Systemfamilienentwicklung bei arvato direct services

Jan Leßner

DCO
arvato direct services
An der Autobahn
33310 Gütersloh
jan.lessner@bertelsmann.de

Abstract: Für die Erbringung von CRM-Dienstleistungen wird bei arvato direct services seit einigen Jahren eine Familie von Softwaresystemen entwickelt. Bei der Betrachtung der konkreten Vorgehensweise lassen sich dabei Analogien zu natürlichen Vorgängen und zur gezielten gentechnischen Steuerung derselben ziehen. Über eine solche Analogiebetrachtung lässt sich Aufschluss über erhaltenswerte Erfolgsfaktoren und über Risiken und Grenzen der angewandten Entwicklungsprozesse gewinnen.

1 Software und Familie

Die arvato direct services GmbH ist einer der größten Direkt-Marketing-Dienstleister weltweit und betreibt zur Erbringung ihrer Dienstleistungen für unterschiedliche Kunden eine Reihe maßgeschneiderter Software-Systeme. Dabei darf einerseits nicht jedes Mal das Rad neu erfunden werden, andererseits muss genügend Raum für die individuellen Bedürfnisse der Geschäfte bleiben. Im Folgenden wird dargestellt, welche Anforderungslage bei arvato dabei zur Bildung einer ganzen Familie verwandter Systeme geführt hat, und welche Konzepte sich auf lange Sicht als entscheidende Erfolgsfaktoren in der produktiven Praxis heraus gestellt haben. Verwendet man hierbei den Begriff der „Familie“ im Zusammenhang mit Software-Entwicklung, dann liegt es nahe, zum Verständnis von Zusammenhängen einen Blick auf natürliche Vorgänge der Familienbildung zu werfen, sowie auf die Möglichkeiten gezielter Einflussnahme durch menschliche Eingriffe in diese Prozesse. Und tatsächlich tun sich bei einer solchen Analogiebetrachtung interessante Parallelen auf, aus denen sich Empfehlungen, Chancen und Risiken für die Software-Entwicklung ableiten lassen. Ein Großteil der hier geschilderten Erkenntnisse ist dabei unabhängig von einem konkreten Geschäftsfeld.

Zum Kerngeschäft von arvato direct services gehört die gesamtheitliche Abbildung von sogenannten Kundenbindungsprogrammen, in denen Endverbraucher durch das Gewähren von Bonusleistungen zum regelmäßigen Einkauf bei den Programmbetreibern animiert werden. Die Betreiber solcher Programme – und damit die Kunden von arvato – kommen aus sehr unterschiedlichen Bereichen, z.B. Krankenversicherungen, Baumärkte, Lebensmittelketten. Mit der Entwicklung und auch der langfristigen Wartung der benötigten Software-Systeme wird üblicherweise eine interne IT-Abteilung beauftragt. Das Volumen der Entwicklungsprojekte liegt überwiegend im Bereich von 100 bis 1000 Personentagen bei einer Projektlaufzeit von 2 – 6 Monaten. Die Entwicklung erfolgt oft unter hohem Kosten- und Zeitdruck und setzt daher einen effizienten Entwicklungsprozess voraus. Die IT-Abteilung umfasst einen Stamm von ca. 15 festen und zurzeit noch einmal 15 externen Mitarbeitern und ist in dieser Form seit ca. 4 Jahren tätig. Dementsprechend wurde bereits eine ganze Reihe von Systemen entwickelt und befindet sich aktuell auch in Wartung. Die Entwicklung basiert vollständig auf Java und den Standards der Java 2 Enterprise Edition (J2EE).

2 Anforderungsprofile

Durch den thematischen Schwerpunkt ergeben sich im Anforderungsprofil für die beauftragten Systeme immer wieder erkennbare Parallelen. So besteht ein immer wiederkehrendes Verständnis dafür, wie bestimmte Geschäftsobjekte IT-technisch abzubilden sind, z.B. Kunde, Kundenkarte, Punktekonto oder Bestellung. Es gilt auch häufig, immer wieder die gleichen externen IT-Systeme zu integrieren, wie z.B. die Adressprüfung einer anderen Konzerntochter, ein Debitorenmanagement oder die Druckstraßen des Hauses Bertelsmann für hochvolumigen Postversand. Bzgl. der abzubildenden Geschäftprozesse lassen sich ebenfalls wiederkehrende Anforderungen finden, z.B. die Registrierung von Neukunden, der Verfall von Bonuspunkten, die Fakturierung von Bestellungen oder der Kündigungsvorgang. Und schließlich gibt es auch bestimmte Schnittstellen, die jedes IT-System nach außen anbieten muss – im Kern zumeist eine effiziente graphische Oberfläche für die Arbeit der Servicecenter und Webservices für den Anschluss von Web-Oberflächen an das Backend.

