Wiederverwendung durch Frameworktechnik - vom Mythos zur Realität

Objektorientierte Programmierung im allgemeinen und Frameworktechnik im besonderen versprechen Wiederverwendung von Software im großen Stil. Dieser Artikel beschreibt grundsätzliche Eigenschaften von Frameworks und wie jene die Wiederverwendung ermöglichen. Anschließend diskutiert er die technischen und organisatorischen Randbedingungen, die einen Einfluß darauf haben, damit Wiederverwendung vom Mythos zur Realität wird.[1]

Andi Birrer
Dr. Walter R. Bischofberger
Thomas Eggenschwiler

UBILAB, Schweizerische Bankgesellschaft
Bahnhofstrasse 45, CH-8021 Zürich
E-mail: {Birrer, Bischofberger, Eggenschwiler}@ubilab.ubs.ch

1. Einleitung

Bei der Entwicklung von Anwendungen wird heute meist die gesamte Funktionalität neu entworfen, implementiert und getestet, obwohl man immer wieder auf ähnliche Anforderungen und Lösungen stößt. Dies ist verblüffend (und unbefriedigend), da sich Anwendungen desselben Bereichs in vielem sehr ähnlich sind.

Die marginale Wiederverwendung, die man heute in der Praxis beobachtet hat zwei Gründe. Erstens ist es mit herkömmlichen Programmiersprachen schwierig Softwaresysteme zu erweitern und anzupassen. Zweitens werden Entwurf und Implementierung nur mit Blick auf ein einzelnes Projekt durchgeführt.

Bei objektorientierten Sprachen stehen dank Vererbung und Polymorphismus mächtige Grundmechanismen zur Verfügung, um adaptierbare und erweiterbare Softwaresysteme zu implementieren. Will man in der Wiederverwendung den Schritt vom Mythos zur Realität machen, muß man jetzt das notwendige technische Know-How aufbauen und sinnvolle organisatorischen Randbedingungen schaffen.

Dieser Artikel beschreibt grundsätzliche Eigenschaften von Frameworks. Er zeigt auf, wie jene die Wiederverwendung ermöglichen und er diskutiert technische und organisatorische Randbedingungen.

2. Frameworks – eine grundsätzliche Betrachtung

Ein Framework besteht aus einer Menge von Objekten, die eine generische Lösung für eine Reihe verwandter Probleme implementieren. Es legt die Rollen der einzelnen Objekte und ihr Zusammenspiel fest. Damit definiert das Framework auch jene Stellen, an denen die Funktionalität erweitert und angepaßt werden kann.

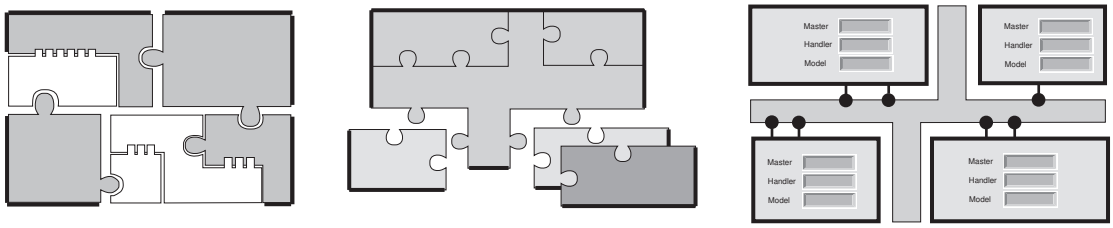
Ein Framework definiert ein Klassenteam als Wiederverwendungseinheit. Dieses erfüllt bestimmte Aufgaben und legt wichtige Entwurfsentscheidungen fest. Wieviel ein Framework von einer Anwendung eines bestimmten Gebiets abdeckt, hängt davon ab, wie weit man die benötigte Funktionalität standardisieren und implementieren kann. Dies wiederum beeinflußt die Benutzung des Frameworks. Dabei kann man drei Arten der Benutzung unterscheiden: White-Box-Wiederverwendung, Black-Box-Wiederverwendung [Joh91] und Wiederverwendung von Komponenten.

