Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Konzept und Umsetzung einer Service-orientierten Datenintegration für verteilte Datenbestände

Masterarbeit

im Studiengang Informatik

von

Chen Jin

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Dr. –Ing. Daniel Lübke
Betreuer: Dr. –Ing. Daniel Lübke
Betreuer: Dipl. Michael Koch

Hannover, 18. März 2009

Danksagung

Ich möchte mich an dieser Stelle bei allen herzlich bedanken, die mich in den letzten sechs Monaten unterstützt haben.

Besonderer Dank geht an Herrn Prof. Dr. Schneider, der mir die Gelegenheit für diese Masterarbeit gab. Desweiteren danke ich Herrn Dr. Daniel Lübke und Herrn Michael Koch herzlich für ihre kompetente Betreuung, die konstruktiven Diskussionen, aus denen ich viele hilfreiche Hinweise für meine Arbeit erhalten habe und für die Zeit und das Engagement, die sie für diese Arbeit aufbringen konnten.

Meinen Eltern möchte ich dafür danken, dass sie mich stets unterstützten. Besonders möchte ich mich bei Herrn Christian Lemke, Herrn Martin Richtarsky und Herrn Sebastian Wittler bedanken, die mir vor allem in der letzten Phase der Arbeit immer wieder Anregungen gaben und mir bei der Korrektur hinsichtlich der Grammatik und der Rechtschreibung halfen.

Hannover, den 18.03.2009

Erklärung

Hiermit versichere ich, Chen Jin, die vorliegende Masterarbeit selbständig, ohne fremde Hilfe und nur unter Verwendung der von mir aufgeführten Quellen und Hilfsmitteln angefertigt zu haben.

Hannover, den 18.03.2009 _____

Unterschrift: Chen Jin

Zusammenfassung

Informationen sind die wichtigsten und wertvollsten Güter für ein Unternehmen. Aufgrund zahlreicher unterschiedlicher Datenbanksysteme mit Daten aus Geschäftsprozessen ist es für Unternehmen immer noch problematisch, die geeigneten Informationen zum richtigen Zeitpunkt an der richtigen Stelle bereitzustellen. Datenintegration bietet eine Möglichkeit, alle in einem Geschäftsprozess benötigten Daten einheitlich an einer zentralen Stelle zusammenzuführen.

Um die Datenintegration zu ermöglichen, stehen vielfältige Integrationstechnologien in der Praxis zur Verfügung. Die service-orientierte Architektur und deren wichtigstes Element - die Services - sind neue Technologien, die immer mehr an Bedeutung gewonnen haben.

Diese Masterarbeit zeigt zuerst, wie Datenintegration die betrieblichen Geschäftsprozesse unterstützen kann. Daraus werden verschiedene Anforderungen an ein Konzept zum einheitlichen Datenzugriff abgeleitet. Darauf aufbauend wird ein eigenes Konzept zur service-orientierten Datenintegration entwickelt. Dessen Vor- und Nachteile werden analysiert und die Umsetzbarkeit in der Praxis mit einer Fallstudie überprüft.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Motivation

Informationen sind für jedes Unternehmen die wertvollsten Güter und spielen eine sehr bedeutende Rolle. In den vergangenen Jahren haben viele Unternehmen eigene Informationssysteme aufgebaut, deren Daten aus sehr unterschiedlichen Quellen stammen und häufig sehr unterschiedliche Strukturen haben. Die Unternehmen selber sind heutzutage geprägt durch „Mergers and Acquisition"-Aktivitäten, vielfach nach Ländergesellschaften und Standorten ausgerichtet. An den Standorten werden normalerweise eigene Anwendungssysteme genutzt und entsprechende Arbeitsverfahren und Prozesse haben sich etabliert.

Für die Informationsnutzer eines Unternehmens ist es aber oft der Fall, dass sie nicht nur lokale Informationen benötigen, sondern auch die von anderen Anwendungen oder Standorten. Es ist daher sinnvoll, die Daten von verschiedenen Systemen unter einem gemeinsamen Dach zu vereinen. Bei einer solchen Datenintegration ist es in vielen Fällen aber auch heute noch so, dass die Benutzer gezwungen sind, manuell die Daten von einem System in ein anderes zu übertragen. Sie müssen, um an die gewünschten Informationen zu kommen, zwischen verschiedenen Applikationen hin und her wechseln. Neben dieser umständlichen Arbeitsweise entstehen durch dieses Vorgehen auch häufig Fehler, beispielsweise wenn beim manuellen Übertragen unbeabsichtigt die Kommastelle eines Wertes verschoben wird. Um die möglichen Fehlerquellen bei der Datenverarbeitung zu minimieren und die Arbeit der Informationsnutzer zu vereinfachen, ist es dann notwendig, automatisiert Kommunikationskanäle zwischen den Systemen zu erstellen und die Daten der verteilten Systeme ohne Benutzerinteraktion zusammenzuführen. Technologien zur Datenintegration haben deswegen immer mehr an Bedeutung gewonnen.

Unter dem Begriff *Datenintegration* versteht man eine logische Integration von Informationen aus mehreren heterogenen Datenquellen. Die Herkunft der Informationen ist für die Benutzer transparent, d.h. die Benutzer brauchen nicht genau zu wissen, woher die Informationen kommen und wohin sie abgelegt werden. Das einzige für den Benutzer Interessante ist die Bereitstellung der korrekten und vollständigen Daten, damit die Benutzer diese in ihren

Werkzeugen verwenden können. Zu diesem Zweck wurden die zurzeit auf dem Markt verfügbaren Ansätze betrachtet. Meistens wird hier aber nur der lesende Zugriff auf die Daten berücksichtigt. In manchen Anwendungen müssen die Daten aber zusätzlich auch noch bearbeitet werden können, d.h. eine rein lesende Datenintegration ist nicht ausreichend.

Ein anderer wichtiger Punkt sind die Technologien, die zur Realisierung der Datenintegration in verteilten Systemen benötigt werden. Unter den zahlreichen Kommunikations-infrastrukturen hat die *service-orientierte Architektur* zuletzt immer mehr an Bedeutung gewonnen. Diese neue Technologie stellt neue Möglichkeiten für die Realisierung einer Integrationslösung zur Verfügung, indem sie Prinzipien definiert, wie Anwendungen über heterogene Programmiersprachen und Betriebssysteme hinweg miteinander interagieren können. Unter Verwendung der service-orientierten Architektur soll es erreicht werden, dass das verwendete Integrationsprotokoll für den Anwendungsentwickler transparent ist. Es werden also keine Kenntnisse über spezielle Remote Procedure Call-, Messaging- oder andere Kommunikationsprotokolle benötigt. Der Entwickler kann sich dann allein auf die Abbildung der zu implementierenden Geschäftsfunktionalität fokussieren und nicht auf die verschiedenen Client-Programmiermodelle mit denen man die einzelnen Funktionalitäten aufrufen kann.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung eines Konzeptes zur service-orientierten Datenintegration. Im Rahmen dieser Arbeit werden zuerst verschiedene Ansätze zur Datenintegration diskutiert. Danach wird eine Architektur definiert, die es ermöglicht, verteilte Datenbestände zu vereinen und für Mandanten lesbar und schreibbar zur Verfügung zu stellen. Bei der Bewertung des entwickelten Konzeptes werden die Vor- und Nachteile aufgelistet und kurz begründet. Daraus lässt sich dann auch ableiten, in welchen Einsatzszenarios dieses Konzept eingesetzt werden sollten. Desweiteren wird in der praktischen Komponente der Arbeit aufgezeigt, wie eine Anwendung dieser neuen Konzepte für die Datenintegrationsunterstützung in der Realität aussehen kann. Das Konzept wird anhand von dem logistischen Prozess des SAP ERP im Bereich Beschaffung auf seine Anwendbarkeit untersucht. Hierzu wird im Rahmen einer Fallstudie ein System spezifiziert, entworfen und implementiert. Das Ergebnis wird anschließend evaluiert.

1.3 Gliederung der Arbeit

Die vorliegende Masterarbeit untergliedert sich in folgende Kapitel:

Im **Kapitel 2** wird zuerst der Hintergrund des Problems anhand eines Unternehmens-szenarios erklärt. Darauf aufbauend werden dann die Datenintegration und die service-orientierte Architektur eingeführt. Die dafür benötigten Grundlagen werden ebenfalls in diesem Kapitel beschrieben.

