Integrationsprojekte auf Basis von EAI-Technologie

Architektur und Entwicklung

Peter Buhrmann, Jan-Peter Richter

sd&m AG, software design & management,
Lübecker Str. 1, 22087 Hamburg

Abstract: Enterprise application integration is often described as the use of some toolset that enables the rapid integration of legacy, packaged, and new applications into new business solutions and thus providing more flexibility. This paper is based on sd&m experiences in a project to integrate two packaged applications from SAP and Siebel. We present all architectural aspects of the developed solution with the main focus on the technical software architecture. We show that EAI toolsets provide technical software for the integration but error handling, process control and more important business logic has to be built within the project.

1 Einleitung

Unternehmen verfügen heute über eine große Zahl von Anwendungen. Dabei handelt es sich sowohl um Standardsoftware als auch um Individualsoftware, die speziell auf die Belange des Unternehmens abgestimmt ist. Die Anwendungen werden in heterogenen Systemumgebungen betrieben. In dieser gewachsenen Landschaft existieren vielfältige Schnittstellen zwischen den Anwendungen, das Abschalten oder Austauschen einer Anwendung kann weitreichende Folgen haben. Die häufig nicht mehr überschaubare Darstellung der Anwendungen eines Unternehmens mit ihren Schnittstellen zueinander hat den Begriff des „Schnittstellen-Spagetti“ geliefert. Enterprise Application Integration verspricht Abhilfe, soll für mehr Transparenz sorgen und eine einfachere Integration der Anwendungen ermöglichen.

Die durch EAI aufgebauten Erwartungen sind groß: mehr Flexibilität, mehr Geschwindigkeit, bessere Wartbarkeit, geringere Abhängigkeiten, geringere Kosten etc. Alle diese Merkmale sind ein Maß für die Qualität der Software-Architektur [BCK98]. Dieses Papier beleuchtet die Software-Architektur einer EAI-Lösung genauer und zeigt auf, was die EAI-Werkzeuge mitbringen und was in den Integrationsprojekten geleistet werden muss. Als Beispiel dient die Integration der Branchenlösung von SAP für die Energiewirtschaft IS-U mit der CRM-Lösung von Siebel.

2 Begriffsklärung

In diesem Artikel wird unter Enterprise Application Integration die technisch und fachlich standardisierte Integration von Anwendungen verstanden (vgl. [Li00], [RW99]).

Hierzu werden Werkzeuge verwendet, die aus den in Bild 1 dargestellten Komponenten bestehen.

Bild 1: Komponenten von EAI-Werkzeugen nach [RW99]

<table>
<tr><td colspan="2">Workflow
Steuerung und Verwaltung
des Transformationsflusses</td><td rowspan="3">Entwicklungs-umgebung
Prozess-modellierung, Transformation, Spezifikation und Schnittstellen-entwicklung</td><td rowspan="3">System-administration
Zuverlässigkeit, Verfügbarkeit, Skalierung und Überwachung</td></tr>
<tr><td colspan="2">Transformation
Transformation, Identifizierung, Validierung, Synchronisierung, Transaktionsunterstützung</td></tr>
<tr><td>Kommunikation
Kommunikation, Adressierung, Lieferung, Sicherheit</td><td>Konnektoren
Schnittstellenübersetzung und Metadatenabbildung</td></tr>
</table>

Die Kommunikationskomponente ermöglicht den gesicherten Datenaustausch zwischen den beteiligten Systemen (z. B. MQSeries, Rendezvous, etc.). Die Konnektoren realisieren den Zugriff auf die einzelnen Anwendungen (z. B. auf eine Siebel API oder ein XML-Dokument). In der darüber liegenden Schicht erfolgt die Transformation der Daten sowie die Ablauflogik zur Verarbeitung der Nachrichten. Transaktionsübergreifende Abläufe, die über einen längeren Zeitraum laufen, werden durch eine eigenständige Workflow-Schicht abgebildet. Quer zu diesen Komponenten liegen Entwicklungs- und Administrationsumgebung.

Unter einer EAI-Lösung soll in diesem Beitrag die Gesamtheit aller Softwarebausteine verstanden werden, die in einem EAI-Projekt zur Realisierung einer Systemkopplung verwendet werden. Idealerweise besteht die EAI-Lösung ausschließlich aus solchen Bausteinen, die direkt einer EAI-Produkt-Suite entnommen werden oder mit Hilfe von Werkzeugen einer EAI-Produkt-Suite generiert werden.

Die Schnittstellen[1] zu den Nachbarsystemen sind ein Bestandteil des Informationssystems, so wie seine Benutzungsschnittstelle oder seine Datenzugriffsschicht. Insofern ist auch die EAI-Schnittstelle ein Teil des Informationssystems und es kann auf die Überlegungen zur Architektur von Informationssystemen zurückgegriffen werden. Im Rahmen dieses Papiers werden die in [DS00] und [Si02] eingeführten Begriffe verwendet, die im Folgenden kurz wieder gegeben wird.

