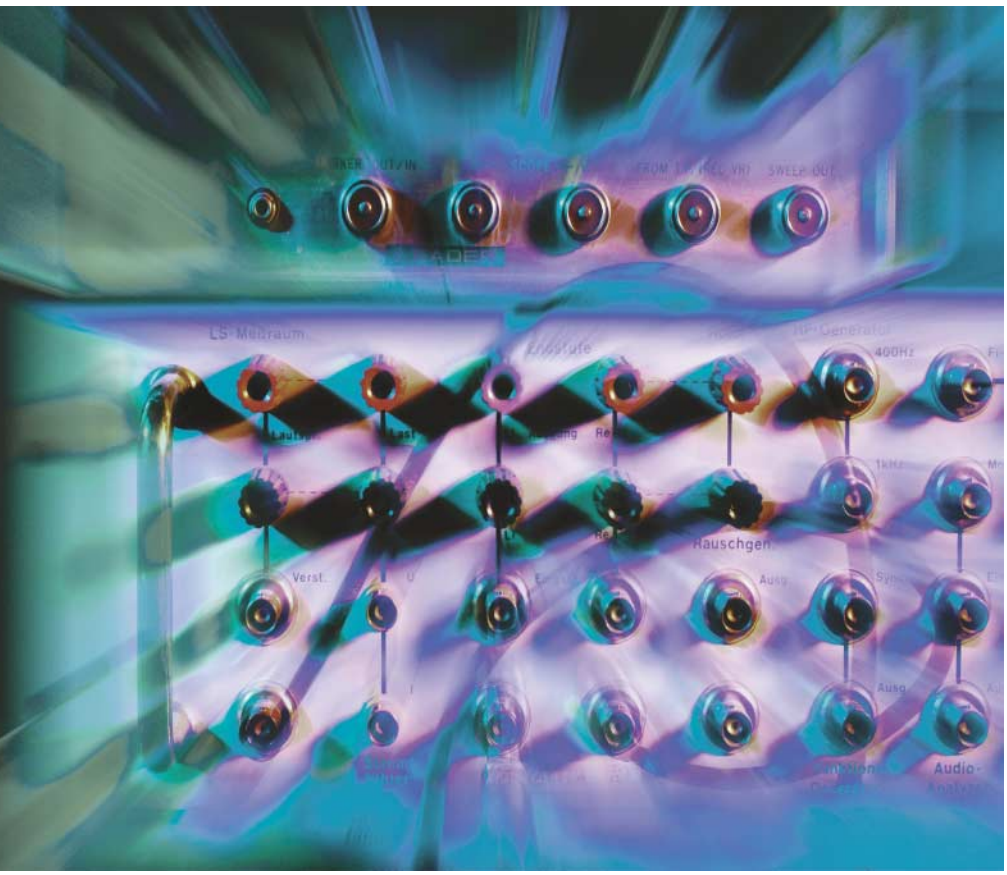
Erfahrungen mit der Webmethods-Plattform

Richtig gesteckt



Jürgen Schuck

Wer mit einem Integrationsprodukt wie Webmethods Projekte durchführen will, darf sich nicht vom einfach erscheinenden Baukastenprinzip blenden lassen. Auch hier gilt: Hinter den bunten GUIs muss dringend ein tragfähiges Konzept stehen.

Um seine Geschäftsprozesse zu straffen, wollte ein Telekommunikationsunternehmen sein Auftragsmanagementsystems (AMS) mit den Bestandsmanagementsystemen (BMS) seiner Kunden verbinden. Das Ganze sollte den Charakter eines elektronischen Marktplatzes erhalten. EAI-Spezialist Webmethods lieferte die Integrationsplattform. Weitere technische Rahmenbedingungen schufen Websphere MQ (MQ) von IBM für die Kommunikation mit den BMS, Action Request System (ARS) von Remedy (heute BMC), auf dem das AMS basiert, sowie die freie XML Common Business Library (XCBL) von Commerce One zur technischen Abbildung der Geschäftsdokumente, die Kunde und Lieferant austauschen.

Grundlage der Webmethods-Plattform ist der Integration Server, der über seine MQ- und ARS-Adapater an die BMSe beziehungsweise das AMS angeschlossen ist (Kasten „Plattform für Dienste“). Für die Kommunikation des AMS mit den BMS ist auf dem Server ein Web Service eingerichtet, mit dem das AMS arbeitet, da der ARS-Adapter den asynchronen Benachrichtigungsmechanismus von ARS nicht unterstützt. Trading Networks, ein Erweiterungsmodul, bietet Funktionen für die B2B-Kommunikation. Die Schnittstellen zu AMS und BMS nutzen davon die Dokumenttyperkennung, die typabhängige Weiterleitung zur Verarbeitung und die persistente Speicherung des gesamten Datenverkehrs.