Gegenstand von **Kapitel 3** sind die aktuellen auf dem Markt schon existierenden Ansätze zur Datenintegration. Sie werden kurz erklärt und ihre Vor- und Nachteile analysiert.

Um die Daten von verschiedenen Systemen in einem zentralen System integrieren zu können, wird eine passende Infrastruktur benötigt. **Kapitel 4** behandelt die auf Web Services basierende Datenintegration. Deren Vor- und Nachteile werden diskutiert.

Im **Kapitel 5** wird ein mögliches Konzept für eine service-basiete Datenintegration entwickelt und vorgestellt.

Um das im Kapitel 5 vorgestellte Konzept und deren Anwendbarkeit zu prüfen, wird es anhand einer Fallstudie im **Kapitel 6** untersucht.

Zum Ende hin wird im **Kapitel 7**, basierend auf den Ergebnissen der Untersuchung der Fallstudie, das entwickelte Konzept kritisch bewertet.

Der gesamte Inhalt wird im **Kapitel 8** noch einmal zusammengefasst. Das Ergebnis der vorliegenden Masterarbeit wird hier präsentiert und ein Ausblick auf mögliche Erweiterungen gegeben.

Kapitel 2

Umfeld und Grundlagen

Ein wichtiger Bestandteil der Standardsoftwarelösungen bei gleichzeitigem Weiterbetrieb ausgewählter vorhandener Anwendungen ist die Zusammenführung der Daten aus allen möglichen Quellen, um die gewünschten Informationen und Reports zeitnah generieren zu können. Datenintegration ist vor allem dort notwendig, wo mehrere gewachsene Systeme miteinander verbunden werden sollen, zum Beispiel bei der Zusammenführung von Firmen, Arbeitsabläufen und Anwendungen.

Die service-orientierte Architektur (SOA) ist ein Architekturkonzept, mit dem es auch möglich ist, eine Datenintegration zu realisieren. Es existieren mehrere Implementierungstechnologien für SOA. Die verbreiteteste und bekannteste Umsetzung von SOA verwendet Web Services.

Dieses Kapitel bietet einen Überblick über das Thema der Datenintegration von verteilten Datenbeständen und wird die für das Verständnis einer service-orientierten Architektur wichtigen Begriffe erklären. Es wird zuerst anhand eines Unternehmensszenarios in die aktuellen Probleme der Informationsintegration und deren Anforderungen eingeführt. Darüber hinaus werden die grundlegenden Begriffe und Techniken, die zur service-basierten Datenintegration dienen, vorgestellt. In den folgenden Kapiteln werden beide Themengebiete dann miteinander verbunden, um ein darauf basierend Konzept zu entwickeln.

2.1 Umfeld

In diesen Abschnitt wird anhand eines virtuellen Unternehmensszenarios Wissen über Datenintegration vermittelt und sich damit dem Thema angenähert.

2.1.1 Unternehmensszenario

Von einem großen Unternehmen DIwS AG[1] werden neben dem Hauptsitz in Stuttgart noch mehrere rechtlich selbständige Standorte, die jeweils in Mannheim, Düsseldorf, Kassel, Berlin und Hannover liegen, unterhalten.

[1] Data Integration with Web Services AG

Die Unternehmensbereiche Beschaffung / Produktion, Logistik / Distribution und Marketing / Vertrieb sind aus historischen Gründen in jeder Stadt dezentral organisiert. Die Organisation der Informationsverarbeitung sowie die Unterstützung von Kernanwendungen, wie z.B. Finanzbuchhaltung, Produktionsplanung und –steuerung und Stammdatenverwaltung, sind ebenfalls dezentral. Außerdem sind die lokal installierten Anwendungssysteme auch aus historischen Gründen sehr unterschiedlich. Zum Zweck der effizienten betrieblichen Informationsverarbeitung wurde in mehreren Standorten hauptsächlich eine SAP-Lösung verwendet, z.B. SAP SRM oder SAP ERP. Eine alte Version von SAP ERP, das sogenannte R/3 System und ein anderes nicht-SAP System sind aber auch noch in Verwendung.

Diese organisatorische Dezentralisierung bringt eine immer schwerere gesamte Unternehmenssteuerung durch verteilte Prozesse mit sich. Die Prozesse konnten zwar in den einzelnen Gesellschaften optimiert, das gesamte Optimum für das Unternehmen damit aber nicht erreicht werden.

Unter Wettbewerbsdruck hat nun dieses Unternehmen die Entscheidung getroffen, einige dezentralisierte Prozesse, z.B. die Rechnungsprüfung, in eine zentrale Stelle zu verlegen, um somit eine zentrale Kontrolle über alle Rechnungen zu erhalten. Diese zentrale Stelle, wo sich solche Geschäftsprozesse befinden und deren Aufgaben bearbeitet werden, ist ein Shared Service Center [ESSC03]. Die untere Abbildung 1 stellt diesen Vorgang dar.

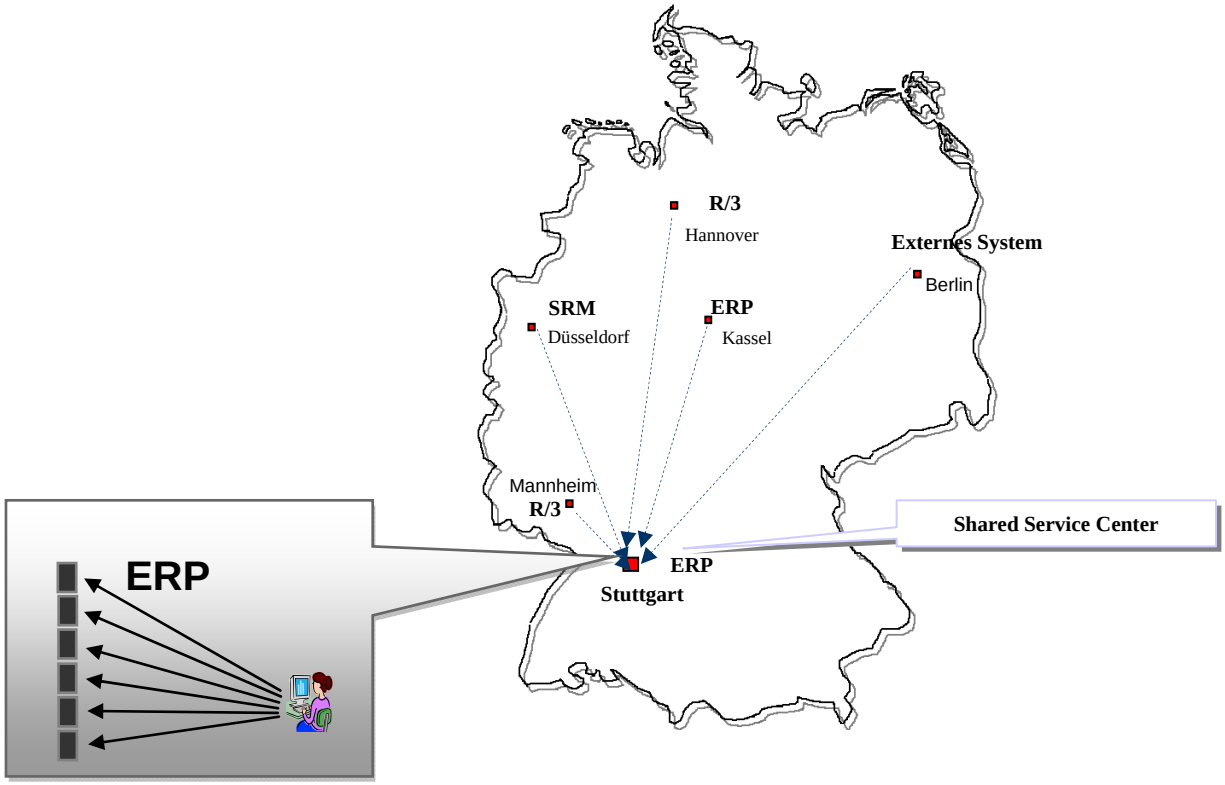

Abbildung 1: Zentralisierter Rechnungsprüfungsprozess im Unternehmensszenario

Im folgenden Abschnitt wird das Konzept des Shared Service Center kurz vorgestellt und in die dafür benötigte Enterprise Application Integration eingeführt.