---

[1] Erschienen in OBJEKTspektrum, September/Oktober 1995

White-Box-Wiederverwendung

Erweitert man ein Framework durch das Überschreiben von Methoden, spricht man von White-Box-Wiederverwendung. Das Framework legt dabei wichtige Teile und Beziehungen fest (in Abb. 1a als dunkle Puzzle-Teile dargestellt). Diese müssen durch anwendungsspezifische Teile ergänzt werden (in Abb. 1a weiß dargestellt).
Will man ein Framework so erweitern, muß man in dieses hineinschauen und das Zusammenspiel der verschiedenen Teile verstanden haben.
Beispielsweise definiert Microsoft Foundation Class Library (MFC) eine generische Struktur für eine Windows-Anwendung, die aus den Klassen CWinApp, CDocument, CFrameWnd und CView besteht. Sehr viel Standardverhalten wird bereits im Framework implementiert, wie das Öffnen und Schließen von Fenstern, Dialoge zum Öffnen und Speichern von Dateien usw. Will man aber eine spezifische Anwendung implementieren, muß man Unterklassen von CWinApp, CDocument und CView realisieren, da die generische Anwendung nicht wissen kann, welche Art von Daten benutzt werden und wie diese zu visualisieren sind.
White-Box-Wiederverwendung ist die mächtigste Art der Wiederverwendung, da man in Unterklassen sehr flexibel Anpassungen und Erweiterungen vornehmen kann. White-Box-Wiederverwendung heißt aber programmieren, was relativ aufwendig ist und ein gutes Verständnis für die im Framework implementierten Mechanismen erfordert.
Wenn wir das offene White-Box Framework einfacher benutzbar machen und die internen Mechanismen verbergen, bewegen wir uns auf ein Black-Box Framework zu.



(a) (b) (c)

Abb. 1: White-Box (a), Black-Box (b), Komponenten mit Umgebung (c)

Black-Box-Wiederverwendung

Ein Black-Box Framework ist dadurch gekennzeichnet, daß es verschiedene direkt benutzbare Klassen zur Instanzierung und Parametrierung zur Verfügung stellt. Es kann verwendet werden, ohne daß man versteht, wie die Objekte intern zusammenspielen. Beispielsweise stellt MFC eine Menge von Dialog-Elementen zur Verfügung. Diese werden mit einem graphischen Editor konfiguriert und plaziert. Auf Ereignisse reagiert man dann in speziellen Prozeduren zur Meldungsbearteitung (Message-Handlers). Um die Darstellung, die Initialisierung und den Aufruf dieser Prozeduren muß man sich nicht mehr kümmern. Das Framework, auf dem die Realisierung einer Dialog-Box basiert, muß man damit nicht mehr durchschauen.
Die Black-Box-Wiederverwendung erlaubt nur beschränkte Anpassungen, dafür kann man sie mit geringem Aufwand durchführen. Black-Box-Frameworks erfordern keine Kenntnisse über das zur Implementierung benutzte White-Box-Framework. Normalerweise kann man aber darauf zurückgreifen, wenn man Funktionalität benötigt, die sich nicht durch Black-Box-Wiederverwendung realisieren läßt.
Der Übergang von White-Box- zu Black-Box-Wiederverwendung ist fließend. Einige Frameworks ermöglichen wesentliche Erweiterungen durch Parameterobjekte. Andere erlauben nur aus einer vorgegebenen Menge von Klassen auszuwählen. Abb. 1b zeigt ein Black-Box Framework, bei dem einzelne Teile austauschbar sind.