Der hohe Grad an wiederkehrenden Aufgaben spricht auf den ersten Blick für die Verwendung eines Standardprodukts für CRM-Lösungen als Basis der Entwicklung. Es bestehen aber andererseits mindestens ebenso viele projekt-individuelle Anforderungen, die diesen Ansatz schwer machen. Dies liegt unter anderem daran, dass Kundenbindung zwar der Schwerpunkt aber nicht das einzige Standbein ist. So liegt beispielsweise auch die Entwicklung einer Plattform zur Verwaltung von Riesterrentenanträgen in der Verantwortung des IT-Bereichs, sowie die Abwicklung von Bestellungen für SIM-Karten eines Mobilfunkbetreibers. Aber selbst die Kundenbindungsprogramme haben teilweise sehr spezifische Profile, die eine Individualentwicklung notwendig machen.

Das oben beschriebene Spannungsfeld aus wiederkehrenden und individuellen Anforderungen prägt das Bild der Software-Entwicklung bei arvato direct services und hat zum Prinzip einer Systemfamilienentwicklung geführt. Bewährte Bausteine aus früheren Projekten müssen in bestmöglicher Weise in neuen Projekten wieder eingesetzt werden, um auf diese Weise Aufwände zu minimieren. Andererseits muss die Verwendung solcher Bausteine in einer Weise erfolgen, die keine Barriere für die spezifische Individualentwicklung darstellt. Wie macht man also in diesem Umfeld das Beste aus seinem Erbe?

Bei der Suche nach einer geeigneten Erläuterung der verfolgten Konzepte ließ sich eine aufschlussreiche Analogie zu natürlichen Vorbildern aufstellen, aus denen sich einige einleuchtende Grundprinzipien für die Systemfamilienentwicklung ableiten lassen.

3 Eine Analogie zur Natur

Spricht man in der Software-Entwicklung von „Familien“, dann lohnt sich ein Blick darauf, wie sich bei natürlichen Lebensformen eine Fortentwicklung und Familienbildung ergibt und welche Einflussmöglichkeiten der Mensch dabei hat. Die Evolution – der natürliche Vorgang aus Mutation und Selektion – ist dabei das Fundament. Sie erzielt zwar allerbeste Ergebnisse von perfekt angepassten Lebensformen, ist aber auch extrem langwierig und produziert wegen des zufälligen Faktors darin eine große Menge von Varianten. Trotz des guten Rufs also kein unmittelbares Vorbild, obwohl der Begriff der evolutionären Entwicklung in der IT-Welt durchaus auftaucht (siehe z.B. [DAHME]).

Deutlich schneller kommt man zu neuen Ergebnissen durch menschlichen Eingriff in den Selektionsprozess – die Züchtung. Dieses Verfahren beschränkt sich üblicherweise auf eine einzige Familie von Pflanzen oder Tieren (was uns ja ausreichen würde) und genießt auch noch einen relativ guten Ruf. Trotzdem braucht es immer noch mehrere Generationen und oft ein paar Jahrzehnte Zeit, bis sich ein gewünschtes Ergebnis einstellt. Will man bereits innerhalb einer einzigen Generation eine nennenswerte Fortentwicklung bewirken, muss auch in den Prozess der Mutation eingegriffen werden, womit man in den Bereich der Genmanipulation kommt. Grundlage für die Mutation ist eine deutlich gezieltere Form der Selektion – und zwar die Extraktion und Isolation besonders gewinnträchtigen Erbguts. Hier ergeben sich die erstaunlichsten Möglichkeiten, wobei der Ruf dieser Technik zu Recht zweifelhaft ist, weil sie schwer in den Griff zu bekommen und vor allem ethisch bedenklich ist [FRAGNER]. Es besteht außerdem die erhebliche Gefahr, dass sich auch genetisch defekte und unerwünschte Lebensformen unkontrolliert selbst fortpflanzen.

In der Genmanipulation liegt also sowohl das größte Potential als auch die Grenze des menschlich Beherrschbaren bei der Fortentwicklung einer Familie von Lebensformen. Was lässt sich nun aus der allgemeinen Anschauung dieses Bereichs für die Softwareentwicklung lernen, wenn wir die Begrifflichkeiten und Methoden auf den Bereich der Systemfamilienentwicklung übertragen?

Für die Arbeitsweise im IT-Bereich von arvato lassen sich folgende Analogien bilden:

- Eine Generation von Lebewesen entspricht einem System der Systemfamilie
- Das „Ausbrüten“ einer solchen Generation entspricht der Entwicklung dieses Systems
- Das Erbgut lässt sich gleichsetzen mit dem arvato-eigenen Anwendungsframework
- Der Vorgang der Selektion (Erbgutextraktion) entspricht der Erweiterung dieses Frameworks
- Die Mutation kann mit der Anwendung des Frameworks in der Entwicklung eines neuen Systems gleichgesetzt werden.

Das Problem der ethischen Bedenklichkeit entfällt an dieser Stelle, da hier keine lebenden Wesen mehr manipuliert werden. Auch die Gefahr der Selbstfortpflanzung destruktiver Gene besteht hier glücklicherweise nicht - das wären sonst Computerwürmer, die bei arvato selbstverständlich *nicht* hergestellt werden. Auf die aber durchaus bestehende Frage der Beherrschbarkeit wird weiter unten noch genau eingegangen.