Jedes Stück Software gehört genau zu einer von fünf Kategorien. Es kann sein:

- *0-Software:* unabhängig von Anwendung und Technik
- *A-Software:* bestimmt durch die Anwendung, unabhängig von der Technik
- *T-Software:* unabhängig von der Anwendung, bestimmt durch die Technik
- *AT-Software:* bestimmt durch Anwendung und Technik
- *R-Software (eine besondere Form der AT-Software):* zur Transformation fachlicher Objekte in eine externe Repräsentation

[1] Für eine Verbindung zweier Anwendungssysteme inklusive Middleware und Ablaufsteuerung wird zuweilen ebenfalls die Bezeichnung "Schnittstelle" verwendet. In diesem Beitrag soll für dieses umfassende Konstrukt statt dessen die Bezeichnung "Systemkopplung" benutzt werden.

Die Systemarchitektur besteht aus der technischen Infrastruktur und der Software-Architektur. Die Software-Architektur eines Anwendungssystems wird untergliedert in die Anwendungsarchitektur und die Technikarchitektur.

- Die Architektur der technischen Infrastruktur (*TI-Architektur*) beschreibt die physischen Geräte (Rechner, Netz), die darauf installierte Systemsoftware (Betriebssysteme, Datenbanken, Middleware, etc.) und das Zusammenspiel von Hard- und Software.
- Die Anwendungsarchitektur (*A-Architektur*) ist Teil der Software-Architektur und befasst sich mit den fachlichen Elementen der Anwendungsdomäne (z. B. Kundenverwaltung, Kunde, Konto).
- Die Technikarchitektur (*T-Architektur*) ist Teil der Software-Architektur und beschreibt auf der obersten Ebene die klassischen drei Schichten (Benutzungsschnittstelle, Anwendungskern, Datenzugriffsschicht), darunter die grundlegenden Basisklassen und Schnittstellen. Die A-Architektur füllt die dafür vorgesehenen Freistellen der T-Architektur mit anwendungsspezifischen Softwarebausteinen.

3 Architektur der EAI-Lösung

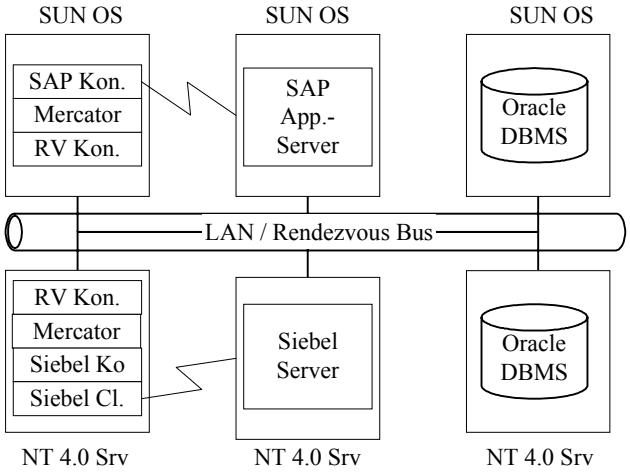
Im folgenden Abschnitt werden die verschiedenen Perspektiven der Architektur zur Integration der Standardsoftware SAP IS-U und Siebel dargestellt. Der Schwerpunkt liegt auf der T-Architektur. Hierfür stellen wir eine idealtypische der realisierten T-Architektur gegenüber. Die Unterschiede dieser beiden Architekturen liegen in den Rahmenbedingungen durch die anzubindenden IT-Systeme sowie in den Beschränkungen des verwendeten EAI-Werkzeugs begründet.

Die Anwendungslandschaft ist idealerweise so gegliedert, dass die Anwendungssysteme des Unternehmens sinnvoll geschnittene Komponenten für die Unterstützung der Geschäftsprozesse darstellen. Die meisten Geschäftsprozesse werden dann überwiegend innerhalb der Komponenten, d.h. innerhalb der Anwendungssysteme ablaufen. Die Interaktion der Komponenten erfolgt über eine lose Kopplung, die von den Details der Nachbarkomponente vollkommen abstrahiert, indem die Komponenten einander fachlich geeignete Anwendungsschnittstellen bereitstellen. Die EAI-Lösung wird verwendet, um diese Anwendungsschnittstellen miteinander zu verbinden.

Oft ist jedoch die Anwendungslandschaft historisch gewachsen. Die Aufteilung in Komponenten entspricht daher nicht den Geschäftsprozessen. Strukturelle Defizite in der Anwendungslandschaft können aber nicht allein durch eine neue Integrationstechnologie geheilt werden. Über reine Integrationsprojekte hinaus muss die Anwendungslandschaft kontinuierlich geplant und fortentwickelt werden.
Die fachlichen Fragen von Integrationslösungen werden bestimmt durch die beteiligten Anwendungen, die Anwendungslandschaft und die anwendungsübergreifenden Geschäftsprozesse des Unternehmens. Eine Anwendungsarchitektur für EAI-Lösungen existiert nicht, da eine EAI-Lösung allein keine Anwendung darstellt. Eine EAI-basierte Systemkopplung ist im Normalfall kein Element einer unternehmensweiten Anwendungsarchitektur, da sie nur das "wie" nicht aber das "was" einer Verbindung der Anwendungen beschreibt.