Zur Entwicklung der EAI-Abläufe bot sich ein Topdown-Verfahren an, in dem der Webmethods Modeler zunächst die Sequenzdiagramme der Geschäftsprozesse in Prozessmodelle überführte. Die Codeentwicklung sowie Debugging und Test der Services erledigt der Programmierer mit der IDE, die den Namen Developer trägt. Schließlich verwendet das System die Publish/Subscribe-Schnittstelle des Broker zur asynchronen Kommunikation zwischen den Services (Abb. 1).

Die Distribution auf einem Windows 2000 Server mit 4 × 550 MHz und 2 GByte Hauptspeicher umfasst neben der EAI-Plattform 6.01 auch Oracle9i – Webmethods empfiehlt eine relationale Datenbank für die persistente Speicherung von Zuständen und Daten. Die zum Erzeugen der Datenbankstruktur notwendigen SQL-Skripts liegen in *<INSTDIR>/common/db/scripts*. Im Installation Guide (*<INSTDIR>/doc*) ist das Vorgehen beschrieben.

Wer flüssig mit den auf Java basierenden Programmen Modeler, Developer und Trading Networks Console sowie diversen Browser-Fenstern und Editoren für Text und XML arbeiten will, benötigt eine Workstation mit 1 GHz und 1 GByte RAM unter Linux oder Windows. Eine Setup-Routine half beim Aufspielen des Gesamtsystems. Die Einrichtung der MQ- und ARS-Adapter sowie von Patches erfolgte über die Im-/Exportfunktion des Administratortools. Webmethods lieferte die notwendigen Packages auf elektronischem Weg, da sie der Installer bei der Auswahl der Komponenten nicht berücksichtigte. Beim Aufsetzen der EAI-Plattform half die Best-Practices-Sammlung Gather Requirements, Explore, Assemble, Rollout (GEAR) von Webmethods. Die Kenntnis der rund 28 MByte umfassenden Dokumentation erleichterte den Umgang mit dem Produkt erheblich.

Mit der Definition der Geschäftsdokumente als Document Types begann die eigentliche Projektarbeit. Auf die-

iX-TRACT

- Mit Standardkomponenten von Webmethods hat ein Unternehmen aus der Telekommunikationsbranche sein Auftragsmanagementsystem mit den Bestandsmanagementsystemen seiner Kunden verbunden.
- Webmethods grafische Werkzeuge und vorgefertigte Schnittstellenadapter suggerierten, dass es möglich sei, die Implementierung des Systems ohne große Vorbereitung vorzunehmen.
- Im EAI-Projekt zeigte sich: Auch der schönste Werkzeugkasten entbindet nicht von der Pflicht, vor Beginn der Implementierung ein plausibles Konzept zu entwerfen.

ser Grundlage erstellten die Entwickler grundlegende Services zur Anbindung von MQ und ARS unter Verwendung der Adapter sowie eine Reihe weiterer Dienste für den späteren Umgang mit den Daten.

Transition von Namespaces

Die Document Types basieren auf der XCBL-Distribution von Commerce One. XCBL 4 macht umfassenden Gebrauch von XML-Namespaces. Beim Anlegen der Document Types aus den XCBL-Schemadefinitionen ersetzt Webmethods allerdings die Namespace-Bezeichnungen mit eigenen (*ns* für den leeren Namespace, *ns1* für den ersten vorkommenden, der nicht leer ist, und jeden weiteren fortzählend). Der Integration Server wandelt Namespaces in eine interne Repräsentation um. Syntaktisch sind die resultierenden Dokumente zwar in Ordnung, die abweichenden Bezeichnungen irritieren jedoch die BMSe. Pragmatische Lösung: Ein Service mit Filterfunktion wandelte die Namespaces des Integration Server in die von XCBL um, und zwar in jedem Dokument, das den Server in Richtung der BMSe verlässt. Nachdem die Entwickler diesen Weg gefunden hatten, konnten sie die XCBL-Distribution in kurzer Zeit implementieren – Webmethods scheint für solche Ad-hoc-Eingriffe gut gerüstet zu sein.

