Sanfte Migration

Architekturmanagement in der Generali-Gruppe

Wolfgang Keller
c/o Generali Office Service & Consulting AG, Kratochwjlestr. 4; A1220 Wien
EMail: wolfgang.keller@generali.at
http://www.objectarchitects.de/

1 Abstract

Java, Internet, Components und Reuse prägen heute stark die eher technologiegetriebene Architekturdiskussion für Informationssysteme. DB2, CICS und PL/I oder COBOL prägen die Realität. Dabei sind Versicherer und andere Finanzdienstleister gezwungen für das Internet neue Kanäle und Prozesse auf teils alter Software aufzusetzen. Fusionsprozesse und schneller organisatorischer Wandel diktieren den Takt der Informatik, die sich nie mehr zum Konsolidieren zurückziehen kann, sondern meist schon mit neuen Projekten und Anforderungen konfrontiert wird, noch bevor sie die letzten unter Kontrolle hat.

Dieser Beitrag zeigt, auf welche Dinge aus dem breiten High-Tech Angebot der Hersteller man bei seinem Architekturmanagement bauen kann. Er zeigt weiter, welchen Platz dabei Technologien einnehmen und wo die Bedeutung von Fachlichkeit und Anwendungs-Portfoliomanagement relativ dazu liegen. Es wird dabei demonstriert, dass zumindest High-Tech nicht die entscheidende Rolle spielt und dass man mit kontrolliert heterogenen Anwendungen teils wirtschaftlicher arbeiten kann, als mit rein homogenen Ansätzen. Dieser Artikel wird sogar zeigen, warum letztere heute bei einem großen Finanzdienstleister nicht mehr umzusetzen sind.

2 Einführung und Überblick

Software Architektur wurde und wird als einer der Schlüssel für erfolgreiche Software-Projekte angesehen. Dabei macht die "gängige Lehre" vor allem zwei Aussagen:

- eine Anwendungsarchitektur wird demnach benötigt, damit in einem größeren Projekt alle Entwickler an einem Strang ziehen und Systeme bauen, die in sich konsistent, wohl strukturiert und so gebaut sind, dass sie den festgelegten funktionalen und nichtfunktionalen Anforderungen gerecht werden.

- eine Anwendungsarchitektur auf der Unternehmensebene wird danach benötigt, wenn man einen möglichst hohen Reuse-Faktor erzielen möchte, indem man alle Systeme nach einer möglichst einheitlichen Architektur baut und so viele Querschnittskomponenten wiederverwenden kann.

Dieser Artikel wird eine neue Art von *Herausforderungen* vorstellen, die man mit einer bestimmten Art von Software-Architekturen gut bewältigen kann. Es geht dabei weniger um Reuse, als um den Umgang mit schnellem technischen Wandel und mit schnellen Veränderungsprozessen in Unternehmensstrukturen.

Dazu stellen wir zunächst die Herausforderungen dar, mit denen man heute vielfach konfrontiert wird, wenn man das Anwendungsportfolio eines großen, internationalen Finanzdienstleistungskonzernes managen muß. Dann werden die wichtigsten *Werkzeuge* vorgestellt mit denen man diese Herausforderungen angehen kann. Um es vorwegzunehmen: Technikzentrierte Anwendungsarchitekturen, mit denen man die obigen Aufgaben früher angegangen ist, werden dabei immer unwichtiger, weil ubiquitär und selbstverständlich.

Schließlich werden wir noch erläutern, welchen Nutzen *High-Tech-Ansätze* bei der Bewältigung schneller Veränderungen haben. Um es wieder vorwegzunehmen - gerade der Erfolg des Internets basiert auf Low-Tech Elementen.

Die *Zusammenfassung* betont noch einmal die wirkliche Essenz dessen, was in schnellen Veränderungsprozessen heute noch Bestand hat.

3 Herausforderungen

Die Herausforderungen an ein Architektur- und Anwendungsportfolio-Management für Software sind vor allem durch folgende Fragen gegeben:

Wie gestalte ich die Software-Anwendungslandschaft eines großen Finanzdienstleisters,

- so dass möglichst alle existenten Anforderungen erfüllt werden,
- neue Anforderungen schnell erfüllt werden können,
- man offen ist für neue Produkte und Vertriebswege,
- und in einer Unternehmensgruppe möglichst geringe Softwarekosten durch möglichst hohe Synergien hat.

3.1 Räumliche Verteilung

Dies kann schon an einem Standort mit 450 Softwareentwicklern herausfordernd zu managen sein. Im Falle heutiger größerer Konzerne kommt jedoch auch noch eine räumliche Verteilungsdimension hinzu, aus der sich zusätzliche Komplexität ergibt.

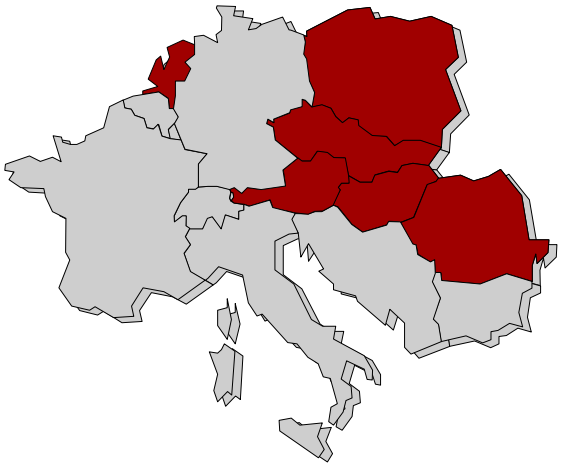

Abbildung 1: Räumliche Verteilung der einen Informatik der Generali Gruppe (Holding Vienna) - nicht zu verwechseln mit der weltweiten Generali Gruppe Triest.