2.1.2 Shared Service Center

Ein *Shared Service Center* (SSC) kann als eine Art des internen Outsourcings angesehen werden. Es ist eine zentrale Stelle oder Abteilung, die Shared Services anbietet. Der Begriff „Shared Service“ bezeichnet konsolidierte und zentralisierte Dienstleistungsprozesse in einem Unternehmen. Diese beziehen sich auf transaktionsbezogene Routineprozesse im Massengeschäft sowie spezialisierte Funktionen und Aktivitäten mit unternehmensweiten Informationsanforderungen. Mit der Einführung von Shared Services können administrative Prozesse verschiedener Standorte standardisiert und effizient an zentraler Stelle zusammengefasst werden. Allerdings dürfen dabei die Bedürfnisse und Kompetenzen der dezentralen Standorte nicht vernachlässigt werden [HSB+08].

Die Prozesse in einem Unternehmen, die als Shared Service angeboten werden können, haben folgende Merkmale:

- Hoher Grad an Standardisierung
- Hohe Anzahl von Wiederholungen desselben Prozesses
- Hoher Grad an Systemunterstützung, z.B. durch ERP-Systeme und dort insbesondere durch Workflows
- Nur wenige Ausnahmen

Wie im Beispielszenario schon beschrieben, wurden die Geschäftsprozesse, wie z.B. die Rechnungsprüfung, vorher in jeder Tochterfirma separat durchgeführt. Solche Prozesse, die gleiche Bearbeitungsvorgänge haben, benötigen an jedem Standort spezielle Mitarbeiter, um die fast gleiche Arbeit jeden Tag zu erledigen. Die so mehrfach durchgeführten Arbeiten und Systemwartungen bringen einen hohen Kostenaufwand mit sich. Außerdem fehlt dem Unternehmen eine globale Überwachung für die gesamten Rechnungen.

Diese Situation kann sich nach Einführung eines SSC ändern. Die Prozesse, die in allen Tochterfirmen ähnlich stattfinden, können jetzt ins SSC verschoben werden, was sowohl qualitative als auch quantitative Vorteile mit sich bringt.

Ein qualitativer Vorteil sind zum einen niedrigere Fehlerraten, die durch die Spezialisierung in einem Shared Service Center zu erwarten sind. Rechtliche und organisatorische Anforderungen führen ebenfalls immer häufiger dazu, dass Prozesse in Shared Service Center verlagert werden, um diese besser kontrollieren zu können. Außerdem können sich durch die Zusammenlegung gleichartiger Prozesse Skalierungseffekte ergeben und durch eine gleichzeitige Verlagerung des Standortes, ist eine Reduzierung von Miet-, Gebäude-, Telekommunikations-, Reise- und Nebenkosten möglich. Eine Kostensenkung ist auch bei

Löhnen und Gehältern möglich. Als weiterer Vorteil ist die Möglichkeit anzusehen, dass sich der einzelne Standort eines Unternehmens mehr auf seine Kernprozesse konzentrieren kann.

Um das Konzept des SSC umsetzen zu können, ist ein Verständnis des Themas der Enterprise Application Integration (EAI) vonnöten, die eine wichtige Grundlage für die Realisierung eines Shared Service Centers ist [ESSC03]. EAI ermöglicht es Unternehmen mit verschiedensten Plattformen, die entweder auf ERP oder kundenspezifischer Software basieren, eine einzige Plattform, wie z.B. SAP ERP Systeme, im Shared Service Center zu verwenden ohne die bestehenden Plattformen von mehreren Abteilungen und Standorten ändern zu müssen. EAI nutzt eine Middleware-Plattform, die die Verwaltung und Wartung von Schnittstellen zwischen der zentralen Plattform und den dezentralen Komponenten übernimmt.

Im Folgenden werden anhand der Anforderungen eines Shared Service Centers die entsprechenden Grundlagen der Anwendungsintegration und des EAI-Konzeptes detailliert erläutert.

2.2 Techniken zur Anwendungsintegration

Statt, wie im Unternehmensszenario beschrieben, ähnliche Prozesse in mehreren Standorten durchzuführen, wird ein einheitlicher Prozess in einem zentralen Shared Service Center gefordert. Die Implementierung des Shared Service Centers basiert auf dieser Anwendungs-integration.

Aufgrund der breiten Verwendung des Integrationsbegriffs sind zunächst einige Begriffsabgrenzungen für die Anwendungsintegration erforderlich. Die im Folgenden besprochene Verwendung des Integrationsbegriffs begrenzt sich auf die Wirtschafts-informatik. Die Integrationsobjekte sind betriebliche Anwendungssysteme.

2.2.1 Betriebliche Anwendungssysteme und deren Integration

Integration wird allgemein als die „Wiederherstellung eines Ganzen durch das Verbinden oder Vereinigen logisch zusammengehörender Teile“ [KAIB02] verstanden. Das „Ganze“, das es bei der Integration betrieblicher Anwendungssysteme wiederherzustellen gilt, ist die betriebliche Realität, die durch die Summe der Anwendungssysteme mit ihren Schnittstellen korrekt abgebildet werden soll.

Anwendungssysteme sind Software- und Hardwaresysteme, mit deren Hilfe die Automatisierung informationsverarbeitender Aufgaben möglich ist. Ihr Ziel ist es, betriebliche Prozesse bestmöglich zu unterstützen und dem Anwender einen deutlichen Mehrwehrt – und damit dem Unternehmen einen Wettbewerbsvorteil – zu bieten.

Ein wichtiges Merkmal von Anwendungssystemen ist deren Heterogenität. Die Heterogenität ergibt sich aus der Verschiedenartigkeit der Anwendungen im Hinblick auf die ihnen zugrundeliegenden Betriebssysteme, Netzwerke, Hardware-Plattformen, Entwicklungsumgebungen oder –konzepte. Aus historischen Gründen wird z.B. eine alte Anwendung in einigen Anwendungsfällen oder in bestimmten Unternehmenseinheiten neben dem neuen System weiter verwendet. Ein weiterer Grund ist das Bestreben von Hardware- und Softwareanbietern, sich im Markt über die verwendeten Technologien zu differenzieren und dadurch Kunden an ihre Produkte zu binden. Insbesondere Betriebssysteme stellen eine solche „Lock-in"-Technologie dar, weil Anwendungen i.d.R. nicht auf beliebigen Betriebssystemplattformen lauffähig sind. Heterogenität ist das Hauptproblem bei der Anwendungsintegration. Deswegen wird in Integrationssystemen oftmals versucht, die Autonomie der einzelnen Anwendungssysteme einzuschränken und damit in bestimmten Aspekten Homogenität zu erzwingen. Dies kann von der Festlegung des verwendeten Systems über die verwendete Sprache bis zur exakten Definition aller relevanten Konzepte reichen [LN06].

2.2.2 Integrationsmodelle

Nachdem in den vorherigen Absätzen allgemeine Grundlagen der Anwendungsintegration erläutert wurden, geht es im Folgenden um die Wege, die Integration von heterogenen betrieblichen Anwendungen in Unternehmen und über Unternehmensgrenzen hinweg zu ermöglichen. Die Integrationstechnik umfasst Konzepte und Instrumente zur Entwicklung unternehmens- oder bereichsübergreifender, integrierter Anwendungssysteme. In diesem Abschnitt werden die drei grundlegenden Integrationsmodelle vorgestellt.

Für die Durchführung der Integrationsaufgabe stehen unterschiedliche, grundlegende Methoden zur Verfügung. Diese allgemeinen Methoden werden als Integrationsmodelle bezeichnet. Sie definieren, wie Anwendungen integriert werden, indem sie die Eigenschaften und die Mechanismen der Integration beschreiben. Die Wahl des zu verwendenden Integrationsmodells hängt von dem zu lösenden Integrationsproblem, sowie den im Unternehmen verwendeten Technologien und Entwicklungswerkzeugen ab. Im Hinblick auf die erwarteten Integrationswirkungen besitzen die verschiedenen Integrationsmodelle sehr unterschiedliche Zielerreichungsgrade [KAIB02].

Generell kann die Integration an drei unterschiedlichen Punkten eines Anwendungssystems ansetzen: der Präsentations-, der Daten- oder der Anwendungsschicht (Funktionsschicht). Entsprechend lassen sich die Präsentationsintegration, die Datenintegration und die Funktionsintegration als häufig genutzte Integrationsmodelle unterscheiden [KAIB02].

Bei der *Präsentationsintegration* wird die Anwendungsintegration über die existierenden Benutzerschnittstellen der Anwendungen realisiert (Abbildung 2). Wenn ein System keine geeigneten Schnittstellen zum Zugriff auf die Daten oder die Funktionalität bereitstellt, wird die Präsentationsintegration eingesetzt.

