Moderne Systeme sehen meist alt aus
Neue Perspektiven für Legacy-Anwendungen

Fred Stefan, Sabine Busch, Patrick Wabnitz

stefan@informatik.uni-leipzig.de
buschs@de.ibm.com
p.wabnitz@googlemail.com

Abstract: An einem konkreten Beispiel der Anbindung eines über Jahrzehnte hinweg gewachsenen Zulieferer-Systemes an ein modernes E-Commerce System (Online Shop), soll das Integrationsmuster der dateibasierten Integration unter Verwendung des IBM WebSphere Message Broker vorgestellt werden. Dabei wird ein besonderer Augenmerk auf die Konstruktion einer einfachen nachrichtenbasierten Integrationslösung für eine anpassungsfähige Informationstechnologie gelegt.

1 Einführung

Das Thema Legacy-Integration ist heutzutage in aller Munde. Bisherige Integrationslösungen setzten häufig den Ansatz von Punk-zu-Punkt- oder Hub-and-Spoke-Verbindungen zwischen diesen Anwendungen ein. Derzeitig „state of the Art“ sind die service-orientierte Architekturen (SOA), deren Herzstück der Enterprise Service Bus (ESB) bildet.

Viele etablierte Integrationslösungen stoßen jedoch schnell an ihre Grenzen, sobald Legacy-Anwendungen der älteren Generation involviert sind. Vorzugsweise wurden diese Anwendungen (und werden auch immer noch) in C, Cobol, PL/l oder auch in Assembler Maschinensprachen programmiert. [LB08, S. 572] Da diese Anwendungen nicht nach den heutigen Prinzipien der Portabilität und Flexibilität konzipiert wurden, fehlen oft geeignete Schnittstellen. Aus diesem Grund werden sie schon seit Jahren als besondere Schwierigkeiten in großen Unternehmen gesehen. [BS95, S. 2]

Eine hierfür typische Anwendung ist meist älter als 10 Jahre, besitzt einen monolithischen Aufbau mit viele (Millionen) Zeilen Code und ist oftmals auf einem traditionellen Mainframe-System beheimatet. Grundlegend zeichnen sie sich weiterhin durch ihren eminenten Widerstand gegenüber jeglichen Veränderungen und Modifikationen aus. Genau diese Veränderungen, sind jedoch für einen heutigen, flexiblen Geschäftsbetrieb unabdingbar.

Die Integration solcher Anwendungen kann in zwei unterschiedliche Vorgehensweisen differenziert werden. Bei einem *invasiven* Vorgehen, muss in bestehende Anwendungen eingegriffen werden. Dies setzt im Allgemeinen ein detailliertes Wissen über die Anwen-

dungen voraus. Da die bei der Erstellung beteiligten Entwickler in den meisten Fällen bereits die Unternehmen verlassen haben, ist eine spätere Anpassung der Anwendungen nur mit äußerst großem Aufwand möglich. [JRHD00, S. 35] Dem gegenüber stehen *nicht-invasive* Vorgehensweisen, welche Anwendungen ohne Eingriff in bestehenden Quellcode integrieren können.

Da längst keine einheitliche Systemarchitektur mehr zu erreichen ist, kann zumindest mit einer Message Oriented Middleware (MOM) eine einheitliche, sichere Kommunikation zwischen den einzelnen Anwendungen sichergestellt werden. Integrationslösungen dieser Art nehmen Daten in Form abgeschlossener Pakete - den Nachrichten - von einer Anwendung entgegen und übermitteln sie sicher an eine oder mehrere andere Anwendungen. Sollten die zu integrierenden Legacy-Anwendungen keine geeigneten Schnittstellen (wie z.B. Web Services) oder sonstige Andockpunkte bereitstellen - was in der überwiegenden Mehrzahl der Fall ist - müssen bestehende Anwendungen von Spezialisten kostspielig angepasst werden. Hierbei muss jedoch in die Legacy-Anwendungen eingegriffen werden (*invasive Integration*).

Ähnlich einer MOM arbeiten Message Broker nachrichtenorientiert. Hierfür zentralisieren sie alle verfügbaren Schnittstellen und Andockpunkte in einem Integration Hub. Eine mögliche Schnittstelle zu den Legacy-Anwendungen liefert ihre Datenhaltung. Hiermit können bestehende Anwendungen ohne Eingriffe in den Quellcode erweitert, bzw. integriert werden. Für diesen Zweck besitzen Message Broker meist proprietäre Adapter und Konnektoren, welche verfügbare Schnittstellen für den Datenaustausch der zu integrierenden Anwendung abbilden. Somit stellen MB gemeinhin eine *nicht-invasive* Möglichkeit der Integration dar. [HW03, S. 82 f.]

