DER PRAGMATISCHE WEG ZUR SOA

Die Integration vorhandener Anwendungen ist eine der größten Herausforderungen bei der Realisierung eines SOA-Projektes. Mit den entsprechenden Tools lassen sich jedoch auch in bestehenden Applikationen funktionelle Services identifizieren, herauslösen und in eine SOA integrieren. So kann bewährte Business-Logik mit neuer Technologie genutzt und erweitert werden, ohne dass die Funktionalitäten neu programmiert werden müssen. Ein pragmatischer Ansatz, mit dem sich die IT wieder Handlungsfreiheit verschaffen kann.

▸ der autor

Rolf Becking
(E-Mail: marketing.de@microfocus.com)
ist Diplom-Mathematiker und Senior Technical Account Manager bei Micro Focus. Er verfügt über 23 Jahre Berufs-Erfahrung in der Software-Industrie und hat dabei in den Bereichen Software-Entwicklung, Support, Training, Presales und Consulting gearbeitet. Seine Themenschwerpunkte sind COBOL, Legacy-Anwendungen und deren Migration sowie die Anbindung an SOA-Architekturen.

Immer wieder kommen beunruhigende Nachrichten aus der SOA-Welt. Obwohl sich das Konzept seit Jahren allseitiger Zustimmung erfreut, bleibt die Realisierung weiter hinter den Erwartungen zurück. Die praktische Umsetzung kommt nicht so recht voran, vor allem sind größere, strategische Projekte, die erfolgreich abgeschlossen wurden, noch immer selten. Die großen Vorteile, die sich die IT von SOA verspricht – Flexibilität, Alignment, Transparenz, Business-Agilität und nicht zuletzt die universelle Wiederverwendbarkeit – stehen im IT-Alltag jedenfalls meist noch aus.

Dabei hat die offenkundige Diskrepanz zwischen Wollen und Tun eine einfache Ursache: die IT-Abteilungen der Unternehmen haben auch noch andere Sorgen. Sie müssen in erster Linie den IT-Betrieb aufrechterhalten und für die Anpassung an aktuelle Anforderungen sorgen. Für die Befassung mit den strukturellen Fragen einer neuen Applikations-Architektur bleibt da nur wenig Zeit, und vor allem aber wenig Budget. So sind vom IT-Budget in der Regel bereits allein für den laufenden IT-Betrieb zwischen 75 und 95 Prozent verplant. Entsprechend gering ist der Handlungsspielraum der IT.

Andererseits kann die IT auch nicht alles beim Alten lassen und bis in alle Ewigkeit mit den vorhandenen Applikationen arbeiten. Unternehmenskritisch hin oder her – die großen Legacy-Systeme, in COBOL oder PL/1 geschrieben und auf dem Mainframe betrieben, passen einfach nicht mehr in die moderne IT-Landschaft: Sie sind unflexibel, kompliziert, schwer zu erweitern und vor allem zu teuer. Oft handelt es sich aber um abgeschlossene Systeme, auf die man nicht einfach mit einem Web-Interface oder mit einem Linux-Server zugreifen kann. Auch relativ simple Anforderungen wie die Synchronisation mit PDAs, Datenintegration mit Windows-Applikationen oder XML-Output lassen sich auf dieser Basis oft nur schwer realisieren.

SOA ganz pragmatisch

Aber zum Glück gibt es SOA. Das Dilemma „funktionelle Leistungsfähigkeit und Kostenaspekte versus Flexibilität und Offenheit" liefert den Ansatzpunkt für einen pragmatischen Weg zur SOA. Unternehmen müssen nämlich nicht auf den einen großen Wurf warten, auf die große Neugestaltung ihrer Anwendungslandschaft, sondern können auch ganz bestimmte, begrenzte Aufgaben mit den Möglichkeiten dieser Architektur lösen. Dieser Pragmatismus ist dem Grundgedanken von SOA übrigens keineswegs fremd, sondern geradezu in deren „DNA" angelegt. Da eine SOA per Definition technologie-, sprach- und systemunabhängig ist, lässt sich damit auch eine vorhandene COBOL-Businesslogik, sobald sie als Service gestaltet wird, ohne weiteres in einem neuen, offenen Kontext betreiben. Der SOA ist es egal, ob die Services mit Java oder COBOL geschrieben wurden. Also muss man nur die bestehenden Algorithmen als Services verpacken und über definierte Schnittstellen anderen Systemen zur Nutzung zur Verfügung stellen.