4 Analogien im Vorgehen

Betrachten wir uns das Vorgehen bei der Systemfamilienentwicklung in Analogie zur Gentechnik, so lässt sich feststellen, dass es sich hierbei um zwei mit einander verschränkte Prozesse handelt, was durch die folgende Graphik verdeutlicht werden soll:

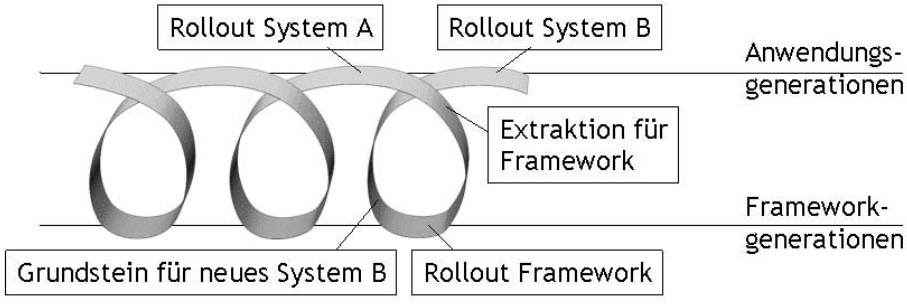

Abbildung 2: Prozess der Systemfamilienentwicklung

Ist ein System der Familie fertig gestellt worden, schließt sich ein Prozess der Extraktion wieder verwendbarer Bausteine an. Die Integration dieser gesäuberten und isolierten Bausteine ins Framework (den „Genpool“) führt zu einer neuen Version desselben. Diese neue Version dient nun wiederum als Grundlage für die Entwicklung der nächsten Generation von Systemen, indem die dafür verwendbaren Bausteine des Frameworks „injiziert“ werden. Dieser Vorgang wiederholt sich bei der nächsten Generation erneut und bildet auf diese Weise einen permanenten Anreicherungs- und Verbesserungsvorgang. Anwendungs- und Frameworkentwicklung sind dabei nicht nebenläufig und unabhängig von einander, sondern bedingen sich gegenseitig. Das Framework stellt dabei nicht nur eine technologische Basis dar, sondern umfasst auch konzeptionelle und organisatorische Aspekte wie z.B. den Aufbau des Software-Enwicklungsprozesses an sich.

Dieses Vorgehen ist auch Kerngedanke des sog. *Software Product Line Engineerings* das einen zunehmend verwendeten Begriff für ganzheitliche Wiederverwendung bildet (siehe z.B. [POHL]). Die darin verankerten Prinzipien sind zur Zeit Ausgangspunkt für Überlegungen zur Formalisierung des Requirements- und Test-Managements bei arvato direct services, wobei die praktische Anwendbarkeit im Vordergrund steht. Aktuelle öffentliche Aktivitäten in diesem Bereich haben oft noch Forschungscharakter, wie ein Blick auf den aktiven Teilnehmerkreis der jüngsten Software Product Line Conference[16] zeigt. Die industrielle Nutzbarkeit wird nur selten belegt, was u.U. auch strategische Gründe in den jeweiligen Unternehmen haben mag [FRAUN].

Funktionieren kann das oben beschriebene Verfahren nur, wenn in den Anwendungsprojekten klar ist, dass die Entwicklung der Software einerseits durch die Nutzung des Frameworks profitiert, dass sie andererseits aber auch ihren Beitrag zum Fortschritt des Frameworks beitragen muss. Dies umfasst sowohl simple Verbesserungen (Bugfixes) als auch das Beisteuern von Innovationen. Die Graphik macht deutlich, dass mit dem Rollout eines Systems die Entwicklung im Grunde noch gar nicht abgeschlossen ist, sondern sich der Extraktionsvorgang anschließt, der (mindestens zu großen Teilen) innerhalb des Projektbudgets geleistet werden muss. Diese Extraktion ist – analog zur Erbgutisolation der Gentechnik – eine anspruchsvolle und durchaus kostspielige Angelegenheit, da es sich hier um weit mehr als bloße Copy-Paste-Arbeit handelt. Hier gilt es Überzeugungsarbeit zu leisten und den Return-on-Invest nachzuweisen, was im IT-Bereich von arvato in Form einer entsprechenden ROI-Liste der Framework-Bausteine vorgenommen wird.

Voraussetzung für die Extraktion brauchbarer Bausteine ist die konsequente Auswertung von „Lessons Learned“, die bei arvato im Rahmen der Anwendungsentwicklung vom gesamten Team arbeitsbegleitend erfasst und nach Projektabschluss priorisiert und besprochen werden. Dabei handelt es sich um Erfahrungswerte sowohl im Umgang mit bestehenden als auch mit neu entwickelten Bausteinen, die helfen sollen, das Verbesserungs- und Wiederverwendungspotential aus praktischer Sicht einzuschätzen. Da nur „gesundes Erbgut“ extrahiert und ins Framework aufgenommen werden soll, sind die Beobachtungen der unmittelbar Beteiligten von besonderer Bedeutung.