2 Problemstellung

Alternativ zu Datenbankmanagementsystemen werden in vielen Legacy-Anwendungen auch Flatfiles[1] zur Datenhaltung eingesetzt. [Her03, S. 492] Diese Entwicklung hat ihren Ursprung in den 60er Jahren und wird auch in der heutigen Zeit von vielen Unternehmen noch in dieser Form eingesetzt, da die damit verbundenen Anwendungen einen gewissen Wert für die Unternehmen darstellen. [Hal08, S. 27 ff.]

Wie können nun solche Anwendungen, die weder über geeignete Schnittstellen oder Andockpunkte, noch über eine relationale Datenbank verfügen, in eine möglichst flexible IT-Infrastruktur integriert werden?

Für alle Beteiligten solcher Integrationsprojekte zeigt sich schon nach kurzer Zeit die außerordentliche Komplexität der Thematik. Ein großes Problem ist meist, dass der Quellcode vieler Legacy-Anwendungen einfach fehlt, zerstört wurde oder nur unzureichend dokumentiert ist. Um Altanwendungen zu modernisieren, oder in eine SOA zu integrieren, bieten sich in den meisten Fällen nur noch Screenscraping- und Wrapping-Lösungen an, die jedoch von geringer Performanz geprägt sind. [Amb98, S. 352], [SPL03, S. 228]

[1] im Deutschen auch häufig als Flache-Dateien bezeichnet

2.1 Zielsetzung

Im Rahmen dieses Papiers soll ein *nicht-invasiver*, nachrichtenbbasierter Ansatz zur Integration Host-basierter Legacy-Anwendungen vorgestellt werden. Stellvertretend für die Vielzahl von Produkten, die schon lange auf dem Markt sind, soll ein solches Ansatz anhand des prominenten Beispiels IBM WebSphere Message Broker beschrieben werden. Für eine möglichst realitätsnahe Testumgebung erfolgten die Betrachtung im Integrationslabor der Universität Leipzig. Somit steht für die Untersuchungen ein S/390 Großrechner zur Verfügung.

3 Lösungsansatz

Einen flexiblen und performanten Lösungsansatz zur Integration solcher Flatfile-basierten Legacy-Anwendungen bietet der IBM WebSphere Message Broker (MB). Der MB agiert im Stil eines Nachrichten- und Protokollschalters, der auf Funktionen zur Transformation und zum intelligenten Routing zurückgreift und damit die benötigten Daten genau am gewünschten Ort und in der gewünschten Form zur Verfügung stellen kann. Sind Anwendungen temporär nicht zu erreichen, so speichert der MB die Nachrichten in einer Warteschlange (Queue).

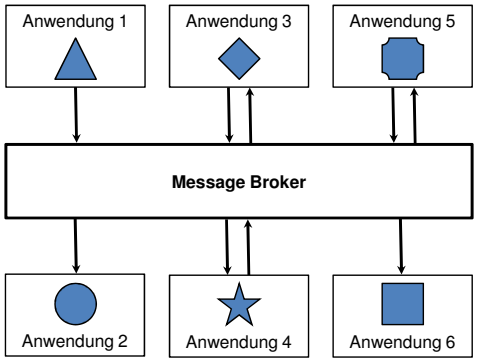

Abbildung 1: Middleware-Architektur für Legacy-Integration

Abbildung 1 zeigt eine Middleware-Architektur für Legacy-Systeme. Die Anwendungen kommunizieren nicht direkt miteinander, sondern schicken Nachrichten an einen MB. Somit wird die Topologie innerhalb einer Systemumgebung erheblich vereinfacht. Die Funktionsweise des MB kann soweit ausgebaut werden, dass er über einen definierten Regelsatz verfügt. Es wird somit bestimmt, welche Nachrichten an welche Anwendungen verschickt werden und inwieweit Nachrichten diesbezüglich zu transformieren sind.

So könnte eine Regel zum Inhalt haben, dass alle Nachrichten, die eine bestimmte Produktkategorie betreffen (z.B. Aufträge für Schreibutensilien) von einer bestimmten Anwendung bearbeitet werden sollen. Durch Adaption der entsprechenden Regel kann die

Anwendung zur Bearbeitung von Aufträgen für Schreibutensilien jederzeit angepasst oder ausgetauscht werden. So ist eine Diversifizierung von Prozessen möglich, indem z.B. spezielle Anwendungen für wiederverwendbare oder wegwerf Schreibutensilien vorgesehen werden.