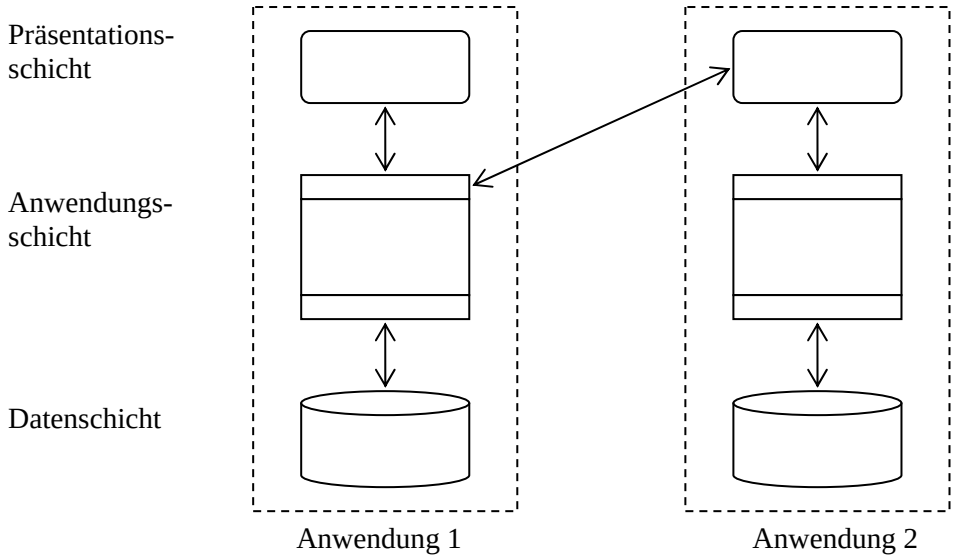

Abbildung 2: Präsentationsintegration [KAIB02]

Die Präsentationsintegration kann unter Einsatz verfügbarer und leistungsfähiger Entwicklungswerkzeuge einfach und schnell durchgeführt werden. Allerdings sind insbesondere die Performanz des Datenaustausches und die Sicherheit des Verfahrens problematisch. Daher bleibt die Präsentationsintegration eine Notlösung für die Integration.

Bei dem Konzept der *Datenintegration* erfolgt die Integration durch den direkten Zugriff auf die Daten, die von den verschiedenen Anwendungen erzeugt, verwaltet und gespeichert werden. Bei der Datenintegration werden die Präsentations- und Anwendungsschicht nicht betrachtet und der Zugriff findet direkt auf den Daten der Anwendungen statt. Die Datenintegration wird angewendet wenn Applikationen gemeinsame oder redundante Daten nutzen.

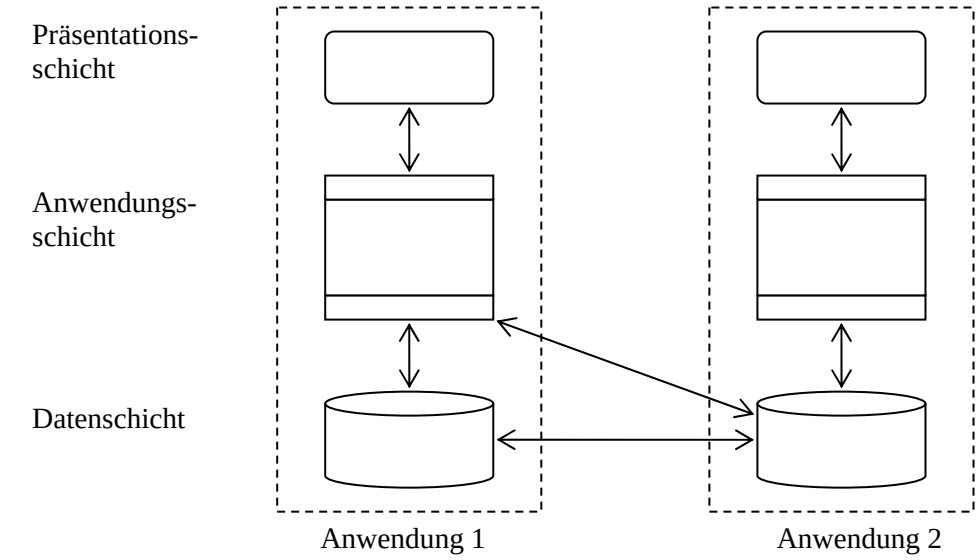

Abbildung 3: Datenintegration [KAIB02]

Der Nachteil bei diesem Integrationskonzept liegt in den möglichen Integritätsproblemen, die durch das Umgehen der Applikationslogik entstehen können.

Die *Funktionsintegration* ist das weitestgehende Konzept. Mit Funktionsintegration lassen sich eine Reihe von Integrationsaufgaben lösen, inklusive der typischen Anwendungen der Präsentations- oder Datenintegration. Sie ermöglicht eine hohe Wiederverwendung und den flexiblen Austausch der Anwendungsfunktionalitäten durch die Integration auf der Ebene der Anwendungslogik. Aus diesem Grund wird die Funktionsintegration als das am weitesten entwickelte Konzept angesehen.

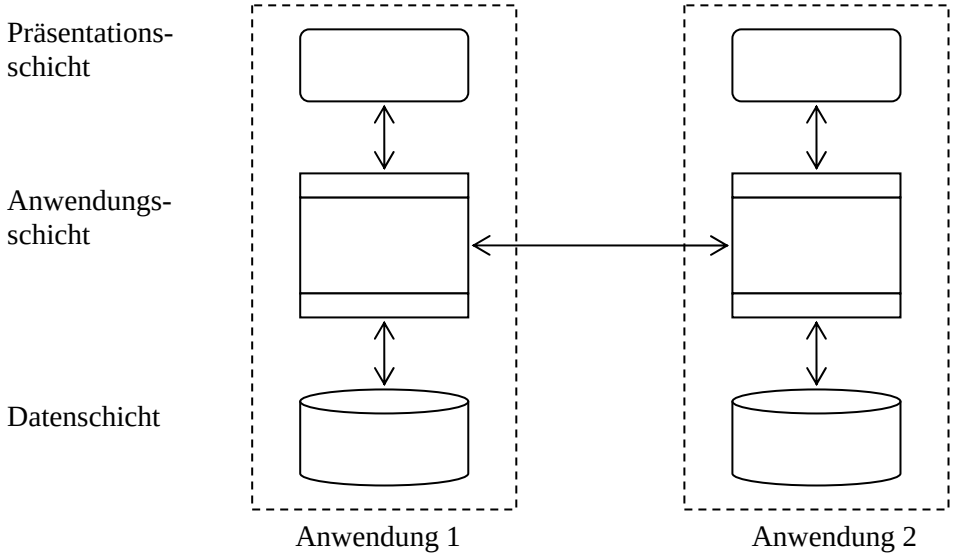

Abbildung 4: Funktionsintegration [KAIB02]

Bei diesem Integrationskonzept ruft eine Anwendung eine Methode auf, die von einer anderen Anwendung bereitgestellt wird. Im Unterschied zur Datenintegration wird hier die Anwendungslogik nicht umgangen. Die Sicherstellung der Konsistenz geschieht auf der Funktionsebene. Auf diese Weise können existierende Funktionalitäten durch andere Systeme genutzt und flexibel kombiniert werden.

Wesentliche Nachteile der Funktionsintegration liegen in ihrer Komplexität bei der Realisierung. Notwendige Änderungen an den zu integrierenden Anwendungen können mit Risiken und Kosten verbunden sein.

2.2.3 Integrationsansätze

Von den Integrationsmodellen als grundlegende Methoden der Anwendungsintegration können pragmatische Integrationsansätze in Unternehmen unterschieden werden. Diese beschreiben aus einem situativen Ansatz heraus Wege, auf denen die oben geschilderten

Integrationsmodelle verwirklicht werden können. Im Folgenden werden drei traditionelle Ansätze, die zur Anwendungsintegration dienen, beschrieben. Die Auswahl des geeigneten Integrationsansatzes hängt maßgeblich von der Ausgangssituation, insbesondere der Systemlandschaft im Unternehmen, und von den verfolgten Integrationszielen ab.