Abbildung 1 zeigt die räumliche Verteilung der EINEn Informatik der Generali Vienna Holding in Europa: Der größte und umsatzstärkste Standort dieser Gruppe ist Österreich gefolgt von den Niederlanden (die nur was die Informatik anlangt assoziiert sind) und mit einem gewissen Abstand gefolgt von den Reformstaaten wie Ungarn, Tschechien, Slowakei, Polen, Rumänien, Bulgarien, Kroatien und Slowenien die ebenfalls mit eigenen Landesgesellschaften und teilweise eigener EDV zur Gruppe gehören. Dabei hat man es mit sehr unterschiedlichen Kulturen zu tun und selbstverständlich auch mit einem hohen Grad an lokaler Autonomie.

3.2 Menge an Software

Eine weitere Herausforderung besteht in der schieren Menge der benötigten Software. Finanzdienstleister gehören zu den Unternehmen, mit einem der höchsten Anteile von Software an der Wertschöpfung. Der GDV schätzt die Ausstattung eines größeren Versicherers mit Software daher auf ca. 150 bis 350 Mio. Euro - wenn man alles in einem Projekt erstellen wollte.

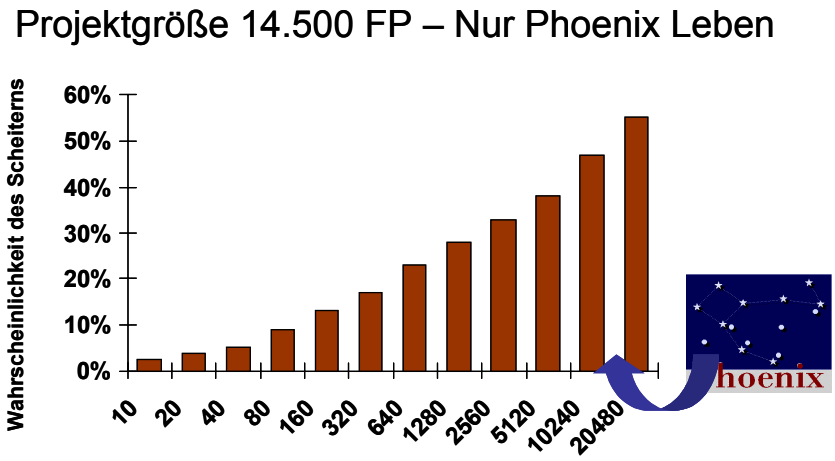

Abbildung 2: Relation zwischen Projektgröße und Wahrscheinlichkeit, dass ein Projekt scheitert. Basiert auf Erhebungen von SPR / Capers Jones. Das Projekt Phoenix-Leben mit seinen 14.500 FP deckt allerdings nur etwa 25% des Anwendungsportfolios einer Kompositversicherung ab. Das System Phoenix-Leben besteht aus Produkt / Vertrag / Leistung / Partner / in- und Exkasso. 14.500 FP entsprechen bei einer Teamgröße von bis zu 150 Personen und einer Projektlaufzeit von 3,5 Jahren etwa 500 Bearbeiterjahren Aufwand.

Mit dem Programm Phoenix hat die Generali einmal den in der Einleitung dargestellten homogenen Ansatz verfolgt. Man wollte alle Anwendungen in einer einheitlichen, modernen objektorientierten Architektur bauen und hat dabei (Anfang der 90er Jahre) sehr fortschrittlich auf Smalltalk gesetzt. 1995 wurde das Projekte Phoenix-Leben gestartet, das eine Größe von ca. 14.500 Function Points hatte. 1998 wurde die Software produktiv gesetzt. Nun wäre es an der Zeit gewesen, mit den dabei

entstandenen Frameworks das komplette Softwareportfolio wie Sachversicherung, Provision, Außendienst etc. zu erneuern. Doch leider war der technologische Wandel schneller und das Internet hatte inzwischen an Popularität gewonnen und Fat-Client Anwendungsarchitekturen weitgehend obsolet gemacht.

In Zeiten, in denen ein Technologiezyklus mit 3 Jahre schon wesentlich kürzer ist, als die Zeit, die man selbst bei bestem Projektmanagement benötigt um 350 Mio. Euro in Projekten umzusetzen, muß also das Argument entfallen, dass man für eine Rundumerneuerung eines Anwendungsportfolios mit genau einer Anwendungsarchitektur ohne Anpassungen auskommen wird.

3.3 Restrukturierungen und Fusionen

In der Zeit, in der nun Phoenix Leben in Produktion gesetzt wurde, wurde in der internationalen Generali Gruppe (Holding Triest) zuletzt kräftig umstrukturiert

- 1998 kauft die Triester Holding die Deutsche AMB Holding (Aachener und Münchener Versicherungsgruppe) und damit in Deutschland einen Versicherungskonzern der ein mindestens doppelt so hohes Prämienvolumen hat, wie die Vienna Holding

- Dadurch wird die Generali/Lloyd Deutschland, die bis dahin zur Vienna Holding gehörte, Ende 1999 aus Vienna Holding und auch aus der EINEn Informatik gelöst und dort werden aus Gründen der Einheitlichkeit AMB Systeme installiert.

- Gleichzeitig werden in Österreich ab 1999 Interunfall und Generali EDV-mäßig und organisatorisch bis zum Euro-Termin 1.1.2002 enger zusammengefaßt. Bis zum Euro müssen die DV Systeme fusioniert sein und parallel entsteht ein neues Außendienstsystem, die Leben-Bestände werden migriert und die SAP Systeme werden runderneuert.

Bevor man heute also eine Fusion oder Umstrukturierung bewältigt hat, kann man damit rechnen, dass man schon mit der nächsten konfrontiert wird, bei der unterschiedliche Systeme, Entwicklungsprozesse und Kulturen zusammengefügt werden müssen. Dabei kann man davon ausgehen, dass die Geschäftsebene sich nicht davon wird bremsen lassen, ob die Informatik solche Dinge verdauen kann. Sie hat sie zu verdauen. Es gibt daher immer mehr Unternehmen, die Fusionsmanagement als eine wesentliche Kompetenz betrachten.

Die Situation in der man sich zurücklehnen kann und fertig ist und eine geordnete Welt vor sich hat, wird es nicht mehr geben. Man hat es vielmehr mit permanenten zyklischen Veränderungsprozessen von zunehmender Geschwindigkeit zu tun.