---

[16] Siehe http://www.sei.cmu.edu/splc2006/

5 Das KISS-Prinzip

Wie oben bereits erwähnt, ist ein wesentliches Problem der Gentechnik ein Mangel an Beherrschbarkeit. Hauptgrund dafür ist (neben der Komplexität der Technik im Allgemeinen) die ungeheure Menge von Erbinformationen, mit denen es die Techniker zu tun haben. Selbst ehemals überzeugte Pioniere der Gentechnik wie Erwin Chargaff kommen auf Dauer zu einer kritischen Betrachtung [CHARGAFF]. Obwohl z.B. das menschliche Genom seit einigen Jahren als vollständig entschlüsselt gilt, ist nur ein Bruchteil dieser Informationen soweit verstanden, dass eine gezielte Manipulation mit einem vorhersagbaren Ergebnis überhaupt möglich ist. Ein Großteil der Erbmasse bleibt ein Rätsel, und stellt auch bei erfolgreichen Eingriffen an anderer Stelle eine schwer einzuschätzende Gefahr dar, weil die Querabhängigkeiten der einzelnen Informationen nur ungenügend einzuschätzen sind. Auch wenn die Erfolge der Gentechnik nicht von der Hand zu weisen sind, ist es an dieser Stelle fraglich, ob wir hier von einer Vorbildfunktion sprechen können. Übertragen wir dieses Problem in die Software-Entwicklung, dann findet es sich mit der bisher getroffenen Analogie tatsächlich wieder: übersteigt das für die Entwicklung verwendete Framework eine gewisse Komplexität, dann nimmt die Beherrschbarkeit ab. Selbst grundsätzlich als vorteilhaft zu wertende Eigenschaften können sich ins Gegenteil verkehren und eine vordergründige Einsparung durch erhöhte Instabilität wieder zunichte machen. Im Gegensatz zu biologischen Erbinformationen sind zumindest öffentlich entwickelte Software-Frameworks i.R. beständiger Anreicherung und Veränderung unterworfen. Es liegt daher in der Natur der Sache, dass sich keine dauerhafte Wissensstabilisierung einstellen kann.

Nun hat die Software-Entwicklung gegenüber der Gentechnik den Vorteil, in gewissen Grenzen selbst über die Komplexität des verwendeten Erbguts zu entscheiden. Es besteht also die Möglichkeit, das Anwendungs-Framework so klein zu halten, dass es auf der einen Seite ausreichende Hilfestellung bietet und auf der anderen Seite auch ausreichende Beherrschbarkeit. In der IT von arvato direct services wird aus diesem Grund konsequent das KISS-Prinzip (Keep It Small & Simple) bei der Pflege des eigenen Frameworks angewendet. Die Bausteine des Frameworks werden gar nicht erst darauf ausgelegt, eine 100%'ige Lösung darzustellen, sind aber dafür leicht durchschaubar und flexibel und so klein, dass jeder Entwickler die Chance hat, zur Not durch Studium des Codes die inneren Zusammenhänge innerhalb kurzer Zeit zu begreifen. Dieses Prinzip ist gerade für die Systemfamilienentwicklung von zentraler Bedeutung, da die benötigten Flexibilitätsdimensionen des Frameworks schwer vorhergesagt werden können. Um bei einem einfachen Bild zu bleiben: es reicht nicht aus, nur das Farb-Gen zu beherrschen, weil zuletzt immer nur rote, grüne und blaue Züchtungen gefordert wurden. Schon die nächste Anforderung kann lauten, dass sie 1, 2 oder 10 Meter groß sein müssen. Und diese Anforderungen in einer ganz anderen Dimension müssen umsetzbar sein, ohne dass zunächst jahrelange Forschung vorweg geht.

Die folgende Graphik zeigt einen ungefähren Größenvergleich zwischen dem bei arvato eingesetzten Framework und einem aus Mainstream-Komponenten aufgebauten Baukasten.