Wiederverwendung von Komponenten

Komponenten sind Softwareteile, die ein eigenes spezifisches Verhalten aufweisen, sich mit speziellen Editoren parametrieren und mit anderen Komponenten verknüpfen lassen. Beispiele hierfür sind Knöpfe (Buttons), Eingabefelder (InputFields), Datenbankwerkzeuge und Komponenten für die Integration von Telefoniediensten. Umgebungen zum Instanzieren und Konfigurieren von Komponenten sind VisualAge von IBM, der InterfaceBuilder von NeXT oder VisualBasic von Microsoft.
Mit Komponenteneditoren kann man sehr schnell formularorientierte Anwendungen zusammenstellen sowie komplexere Komponenten in einer Benutzungsschnittstelle plazieren und konfigurieren. Komponenten sind eine Weiterentwicklung von Black-Box Frameworks. Die Komponentenumgebung verbirgt das Zusammenspiel mehrerer Komponenten hinter definierten Schnittstellen (in Abb. 1c als Bus dargestellt). Dazu definiert sie ein Framework, das den einzelnen Teilen erlaubt zusammenzuarbeiten.
Beim Implementieren neuer Komponenten müssen die vorgegebenen Mechanismen berücksichtigen werden. Dazu kann eine beliebige Programmiersprache benutzt werden. Es ist aber einfacher Komponenten mit einem White-Box Framework zu implementieren, beispielsweise mit MFC [Bro94] für OLE 2 oder mit OpenParts für OpenDoc.

3. Anwendungsfelder für Frameworktechnik

Wir sehen drei typische Anwendungsfelder für Frameworktechnik: Den Ersatz von 4. Generationssystemen, den Einsatz einer Anwendungsumgebung und die Entwicklung eigener Frameworks zur Implementierung von Business Objects. Diese Arten werden im folgenden genauer beschrieben.

Ersatz für 4. Generationssysteme (4GS)

Als Einstiegsmöglichkeit für die Anwendung von Frameworks gibt es Umgebungen, die es ermöglichen, schnell und einfach Informationssysteme zu bauen. Typische Beispiele hierfür sind VisualWorks, NeXT-Step mit dem DB-Kit oder Visual C++. Daneben haben einige Firmen spezialisierte 4GS gebaut, die es ermöglichen eine firmenspezifische Untermenge von Informationssystemen in sehr kurzer Zeit zu realisieren. Publizierte Erfolge auf diesem Gebiet sind unter anderem das GEBOS Projekt bei der Firma RWG [RWG94] oder das ISOV-Handelsregister Projekt von IBM Schweiz [IBM94].
Diese Umgebungen erlauben es, schnell und kostengünstig Anwendungen zu bauen, die man teilweise auch mit typischen 4GS, wie Gupta Tools, Windows 4GL oder MS Access ökonomisch implementieren kann.
Es wird dabei hauptsächlich Black-Box-Wiederverwendung eingesetzt. Für die Implementierung steht eine objektorientierte Sprache zur Verfügung, im Gegensatz zu einem typischen 4GS, welches zumeist eine softwaretechnisch zweifelhafte Skriptingsprache hat. Zusätzlich ist es möglich, fehlende Komponenten mit den vorhandenen White-Box-Frameworks zu implementieren. Es ist deshalb nicht verwunderlich, daß sich diese Art der Entwicklung von Informationssystemen momentan weltweit auf einem Siegeszug befindet.

Einsatz einer Anwendungsumgebung

Frameworks wurden bisher meist für graphische Benutzungsschnittstellenteile und für ereignisorientierte, dokumentbasierte Editoren erstellt. Sie haben sich in der Folge zu Anwendungsumgebungen (Application-Frameworks) weiterentwickelt. Diese implementieren die Gemeinsamkeiten von interaktiven, dokumentbasierten Anwendungen. Sie enthalten unter anderem eine ganze Palette von Benutzungsschnittstellenteilen mit entsprechender Logik wie Dialoge für das Öffnen und Speichern von Dokumenten, Mechanismen für das Rückgängigmachen von Befehlen, für das Rollen und Splitten von Fenstern und für das Verwalten von Menübalken.