Die TI-Architektur einer EAI-Lösung wird im Wesentlichen durch die Auswahl der eingesetzten EAI-Werkzeuge sowie der anzuschließenden Systeme bestimmt. Architektonisch interessant ist in diesem Zusammenhang allein die Unterscheidung zwischen einer zentralisierten, EAI-Server-basierten Architektur ("Hub and Spoke") und einer verteilten, Kommunikationsbus-basierten Architektur. Bei der letztgenannten Architektur können auch die bereits vorhandenen Rechner benutzt werden, auf denen die Anwendungssysteme ablaufen. Eine zentralisierte Architektur hat häufig den Nachteil, dass sie schlecht skaliert und hoher zusätzlicher Aufwand für besondere Verfügbarkeitsanforderungen betrieben werden muss.

Bild 2: TI-Architektur der realisierten Lösung



Im Realisierungprojekt, das diesem Beitrag zugrunde liegt, wurden folgende EAI-Werkzeuge verwendet: MERACTOR e-Business Integration Suite 5.0 sowie als Middleware TIBCO Rendezvous 6.3. Diese Produkte zusammen mit den angebundenen Anwendungssystemen Siebel 99 und SAP IS-U bestimmen eine verteilte TI-Architektur, die in Bild 2 wiedergegeben ist. Bemerkenswert ist, dass der MERCATOR-Konnektor für Siebel 99 an einen Siebel Client gebunden wird, der seinerseits über eine Siebel-interne Kommunikationsinfrastruktur mit dem Siebel-Server verbunden ist. Der MERCATOR SAP-Konnektor wiederum verwendet eine SAP-interne Kommunikationsinfrastruktur, um das SAP-System anzusprechen und wurde aus organisatorischen Gründen ebenfalls nicht auf dem Rechner der Anwendung installiert.

3.1 Ideale T-Architektur

Bevor die Struktur der idealen T-Architektur für eine EAI-basierte Systemkopplung dargestellt wird, sollen die dafür benötigten Bausteine kurz vorgestellt werden. Die Bausteine sind zum Teil mehrfach in einer Systemkopplung vorhanden.

Berücksichtigt werden Bausteine für eine Daten- und Prozessintegration, die im allgemeinen im Rahmen von EAI-Projekten angestrebt werden. Bausteine für eine Integration mehrerer Anwendungssysteme in einem gemeinsamen GUI, z.B. durch eine Portallösung, werden hingegen nicht weiter betrachtet.

- Middleware zur Überbrückung physikalischer Rechnergrenzen sowie der Grenzen zwischen den abgeschotteten Prozessen der verschiedenen Anwendungssysteme auf einem Rechner.
- Generische Konnektoren als T-Adapter zwischen spezifischen Anwendungssystemen und den weiteren Bausteinen der EAI-Lösung. Generisch in dem Sinne, dass sie universell für jede Anwendungsschnittstelle des betrachteten spezifischen IT-Systems eingesetzt werden können.
- Parser und Zeichenkettengeneratoren als Bestandteil der Konnektoren. Parser und Zeichenkettengeneratoren bilden einen Teil der Konfiguration des generischen Konnektors und tragen dazu bei, ihm damit eine *spezifische* Anwendungsschnittstelle zu verleihen. So realisiert der Konnektor einen Adapter im Sinne von [DS00]. Ein Parser oder Zeichenkettengenerator ist immer dann notwendig, wenn die Schnittstelle des Anwendungssystems strukturierte Daten als strukturierte Zeichenkette übergibt bzw. entgegennimmt. Dies ist in der Praxis häufig der Fall. Ein prominentes Beispiel ist das SAP IDOC als Zeichenkette mit Feldern fester Länge und variabler Segmentstruktur. Ein solcher Parser oder Zeichenkettengenerator enthält damit Wissen über das Datenmodell der Anwendung und ist somit keine reine T-Software mehr. Vielmehr handelt es sich bei dieser Mischung aus A- und T-Software um einen Baustein zur Umwandlung von Repräsentationen, also R-Software. Zur Erzeugung von Parsern und Zeichenkettengeneratoren verfügen EAI-Werkzeugsammlungen typischerweise über Generatoren, die diese Bausteine aus Syntaxbeschreibungen generieren. Die Syntaxbeschreibungen selbst werden im Idealfall vom anzubindenden IT-System in Form exportierter Metadaten geliefert. In der Praxis müssen die Syntaxbeschreibungen jedoch zum Teil händisch erstellt werden, was den Aufwand von EAI-Projekten erhöht. Im hier zunächst dargestellten Idealfall besteht keine Notwendigkeit für den Einsatz dieser Bausteine.
- Logbuchverwalter zur Führung von Logbüchern, die die Abläufe in der Systemkopplung dokumentieren. Die Aufgabe des Logbuchverwalters variiert in Abhängigkeit von den bereits vorhandenen Möglichkeiten des angebundenen Systems und den Anforderungen an Recovery-Fähigkeit und Nachvollziehbarkeit des Datenverkehrs. Bestehen hohe Anforderungen und verfügt das Anwendungssystem über keine eigene Logbuchführung, ist es notwendig, alle relevanten Abläufe bis hin zu den übertragenen Nutzdaten zu protokollieren. Die Logbuchverwaltung sollte dezentral in der EAI-Infrastruktur positioniert werden, da sie gerade in solchen Fällen zur Fehlerverfolgung wichtig ist, in denen Kommunikationswege gestört sind.
- Fehlerbehandlung für Fehler innerhalb der EAI-Lösung. Aufgabe der Fehlerbehandlung ist die geordnete Weitergabe von Fehlermeldungen an das Betriebspersonal sowie die automatische Administration der weiteren Bausteine, falls die Fehlersituation dies verlangt. So kann es beispielsweise notwendig sein, die weitere Verarbeitung anzuhalten, wenn ein Strom von Datenelementen zwingend in der richtigen Reihenfolge verarbeitet werden muss. Die Fehlerbehandlung wird typischerweise ihrerseits in mehrere Teilbausteine gegliedert sein, die teils zen-tral, teils dezentral positioniert werden sollten. Aus Gründen der Einfachheit soll hier die Fehlerbehandlung als monolithische, zentralisierte Komponente betrachtet werden.