3.4 Internet, eCommerce und Vermehrung der Vertriebswege

Noch vor ca. 10 Jahren war die EDV-Welt eines Versicherers im Vergleich zu heute relativ "einfach gestrickt". Ein Backoffice System verwaltete die Verträge und mancher hatte ein Außendienst-System mit dem Anträge erstellt und meist per Papier zur Verarbeitung eingereicht wurden. Abbildung 3 stellt ein solches Szenario dar.

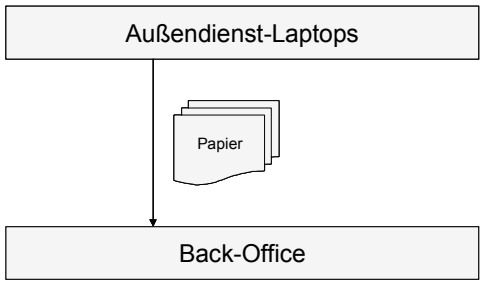

Abbildung 3: Einfache EDV Welt eines Versicherers vor ca. 10 Jahren

In den Zeiten von Internet und eCommerce hat sich die Integration erhöht, es rückt mehr EDV näher an den Kunden, es gibt mehr sog. Channels, Wertketten werden

integriert und die Menge der Software steigt an.

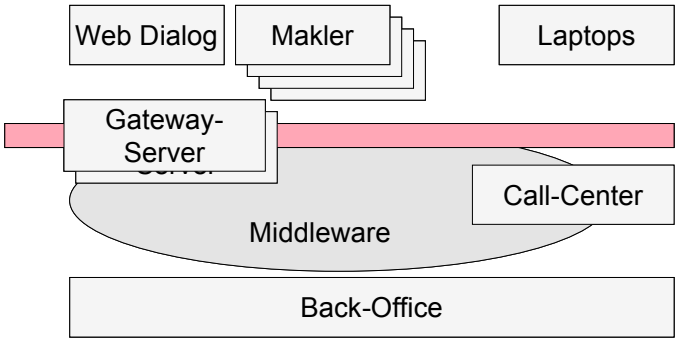

Abbildung 4 gibt einen Überblick über die heutige Welt. Außerhalb der Firewall finden wir Anwendungen für Endkunden, für Makler und für den Außendienst. Dies können Vertriebs- und Schadenanwendungen sein. Innerhalb der Firewall finden wir Call- und Servicecenter und die Backoffice Systeme. Es wird heute erwartet, dass die Frontoffice mit den Backofficesystemen gut integriert sind.

Abbildung 4: EDV-Welt eines Versicherers heute - mehr Channels und Anbindungen sind möglich

Dazu kommt dann noch, dass aus Zusammenlegungen von EDV-Einheiten in einer Einheit dann zum Teil mehrere Außendienstsysteme oder sonstige Teile vorhanden sind, die zumindest für eine Weile weiter unterstützt werden und dann vereinigt werden müssen.

4 Werkzeuge, Methoden, Verfahren

Es dürfte klar sein, dass man mit den Mitteln und der Denkweise einer klassischen Abteilung MVT (Methoden / Verfahren / Tools) in dieser neuen Welt nicht mehr zurechtkommt, weil die Geschwindigkeit des technologischen und organisatorischen Wandels zu hoch ist, um jemals in den instinktiv erwünschten Zustand des "fertig Seins" (und sei es auch nur für einen Monat) zu kommen.

Die MVT-Denkweise, die in vielen großen Firmen anzutreffen war und ist, basiert auf der Idee durch "eine" Architektur und Reuse schneller, besser und produktiver zu werden. Typische Silver-Bullets dieser Ära waren und sind: Anwendungsarchitektur (am besten durch Tools unterstützt), Querschnittsfunktionen, Frameworks und Reuse-Management. Wir wollen nicht behaupten, dass solche Ansätze komplett nutzlos geworden wären - nur stehen sie nicht mehr im Vordergrund und ihre Überbetonung macht zu unflexibel um den obigen Herausforderungen gut begegnen zu können.

Man muß daher versuchen, die Dinge zu finden, die nicht so schnellem Wandel unterworfen sind, wie die Anwendungsarchitekturen. Fündig wird man bei folgenden Punkten, die wir nachfolgend auch erläutern werden:

- Bei Architekturen legt man den Fokus von der technischen Anwendungsarchitektur auf den langlebigeren Kern des Geschäfts - die sog. *Facharchitektur* (siehe 4.1).

- Fusionen, Zusammenlegungen und Portfolioverbesserungen managt man mit sog. *Anwendungsportfolio-Management* (siehe 4.2)

- Und um heterogene Teile kommunizieren lassen zu können benötigt man minimal eine möglichst simple und robuste Middleware-Architektur (siehe 4.3).

4.1 Architekturebenen und Facharchitektur

Software-Architekturen kann man in mehrere Ebenen unterteilen.

- **Facharchitektur:** Eine Facharchitektur gibt vor, in welche Teilsysteme eine Gesamtanwendung eines Unternehmens (und das ist dann meist gültig für die gesamte Branche) zerfällt und welche Zuständigkeiten und Verantwortlichkeiten diese einzelnen Subsysteme haben.

- **Anwendungsarchitektur:** Die Anwendungsarchitektur definiert in welche Subsysteme und Schichten eine Anwendung (im wesentlichen ohne Betrachtung der Fachlichkeit) zerfällt und man findet dann dort typischerweise Schichten wie Benutzungsschnittstelle, Geschäftsobjekte und Persistenzschicht.

- **Systemarchitektur:** Die Systemarchitektur sagt, auf welche physischen und Systemsoftware-Komponenten die Subsysteme und Module der Anwendungsarchitektur verteilt werden. Hier wird dann zum Beispiel auch über die Verteilung auf eine 2Tier- oder 3Tier-Architektur entschieden.

Abbildung 5 zeigt eine solche Aufteilung, wobei man, wenn man die Architekturmodelle sauber aufbaut die Abbildungen der Elemente von der Facharchitektur bis in die Systemarchitektur gut nachvollziehen kann. Warum wir in der Abbildung die Facharchitektur gegenüber den beiden anderen starken Schichten stark betont haben, wird im folgenden noch deutlich werden.

