J2EE Connector Architecture

Erik Graf
Hochschule der Medien
Studiengang Medieninformatik

Ausarbeitung zur Vorlesung Design Patterns
· SS 2004 ·

Zusammenfassung

Die Absicht dieser Ausarbeitung ist es einen grundlegenden Einblick in die hinter der J2EE Connector Architecture stehenden Konzepte zu geben. Das Verständniss der Architektur steht dabei im Vordergrund. Es wurde versucht diese Konzepte in möglichst neutraler Weise darzustellen und die Erklärungen so zu gestalten, das nur wenig Vorkenntnisse aus dem J2EE Bereich von Nöten sind. Ausgehend von Erläuterungen zum Einsatzbereich der J2EE/CA wird schrittweise die grobe Architektur derselben erläutert. Wer den unwiderstehlichen Drang verspürt sich tiefer mit der Materie zu befassen, dem empfehle ich die unter Literatur aufgelisteten Quellen.

Science is to see what everyone else has seen but think what no one else has thought . . .

Albert Szent-Gyorgyi

Inhaltsverzeichnis

1 Enterprise Application Integration (EAI) 4

2 Applikationsübergreifende Kommunikation 7

3 Die Architektur der J2EE Connector Architecture 8

3.1 System Contracts 10

3.1.1 Connection Managment 11

3.1.2 Security management 11

3.1.3 Transaktions Management 11

4 Fazit 12

1 Enterprise Application Integration (EAI)

Man sagt, das die Notwendigkeit die Mutter der Erfindung ist. Bevor wir uns also näher mit der J2EE Connector Architecture auseinandersetzen ist es hilfreich sich zunächst einmal mit dem von ihr zu lösenden Problem ein wenig näher auseinanderzusetzen. Die J2EE/CA ist Bestandteil der Bemühungen unternehmensinterne Applikationen miteinander zu verknüpfen. Software Umgebungen heutiger Unternehmen sind zumeist heterogene und aus vielen Einzelelementen bestehende Systeme. Die einzelnen Bausteine dieser Umgebung werden oft als „Enterprise Informations System“ bezeichnet. Sharma, Stearns and Ng [SN01] definieren ein EIS als:

> We define an enterprise information system as an application or enterprise system that provides the information infrastructure for an enterprise. An EIS consists of one or more applications deployed on an enterprise system. An EIS provides a set of services to its users. Services exposed to clients may be at different levels of abstraction-including the system level, data level, function level, and business object or process level.

Beispiele für Enterprise Informations Systeme sind:

- Enterprise Resource Planning (ERP) Systeme : SAP, Peoplesoft
- Customer Relationship Management (CRM) Systeme : Siebel
- Mainframe Transaction Systeme: CICS
- Unternehmensspezifische Programme; Programme die für ein spezifisches Unternehmen geschrieben wurden. Solche Applikationen werden oftmals als Legacy [engl. : Altlast, Hinterlassenschaft] Applikationen bezeichnet. Ein Beispiel für Legacy Applikationen wären die im Bankenbereich noch weit verbreiteteten COBOL basierten Systeme.
- Nicht relationale Datenbanksysteme, z.B. IMS.

Das Aufgabengebiet des sogenannten Enterprise Application Integration (EAI) ist es die Kommunikation zwischen den einzelnen EIS eines Unternehmens zu

ermöglichen. Im Zuge der stetigen Ausbreitung des Internets tritt zudem die Web-Anbindung solcher EIS Systeme immer mehr in den Vordergrund. Die folgende Graphik skizziert den beispielhaften Einsatz einzelner EIS innerhalb verschiedener Geschäftsbereiche eines Unternehmens.