Durch die MOM wird eine einheitliche und sichere Kommunikation zwischen den einzelnen Anwendungen sichergestellt. Der Entwickler wird hierdurch von allem entlastet, was mit dem Transport der Daten zu tun hat. Die Daten müssen der Middleware nur in geeigneter Form übergeben werden.

MOM-Lösungen sind konzeptuell nicht so weit von service-orientierten Architekturen entfernt, da sie Nachrichten für die Kommunikation verwenden. Durch den MB werden Legacy-Anwendungen voneinander entkoppelt und Punkt-zu-Punkt-Verbindungen werden vermieden.

Sollten also bei Altanwendungen keine Schnittstellen oder Quellcode verfügbar sein, so bietet die Integration auf Datenhaltungsebene, also unter Einbeziehung der Flatfiles, einen geeigneten, *nicht-invasiven* Integrationsansatz.

3.1 Warum Flatfiles?

Flatfiles bieten zum einen ein äußerst kompatibles Format für den Austausch von Informationen zwischen verschiedenen Anwendungssystemen und Datenbanken und zum anderen eine schnelle und einfache Möglichkeit Daten zu speichern und wieder einzulesen. Aus diesem Grund sind Flatfiles noch immer sehr weit verbreitet.

Generell sind Flatfiles äußerst einfach konstruiert. Weil sie keine Informationen über ihre eigene Struktur einschließen (Metadaten), ist der Overhead, also die Daten, die nicht primär zu den Nutzdaten zählen, minimal. Ein einfaches Beispiel hierfür wäre die Speicherung des Warenbestandes einer Firma in einem Flatfile, wie es auszugsweise in Abbildung 2 dargestellt ist.

```
0010  Ball Pens Black 24pk              2.90    458
0020  Ball Pens Blue 24pk               2.90     12
0030  Ball Pens Red 24pk                2.90    687
0040  Ball Pens Green 24pk              2.90    145
0050  Pencil with eraser 12pk           1.78    125
...           ...                       ...    ...
```

Abbildung 2: Auszug aus einem Flatfile

Leicht zu erkennen ist, dass die Datei nur Daten enthält. Jedes Feld hat eine genau festgelegte Länge (so ist das Feld mit der Artikelnummer beispielsweise immer genau 4 Zeichen lang) und keine Struktur trennt ein Feld vom anderen. Diese Feldpositionen und Längen wurden in einer Initialen-Phase festgelegt. Infolgedessen muss jede Anwendung, die diese Datei benutzt, diese Felddefinitionen kennen, da diese Informationen nicht in der Datei selbst gespeichert sind.

In der Literatur wird das Flatfile-Design meist mit „throw-everything-into-one-big-table“ beschrieben. Genau so vielseitig kann auch deren Aufbau sein. Dies führt dann zu Dateien, die eine minimale Struktur aufweisen. Damit der MB mit solch einer einfachen Sammlung von Datensätzen, die nacheinander in einem bestimmten Format in einer Datei angeordnet sind, auch arbeiten kann, müssen diese Informationen zunächst in kleine Einheiten zerlegt werde. Dieser Prozess wird parsen genannt. [Mic02, S. 534] Der Bitstrom einer ankommenden Nachricht wird interpretiert und in eine interne Darstellung der Nachricht überführt. Darüber hinaus generiert ein anderer Parser den Bitstrom für eine abgehende Nachricht auf Basis der internen Darstellung. [IBM07]

3.2 Aufbau des IBM WebSphere Message Broker

Der IBM WebSphere Message Broker ist ein vielseitig einsetzbare Middleware-Lösung zum Steuern, Transportieren und Transformieren von Daten und Informationen die von typischen Business Umgebungen erzeugt werden. Verfügbar ist der MB für eine Vielzahl von Systemen. Hierzu zählt unter anderem IBM AIX, HP-UX, Linux (x86/64/System z Plattformen), Sun Solaris, Microsoft Windows XP und Windows 2003 sowie IBM z/OS. [IBM08]

Der Broker unterstützt eine Vielzahl von Transport-Protokollen und Datenformaten, die in sogenannten Message Domänen zusammengefasst werden, um eine Reihe einfacher Werkzeuge für die entsprechenden Formate bereitzustellen. Diese existieren beispielsweise für binäre Daten die durch C / COBOL erzeugt wurden, formatierte Texte, SWIFT, X12, Comma Separated Values (CSV), XML und andere.

Dabei ist ein plattformunabhängiger Einsatz möglich, da mehrere Netzwerk-, Anwendungs- und Schnittstellentypen integriert und administrativ verwaltbar sind.

