Abap Objects und die Bedeutung der objektorientierten Programmierung im SAP-Umfeld

Die neuen Programmierrichtlinien

Mit Abap Objects hat SAP die objektorientierte Programmierung ermöglicht. Mittlerweile hat sich das Programmiermodell für Neu- und Weiterentwicklungen durchgesetzt.

Christian Buschmann, Senior Consultant bei Innobis

Welche Vorteile, aber auch Nachteile gehen mit der objektorientierten Programmierung einher? Was sind die Schwierigkeiten im Wandel vom prozeduralen Modell hin zum objektorientierten und welche neuen Chancen bieten sich? Ursprünglich ist Abap eine Sprache für die prozedurale Programmierung, in der die Anwendungsentwicklung mit Hilfe von Funktionsbausteinen und Form-Routinen erfolgt. Mit dem Erscheinen des SAP Release 6.4 wurde Abap um die Eigenschaften einer objektorientierten Sprache erweitert. SAP entwickelte Abap Objects kontinuierlich weiter und mittlerweile wird das Programmiermodell in den Abap-Programmierrichtlinien von SAP für Neuentwicklungen und Weiterentwicklungen empfohlen. Dies 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 weiterhin die prozedurale Programmierung unterstützt, damit die Abwärtskompatibilität erhalten bleibt. Auch durch 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 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 liegen in Abap Objects gegenüber dem klassischen Abap? Zunächst ist festzuhalten, dass eine gute prozedurale Architektur keinen funktionalen Nachteil gegenüber einer objektorientierten 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 in Hinblick auf die Kopplung von Anwendungskomponenten und die Datenkapselung haben Funktionsgruppen gegenüber Klassen Nachteile. Die objektorientierte Programmierung erhebt an sich selbst den Anspruch, die steigende Komplexität von Anwendungsarchitekturen besser zu kontrollieren – unter Beibehaltung von Flexibilität und Wartbarkeit. Im Folgenden werden drei Vorteile von Abap Objects genannt, die dies beispielhaft zeigen sollen.

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 von Schnittstellendefinition und Implementierung und der fehlenden Syntaxprüfung bei dynamischen Aufrufen bleibt aber bestehen. Erst die Interfaces von Abap 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 Wege können mehrere Implementierungen zu Komponentenschnittstellen angeboten und je nach Kontext eingesetzt werden.

Christian Buschmann (36), Diplom-Wirtschaftsinformatiker (FH), ist seit 2005 bei der Innobis tätig und in seiner Funktion als Senior Consultant unter anderem verantwortlich für Softwareentwicklungsprojekte im Abap/SAP-Umfeld.

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 Imitiert das Verhalten der Komponenten, ohne sie 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. Die Attribute lassen sich darin schützen und sind so nur von den mit jeder Klasse definierten Methoden veränderbar. Zu Klassen lassen sich ebenfalls mehrfach Objekte 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 die Abstraktion verstehen, wenn er die Komponente intern verstehen möchte. Allerdings ergeben sich aus dem Nachteil auch wieder Vorteile, denn eine Codestrecke, die oft durchlaufen wird, ist gut getestet und stabil. Zudem vermeidet jedes Wiederverwenden das Kopieren von Codestrecken, was 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. Außerdem erhöht sich im objektorientierten Modell die Anzahl der verwendeten Entwicklungsobjekte. Es werden mehr Klassen und Interfaces erzeugt als im prozeduralen Modell Funktionsgruppen. Durch eine saubere Paketstruktur kann dem aber einfach und gut begegnet werden.

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; es werden das Modulgedächtnis über Klassenattribute und die Parameterübergabe der privaten Methoden über die „globalen“ Attribute der Klasse abgebildet. Der Entwickler findet eine prozedurale Programmierung vor, die sich in einem objektorientierten Gewand zeigt. Die oben beschriebenen Vorteile kommen nicht zum Tragen.

Das Erstellen einer guten objektorientierten Architektur wie auch einer guten prozeduralen Architektur ist keine einfache Aufgabe und erfordert Erfahrung. Wer nicht nur die objektorientierte Syntax übernehmen möchte, sondern auch das Programmiermodell, muss sich mit den Konzepten der objektorientierten Programmierung (OO) näher auseinandersetzen. Dazu gehören die Kenntnisse der Solid-Prinzipien, das Nutzen von Entwurfsmustern und 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 sogenannte 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 in 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 mit Abap Objects

Zum Schluss noch ein 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. Darüber schreiben die Entwickler Tests für Klassen und Komponenten, wie in einem der vorherigen Abschnitte bereits ausgeführt. Diese Tests können dann manuell oder auch automatisiert periodisch über den SAP-Code-Inspector ausgeführt werden. Mittels 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 Projektes gültig, sondern auch dann weiter im Einsatz, wenn die Entwicklung in die Wartung übergeht. Damit sichert der Entwickler ab, dass Veränderungen, die in der laufenden Wartung vorgenommen werden, keinen Einfluss auf bestehende Funktionalitäten nehmen. Werden Fehler gefunden, die ein Unit-Test noch nicht abdeckt, so 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 auseinanderzusetzen, da SAP dieses Programmiermodell weiter favorisiert. Da Individualentwicklungen vom Einsatz von Abap Objects profitieren, sollte der jeweilige Entwickler mit dem objektorientierten Modell vertraut sein.

