

Software-Entwicklung - Produktivitätssteigerungen sind Pflicht

Autor: Dieter E. Jenz, Febr. 1999

Unternehmen stehen vielerlei Herausforderungen gegenüber, wenn sie sich im Markt behaupten und expandieren wollen. Ständige Produktivitätssteigerungen in sämtlichen Bereichen sind pure Notwendigkeit. Mit der zunehmenden Abhängigkeit der Geschäftsprozesse von der Informationstechnologie muß auch die Software-Entwicklung signifikante Produktivitätsfortschritte erzielen. Neue Anwendungssysteme müssen schneller in Produktion gebracht werden, ohne Einbußen bei der Software-Qualität hinzunehmen.

In den letzten Jahren sind, begünstigt durch mehrere Faktoren, bereits ansehnliche Produktivitätssteigerungen im Software-Entwicklungsprozeß erzielt worden. Die Software-Entwicklungsabteilungen in den Anwenderunternehmen haben jedoch im einzelnen sehr unterschiedliche Fortschritte gemacht. Es gibt Unternehmen, die den Software-Entwicklungsprozeß konsequent durchstrukturiert und große Anstrengungen unternommen haben, bereits vorhandene Software-Bausteine konsequent wiederzuverwenden. In einem bereits 1996 veröffentlichten Bericht zeigen Beispiele aus Großunternehmen, daß eine höhere Qualität (Faktor 4), kürzere Entwicklungszeiten (40% Einsparung) und eine höhere Produktivität (50%) erreicht werden konnten.

Andererseits gibt es eine sehr viel höhere Zahl von Unternehmen, die keine bzw. nur sehr geringe Produktivitätsfortschritte erzielt haben. Man arbeitet ständig unter Zeitdruck und findet keine Zeit, Grundsatzaufgaben anzugehen. Das Bild des Baumfällers, der mit einer stumpfen Axt arbeitet und mit immer höherem Einsatz immer weniger erreicht, drängt sich auf. Würde er sich die Zeit nehmen, seine Axt zu schärfen, würde sich seine Produktivität schlagartig erhöhen und der Zeitverlust wäre bald mehr als nur wettgemacht.

Während die Frage der Wiederverwendbarkeit in vielen Unternehmen noch nicht geklärt ist, eröffnen sich mit Komponententechnologien neue, mächtigere Möglichkeiten. Komponenten sind Software-Bausteine beliebiger Größe, die sich über definierte Schnittstellen mit anderen Komponenten und auch mit konventionellen Anwendungsprogrammen verständigen können. Damit Komponenten mit ihrem Umfeld zusammenwirken können bedarf es einer klar spezifizierten Infrastruktur. Ein wichtiger Teil dieser Infrastruktur sind Frameworks (Rahmenwerke). Software-Hersteller wie etwa die SAP sind seit Jahren damit beschäftigt, ihre früher monolithischen Anwendungssysteme in Anwendungsplattformen, bestehend aus komplexen Frameworks, zu überführen. Anwenderunternehmen können auf Grundlage dieser erweiterbaren Frameworks eigene Anwendungselemente "anzuflanschen". Außerdem sind in den letzten Jahren Frameworks entwickelt worden, die von vornherein neuere Technologien nutzen. Beispiele sind die auf Basis der Java-Plattform entwickelten SanFrancisco-Frameworks (IBM).

Das Anwenderunternehmen kann, vereinfacht ausgedrückt, eines der existierenden Frameworks lizenzieren und dann selbst Komponenten entwickeln oder verfügbare Komponenten von Software-Hestellern lizenzieren und damit die Fertigungstiefe verringern. Ein Komponentenmarkt wird sich etablieren, jedoch wird dies kein globaler Komponentenmarkt sein. Der Markt wird in einzelne Segmente gegliedert sein, wobei jedes Segment durch eine Framework-Familie (z. B. SanFrancisco-Frameworks) besetzt wird. Frameworks unterschiedlicher Hersteller lassen sich nicht bzw. nur schwer miteinander kombinieren, es sei denn, ein einheitliches Architekturprinzip liegt sämtlichen Frameworks zugrunde. Dies ist jedoch nur bei homogenen Framework-Familien der Fall. Dennoch können die Unternehmen eine Make and Buy-Strategie realisieren. "Alles was bereits auf dem Komponentenmarkt verfügbar ist, kaufen bzw. lizenzieren - alles was wirklich unternehmensspezifisch ist selbst entwickeln", lautet die Devise.

Der Übergang zur komponentenbasierten Anwendungsentwicklung erfordert zunächst erhebliche Initialinvestitionen und braucht Zeit. Die Abkehr von der bisherigen projektzentrierten Organisation stellt einen schwerwiegenden Eingriff in oft jahrzehntelang gewachsene Strukturen dar. Mit entsprechendem Widerstand muß gerechnet werden. Es ist durchaus möglich, daß Wiederverwendbarkeit scheitert, sofern der Umdenkprozeß "in den Köpfen" nicht gelingt. Das Ziel einer deutlichen Steigerung der Wiederverwendbarkeit ist ohne eine "Wiederverwendungs-Organisation" zum Scheitern verurteilt. Die "Kulturfragen" sollten deshalb zuerst angegangen werden, dann erst die technischen. Der zeitliche Vorlauf, bis mit der komponentenbasierten Anwendungsentwicklung tatsächlich begonnen werden kann, beträgt mindestens ein halbes Jahr. Abhängig von der unternehmensspezifischen Software-Entwicklungsverfahren kann es mehrere Jahre benötigen bis sich eine Wiederverwendungskultur etabliert hat. Vor allem ist wichtig, den Transformationsprozeß in jeder Hinsicht zu unterstützen und ihm eine hohe mentale Priorität beizumessen. Eine halbherzige Wiederverwendungsstrategie kann scheitern und einen wahren Scherbenhaufen hinterlassen. Beispiele aus der Vergangenheit gibt es zur

Genüge. Der Return on Investment, der einer konsequent durchgesetzten Wiederverwendungsstrategie folgt, lohnt jedoch auch den hohen Aufwand allemal.

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 Untermuerung 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.