Der elementare Aufbau des MB und das Umfeld der Integrationslösung ist in Abbildung 3 dargestellt. Grundlegend besteht der MB aus zwei Komponenten:

- **Message Broker Toolkit:** Das Message Broker Toolkit ist eine auf Eclipse basierende Entwicklungsumgebung und grafische Benutzerschnittstelle, die für Windows- und Linux-Rechner verfügbar ist. Anwendungsentwickler entwickeln hiermit Nachrichtenflüsse und Nachrichtengruppen. In den einzelnen Komponenten der Nachrichtenflüsse wird als Programmiersprache ESQL oder Java eingesetzt. Um diese dann auf einem oder mehreren Brokern einzusetzen, muss das Toolkit mit einem Konfigurationsmanager kommuniziern, der wiederum ein oder mehrere Broker verwaltet. Sollten mehrere Entwickler parallel an den gleichen Ressourcen arbeiten wollen, kann mit einem Repository die Zugriffsteuerung und Versionierung vorgenommen werden. Dabei kann jedes von Eclipse unterstützte Repository verwendet werden.
- **Laufzeitumgebung:** Die Laufzeitumgebung umfasst eine Gruppe von Ressourcen, die zur Laufzeit vorhanden sind. Hierzu zählen der Konfigurationsmanager, die Broker und verschiedene Ausführungsgruppen. Broker mit einer einheitlichen

Konfiguration können zu einer Brokerdomäne zusammengefasst werden. Diese Brokerdomäne wird anschließend von einem gemeinsamen Konfigurationsmanager koordiniert. So können zum Beispiel unterschiedliche Prozessabläufe in verschiedenen Unternehmensbereichen getrennt voneinander bearbeitet werden. Sobald Nachrichten im Broker ankommen, werden sie in der dem Broker zugeordneten Ausführungsgruppe anhand definierter Nachrichtenflüsse verarbeitet.

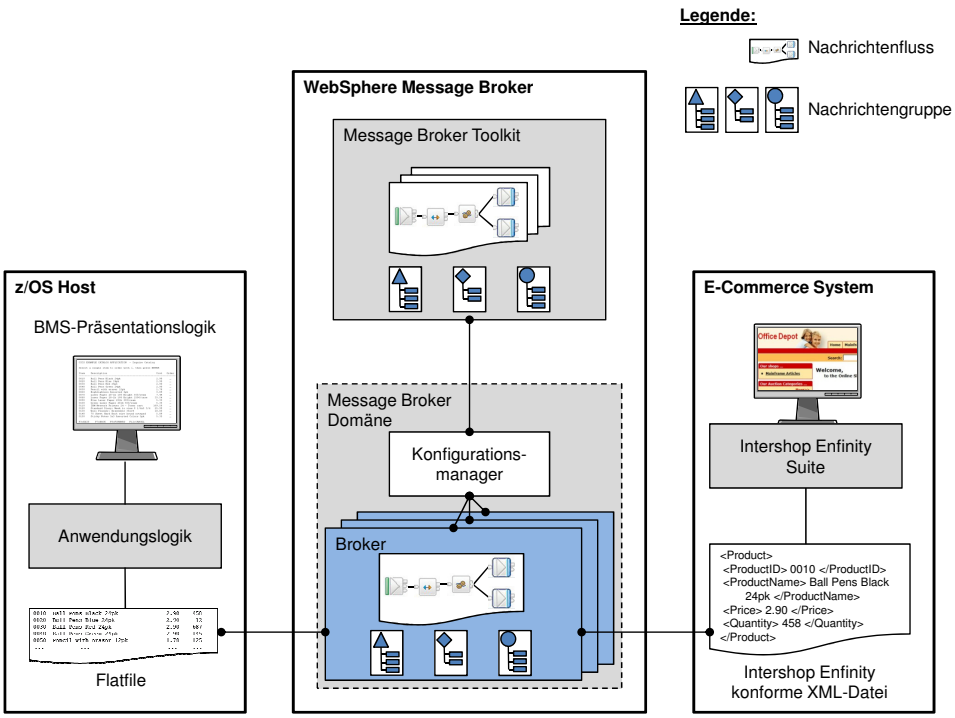

Abbildung 3: Einbindung des IBM WebSphere Message Brokers in die vorherrschende Systemumgebung

Zur Laufzeit wird jede Ausführungsgruppe als separater Betriebssystemprozess gestartet. Somit bekommt jede Gruppe von Nachrichtenflüssen jeweils eine eigene Laufzeitumgebung zur Verfügung gestellt. In der eigentlichen Kernkomponente, dem Broker, können gleichzeitig mehrere Nachrichtenflüsse ablaufen. Diese Nachrichtenflüsse, welche mit dem Message Broker Toolkit entwickelt werden, beschreiben detailliert die Weiterleitung, Umwandelung und Aufbereitung der aufgelaufenen Nachrichten.