Beispiele hierfür sind MFC in der Windows Umgebung, MacApp in der Macintosh Umgebung und Visix's Galaxy oder ET++ [Wei94] in der Unix Umgebung.
Die Anwendungsumgebung definiert eine Systemumgebung. Das bedeutet, daß man sich nicht mehr um Details der zugrundeliegenden Fenster-, Datei- oder Datenbanksysteme kümmern muß. Zum Beispiel resultiert der Einsatz von Galaxy zusätzlich noch in einer automatischen Lauffähigkeit auf einer Vielzahl von Betriebs- und Fenstersystemen.
Anwendungsumgebungen erleichtern das evolutionäre Entwickeln von Softwaresystemen, da man auf einer ausgereiften Architektur aufbauen kann und deshalb weniger grundlegende Fehler macht als bei einer völligen Neuentwicklung. Außerdem ist der Code von Frameworks durch die Verwendung in mehreren Anwendungen bereits ausgiebig getestet. Dies führt zu zuverlässigeren Produkten.
Eine Anwendungsumgebung kann man aber erst dann sinnvoll nutzen, wenn man ihre Architektur versteht. Dies erfordert je nach Qualität des Entwicklungsteams mehrere Wochen bis Monate Einarbeitung, ein Grund für viele Firmen diese Technik nicht einzusetzen.
Eine Anwendungsumgebung erleichtert das Entwickeln von interaktiven Anwendungen, ohne deren Flexibilität einzuschränken. Noch besser unterstützt man die Erstellung von Anwendungen für spezielle Bereiche, indem man zusätzliche auf der Anwendungsumgebung basierende Frameworks baut. Wie oben erwähnt, kann man heute Frameworks zur Entwicklung von Informationssystemen kaufen. Spezialisierte Anwendungsframeworks, oder Business Objects sind aber noch kaum auf dem Markt. Ihre Entwicklung ist aber unter Umständen äußerst attraktiv, wie der nächste Abschnitt zeigt.

Erstellen von spezialisierten Frameworks

Seit längerer Zeit diskutiert man die strategischen Vorteile, spezialisierter Anwendungsframeworks.
Leider erfordert deren Entwicklung neben großem softwaretechnischen Wissen auch noch ein sehr tiefes Verständnis für das Anwendungsgebiet. Aber selbst wenn diese Voraussetzungen gegeben sind, muß das Framework in mehreren Iterationen aufgebaut werden [Opd90, Joh91, Bir93]. Man geht dabei so vor, daß man aufbauend auf den Erfahrungen aus der Entwicklung mehrerer Anwendungen, iterativ die generischen von den spezifischen Aspekten einer Lösung trennt. Die als generisch erkannten Teile werden dann als Framework reimplementiert bevor sie für weitere Entwicklungen zur Verfügung gestellt werden.
Ein Beispiel für ein spezialisiertes Framework ist fACTs++ (Objective Edge, http://www.objectiveEdge.com/). fACTs++ ist ein Framework für die Implementierung von Finanzanwendungen. Es stellt Abstraktionen für Finanzinstrumente, Märkte, Bewertungen u.ä. zur Verfügung, welche die Implementierung von Handels- und Portfoliomanagement-Anwendungen vereinfachen.
Der Einsatz von fACTs++ erlaubt, die einfachere Implementierung von komplexen Bewertungsmodellen. Sie können deshalb schneller eingesetzt werden und sichern damit einer Handelsbank einen Konkurrenzvorteil. Gleichzeitig dient dieses Framework als Bezugspunkt für alle Bewertungsmodelle und als Basis für die Durchsetzung von Managementmaßnahmen für die Absicherung entstehender Risiken.
Mit Cameo, einem Finanz-CAD Werkzeug aufbauend auf fACTs++, kann man Rapid Application Development (RAD, es gibt unseres Wissens keinen dt. Begriff dafür) betreiben, aber nicht auf der Basis von Benutzungsschnittstellenteilen, sondern auf der Basis von Business-Objects. Dies beinhaltet die Entwicklung von neuen Finanzinstrumenten und deren Analyse hinsichtlich Ertrag und Risiko in simulierten Märkten (Prototyping). Nach erfolgreichem Test können die neu erzeugten Business-Objects direkt in die produktive Umgebung übernommen werden.
Heute muß man spezialisierte Frameworks noch weitgehend selbst realisieren. Das ist ohne entsprechendes Know-how nicht möglich und ohne genügend Anwendungsfälle