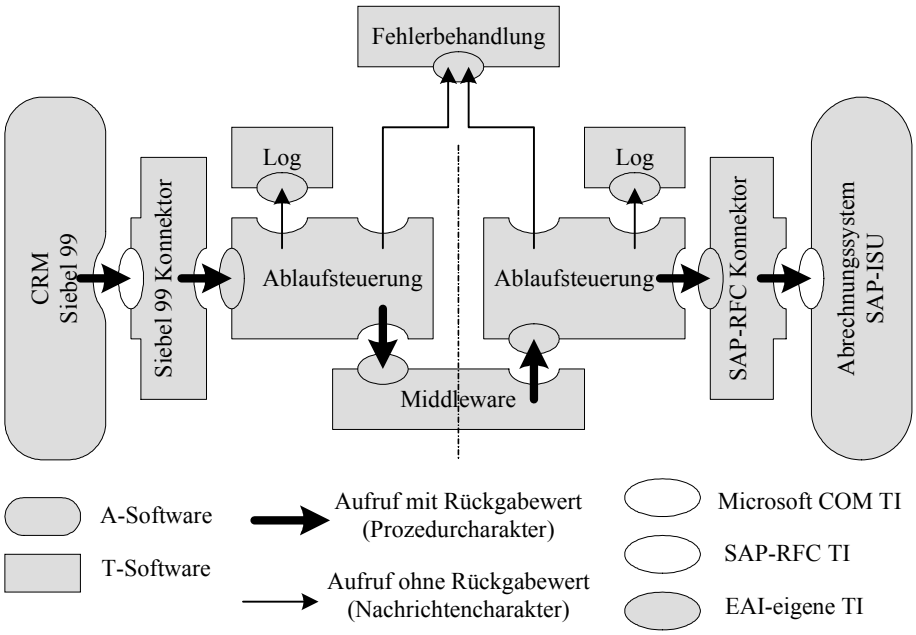
- Ablaufsteuerung zur Steuerung des Zusammenspiels der genannten Bausteine sowie zur Realisierung der Ablauflogik der Systemkopplung. Da die Systemkopplung typischerweise selbst auf mehrere Rechner verteilt ist, ist auch die Ablaufsteuerung im Allgemeinen verteilt.

Im hier zunächst dargestellten Idealfall wird ein Mapping-Baustein zur Abbildung der Datenstrukturen nicht benötigt, da die importierte und exportierte Schnittstelle der Anwendungssysteme bis auf Aspekte der technischen Infrastruktur übereinstimmen. Die Abbildung auf TI-Ebene wird durch die generischen Konnektoren implizit durchgeführt. Die Abbildung wird vollständig durch die Metadaten der Schnittstellen definiert, mit der die Konnektoren konfiguriert werden.

Anhand des konkreten Beispiels der Kopplung eines Siebel 99 CRM-Systems an ein SAP-basiertes Abrechnungssystem soll nun der Datenaustausch in einer Systemkopplung mit idealer T-Architektur dargestellt werden. Der fachliche Hintergrund ist ein Geschäftsprozess, bei dem ein Vertragsabschluß im CRM-System erfasst wird. Die Daten des neu abgeschlossenen Vertrags werden daraufhin zusammen mit den Kundendaten in das Abrechnungssystem übertragen. Der Geschäftsprozess erfordert, dass nach Anlage der Stammdaten im Abrechnungssystem spezielle Vertragsdaten an das CRM-System zurückgemeldet werden. Der Prozess hat daher den Charakter eines RPC (Remote Procedure Call).