Ein Broker speichert all seine relevanten Informationen in einer Brokerdatenbank. Hierzu zählen beispielsweise Steuerdaten für die vom Broker definierten Ressourcen, wie etwa die der implementierten Nachrichtenflüsse.

3.3 Aufbau und Funktionalität der Entwicklerumgebung

Die Entwicklung, Modellierung und Konfiguration von Brokeranwendungen erfolgt mit dem Message Broker Toolkit. Um die Übersicht und Funktionalität zu vereinfachen, stellt das Toolkit verschiedene Perspektiven bereit. Diese repräsentieren eine, auch individuell anpassbare, Sammlung von Anzeigen und Editoren, welche die Entwickler bei der Arbeit mit bestimmten Ressourcentypen unterstützt. Die zwei wichtigsten Perspektiven bei der Arbeit mit Brokern sind die Brokeranwendungsentwicklungs- und Brokerverwaltungs-Perspektive.

Standardmäßig wird für die Entwicklung von Nachrichtengruppen, Nachrichtenflüsse und anderen relevanten Ressourcen die Brokeranwendungsentwicklungs-Perspektive geöffnet. Hier werden alle für die Erstellung und Verwaltung neuer und bestehender Projekte nötigen Funktionalitäten, wie z.B. Editoren, Wizards, sowie Im- und Export-Funktionalitäten bereitgestellt. Um Brokerdomänen zu verwalten und um mit den Kommunikationsmanagern zu interagieren, wird die Brokerverwaltungs-Perspektive genutzt. Fertige Projekte können hier zusammengefügt und getestet werden. Diese werden zu einen komprimierten Broker-Archiv, einer Art Implementierungseinheit für den Broker, zusammengefasst. Das Broker-Archiv beinhaltet die entwickelten Nachrichtenflüsse und Nachrichtengruppen in einer ausführbaren Form. Diese Archiv-Datei kann auf den entsprechenden Broker eingesetzt werden. Weiterhin können in der Brokerverwaltungs-Perspektive auch einzelne Nachrichtenflüsse gestartet, kontrolliert oder gestoppt werden.

Um Ressourcen, wie Dateien, Ordner oder Projekte, mit dem Toolkit zu bearbeiten, stehen eine Reihe an Editoren zur Verfügung. So existieren beispielsweise für Brokerarchive, ESQL, Ereignisprotokolle, Nachrichtendefinitionen, Nachrichtenzuordnungen, Nachrichtenknoten und Nachrichtenflüsse spezielle Editoren. Bei der Modellierung kann auch auf die bereitgestellten Editoren verzichtet werden und die Entwicklung auf Quellcode-Ebene (XML oder C) verlagert werden. Daneben besteht die Möglichkeit, auch eigene Editoren zu verwenden, um besondere Funktionsweisen besser zu beschreiben.

Ist mit dem Toolkit ein Nachrichtenfluss modelliert wurden, so kann die Broker-Archivdatei erzeugt werden. Anschließend erhält man die im Working Set in Abbildung 4 abgebildeten Ressourcen, die vom Toolkit generiert werden. Letztendlich kann die Broker-Archivdatei mit der Brokerverwaltungs-Perspektive in einen oder mehrere Broker eingesetzt werden. Während der Entwicklung eines Nachrichtenflusses oder einer Nachrichtengruppe werden automatisch Ressourcen, wie z.B. esql-Dateien erzeugt. Die generisch erzeugten Ressourcen sind im Einzelnen:

- Die msgflow-Datei ist die Nachrichtenflussdatei, welche Informationen für die grafische Darstellung des im Projekt entwickelten und modellierten Flusses beinhaltet. In ihr enthalten sind alle Knoten und Verbindungen der Nachrichten, Warteschlangen und Transformationen sowie deren Eigenschaften und Konfigurationen.
- Die esql-Datei beinhalten Code in dem im MB definierte Programmiersprache ESQL. Welche beispielsweise im Compute-Knoten benutzt wird um Veränderungen an Variablen für Ein und Ausgabe vorzunehmen oder die Nachrichten einfach Bitweise