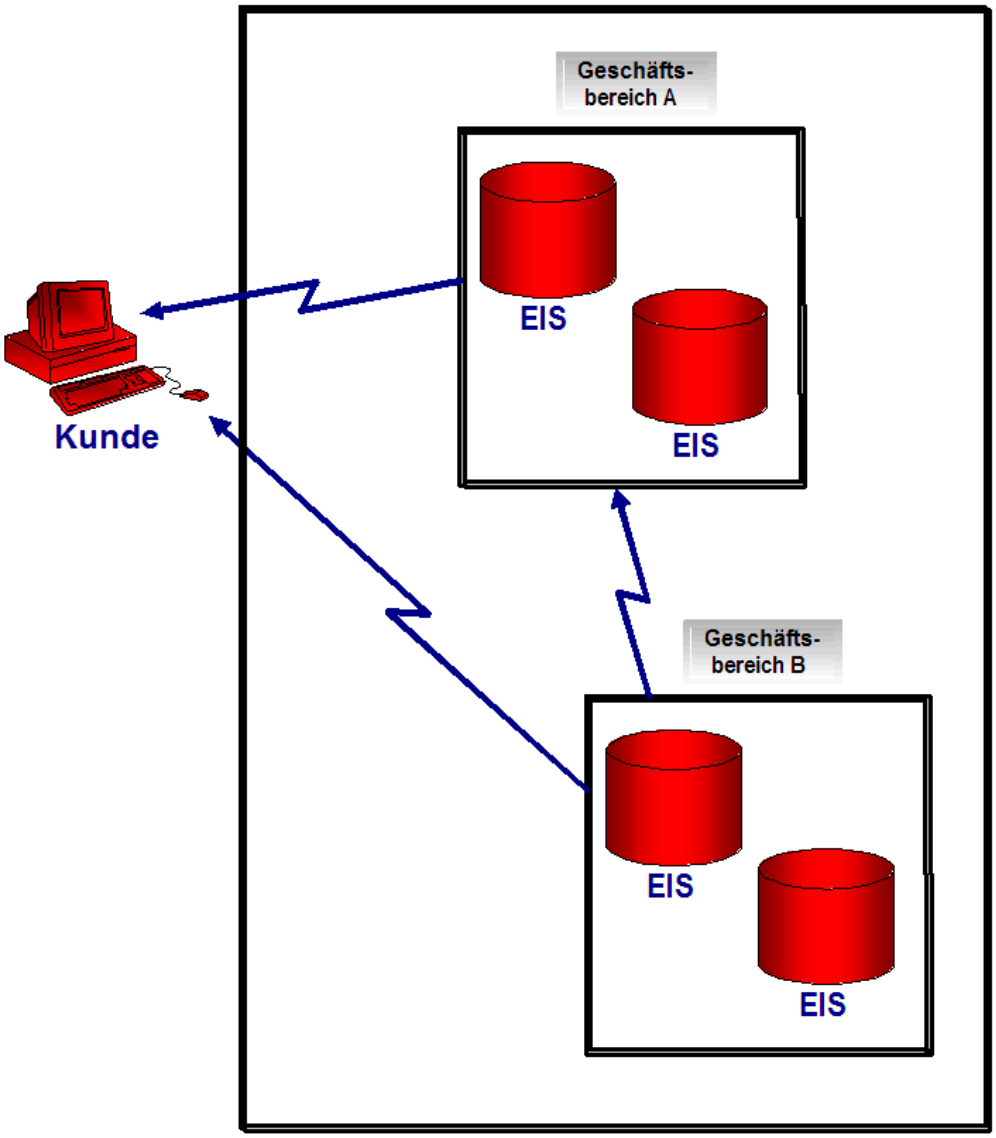

Abbildung 1: Beispielhafte Kommunikation verschiedener EIS

Die grundsätzliche Aufgabe von Enterprise Application Integration ist es also sowohl interne als auch externe Kommunikation zwischen EIS Applikationen zu erlauben die auf verschiedenen Platformen laufen und verschiedene Protokolle zur Kommunikation verwenden. Die hohe Heterogenität der Enterprise Informations

Systeme führte dazu, dass Enterprise Application Integration in der Vergangenheit von allen Beteiligten oft als ein schmerzvoller und zeitaufwändiger Prozess empfunden wurde. Das nächste Kapitel wird sich näher mit den Grundsätzlichkeiten von appliaktionsübergreifender Kommunikation befassen.

2 Applikationsübergreifende Kommunikation

Kapitel 1 gab uns einen Einblick in den Einsatzbereich der J2EE Connector Architecture. Im zweiten Kapital möchten wir nun näher auf die grundsätzlichen Merkmale applikationsübergreifender Kommunikation eingehen um uns ein Bild über die Anforderungen an eine Kommunikationsarchitektur machen zu können.