Dieses „Nur" bedarf freilich der Interpretation. Grundsätzlich sollte man sich bei solchen Projekten immer bewusst sein, dass die SOA auf einer gänzlich anderen „Software-Philosophie" beruht als die vorhandenen Enterprise-Applikationen, die von Haus aus eben *nicht* „SOA-fähig" sind. Selbst wenn sie durchgehend aus sauber strukturierten, modular aufgebauten Programmen bestehen, liegt die Herausforderung darin, in diesen Programmen jene Bestandteile zu identifizieren, die als funktionelle Services genutzt werden können, und sie anschließend herauszulösen. Eine sehr sorgfältige Analyse der bestehenden Anwendung ist dabei unerlässlich. In größeren Applikation kann man durchaus ein paar Tausend Module finden, die zwar funktionell für eine Verwendung als Service in Frage kommen, die aber technisch (noch) nicht den Kriterien eines Services entsprechen.

Mittlerweile sind kommerzielle Lösungen verfügbar, mit denen sich die richtigen Module in einer vorhandenen Enterprise-Applikation identifizieren und herauslösen lassen. So können Unternehmen, die einen IBM-Mainframe (System z) als Host-System verwenden, mit SOA Express von Micro Focus die Erstellung von Services aus bestehenden COBOL-Programmen in hohem Maße automatisieren. Die betreffenden Applikationen verwenden für die Dialoganwendungen die Transaktionssysteme CICS oder IMS. SOA Express generiert aus diesen Host-Transaktionen Services, ohne dabei Veränderungen an der jeweiligen Host-Applikation vorzunehmen. Unternehmen können damit genau das vermeiden, was sie am meisten fürchten: Eingriffe in die funktionellen Aspekte funktionierender Applikationen.

Ausgangspunkt für die Definition eines Service ist die Host-Applikation. Ein Service kann generiert werden auf Basis:

- einer Bildschirmmaske der Host-Anwendung (BMS für CICS, MFS für IMS)
- einer Datenstruktur, die jedem Transaktionsprogramm zur Verfügung ge-

stellt wird und den Ablauf des Programms steuert (COMMAREA bei CICS, Scratch Pad Area bei IMS)
- eines protokollierten Anwender-Dialogs, der sich über mehrere Masken und auch mehrere Transaktionen erstrecken kann.

Auf diese Weise lassen sich zunächst die Funktionalität und der Umfang des zu erstellenden Service auf dem Mainframe definieren. Der Service besteht aus einer oder mehreren Transaktionen, die ihre Eingabedaten aus der entsprechenden Maske oder dem betreffenden Datenbereich erhalten. Diese Eingabedaten und die von der Transaktion zurückgelieferten Ausgabedaten stellen die Basis für die Service-Schnittstelle dar, die nach außen exponiert wird. Dabei ist es durchaus möglich, den Umfang der Datenübernahme flexibel an die Anforderungen des zu erstellenden Services anzupassen. So können zum Beispiel bestimmte Felder der Maske an der Service-Schnittstelle oder einzelne Felder, die konstante Informationen enthalten, ausgenommen werden, so dass die Daten im Service-Interface nicht auftauchen. Dieses Vorgehen ermöglicht es, die Funktionalität der Transaktion für die Verwendung als Service einzuschränken, ohne die Transaktion selbst zu verändern.

Services automatisch herstellen

Bei SOA Express erfolgt die Modellierung des Service-Interfaces mit Hilfe eines grafischen Tools. Wurde zur Erstellung des Services ein ganzer Dialog mit mehreren Bildschirmen aufgezeichnet, so kann mit diesem Tool das Service-Interface definiert und anschließend festgelegt werden, welche Informationen von einer Transaktion zur anderen weitergegeben werden, ohne dass sie an der Service-Schnittstelle nach außen sichtbar werden. Dadurch entsteht eine interne Verarbeitungslogik innerhalb des Services, mit der auch Feldinhalte abgefragt und auf einen bestimmten Inhalt überprüft werden können. SOA Express generiert immer zwei Module, ein Host-Modul in COBOL und ein Modul in der Sprache des Application-Servers, auf dem der Service läuft, also entweder Java oder C# für die .NET-Welt.

