

FACHHOCHSCHULE WEDEL

SEMINARARBEIT

in der Fachrichtung

Wirtschaftsinformatik

Thema:

Service Oriented Architectures

Einführung in die Integration verschiedener
Anwendungssysteme

-

Problematik und allgemeine Architektur

Eingereicht von: Julia Weisheitel (Mat.-Nr. 5773)
Johann-Diedrich-Möller-Str. 90
22880 Wedel
Tel. (0 4103) 8035863
Mobil: 0174-3891747
Email: julia.weisheitel@web.de

Erarbeitet im: 7. Semester

Abgegeben am: 07. Dezember 2005

Referent (FH Wedel): Prof. Dr. Sebastian Iwanowski
Fachhochschule Wedel
Feldstraße 143
22880 Wedel
Tel. (04103) 8048- 63
Email: iw@fh-wedel.de

Inhaltsverzeichnis

1. Überblick.....	1
1.1.Prozessorientierte Sicht.....	1
1.2.Integrationsebenen.....	2
1.2.1.Integration auf Datenebene.....	2
1.2.2.Integration auf Anwendungsebene.....	2
1.2.3.Integration auf Prozessebene.....	3
1.2.4.Serviceorientierte Integration.....	3
2. Probleme und Entscheidungen im Vorfeld.....	5
2.1.Middleware und EAI.....	5
2.2.Hilfestellung auf dem Weg zum Service Oriented Enterprise.....	5
3. Einführung einer SOA.....	7
3.1.Service-Modelle für Enterprise-Integration-Architekturen.....	7
3.1.1.Prozessservices.....	7
3.1.2.Koordinationservices (für Geschäftszwecke).....	7
3.2.Grundlegende Architekturkomponenten der Enterprise Integration.....	8
3.2.1.Broker.....	8
3.2.2.Orchestration.....	9
3.3.Konkrete Architekturen zur Enterprise Integration.....	10
3.3.1.Hub and Spoke.....	10
3.3.2.Messaging Bus.....	11
3.3.3.Enterprise Service Bus.....	11

Abbildungsverzeichnis

Abb. 1: Integrationsebenen.....	2
Abb. 2: Punkt-Zu-Punkt-Integration auf Anwendungsebene.....	3
Abb. 3: EAI-Architektur.....	3
Abb. 4: Integrationsschichten für die serviceorientierte Architektur.....	4
Abb. 5: Prozessservice.....	7
Abb. 6: Koordinationsservice.....	8
Abb. 7: EAI-Architektur.....	8
Abb. 8: Broker.....	9
Abb. 9: Orchestration.....	9

Abkürzungsverzeichnis

API Application Programming Interface

EAI Enterprise Application Integration

ESB Enterprise Service Bus

SOA Service Oriented Architecture

1. Überblick

Ziel der Integration ist es, zwei oder mehr Anwendungen miteinander zu verbinden. Diese Verbindung umfasst im wesentlichen drei Bereiche: den Zugang zu Daten (auch in einer Datenbank), Zugang zu fremder Geschäftslogik oder das Zusammenspiel der kompletten Anwendungen als Teil eines großen Ganzen.

Die Integrationsbestrebungen können verschiedene Ursachen haben. Zum einen entsteht ein solcher Bedarf häufig als Reaktion auf unmittelbare Anforderungen im Unternehmen. Zum anderen ergeben sich durch den Ansatz, Integration an sich vereinfachen zu wollen, Vorteile für das ganze Unternehmen. Vieles wird simpler, da man sich bereits im Vorfeld Gedanken über Auswirkungen auf andere Bereiche und das Zusammenspiel untereinander Gedanken macht. Oft lassen sich kleinere oder auch größere Teile wiederverwenden und die Effizienz verschiedener Ansätze vergleichen. Auf der anderen Seite sind mit Standardisationsbestrebungen in der Regel auch erhöhte Kosten und ein genereller Mehraufwand verbunden, da die Anwendung nicht für sich allein, sondern als Teil des Ganzen betrachtet und miteingepasst und für spätere Erweiterungen offen gehalten wird.¹

1.1 Prozessorientierte Sicht²

Integration wird heutzutage, in Anlehnung an die Entwicklung hin zur prozessorientierten Geschäftssicht, hauptsächlich prozessorientiert betrieben. Dabei treiben sich Prozessautomation und Integrationstechnologie gegenseitig voran. Es haben sich zwei Lösungsansätze zur Integration durchgesetzt.