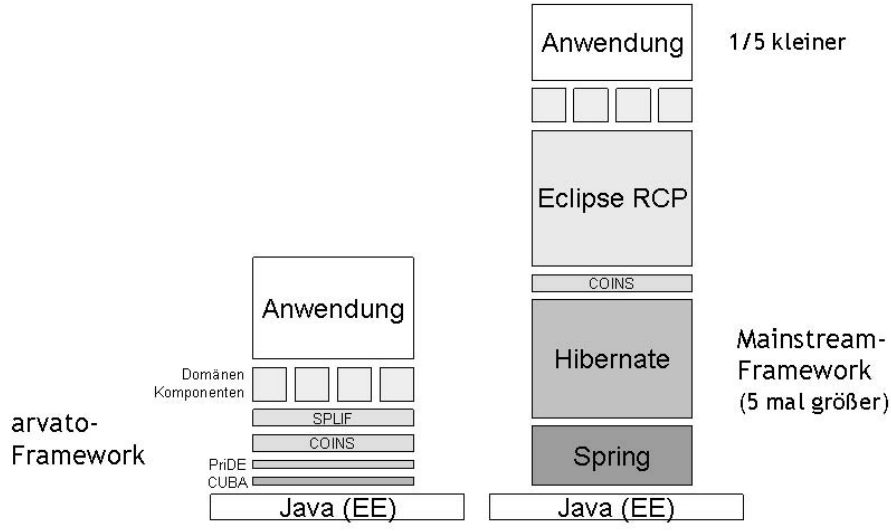

Abbildung 3: Frameworks im Größenvergleich

Als Basis für die Entwicklung der Geschäftslogik wird das OpenSource-Framework CUBA eingesetzt (dazu später noch mehr), und auch der Datenbankzugriff wird über ein OpenSource-Framework, den O/R-Mapper PriDE[17], unterstützt. Beide Frameworks haben Bibliotheken von unter 200 KByte Größe (inkl. zugehöriger Generatorwerkzeuge). Die Hilfsbibliothek COINS, sowie die Domänenkomponenten (Kundenverwaltung, Kontoführung, Adressvalidierung u.v.m) sind arvato-spezifisch und auch in einem anders gearteten Framework alternativlos. Der graphische Rahmen für Servicecenter-Applikationen wird von dem ebenfalls arvato-eigenen Framework SPLIF gestellt. In Summe ergibt sich dadurch ein Framework von weniger als 2 MByte Größe. Würde man die einzelnen Bestandteile durch Komponenten des aktuellen Mainstreams ersetzen (Hibernate[18] für den DB-Zugriff, Spring[19] als Basis für die Geschäftslogik und die Eclipse Rich Client Platform[20] als graphischen Rahmen), dann entsteht ein Framework von mindestens 5-facher Größe bei einer geschätzten Einsparung von ca. 20% in der Größe der Anwendung. Diese 20% sind jedoch größtenteils simpler, kanonischer Code, so dass die damit verbundene Aufwandsersparnis schätzungsweise bei 5-10% läge. Und auch dieser vordergründige Gewinn würde noch einmal durch Aufwände geschmälert, die durch das schwerer zu beherrschende Framework entstehen. Betrachtet man die Kosten für Erstellung und Wartung einer ganzen Herde von Systemen derselben Familie, stellt sich das KISS-Prinzip schließlich als der günstigere Weg heraus.

[17] Siehe http://pride.sourceforge.net
[18] Siehe http://www.hibernate.org/
[19] Siehe http://www.springframework.org/
[20] Siehe http://www.eclipse.org/

Insbesondere bei Themen der allgemeinen Infrastruktur (z.B. DB-Zugriff) sind häufig OpenSource-Lösungen oder Fremdprodukte verfügbar. Es macht durchaus Sinn, solche Komponenten als Bestandteile des eigenen Frameworks aufzunehmen, um hier die eigenen Entwicklungsressourcen zu schonen. Aber ein wachsames Auge ist trotzdem von Nöten: was hier zählt ist nicht ein Maximum an Features und Modernität, sondern die beste Eignung der Komponente für die Domäne und für die Einbettung in den Entwicklungsprozess und den Gesamtzusammenhang des „Erbguts". Auch fremd entwickelte Software muss vollständig verstanden und passend sein! Ein überschaubares Framework ist in diesem Zusammenhang auch deswegen von Interesse, weil es gefährliche Kopfmonopole vermeidet. Das Wissen um die Funktionsweise der Bausteine muss leicht transferierbar sein und sollte insbesondere nicht nur in den Händen externer Mitarbeiter liegen.

Im Gegensatz zu Infrastruktur-Bausteinen ist für die Komponenten der eigenen Kerndomäne üblicherweise nichts Vorgefertigtes auf dem freien Markt zu finden. Zur Bildung eines leistungsstarken Frameworks gehört also auch der Mut zur Extraktion eigener Bausteine. Frameworkbildung ist daher für die Systemfamilienentwicklung eine Kernkompetenz des Entwicklungsbereichs, wenngleich diese Aufgabe des hohen Anspruchs wegen auf wenige, konzeptionell starke Köpfe beschränkt ist. Trotzdem ist wichtig: auch diese Leute arbeiten im Normalfall in den Anwendungsprojekten und stellen kein separates Team im Elfenbeinturm dar. Die Extraktionsarbeit ist ein Vorgang, der in die Anwendungsentwicklung als nachgelagerte Phase einbezogen wird (s.o. Graphik zum Vorgehen).