An der Schnittstelle zum AMS sind die Geschäftsdokumente als flache Strukturen ausgebildet. Dazu gibt es zwei Document Types, die namentlich den ARS-Schemata *Header* und *Detail* entsprechen. Auch die Namen ihrer Elemente stimmen mit den Feldbezeichnern (*Label*) in den Schemata überein. *Header* und *Detail* spiegeln die Struktur sämtlicher XCBL-Dokumente wider: Auf einen Header folgt eine Liste variabler Länge mit den Details. Das EAI-System bildet die Listenstruktur der XCBL-Geschäftsdokumente vom BMS auf eine relationale Struktur ab, wie sie das AMS erwartet.

Wer die MQ- und ARS-Adapter verwenden will, sollte die jeweilige C-API kennen. Ersterer benötigt den Websphere MQ Client, den man unentgeltlich von der IBM-Website herunterladen kann. Während der Installation des Adapters nistet sich ein Verwaltungs-GUI in die Administratorkonsole ein, das zur Deklaration von Queue-Managern und Queues dient sowie zur Definition von Services, die von diesen Queues lesen und darauf schreiben. Das GUI hat die alleinige Hoheit über diese Dienste. Jede Änderung, die man nun mit dem Developer vornimmt, führt dazu, dass der betreffende Service nicht mehr funktioniert. Der Test der MQ-Services kann mit den Tools *amqsputc.exe* und *amqsgetc.exe* (*C:\Programme\IBM\WebSphere MQ\Tools\c\Samples\Bin*) erfolgen.

Über die numerischen Feld-IDs greifen die Services des ARS-Adapters auf die Felder eines Schemas zu. Da dies aufgrund von mehr als 100 Elementen für die beiden Document Types *Header* und *Detail* nicht praktikabel ist, sind die Services zum Schreiben und Lesen vom ARS in Wrapper-Services eingebettet, die den Umgang mit Labels anstelle IDs ermöglichen. Diese pragmatische Lösung mit den einfachen Mitteln des Developer zeigte leider einen schwachen Durchsatz. Die Ursache dafür liegt in der Interpretation von Services durch den Integration Server, wenn diese die eingebauten Sprachmittel des Developer verwenden. Künftige

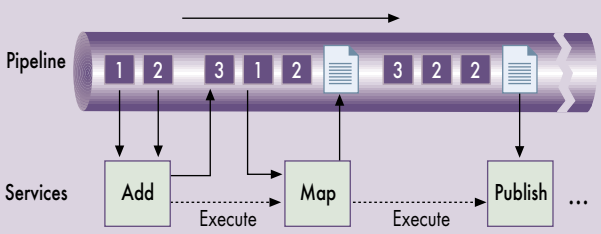
Plattform für Dienste

Das Kernstück der Webmethods-Plattform ist der Integration Server, ein dedizierter Application Server für die Ausführung von EAI-Code. Die Organisation des Codes erfolgt in so genannten Services. Der Grunddatentyp ist *String*. Weitere Typen sind *List-of-Strings*, *Document* und *List-of-Documents*, mit denen sich aus dem Grundtyp verschachtelte Strukturen zur Abbildung von XML zusammenstellen lassen. Solche strukturierten Variablen werden als *Documents* bezeichnet. Ihre Definition im Integration Server erfolgt aus XML- oder XSD-Dateien (XML Schema Definition).

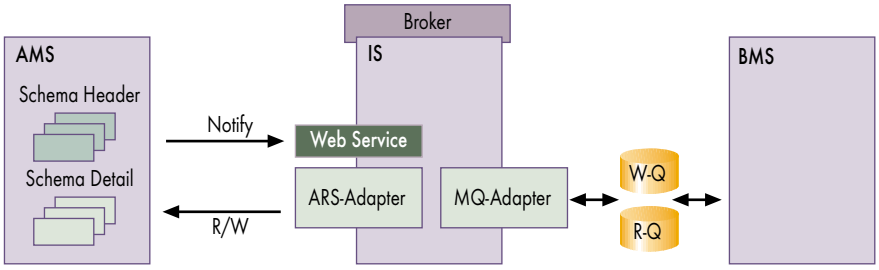
Packages gruppieren Services und Document Types zu administrierbaren Einheiten, die zur Speicherung in Folder untergliedert sind. Spezielle Services kann der Integration Server beim Laden und Entladen eines Package zur Initialisierung ausführen. Für den Datentransport gibt es im Server eine Pipeline, aus der Services Variablen lesen beziehungsweise schreiben (Abb. 3). Zur Systemanbindung stellt der Server jeden Dienst als Web Service bereit. Adapter sind Erweiterungen des Servers für Kopplungen mit Anwendungssoftware ohne Web-Services-Schnittstellen.