- Punkt-zu-Punkt-Verbindung

  Unter einer Punkt-zu-Punkt-Verbindung versteht man die dedizierte Verbindung zwischen zwei gleichberechtigten Systemen bzw. Anwendungen. Zur Integration einer Vielzahl von Anwendungssystemen sind entsprechend zahlreiche solcher Punkt-zu-Punkt-Verbindungen zwischen den jeweiligen Anwendungspaaren erforderlich. Dies bedeutet, dass immer wenn neue Anwendungen in einen bestehenden Systemverbund eingefügt werden sollen, müssen die Verbindungen zwischen Systempaaren je nach Bedarf eingerichtet werden. Zu keinem Zeitpunkt muss man die Systemlandschaft umfassend neu organisieren. Das Ergebnis sind die im Laufe der Zeit gewachsenen Verbindungsstrukturen einer komplexen, oft unübersichtlichen Schnittstellenlandschaft.

  Die Nachteile liegen vor allem in den hohen Betriebskosten für die Wartung der Schnittstellen und in dem hohen Aufwand bei der mehrfachen Integration zusätzlicher Anwendungen. Zudem erschwert die komplexe Schnittstellenstruktur die reibungslose Integration durch die suboptimale Nutzung der Ressourcen [KAIB02].

  Abbildung 5 zeigt das Ergebnis bei der Verwendung der Punkt-zu-Punkt-Integration.

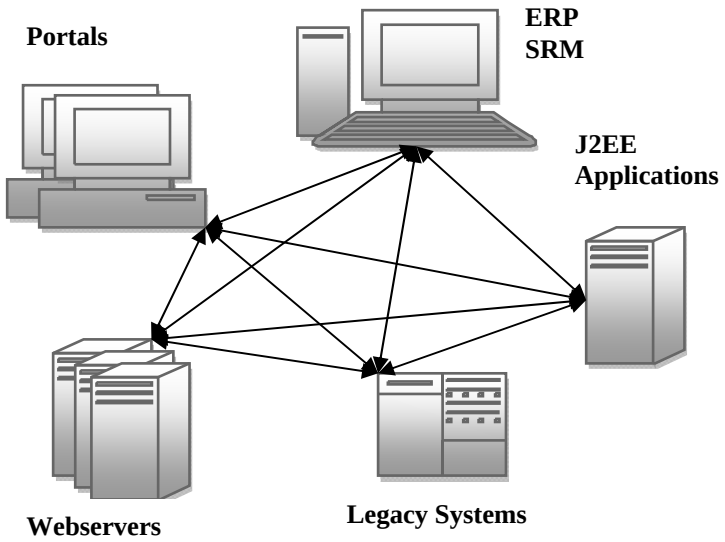

Abbildung 5: Punkt-zu-Punkt-Integrationen

- ERP-basierte Integration

  Ein Enterprise Resource Planning (ERP) System ist eine komplexe Anwendungssoftware zur Unterstützung der Ressourcenplanung eines gesamten Unternehmens. Die Einführung von ERP-Paketen ist für viele Unternehmen ein wesentlicher Fortschritt bezüglich der Anwendungsintegration. Der Integrationsbedarf sinkt, da die umfangreichen funktionalen Module der ERP-Pakete vom Hersteller in vorintegrierter Form angeboten werden. Das ERP-Paket als solches ist in der Lage, wesentliche Geschäftsprozesse umfassend zu unterstützen. Da die Funktionalitäten des ERP-Pakets durch branchen- und/oder unternehmensspezifische Lösungen ergänzt werden sollen, besteht nach Einführung eines ERP-Pakets der Bedarf zur Integration betriebswirtschaftlicher Anwendungen weiter.

  Es bleibt aber anzuerkennen, dass ERP-Pakete in vielen Unternehmen die Rolle der zentralen Kernanwendung eingenommen haben. Hierauf baut der ERP-basierte Integrationsansatz auf. Das ERP-Paket übernimmt hier die Rolle einer zentralen Integrationsplattform, indem Einzelanwendungen für den spezifischen Bedarf auf dem gemeinsamen ERP-„Backbone“ aufgesetzt werden.

  Insgesamt erscheint der ERP-basierte Ansatz nur dann als sinnvoll, wenn tatsächlich ein Großteil der Anwendungsunterstützung durch das zentrale ERP-Paket eines Anbieters unterstützt wird und wenn nicht gleichzeitig ein hoher Abstimmungsbedarf zwischen den verteilten Anwendungen besteht. Sind diese Voraussetzungen gegeben, erscheint der ERP-basierte Integrationsansatz als durchaus attraktiv, insbesondere vor dem Hintergrund der zunehmenden Öffnung der ERP-Systeme durch weitere standardisierte Schnittstellen.

- Middleware-basierter Integrationsansatz

  Beim Middleware-basierten Integrationsansatz (Abbildung 6) wird zur Lösung des Integrationsproblems eine vermittelnde Softwareschicht, sog. Middleware, zwischen zwei oder mehrere Systeme geschaltet. Diese ermöglicht es den angebundenen Anwendungen, hersteller- und gegebenenfalls plattformunabhängig Daten auszutauschen. Auf diese Weise erfolgt die Integration in einer stärker standardisierten Weise, als dies bei Punkt-zu-Punkt-Verbindungen der Fall ist. Entsprechend geringer sind in der Regel die Integrationskosten und die Projektdauer.

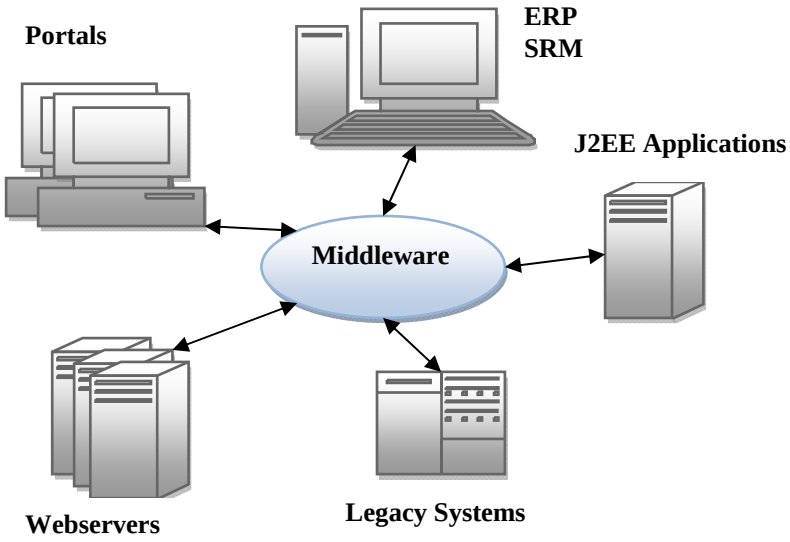

Abbildung 6: Middleware-basierte Integration

Middleware schafft die Konnektivität zwischen verschiedenen Anwendungen und unterstützt somit die Datenintegration.

Während der Middleware-basierte Integrationsansatz im Hinblick auf die mit Punkt-zu-Punkt-Verbindungen verbundenen Nachteile durch die stärkere Standardisierung des Vorgehens eine Verbesserung darstellt, bleibt jedoch ein großer Entwicklungsaufwand. Insbesondere da Middleware sich auf die Datenintegration auf syntaktischer Ebene beschränkt, werden neben der Auswahl und Implementierung der Middleware weitere Entwicklungsmaßnahmen zur Datentransformation, zur Prozesssteuerung und unter Umständen zur Transaktionsverwaltung notwendig.

Diese drei Grundtypen sind als mögliche Vorgehensweisen zu verstehen. In der Realität sind in Unternehmen viele Mischformen dieser Grundtypen vorzufinden[LN06].

2.2.4 Integration über den EAI-Ansatz

Enterprise Application Integration (EAI) ist ein Konzept zur unternehmensweiten Integration von Geschäftsprozessen, die über verschiedenen Applikationen auf unterschiedlichen Plattformen verteilt sind.

Es geht bei der Enterprise Application Integration (EAI) darum, systemübergreifende Geschäftsprozesse mit Hilfe einer geeigneten Technik so abzubilden, dass die unterschiedlichen, am Geschäftsprozess beteiligten Systeme in der Lage sind, prozessrelevante Daten untereinander in geeigneter Form austauschen zu können und dabei ein gleiches Verständnis dieser Daten haben.

Unter dem Begriff Enterprise Application Integration werden Lösungsansätze für die interne als auch die externe Integration vorhandener Anwendungen verstanden. EAI beschreibt dabei nicht nur die Integrationsansätze und Integrationsmodelle, sondern behandelt auch die zur Realisierung benötigten Technologien. EAI unterstützt die Daten-, Programm- und Prozessintegration und umfasst die Planung, die Methoden und die Software, um heterogene, autonome Anwendungssysteme zu integrieren. Das Ziel von EAI besteht in der durchgängig automatisierten Unterstützung von Geschäftsprozessen. Diese Prozessautomatisierung soll durch eine nahtlose Integration aller Anwendungssysteme, die an diesen Prozessen beteiligt sind, erreicht werden. Auf diese Weise wird ein aus Benutzersicht einheitliches, virtuelles System geschaffen, das die Komplexität der darunter liegenden technologischen Lösungen verbirgt [KAIB02].