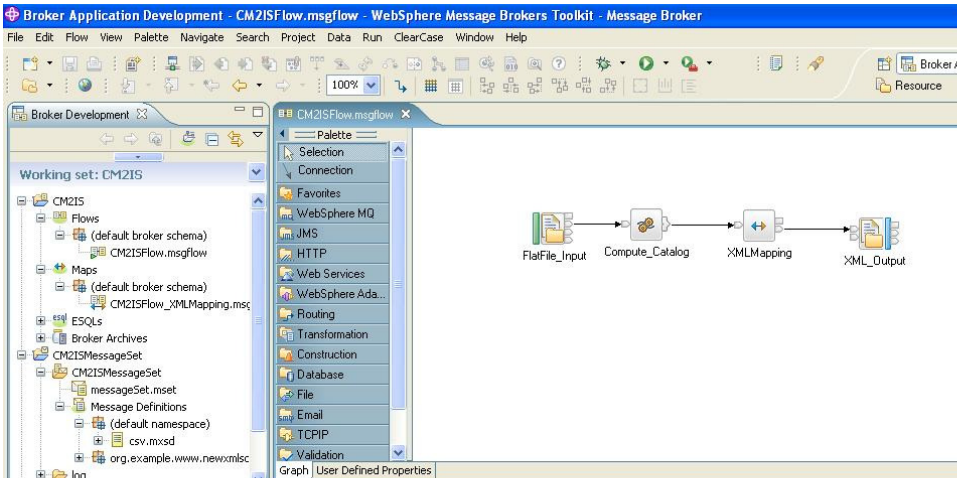

Abbildung 4: Das Message Broker Toolkit in der Brokerverwaltungs-Perspektive mit den im Projekt generierten Ressourcen

einzulesen. ESQL liefert Funktionen und Prozeduren für Rechenknoten, Filterknoten und Datenbanken.

- Dateien vom Typ msgmap werden benötigt, um Zuordnungen sowie Konvertierungen von Nachrichten und Datenbanken zu speichern. So können beispielsweise Elemente und Attribute einer Quelldatei auf eine Zieldatei abgebildet (gemapped)werden. In der vorliegenden Problemstellung werden die Daten des Faltfiles mithilfe dieser Nachrichtendefinitionsdatei und Modellierungen (Schemata) auf die gewünschte XML Datei abgebildet.

- Die messageSet.mset-Datei enthält u.a. die physischen Formatdefinitionen der Nachrichten und Gruppe des Projektes, sowie die Modell Eigenschaften die in alle Nachrichten vereinheitlicht sind. Dies umfasst somit alle allgemeine Einstellungen bezüglich der Nachrichten innerhalb eines Projektes.

- Eine so genannte Nachrichtendefinitionsdatei vom Typ mxsd beinhaltet die logische Struktur der zugehörigen Nachricht, also das Modell bzw. Schema wie die Nachrichten aufgebaut sind. Diese werden u.a. für Mappings der Nachrichten benötigt.

3.4 nicht-invasive Integrationslösung mit dem WebSphere Message Broker

Die zu integrierende Legacy-Anwendung auf dem z/OS-Host, der CICS Catalog Manager, verwendet Flatfiles zur Datenhaltung. Abbildung 5 zeigt den in drei Phasen eingeteilten Ablauf der MB-basierten Integrationslösung.

Sobald in Phase 1 neue Produkte, Preise oder Warenbestände auftreten, wird eine solche

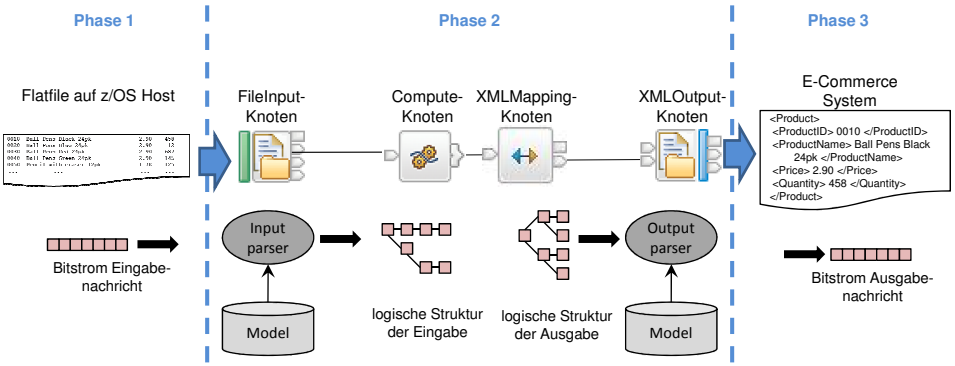

Abbildung 5: Integration von Daten einer Legacy-Anwendung in eine moderne E-Commerce-Suite

Datei geschrieben, bzw. verändert. Da die Legacy-Anwendung selbst keine Nachrichten an den Broker schicken kann, muss der Broker so konfiguriert werden, dass er Nachrichtenflussaktivitäten selbständig initiiert, sobald eine Datei in einem zu überwachendem Verzeichnis erstellt, bzw. verändert wird.