Der Broker ist eine optionale Komponente, die eine Publish-/Subscribe-Schnittstelle zur asynchronen Kommunikation zwischen Services bereitstellt, die sich auf unterschiedlichen Integration Servers befinden können. Seine Fähigkeiten schaffen die Voraussetzungen für verteilte EAI-Lösungen und machen ihn zum Skalierungswerkzeug, mit dem sich auf Änderungen der Leistungsanforderungen reagieren lässt. Zur Entwicklung von Packages und Services dient der auf Java basierende Developer.

Trading Networks Console und der Modeler sind spezialisierte GUIs: Erstere generalisiert den Datenaustausch mit Fremdsystemen durch die Funktionen „erkennen“, „persistieren“ und typspezifische Verarbeitung von Dokumenten. Der Modeler ist ein Werkzeug für die Entwicklung von Prozessmodellen für Geschäftsprozesse. Es beinhaltet einen Codegenerator, der aus den Modellen Services erzeugt und sie auf einen Integration Server überträgt.



Services lesen Daten aus der Pipeline, schreiben ihre Verarbeitungsergebnisse zurück und rufen weitere Dienste auf (Abb. 3).



Die Systemarchitektur: Die proprietären Dokumente an der Schnittstelle zum Auftragsmanagementsystem bilden die ARS-Schemata *Header* und *Detail* auf eine flache XML-Struktur ab (Abb. 1).

Releases der Schnittstelle zum ARS-System werden daher wohl Wrapper-Services vorsehen, die Hash-Algorithmen anstelle der Schleifen für das Label-/ID-Mapping verwenden.

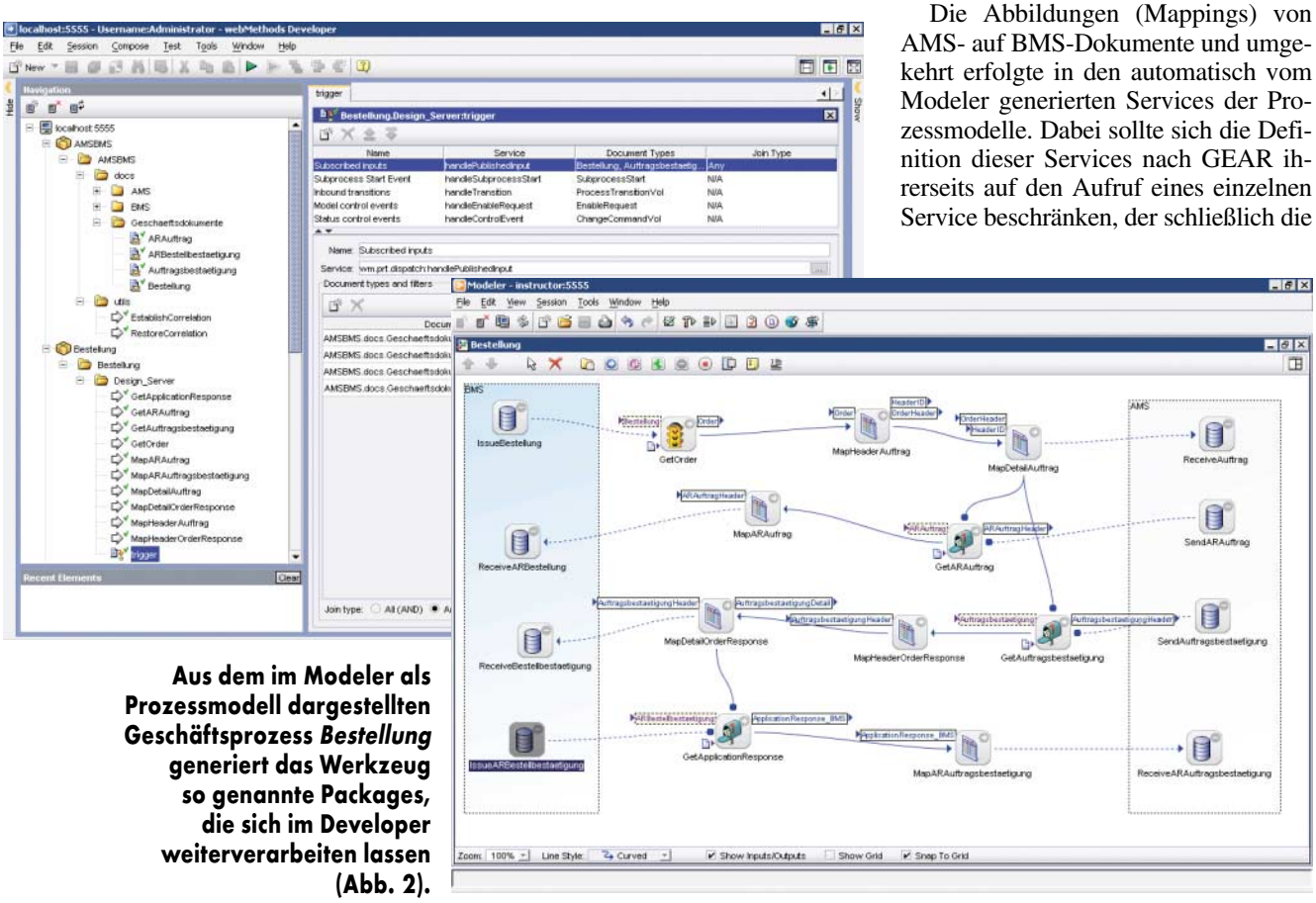
Trading Networks leitet die von AMS und BMS empfangenen Dokumente typspezifisch weiter. Dazu ist für jeden Typ ein eigener Service vorgesehen, der die Dokumente über die Publish-/Subscribe-Schnittstelle zum Broker schickt. Den asynchronen Empfang aller Dokumente vom AMS erledigt ein Web Service, da der ARS-Adapater, wie oben schon erwähnt, die erforderlichen Funktionen nicht bietet. Dieser einfach gehaltene Dienst erhält vom AMS als einzigen Parameter eine ID, mit der er den zugehörigen Eintrag aus dem Header Schema liest, den er anschließend an Trading Networks sendet.

