

Aus Standards gebaut: Enterprise Service Bus

Pendelverkehr



Stefan Krieger, Carsten Lill

Analysten haben eine neue Softwarekategorie ausgemacht, die den etablierten EAI-Anbietern Kopfschmerzen bereiten. könnte Die entsprechenden Produkte basieren auf Standards und versprechen Flexibilität, Herstellerunabhängigkeit und niedrige Kosten.

Alle Integrationsprojekte beginnen mit dem selben Szenario: Über die Jahre hat das Unternehmen zahlreiche Anwendungen eingeführt, die ihre Arbeit auf unterschiedlichen Plattformen verrichten und sich untereinander kaum verstehen. In der Regel spielt mindestens ein ERP-System dabei mit.

Da in diesen Programmen jahrelange Arbeit und Erfahrung steckt, sind sie ausgereift und laufen stabil. Eine vollständige Neuimplementierung auf Basis einer moderneren Technik ist deshalb oft nicht wirtschaftlich. Was soll man also tun, wenn eine solche Hinterlas-

senschaft, etwas abwertend auch gern Legacy genannt, genau die Funktionen enthält, die man für das geplante strategische Kundenportalprojekt benötigt?

Die Lösung scheint einfach: Über einen Adapter wird die Anwendung an eine Middleware angeschlossen. Der Adapter versucht eine möglichst serviceorientierte Sicht auf die Altanwendung bereitzustellen. Die Middleware bildet quasi die Adern, durch welche die Daten zwischen den Programmen fließen.

Damit unterschiedliche Anwendungen Informationen austauschen können, benötigen

sie einen universellen Dolmetscher, eine Transformation Engine, die beispielsweise Cobol-Datentypen, Textdaten oder XML-Dokumente kreuzweise übersetzen kann. Jetzt braucht man noch ein Werkzeug, um die Regeln zu definieren, wie und in welcher Reihenfolge die von den Adaptern veröffentlichten Services aufgerufen werden sollen, und schon wäre die EAI-Lösung fertig.

Leider ist das Ganze in der Praxis nicht einfach umzusetzen. Die Beteiligten benötigen reichlich Spezialwissen über Adapterentwicklung, die Fremdsysteme und die Middleware. Dass mit solchen Kenntnissen Geld zu verdienen ist, haben Anbieter wie IBM, Tibco oder SeeBeyond erkannt und deshalb Lösungen für Integrationsprobleme im Portfolio. Zunächst integrierte man eine proprietäre Middleware, erfand eine Schnittstellenbeschreibung und baute eine eigene Transformation Engine. Die Adapter funktionieren nur, wenn die übrigen Komponenten ebenfalls vorhanden waren.

Kunden betrachten die daraus resultierende Abhängigkeit von einem Hersteller mit gemischten Gefühlen. Manchem Anwender sind nach getaner Anfangsinvestition in eine Technik die Argumente bei weiteren Preisverhandlungen ausgegangen, etwa beim Kauf zusätzlicher Adapter. So entstand der Eindruck, die Einführung einer EAI-Lösung sei langwierig, teuer und nur etwas für Banken oder Versicherungen.

Teuer und proprietär: EAI-Produkte

Als interessante Alternative empfiehlt sich der Enterprise Service Bus (ESB). Er verspricht, dieselben Probleme wie traditionelle EAI-Produkte zu lösen, nur einfacher, schneller und damit auch günstiger. Die Begründung ist einfach: Der ESB setzt bei all seinen Kernfunktionen auf etablierte Standards und soll damit die Kombination von Produkten verschiedener Hersteller ermöglichen. Anwender profitieren davon in doppelter Hinsicht. Erstens entsteht, da sich einzelne Komponenten austauschen lassen und dadurch vergleichbar werden, Druck auf die Preise.

Zweitens erhöhen standardisierte Techniken die Wahrscheinlichkeit, dass sich vorhandenes Know-how auch für die ESB-Lösung gebrauchen lässt. Und falls das Unternehmen neue Fähigkeiten bei der Belegschaft aufbauen muss, kann es diese vielfältiger einsetzen. Die Einführung eines ESB soll insgesamt einfacher sein als die einer herkömmlichen EAI-Lösung und nicht zwingend die Mitarbeit teurer Spezialisten erfordern.