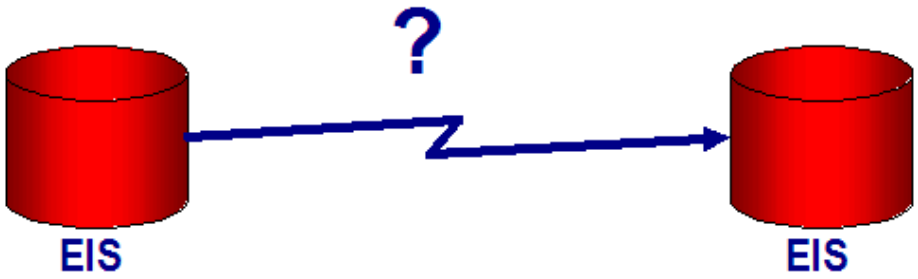

Abbildung 2: Kommunikation zweier EIS

Die grundsätzlichsten Anforderungen an Kommunikation sind sicherlich das Vorhandensein eines gemeinsamen Kanals und einer gemeinsamen Sprache (Protokoll). Daneben sind jedoch auch höherwertige Anforderungen in Betracht zu ziehen. Solche höherwertigen Anforderungen sind beispielsweise:

- Security: Die Authentizität und Authorisierung während der Kommunikation
- Load Balancing und Failover: Skalierbarkeit und Fehlerverhalten während der Kommunikation
- Transaction Managment: Unterstützung von Transaktionsverwaltung; z.B. Rollback Mechanismen
- Synchrone und Asynchrone Kommunikationsuntersützung.

Eine ausgereifte Kommunikationsarchitektur muss daher all diese Punkte berücksichtigen. Wie wir später sehen werden ist dies bei der J2EE/CA der Fall.

3 Die Architektur der J2EE Connector Architecture

Die CA stellt einen der Grundbausteine der J2EE Architektur dar. Die grundsätzliche Philosophie derselben ist es einen Applikationsserver als Integrationspunkt für verschiedene EI Systeme zu etablieren. Der Applikationsserver stellt dabei sowohl den zentralen Punkt für die Kommunikation interner Enterprise Informations System als auch eine externe Schnittstelle zu externen Applikationen dar. Die Kommunikation mit den einzelnen EI Systemen basiert dabei auf sogenannten Resource Adaptern.