Auf der einen Seite werden bestehende Anwendungen erweitert, in dem Unterstützung für neue Features hinzugefügt, der Geschäftsprozess als solches verändert oder auch der Zugang zu fremden Daten und/oder Geschäftslogik ermöglicht wird. Dazu ist die Schaffung eines Integrationskanals nötig. Die Schwierigkeit der Einrichtung erhöht sich zusätzlich, wenn die zu integrierenden Anwendungen auf verschiedenen Plattformen arbeiten, in einer sogenannten heterogenen Umgebung.

Bei der Überarbeitung bestehender Modelle kann man aber auch zum Schluss kommen, dass ein komplettes Redesign nötig ist, Prozesse vielleicht zusammengeschlossen werden können oder viel enger miteinander zusammenarbeiten müssen, als das bisherige Modell es erlaubt. In diesem Fall entstehen komplett neue Geschäftsprozesse. Das führt oftmals auch zu hohen Anforderungen an die technische Umgebung, die den neuen Gegebenheiten angepasst werden muss.

1 Vgl. Erl (2004), S. 284f.

2 Vgl. Erl (2004), S. 286f.

1.2 Integrationsebenen

Das übergreifende Arbeiten von Prozessen führt zu Kommunikationsbedarf. Ausgangspunkt hierbei war der reine Datenaustausch, der später erweitert wurde um die gemeinsame Nutzung der Geschäftslogik der Anwendungen. Diese Ebenen stellen die traditionelle Integration dar (s. Abb. 1). Im Laufe der Entwicklung beschränkt sich die Kommunikation allerdings nicht mehr nur auf diese zwei Bereiche, sondern dehnt sich auch auf die Prozesse als Ganzes aus. Dies geht einher mit der Entstehung von EAI-Systemen.

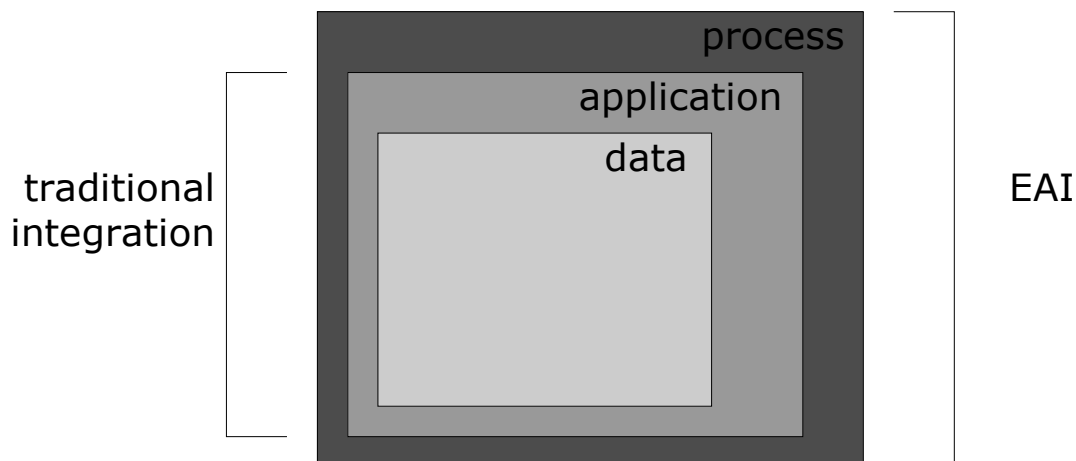


Abb. 1: Integrationsebenen³

1.2.1 Integration auf Datenebene⁴

Die Integration auf dieser Ebene beschreibt die Nutzung der Daten einer Anwendung durch eine andere, ohne dabei auf die Geschäftslogik der Fremdanwendung zuzugreifen. Dieser Zugriff kann auf zwei Arten stattfinden. Die erste Möglichkeit ist es, Anwendung A direkten Zugriff auf die Daten(bank) der Anwendung B zu gestatten. Dies ist vor allem dann angebracht, wenn Anwendung A auch schreibend auf die Daten zugreifen muss. Alternativ könne die Daten von Anwendung B auch in die Datenbank von Anwendung A repliziert werden. Das bietet sich aber nur an, wenn Anwendung A lediglich lesend auf die Daten zugreifen muss und wenn die Daten hauptsächlich statisch sind, da die Replikation sonst sehr häufig ausgeführt werden muss.