Der Vorgang beginnt damit, dass im CRM-System für einen neuen Vertrag die importierte Schnittstelle *Vertrag_Neuanlage()* am EAI-Anwendungsadapter aufgerufen wird. Auf Ebene der technischen Infrastruktur geschieht dies durch einen Microsoft-COM Aufruf, für den der Anwendungsadapter sich zuvor am CRM-System registriert hat. Der Anwendungsadapter wandelt den COM-Aufruf in einen Aufruf der EAI-eigenen technischen Infrastruktur um. Dabei werden die Aufrufparameter ebenfalls umgewandelt: von der abstrakten Syntax der COM-Infrastruktur zur abstrakten Syntax des EAI-Werkzeugs. So stehen den EAI-Werkzeugen die Zugriffsfunktionen zu den Feldern und Segmenten der strukturierten Datentypen bzw. zu den Aufrufparametern zur Verfügung. Der Anwendungsadapter gibt den Aufruf daraufhin an die Ablaufsteuerung weiter. Diese ruft den Logbuchbaustein sowie den Middleware-Baustein auf, um den Aufruf auf den Zielrechner zu übertragen. Dort wird er wiederum der Ablaufsteuerung übergeben, die ihrerseits den Logbuchbaustein sowie den Anwendungsadapter des SAP Systems anspricht. Dieser Anwendungsadapter wandelt die EAI-eigene abstrakte Syntaxdarstellung der Datenstrukturen in die des IT-Systems – hier im Idealfall SAP-RFC – um und führt den Aufruf im Abrechnungssystem durch. Schließlich liefert dieser SAP-RFC-Aufruf eine Menge von Rückgabewerten, die den umgekehrten Weg bis in das aufrufende CRM-System finden. Die Abläufe sind dabei entsprechend; die Systemkopplung berücksichtigt bei ihrer Arbeit jedoch, dass es sich dabei um Rückgabedaten des zuvor übertragenen Aufrufs handelt.

Bild 3: Ideale T-Architektur



Der Ablauf, der hier für *einen* RPC-Aufruf dargestellt wurde, läuft nun zeitlich verschränkt parallel für mehrere Verträge ab. Die Ablaufsteuerung ist so konfiguriert, dass sie Aufrufe und Rückmeldungen einander korrekt zuordnet und auch im Falle eines Systemzusammenbruches wieder aufsetzen kann. Gleichzeitig betreibt sie eine Flusskontrolle, um die Warteschlangen vor den einzelnen Verarbeitungsschritten und im Zielsystem nicht zu lang werden zu lassen.

Die hier skizzierte ideale T-Architektur besteht ausschließlich aus Bausteinen, die eine vollständige EAI-Werkzeugsammlung liefern sollte. Um eine solche Systemkopplung zu entwickeln, ist nur eine geringfügige Konfiguration der Bausteine notwendig. Falls die zu verbindenden Systeme Metadaten über die Schnittstellen in geeigneter Form exportieren, kann die Systemkopplung sogar ohne jedes Wissen um die fachlichen Zusammenhänge und Datenmodelle erstellt werden, wie dies von den Herstellern der EAI-Werkzeuge gern suggeriert wird.

Die ideale T-Architektur besteht damit ausschließlich aus T-Bausteinen und ggf. R-Bausteinen. Dabei kann die Konfiguration der R-Bausteine im Idealfall generiert werden. Anwendungswissen bleibt daher in den Komponenten der Anwendungslandschaft gekapselt.

3.2 Realisierte T-Architektur

Das oben dargestellte Ideal-Szenario basiert auf Annahmen und Randbedingungen, die in der Praxis selten zutreffen.

3.2.1 Mapping und Schnittstellendatenmodell

Zunächst stimmen typischerweise die Datenmodelle der anzubindenden Systeme und damit die Aufrufparameter der importierten und exportierten Schnittstelle nicht überein. Dieses Problem bewältigen die EAI-Werkzeuge durch die Bereitstellung eines generischen Mapping-Bausteins. Er wird konfiguriert mit zwei abstrakten Syntaxbeschreibungen für eingehende und ausgehende Datenstrukturen sowie einer Abbildungsvorschrift. Diese bestimmt die Struktur (z.B. Anzahl der Segmente) der Ausgangsdatenstruktur sowie die Werte der Felder in Anhängigkeit der Eingangsdatenstruktur. Die Ausgangsdatenstruktur wird zunächst wieder in der Darstellung der EAI-internen abstrakten Syntax erzeugt.