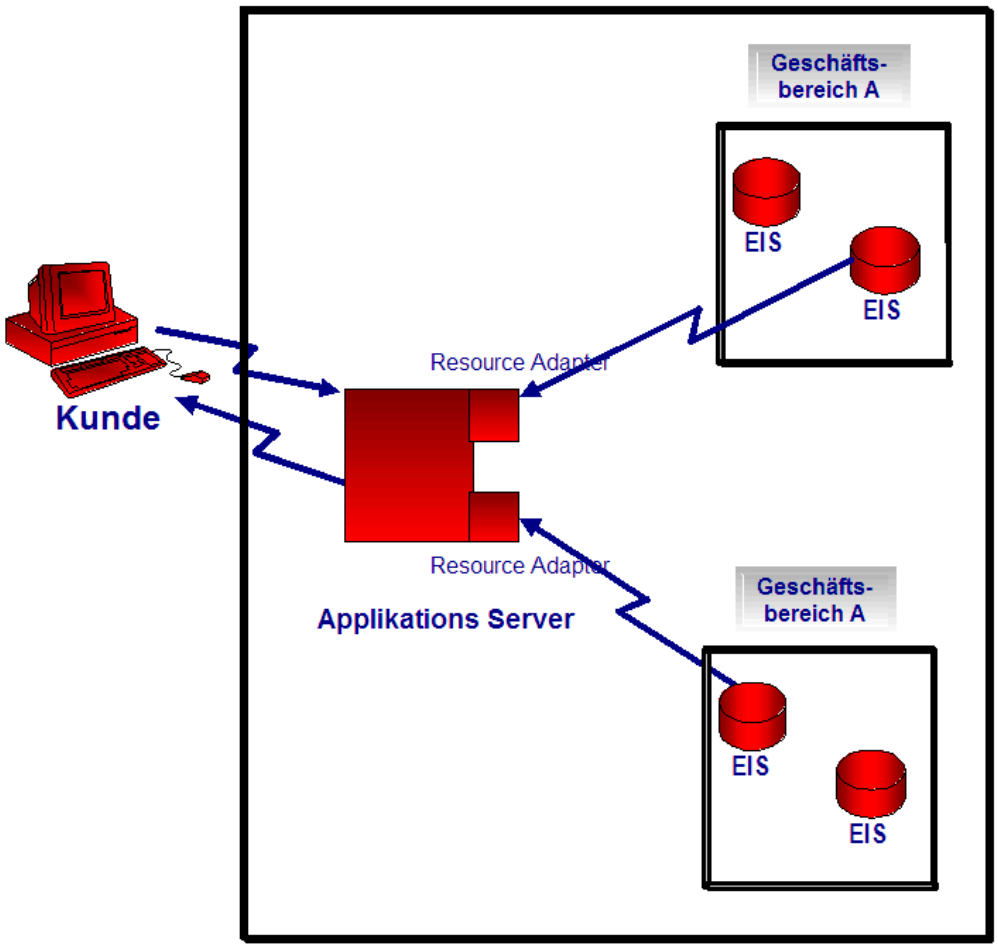

Abbildung 3: Applikationsserver als Integrationspunkt

Ein Resource Adapter ist eine vom EIS Hersteller zur Verfügung gestellte

Library die per Definition in jeden J2EE kompatiblen Applikationsserver eingebunden werden kann. Eine auf dem Applikationsserver laufende Applikation kann über diesen Resource Adapter auf ein Enterprise Information System zugreifen. Die Implementierung dieser Resource Adapter beruht dabei auf sogenannten „Contracts“. Ein Contract stellt dabei mehr als nur eine Interface Definition dar, indem er auch Anforderungen an Skalierbarkeit, Sicherheit und Transaktionsverhalten stellt.

Dieses Grundprinzip ermöglicht es somit einem EIS Hersteller einen für alle Applikationsserver verwendbaren Adapter zu erstellen. Ein Applikationsserver hingegen wird in die Lage versetzt mittels ihrer spezifischen Adapter eine Vielzahl von Enterprise Informationssystemen einzubinden.

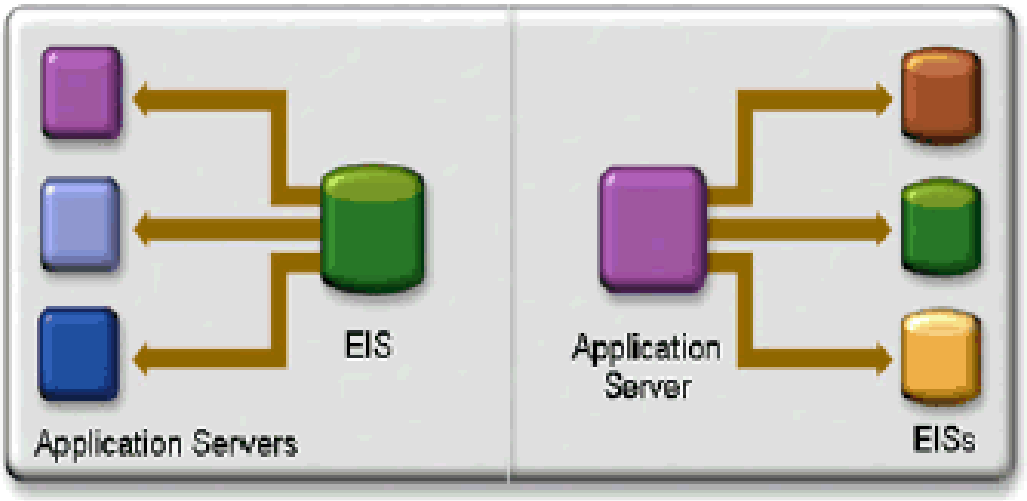

Abbildung 4: Vorteile der CA [eA02]

3.1 System Contracts

Um die Konnektivität zwischen dem Applikationsserver und dem EIS sicher zu stellen definiert die J2EE/CA sogenannte System Contracts.