In diesem Abschnitt wird zunächst der Integrationsansatz von EAI näher beschrieben. Im Rahmen der Erläuterung der wesentlichen Elemente einer EAI-Architektur werden im Abschluss typische Topologien und wesentliche Standards zum Datenaustausch diskutiert.

2.2.4.1 Der Lösungsansatz

EAI ist ein neuer Ansatz zur umfassenden Integration heterogener betrieblicher Anwendungen auf Daten-, Programm- und Prozessebene innerhalb eines Unternehmens und über Unternehmensgrenzen hinweg. Er erlaubt die Kommunikation zwischen verschiedenen Systemen und Anwendungen in einer heterogenen Systemlandschaft und baut auf dem dargestellten Middleware-basierten Integrationsansatz auf. Erklärtes Ziel von EAI ist es zudem, Integrationsmaßnahmen mit minimalen oder gar keinen Veränderungen an den existierenden Anwendungen oder Daten zu ermöglichen [KAIB02].

Ein charakteristisches Merkmal von EAI-Lösungen ist ihre Schnittstellenkonzeption. Anstelle des Designs und der Entwicklung einer Reihe von Punkt-zu-Punkt-Verbindungen sind hier alle Anwendungen gleichberechtigt mit einer zentralen EAI-Komponente verbunden. Über diese zentrale Komponente steht jede verbundene Anwendung mit allen anderen verbundenen Systemen oder Anwendungen in Verbindung, um Nachrichten oder Daten auszutauschen. Im Idealfall besitzt so jede Anwendung nur eine Schnittstelle – und zwar die zur zentralen EAI-Software. Zur physischen Realisierung kommen dabei unterschiedliche EAI-Architekturen in Frage. Potentielle Integrationsobjekte sind kommerzielle Standardanwendungen, Datenbanken, Dateien, Individualanwendungen, Middleware etc. – und zwar unabhängig von Herstellern, Betriebssystemen oder Entwicklungskonzepten.

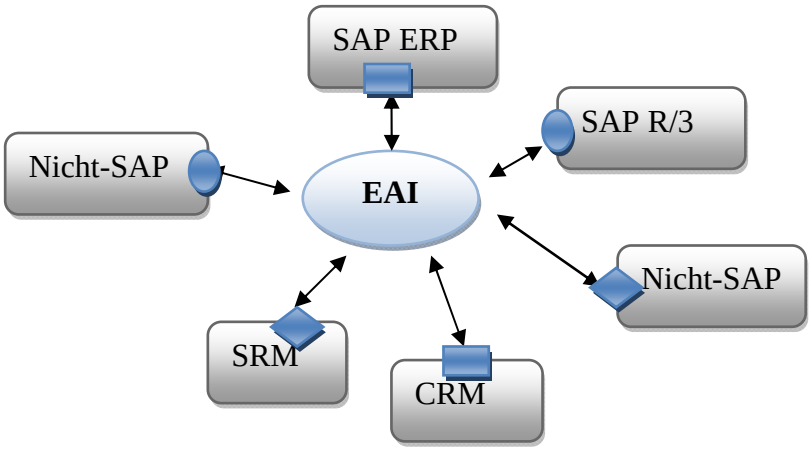

Abbildung 7: EAI-Lösung

EAI ist eine Erweiterung der traditionellen Integrationsansätze, indem dieser Ansatz die Integration über die Daten-, Programm- und Prozessebene hinweg unterstützt.

2.2.4.2 Vorstellung einer möglichen EAI-Integrationslösung

Nachdem im vorherigen Abschnitt die Grundlagen der Anwendungsintegration erläutert wurden, geht es im Folgenden zurück zum Unternehmensszenario und dessen Anforderungen. Das Unternehmen DIwS AG plant die Einrichtung eines zentralen Shared Service Centers, damit die Prozesse, die zwar dezentral verteilt sind, aber gleiche oder ähnliche Anforderungen haben, an zentraler Stelle vereinigt werden können, um dort die entsprechenden Daten zu bearbeiten. Die Verschiebung der Prozesse von den verteilten Systemen hin zu einem zentralen System benötigt nur die Installation prozessorientierter Software. Das Problem ist, wie die neuen zentralisierten Prozesse auf Daten (z.B. Rechnungen) aus den verschiedenen Anwendungssystemen zugreifen können und wie die Daten aus verschiedenen Datenquellen integriert werden können, um eine einheitliche Sicht für die Anwendung anzubieten.

Wie im letzten Abschnitt schon erwähnt, unterstützt das EAI-Konzept auch die Datenintegration. Dieses Konzept wird hier anhand der realen Situation des Unternehmens und dessen Anforderungen ausgewählt.

Diese Arbeit konzentriert sich im weiteren Verlauf auf die Integration im Sinn der Daten als Integrationsgegenstand. Diskutiert wird dann eine Datenintegration für verteilte Anwendungssysteme, ohne zu berücksichtigen, ob es sich um eine inner- oder zwischenbetriebliche Integration handelt. Die Anwendungssysteme werden dann als Datenquellen abstrahiert. Der Grund für die Verwendung der Datenintegration wird später durch das Unternehmensszenario verdeutlicht.

Im Folgenden werden die Grundlagen dieses Integrationskonzeptes weitergeführt. Zuerst wird der allgemeine Datenintegrationsbegriff definiert und darauf basierend wird sich die Arbeit der EAI-spezifischen Datenintegration zuwenden.

2.2.5 Datenintegration

In dieser Arbeit wird die allgemeine Datenintegration nicht im Detail betrachtet, sondern die Integration im Zusammenhang mit Geschäftsprozessen. In den nächsten Abschnitten wird zunächst die Datenintegration allgemein kurz erklärt, darauf aufbauend wird dann deren Anwendbarkeit in Unternehmen erläutert.

2.2.5.1 Datenintegration allgemein

Datenintegration wird allgemein als ein Synonym für den Begriff „Informationsintegration“ verwendet [LN06], der die Integration von Daten bezeichnet, die aus verschiedenen Datenbeständen (Datenquelle) mit in der Regel unterschiedlichen Datenstrukturen stammen. Das Ziel der Datenintegration ist es, den Zugriff auf eine Reihe bestehender Informationssysteme durch ein zentrales, integriertes Informationssystem zu steuern, damit eine einheitliche Sicht auf die Datenquellen zur Verfügung steht. (Buch: Informationsintegration) Verschiedene Anwendungen greifen auf das integrierte System mittels definierter Softwareschnittstellen zu. Die bestehenden Informationssysteme können vielfältig sein: klassische relationale Datenbanksysteme, Dateien, Daten, auf die man mittels Web-Services oder HTML-Formularen zugreift, Daten produzierende Anwendungen oder auch andere integrierte Informationssysteme.

Die untere Abbildung 8 stellt eine mögliche Grundarchitektur der Datenintegration dar.

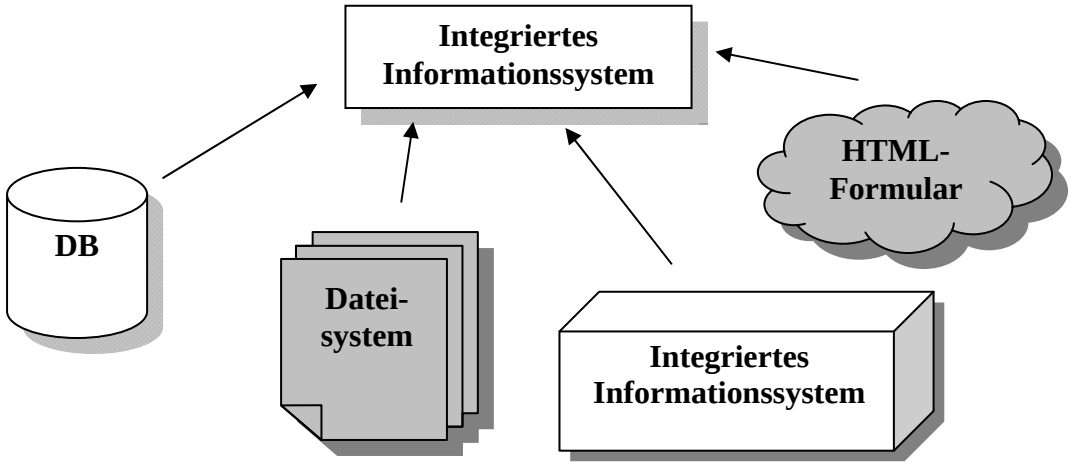

Abbildung 8: Architektur der Datenintegration

Die Aufgabe eines integrierten Informationssystems ist es, einen Zugang zu einer Menge von Datenquellen zu schaffen. Für den Benutzer soll der Umgang mit dem System möglichst einfach sein, was bedeutet, dass ein hoher Grad an Transparenz erreicht werden sollte.