nicht ökonomisch. Deshalb gibt es auch noch kaum Erfolgsgeschichten. In Zukunft wird sich diese Situation aber mit Sicherheit ändern. Es sind mehr Produkte auf dem Markt zu erwarten und das Wissen über Frameworks nimmt bei den eigenen Informatikern zu, so daß auch Eigenentwicklungen in Zukunft realistischer sind.

4. Frameworks – aus Managementsicht

Kosten-Nutzen Überlegungen

Entwicklung und Einsatz von Frameworks macht nur Sinn, falls dies auch ökonomischer ist, als wenn man Anwendungen völlig neu entwickelt. Sofern man die einzusetzenden Frameworks auf dem Markt erwerben kann, ist das praktisch immer der Fall. Wie schon oben diskutiert, ist das heute aber nur für wenige Gebiete möglich.
Das Entwickeln eines Frameworks kostet aufgrund unserer Erfahrungen häufig ein Vielfaches dessen, was eine spezifische Lösung mit der selben Funktionalität kostet. Daneben muß man eine Organisation unterhalten, die es ermöglicht, wiederverwendbare Komponenten zu verwalten und Informationen darüber zu verbreiten.
Auf den ersten Blick könnte man daraus schließen, daß Wiederverwendung erst dann sinnvoll ist, wenn ein Framework so oft wiederverwendet wird, daß die entstandenen Kosten durch die n-fachen Einsparungen aufgewogen werden. Dabei werden aber folgende Vorteile übersehen:

- Wiederverwendbare Lösungen senken den Wartungsaufwand, da nur noch eine statt einer Vielzahl von Varianten gewartet wird.
- Frameworks sind generell stabiler als spezifische Lösungen, da ihr mehrfacher Einsatz frühzeitig Fehler aufdeckt.
- Gutes objektorientiertes Design lernt man am besten an Beispielen. Die Wiederverwendung von qualitativ hochstehenden Komponenten hat deshalb einen wichtigen Ausbildungseffekt.
- Da die Arbeit der besten Leute mehrfach wiederverwendet wird, potenziert sich deren Leistung.

Wenn man den Aufwand der Entwicklung und Verbreitung eines Frameworks auf wenige Projekte verteilt, so rechnet sich eine Eigenentwicklung nur selten kurzfristig. Betrachtet man aber den zusätzlichen Nutzen, den die Entwicklung und der Einsatz eines Frameworks mit sich bringt, dann sieht man, daß ein solches Vorgehen mittelfristig auch ökonomisch sehr attraktiv ist.

Voraussetzungen für die Eigenentwicklung von Frameworks

Ein wichtiger Anreiz für den Bau von Frameworks besteht darin, daß wir damit eine bewährte Architektur in einer wiederverwendbaren Form zur Verfügung stellen. Wenn diese Entwicklungen erfolgreiche Lösungen für Geschäftskernfunktionalität enthalten, erreichen wir eine längerfristige Sicherung des betrieblichen Know-hows.
Die Entwicklung von eigenen Frameworks setzt allerdings ein qualifiziertes, erfahrenes Entwicklungsteam voraus. Darüber hinaus braucht es genügend Freiraum, die gemachten Anwendungserfahrungen in konsolidierte Architekturen zu überführen.
Diese Entwicklungen sind kostspielig und machen deshalb nur Sinn, wenn ein entsprechender Nutzen vorhanden ist. Neben den oben erörterten ökonomischen Gesichtspunkten muß man dabei noch die folgenden zwei k.o.-Kriterien beachten:

- Ist das Entwicklungsteam genügend erfahren, um ein Framework zu entwickeln?
- Besteht auf einem bestimmten Anwendungsgebiet genügend Erfahrung, um eine generische Architektur zu entwickeln?