Die Konfiguration des Mappings erfordert in der Praxis erhebliches fachliches Wissen sowie Wissen um die technischen Aspekte der Datenmodelle der anzubindenden Systeme. Ein konfigurierter Mapping-Baustein enthält damit Anwendungslogik. Er ist wiederum ein Baustein zur Umwandlung von Repräsentationen, also R-Software. Diese R-Software kann im Gegensatz zu den Parsern und Zeichenkettengeneratoren nicht generiert werden, da sie nicht nur auf der syntaktischen, sondern auch auf der semantischen Ebene arbeitet. Die Metadaten-Darstellung der Datenmodelle enthalten typischerweise zu wenig oder gar keine Information über die Semantik der Datenmodelle. Oft genug liegen diese semantischen Metadaten ausschließlich in Form von nicht-formalisierten Systemdokumentationen vor.

Häufig besteht der Anspruch, dass eine Systemkopplung nicht isoliert betrachtet werden soll. Vielmehr wird angestrebt, die exportierte Schnittstelle eines Systems mehreren anderen Systemen zur Verfügung zu stellen. Dann stellt sich die Frage, auf welcher Seite der verteilten Systemkopplung die Abbildung der Datenmodelle vorgenommen werden soll. Gleichgültig auf welcher Seite der Systemkopplung ein solches einfaches Mapping positioniert wird, es besteht immer die Notwendigkeit, Informationen über das Datenmodell eines entfernten Systems lokal abzulegen. Das bedeutet, dass Strukturinformationen einzelner Anwendungen in der Landschaft verstreut werden müssen[2].

Wir schlagen statt dessen vor, die Abbildung der Datenstrukturen in zwei Teilschritten zu vollziehen. Dabei wird die Datenstruktur des aufrufenden Systems zunächst in ein systemneutrales Schnittstellendatenmodell umgewandelt und über die Middleware zum Zielrechner transportiert. Die Abbildung der Datenstruktur in das Datenmodell des Zielsystems geschieht dann lokal beim Zielsystem. Dieses Vorgehen eröffnet die Möglichkeit, durchgängig bei allen Kommunikationen über die Middleware solche Datenstrukturen zu verwenden, wie sie den Bedürfnissen der gesamten Anwendungslandschaft entsprechen. Dies entkoppelt die Systeme an einer wohldefinierten Stelle und in wohldefinierter Weise. Das Vorgehen verlangt jedoch die Definition eines entsprechenden Schnittstellendatenmodells, das erneut fachliches Wissen – jedoch kein zusätzliches systemtechnisches Wissen – erfordert.

[2] Dies gilt nicht für vollständig zentralisierte EAI-Lösungen.

3.2.2 Ablaufsteuerung des RPC

Das verwendete EAI-Werkzeug MERCATOR verfügt nicht über einen generischen Baustein zum Aufbau von Ablaufsteuerungen. Daher konnte eine Ablaufsteuerung für die RPC-Aufrufe mit den oben geforderten Eigenschaften nicht realisiert werden. Vielmehr konnte nur durch die Ausnutzung von "Seiteneffektprogrammierung", sequenzieller Abarbeitung und bedingter Auswertung von Teiltermen der Abbildungsvorschrift der Mapping-Baustein ansatzweise zur Ablaufsteuerung herangezogen werden.

Diese Beschränkung hatte weitreichende Konsequenzen. So musste die Systemkopplung nachrichten-orientiert strukturiert werden, d.h. Aufrufe und Rückantworten werden nicht mehr einander zugeordnet betrachtet. Vielmehr sendet das CRM-System eine Folge von Nachrichten der Bedeutung *Vertrag_Neuanlage.request* und das Abrechnungssystem schickt davon unabhängige Nachrichten der Bedeutung *Vertrag_Neuanlage.response*. Das CRM-System ist damit prinzipiell dafür verantwortlich, die Antworten mit den Anfragen in Übereinstimmung zu bringen. Dadurch wird T-Funktionalität, die sinnvollerweise in der EAI-Lösung realisiert werden sollte, in die Anwendung verlagert. Im vorliegenden Fall war es allerdings fachlich vertretbar, auf einen derartigen Abgleich zu verzichten.

Aus ähnlichen Gründen wurde darauf verzichtet, die exportierte Schnittstelle des SAP-basierten Abrechnungssystems mit SAP-RFC zu realisieren. Statt dessen wurde SAP ALE so eingesetzt, dass die beiden Nachrichtentypen *Vertrag_Neuanlage.request* und *.response* als zwei verschiedene IDOCs empfangen bzw. versendet werden.

3.2.3 Implementation der Anwendungsschnittstelle

Über die Realisierung der exportierten Anwendungsschnittstelle auf Ebene der TI-Architektur hinaus war es notwendig, eine Verarbeitungslogik zu entwickeln, die die gewünschte Funktionalität implementiert. Eine fachlich geeignete Schnittstelle am Abrechnungssystem war nämlich zunächst nicht vorhanden.

Konkret muss bei jeder Neuanlage eines Vertrages geprüft werden, ob der zugeordnete Kunde bereits vorhanden ist. Ist dies nicht der Fall, so muss der Kunde ebenfalls neu angelegt werden. Nach der eigentlichen Neuanlage des Vertrags müssen dann Vertrag und Kunde sowie weitere Objekte des Abrechnungssystems verknüpft werden.