Bitte beachten Sie auch den Community-Info-Eintrag ab Seite 99

Echtzeit-Analyse von Daten

Intel läutet die nächste Phase von Big Data ein

Die Intel-Xeon-E7-v2-Prozessor-Familie liefert moderne Technologie für In-memory-Analysen zur Beschleunigung wichtiger Änderungen im Geschäftsleben. Die neuen Prozessoren sind für unternehmenskritische Anwendungen konzipiert und bieten 1,5 TB pro Sockel.

Die Prozessoren bieten neue Funktionen, um große Mengen komplexer und unterschiedlicher Daten zu verarbeiten und in Echtzeit zu analysieren. Der Markt für Big-Data-Technologien und -Services soll bis 2017 jährlich um 27 Prozent wachsen.

Erhöhte Speicherkapazität für komplexe Datenanalysen

Im Vergleich zur Vorgänger-Generation verdreifachen die Prozessoren die Speicherkapazität. So können viele Unternehmen ihre gesamte Kunden-Datenbank in den Speicher laden und analysieren. In-memory-Datenbanken platzieren und analysieren die Daten direkt im schnellen Arbeitsspeicher und sparen damit die bisher nötigen Schreib- und Lesevorgänge aus den Festplatten der Datenbank-Systeme. Solche Systeme gewinnen zunehmend an Bedeutung. Während 2012 rund zehn Prozent der mittleren und größeren Unternehmen In-memory-Analyse einsetzten, sollen es den Marktforschern von Gartner zufolge bis 2015 schon 35 Prozent sein. Die Analysten rechnen damit, dass mindestens 50 Prozent der 2000 weltgrößten Unternehmen In-memory-Computing einsetzen werden, um stärker von ERP-Investitionen (Enterprise Resource Planning) zu profitieren. „Unternehmen, die das Potenzial von Daten für ihre geschäftlichen Entscheidungen bestmöglich nutzen, haben entscheidende Wettbewerbsvorteile“, bekräftigt Diane Bryant, Senior Vice President und General Manager der Intel Data Center Group.

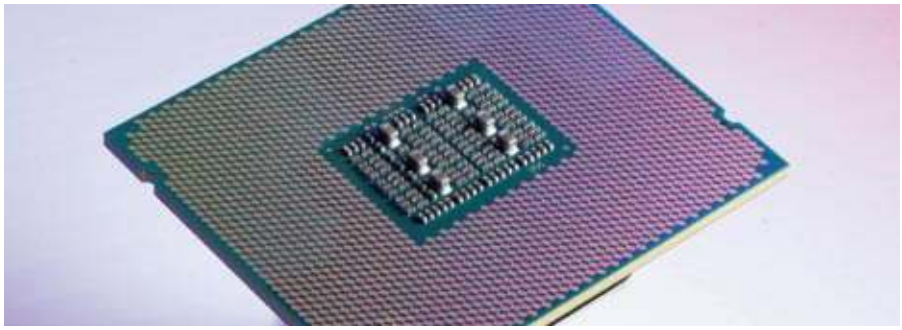
Signifikante Leistungssteigerung bei Hana

eBay verwaltet beispielsweise die Daten von mehr als 100 Millionen Nutzern weltweit mit einer riesigen Datenmenge von mehr als 50 PetaByte. Erste Tests einer Hana-In-memory-Datenbank auf Basis der neuen Xeon-Prozessoren ergaben eine signifikante Leistungssteigerung und ein besseres Verständnis bei der Analyse großer Datenmengen.

Doppelte Leistung

Dank der Bauweise für Server bis zu 32 Sockel, Konfigurationen mit bis zu 15 Kernen und bis zu 1,5 TB Speicher pro Sockel bietet der neue Prozessor im Vergleich zum Vorgänger auch die doppelte durchschnittliche Leistung. Dadurch lässt sich das Potenzial der Daten von unternehmenskritischen Anwendungen wie Business-Support-Systemen (BSS), Customer-Relationship-Management-(CRM)- und ERP-Lösungen effizienter ausschöpfen. 21 Systemhersteller stellen weltweit mehr als 40 Plattformen auf Basis der Intel-Xeon-E7-v2-Prozessor-Familie vor. Dazu gehören unter anderem Hitachi, HP, Huawei, IBM und Unisys. Auch Anbieter von Analyse-Software-Lösungen, wie SAP unterstützen die neuen Plattformen.

www.intel.com



Intel-Xeon-E7-v2-Prozessor.