Alles zur BASTA! auf S. 35

Deutschland € 6,50 Österreich € 7,00 Schweiz sFr 13,40

entwickler magazin

www.entwickler-magazin.de

März/April 2.2015

Klötzchenspaß

Entwickeln Sie Ihren eigenen Minecraft-Klon

Swift

The Good, the Bad and the Ugly

Soft Skills

Basiswerte als IT-Fundament

Robotik

Einstieg in LEGO MINDSTORMS EV3

SERIE **openHAB**
Standards und Open Source

SERIE **PIC**
Mikrocontroller erweitern

4 194156 606509 02

ABAP Objects und objektorientierte Programmierung im SAP-Umfeld

Neue Chancen durch ABAP Objects?

Mit ABAP Objects hat SAP die objektorientierte Programmierung in SAP ermöglicht. Mittlerweile ist sie als Programmiermodell für Neu- und Weiterentwicklungen durch SAP gesetzt. Welche Vorteile aber auch Nachteile gehen mit der objektorientierten Programmierung einher? Was sind die Schwierigkeiten im Wandel vom prozeduralen hin zum objektorientierten Modell, und welche neuen Chancen bieten sich dadurch? Dieser Artikel gibt Antwort auf diese Fragen.

von Christian Buschmann

Ursprünglich ist ABAP eine Sprache für die prozedurale Programmierung, in der die Anwendungsentwicklung mithilfe von Funktionsbausteinen und Form-Routinen erfolgt. Mit dem Erscheinen der SAP-Version 6.4 wurde ABAP um die Eigenschaften einer objektorientierten Sprache erweitert. Der Softwareentwickler SAP entwickelte ABAP Objects kontinuierlich weiter, und mittlerweile wird das Programmiermodell in den ABAP-Programmierrichtlinien von SAP für Neu- und Weiterentwicklungen empfohlen. Das bedeutet, dass anstelle von Funktionsbausteinen und Form-Routinen nur noch Klassen und deren Methoden zu verwenden sind. In den ABAP-Programmierrichtlinien, die mittlerweile der Schlüsselwortdokumentation von SAP beiliegen, wurden hierzu entsprechende Grundsätze verfasst. Allerdings wird die prozedurale Programmierung auch weiterhin unterstützt, damit die Abwärtskompatibilität erhalten bleibt.

Auch durch SAP HANA und die neuen Möglichkeiten in der Anwendungsentwicklung verliert der Application Server ABAP nicht an Bedeutung. Die ABAP-Programmiersprache wird von SAP kontinuierlich weiterentwickelt. Gerade im Zusammenwirken mit SAP HANA hat SAP Möglichkeiten geschaffen, diese neue Technologie effizient mit ABAP zu verbinden.

Derzeit ist in vielen gewachsenen Entwicklungen aber noch das klassische ABAP vorzufinden. Der Schritt hin zu einem rein objektorientierten Programmiermodell wirft die Frage nach den Gründen auf. Welche Vorteile bieten ABAP Objects gegenüber dem klassischen ABAP?

Zunächst ist festzuhalten, dass eine gute prozedurale Architektur keinen funktionalen Nachteil gegenüber einer objektorientierten Architektur hat. Im Laufe der letzten Jahre haben sich allerdings Anforderungen an eine Softwarearchitektur ergeben, die durch den prozeduralen Ansatz nur schwer zu leisten sind. Insbesondere im Hinblick auf die Kopplung von Anwendungskomponenten und die Datenkapselung haben Funktionsgruppen Klassen gegenüber Nachteile. Die objektorientierte Programmierung stellt an sich selbst den Anspruch, die steigende Komplexität von Anwendungsarchitekturen unter Beibehaltung von Flexibilität und Wartbarkeit besser zu kontrollieren. Im Folgenden werden drei Vorteile von ABAP Objects genannt, die dies beispielhaft zeigen sollen.