In Phase 2 werden die zu transformierenden Nachrichten als Bitstream aus einer Eingabe-Datei in den Broker eingelesen. Das geschieht mittels eines Input-Knotens, der die eigentliche Nachricht empfängt und gemäß der definierten Input Parsing-Eigenschaften in ihre logische Struktur zerlegt und in eine interne (Broker-) Darstellung überführt. Anschließend kann die Nachricht vom untergeordneten Fluss verarbeitet werden. Inhaltlich stimmt die Baumstruktur zwar mit dem eingelesenen Bitstrom überein, jedoch lässt sie sich so innerhalb des Nachrichtenflusses leichter bearbeiten.

Da die Flatfiles keinen selbstdefinierenden Aufbau besitzen, muss in einer initialen Phase erst die logische Struktur analysiert und modelliert werden. Das geschieht durch so genannte „message sets“. Bestandteile eines „message sets“ sind die „message definition files“ die man zum Beispiel aus einer XML-Schema oder C-header Datei erzeugen kann. In ihr sind z.B. die Elemente und Attribute sowie ihre Kardinalitäten und logische Anordnung (Zeilen, Spalten, Tree, Subtree,...) der einzulesenden Nachricht enthalten. Damit können die Nachrichten vom Broker „verstanden“ und zur Weiterverarbeitung genutzt werden. Die so generieten „message sets“ werden über den empfangenen Bitstream der Flatfiles gelegt und entsprechend der Definitionen des „message sets“ interpretiert.

In der Palette stehen stehen dem Entwickler bereits zahlreiche integrierte Knoten zur Verfügung. Neben der Ein- und Ausgabeverarbeitung stellen weitere Knoten auch Funktionen für die Bearbeitung, Entscheidungshilfen, Fehlerbehandlung, Sortierung, Umwandlung bis hin zu Berichtsfunktionalitäten zur Verfügung. Existieren für spezielle Aufgaben keine Knoten, so kann der Broker mittels benutzerdefinierter Knoten erweitert werden. Für diesen Zweck wird vom MB eine API für die Programmiersprachen C und Java bereitgestellt.

Zur Lösung des Integrationsproblems wurde mit dem MB Toolkit ein einfacher Nachrichtenfluss modelliert. Wie in Abbildung 4 und 5 zu sehen, besteht der Nachrichten-

fluss in diesem Fall aus vier Nachrichtenflussknoten und Konnektoren zwischen diesen. Die Eigentliche Transformation der Katalogdaten findet im Compute-Knoten statt. Hier können neue Informationen, die aus anderen Datenbanken oder Quellen stammen, hinzugefügt oder das Format der Ausgabenachricht angepasst werden. Im konkreten Fall ist eine Formatumwandlung der Katalogdaten in das Intershop Enfinity Suite proprietäre XML-Format nötig. Gleichzeitig werden die Produkte des CICS Catalog Managers mit passenden Bildern, Versandinformationen und Gruppenzuordnungen versehen. Anschließend wird mit dem XMLMapping-Knoten aus der inneren logischen Struktur Intershop Enfinity Suite konformer XML-Code generiert. Als Output-Knoten wird ein FileOutput-Knoten verwendet, der seinen integrierten XML-Parser einsetzt um die eigentliche XML-Datei zu erzeugen. Diese XML-Datei wird letztendlich in Phase 3 auf dem Intershop Enfinity System abgelegt. Das Intershop Enfinity System registriert in zyklischen Zeitabständen Veränderungen in den Katalogdateien und pflegt sie nach Einrichtung eines periodisch arbeitenden Import-Jobs in das laufende System ein. Nun sind die Artikel aus dem CICS Catalog Manager im Webshop der modernen E-Commerce-Software verfügbar.

4 Zusammenfassung und Ausblick

MOM ermöglichen im Vergleich zu anderen Integrationsansätzen eine leichte Integration von Legacy-Anwendungen in neue Anwendungssysteme. In diesem Beitrag wurde ein Lösungsweg aufgezeigt, um eine flatfile-basierte Alt-Anwendung mit einem modernen Online Shop System zu integrieren, ohne dabei in die originären Anwendungen einzugreifen. Auf Message Brokern basierende Integrationsprodukte bieten im Allgemeinen mehr als nur den Broker. So existieren in vielen Fällen standardisierte, anwendungsspezifische Adapter zur Anbindung gängiger kommerzieller Systeme. Moderne E-Commerce Suiten verfügen meist von Haus aus über umfassende Möglichkeiten für die Integration anderer Systeme. So könnte man in einer alternativen Lösung auch die Kommunikation zu E-Commerce Systemen mittels SOAP und Web Services untersuchen.