Die Funktionen des Services sind in den generierten Java- beziehungsweise C#-Modulen als Methoden enthalten. Sie können mit der entsprechenden Service-Schnittstelle von außen aufgerufen werden. In den Service-Methoden gibt es Aufrufe an eine Middleware, die die an der Service-Schnittstelle entgegengenommenen Daten zum Mainframe überträgt. Bezüglich der Middleware kann der Anwender zwischen MQ-Series und ECI (External Call Interface) wählen. Als Partner-Applikation steht auf der Mainframe-Seite das generierte COBOL-Modul zur Verfügung, welches auf dem Host compiliert und in das Transaktionssystem geladen werden muss.

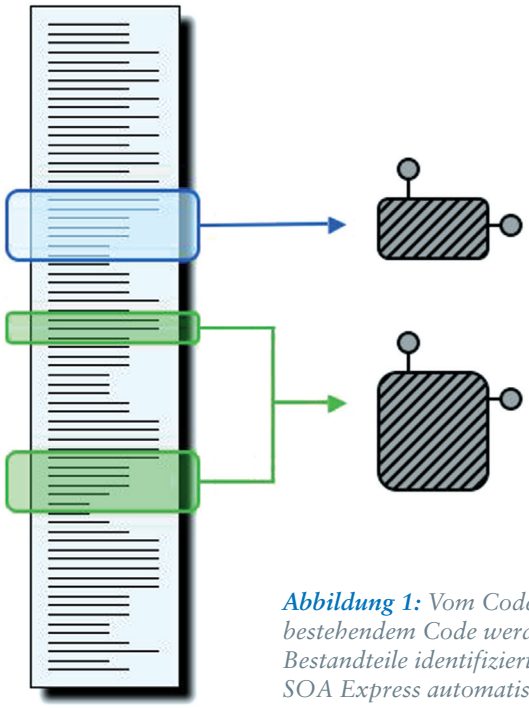

Abbildung 1: Vom Code zum Service: In bestehendem Code werden Service-fähige Bestandteile identifiziert, isoliert und von SOA Express automatisch generiert.

Zur Laufzeit nimmt das COBOL-Modul die übertragenen Daten entgegen und steuert dann die Host-Transaktionen des Services. Wurde eine ganze Sequenz von Maskendialogen protokolliert, so enthält das COBOL-Programm die entsprechende Logik, um all diese Transaktionen in der entsprechenden Reihenfolge aufzurufen, sie mit ihren Eingabedaten zu versorgen und die Antwortdaten entgegenzunehmen.

Für die Generierung der Services wird ausschließlich standardisierter Code generiert (COBOL, Java, C#), der als Source-Code zur Verfügung steht, aber nicht gepflegt werden muss. Dadurch entsteht kein zusätzlicher Aufwand für die Pflege der neuen Software. Der Aufwand beschränkt sich auf die Pflege der Service-Schnittstellen, der internen Logik und der User Interfaces, was die Kosten eines pragmatischen SOA-Projekts erheblich reduziert. Insbesondere aber sind an der Mainframe-Anwendung selbst keinerlei Veränderungen erforderlich, womit sich für den klassischen Mainframe-Anwender nichts ändert. Seine Applikation kann parallel weiter betrieben werden. Auf diese Weise lässt sich das Risiko einer SOA-Adaption beträchtlich vermindern.

Der wichtigste Aspekt für Softwareentwickler liegt sicher darin, dass die Funktionalitäten nicht neu programmiert werden müssen. Das spart teure Entwickler-Ressourcen. Die Kommunikation zwischen traditionellen Host-Programmierern und Software-Entwicklern der neuen Generation, die Java oder C# programmieren, ist wegen unterschiedlicher Denkweisen und Konzepte oft schwierig. Daher ist es nicht einfach, einem Java-Programmierer zu erklären, was die Mainframe-COBOL-CICS-Anwendung von ihm erwartet. Der COBOL-Programmierer versteht wiederum die Problematik der objektorientierten Web-Programmierung nicht. Beide Entwickler-Typen können nur zusammen arbeiten, wenn jemand, der beide Welten versteht, eine klar definierte Schnittstelle vorgibt und diese in beiden Programmiersprachen beschreibt. SOA Express bringt diese Leistung bereits mit.

Es versteht sich, dass man durch den Einsatz eines schlauen Tools nicht eine komplett neue Anwendungsarchitektur erhält. Aber man kann über Services die Logik einer sonst nicht zugänglichen Applikation nutzen. Auf diese Weise gewinnt man Flexibilität und Agilität, man spart Kosten und erhöht die Transparenz der Anwendungen. Vor allem gewinnt die IT durch diesen pragmatischen Ansatz wieder Handlungsfreiheit. ■