Drei Vorteile von ABAP Objects

Reduktion der Komponentenkopplung: In der prozeduralen Programmierung stellen Funktionsgruppen die Schnittstellen von Anwendungskomponenten dar. Zu einer starren Kopplung zweier Anwendungskomponenten kommt es, wenn sich diese gegenseitig über Funktionsbausteine aufrufen. Es gibt zwar Möglichkeiten, diese Verbindung zu flexibilisieren, das Problem von Funktionsgruppen mit der fehlenden Trennung zwischen Schnittstellendefinition und Implementierung und der fehlenden Syntaxprüfung bei dynamischen Aufrufen bleibt aber bestehen. Erst die Interfaces von ABAP

Der Einsatz von objektorientierten Sprachelementen in ABAP hat keine funktionalen Nachteile gegenüber prozeduralen Elementen.

Objects erlauben es, eine Komponentenschnittstelle zu definieren, ohne dass ein Aufrufer fest an die dahinterliegende Implementierung gebunden ist. Dies führt zu einer reduzierten Kopplung der Komponenten. Auf diesem Weg können mehrere Implementierungen zu Komponentenschnittstellen angeboten und je nach Kontext eingesetzt werden.

Bessere Testbedingungen für Komponenten: Ein weiterer Vorteil der Interfaces ist das verbesserte Testen von Komponenten. Bei einem Komponententest mit ABAP Objects werden alle abhängigen Komponenten für den Testlauf durch eine Testimplementierung ersetzt (Mock-ups). Diese imitieren das Verhalten der Komponenten, ohne es vollständig nachzubilden. In den meisten Fällen bestehen die Rückgaben der Mock-ups nur aus Konstanten, die der Testentwickler fest vorgibt. Die zu testende Komponente durchläuft im Testlauf so die exakt gleichen Codestrecken wie im Echtlauf. Die Mock-ups sorgen dafür, dass die Komponenten mit Testdaten versorgt werden beziehungsweise keine Änderungen im System stattfinden. Ein solcher Test muss beliebig oft wiederholbar sein und darf den Systemzustand nicht beeinflussen. Im Gegensatz zum gebräuchlichen Test-Flag bei Funktionsbausteinen, bei denen der Testcode implizit mit dem Anwendungscode vermischt ist, kann bei der Lösung mit den Interfaces der Testcode explizit über Mock-ups vorgegeben werden. Der Entwickler muss den vorhandenen Anwendungscode nicht modifizieren; dieser verhält sich im Test und im Echtlauf gleich.

Datenkapselung: Die Datenkapselung ist ein weiterer Vorteil von ABAP Objects. In der klassischen Programmierung wird der Zustand der Anwendung oft in globalen Variablen verwaltet. Dies ist fehlerträchtig, da der Zugriff auf die globalen Variablen nicht geschützt ist und diese von beliebigen Programmstellen manipuliert werden können. Die Folge sind oftmals ungewünschte Seiteneffekte und Fehler in der Anwendung. Ähnliche Probleme gibt es bei dem Modulgedächtnis einer Funktionsgruppe. Hier kann es passieren, dass eine Funktionsgruppe konkurrierend von verschiedenen Programmkontexten verwendet wird. Da das Modulgedächtnis nur einmalig vorhanden ist, kommt es beim konkurrierenden Zugriff auch hier zu unerwünschten Seiteneffekten und zu Fehlern in der Anwendung. Über die Klassen im ABAP Objects erreicht der Entwickler eine saubere Datenkapselung, da die Attribute sich darin schützen lassen und so nur von den mit jeder Klasse definierten Methoden veränderbar sind. Objekte lassen sich ebenfalls mehrfach zu Klassen erzeugen, wodurch eine Klasse in unterschiedlichen Kontexten verwendbar ist, ohne dass Seiteneffekte auftreten.