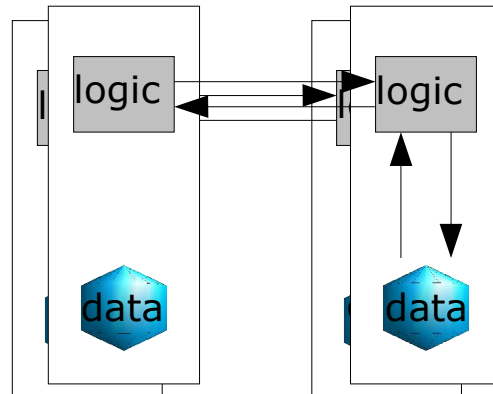
1.2.2 Integration auf Anwendungsebene⁵

Auf dieser Ebene ist die Anwendungslogik beider Anwendungen beteiligt. Anwendung A baut eine Punkt-Zu-Punkt-Verbindung zu Anwendung B auf (Abb. 2). Dabei wird sämtliche Logik, die Anwendung B beim Auslesen auf die Daten anwendet, ebenfalls genutzt. Die Daten sind also bereits vorvalidiert.

³ Erl (2004), S.289

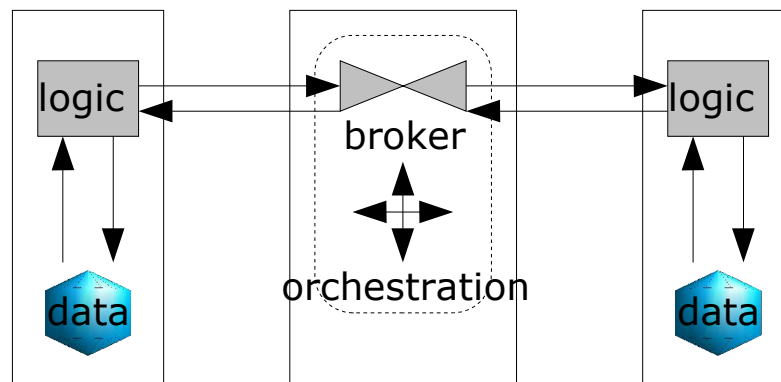
⁴ Vgl. Erl (2004), S.289

⁵ Vgl. Erl (2004), S. 290f.

Abb. 2: Punkt-zu-Punkt-Integration auf Anwendungsebene⁶

1.2.3 Integration auf Prozessebene⁷

Traditionell bedeutet Integration die Verbindung zweier Anwendungen auf Punkt-zu-Punkt-Ebene. Aktueller wird dies durch EAI und spezielle Middleware mit Hilfe eines sogenannten Messaging Frameworks abgebildet. Dadurch ergibt sich die Möglichkeit, zwei oder auch mehrere Prozesse miteinander zu verknüpfen. Hierbei entsteht ein komplett neuer Prozess. Dieser Prozess übernimmt die gesamte Kommunikation an sich und auch deren Steuerung (Abb. 3).

Abb. 3: EAI-Architektur⁸

Der Broker ist dabei für die reine Kommunikation, also die Übermittlung der Nachrichten, zuständig. Die Orchestration-Komponente managt die einzelnen Prozesse und sorgt für deren Ausführung an der richtigen Stelle.

1.2.4 Serviceorientierte Integration⁹

Web Services sind keine eigenständige Integrationslösung. Sie stellen lediglich neue Komponenten in einer EAI-Umgebung dar, die EAI oder Middleware wird zur Integration weiterhin

6 Angelehnt an Erl (2004), S. 290

7 Vgl. Erl (2004), S. 291

8 Angelehnt an Erl (2004), S. 291

9 Vgl. Erl (2004), S. 292

benötigt. Web Services bieten sich aber als Zugang zu fremden Daten und Geschäftslogik an (Abb. 4), da sie eine einheitliche Schnittstelle bieten und es der sie nutzenden Anwendung überlassen, die Daten in eine für sie weiterverarbeitbare Form zu bringen. Dies erspart Arbeit auf der Integrationsanwendungsseite.