Geschäftsprozesse stehen Modell

Der Broker stellt die Dokumente den so genannten Prozessmodellen anhand von entsprechenden Subscriptions zu. Auf dem Integration Server ist ein Prozessmodell ein Package, das für jeden Prozessschritt einen Service enthält. Diese Packages generiert der Modeler aus den grafischen Beschreibungen der Geschäftsprozesse (Abbildungen 2 und 3). Mit der Zustellung eines Dokuments löst der Broker den Start oder die Fortsetzung einer Prozessmodellinstanz aus, je nachdem, ob es sich um das erste oder eines der weiteren Dokumente im Ablauf des Geschäftsprozesses handelt. Instanzen führen so genannte Correlation-IDs mit sich, die Vorgänge eindeutig kennzeichnen.

Bei der Arbeit mit dem Modeler an den Prozessmodellen sollte man für Variablen unbedingt nachvollziehbare Namenskonventionen einführen, denn allzu leicht verheddert man sich im Gestrüpp spontan ersonnener Bezeichnungen, die schließlich im Kontext des gesamten Prozessmodells jedweder Logik entbehren. GEAR gibt dazu leider keinerlei Hinweise. Es erwies sich bei der Entwicklung der Prozessmodelle als äußerst hilfreich, die Funktion des Modeler zur Angabe von Eingangs- und Ausgangsvariablen für jeden Prozessschritt zu verwenden. Dies dient zum einen dem Verständnis der Abläufe beim Betrachten des Modells, zum anderen definiert der Modeler bei der Generierung der Packages die mit den Prozessschritten korrespondierenden Services inklusive Variablendeklaration. Das wiederum erleichtert die spätere Arbeit im Developer, da Typen und Namen der besagten Variablen bereits vorgegeben sind.

Die Abbildungen (Mappings) von AMS- auf BMS-Dokumente und umgekehrt erfolgte in den automatisch vom Modeler generierten Services der Prozessmodelle. Dabei sollte sich die Definition dieser Services nach GEAR ihrerseits auf den Aufruf eines einzelnen Service beschränken, der schließlich die



Aus dem im Modeler als Prozessmodell dargestellten Geschäftsprozess *Bestellung* generiert das Werkzeug so genannte Packages, die sich im Developer weiterverarbeiten lassen (Abb. 2).