An dieser Stelle wird nach [LN06] eine Reihe von Begriffen, die zum Verständnis der Datenintegration notwendig sind, erläutert. Einige werden im Laufe der Arbeit noch eine genauere Definition erfahren.

- Verteilte Datenbestände und integrierte Informationssysteme

  Datenbestände sind beliebig aufgebaute Informationsspeicher, z.B. eine Datenbank oder ein Dateisystem. Sie sind der Ursprungsort, an dem die Daten generiert und/oder gespeichert wurden. Bei *verteilten Datenbeständen* wird davon ausgegangen, dass mehrere Datenbestände zwar physisch voneinander unabhängig arbeiten, aber wie ein einziges logisches System erscheinen. Die Daten der verschiedenen Datenbestände werden zu einer einheitlichen Informationsmenge zusammengeführt, die dann als ein *integriertes Informationssystem* bezeichnet wird.

- Informations- / Datenintegration

  Die Daten aus verschiedenen Datenbeständen haben in der Regel unterschiedliche Datenstrukturen. Das Zusammenführen von Informationen erfordert aber eine gemeinsame einheitliche Datenstruktur, die durch den Prozess *der Informations- oder Datenintegration* realisiert werden kann. Dabei sollen alle heterogenen Datenbestände möglichst vollständig und effizient zu einer strukturierten Einheit zusammengeführt werden.

- Transparenter Zugriff

  Die verteilten Datenbestände werden in ein gemeinsames Informationssystem integriert und der Systemanwender benötigt in der Regel keine detaillierte Kenntnis über bestimmte interne Informationen. Anders ausgedrückt, bietet das integrierte Informationssystem einen transparenten Zugriff auf die integrierten Datenquellen. Anstatt dass der Benutzer selber explizit oder implizit die Datenquelle auswählt, arbeitet er mit dem integrierten System, ohne zu wissen, welche Datenquellen im System integriert sind, welche gerade verwendet werden oder in welchen Datenquellen welche Daten vorhanden sind. Die Interna des Systems sind für ihn unsichtbar.

Die Datenintegration im allgemeinen Sinne verbirgt sich hinter vielen Begriffen wie etwa Informationsfusion oder Datenkonsolidierung.

2.2.5.2 Datenintegration im EAI-Konzept

Bei dem Konzept der Datenintegration im Kontext der Enterprise Application Integration erfolgt die Integration durch den direkten Zugriff auf die Daten, die von den verschiedenen Anwendungen erzeugt, verwaltet und gespeichert werden [KAIB02]. Im Vergleich zur allgemeinen Datenintegration mit dem Zweck der Informationsintegration beschäftigt sich diese Anwendungsintegration mit der Integration von IT-Prozessen. Hier werden die Anfragen durch den Austausch von Nachrichten realisiert. Trotzdem haben die in den beiden Formen der Integration anfallenden Aufgaben Gemeinsamkeiten, wie die Abbildung heterogener Strukturen oder Probleme bei der Semantik von Begriffen.

Das Konzept der Datenintegration lässt sich auf zwei Weisen verwirklichen. Zum einen kann der direkte Austausch von Daten zwischen verschiedenen, spezifizierten Datenbanken ermöglicht werden, zum anderen kann der flexible Zugriff auf eine Vielzahl von Datenbanken durch die Verwendung eines einheitlichen konzeptuellen Datenschemas im Sinne einer virtuellen Datenbank realisiert werden. In diesem Fall spricht man auch von föderierter Datenintegration. Ein alle Daten umfassendes Datenschema im Sinne eines Unternehmens-datenmodells ist jedoch aufgrund seiner Komplexität und des hohen Erstellungsaufwands meist nicht realisierbar. Die damit verbundenen Schwierigkeiten, werden in einem Forschungsgebiet der Informatik bearbeitet (Semantische Integration [Hul97]). Vielmehr sind, basierend auf den wesentlichen Geschäftsprozessen, Datenmodelle für Teilbereiche aufzustellen [KAIB02] – d.h. es ist z.B. möglich, ein Datenmodell für alle eingehenden Rechnungen zu erstellen.

Wie vorher schon erwähnt, ist der zentrale Bestandteil einer EAI-Lösung die darunter liegende Middleware. Diese unterstützt die Integration auf Datenebene, indem sie den Austausch von Informationen zwischen zwei oder mehreren Anwendungssystemen in einer heterogenen Landschaft ermöglicht. Middleware bezeichnet allgemein eine Softwareschicht zwischen betrieblichen Anwendungen und Systemsoftware. Diese Softwareschicht stellt auf Basis standardisierter Schnittstellen und Protokolle Dienste für eine transparente Kommunikation verschiedener Komponenten in einem heterogenen und verteilten Umfeld zur Verfügung. Zum Zweck der Datenintegration im Sinne des EAI-Ansatzes wird dann ein datenzugriffsorientiertes Produkt gewählt. Unter dem Begriff datenzugriffsorientierte Middleware werden verschiedene, erprobte Technologien zusammengefasst, die über eine einheitliche Schnittstelle den transparenten Zugriff auf heterogene Daten und die Replikation und Transformation derselben bei der Übertragung ermöglichen. Sie schaffen damit den Ausgleich zwischen den Datenbanktypen unterschiedlicher Hersteller, den verschiedenen Datenbankmodellen und den verwendeten Datenschemata. Ziel ist die integrierte Sicht auf verteilte Daten im Sinne einer einzelnen „virtuellen“ Datenbank.

Ein wesentlicher Vorteil dieses Integrationskonzeptes liegt darin, dass es keine Modifikationen der Datenbanken oder der Anwendungslogik der zu integrierenden Anwendungen erfordert. Dies ermöglicht oft eine schnelle Lösung spezifischer

Integrationsprobleme und reduziert die mit der Integration verbundenen Risiken und Kosten [KAIB02].

2.2.5.3 Probleme bei der Datenintegration

Die umfassende Integration heterogener betrieblicher Anwendungssysteme ist außerordentlich komplex. Unabhängig vom gewählten Lösungsansatz oder von der unterstützenden Technologie entstehen eine Reihe kritischer Faktoren im Rahmen der Planung von Integrationsvorhaben. Solche kritischen Faktoren sollten bei der Entwickelung des Integrationssystems berücksichtigt werden.

Aus der Natur der Informationsverarbeitung ergeben sich:

- Die Notwendigkeit einer vollständigen Erfassung aller Vorgänge

  Es ist zu beachten, dass die Integration aufgrund der Datenabhängigkeiten die vollständige Erfassung aller vom System unterstützten Vorgänge erfordert.

- Spezielle Risiken bezüglich Performanz und Sicherheit der Informationsverarbeitung.

  In einem integrierten Gesamtsystem hängen die Performanz und Sicherheit der einzelnen Anwendungskomponenten nicht mehr ausschließlich von ihren spezifischen Eigenschaften ab, sondern individuelle Verarbeitungsengpässe oder Sicherheitsrisiken wirken sich auch auf das Gesamtsystem aus.

  Im Hinblick auf die Performanz besteht in integrierten Anwendungssystemen die Gefahr, dass gerade die Middleware oder zentrale EAI-Komponenten, deren Aufgabe einzig in der Integration der Anwendungssysteme liegt, zum Engpass des Gesamtsystems werden. Daher sind besondere Anforderungen an die Stabilität, Verfügbarkeit und die Skalierbarkeit dieser Integrationskomponenten zu stellen.

  Ähnlich verhält es sich mit der Sicherheit bei der Informationsverarbeitung. Durch die Integration der Systeme erhöht sich im Vergleich zu isolierten Systemen mit einigen wenigen klaren Schnittstellen die Anzahl der potentiellen Angriffspunkte auf zentrale Anwendungskomponenten. So besteht z.B. die Gefahr, dass über Middleware direkt auf Anwendungsdaten unter Umgehung der Applikationslogik mit entsprechenden Login- oder Authorisierungsfunktionalitäten zugegriffen wird.

