Form von Lastenheften oder Backlogs, mit Use-Cases, User-Stories oder anderen Formen von funktionalen und Qualitätsanforderungen. Es geht hierbei um Erhebung, Erfassung, Analyse, Dokumentation und Management von Anforderungen.

Diese Werkzeuge sollten textuelle und/oder grafische Darstellung ermöglichen, Redundanzen reduzieren helfen sowie im Bedarfsfall die Rückverfolgbarkeit zwischen Anforderungen und Architektur/Code erlauben.

Herausforderungen sind insbesondere Mehrbenutzerfähigkeit, Zusammenführen/Merge von Modellen sowie die Unterstützung von Versions- und Konfigurationswerkzeugen.

### Modellierungswerkzeuge

Modellierungswerkzeuge können fachliche und technische Modelle von Software sowie Anforderungs- und Problemdomänen darstellen. Sie helfen, (meist grafische) Modelle zu erstellen und zu pflegen.

Diese Kategorie von Werkzeuge unterstützt ab der Ebene von Geschäftsprozessen bis hin zur Modellierung von Klassen, Detailabläufen oder Zustandsübergängen beliebig abstrakte oder detaillierte Darstellung statischer und dynamischer Abstraktionen.

Herausforderungen dieser Werkzeugkategorie sind Mehrbenutzerfähigkeit, Vergleich und Analyse von Modellen, Trennung von Modellobjekten und deren grafische Darstellung in Diagrammen, explizite Metamodelle sowie ein Kompromiss aus Standardkompatibilität (etwa zu UML) und Pragmatik (etwa: Freiform-Diagramme). Hilfreich können Reverse-Engineering-Fähigkeiten sein sowie die Generierung von Dokumenten aus Modellen.

### Werkzeuge zur Generierung von Code oder sonstigen Artefakten

Generierungswerkzeuge können ausgehend von abstrakten Beschreibungen (grafischen oder textbasierten Modellen in standardisierten oder domänenspezifischen Sprachen) beliebige Artefakte generieren. Primär finden sie Anwendung zur Erzeugung von Quellcode, Datenbanktabellen oder -skripten, Konfigurationsdateien, Lexern oder Parsern, XML-Schemata oder in Form von Web-Service-Unterstützung. Interessant für Entwickler ist die Möglichkeit der programmatischen Nutzung (über Programmierschnittstellen oder die Integration in Build-Prozesse).

Herausforderung dieser Werkzeuge ist die häufig geringe Akzeptanz bei Entwicklern, die aufgrund des hohen Leistungsumfangs lernintensive Bedienung, die Flexibilität der Generierung beziehungsweise Transformationen sowie die Unterstützung frei definierbarer oder veränderbarer Metamodelle.

### Werkzeuge zur Generierung von Dokumentation

Als Spezialfall der oben genannten Generierungswerkzeuge können solche Werkzeuge beispielsweise aus Vorlagen im Plaintext (etwa: AsciiDoc oder Markdown) beliebige Artefakte generieren.

Nachfolgende Abbildung zeigt das Beispiel der docToolchain, die als Eingabeformat primär AsciiDoc verwendet, aber auch EA-Modelle, PowerPoint und Grafiken versteht, und daraus dann pdf, html und auch Confluence-Wikis erzeugen kann.

### Werkzeuge zum Codemanagement/Implementierungswerkzeuge

Diese Tools unterstützen Sie bei der Erstellung, Verarbeitung, Modifikation, Prüfung, Übersetzung und Ausführung von Quellcode. Primär zähle ich jede Art von Entwicklungsumgebung hierzu, aber auch (syntaxbewusste) Editoren, Refactoring-Tools, Debugger und alle Hilfsmittel rund um die Versionierung (siehe dazu auch das Thema Build- und Konfigurationsmanagement).

Die Unterstützung von Merging, Branching sowie garantiert verlustfreie Speicherung sämtlicher Versionen aller kontrollierten Artefakte für eine Vielzahl von Benutzern zählt zu den Grundforderungen. Für die Akzeptanz sind Verbreitung, Verständlichkeit sowie Anpassbarkeit an spezielle Aufgaben/Anforderungen hilfreich.