Die Implementation dieses Anwendungsbausteins erfolgte architektonisch korrekt innerhalb des SAP-Systems, also getrennt von der EAI-Lösung. Die Notwendigkeit derartiger Systemergänzungen zeigt jedoch, dass die Verwendung von EAI-Werkzeugen allein keine Anwendungsintegration erlaubt. Vielmehr ist bei der Planung eines Integrationsprojektes zu prüfen, ob die fachlich geeigneten Schnittstellen überhaupt von den vorhandenen Anwendungssystemen bereitgestellt bzw. benutzt werden. Ist dies nicht der Fall, so ist bei der Projektplanung erheblicher zusätzlicher Aufwand sowie die notwendige Verfügbarkeit entsprechend ausgebildeter Mitarbeiter zu berücksichtigen, um die fehlenden Anwendungsbausteine konventionell zu entwickeln.

3.2.4 Einsatz zusätzlicher generierter Parser und Zeichenkettengeneratoren

Die Werkzeugsammlung MERCATOR ist so strukturiert, dass alle Konnektoren, d.h. die generischen Anwendungsadapter sowie die Adapter zu den Middleware-Bausteinen die Datenstrukturen nicht in einer EAI-internen abstrakten Syntax darstellen. Vielmehr werden alle Datenstrukturen immer in einer konkreten Syntax dargestellt, die jedoch von Konnektor zu Konnektor differiert. So wird der Siebel 99-Konnektor über eine Liste von Komma-separierten Feldern angesprochen, während der Konnektor zur TIBCO-Middleware eine XML-artige Darstellung verlangt. Die Konnektoren verfügen dazu jeweils über interne Parser bzw. Zeichenkettengeneratoren, die nicht weiter konfiguriert werden müssen.

Darüber hinaus sind die generierten Parser, die konfigurierten Mapping-Bausteine und die generierten Zeichenkettengeneratoren jeweils zu einem Baustein-Tripel fest verbunden, d.h. eine Abbildung von Datenstrukturen erfolgt immer von einer konkreten Syntax in eine konkrete Syntax.

Diese Beschränkung führte dazu, dass für die Darstellung der Datenstrukturen des Schnittstellendatenmodells ein zusätzlicher Generator entwickelt werden musste. Dieser hat die Aufgabe, die Metadaten des Schnittstellendatenmodells in die Definition der XML-artigen konkreten Syntax des TIBCO-Konnektors zu überführen.

Außerdem führt die genannte Beschränkung dazu, dass die Datenstrukturen in der Verarbeitungskette aus Mapping-Bausteinen und Middleware-Bausteinen mehrfach zwischen abstrakter und konkreter Syntax abgebildet werden: Zunächst generiert der Mapping-Baustein die XML-artige Zeichenkette, die anschließend vom integrierten Parser im TIBCO-Konnektor wieder geparst werden muss. Das entsprechende wiederholt sich auf der Empfängerseite, wo die Nachricht zunächst vom Middlewarebaustein in dessen konkrete, XML-artige Syntax konvertiert wird, um daraufhin vom Mapping-Baustein für die weitere Bearbeitung wieder geparst zu werden.

3.2.5 Darstellung der realisierten T-Architektur

Die dargestellten Beschränkungen führten zu einer realisierten T-Architektur, die in den folgenden Abbildungen dargestellt ist.

Bild 4 zeigt zunächst eine Übersicht über die realisierte T-Architektur. Es ist erkennbar, dass die Systemkopplung aus zwei zueinander gegenläufigen Hälften besteht, die die beiden Richtungen des ursprünglichen RPC-Aufrufs realisieren.

Bild 5 schließlich stellt im Detail das obere rechte Viertel der gesamten Struktur dar. Es wird hervorgehoben, welche Bausteine nicht direkt durch Verwendung vorgefertigter EAI-Werkzeuge realisiert werden konnten. Sie wurden separat entwickelt, wobei zum Teil die EAI-Werkzeuge als Basis verwendet werden mussten, um die notwendige enge Kopplung mit den anderen Bausteinen erreichen zu können. Ebenfalls dargestellt ist die Rolle der zusätzlichen Parser und Zeichenkettengeneratoren, die bei der Verwendung der EAI-Werkzeugsammlung MERCATOR unumgänglich in die Verarbeitungskette eingereiht werden.