Allgemeine organisatorische Maßnahmen

Um im Bereich Wiederverwendung den Schritt vom Mythos zur Realität zu machen, müssen zwei Ziele erreicht werden.

Erstens muß das Grundwissen über objektorientierte Softwareentwicklung und Frameworktechnologie aufgebaut werden und es müssen die oganisatorischen Rahmenbedingungen geschaffen werden, um diese Kompetenz zu bewahren. Technisch kompetente Leute dürfen nicht nach einem oder zwei erfolgreichen Projekten in eine Managementkarierre wegbefördert werden. Es braucht eine technische Laufbahn, mit der sie die Verantwortung für technische Entscheidungen erhalten.
Zweitens muß eine Wiederverwendungs- oder auch Framework-Kultur entstehen. Diese setzt voraus, daß sich Entwicklende eine Grundhaltung aneignen, bei der sie zuerst versuchen, Bestehendes zu verwenden, bevor sie Neues produzieren. Diese Grundeinstellung läßt sich nicht erzwingen, sondern sie muß sich langsam entwickeln.
Eine sehr wichtige Aufgabe hat dabei die informelle und halbformelle Weitergabe von Informationen. Damit erreichen wir, daß alle Mitglieder einer Abteilung oder gar einer Firma im Bilde sind, was links und rechts von ihrem Projekt läuft. Dazu dienen z.B.:
• gemeinsame Pausenräume
• News-Gruppen
• WWW-Seiten
• Projektvorträge
• Architektur-Workshops
• Entwicklerkonferenzen
Zusätzlich muß bei der Erfolgsbewertung einer Entwicklungsabteilung eine erfolgreiche Wiederverwendung gleich hoch eingeschätzt werden wie eine Neuerstellung von Code.

Organisation von Frameworkentwicklung und -einsatz

Will man selber Frameworks entwickeln, so kommt man mit einer herkömmlichen, projektorientierten Organisation nicht sehr weit. Projektorientierte Organisationen (Abb. 2a) haben die folgenden zwei Schwächen. Erstens wird der Erfolg eines Projektteams normalerweise am Produkt und nicht an der Wiederverwendbarkeit einzelner Komponenten gemessen. Zweitens gibt es zuwenig Kontakte zwischen den einzelnen Projektteams, so daß man Überschneidungen zwischen den einzelnen Projekten häufig nicht bemerkt. Stellt man sie trotzdem fest, so fehlt die Instanz, welche diese in einem Framework konsolidiert.
Wir schlagen dehalb die Bildung eines Kernteams vor, in dem talentierte Softwaretechniker mit technischen Karrierezielen zusammengefaßt werden. Die Mitglieder dieses Teams (Abb. 2b) haben die Aufgabe in Projekten mitzuarbeiten und dort die Wiederverwendungs- und Frameworkkultur zu verbreiten. Daneben brauchen sie einen fixen Freiraum, der nicht durch Projektarbeiten verplant ist. In dieser Zeit arbeiten sie an Frameworks.