6 Schneller Brüter: Die Systementwicklung

Betrachtet man die bisher getroffene Analogie, dann entspricht die Systementwicklung einer Art „Brutvorgang". Durch das Framework folgt dieser Vorgang zu gewissen Teilen bestimmten, wiederkehrenden Mustern. Die Entsprechung für auszubrütende Eier - genauer gesagt: die *Eiform* - sind die Modelle, denen eine Software-Entwicklung folgt. Solche Modelle (Denk-, Prozess-, Programmier-, Komponentenmodelle) müssen vom Framework mitgeliefert und selbst eingehalten werden, und sie müssen einfach und stabil sein. Gelingt es dabei, die Modelle über verschiedene Projekte und Anwendungsgenerationen gleich zu halten, ist damit eine wesentliche Basis für zielgerichtete und effiziente Entwicklung geschaffen. Der Aufbau nach einheitlichen und einfachen Modellen ist daher eine grundlegende Eigenschaft des bei arvato eingesetzten Frameworks. Die folgende Graphik macht dies am Beispiel der Verwendung des OpenSource-Toolkits CUBA[21] für die Entwicklung von Geschäftslogik deutlich.

[21] Siehe http://cuba.sourceforge.net

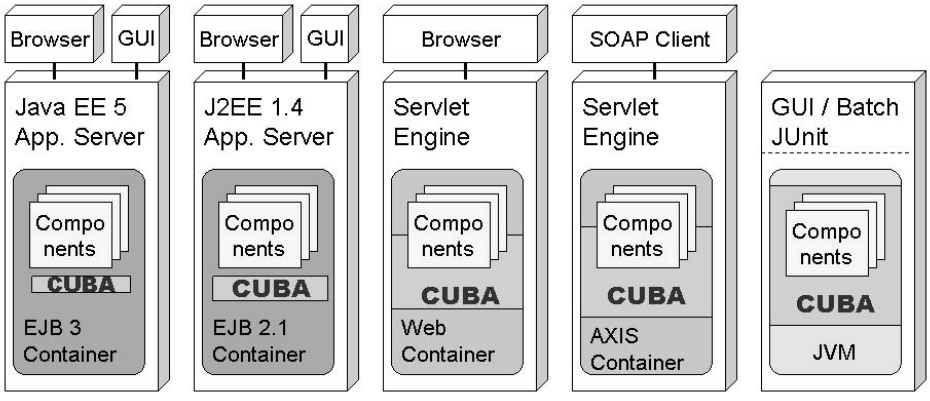

Abbildung 4: Runtime-Umgebungen für CUBA-Komponenten

Serverseitige (besser gesagt: containerseitige) Geschäftslogik wird nach den Mustern des EJB-3-Standards entwickelt, kann aber mit Hilfe des CUBA-Frameworks sowohl in EJB-3- und EJB-2.1-Containern betrieben werden als auch als AXIS Webservices oder in Stand-Alone-Applikationen. Eine dieser Stand-Alone-Umgebungen sind z.B. auch Unittests, die jederzeit einen Test der Komponenten ohne Server-Deployment möglich machen. Da also auf Basis desselben Modells jegliche aktuell benötigte Architekturen aufgebaut werden können, erfolgt die Entwicklung der Geschäftslogik grundsätzlich einheitlich nach den entsprechenden Mustern, unabhängig von der späteren Runtime-Umgebung. Diese Unabhängigkeit der Bausteine erhöht natürlich deutlich ihr Potential für Wiederverwendung in ganz anderen Kontexten. Die eventuell leichte Überdimensionierung des Modells in sehr kleinen Projekten wird bewusst in Kauf genommen, um die Einheitlichkeit zu gewährleisten.

Mir dem arvato-eigenen und java-swing-basierten GUI-Framework SPLIF für die Erstellung von Servicecenter-Applikationen wird das EJB-Modell teilweise auch auf die Client-Seite übertragen. Während die APIs hier natürlich ganz anders aussehen, wird trotzdem grundsätzlich das Konzept unabhängiger Module (sog. Teilapplikationen) verfolgt. Es gibt eine Unterscheidung zwischen „Stateful“ und „Stateless“ bei den Client-Bausteinen und dementsprechend den Vorgang der Aktivierung und Passivierung.

Was sich in der IT-Abteilung bei arvato direct services zur Zeit außerdem in der „Genforschung“ befindet ist die Schaffung eines einheitlichen Programmiermodells für Swing- und Web-Oberflächen. Dabei basiert die Web-Variante des SPLIF-Frameworks auf dem OpenSource-Framework Echo[22], das nach AJAX[23]-Verfahren arbeitet und Swing-Controls für das Web nachbildet. Die Programmierung erfolgt nach Observer-Pattern durch Anhängen von Listenern an die graphischen Controls, wie Entwickler dies bereits von Swing gewohnt sind. Die Eignung für die Aufnahme in die bereits bestehende Framework- und Modell-Landschaft steht hier im Vordergrund und weniger die Betrachtung des Mainstreams, der mit JSF[24] und Struts[25] im Web-Bereich zurzeit weitgehend einen anderen Weg verfolgt. Ein gewisser Mut zum Verzicht auf den maximalen Featureset ist auch hier wieder notwendig, denn Echo 2 bildet zwar eine gute Grundlage für die bei arvato selbst zu entwickelnden Web-Oberflächen, eine Auszeichnung für aller modernstes Webdesign ließe sich aber sicher nicht gewinnen. Auch hier spielt vor allem die Eignung für die Domäne der Systemfamilie eine Rolle und die Möglichkeit, nach gleichartigen Modellen zu arbeiten.

