

Wege zur Aufwandsreduktion im Software-Lebenszyklus

Autor: Dieter E. Jenz

Nicht die Entwicklung neuer Anwendungen bereitet den meisten Zeitaufwand und Probleme, sondern die Wartung und Weiterentwicklung vorhandener Anwendungen. Vielerlei Probleme können sich einstellen, die zum Teil möglicherweise erst im Produktionsbetrieb erkannt werden können. Gerade mit verteilten Anwendungen steigt das Fehlerpotential beträchtlich. Deshalb müssen Wege gefunden werden, den Zeitaufwand für die Durchführung von Programmänderungen möglichst gering zu halten, die Qualität nicht zu beeinträchtigen und zudem das Fehlerrisiko zu minimieren. Die objektorientierte, komponentenbasierte Software-Entwicklung eröffnet Möglichkeiten, diese Ziele gleichzeitig zu verfolgen.

Bisher ergeben sich bei der Entwicklung neuer Anwendungen oft auch Rückkopplungen auf bereits vorhandene Programme. Der Bedarf nach zusätzlicher Funktionalität erfordert neben der Entwicklung neuer auch die Weiterentwicklung bestehender Objektklassen. Selbstverständlich wäre die optimale Lösung, eine Art „Bestandsschutz“ zu realisieren und vorhandene Programme möglichst unverändert zu belassen. Die Trennung von Schnittstelle und Implementierung hilft bei diesem Bestreben ganz wesentlich. Im Prinzip können Schnittstellen und Implementierungen unabhängig voneinander geändert werden. So kommt es beispielsweise sehr häufig vor, daß die Implementierung (die Methoden) geändert wird, die Schnittstelle jedoch unverändert bleiben kann.

Objekte kommunizieren über eine oder mehrere definierte Schnittstellen, wobei eine Schnittstelle eine Liste von Methoden und Attributen umfaßt, die die von einem Objekt zur Verfügung gestellt werden und damit von anderen Objekten bzw. Anwendungen über diese Schnittstelle aufgerufen werden können. Die Signaturen der Methoden und die Attributtypen sind in der Schnittstelle mit inbegriffen. Sämtliche nicht in einer Schnittstelle verzeichneten Methoden sind „von außen“ nicht aufrufbar.

Die Frage stellt sich, über wieviele Schnittstellen ein Objekt angesprochen werden kann. Mittlerweile können alle Plattformen für verteilte Anwendungen mehrere Schnittstellen für ein Objekt unterstützen (auch eine entsprechende Spezifikation zur Erweiterung der CORBA-Plattform wurde mittlerweile verabschiedet). Es bietet sich an, diese Möglichkeit zu nutzen, um bei Änderungen an existierenden Objekten neue Schnittstellen definieren zu können. Die vorhandenen Schnittstellendefinitionen dienen als Ausgangsbasis. Die Unveränderbarkeit von Schnittstellen wird möglicherweise von der Basistechnologie ohnehin erzwungen (z. B. Microsoft COM). Sofern nachträglich Änderungsbedarf an einer Schnittstelle festgestellt wird, muß zwingend eine neue Schnittstelle erzeugt werden. Die Erzeugung einer neuen Schnittstelle erfordert keinen Eingriff in die Implementierung.

Wenn das Unveränderbarkeitsprinzip auch auf Methoden ausgedehnt wird, entsteht automatisch ein eingebautes Versionenkonzept. Eine neue Methodenversion wird stets aus der vorhergehenden abgeleitet und an die Anforderungen angepaßt. Dies bedeutet, daß nicht nur vorhandene Schnittstellen, sondern auch Methoden nachträglich nicht mehr veränderbar sind. „Neue“ und „alte“ Methoden existieren somit parallel, müssen jedoch selbstverständlich eindeutige Methodennamen besitzen. Die Eindeutigkeit kann problemlos über eine Versionsnummer hergestellt werden.

Vorhandene Anwendungen können unverändert weiterhin genutzt werden, sofern die neu hinzugefügte Funktionalität nicht benötigt wird. Dadurch bedingt wird die Anzahl möglicher Eingriffe in existierende Anwendungen bzw. Komponenten reduziert. In der Regel benötigen bei weitem nicht alle vorhandenen Anwendungen sofort die neue Funktionalität.

Aufwärtskompatibilität ist eine implizite Anforderung. Als Problem erweist sich im Zeitverlauf zweifellos das schleichende Wachstum von Schnittstellen und Methoden und die dadurch entstehende Unübersichtlichkeit. Schnittstellen und/oder Methoden können erst dann wirklich entfernt werden, wenn diese mit Sicherheit von keiner Anwendung mehr genutzt werden.

Sämtliche neuen Anwendungen nutzen grundsätzlich aktuelle Schnittstellen- und Implementierungsversionen. Zur Überarbeitung anstehende Programme können daraufhin überprüft werden, ob eine Anpassung an aktuelle Versionen sinnvoll erscheint. Im Zeitverlauf werden bestehende Anwendungen kontinuierlich aktualisiert, ohne breit angelegte „Hauruck-Aktionen“ erforderlich zu machen.

Auch Änderungen im Datenmodell machen bisher oft Code-Anpassungen erforderlich. Als Lösung, den Änderungsaufwand zu vermindern, bietet sich die Entkopplung von Anwendung und Datenmodell über eine objektori-

enterte Zugriffsschicht an, die entweder selbst entwickelt oder als Framework lizenziert werden kann (z. B. Toplink, JavaBlend). Eine objektorientierte Zugriffsschicht kann darüber hinaus gleichzeitig das objektorientierte Modell der Programmiersprache erhalten, auch wenn kein objektorientiertes DBMS eingesetzt wird. Änderungen im Anwendungs-Code lassen sich zumeist nicht vermeiden, da z. B. zusätzliche Attribute schließlich verarbeitet werden müssen. Die Auswirkungen sind jedoch wesentlich geringer als wenn Anwendungen das jeweilige Datenbank-Schema kennen. Die Weiterentwicklung der Anwendungen erfolgt nach dem oben bereits beschriebenen Muster.

Im Überblick betrachtet ergeben sich wesentliche Einsparpotentiale für die Software-Entwicklung bei gleichzeitiger deutlicher Senkung des Fehlerrisikos. Auch die Qualität der Software wird in keiner Weise negativ beeinträchtigt. Im allgemeiner zeitlicher Betrachtung liegen die Einsparpotentiale bei etwa 10-20%, wobei jedoch für den Einzelfall, das einzelne Unternehmen keine zielgenaue Aussage getroffen werden kann. Unterschwellig wird vorausgesetzt, daß den Anwendungen eine klar definierte Anwendungsarchitektur zugrunde liegt. Die Einsparpotentiale sind beachtlich und lohnen in jedem Fall, sofern noch nicht geschehen, die Einführung eines Versionenkonzepts für Schnittstellen und Implementierungen.