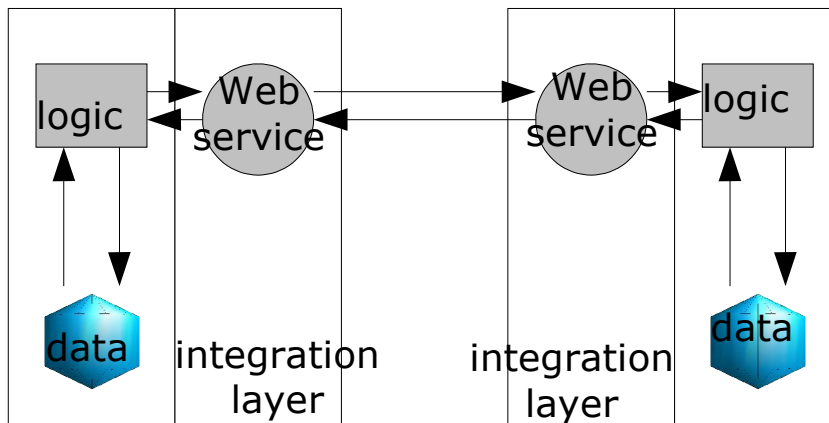


Abb. 4: Integrationsschichten für die serviceorientierte Architektur¹⁰

2. Probleme und Entscheidungen im Vorfeld

2.1 Middleware und EAI

Middleware entstand aus der Entwicklung von Mainframes hin zu verteilten Anwendungen. „Middleware stellt eine Ebene in einem komplexen Software-System dar, die als 'Dienstleister' anderen ansonsten entkoppelten Softwarekomponenten die Kommunikation untereinander ermöglicht. ... Middleware organisiert den Transport komplexer Daten (sog. Messaging),“ den Zugang zu Daten (über Protokolle, APIs), „vermittelt Funktionsaufrufe zwischen den Komponenten (sog. Remote Procedure Calls), stellt die Transaktionssicherheit über ansonsten unabhängige Teilsysteme her.“¹¹ Dabei bietet sie nach außen hin Schnittstellen an, die Anwendungen nutzen können. Intern regelt sie die Weiterverteilung der Aufgaben und Daten an andere angeschlossene Anwendungen.

EAI (Enterprise Application Integration) ist eine modernere Form der Middleware. Sie umfasst „die prozessorientierte Integration von Anwendungssystemen in heterogenen IT-Anwendungsarchitekturen“.¹² Das bedeutet, dass sie im Gegensatz zu herkömmlicher Middleware nicht nur als „Schnittstellensammlung“ fungiert, sondern auch komplexe Prozesslogik beherbergen kann. Dadurch ist ein komplettes Workflow-Management an einer Stelle realisierbar. Diese Funktionalität wird aber inzwischen auch von einigen herkömmlichen Middleware-Anbietern mit einer sogenannten Business Process Engine nachgerüstet, um konkurrenzfähig zu bleiben. EAI bietet also eine Art universelles Kommunikationsmedium, an das (fast) alle Anwendungen angeschlossen werden können, um die Kommunikation untereinander und das Arbeiten miteinander zu ermöglichen.

2.2 Hilfestellung auf dem Weg zum Service Oriented Enterprise

Zuerst sollte man sich darüber klar werden, ob EAI in diesem Umfang tatsächlich nötig ist. Möchte man nur Daten zwischen Anwendung A und B austauschen, ist ein solcher Komplettansatz häufig überdimensioniert. Stattdessen ist vielleicht tatsächlich nur diese eine Punkt-Zu-Punkt-Verbindung notwendig. Sollte aber die Entscheidung trotzdem auf einen ganzheitlichen Ansatz fallen, gibt es einiges zu beachten.

Zuerst gilt es, eine Bestandsaufnahme dessen vorzunehmen, was das Unternehmen schon unternommen hat. Welche Anstrengungen in dieser Richtung gibt es bereits, welche Anwendungen müssen verbunden werden, welche Verbindungen gibt es bereits, welche Einschränkungen müssen berücksichtigt werden... Danach ist zu klären, ob man auf dem aufbaut,

11 de.wikipedia.org/wiki/Middleware

12 de.wikipedia.org/wiki/Middleware

was vorhanden ist, z.B. in dem nur eine zusätzliche Web-Service-Integrationsschicht eingebunden wird, oder der direkte Sprung auf ein EAI-System unternommen wird. Dies kann auch in mehreren Phasen erfolgen, so dass eine bessere Planung und Steuerung gewährleistet ist.¹³

Für die Auswahl eines passenden Produktes gibt es verschiedene zu bedenken.¹⁴ Man sollte darauf achten, dass das gewählte System mehrere Modelle zur Integration zulässt und diese nicht proprietär diesem Produkt zugeordnet werden. Das führt häufig dazu, dass man an diese EAI-Anwendung gebunden ist und ein Umstieg auf eine neue erhebliche Probleme mit sich bringt.

Außerdem sollte man auf regelmäßige Updates achten. Nur ein Programm, das kontinuierlich weiterentwickelt wird und mit dem Stand der Technik Schritt hält, kann auch mit den wachsenden Anforderungen des Unternehmens mithalten und -wachsen.

Des Weiteren sollte es möglich sein Geschäftsprozesse auf möglichst einfache Art abzubilden und mit der Software nachzustellen. Es sollte Workflows, gezielte Nachrichtenweiterleitung und das Managen der Prozesse so einfach wie möglich für den Benutzer machen, ohne ihn in der Wahl einzuschränken.

Der Aufwand zur Integration der Anwendung selbst sollte möglichst gering sein. Dazu gehören die Installation und Konfiguration, das Anlegen neuer Kanäle und XML-Modelle der Nachrichten, aber auch die Sicherheit. Weiteren Aufwand stellen nötige Hilfe von außen und das Ausmaß an Codeanpassungen in den internen Anwendungen dar. Ein Plus ist die Einführung in Phasen.