Erstaunlich ist die Erkenntnis, dass die konkrete Ausgestaltung des Entwicklungsprozesses eher zweitrangig ist. Aufgabe des Prozesses ist es, stabile „Brütbedingungen“ zu schaffen. Der Prozess muss bekannt sein und konsequent gelebt werden und muss (wie eingangs schon festgestellt) die Entwicklung des Frameworks mittragen. Ob aber ein agiler oder klassisch wasserfallartiger Vorgang besser geeignet ist, lässt sich aus der Analogie zur Gentechnik nicht ablesen und hat sich auch im IT-Bereich von arvato nicht als Erfolgsfaktor heraus gebildet. Der Prozess muss lediglich neben den oben genannten, eher nicht-funktionalen Aspekten, zur Organisationsstruktur des Unternehmens passen. Bei arvato ist dies ein usecase-orientierter Ansatz und ein iterativ-inkrementelles Vorgehen auf dieser Basis, um ausreichend Schnittpunkte für eine stufenweise Produktivstellung zu bieten. Das Verfahren hat sich bewährt, ist aber im Grunde wenig spektakulär und sicher nur bedingt übertragbar auf andere Unternehmen.

Interessanter Weise verfolgen die inkrementelle und agile Software-Entwicklung und das bis hierhin beschriebenen Vorgehen des Systemfamilienentwicklung bei arvato ein analoges Prinzip der schrittweisen Verbesserung (siehe z.B. [LARMAN]). Dabei bezieht sich die agile Software-Entwicklung jedoch auf Iterationen *ein und desselben* Systems, während die Systemfamilienentwicklung einen Prozess über *mehrere Systeme* hinweg betrachtet. Die Verfahren lassen sich also völlig unabhängig voneinander betrachten.

7 Chancen und Risiken

Die Gestaltung der Systemfamilienentwicklung nach den oben beschriebenen Prinzipien birgt eine ganze Reihe von Vorteilen, die für den IT-Bereich von arvato strategisch interessant sind.

---

[22] Siehe http://www.nextapp.com/platform/echo2/echo/
[23] Asynchronous JavaScript And XML, ein modernes Kommunikationsverfahren für Web-Clients und -Server
[24] Siehe http://java.sun.com/javaee/javaserverfaces/
[25] Siehe http://struts.apache.org/

- Da ein nennenswerter Teil der Entwicklung nach wiederkehrenden Mustern erfolgt, gibt es einen definierten, projektunabhängigen Einarbeitungspfad, der es erlaubt, neue Kräfte schnell einzuweisen. Dazu existieren eine Reihe von Workshops mit praktischen Übungen, die den Einstieg erleichtern, ohne dass produktive Kräfte zu stark durch die Einarbeitung neuer Kollegen gebunden werden.

- Durch die relativ kurze Einarbeitungszeit kann das Team recht kurzfristig (innerhalb von 1-2 Wochen) im Bedarfsfall verstärkt werden. Die Mannschaft wird dadurch gut skalierbar, was im Projektgeschäft ein großer Vorteil ist.

- Durch die Einheitlichkeit der Infrastruktur können Entwickler auch mit vergleichsweise geringem Aufwand zwischen Projekten wechseln und ausgetauscht werden. Insbesondere für das schwer voraussagbare Aufkommen von Änderungswünschen ist es vorteilhaft, wenn die Ressourcenverteilung möglichst wenig an bestimmte Personen geknüpft ist.

- Durch die Stringenz im Vorgehen wird eine gute Vorhersagbarkeit von Aufwänden und Kosten bei Projektkalkulationen erreicht.

- Die Einheitlichkeit der Komponenten- und Programmiermodelle ist eine gute Basis für generative Ansätze (Model-driven Software-Development). Letztere bieten momentan zwar wegen des leicht anwendbaren Frameworks wenig Geschwindigkeitsvorteile in der Entwicklung, wohl aber eine deutliche Qualitätssteigerung.

Die Risiken bei dem geschilderten Vorgehen sind eine schleichende Überalterung des Frameworks und ein Verlust an Arbeitsqualität, da das Konzept der Vereinheitlichung von den Software-Entwicklern als massive Einschränkung ihrer Kreativität empfunden werden könnte. Beiden Punkten wird durch eine Reihe explizit etablierter Maßnahmen bei arvato entgegen gewirkt:

7.1 Gezielte Innovation

Jedes Anwendungsprojekt ist gehalten, eine Innovation auszuprobieren. Dabei ist es wichtig, dass dieser experimentelle Raum tatsächlich auf *eine* Innovation beschränkt ist, um sich nicht der Gefahr der Instabilität auszusetzen. In den jüngsten Projekten wurde beispielsweise die Umstellung des Build-Managements von Make auf Ant vollzogen, ein Versuch der Entwicklung von Webservices nach WSDL-First-Ansatz und die Anbindung von Rich-Clients an das Backend über eine generierte SOAP[26]-Kommunikationsschicht. Die Innovationen werden üblicherweise in einer regelmäßig stattfindenden Software-Architektenrunde abgesprochen, um sicher zu gehen, dass ein von allen Seiten als sinnvoll erachteter Ansatz verfolgt wird.

