

## Komponentenbasierte Anwendungsentwicklung - aber nicht ohne organisatorischen Unterbau

Autor: Dieter E. Jenz, Febr. 1999

Es ist längst bekannt, daß etwa die Hälfte bis zwei Drittel der schöpferischen Design-Leistung in einem Software-Entwicklungsprojekt ohne adäquaten Nutzen erbracht werden, da über sämtliche Anwendungen hinweg viele Gemeinsamkeiten hinsichtlich Architektur, Infrastruktur, Design, Funktionen und Code bestehen. Dieser Anteil entspricht in etwa dem Anteil an Infrastrukturfunktionen einer typischen Anwendung, der im allgemeinen etwa 50-70% beträgt, wobei der Anteil in Einzelfällen durchaus signifikant höher oder niedriger liegen kann. Es liegt deshalb nahe, eine möglichst hohe Wiederverwendbarkeit von Architektur, Design und Code zu erzielen. Verteilte Anwendungen nutzen Infrastruktur-Dienste viel intensiver als konventionelle nicht-verteilte Anwendungen. Außerdem sind Infrastruktur-Funktionen in einer verteilten Umgebung wesentlich komplexer und aufwendiger. Damit ist zusätzlicher Anlaß gegeben, die bisher üblichen Projektorganisationsformen zu überdenken.

Trotz der längst bekannten Fakten wurden nur in wenigen Unternehmen bisher wirklich ernsthafte Anstrengungen unternommen, die komponentenbasierte Anwendungsentwicklung konsequent anzugehen. Die Hinderungsgründe liegen nur zum geringen Teil in technologischen Defiziten, hauptsächlich jedoch in "entwicklungskulturellen" Fragen. Jahrzehntlang eingeschliffene Muster weisen ein beträchtliches Beharrungsvermögen auf und lassen sich nicht mit einer leichten Kursänderung korrigieren. Wenig überraschend geht die Initiative für einen Umschwung der Software-Entwicklungskultur nur in relativ wenigen Fällen vom IT-Management aus. Oft ist es ein Mitarbeiter oder ein externer Berater, der das IT-Management als Promoter davon überzeugt, die komponentenbasierte Anwendungsentwicklung zu explorieren und ein entsprechendes Organisationsprojekt aufzusetzen.

Die komponentenbasierte Anwendungsentwicklung eröffnet verlockende Perspektiven. Im Idealszenario werden Anwendungen aus vorhandenen, von Herstellern lizenzierten und/oder selbst entwickelten Komponenten, "zusammengestöpselt". Sämtliche Komponenten sind bereits vollständig getestet und verfügen über definierte Schnittstellen. Sie lassen sich in unterschiedlichem Kontext einsetzen, so beispielsweise einmal als Element einer transaktionsorientierten, ein andermal als Element einer nicht transaktionsorientierten Anwendung. Wenn festgestellt wird, daß eine bestimmte Funktionalität noch nicht abgedeckt wird, ist die Frage zu stellen, ob diese auch in anderen Anwendungen benötigt wird und somit mit einer Komponente abgedeckt werden könnte. Anschließend wäre zu klären, ob die gewünschte Komponente auf dem Komponentenmarkt verfügbar ist oder doch selbst entwickelt werden muß. Die aktive Beobachtung des Komponentenmarkts wird zu einer sinnvollen Aufgabe.

Die in größeren Unternehmen übliche projektzentrierte Organisation ist kaum auf Wiederverwendung vorhandener Software (Module, Objekt-Klassen) ausgelegt. Jedes Projekt arbeitet eigenständig auf Basis vereinbarter Methoden und Verfahren, die z. B. im Projekt-Handbuch niedergelegt sind. Zwischen den Projektteams bestehen kaum Kontakte und die wenigen meist nur informell. Das Grundübel liegt jedoch im fehlenden Interesse, wiederverwendbare Software zu erstellen. Die Begründung liegt im üblichen Zeitdruck, unter dem das Projekt steht, und den höheren Kosten für wiederverwendbare Software. Der Mehraufwand, verursacht durch zusätzliche Funktionalität, kann den Projektfertigstellungstermin gefährden. Das Projektbudget ist ohnehin nicht üppig ausgestattet und läßt keinen Spielraum, um die etwa um den Faktor 5-7 höheren Kosten unterzubringen. Wiederverwendbarkeit wird faktisch nicht bezahlt. Davon abgesehen ist meist kein Verfahren etabliert, um wiederverwendbare Software innerbetrieblich zur Nutzung anzubieten.