Nachteile von ABAP Objects

Allerdings gibt es auch Nachteile im objektorientierten Ansatz, die hier nicht unerwähnt bleiben sollen. So geht mit dem Ziel einer hohen Wiederverwendung auch ein hoher Grad an Abstraktion einher. Durch diese Abstraktion ist es schwierig, den internen Aufbau einer Komponente schnell zu erfassen. Ein Entwickler muss darum die Abstraktion verstehen, wenn er die Komponente intern verstehen möchte. Allerdings ergeben sich aus diesem Nachteil auch wieder neue Vorteile, denn eine Codestrecke, die oft durchlaufen wird, ist gut getestet und stabil. Zudem vermeidet jedes Wiederverwenden das Kopieren von Codestrecken, was sonst spätestens bei der Wartung und Fehlerkorrektur zu Problemen führt. Der Einsatz und die Kenntnis der Entwurfsmuster der objektorientierten Programmierung helfen, die Abstraktion für Dritte verständlich zu halten.

Ein weiterer Nachteil des objektorientierten Modells ist außerdem, dass sich die Anzahl der verwendeten Entwicklungsobjekte erhöht und mehr Klassen und Interfaces erzeugt werden als Funktionsgruppen im prozeduralen Modell. Durch eine saubere Paketstruktur kann dem aber einfach und gut begegnet werden.

Übergang vom prozeduralen zum objektorientierten Modell

Der Einsatz von objektorientierten Sprachelementen in ABAP hat keine funktionalen Nachteile gegenüber prozeduralen Sprachelementen. So kann eine Funktionsgruppe ebenso gut über einer Klasse abgebildet und Form-Routinen durch Methoden gleichwertig ersetzt werden. Wer allerdings in seinem Softwaredesign Funktionsgruppen einfach durch Klassen ersetzt, hat dadurch noch nicht das objektorientierte Programmiermodell adaptiert. Die Erfahrung zeigt, dass Entwickler, die lange im prozeduralen Modell entwickelt haben, dazu tendieren, ihre Klassen ähnlich einer Funktionsgruppe zu erstellen. Aus den Funktionsbausteinen entstehen statische Methoden; das Modulgedächtnis wird über Klassenattributen und die Parameterübergabe der privaten Methoden über den globalen Attributen der Klasse abgebildet. Der Entwickler findet eine prozedurale Programmierung vor, die sich in einem objektorientierten Gewand zeigt. Die oben beschriebenen Vorteile kommen so nicht zum Tragen.

Die SOLID-Prinzipien sind belastbare Leitlinien für ein gutes objektorientiertes Design.

Das Erstellen einer guten objektorientierten wie auch einer guten prozeduralen Architektur ist keine einfache Aufgabe und erfordert Erfahrung. Wer nicht nur die objektorientierte Syntax, sondern auch das Programmiermodell übernehmen möchte, muss sich mit den Konzepten der objektorientierten Programmierung (OO) näher auseinandersetzen. Dazu gehören:

- die Kenntnisse der SOLID-Prinzipien
- das Nutzen von Entwurfsmustern
- das Einhalten der ABAP-spezifischen Programmierrichtlinien

Die SOLID-Prinzipien sind Leitlinien für ein gutes objektorientiertes Design. Hierzu zählt beispielsweise das Single-Responsibility-Prinzip, das besagt, dass eine Klasse stets eine so genannte Verantwortung haben soll. Monsterklassen, die sich für viele Subjekte und Aufgaben zuständig fühlen und so zu unerwünschten Seiteneffekten tendieren, sollen damit untersagt werden. Die Entwurfsmuster bieten dem Entwickler einen Katalog von wiederverwendbaren Lösungsschablonen für wiederkehrende Probleme. Bekannte Muster sind beispielsweise die Fabrik, die sich um die Erzeugung von Objekten kümmert, ohne dass der Aufrufer die Implementierung kennt, oder die Strategie, über die der Entwickler beliebige Algorithmen austauschbar in Programme einbinden kann.

Zuletzt seien da noch die ABAP-Programmierrichtlinien genannt. Diese formulieren mehrere Regeln, an denen sich eine gute Anwendungsarchitektur mit ABAP orientieren sollte – insbesondere auch der Hinweis, welche Unterschiede im Vergleich zu Java zu berücksichtigen sind. So sollte beispielsweise auf die in Java üblichen Datenklassen mit *getter-/setter*-Methoden im ABAP verzichtet werden, da sich diese Anforderungen mithilfe der ABAP-Datentypen besser abbilden lassen.