Bei der konkreten Realisierung mit dem MB im Integrationslabor, wurde Wert auf ein nicht-invasives Vorgehen gelegt. Alt-Anwendungen können somit inkrementell in neue, zeitgemäße Anwendungssysteme integriert werden, ohne dass die Notwendigkeit besteht, den Anwendungscode verstehen oder gar verändern zu müssen. Der MB dient hier lediglich als eine Art Vermittler von Informationen zwischen Quell- und Zielsystem. Damit die Nachrichten vom Zielsystem auch verstanden werden können, werden sie nach einem Regelwerk umgewandelt, bzw. es wird zusätzliches Wissen hinzugefügt.

Mit diesem Integrationsansatz können viele Arten von Legacy-Anwendungen und -Daten, unabhängig von Umgebung, Programmiersprachen oder Schnittstellen, modernisiert werden. Üblicherweise ist ein zentral eingesetzter Message Broker für geschlossene und lokale Systeme ausreichend. Offene und verteilte Umgebungen verlangen jedoch ein höheres Maß an Skalierbarkeit, Effizienz, Fehlertoleranz und Flexibilität. Hier kann eine verteilte Organisation von Message Brokern notwendig sein, die es ermöglicht, Nachrichten über mehrere Message Broker hinweg zum Empfänger zu leiten.

Das nachrichtenbasierte Integrationspattern ist jedoch nicht für jede Situation geeignet, da es nur die Datenströme der Flatfiles interpretier- und änderbar macht. Wird zum Beispiel für die zu ändernde Information ein Informationsinhalt von einer nicht in dem Datenstrom enthaltenen Variablen benötigt, welcher aber nicht im eingelesenen Datenstrom enthalten ist, so ist diese Art der Anwendungsintegration nur eingeschränkt nutzbar.

Das dargestellte Integrationspattern bildet eine Möglichkeit, Anwendungen mit möglichst geringem Aufwand und keinem Eingriff in Bestandssysteme zu integrieren. In diesem Zusammenhang werden weitere solcher Patterns im Integrationslabor untersucht, um ein möglichst breites Spektrum an EAI- und Workflowlösungen abzudecken. Insbesondere wird hier neben Software unterschiedlicher Hersteller auch ausgesprochener Wert auf eigene Entwicklungen bzw. Weiterentwicklung bestehender Ansätze gelegt.

Literatur

[Amb98] S. W. Ambler. *Building Object Applications that Work: Your Step-By-Step Handbook for Developing Robust Systems with Object Technology*. Cambridge University Press, 1998.

[BS95] Michael L. Brodie und Michael Stonebraker. *Migrating Legacy Systems. Gateways, Interfaces, and the Incremental Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[Hal08] James A. Hall. *Accounting Information Systems*. International Student Edition, 6th ed. South-Western College Pub, Cincinnati, Ohio, USA, 2008.

[Her03] Michael J. Hernandez. *Database Design for Mere Mort: A Hands-On Guide to Relational Database Design*. Addison-Wesley Professional, 2nd edition. Auflage, 2003.

[HW03] Gregor Hohpe und Bobby Woolf. *Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, Boston, MA, USA, October 2003.

[IBM07] IBM Corporation. WebSphere Message Broker Dokumentation, 2007. Abruf: 2009-06-10.

[IBM08] IBM Corporation. WebSphere Message Broker Installation Guide Version 6.1, 2008. Abruf: 2009-06-10.

[JRHD00] Matjaz B. Juric, Ivan Rozman, Marjan Hericko und Tomaz Domajnko. Integrating legacy systems in distributed object architecture. In *ACM SIGSOFT Software Engineering Notes vol*, Jgg. 25. ACM Press, 2000.

[LB08] Thilo Liedloff und Heiko Bromberger. Informationen für morgen aus Systemen von gestern? Der IBM Mainframe im Mittelpunkt zentraler Datenhaltung im Jahr 2020. In Frank Keuper und Fritz Neumann, Hrsg., *Wissens- und Informationsmanagement: Strategien, Organisation und Prozesse*, Seiten 569–596. Gabler, Wiesbaden, 2008.

[Mic02] Microsoft Press. *Computer-Lexikon mit Fachwörterbuch*. Ausgabe 2002. Microsoft Press Deutschland, Unterschleißheim, 2002.

[SPL03] R. C. Seacord, D. Plakosh und G. A. Lewis. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. Addison-Wesley Professional, 2003.