Der logische Aufbau eines Enterprise Service Bus unterscheidet sich nicht wesentlich von dem traditioneller Systeme; alle beteiligten Anwendungen schließt man mittels Adapter an eine asynchrone Middleware an, die die Daten zwischen den Pro-



- Ein Enterprise Service Bus ist ein neuer Ansatz für die Integration heterogener Softwarelandschaften.
- Im Unterschied zu traditionellen EAI-Lösungen arbeitet der ESB auf Basis standardisierter Middleware-Techniken und XML.
- Analysten erwarten den Durchbruch der ESBs spätestens mit dem Markteintritt von IBM und Microsoft.
- Einige Open-Source-Produkte in Kombination mit Application Servern bilden eine interessante Alternative zu den kommerziellen Angeboten.

grammen transportiert (Abbildung 1). Feiner, aber entscheidender Unterschied ist, dass die wesentlichen Funktionen des ESBs – im Gegensatz zu traditionellen EAI-Produkten – auf folgenden standardisierten Techniken basieren, die sich den charakteristischen Eigenschaften und Aufgabengebieten eines ESB zuordnen lassen:

Kommunikation/Datentransport: Seit 1998 ist der Java Message Service (JMS) der De-facto-Standard für Messaging-Systeme. Jede relevante nachrichtenorientierten Middleware, etwa IBM Websphere MQ, unterstützt diese Spezifikation. Eine wesentliche Eigenschaft von JMS ist die garantierte Zustellung. Der Gebrauch dieses Services im ESB erlaubt, jeden Messaging Provider für den Datentransport zu verwenden. Für die zwingend notwendige Entkopplung der Anwendungen untereinander sorgt der asynchrone Datentransport. Die Nachrichten werden im ESB zwischengespeichert und erst zugestellt, wenn die Anwendung sie bearbeiten kann.

Zusammenarbeit ohne Grenzen

Konnektivität/Adapter: In diesem Bereich haben sich Web Services, das Simple Object Access Protokoll (SOAP) sowie die Java Connector Architecture (JCA) der Java Enterprise Edition (J2EE) durchgesetzt. Neuere Anwendungen veröffentlichen ihre Dienste

zum Teil schon als Web Services. Für ältere Anwendungen hat sich ein Markt von JCA-konformen Resource-Adaptoren gebildet. Letztere sind eine Art Treiber mit einer einheitlichen Programmierschnittstelle für beliebige Backend-Systeme. Die Details der Kommunikation bleiben innerhalb des Adapters verborgen. ESBs setzen auf diese Standards und bieten Services an, die SOAP-Aufrufe erzeugen und JCA-Adapter nutzen können. Enterprise Java Beans oder .Net-Komponenten lassen sich selbstverständlich einbinden.

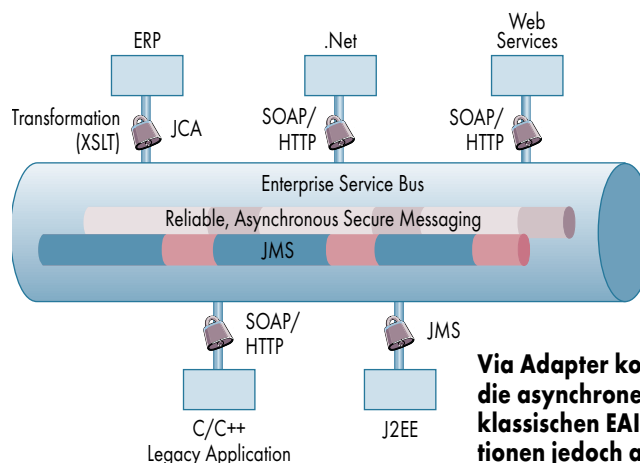
Transformation: In den letzten Jahren hat sich XSLT (Extensible Stylesheet Language Transformation) unter Herstellern und Anwendern verbreitet. Auch ESBs verfügen über eine Transformation Engine auf Basis von XML und XSLT. Mittels XSLT werden XML-Dokumente von einem XML-Schema in ein anderes überführt. Man verwendet XSLT, um die unterschiedlichen Datenstrukturen zwi-

schen Sender und Empfänger abzugleichen oder um XML-Dokumente in andere Formate umzuwandeln. Mittlerweile gibt es eine ganze Reihe von grafischen Werkzeugen, die den Umgang mit dieser Technik erleichtern.

Serviceorientierte Architektur (SOA): Während die Daten durch den ESB wandern, können spezialisierte Services sie bearbeiten. Idealerweise sind dafür vorgefertigte Bausteine vorhanden, wahrscheinlich muss man jedoch individuelle Services erstellen. Der ESB bringt Werkzeuge für deren Entwicklung mit. Standards existieren in diesem Bereich noch nicht, was bedeutet, dass Services nicht zwischen verschiedenen ESB-Implementierungen portabel sind. Wer dieses Problem umgehen möchte, sollte die eigentliche Servicelogik in eine Enterprise Java Bean oder .Net-Komponente auslagern.