4.1.1 Anwendungsarchitektur

Die von Technikern am meisten und am liebsten diskutierte Architekturebene ist die der Anwendungsarchitektur. Dies wohl auch darum, weil man sie fachfrei und fast nur mit technischem Wissen diskutieren kann. Hier geht es darum zu definieren, in welche Schichten und Module eine Informatik-Anwendung zerfällt und auf welche Module und Schichten die "Fachlichkeit" wie verteilt wird. Die für "Business Systems" am meisten verwendete Variante ist dabei die sog. 3-Schichten-Architektur (Benutzungsschnittstelle, Business Logik, Datenbank) (siehe zum Beispiel [Ren+97]).

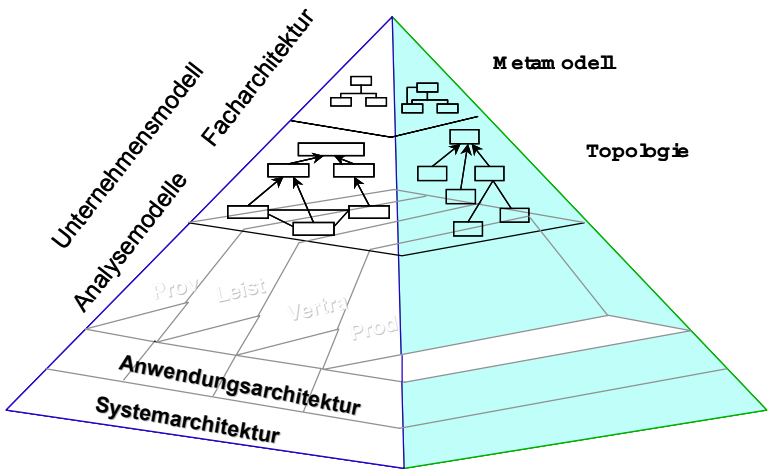

Abbildung 5: Architekturebenen: Facharchitektur, Anwendungsarchitektur und Systemarchitektur können hierarchisch aufeinander abgebildet werden.

Die Diskussion über Anwendungsarchitekturen wurde und wird immer noch gerne geführt. Es hat sich allerdings in letzter Zeit etwas Wesentliches geändert: Anwendungsarchitekturen sind durch Wellen wie Internet, Java, J2EE und ähnliches relativ schnell wieder technisch obsolet (jedenfalls in den Varianten) weil die Programmiermodelle erheblichen Einfluß auf die APIs der Subsysteme und auch auf die Interaktion der Komponenten haben und man bekommt sie heute mit einem neuen Programmiermodell wie zum Beispiel J2EE fast gratis dazu. Im Bereich Enterprise Java Beans bekommt man zum Teil nicht nur die Anwendungscontainer als Open Source sondern dazu auch komplette Anwendungsbeispiele und zum Beispiel eine komplette eCommerce Shopping Anwendung.

Aufgrund des schnellen Wandels und aufgrund der Tatsache, dass man heute große Teile der Anwendungsarchitekturen frei vom Markt bekommt, wird man durch Beschäftigung mit diesem Thema kaum noch Wettbewerbsvorteile erreichen können.

4.1.2 Systemarchitektur

Bei der Systemarchitektur wird die Frage behandelt, wie man die Subsysteme und Komponenten einer Anwendungsarchitektur konkret auf Rechner verteilt, welche Systemsoftware man dabei zur Kommunikation einsetzt und wo man zum Beispiel Server postiert.

Dabei ist im Gegensatz zu Anwendungsarchitekturen zu beobachten, dass Systemarchitekturen einem wesentlich langsameren Wandel unterliegen: Jedenfalls bei einem großen Finanzdienstleister. Die Komponenten der Systemarchitektur müssen besonders stabil und besonders gut ausgetestet sein, bevor sie in der Breite in Betrieb gehen können. In einem CICS, einer DB2 Datenbank oder auch einem SNA- oder TCP/IP-Netzwerk stecken erhebliche Aufwände für die Inbetriebnahme und Stabilisierung, die man nicht alle Jahre wieder wegwerfen kann und wird. Von daher sind wir mit Änderungen in der Systemarchitektur sehr vorsichtig und konservativ.

4.1.3 Facharchitektur

Die Facharchitektur definiert, welche fachlichen Geschäftssysteme wir bauen und wie diese zusammenspielen. Dabei ist die Facharchitektur der Generali Vienna sehr ähnlich zu anderen Ansätzen in der Branche, wie zum Beispiel VAA oder IAA.

Abbildung 6 alleine als Darstellung auf der obersten Ebene würde allerdings mehr verwirren als Klarheit stiften. Wir haben daher mehrere Ebenen der Beschreibung

- **Konstruktionsprinzpien und Entwurfsmuster:** Beschreiben, wie die Geschäftssysteme zusammenspielen. So haben wir als unsere wesentlichen Konstruktionsprinzipien Produktorientierung, Geschäftsprozeßorientierung und modellbasierte Entwicklung (Meta-Systeme) und einige mehr, die wir auch in Form von Entwurfsmustern[1][Kel98, Kel99a] beschreiben.

- **Nennung der Geschäftssysteme:** In der nächst tieferen Ebene werden die einzelnen Geschäftssysteme beschrieben und ihre wesentlichen Schnittstelleneigenschaften werden genannt.

- **Subsysteme:** Die Geschäftssysteme zerfallen wieder in fachliche Subsysteme und diese werden zum Teil (allerdings noch flächendeckend) bis auf Klassenebene beschrieben.

[1] siehe dazu zum Beispiel [Bus+96, GOF95]

Die Facharchitektur besteht also aus e: iner Menge von Entwurfsmustern (Patterns), einem Dokument, das die Geschäftssysteme beschreibt und aus einem Subsystemmodell, welches in Rational Rose dokumentiert ist.