Bild 4: Realisierte T-Architektur mit Schnittstellendatenmodell

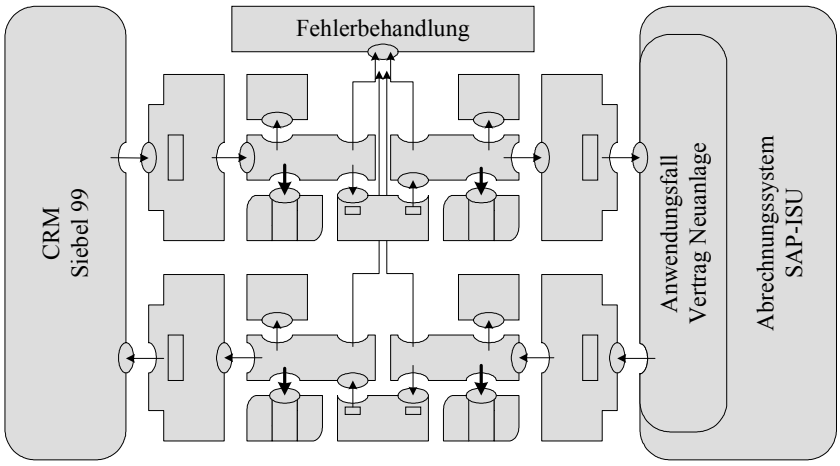
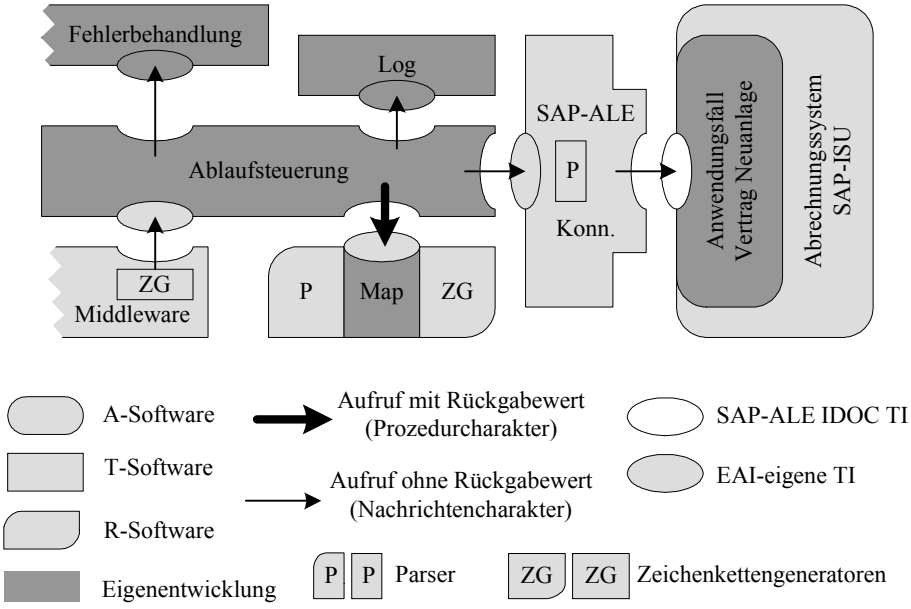


Bild 5: Realisierte T-Architektur im Detail



4 Fazit

Anwendungsintegration beginnt mit Überlegungen zur Anwendungslandschaft. Die wichtigsten Voraussetzungen für den Erfolg von Integrationsprojekten sind eine gute Architektur der Anwendungslandschaft und eine genaue Kenntnis der anwendungsübergreifenden Geschäftsprozesse. Die Auswahl der EAI-Werkzeuge ist eine Aufgabe im Rahmen des Designs der TI-Architektur. Die Werkzeuge liefern ausschließlich T-Software und beeinflussen die T-Architektur maßgeblich. Die Restriktionen der Werkzeuge führen häufig zu den aufgezeigten aufwändigen Workarounds. Die benötigte fachliche Software zur Integration, also die A-Software, wird nicht mitgeliefert; sie gehört in die Anwendungssysteme und implementiert die von den Konnektoren benötigten Schnittstellen. Spezifikation und Implementierung der fachlichen Logik sowie Konzepte zur Fehlerbehandlung und Ablaufsteuerung sind im Rahmen der Projekte zu erstellen. Der Aufwand hierfür ist keinesfalls zu vernachlässigen. Die Stärke der EAI-Werkzeuge liegt bei den erhältlichen Konnektoren, den Generatoren für Parser und Zeichenkettengeneratoren sowie in der Middleware zum Transport der Nachrichten.

Literaturverzeichnis

[BCK98] Bass, L., Clements, P., Kazman, R.: Software-Architecture in Practice. Addison-Wesley, Reading 1998

[DS00] Ernst Denert; Johannes Siedersleben: Wie baut man Informationssysteme? Überlegungen zur Standardarchitektur. Informatik-Spektrum 8/2000, S. 247 - 257, 23.08.2000

[Li00] David S. Linthicum: Enterprise Application Integration. Addison-Wesley, 2000

[RW99] Katy Ring; Neil Ward-Dutton: Enterprise Application Integration - Making the right connection. Ovum-Report, 1999

[Si02] Johannes Siedersleben: Architekturbegriffe für betriebliche Informationssysteme. Modellierung 2002: Modellierung in der Praxis - Modellierung für die Praxis; Tutzing, 25.-27. März 2002