Wartung und Administration des Systems sollte einfach sein (Usability). Dazu gehören Sicherheit, Versionierungsmöglichkeiten, Performanzüberwachung, Monitoring/Logging, Auffinden und Beseitigen von Fehlern, Neuanlegen von Geschäftsregeln im laufenden Betrieb usw.

Ein ebenfalls wichtiger Punkt ist die Skalierbarkeit des Systems. Es muss mit wechselnden Lasten klarkommen. Eventuell lässt es sich als verteiltes System realisieren. Einige Produkte haben auch interne Begrenzungen, z.B. die Anzahl der anpassbaren Adapter.

Integration ist also ein oftmals geld- und zeitintensives Unterfangen. Lizenzen, Hardware und Experten müssen herangezogen werden, Daten müssen migriert, Teile der eigenen Anwendungen neu entwickelt und alles zusammen getestet werden. Dies kann auch die gesamte Verfügbarkeit beeinträchtigen. Allerdings steht all diesem direkten Aufwand ein indirekter Wert gegenüber, der sich im Laufe der Zeit bemerkbar macht durch leichtere Wartbarkeit, weniger Wartezeit zwischen einzelnen Schritten innerhalb von Prozessen usw. In den wenigsten Fällen wird der direkte Aufwand nicht durch diesen indirekten Wert ausgeglichen.

13 Vgl. Erl (2004), S. 303

14 Vgl. Erl (2004), S. 296ff.

3. Einführung einer SOA

3.1 Service-Modelle für Enterprise-Integration-Architekturen

Da EAI prozessorientiert arbeitet, sind Modelle gefragt, die verschiedene Aspekte des Geschäftsgeschehens abbilden und koordinieren können. Dazu bieten sich hauptsächlich zwei Modelle an, Prozessservices und Koordinationsservices (für Geschäftszwecke).

3.1.1 Prozessservices¹⁵

Sie kapseln alle Teilservices, die für einen Geschäftsprozess nötig sind, in einem dedizierten Prozess. So ist die gesamte Logik mit allen Regeln, Ausnahmen und Bedingungen bezüglich des Workflows an einer Stelle verfügbar. Die Kapselung ist dabei über mehrere Ebenen möglich - jeder der Partnerservices kann selbst wieder ein Prozessservice sein.

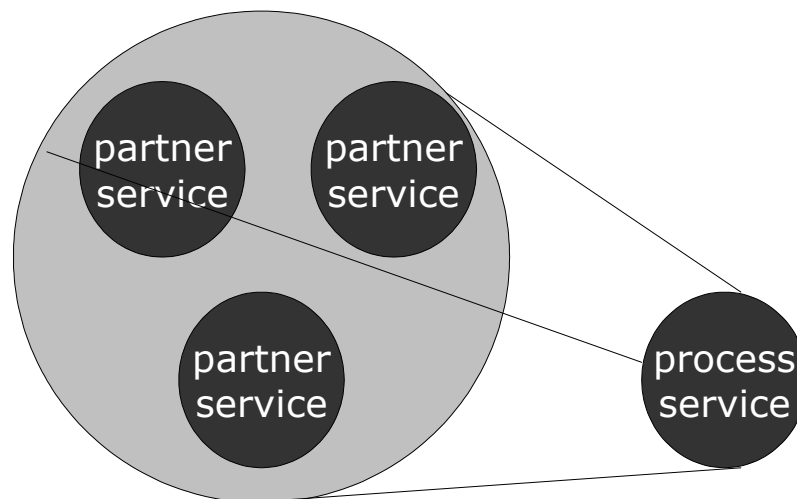


Abb. 5: Prozessservice¹⁶

Ein solcher Service wird viel genutzt, da er an jedem einzelnen Schritt des Prozesses beteiligt ist. Für das Schnittstellendesign hat sich BPEL4WS durchgesetzt. Es wird implementationsunabhängig durch das Weglassen der Binding-Information im dazugehörigen WSDL-Dokument. Diese Services werden oft im Zusammenhang mit einer Orchestration Engine genutzt, die die Prozessservices steuert.

3.1.2 Koordinationsservices (für Geschäftszwecke)¹⁷

Diese Prozesse stehen in engem Zusammenhang mit Transaktionen (WS-Coordination + WS-Transaction bei Web Services der zweiten Generation). Dabei werden zwei Arten von Transaktionen unterschieden: Atomare Transaktionen nach dem ACID-Prinzip und Langzeittransaktionen, um die es im Folgenden hauptsächlich gehen wird. Langzeittransaktionen

¹⁵ Vgl. Erl (2004), S. 356ff.