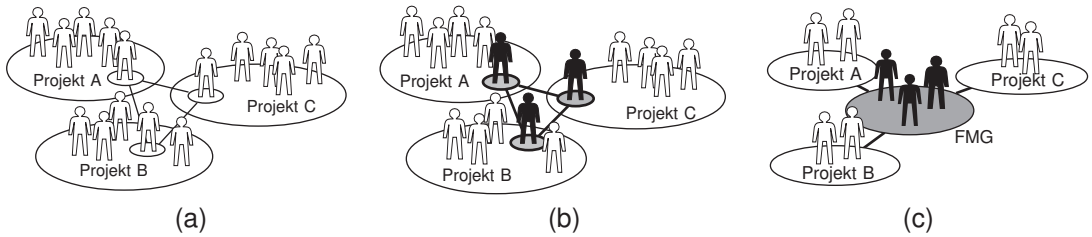


Abb. 2: Die Entwicklung von einer projektzentrierten hin zu einer wiederverwendungszentrierten Organisation.

Sobald das Kernteam eine gewisse Menge von wiederverwendbaren Komponenten aufgebaut hat, muß man ein Framework-Team schaffen, das für die Wartung und Verbreitung dieser Frameworks zuständig ist (Abb. 2c). Es ist nicht sinnvoll ein Framework-

Team zu schaffen, bevor ein Kernteam die nötigen Vorarbeiten geleistet hat. Ohne direkte Mitarbeit in den Projekten hängen die Mitglieder des Teams in der Luft und es entstehen kaum sinnvolle Frameworks.

5. Erfahrungen aus dem Einsatz in der Praxis

Wenn man untersucht, in welchen Bereichen in der Praxis erfolgreich Wiederverwendung betrieben wird, dann findet man kaum Erfolgsgeschichten. Am ehesten entstehen wiederverwendbare Bibliotheken im Bereich der Middleware. Hier stellt man Systemdienste in Form von Objekten zur Verfügung. Diese Art der Wiederverwendung ist aber noch kein wesentlicher Fortschritt gegenüber der bekannten Modultechnik. Es wird insbesondere kein Framework produziert, das einen wiederverwendbaren Entwurf darstellt.

Viele Firmen beginnen heute Frameworks als Ersatz von 4GS einzusetzen und in ein bis zwei Jahren wird es in diesem Bereich einiges zu berichten geben. Im Moment bestehen hautsächlich Projekte wie GEBOS bei der Firma RWG [RWG94] oder ISOV-Handelsregister bei IBM Schweiz [IBM94]. In diesen Projekten wurden eigene Frameworks implementiert, um damit schnell und flexibel Informationssysteme zu bauen.

Viel häufiger als Erfolgsgeschichten hört man hingegen von Firmen, bei denen die Einführung von Objekttechnologie nicht zur Entwicklung von Frameworks geführt hat. Dies liegt häufig daran, daß man die Einführung von Objekttechnologie und darauf aufbauender Wiederverwendung schlecht top-down managen kann. Ein Team braucht mehrere Monate, bis es genug Erfahrung hat, um wiederverwendbare Klassen zu schreiben. es benötigt zwei bis drei Jahre, selbst wenn es das nötige Talent hat, bis es in der Lage ist, größere Frameworks zu konstruieren. Versucht man Frameworks zu bauen, bevor das nötige Know-how vorhanden ist, dann führt das meistens zum Mißerfolg.

Wiederverwendung erreicht man nicht durch isolierte Maßnahmen wie etwa eine Person zum Klassenmanager zu bestimmen oder ein Repository einzukaufen. Einzelne abgelegte Klassen sind nur in Ausnahmefällen wiederverwendbar. Es nützt auch nicht viel mehr, alle entstehenden Artefakte in ein Repository zu legen und mit guten Werkzeugen einen einfachen Zugang zu schaffen. Es ist nämlich nicht das Problem aus hunderten von wiederverwendbaren Klassen die richtige herauszufinden, sondern aus wenigen duzenden Frameworks das richtige auszuwählen.

6. Schlußfolgerung

Der Einsatz von Frameworks als Ersatz für 4GS lohnt sich relativ schnell und hat in vielen Firmen einen Siegeszug angetreten. Die Wiederverwendung, die dabei erreicht wird, ist aber genauso begrenzt, wie die Risiken und die benötigten Umstellungen in der Entwicklungsorganisation.