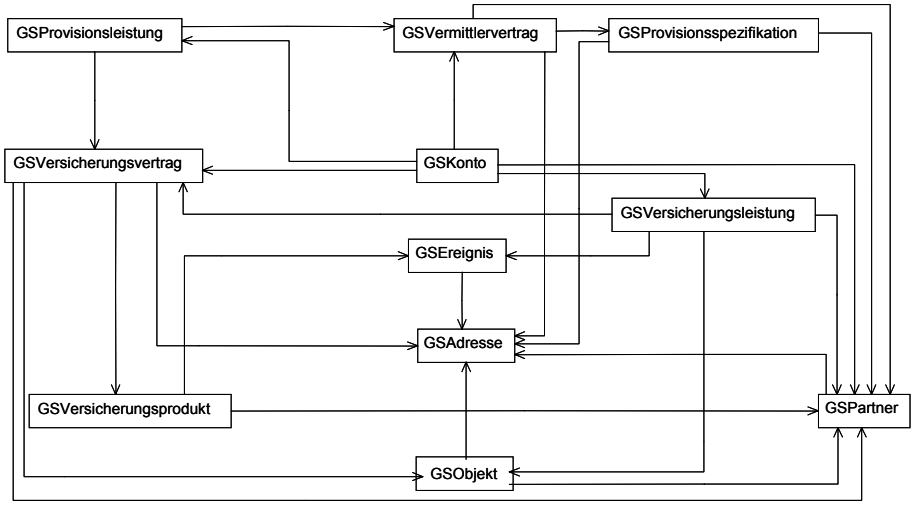

Abbildung 6: Geschäftssysteme der Generali Facharchitektur: Ähnliche Darstellungen findet man häufig als oberste Ebene einer Facharchtektur. Daraus werden jedoch die Ideen nicht klar, die die Konstruktion der Software antreiben - um diese Ideen zu transportieren, sind Entwurfsmuster besser geeignet.

Dabei macht man die Erfahrung, dass man mit der Facharchitektur "nie fertig" wird, wenn man sie unter der Ebene von Subsystemen versucht auszuspezifizieren. Wir verzichten daher bewusst darauf. Man macht ferner die Erfahrung, dass die Prinzipien der Architektur über mehrere Geschäftssysteme wesentlich besser verständlich werden, wenn man Entwurfsmuster zur übergreifenden Erklärung von Konstruktionsprinzipien wie Produktorientierung einsetzt.

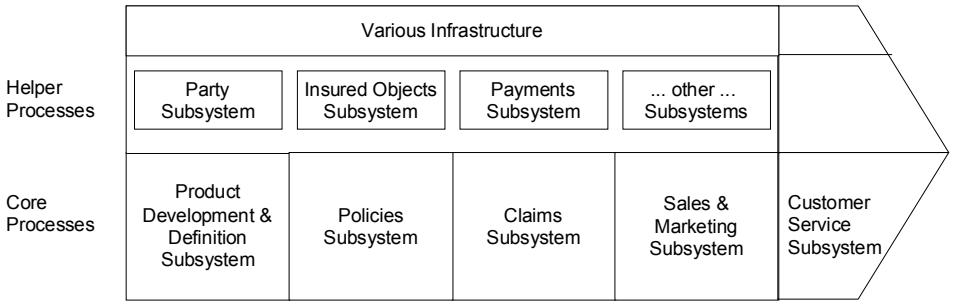

Abbildung 7: Darstellung der fachlichen Subsysteme analog Abbildung 6 aber entnommen aus einem unserer Entwurfsmuster (Value Chain), die die Zusammenhänge zwischen den Geschäftssystemen erklären sollen. Details siehe [Kel98, Kel99b]

Während durch technologische Wellen wie das Internet die zwei anderen Architekturebenen einem starken Änderungsdruck unterworfen sind, gilt das weniger für die Essenz des Versicherungsgeschäfts, die sich in der Facharchitektur niederschlägt. Wir werden zum Thema APM (siehe Abschnitt 4.2) dann sehen, dass genau die Facharchitektur daher heute der wesentliche Asset ist, den man auch für das Fusionsmanagement einsetzen kann und muß.

4.1.4 Wie viele Architekturen kann man zulassen?

Aus den obigen Ausführungen ergibt sich dann, mit wie vielen "Architekturvarianten" man es jeweils zu tun haben wird:

- auf der Ebene der Facharchitektur gibt es genau eine Ausprägung, die den übergreifenden Generalbebauungsplan darstellt. Die Facharchitektur ist dabei ein Referenzmodell, das sicher nicht in jeder Systemgeneration voll zu 100% implementiert ist - man weiß aber damit immer, wohin man seine Systeme fachlich konvergieren lassen möchte.

- schon aufgrund der Technologiegenerationen wird es viele Anwendungsarchitekturen geben. Nur weil Java-Serverarchitekturen gerade modern sind, ist das noch lange kein wirtschaftlicher Grund deshalb alle Smalltalk- oder Host-PL/I-Systeme zu entsorgen. Damit hat man dann allerdings viele Anwendungsarchitekturen gleichzeitig im Betrieb.

- da alle diese Varianten von Anwendungsarchitekturen auf einer gemeinsamen technischen Infrastruktur laufen, kann es nur eine Systemarchitektur geben, die die Vereinigungsmenge dessen darstellt, was man benötigt, um seine Anwendungsarchitekturen lauffähig zu machen. Man sollte allerdings darauf achten, dass diese Menge möglichst klein ist.

Damit ist man allerdings von der Idealvorstellung der klassischen MVT Abteilung eines Großunternehmens einigermaßen weit entfernt, die mehr als eine Anwendungsarchitektur als eine Zumutung empfinden würden. Da man allerdings in einem Technologiezyklus nicht alles neu bauen kann und dies nicht im entferntesten wirtschaftlich wäre, haben wir uns von dieser Vorstellung verabschiedet.