¹⁶ Angelehnt an Erl (2004), S. 357

¹⁷ Vgl. Erl (2004), S. 358ff.

sind immer dann nötig, wenn Erfolgs- und Fehlerbedingungen behandelt werden müssen, die den Ablauf des gesamten Prozesses beeinflussen.

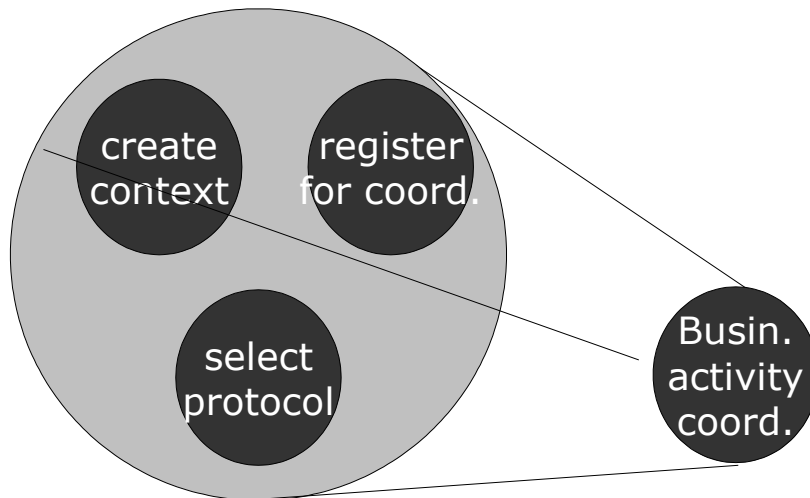


Abb. 6: Koordinationsservice¹⁸

Das Nutzungsaufkommen lässt sich nur schwer einschätzen, da die Geschäftsaktivität von wenigen Minuten bis zu mehreren Tagen dauern kann und die tatsächliche Bearbeitung meist sporadisch stattfindet. Die Schnittstelle nach außen und der Aufbau sind vordefiniert durch WS-Coordination, können aber erweitert werden. Diese Services werden häufig im Rahmen von EAI-Servern genutzt.

3.2 Grundlegende Architekturkomponenten der Enterprise Integration

Die Kommunikation zwischen den Anwendungen und die Automation werden durch zwei Kernkomponenten gesteuert (Abb. 7).

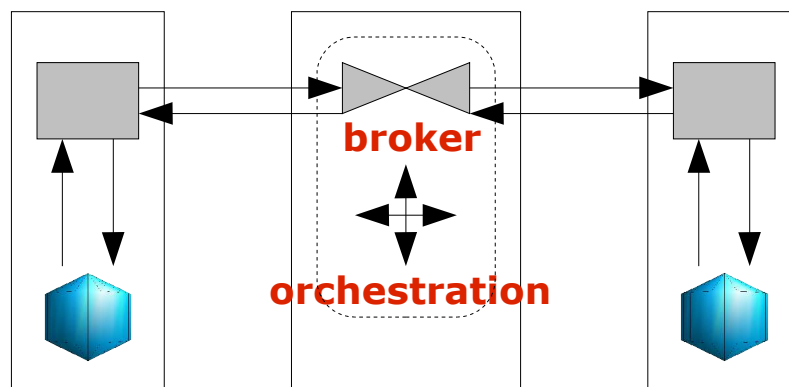


Abb. 7: EAI-Architektur

3.2.1 Broker¹⁹

Der Broker übernimmt vor allem Laufzeitfunktionen wie das Zusammenführen von Dokumenten aus verschiedenen Quellen, Datentransformationen usw. Außerdem sorgt er dafür, dass die Daten

¹⁸ Angelehnt an Erl (2004), S. 361

¹⁹ Vgl. Erl (2004), S. 362ff

immer in genau dem Format vorliegen, in dem sie erwartet werden. Das lässt sich anhand eines Beispiels für den Datenaustausch in 5 Schritten nachvollziehen.

Ausgangspunkt ist der Aufbau aus Abb. 8.

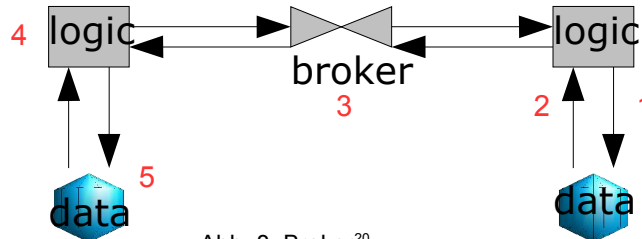


Abb. 8: Broker²⁰

Zuerst müssen die gewünschten Daten von Anwendung B aus der Datenbank ermittelt werden. Als nächstes werden die Daten durch die Anwendungslogik der Anwendung B mit Hilfe eines Schemas validiert. Im dritten Schritt ändert der Broker das Datenformat. Nun kann Anwendung A die Zieldaten mit dem Zielschema validieren. Als letztes werden die Daten in die Zieldatenbank eingefügt.