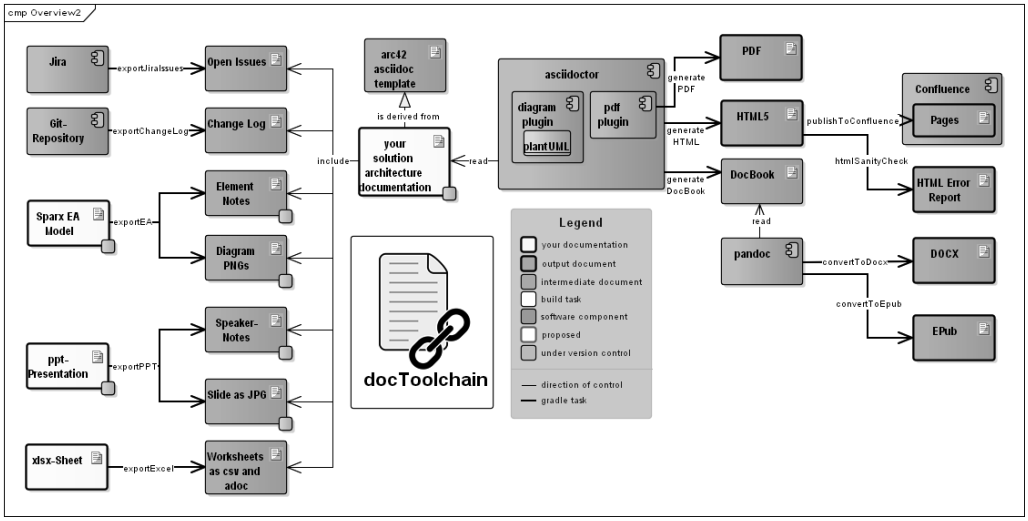


**Bild 13.1** Generierung von Dokumentation am Beispiel docToolchain[1]

### Werkzeuge zur statischen Codeanalyse

Diese Tools prüfen statische Eigenschaften von Quellcode – beispielsweise Abhängigkeiten (Kopplung), Komplexität, Kohäsion, Größe, Testabdeckung oder Einhaltung von Programmierkonventionen.

Eine große Herausforderung dieser Kategorie liegt im heute verbreiteten *polyglott programming*, d. h. der Unterstützung unterschiedlicher Programmiersprachen. Weiterhin hat die starke Verbreitung von Dependency-Injection Frameworks die direkte Analyse von Abhängigkeiten deutlich erschwert – weil Abhängigkeiten in solchen Frameworks dynamisch per Konfiguration aufgebaut werden können.

### Werkzeuge zur Laufzeitanalyse von Software

Laufzeit- oder dynamische Analyse betrachtet Softwaresysteme während ihrer Ausführung. Wesentliche Arten von Resultaten sind Laufzeit- und Geschwindigkeitsmessungen, Zeitmessungen bestimmter Systemteile (absolut oder relativ), Messung von Speicher- oder Ressourcennutzung, Kommunikationsverhalten sowie statistische Auswertungen.

Die große Herausforderung besteht darin, durch die Messungen das Laufzeit- oder Speicherverhalten des Systems möglichst wenig zu beeinflussen sowie die Ergebnisse zielgruppengerecht aufzubereiten. In verteilten Systemen gestaltet sich die Korrelation der auf unterschiedlichen Knoten gemessenen Resultate beliebig schwierig. ☹

### Werkzeuge zum Build- und Konfigurationsmanagement

Diese Werkzeuge erlauben die teilweise oder vollständige Automatisierung von Übersetzungs-, Paketierungs-, Prüf-, Test- oder Generierungsaufgaben von Software. Zu ihnen zählen

[1] docToolchain ist ein OpenSource-Projekt von Ralf D. Müller zur flexiblen Generierung von (Architektur-)Dokumentation, primär auf Basis von AsciiDoc. Siehe: *https://github.com/rdmueller/docToolchain*

Continuous-Integration-Systeme (die beispielsweise nach Commits im Versionsmanagement die benötigten Teile des Build-Prozesses gezielt anstoßen).

Schwierig für diese Werkzeuge ist der Kompromiss aus deklarativer und imperativer Darstellung von Build-Aufgaben, die Behandlung direkter oder transitiv abhängiger Artefakte und Bibliotheken sowie die Erreichung möglichst hoher Automatisierungsgrade.

Management-Aufgaben wie Inventarisierung sowie Wiederherstellung beliebiger Konfigurationen (inklusive Betriebssystem, benötigte Basissoftware und Datenbestände, Dokumentation sowie der eigentlichen Systeme) sind in manchen Fällen notwendig.

### Werkzeuge zum Test von Software

Insbesondere automatisierte Tests geben ein frühes Feedback zu Strukturen, Konzepten, Schnittstellen und Implementierung bereits während der Konstruktion und Entwicklung. Unit-, Integrations- und Akzeptanztests erleichtern oft die Kommunikation mit Testern und auch Fachabteilungen. Laufzeit-, Last- und Performance-Tests während der Entwicklung erlauben Prognosen über das Zeit- und Ressourcenverhalten des fertigen Systems. Penetrations- und Sicherheitstests adressieren die Sicherheit der Systeme und beteiligten Daten.