4.2 Anwendungsportfolio-Management

Wir haben bisher dargelegt, dass die Facharchitektur als fachlicher Generalbebauungsplan für alle Anwendungen eine zentrale Rolle spielt. Wir wollen nun zeigen, wie man sie in konkreten Entscheidungssituationen einsetzen kann. Dazu bedienen wir uns einer Entscheidungssituation, die so wirklich anstand. Ende 1998 wurde die Frage diskutiert, wie denn die wesentlichen Anwendungen der Generali und Interunfall in Österreich euro-fit zu machen seien. Dabei gab es folgende wesentliche Alternativen

- Beide Systeme auf den Euro umstellen - wurde schnell ausgeschlossen, weil sehr teuer.

- Nur eines der Systeme weiterverwenden:

    - Das der Generali,

    - oder das der Interunfall.

- Oder beide Systeme auf ein neues migrieren - diese Variante wurde wegen des zu hohen Risikos verworfen.

Es blieb also die Frage, welches der beiden Kernsysteme (Generali oder Interunfall) weiterverwendet werden sollte. Zur Entscheidung dieser Frage wurde ein sog. Detail-APM eingesetzt.

4.2.1 Detail-APM

Bei einem Detail-APM werden Systeme dadurch bewertet, dass sie gegen eine Norm oder auch Checkliste detailliert abgeglichen werden. In unserem Fall ist diese Checkliste die Facharchitektur. Dabei wird pro Geschäftssystem und pro Subsystem vergleichen ob diese Systeme vorhanden sind und in welcher Qualität und fachlichen Vollständigkeit diese Systeme vorhanden sind (durch Abgleicht mit der Checkliste = Facharchitektur)

Dabei kann man auf folgende Fälle stoßen:

- Ein Element der Facharchitektur ist in dem zu untersuchenden System nicht vorhanden - es "fehlt" also - je nachdem um was es sich konkret handelt, kann dies ein Killerkriterium darstellen oder aber auch nicht - die Bewertung muß inhaltlich vorgenommen werden und kann nicht rein formal ohne Fachwissen erfolgen

- das Element der Facharchitektur ist vorhanden

- in dem zu untersuchenden System ist etwas vorhanden, was in der Facharchitektur nicht vorhanden ist. In diesem Fall gibt es die Möglichkeiten,

    - dies einfach zu notieren

    - oder aber wenn man der Meinung ist, dass es sich um eine sehr nützliche und wichtige Funktion handelt, die Facharchitektur zu ergänzen und fortzuschreiben.

Bei jedem Detail-APM wird also auch die Facharchitektur besser werden, weil sie realistischerweise nie den Anspruch erheben kann, besser zu sein als die Menge aller Designer, die die bisherigen Ist-Systeme gebaut haben, die damit verglichen werden.

Um die Fallstudie abzuschließen: Im konkreten Fall wurde für beide Ist-Systeme eine Überdeckung von ca. 70% mit der Facharchitektur festgestellt und die Entscheidung fiel dann für das System, das technisch übersichtlicher und leichter wartbar erschien.

4.2.2 Überblicks-APM

Es ist nicht gerade üblich, eine Diskussion bottom-up zu beginnen. Wir haben gezeigt, wie APM bei einer Systementscheidung System A gegen System B helfen kann (Detail-APM). Wir werden jetzt an einem weiteren Fall demonstrieren, wie APM bei der Konsolidierung mehrerer Standorte helfen kann. Aktuell gibt es bei uns die Aufgabe das Anwendungsportfolio der Reformstaaten einer Routineüberprüfung zu unterziehen. Dazu benutzt man ein sog. Überblicks-APM, bei dem man sich pro Geschäftssystem ansieht, durch welche Anwendungen dieses Geschäftssystem an welchem Standort abgedeckt wird. Für einen Standort (mittleres bis kleineres Versicherungsunternehmen) benötigt man dabei mit Vor- und Nachbearbeitung ca. eine Woche Arbeit für zwei interne Berater.

4.2.3 Und was macht man mit den gewonnenen Informationen?

Durch ein Überblicks-APM gewinnt man Anhaltspunkte für Handlungsbedarf, wenn man zum Beispiel feststellt, dass an 4 Standorten 4 verschiedene Inkasso-Systeme verwendet würden[2]. In

[2] Dies ist allerdings kein realer Fall, sondern ein fiktives Beispiel.

diesem Fall würde man mittels Detail-APM feststellen, welches der Inkasso-Systeme weiterverwendet werden soll oder ob man sich für externe Software oder ein komplett anderes System entscheiden möchte.

Insgesamt hat es jeder lokal verantwortliche Informatik-Manager einer Gesellschaft mit einem zyklischen Prozeß zu tun. Dieser ist in Abbildung 8 dargestellt.

- Zunächst hat man ein existierendes Portfolio vor sich, das man mit einem Überblicks-APM auf Handlungsbedarf untersuchen kann.
- Man wird dann daraus seine Maßnahmen ableiten, wie bestimmte Systeme weiterzuentwickeln, Systeme zuzukaufen oder punktuell neue zu entwickeln oder sie durch Systeme andere Konzerngesellschaften zu ersetzen.
- Daraus ergibt sich ein Arbeitsplan der umgesetzt werden muß (Systeme integrieren). Man erhält eine neue Systemlandschaft.

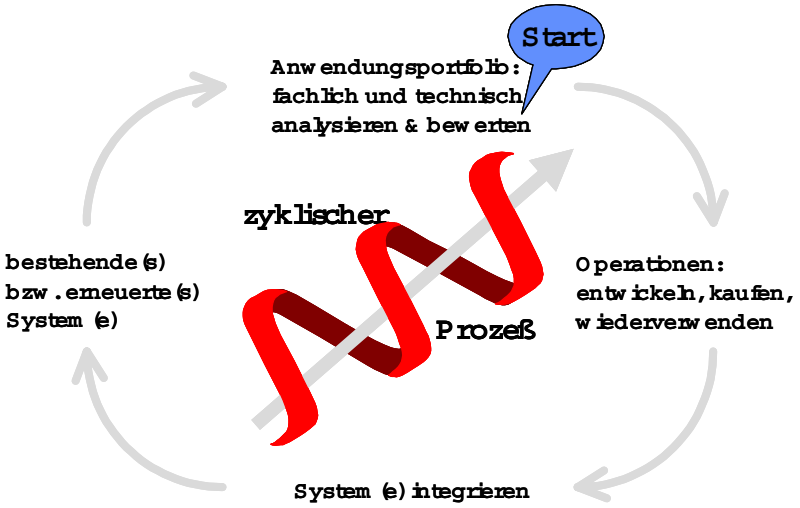

Abbildung 8: Zyklischer APM Prozeß

Und an dem Tag, an dem man die in Betrieb genommen hat, beginnt der Zyklus von vorne, weil wieder neue Anforderungen vorliegen, sich die Technik weiterentwickelt hat oder es neue Geschäftsfelder oder Konzernkonstellationen gibt. Wie oben schon erwähnt, wird für die Evaluierungen jeweils die Anwendungsarchitektur als Referenzmodell verwendet und fortgeschrieben. Auch sie ist also keine fixe Größe, sondern entwickelt sich dynamisch weiter.