3.2.2 Orchestration²¹

Das Orchestration-Tool dient der Kapselung und Ausführung der Geschäftslogik. Durch die Integration anderer Anwendungen wird es erst möglich Zusatzdaten zu ermitteln. Der Broker wird dabei zur tatsächlichen Manipulation der Daten genutzt. Diese Komponente entscheidet z.B. auch die Zurückweisung im Fehlerfall. Erst durch die Orchestration sind Nachrichten mit zusätzlichen Routing- und Transaktionsinformationen sinnvoll, da diese hier verarbeitet werden. Außerdem sorgt dieser Teil für den Nachrichtenstatus, abhängig davon, ob die Bearbeitung abgeschlossen ist oder nicht.

Das Beispiel für den Datenaustausch lässt sich auch auf die Orchestration übertragen (Abb. 9).

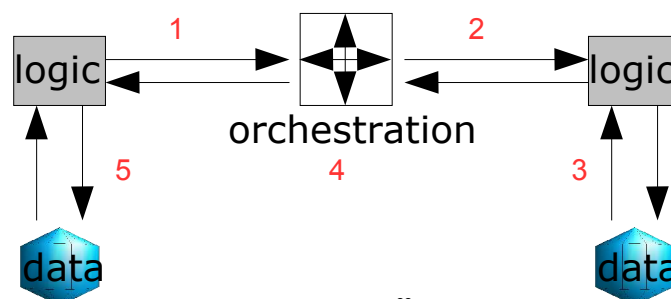


Abb. 9: Orchestration²²

Im ersten Schritt erfragt Anwendung A Daten. Dabei muss A von Bs Existenz nichts wissen. Einstiegspunkt ist der Prozess oder Workflow. Die Orchestration findet nun heraus, an wen diese Anfrage weitergeleitet werden muss und erfragt dort die Daten. Die Antwort geht zurück an die

²⁰ Angelehnt an Erl (2004), S. 363

²¹ Vgl. Erl (2004), S. 365ff.

²² Angelehnt an Erl (2004), S. 367

Orchestration. Diese bearbeitet nun die Daten entsprechend der Logik des Geschäftsprozesses (das kann auch bereits mit der Anfrage in Schritt 1 passieren). Dazu nutzt sie den Zugang zu anderen Quellen und die Möglichkeiten des Brokers, die Nachricht entsprechend zu manipulieren. Nachdem dieser Vorgang abgeschlossen ist, werden die Daten an die fragende Anwendung weitergeleitet.

3.3 Konkrete Architekturen zur Enterprise Integration

Im folgenden werden drei konkrete Architekturmodelle vorgestellt. Außerdem wirft Thomas Erl einen Blick auf die Kombination der jeweiligen Architektur mit Web Services, denn die hauptsächlich asynchron mit Nachrichten arbeitenden EAI-Systeme sind prädestiniert für die Nutzung von XML. Daher sind in viele serviceorientierte Integrationsprodukte Web Services schon integriert, zum Teil gibt es sogar rein darauf basierende Anwendungen. Web Services schlagen sozusagen die Brücke zwischen anbieterspezifischen EAI-Plattformen und plattformübergreifenden Wünschen und Bedürfnissen der Anwender.

3.3.1 Hub and Spoke²³

Bei dieser Architekturform fungiert eine proprietäre Middleware als Hub, an das alle Anwendungen über Adapter angeschlossen werden. Das heißt, es gibt einen zentralen Server, der die Integrationslogik hostet. Dort wird zentralisiert alles verarbeitet und der Server kontrolliert auch Orchestration und Broker. Durch die Adapter ist die Einbindung vieler unterschiedlicher Anwendungen möglich. Das Hub ist der Einstiegspunkt für alle Anwendungen, es ist kein Wissen um die physische Existenz des Gegenstücks notwendig. Die Anwendungen sind logisch autonom. Durch die Punkt-Zu-Punkt-Verbindung zwischen Hub und jeder einzelnen Anwendung sind beliebige und beliebig viele Verbindungen zwischen einzelnen Anwendungen realisierbar.