Andererseits machen sich die Vorteile eines hohen Wiederverwendungsgrades deutlich bemerkbar. Die wichtigsten sind:

- es wird Druck erzeugt, Anwendungsarchitektur und -infrastruktur hochgradig zu standardisieren,
- die Wartbarkeit der Software wird deutlich verbessert,
- die Entwicklungszeit und der Testaufwand sinken,
- die Qualität der entwickelten Software erreicht ein höheres Niveau.

Diese Nutzen können ohne Abkehr von der projektzentrierten Organisation nicht erschlossen werden. Eine Aufgabenzentralisierung wird erforderlich, wobei die Kosten für die zentralen Organisationseinheiten auf sämtliche Projekte umgelegt werden müssen. Wenn dies konsequent geschieht, sind nicht nur die zentralen Organisationseinheiten finanziert, sondern den Projekten wird ein notwendiger finanzieller Anreiz zur Zusammenarbeit mit den zentralen Teams gegeben. Widerstände vonseiten der Projekte gibt es genügend, überwiegend jedoch motiviert durch psychologische Hemmnisse, weniger durch rational nachvollziehbare Gründe.



Neue Projekte werden der Organisationseinheit Anwendungsentwicklung und -montage (AM) übertragen. Mit jedem Projekt wird zunächst wie gewohnt die Anforderungsanalyse durchgeführt. Das Ergebnis werden Use Cases sein. Die weiteren Aktivitäten führen zur Erarbeitung diverser Diagramme (z. B. Klassendiagramm, Sequenzdiagramm usw.), wobei zunächst nur anmodelliert wird, um einen Überblick zu gewinnen. Die Detaillierung muß lediglich ausreichen, um abgleichen zu können, ob vorhandene Frameworks, Klassen-Bibliotheken und Komponenten die ermittelten Funktionen abdecken. Da der Abgleich schon sehr frühzeitig erfolgt, wird redundanter Designaufwand für bereits vorhandene Funktionalität vermieden. Der Klärungsprozeß bringt wiederverwendbare "Teile" hervor, die dann der weiteren Anwendungsentwicklung und -montage zugrundeliegen.

Frameworks versprechen das höchste Maß an Wiederverwendbarkeit. Die Frage lautet heute nicht mehr, ob Frameworks eingesetzt werden sollen, sondern welches bzw. welche Frameworks am geeignetsten sind. Frameworks stellen Architektur und Design zur Verfügung und damit auch eine solide Grundlage für die weiteren Analyse- und Designaktivitäten. Analyse und Design werden im Software-Entwicklungsprozeß dadurch jedoch keineswegs überflüssig. Andererseits können Analyse- und Designaktivitäten nicht im luftleeren Raum geschehen, sondern müssen sich immer am Prinzip der Wiederverwendung bereits vorhandener (Zwischen)produkte orientieren. Mit der frühen Bestimmung von Frameworks wird gleichzeitig das Fehlerrisiko deutlich gesenkt. Je später Fehler erkannt werden, desto aufwendiger wird die Fehlerbeseitigung. Frameworks sind bereits getestet und haben ihre Einsatzfähigkeit nachgewiesen. Einer bekannten Faustregel zufolge betragen die Kosten für die Beseitigung eines Fehlers im Produktionsbetrieb bereits das 1000fache im Vergleich zum 50fachen in der Design"phase" (Bezugspunkt ist mit Faktor 1 die Anforderungsspezifikation). Durch Wiederverwendung wird somit auch das Risiko neuer Designfehler in Anwendungen deutlich vermindert.