[26] Simple Object Access Protocol, siehe http://www.w3.org/TR/soap/

7.2 Mitwirkung am Framework

Da die Arbeit am Framework einen höheren Anspruch als die Anwendungsentwicklung hat, ist die Mitwirkung daran auch nicht jedem Entwickler gleichermaßen möglich. Andererseits hat es auch die Wirkung eines „Ritterschlags“, wenn eine bis dato rein als Anwendungsentwickler tätige Person eingebunden wird. Auch hier wirkt sich wieder positiv aus, wenn das Framework möglichst klein ist und sich auf diese Weise möglichst vielen Leuten erschließt. Ein indirektes Mittel der Mitwirkung ist ein für alle zugänglicher Bugtracker für die Sammlung von Feedback zum Framework. Besonders intensiv und konstruktiv berichtende Entwickler können sich hier gleichzeitig einen Namen machen und die Fortentwicklung des Frameworks mitbestimmen.

7.3 „Spiel“-Räume

Neben den projektbezogenen Maßnahmen hat es sich bewährt, eine unabhängige Vortragsreihe ohne konkreten Projektbezug ins Leben zu rufen, in der über den Tellerrand hinaus geschaut werden kann. Wegen des hohen Anteils externer Mitarbeiter, die in dieser Reihe ebenfalls aktiv tätig sind, werden auch bereichernde Ansätze aus ganz anderen „Denkfabriken“ in die Breite getragen. Die Reihe wird von allen für alle gestaltet, trägt sich also ausschließlich über das Interesse und Engagement der Mitarbeiter. Es gilt die Regel, dass im Rahmen der Veranstaltung selbst keine Entscheidungen getroffen werden, was schon wegen der überwiegend projektfinanzierten Organisation schwierig wäre. Wohl aber dienen die Informationen als Anregung für das weiter oben beschriebene Innovationswesen.

8 Zu guter Letzt…

Die Analogiebetrachtung hat im IT-Bereich von arvato direct services geholfen, das Wesen der eigenen Systemfamilienentwicklung besser zu verstehen und einzuordnen. Interessant ist zu erwähnen, dass die Betrachtung erst im Nachhinein beim „Pattern Mining“ festgestellt wurde, als es darum ging zu ergründen, welche spezifischen Dinge im Vorgehen man als Erfolgsfaktoren verstehen und erhalten muss. Für den aktuell vielerorts hoch gehandelten Begriff der *service-orientierten Architektur* (SOA) lässt sich übrigens keine passende Analogie finden – dementsprechend kritisch und eher abwartend werden die Konzepte zur Zeit bei arvato beobachtet. Hierbei ergeben sich Querabhängigkeiten zwischen Systemen unterschiedlicher Generationen, die eine nicht tolerierbare Fortschrittseinbuße bedeuten könnten. Davon abgesehen lässt sich die Analogie aber vermutlich zukünftig auch auf andere Bereiche ausdehnen. So gibt es im Hause eine Initiative, um über die feature-orientierte Modellierung (siehe [CZARNECKI]) den Gedanken der Systemfamilie auch bis in die Geschäftsmodellierung in Schwesterabteilungen zu tragen. Auch für die Verfolgung von Test-First-Strategien lassen sich hieraus Erkenntnisse ableiten.

Literaturverzeichnis

[DAHME] Ch. Dahme, W. Hesse: Editorial - Evolutionäre und kooperative Software-Entwicklung; Informatik-Spektrum 20.1, S. 3-4 (1997)

[FRAGNER] Fragner, J./Greiner, U./Vorauer, M. (Hg.): Menschenbilder. Zur Auslöschung der anthropologischen Differenz. Schriften der Pädagogischen Akademie des Bundes in Oberösterreich Bd. 15. Linz (2003)

[CHARGAFF] Erwin Chargaff: Das Feuer des Heraklit, Skizzen aus einem Leben vor der Natur, Klett-Cotta (2002)

[POHL] Klaus Pohl, u. a: Software Product Line Engineering, Springer Verlag (2005)

[LARMAN] Craig Larmann: Agile and Iterative Development, Addison-Wesley Professional (2003)

[CZARNECKI] Krysztof Czarnecki, Ulrich W. Eisenecker: Generative Programming, Addison-Wesley Professional (2000)

[FRAUN] Fraunhofer IESE, MARKET MAKER Software AG: Pressemitteilung, http://www.iese.fhg.de/Press_Media/2003/pm_pulse-cebit2003/ (2003)