Durch die Zentralisierung wird die Wartung enorm vereinfacht. Der Administrator hat genau einen Anlaufpunkt bei Problemen. Außerdem kann so die Logik wiederverwendet werden und muss nicht für jeden Prozess neu eingebettet werden. Das erspart auch einen Teil der redundanten Mehrfachverarbeitung und -speicherung von Daten.

Allerdings entstehen durch die Zentralisation auch leichter Engpässe, abhängig von der Skalierbarkeit des Systems, da das Hub an allen Vorgängen beteiligt ist. Damit ist es auch ein sogenannter Single Point of Failure, wenn es ausfällt, kommt das gesamte System zum Erliegen.

All das und auch die oftmals teure Anschaffung und Wartung eines solch großen Systems ließe vermuten, dass es wenig genutzt wird. Stattdessen ist es eines der am häufigsten eingesetzten Modelle in der Realität.

Web Services spielen in diesem Modell lediglich die Rolle eines zusätzlichen, allgemeinen

²³ Vgl. Erl (2004), S. 371ff.

Adapters zum Hub, der nicht auf jede Anwendung spezialisiert werden muss, sondern von Applikationen mit Service-Möglichkeit als einfache Anbindungsmöglichkeit genutzt werden kann. Allerdings macht das die Zugangskontrolle schwieriger, da ohne weitere Sicherheitsbeschränkungen im Prinzip jeder Service mit Wissen um den Zugang diesen frei nutzen kann.

3.3.2 Messaging Bus²⁴

Der Messaging Bus oder auch Informationsbus oder publish/subscribe-Modell arbeitet vom Konzept her ähnlich wie das Hub and Spoke. Es gibt eine zentral gelegene Umgebung und individuelle Verbindungen zu den einzelnen Anwendungen. Die interne Struktur weicht jedoch ab. Es gibt eine sogenannte Information Pipeline für ein- und ausgehende Nachrichten, an die per Adapter wieder jede Anwendung angeschlossen ist. Dabei nehmen die Anwendungen die Rolle des Veröfentlichters oder des Abonnenten ein. Jede Integrationsquelle ist dabei Veröfentlichter, andere Anwendungen können diese Quelle abonnieren. Die Nachricht wird über die entsprechende Pipeline als Broadcast an alle dortigen Abonnenten gesendet. Für Transformation, Routing und Prozesslogik ist weiterhin die Orchestration zuständig.

Hierbei können intelligente Adapter einen Teil der Aufgaben übernehmen, z.B. das Umwandeln der Formate. Durch die stärkere Verteilung der einzelnen Prozesse und die damit einhergehende geringere Zentralisation kommt es auf der einen Seite zu weniger Engpässen, auf der anderen Seite verschlechtert sich aber auch die Wartbarkeit.

Durch die Einbindung von Web Services in Form einer Integrationsschicht für standardisierte Kommunikation werden die Adapter abstrahiert und die Verarbeitungslast der Messaging Bus Server sinkt weiter.

3.3.3 Enterprise Service Bus²⁵

Diese Architekturmodell ist von Grund auf serviceorientiert designt und damit die „Reinform“ der Service Oriented Architecture. Alle EAI-Komponenten sind einzelne Services oder in sich abgeschlossene Anwendungen, die durch Services steuerbar werden. Die Grundlage ist eine Integrationsplattform als Hosting-Umgebung der einzelnen Komponenten. Diese Plattform kann durchaus proprietär sein, hat aber standardisierte Schnittstellen, die sie universell einsetzbar machen. Auf dieser Grundlage setzen die (freien) Services für die einzelnen Teile einer EAI-Umgebung auf. Durch die Standardschnittstellen ist es möglich, diese Services jederzeit durch andere, passendere auszutauschen oder sogar durch eigene zu ersetzen. Auch die Plattform als solche ist austauschbar, ohne dass Logik verloren geht, denn die ist in den einzelnen Services gekapselt.

24 Vgl. Erl (2004), S. 374ff.

25 Vgl. Erl (2004), S. 377ff.

Dadurch wird der Benutzer freier in der Wahl dessen, was er braucht und was er nutzen möchte. Die Bindung an einen festen Anbieter wird gelockert. Dies ist für den Moment sicherlich das erstrebenswerteste Modell, um ein Unternehmen für die Zukunft abzusichern.

Literaturverzeichnis

Literaturverzeichnis

Erl, Thomas (2004) „Service-Oriented Architecture – A Field Guide to Integrating XML and Web Services“ Upper Saddle River (2004): Pearson Education, Inc. (New Jersey, USA), Seite 284 – 303, 356 – 379

Quellen im Internet

o.V. (2005) Wikipedia, freie Enzyklopädie: “Middleware”; Internet:
<http://de.wikipedia.org/wiki/Middleware>, Stand: 24.11.2005, Abruf: 01.12.2005