In der Regel werden funktionale Lücken entdeckt, die noch nicht von wiederverwendbaren "Teilen" abgedeckt werden. Die AM wird mit der Organisationseinheit Architektur- und Framework-Management (AFM) Kontakt aufnehmen, die dann einen Mitarbeiter zur Projektunterstützung abstellt. Der AFM-Mitarbeiter wirkt für das Projekt gleichzeitig als Coach. Seine Aufgabe ist es, mit den Projektmitarbeitern zusammen abzugleichen, ob noch weitere Funktionalität vorhandener Frameworks genutzt werden kann. Andererseits wird er prüfen, ob bisher noch nicht abgedeckte Funktionalität in einer Framework-Erweiterung resultieren sollte. Er versucht, abstrahierbare Verfahren zu finden, die dann auch für andere Projekte nutzbar gemacht werden können. Sofern dies der Fall ist, kann die AFM-Einheit das Framework entsprechend erweitern. Vorhandene Frameworks werden als Packages in das Designmodell importiert.

Nachdem geklärt ist, welche Frameworks die Grundlage für die weitere Projektentwicklung bilden sollen, stellt sich als nächstes die Frage, welche bereits vorhandenen Komponenten genutzt werden können. Dazu muß bekannt sein, welche Schnittstellen von Komponenten unterstützt werden müssen. Um darüber Klarheit zu gewinnen, wird das Klassen-Diagramm weiterentwickelt und verfeinert. Sodann werden die Soll-Schnittstellen der Komponenten skizziert. Anschließend sucht die AM-Einheit zunächst im Komponenten-Repository nach passenden Komponenten. Im nächsten Schritt kann bei externen Anbietern recherchiert werden. Die Suchergebnisse werden abgeglichen, um erkennen zu können, welche Funktionalität noch nicht abgedeckt wird. Das Klassen-Diagramm dient hierzu wieder als Anhaltspunkt.

Die noch fehlende Funktionalität soll ebenfalls durch Komponenten bereitgestellt werden. Auf diese Weise wird ermöglicht, die Anwendung vollständig aus Komponenten zusammenzubauen. Die Wiederverwendbarkeit steht nicht mehr im Zentrum, wird jedoch schon alleine durch Verkapselung von Funktionalität in Komponenten begünstigt.

Die Organisationseinheit "Software-Produkt-Management" (SPM) ist als zentrale Instanz Anlaufstelle für das Architektur- und Framework-Management und die Anwendungsentwicklung (bzw. zukünftig "Anwendungs-Montage") und ist außerdem für die Verwaltung des Komponenten-Portfolio verantwortlich. Sie ist die zentrale, projektübergordnete Schaltstelle. Die Organisationseinheit ist besetzt mit Fachleuten, die einen umfassenden Überblick über das interne und externe Komponentenangebot haben. Sie übernimmt die folgenden Aufgaben:

- Entgegennahme von Komponentenentwicklungsanforderungen. Dabei kann es sich sowohl um Neuentwicklung oder Weiterentwicklung existierender Komponenten handeln. Auch wenn lediglich eine zusätzliche Komponenten-Schnittstelle geschaffen werden soll, die Implementierung jedoch unverändert bleibt, erhält die Organisationseinheit eine Entwicklungsanforderung;
- Ausarbeiten von Entwicklungsaufträgen an die Organisationseinheit "Komponentenentwicklung" (KE) nach Abstimmung mit den Organisationseinheiten AFM sowie AM. Die Komponentenspezifikation wird von AFM erarbeitet;
- Überwachen der Entwicklungsaufträge (Termin, formale Einhaltung der Auftragsbedingungen),

- Recherche bei externen Komponentenanbietern auf Grundlage der Schnittstellenspezifikationen und Beschreibungen der geforderten Funktionalität;
- Beschaffung von Komponenten nach definierten Spezifikationen bei externen Komponentenanbietern und Auslieferung an die anfordernde Organisationseinheit;
- Koordination der Komponentenverwaltung im Repository. Dazu zählt auch die Entwicklung und Pflege einer Systematik, die das Aufsuchen und Wiederfinden von Komponenten erleichtert. Testweise und im Produktionsbetrieb eingesetzte Komponenten sind logisch getrennt.