Aufgrund der Ausrichtung von SAP auf das objektorientierte Modell enthalten die Programmierrichtlinien umfassende Tipps zur Entwicklung mit ABAP Objects. Trotz der mittlerweile guten und reichhaltigen Dokumentation ist dennoch Erfahrung im objektorientierten Programmiermodell unerlässlich, um eine gute Architektur umzusetzen.

Qualitätssicherung als Chance von ABAP Objects

Werfen wir zum Schluss noch einen Blick auf die Chancen, die sich aus der Verwendung von ABAP Objects in Bezug auf die Qualitätssicherung von Anwendungskomponenten ergeben.

Im Bereich der Java-Entwicklung sind Tools wie JUnit für das Durchführen automatischer Tests populär und erfolgreich im Einsatz. ABAP Objects hat mit ABAP Unit ein gleichwertiges Tool. Damit schreiben die Entwickler Tests für Klassen und Komponenten, die dann manuell oder automatisiert periodisch über den SAP-Code-Inspector ausgeführt werden können. Mittels der Einbindung in diesen erhält der Entwickler laufend Kontrolle über die Korrektheit der Komponenten und Klassen. Diese Unit Tests sind nicht nur für die Laufzeit eines Projekts gültig, sondern auch dann weiter im Einsatz, wenn die Entwicklung in die Wartung übergeht. Damit stellt der Entwickler sicher, dass Veränderungen, die in der laufenden Wartung vorgenommen werden, keinen Einfluss auf bestehende Funktionalitäten haben. Werden Fehler gefunden, die ein Unit Test noch nicht abdeckt, wird hierfür ein weiterer Test hinzugefügt, der verhindert, dass der Fehler bei zukünftigen Änderungen erneut auftritt.

Ein weiterer Vorteil von Unit Tests ist, dass diese wie eine zusätzliche Dokumentation wirken. Der Testcode des Unit Tests ist ein Beispielcode, der anderen Entwicklern zeigt, wie die Komponenten aufzurufen und zu benutzen sind.

Voraussetzung für einen guten Unit Test ist, dass die Klassen sich an die oben genannten Regeln für objektorientiertes Design halten. Ein langlebiger Unit Test ist nur möglich, wenn Abhängigkeiten konfigurierbar bleiben und die zu testende Klasse eine überschaubare Größe behält.

Fazit

SAP hat das objektorientierte Modell eingeführt, um die wachsende Komplexität von Anwendungen beherrschbar zu machen. Für eine erfolgreiche objektorientierte Anwendungsarchitektur ist die Kenntnis der grundlegenden OO-Prinzipien essenziell. Unit Tests profitieren vom objektorientierten Design und erzielen einen qualitativen Mehrwert.

Das prozedurale Modell wird weiterhin in ABAP bestehen bleiben und in den älteren SAP-Modulen existieren. Aber bei den SAP-Neuentwicklungen muss sich der klassisch geschulte ABAP-Entwickler darauf einstellen, sich mit ABAP Objects auseinandersetzen zu müssen, da SAP dieses Programmiermodell weiter favorisiert. Da Individualentwicklungen vom Einsatz von ABAP Objects profitieren, sollte der jeweilige ABAP-Entwickler mit dem objektorientierten Modell vertraut sein.

Christian Buschmann, Diplom Wirtschaftsinformatiker (FH), ist seit 2005 bei der innobis AG tätig und in seiner Funktion als Senior Consultant unter anderem verantwortlich für Softwareentwicklungsprojekte im ABAP-/SAP-Umfeld. Die innobis AG ist IT- und SAP-Dienstleister für Banken und Finanzdienstleister. Das Unternehmen verfügt über zwanzig Jahre Projekterfahrung im Bereich der (Individual-)Softwareentwicklung im SAP-Umfeld.