4.3 Middleware-Architektur

In Abschnitt 3.4 wurde gezeigt, dass wir durch Internet und eCommerce eine rasante Vermehrung von zu bedienenden sog. Channels bekommen. Durch die Diskussion von Anwendungsarchitektur und Anwendungsportfolio wurde gezeigt, dass man dabei heute Systeme aus unterschiedlichsten Technologiegenerationen von Host über Smalltalk-Fat-Client bis zu HTML-User-Interfaces mit Java-Servern zu einem Gesamtsystem integrieren können muß. All diese Systeme müssen sinnvoll zusammenarbeiten können.

Wenn man dem Marketing von EAI-Toolherstellern glaubt, ist genau dies das Einsatzfeld für unternehmensweite Broker-Architekturen, Sternverteiler und ähnliche technische Mittel, die eine

ziemlich komplexe, unternehmensweite Kommunikationsinfrastruktur zur Verfügung stellen. Auch CORBA zielt in eine solche Richtung.

Eine irgendwie geartete Middleware-Strategie wird man also benötigen - fragt sich nur auf welcher technischen Basis und wie komplex die technisch sein muß. Um festzustellen, was ausreicht, um die Herausforderungen der Integration heterogener Anwendungen zu bewältigen, ist ein Blick auf die Marktanteile im Middleware-Markt recht hilfreich.

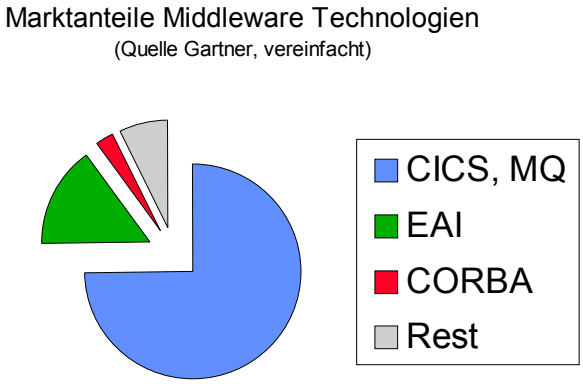

Abbildung 9: Marktanteile auf dem Middleware-Markt - Quelle Gartner Group / leicht vereinfacht

Abbildung 9 zeigt, dass etwa 75% des Middleware-Marktes auf CICS und MQSeries entfallen. Ca. 10% entfallen auf den EAI Markt (Enterprise Application Integration) und nur etwa 3% auf CORBA-Produkte. von den ca. 10% EAI entfällt wieder der Löwenanteil auf Add-ons zu CICS und MQSeries. Wir haben uns daher aus pragmatischen Gründen für eine Low-Tech-Middleware Strategie entscheiden, die auf CICS und MQSeries basiert und bei der die Anwendungsserver in den meisten Fällen noch hostbasiert bleiben. Angesichts des hohen Anteils an existierenden Anwendungen ist dies der schnellste und einfachste Weg, um die Internet-Channels mit zuverlässigen (weil großteils schon implementierten) Services zu bedienen.

5 Low-Tech statt High-Tech: Was machen wir mit ...

Wenn man die Computerfachpresse liest, die vor allem von Technikern von der Hersteller- und Beraterseite mit Beiträgen versorgt wird, kann man zu dem Eindruck kommen, dass man ziemlich "hinter dem Mond" lebt, wenn man nicht permanent das volle Spektrum der aktuellen technologischen Innovationen (oder auch Silver-Bullets) einsetzt.

Auf der STJA99, einer Konferenz für Smalltalk und Java Entwickler, provozierte Adele Goldberg die Teilnehmer damit, dass sie sie fragte, ob sie glaubten, dass High-Tech relevant für die Anwendungen sei, mit denen im Internet derzeit Geld verdient würde [Gol99]. Sie zählte dann eine ganze Menge Low-Tech-Technologien auf, die alle ubiquitär und sehr wirksam sind und mit denen die Menge des eCommerce und des Webs betrieben wird: TCP/IP, CGI Perl Programme, HTML. In einer zweiten Keynote wies Jim Coplien [Cop99] noch darauf hin, dass er nicht glaube, dass sich die Lego-Block Ideen von Component Software durchsetzen würden, weil reines Objektdenken sich an vielen Stellen als an der Praxis vorbei herausgestellt hat - er setzte dem das sog "Multiparadigm-Design" [Cop98] gegenüber, das andere Ansätze, wie zum Beispiel Meta-Programmierung oder generative Programmierung mit einbezieht.

Im folgendenden soll daher noch kurz diskutiert werden, wie wir mit einigen technologisch als "heiß" gehandelten Ansätzen im Rahmen unseres Architekturmanagements umgehen. Dabei soll kurz auf Componentware (siehe Abschnitt) und CORBA und andere objektorientierte Middleware (siehe Abschnitt) eingegangen werden.