Die AFM-Organisationseinheit ist, wie bereits erwähnt, für die system- und anwendungsorientierte Infrastruktur verantwortlich. Die Infrastruktur hat Rückgratfunktion für sämtliche selbst entwickelten und in zunehmendem Umfang auch für fremdentwickelte Anwendungen. Je umfangreicher die Infrastruktur, desto weniger Funktionalität muß anwendungsspezifisch entwickelt werden. Aufgrund der großen Bedeutung der Infrastruktur als Aktivposten für die betriebliche Software-Entwicklung ist eine aufwärtskompatible Weiterentwicklung unbedingt erforderlich. Die Verantwortlichkeit der AFM-Organisationseinheit überspannt Software-Architekturen und Frameworks. Die Organisationseinheit entwickelt Frameworks kontinuierlich weiter, indem jedes Projekt auf funktionale Anforderungen hin überprüft wird, die in Zukunft auch von anderen Projekten benötigt werden. So entstehen im Zeitverlauf immer mehr und mächtigere Frameworks.

Die KE-Organisationseinheit betreibt ausschließlich Komponentenentwicklung im Auftrag des Software-Produktmanagement. Sie beschäftigt Komponentenarchitekten und -entwickler. Komponentenentwicklung unterscheidet sich von der Framework-Entwicklung in der Zielsetzung. Die Spezifikationen für die Komponentenentwicklung stammen von AFM oder AM. Letzteres wird der Fall sein, wenn es sich um eine Komponente handelt, die voraussichtlich nicht wiederverwendet wird. Da jede selbst entwickelte Komponente auch in die system- und anwendungsorientierte Infrastruktur eingebettet werden muß, können die Organisationseinheiten AFM und KE auch miteinander verschmolzen werden.

Selbstverständlich muß auch der Erfolg der komponentenbasierten Anwendungsentwicklung nachgewiesen werden können. Zu diesem Zweck müssen geeignete Metriken entwickelt werden. Eine der Meßgrößen ist z. B. die Wiederverwendungshäufigkeit von Komponenten. Da die Früchte der komponentenbasierten Anwendungsentwicklung nicht in den ersten Projekten geerntet werden können, muß ein langer Beobachtungszeitraum zugrunde gelegt werden. Die Meßgrößen werden alle drei Monate über mehrere Jahre hinweg aufgezeichnet, um Erfolg oder Mißerfolg sichtbar zu machen. Korrekturmaßnahmen können rechtzeitig initiiert werden, ohne den Langfristcharakter der komponentenbasierten Anwendungsentwicklung zu mißachten.

Es ist unbedingt erforderlich, daß alle beteiligten Organisationseinheiten dieselben Analyse- und Designmethoden nutzen und einem gemeinsamen Software-Entwicklungsprozeß folgen. Die verschiedenen Organisationseinheiten müssen auf dieselben Modellinformationen zugreifen und die von einer Organisationseinheit erzielten Ergebnisse müssen auch von anderen Organisationseinheiten weiterverarbeitbar sein. Es muß somit sichergestellt werden, daß Analyse- und Designergebnisse zwischen den Organisationseinheiten beliebig ausgetauscht werden können. Für viele Unternehmen bedeutet die geforderte einheitliche Basis noch ein Hindernis, da in der Vergangenheit unterschiedliche Werkzeuge eingesetzt wurden.

Die komponentenbasierte Software-Entwicklung kann heute durchaus gelingen. Technologien, Methodologien und Werkzeuge sind vorhanden. Auf dieser Grundlage ist es mit sehr begrenztem Risiko möglich, die Produktivität der Software-Produktion deutlich zu steigern, die erheblichen Initialinvestitionen in kurzer Zeit aufzuholen und einen signifikanten Return on investment zu erzielen. Ein Ausblick auf die absehbare technologische Entwicklung zeigt, daß das technische Risiko weiter schwindet. Der springende Punkt, die Untermauerung der komponentenbasierten Anwendungsentwicklung mit organisatorischen Maßnahmen, entscheidet jedoch letzten Endes über Erfolg oder Mißerfolg. Der wichtigste Erfolgsfaktor ist nicht die Technologie, auch nicht die Methodologie, sondern die Organisation.