Portabilität: Hier führt kaum ein Weg an Java vorbei. Damit er möglichst viele Plattformen unterstützt, ist ein ESB typischerweise in dieser Sprache realisiert. Microsofts ESB Indigo (alle Links im Kasten „Infos im Web“) wird als Ausnahme diese Regel wohl bestätigen.

Sicherheit: Dafür sorgen im ESB Mechanismen wie Secure Socket Layer (SSL) und das Light-weight Directory Access Protocol (LDAP). SSL sichert die Netzwerkübertragung, es verschlüsselt



Via Adapter kommunizieren die Anwendungen die asynchrone Middleware. Im Gegensatz zu klassischen EAI-Produkten basieren die Funktionen jedoch auf etablierten Standards (Abb. 1).

ANBIETER VON ESB-LÖSUNGEN

Fiorano Software	www.fiorano.com
Iona Technologies	www.iona.com
Kenamea	www.kenamea.com
Knownow	www.knownow.com
Polarlake	www.polarlake.com
Sonic Software	www.sonicsoftware.com
Spiritsoft	www.spiritsoft.com
Webv2	www.webv2.com

die Nachrichten, prüft die Integrität und authentifiziert die Kommunikationspartner. Über LDAP lassen sich Verzeichnisdienste wie das Microsoft Active Directory zur Authentifizierung und Autorisierung von Benutzern an den ESB anbinden.

Goldene Zeiten prognostiziert

Als Produktkategorie sind die ESBs noch neu und es tummeln sich vor allem kleinere Spezialanbieter in diesem Markt. Laut Gartner stehen die Großen der Branche aber in den Startlöchern. Noch in diesem Jahr wollen sie sich mit ersten Produkten ein Stück vom Kuchen abschneiden. IBM beispielsweise will einen Services Integration Bus vorstellen, der als weiteres Mitglied die Websphere-Business-Integration-Familie bereichern soll. Rivale Microsoft möchte seine ESB-Lösung Indigo, eine servicebasierte Infrastruktur, als Bestandteil des Windows-XP-Nachfolgers Longhorn ausliefern. Indigo arbeitet mit den hauseigenen Standards COM+, MSMQ (Microsoft Message Queues) und ASP.Net.

Gartner erwartet in den nächsten Jahren einen Siegeszug der ESBs. So sollen diese bis zum Jahre 2007 die traditionelle Middleware – zumindest in neu entwickelten An-

Digitale Rohrpost

Ein mittelständisches Unternehmen mit mehreren Niederlassungen wollte den internen Transport von Rechnungsbelegen aus über zehn Anwendungen zum zentralen SAP-System neu strukturieren. Die bisherige Lösung bestand aus PL/1- und Java-Programmen und verursachte mittlerweile hohen Wartungsaufwand.

Zukünftig möchte man vermehrt Anwendungen auf Basis der J2EE-Technik erstellen. Es bot sich an, in einem ersten Projekt eine Integrationsplattform aufzubauen und den Belegfluss anschließend auf dieser Basis neu zu implementieren. Idee war, die Open-Source-Projekte Babeldoc Universal Document Processor und den Quartz Enterprise Job Scheduler (siehe Kasten „Infos im Web“) in einen Application Server zu integrieren und so einen Enterprise Service Bus zu schaffen.

Das Framework Babeldoc übernimmt dabei die Hauptaufgabe. Es verarbeitet beliebig strukturierte Dokumente in Pipelines, die ihre Arbeit in einer Abfolge so genannter Stages erledigen. Für die Integration in die J2EE-Umge-

bung sorgt eine vorgeschaltete Message Driven Bean und so wird aus einer Pipeline ein Enterprise Service.

Weil jede Pipeline in einer eigenen Transaktion abläuft, können abgebrochene Prozesse zu einem späteren Zeitpunkt zu Ende geführt werden, nachdem die Fehlerursache beseitigt ist. Dass die Datenversorgung für SAP auch im Batch-Betrieb erfolgen kann, garantiert der Open Source Scheduler Quartz, der als Taktgeber die Verarbeitung zeitgesteuert startet. Dazu kommt noch eine Webkonsole für Administration und Tracking.

Für die eigentliche Behandlung der Buchungsbelege haben die Verantwortlichen mehrere Pipelines im ESB angelegt, die über JMS Queues entkoppelt sind. Aus den Belegen erzeugt die erste Pipeline XML-Dokumente. XSL-Transformationen formatieren die Daten danach um, bevor sie schließlich wieder in einem Textformat in das SAP-System gelangen. Über die Konsole können sich berechnete Anwender den Status einer Datenlieferung anzeigen lassen.

wendungen – vollständig ersetzen. Wer sich darauf schon jetzt vorbereiten möchte, wird bei Firmen wie Sonic, Fiorano, Polarlake, Spiritsoft und weiteren fündig (siehe Tabelle „Anbieter von ESB-Lösungen“).