Der Einsatz von Anwendungssystemen hat beträchtliche Vorteile. Die Wiederverwendung ist hoch und die Anwendungsarchitektur wird standardisiert. Man erhält einen Integrationseffekt und verbessert die Wartbarkeit. Diese Vorteile haben aber ihren nicht zu vernachläßigenden Preis. Das Engagement ist längerfristig, die Abhängigkeit von dieser Umgebung ist groß. Falls keine entsprechend ausgebildeten Leute vorhanden sind, sind die Fehlschläge vorprogrammiert.

Damit wird klar, daß die Eigenentwicklung von Frameworks nur dann unternommen werden sollte, wenn ein gut ausgebildetes, erfahrenes Entwicklungsteam vorhanden ist und sich bereits eine Wiederverwendungskultur etabliert hat. Ebenso muß den beträchtlichen Kosten ein entsprechender Nutzen gegenüber stehen. Mögliche Vorteile sind dabei gute technische Lösung und damit längerfristige Konkurrenzvorteile, verbesserte Robustheit und Flexibilität bei der Erstellung, Weiterentwicklung und Wartung. Sind die personnellen Rahmenbedingungen gegeben, ist der Einsatz von Frameworktechnik sehr gut geeignet, einer Organisation ein sauberes, tragfähiges Fundament zu liefern und ihr damit einen langfristigen Vorteil am Markt zu verschaffen.

Zusammenfaßend kann man sagen, daß frameworkbasierte Wiederverwendung im großen Rahmen nicht einfach zu realisieren ist. Aufgrund der Anwendung von Objekttechnologie darf deshalb nicht automatisch mit einer großen Rationalisierung durch Wiederverwendung gerechnet werden. Zuerst müssen die Entwickler die Technologie beherrschen und ihre Abstraktionsfähigkeiten ausbilden. Dann muß man die entsprechende Kultur und das nötige organisatorische Umfeld aufbauen. Es kann deshalb Jahre dauern, bis die Vorarbeit auch wirklich Früchte trägt. Wenn man diesen Weg aber mit der nötigen Konsequenz und Geduld beschreitet, hat man gute Chancen, daß Wiederverwendung vom Mythos zur Realität wird.

7. Referenzen

[Bir93] Birrer A, Eggenschwiler T: Frameworks in the Financial Engineering Domain: An Experience Report. in ECOOP '93 Conference Proceedings, Springer Verlag, 1993

[Bro94] Brockschmidt K: Inside OLE 2, The fast track to building powerful object-oriented applications with Windows™ Objects, Microsoft Press 1994

[Egg92] Eggenschwiler T, Gamma E: ET++ SwapsManager: Using Object Technology in the Financial Engineering Domain. in Procs. of OOPSLA '92, ACM SIGPLAN Notices, Vol. 27, No. 10

[IBM94] Ringger W, Scholz D: Erfahrungsbericht: ISOV-Handelsregister Objektspektrum, Januar/Februar 95

[Joh91] Johnson R E and Russo V F: Reusing Object-Oriented Designs, Department of Computer Science, University of Illinios at Urbana-Champaign, 1991

[Opd90] Opdyke W F and Johnson R E: “Refactoring: An Aid in Designing Application Frameworks and Evolving Object-Oriented Systems”, In SOOPPA Conference Proceedings (September 14-15, Marist College, Poughkeepsie, NY), ACM, New York, September 1990

[RWG94] Baumer D , Gryczan G , Züllighoven H: Objektorientierte Software-Entwicklung fur Banken - Methodik und Erfahrung aus einer mehrjährigen Projektpraxis, OBJEKTspektrum 3/95

[Wei94] Weinand A, Gamma E: ET++ - a Portable, Homogenous Class Library and Application Framework. Computer Science Research at UBILAB, Strategy and Projects; Proceedings of the UBILAB '94 Conference, Zurich, September 1994. Universitätsverlag Konstanz, Konstanz, 1994