5.1 Componentware

Wenn man sich mit dem Begriff Componentware befasst, stellt man zunächst fest, dass er sehr schwach definiert ist. Die Definitionen reichen dabei von "allem, was ein definiertes Interface" hat, bis zu der Forderungen, dass Components plattformunabhängig und weitgehend kontextfrei einsetzbar sein sollen. Je nach Definition ist fast jedes Stück Software eine Komponente oder aber die Definitionen sind so hart, dass es Komponenten praktisch nicht gibt. Allein der Hype lässt einen aber nicht umhinkommen dazu Stellung zu beziehen. Wir verwenden daher Komponenten in mehreren Ausprägungen:

- Zum einen bei der Oberflächenprogrammierung: Dort haben sich Components im Sinne von Microsoft's COM Standard sehr weit verbreitet. Solche Components erlauben aber auch noch keinen unternehmensweiten Reuse von geschäftsspezifischen Komponenten vor - aber auch ein fertig einsetzbares Tabellenwidget kann schon sehr viel Aufwand einsparen. Komponentenbasierter Reuse auf Architekturlevel, von dem zum Beispiel bei der Branchenarchitektur VAA gesprochen wird, ist das allerdings noch nicht.

- Zum anderen betrachten wir komplette Geschäftssysteme unserer Facharchitektur zugleich auch als Komponenten. Sie haben eine klar definierte Schnittstelle und lassen sich mit etwas Glue-Code zu einem sinnvollen Gesamtsystem integrieren und einzeln sinnvoll installieren. Dabei kann ein in ABAP geschriebenes SAP-In/Exkasso-System mit einem in Smalltalk geschrieben Vertragssystem zusammenarbeiten.

Der Reuse von "Geschäftsobjekten" quer über Geschäftssysteme steht bei uns allerdings nicht im Mittelpunkt. Die Erfolge diesbezüglicher Anstrengungen und Normungsbemühungen hielten sich meist in engen Grenzen

Damit begegnen wir dem Component-Hype in unserem Architekturmanagement insgesamt eher bodenständig.

5.2 CORBA und andere objektorientierte Middleware

Wenn CORBA und objektorientierte Middleware die Lösung sind - was ist dann das Problem. Wenn man dieser Frage nachgeht, wird es einem teilweise schwer fallen, zumindest im Umfeld der Anwendungslandschaft eines Finanzdienstleisters die Antwort darauf zu finden. Wir behandeln CORBA und EAI daher im Rahmen unseres Architekturmanagements recht zurückhaltend.

Wenn wir eine Anwendung zukaufen, die intern CORBA verwendet und sich nach außen wie eines unserer anderen Geschäftssysteme verhält, wird uns das nicht stören und wir werden sie in unser Portfolio nach den Vorgaben der Facharchitektur integrieren. Wir werden allerdings nicht die Forderung stellen, dass irgendwelche Anwendungen CORBA-compliant sein müssen.

6 Zusammenfassung

Dieser Artikel hat einen Überblick über die Herausforderungen gegeben, mit denen heute das Architekturmanagement eines Großanwenders konfrontiert ist. Die wesentlichen sind vor allem

Menge der zu managenden Software, Fusionsprozesse und Integration neuer Kanäle für den eCommerce. Dabei treten traditionelle homogene Ansätze, die über lange Jahre große Unternehmen dominiert haben, wie absolut einheitliche Anwendungsarchitektur und Reuse-Management in den Hintergrund gegen die Fähigkeit Fusionen zu managen und existierende Anwendungen, eigene Neuentwicklungen und zugekaufte Software zu einem sinnvollen Ganzen integrieren zu können.

Bei der Anwendungs- und Systemarchitektur liegen dabei weniger die Schlüssel des Erfolges - auch nicht bei High-Tech Lösungen. Der Schlüssel liegt eher in der Fähigkeit Software mit Hilfe einer Facharchitektur schnell fachlich beurteilen zu können und eine Low-Tech Middleware-Strategie zu haben, die es einem erlaubt, ausreichend schnell Anwendungen auszutauschen. Anwendungsportfolio-Management ist eine Methode, die dieses Vorgehen systematisiert. Eine minimale Middleware-Strategie basierend auf Technologien mit hohem Marktanteil im klassischen Bereich ist dazu sehr hilfreich.

7 Danksagung

Die in diesem Artikel vorgestellten Ansätze sind durch viele Beiträge geprägt und wurden vom Autor lediglich zusammengefasst. Ich bedanke mich daher bei Harry Fräser, dem Erfinder unseres APM und Gertrude Rabl unserer Facharchitektin. Bernhard Anzeletti hat unsere Middleware-Strategie entscheidend vorangebracht, Rudolf Lewandowski prägt stark unsere Java-Architekturen. Die Herren Robert Aldrup und Martin Friedrichsen (beide agens consulting) haben unsere Facharchitektur entscheidend mitgeprägt und Herr Rüdiger Lang (agens consulting) hat als langjähriger Projektleiter von Phoenix-Leben dafür gesorgt, dass die Facharchitektur mit Phoenix-Leben in einem System flächendeckend in Produktion gehen konnte.

8 Literatur

[Bus+96] **Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal:** *Pattern Oriented Software Architecture - A System of Patterns*, Wiley 1996.

[Cop98] **Jim Coplien**: Multipradigm-Design for C++, Addisson-Wesley 1998.

[Cop99] **Jim Coplien**: *"Design after Modernism: Beyond the Object", with apologies to John Thackara*, Keynote, STJA 1999.

[GOF95] **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides**: *Design Patterns, Elements of Reusable Object-oriented Software*, Addison-Wesley 1995.

[Gol99] **Adele Goldberg**: *Software Development as Patterns of Interaction*, Keynote, STJA 1999.

[Kel98] **Wolfgang Keller:** *Some Patterns for Insurance Systems*, Proceedings, PLoP 1998, Allerton Park, IL.

[Kel99a] **Wolfgang Keller**: *The Pitfalls of Business Rules and Meta Systems*, White Paper, Background Material for a Tuturial [Kel99b] at STJA 1999.

[Kel99b] **Wolfgang Keller**: *Building Flexible Insurance Applications*, Tutorial, STJA 1999, Erfurt, Germany.

[Ren+97] **Klaus Renzel, Wolfgang Keller:** *Three Layer Architecture* **in Manfred Broy, Ernst Denert, Klaus Renzel, Monika Schmidt (Eds.)** *Software Architectures and Design Patterns in Business Applications*, Technical Report TUM-I9746, Technische Universität München, 1997.