Zu den Herausforderungen zählen die Verwaltung von Testfällen, Testdaten sowie Testergebnissen, die Unterstützung für den Test verteilter Systeme, der Test grafischer Oberflächen sowie lange laufender fachlicher Abläufe. Es hat sich als gute Praxis herausgestellt, Fremdsysteme über sogenannte Mocks zu entkoppeln, die zu Testzwecken das Verhalten dieser Fremdsysteme simulieren.

Ich halte die testorientierte Entwicklung (in Form von Test-Driven oder Acceptance-Test-Driven Development) für ein sehr hilfreiches Entwurfsinstrument, das die Qualität detaillierter Entwurfs- und Implementierungsentscheidungen beträchtlich steigern kann.

Werkzeuge* wie Spockframework, Cucumber, Fitnesse oder andere sollten zur Standardausstattung praktizierender Softwarearchitekten gehören.

* Siehe etwa *http://spockframework.org*, *http://cukes.info*, *http://fitnesse.org* oder [Freeman+10].

### Werkzeuge zur Dokumentation

Werkzeuge zur Dokumentation sollen uns bei der langfristigen Kommunikation von Entscheidungen, Strukturen, Konzepten sowie sonstiger relevanter Sachverhalte unterstützen.

Zu dieser Kategorie gehören text- und grafikbasierte Werkzeuge zur Erstellung, Pflege oder Generierung von Dokumenten.

Für alle Dokumentationswerkzeuge auf der Basis binärer Formate stellen Teamfähigkeit (parallele Bearbeitung) und Zusammenführung (merge) Herausforderungen dar. Rein textbasierte Tools haben dafür Probleme mit formalen oder gestalterischen Vorgaben (z. B. Corporate Layout, Corporate Design).

## 13.2 Typische Auswahlkriterien

Im realen Leben müssen Sie selbst die Menge der für Sie und Ihre Stakeholder relevanten Auswahlkriterien für Werkzeuge ermitteln – aber ein paar typische Vertreter beziehungsweise Oberbegriffe möchte ich Ihnen dazu mitgeben.

### Technische Kriterien

Arbeitsergebnisse (Artefakte, Dokumente, Modelle) sollten in jedem Fall versioniert gespeichert werden können, d. h. fast alle Werkzeuge benötigen eine Integration mit Subversion, git und Co.

Da wir die meiste Software in Teams entwickeln, müssen oftmals mehrere Benutzer parallel mit Werkzeugen arbeiten – d. h. die sie sollten arbeitsteiliges Vorgehen unterstützen. Mehrere Bearbeiter sollten parallel agieren können, ohne sich gegenseitig zu behindern.

### Organisatorische Kriterien

Lizenzmodalitätenkosten spielen bei der Auswahl von Werkzeugen oftmals eine bedeutende Rolle. Beachten Sie, dass eine gesamthafte Kostenbetrachtung von Werkzeugen neben den Anschaffungskosten auch die Kosten der Einführung, Konfiguration sowie eventuelle regelmäßige Wartungs- und Betriebskosten berücksichtigen sollte.

Für manche Produkte benötigen Sie professionelle Unterstützung – dann sollten Sie die Verfügbarkeit dieser Unterstützungsleistung sowie deren Kosten ebenfalls in Ihre Entscheidungskriterien aufnehmen.

Weiterhin spielen gerade im Open-Source-Umfeld die Lizenzbedingungen von Produkten oder Frameworks eine Rolle: Klären Sie, welche Nutzungs- und Vertriebsrechte Ihnen diese Lizenzen einräumen. Manche sind in Dual-Lizenz-Modellen erstellt, bei denen die kommerzielle Nutzung anderen Einschränkungen unterworfen ist als die private Nutzung.

Hüten Sie sich vor juristischen Fallstricken – und ziehen Sie im Zweifel einen in IT- und Lizenzrecht versierten Experten zurate.

Schließlich möchte ich Akzeptanz (im Entwicklungsteam) als ein wesentliches Kriterium für die Auswahl von Werkzeugen nennen. Nicht jedes Werkzeug passt zu jedem Team – und ohne Akzeptanz sind manche Werkzeuge zum Scheitern verurteilt. Akzeptanz können Sie durch gezieltes Coaching und Begleitung steigern – aber nur in seltenen Fällen gegen die Überzeugung eines Teams erzwingen.

# Stichwortverzeichnis