INFOS IM WEB

deutschsprachiges Forum zum Thema EAI	www.eaiforum.de
XCBL-4-Distribution von Commerce One	www.xcbl.org/xcbl40/xcbl40.html
Websphere MQ Clients	www.ibm.com/developerworks/websphere/downloads/

Verarbeitung steuert. Für die Mappings zwischen umfangreichen und tief geschachtelten Listenstrukturen einerseits (XCBL) und flachen relationalen Strukturen andererseits (ARS) sorgen eine Reihe von Hilfsdiensten, deren Entwicklung einem grundlegenden Muster folgte: Für jede benötigte Liste gibt es ein Paar komplementärer Services. Davon zerlegt der eine die Liste in diskrete Werte, und der andere fügt sie aus diesen wieder zusammen. Weiterhin enthält jeder Service nur eine Schleife. Bei der Verarbeitung geschachtelter Listen führt der Service die innere Liste im Schleifenkörper einem weiteren Dienst zu, der ebenfalls nur diese Liste verarbeitet und gegebenenfalls weitere Services aufruft.

Solche Services lassen sich leicht testen und wie Bausteine zusammensetzen. Leider waren sie als Transformer ungeeignet – als Dienste also, die im Zuge eines Mapping „on-the-fly" aufgerufen werden – da der Integration Server dies aus ungeklärten Gründen verhinderte. Ein weiterer Grund, die Komplexität von Services auf das notwendige Minimum zu beschränken, lieferte die Eigenschaft des Developer, dass Änderungen an Services gelegentlich keine Wirkung zeigten. Oft hilft es in einem solchen Fall, den Developer neu zu starten, manchmal ist es jedoch erforderlich, den Service zu löschen und neu zu definieren.

Eine weitere Besonderheit in diesem Zusammenhang betrifft die Möglichkeit, Variablen zu referenzieren, sofern es sich um Dokument Types handelt. Änderungen an Document Types, etwa: „Typ eines Elements von String nach List-of-Strings", haben keine Auswirkungen auf Services, die den betreffenden Document Type referenzieren und das geänderte Element in einem Mapping oder in sonstiger Weise verwenden (beispielsweise in Schleifen oder Bedingungen). Eine derartige Modifikation zieht ansonsten erhebliche manuelle Arbeiten mit entsprechendem Fehlerpotenzial nach sich. Wenn es sich nicht vermeiden lässt, hilft der Developer mit dem Befehl *Find Dependencies*, der alle Services mit Referenzen zum Objekt auflistet.

Die Enterprise Integration Platform von Webmethods erweckte anfangs den Eindruck eines Werkzeugkastens, dessen Fähigkeiten zur Kommunikation mit heterogenen Systemen (MQ, ARS) sowie die konsequente Bedienung über GUIs einen leichten Umgang mit den vielfältigen und teils komplexen Aufgaben eines Integrationsprojekts versprachen. Im Laufe des Projektes stellte sich jedoch heraus, dass diese Eigenschaften die Arbeit zwar erleichtern, aber nicht die Notwendigkeit eines Konzepts eliminieren, das den Rahmen der Tätigkeiten absteckt und die Optionen entsprechend einschränkt.

Ohne Konzept geht nichts

Im Nachhinein gesehen hat sich das folgende grobe Vorgehen bewährt. Am Anfang steht die gründliche Auseinandersetzung mit den beteiligten Geschäftsprozessen. Dabei muss man insbesondere die Rollen der beteiligten Systeme (AMS, BMS) verstehen und sie anschließend in Sequenzdiagramme überführen. Darauf folgt die Definition der technischen Geschäftsdokumente (XCBL, Header, Detail). Nun steht die Evaluation und Erprobung der Adapter (MQ, ARS) an sowie die Ableitung elementarer Services. Das Gleiche gilt für die Abbildungen der technischen Dokumente in den beteiligten Systemen. Ist das erledigt, überführt der Entwickler die Geschäftsprozesse (Sequenzdiagramme) in Prozessmodelle und legt dabei die Namenskonventionen fest. Zu guter Letzt erstellt er die EAI-Abläufe mit den bisher generierten Services. (jd)

JÜRGEN SCHUCK

arbeitet bei der Materna GmbH als Projektleiter im IT-Service-Management.

Literatur

[1] Achim Born, Jürgen Diercks; EAI; Abgeschirmt; Anwendungsintegration: grenzenlose Zusammenarbeit; *iX* 11/2003, S. 88.

[2] Richard Nußdorfer; Das EAI-Buch; 1. Auflage, München; CSA Consulting GmbH, 2000.