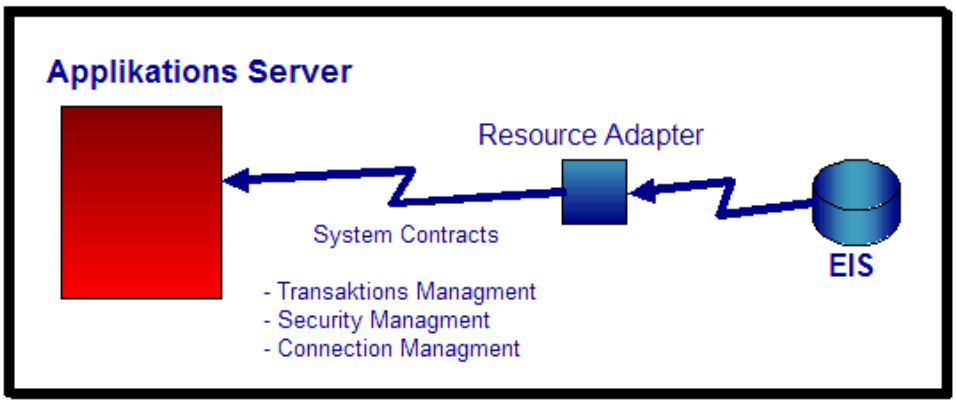

Abbildung 5: System Contracts

Diese Contracts definieren Anforderungen bezüglich der Kommunikation zwischen dem EIS und dem J2EE Server in den folgenden Bereichen:

- Connection Managment
- Transaktions Managment
- Security Managment Mechanismen

Die Einhaltung dieser Anforderungen auf Seiten der Hersteller der Enterprise Information Systeme und der J2EE Applikationsserver ermöglicht es EI Systemen und Appserver unterschiedlicher Hersteller miteinander zu kommunizieren.

3.1.1 Connection Managment

In den Bereich des Connection Managments fallen beispielsweise der Auf- und Abbau von Verbindungen sowie das sogenannte Connection Pooling. Aus Sicht des Resource Adapters bedeutet dies, das bezüglich Skalierbarkeit und Geschwindigkeit der Verbindungen bestimmte Anforderungen eingehalten werden müssen. Um zum Beispiel Connection Pooling zu ermöglichen ist es notwendig das die Implementierungen beider Seiten das Cachen und das Verwalten von gecachten Verbindungen unterstützen.

3.1.2 Security management

Beim Security Management stehen folgende Gesichtspunkte im Vordergrund:

- Authentizierung : Die Verifizierung von Benutzern (Systemen)
- Authorisierung : Die Kontrolle von bestimmten Rechten
- Kommunikationssicherheit: zum Beispiel Point-to-Point Sicherheit mittels SSL

3.1.3 Transaktions Management

Eine weitere sehr wichtige in den System Contracts festgelegte Anforderung ist das Transaktionsmanagment. Transaktionsmanagment ist notwendig um die Integrität von Daten zu wahren. Ein beliebtes Beispiel für den Einsatz von Transaktionsmanagment kommt aus dem Bankenbereich. Man stelle sich vor, das ein System Geld von Konto A nach Konto B transferieren möchte. Zu diesem Zweck „hebt“ das System einen bestimmten Geldbetrag von Konta A ab. Bevor das Geld jedoch auf Konto B transferiert werden kann, fällt der Strom aus. Stunden später wird die Maschine wieder hochgefahren. Die Kohle von Konto A wurde abgehoben. Selbige landete jedoch nicht auf Konto B sondern im digitalen Nirvana. Transaktionsmanagment setzt dem die Einführung von Transaktionsklammern entgegen. Eine Transaktionsklammer bündelt verschiedene Arbeitsschritte zusammen. Veränderungen an wichtigen Daten werden dabei nur vorgenommen wenn alle Schritte erfolgreich durchlaufen werden konnten. Sollte beispielsweise nach dem sechsten von sieben Schritten ein Fehler aufgetreten sein, so wird die gesamte Transaktion rück-abgewickelt (Rollback).

4 Fazit

Die J2EE Connector Architecture stellt für Enterprise Informations Systeme das dar, was JDBC für relationale Datenbanken darstellt. Ein Rückblick an die Zeit vor JDBC und die damaligen Schwierigkeiten beim Einbinden von Datenbanken lässt erahnen welche Bedeutung die J2EE/CA bei der Integration von EI Systemen zukommt.Genauso wie heutzutage wohl die wenigsten auf den Einsatz von JDBC verzichten, wird in Zukunft wohl niemand im Bereich der Enterprise Application Integration um den Einsatz der Java Connector Architecture herumkommen.

Abbildungsverzeichnis

1 Beispielhafte Kommunikation verschiedener EIS 5
2 Kommunikation zweier EIS 7
3 Applikationsserver als Integrationspunkt 8
4 Vorteile der CA [eA02] 9
5 System Contracts 10

Literatur

[eA02] Robert Youngs et Al. *Enterprise Integration with IBM Connectors and Adapters*. IBM, 2002.

[SN01] Stearns Sharma and Ng. *J2EE Connector Architecture and Enterprise Application Integration*. Addison Wesley, 2001.