Interessante Alternativen bieten einige Open-Source-Projekte, die einen vergleichbaren Ansatz verfolgen. Babeldoc etwa ermöglicht die Definition so genannter Pipelines, in denen beliebige Dokumente verarbeitet werden können (siehe Kasten „Digitale Rohrpost“). Eine Pipeline besteht aus einer Menge von Stages, die das Dokument bearbeiten. Stages für die Umwandlung von strukturierten Textdateien in XML-Dokumente, für XSL-Transformationen, zum Versand von E-Mails oder SOAP-Nachrichten und andere gehören zum

Lieferumfang. Das Einspeisen der Daten in die Pipeline erledigen eine Reihe von Scannern, die Daten aus dem Dateisystem oder aus einer JMS Queue auslesen. Damit hat man schon einige wichtige Bausteine zusammen, die auch kommerzielle ESBs bieten.

Bemerkenswert ist die Tatsache, dass die wenigsten ESB-Produkte einen J2EE Application Server als Infrastrukturkomponente nutzen. Dabei würde der eine ideale Grundlage für einen ESB bie-

ten, bringt er doch einen JMS-Provider, Unterstützung für Web Services und die Java Connector Architecture mit. Vom Transaktionsmanager und den vielfältigen Kommunikationsprotokollen eines Application-Servers würde der ESB ebenfalls profitieren.

Ein ESB wäre in diesem Fall eine normale J2EE-Anwendung, vielleicht bestehend aus ein paar EAR-Dateien und einem Schwung Konfigurationseinträgen, um die benötigten JMS-Objekte und Datenquellen im Application Server anzulegen. Damit würde der ESB gleichzeitig den Beweis erbringen, dass er sich an Standards hält. Und der Kunde könnte entscheiden, auf welcher technischen Basis er ihn betreiben möchte.

Zwar unterstützen nicht alle Application Server bestimmte Features (irgendwie muss man sich schließlich unterscheiden). Wer vielleicht auf JMS Clustering für seinen ESB verzichten kann, muss sich daran jedoch nicht stören. Vielleicht könnte er aber später beim Hersteller seines Application-Servers nachfragen, wann er denn gedenkt, die neuen Funktionen bereitzustellen.

Fazit

Gegenwärtig kann man sich dem Eindruck nicht verschließen, dass auch die Anbieter von ESB-Produkten sich schwer tun, ihre Bausteine mit den Infrastruktur-Lösungen anderer Hersteller zu verbinden und den ESB als portablen Integrations-service der J2EE-Plattform zu positionieren. Eine mögliche Erklärung mag in der Vergangenheit der Protas-

gonisten zu finden sein: Ein Teil der ESBs kommt von Messaging-Systemherstellern. Diese Lösungen sind stark auf eine verteilte Umgebung mit vielen verteilten Message-Brokern ausgerichtet. Das passt aber so gar nicht zu der logischen Sternstruktur, die eine Installation auf einem zentralen Application Server darstellt.

Die Hersteller von Application-Servern wie BEA, IBM und Oracle hingegen haben ihre Produkte bereits als Integrationsplattform positioniert und bieten entsprechende Add-ons an. Diese Lösungen sind jedoch im Premium-Marktsegment angesiedelt und haben Schwierigkeiten, auch die unteren Regionen zu besiedeln. Wer also einen ESB auf seinem Application Server betreiben möchte, muss schon etwas suchen und darf gegebenenfalls auch nicht davor zurückschrecken, selbst für die notwendige Integration der ESB-Bausteine auf dem Server zu sorgen.

Insgesamt darf man aber gespannt sein, wie der Markt für Integrationslösungen auf die ESBs reagiert. Die Idee scheint viel versprechend, kombiniert sie doch die in traditionellen EAI-Lösungen gewonnenen Erfahrungen mit aktuellen Techniken und lässt dabei noch Raum für eigene Kreativität. (jd)

STEFAN KRIEGER

ist Principal Consultant bei der Comporsys Hansa GmbH in Hamburg.

CARSTEN LILL

arbeitet als Projektmanager bei der Soflabb GmbH in Hamburg.

INFOS IM WEB

Enterprise Service Bus entmystifiziert
Integrationstool Babeldoc
FAQs zu Microsofts Indigo
IBMs ESB-Ansatz erklärt
Quartz Enterprise Job Scheduler

www.fiorano.com/whitepapers/fiorano_esb.htm
sourceforge.net/projects/babeldoc
www.microsoft.com/indonesia/msdn/indigofaq1.asp
zdnet.com.com/2100-1104_2-5069335.html
www.opensymphony.com/quartz/