Außer den oben genannten Problemen bei der Datenintegration sind noch die folgenden Aspekte zu berücksichtigen:

- Probleme bei Ausfall eines einzelnen Systems
- Erweiterbarkeit des integrierten Informationssystems

- Eindeutigkeit der globalen Daten

Datenintegration ist ein Integrationsansatz für Unternehmen. Die service-orientierte Architektur ist eine mögliche Infrastruktur, die eine Integration von Daten realisieren kann. In den folgenden Abschnitten werden die wichtigsten Grundlagen einer service-orientierten Architektur eingeführt.

2.3 Service-orientierte Architektur

Mit dem Schlagwort Service-Orientierte Architektur sind zurzeit viele Versprechungen und Hoffnungen verknüpft. Um einen besseren Überblick über SOA zu gewinnen, wird in diesem Abschnitt zuerst die Idee einer Service-Orientierten Architektur erklärt. Die Rollen und Aktionen in einer Service-Orientierten Architektur, sowie die Komponenten und Technologien der Web Services werden näher betrachtet. Anschließend werden zwei konkrete Modelle des SOAs als Beispiele vorgestellt. Ein Modell ist das heute im Markt bekannte Modell: die Web-Service-orientierte Architektur. Zum Vergleich wird noch das von der SAP AG entwickelte Modell – die Enterprise-Service-orientierte Architektur – vorgestellt.

Die Idee der Service-Orientierten Architektur geht auf Hewlett-Packard Co. zurück. Im Jahr 1999 sollte mit e-speak eine Plattform geschaffen werden, auf der Daten und Funktionalitäten in der Form von Services zur Verfügung stehen. Aufgrund der Marketingstrategie ist die Plattform ein Jahr später gescheitert, dennoch ist der grundlegende Gedanke der SOA geblieben [HL04].

2.3.1 Definition und Anforderungen

Was ist SOA? Nach der Definition ist SOA eine unternehmensweite, verteilte Softwarearchitektur für betriebliche Informationssysteme. Die elementaren Bestandteile von SOA sind Dienste/Services. Nach [DJMZ05] versteht man unter einem Dienst ein Programm oder eine Softwarekomponente, die über ein Netzwerk benutzt werden kann. Das Hauptziel der SOA ist die Erhöhung der IT- und Geschäftsflexibilität. Daraus ergeben sich für das Unternehmen Produktivitäts-, Wirtschaftlichkeits- und Agilitätssteigerungen.

An SOA als Architektur werden bestimmte Anforderungen gestellt. In der Abbildung 9 wird das Grundkonzept der SOA dargestellt und die Anforderungen zusammengefasst.

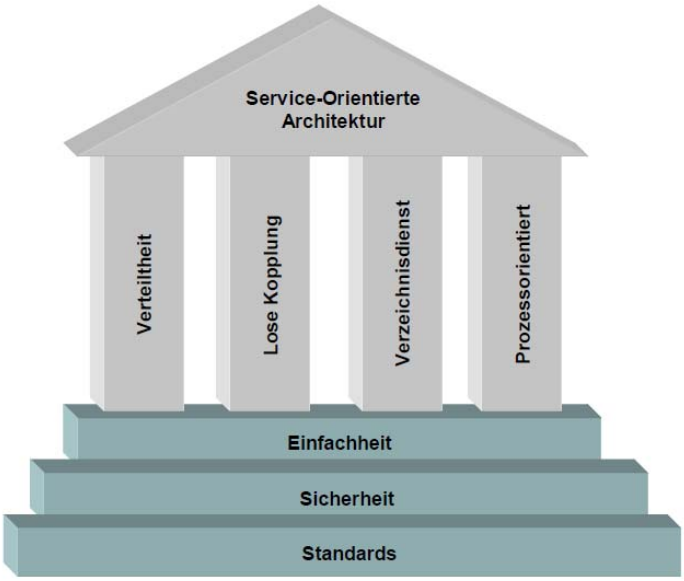

Abbildung 9: SOA-Tempel [DJMZ05]

Das Grundkonzept geht davon aus, dass das Fundament der SOA von offenen Standards, Sicherheit und Einfachheit gebildet wird.

- Offene Standards

  Durch offene Standards wird sichergestellt, dass der Nutzer mit jedem Dienst kommunizieren kann. Zudem führt die Verwendung von offenen Standards dazu, dass man Unabhängigkeit von Softwareanbietern und einer bestimmten Technologie erreicht. Sie bieten Investitionssicherheit für neu erstellte Software und ermöglichen im Idealfall die einfache Integration von Produkten verschiedener Hersteller [DJ04].

- Sicherheit

  Bei der Entwicklung von Anwendungen bietet SOA eine Reihe von positiven Eigenschaften. Durch diese Vorteile sind die generierten Anwendungen aber auch zahlreichen Sicherheitsrisiken ausgesetzt. Werden in einem Unternehmen bestimmte Sicherheitsanforderungen, vor allem im Bezug auf die Nutzung externer Services, nicht eingehalten, kann dies schnell zu Haftungs- und Imageschäden führen. Tabelle 1 enthält die wesentlichen Sicherheitsaspekte im Überblick [WIEH04].

Sicherheitsaspekt	Beschreibung	Mögliche Lösungen
Vertraulichkeit	Abhören von Daten verhindern bzw. Vertrauliche Daten vor unbefugtem Zugriff schützen	Verschlüsselung, Autorisierungsschemata
Integrität	Unbefugtes Verändern von Daten verhindern	MAC (Message Authentication Codes)
Verfügbarkeit	Ununterbrochener Zugriff	Backup, Replikation, Firewall
Authentizität	Nachweis des Ursprungs von Daten	Digitale Unterschrift

Tabelle 1: Sicherheitsaspekte und Lösungen [SCHA03]

- Einfachheit

  Das Ziel eines IT-Systems muss es sein, die Komplexität zu reduzieren und die Systemfunktionalität zu beherrschen. SOA hilft hier, indem durch die Verwendung von Services, die Anwendungslandschaft klar und verständlich strukturiert wird. Zudem werden die technischen Implementierungsdetails durch definierte Schnittstellen gekapselt, was eine Konzentration auf die fachlichen Aspekte ermöglicht [BW06] und [DJ04].

  Auf den oben beschriebenen drei Fundamenten ruhen vier Säulen: Verteiltheit, lose Kopplung, Verzeichnisdienst und Prozessorientierung. Sie wiederum stützen das Dach der Service-Orientierten Architektur.

- Verteiltheit

  SOA ist ein Ansatz der Informationstechnik aus dem Bereich der verteilten Systeme, in dem sich Hardware- und Softwarekomponenten auf vernetzten Computern befinden und miteinander durch den Austausch von Nachrichten kommunizieren. Verteilte Systeme werden von verteilten Anwendungen gebraucht, um eine in sich geschlossene, spezialisierte Funktionalität zur Verfügung zu stellen [HAMM05].

- Lose Kopplung

  Unter loser Kopplung versteht man einen Architekturansatz, bei dem die Abhängigkeiten zwischen Dienstanbietern und Dienstnutzern auf ein Minimum reduziert werden. Durch Programmiersprachen- und Betriebssystemunabhängigkeit und den Einsatz von standardisierten Technologien wie XML erreichen Web Services dieses Ziel. Die lose Kopplung der Dienste ermöglicht ihre dynamische Suche und flexible Einbindung in Anwendungen und andere Dienste [HANS05].

- Verzeichnisdienst

  Die angebotenen Dienste sind alle in einem Verzeichnisdienst registriert, damit die Dienstnutzer den gewünschten Dienst einfach finden können.

- Prozessorientiert

  Die Hauptaufgabe von IT-Produkten besteht in der Unterstützung der Geschäftsprozesse eines Unternehmens. Diese Aufgabe kann durch den Einsatz von SOA realisiert werden. Mit Hilfe einer SOA sind Geschäftsprozesse schnell, effizient und kostengünstig an neue Marktbedingungen anpassbar [STRN06].

Jeder Service bietet eine wohl definierte Funktionalität an. Er ist selbstständig und unabhängig von anderen Services, was die Wiederverwendbarkeit ermöglicht. Wegen der losen Kopplung werden Services dynamisch gesucht, gefunden und eingebunden. In der Abbildung 10 wird der grundlegende Ablauf dargestellt.

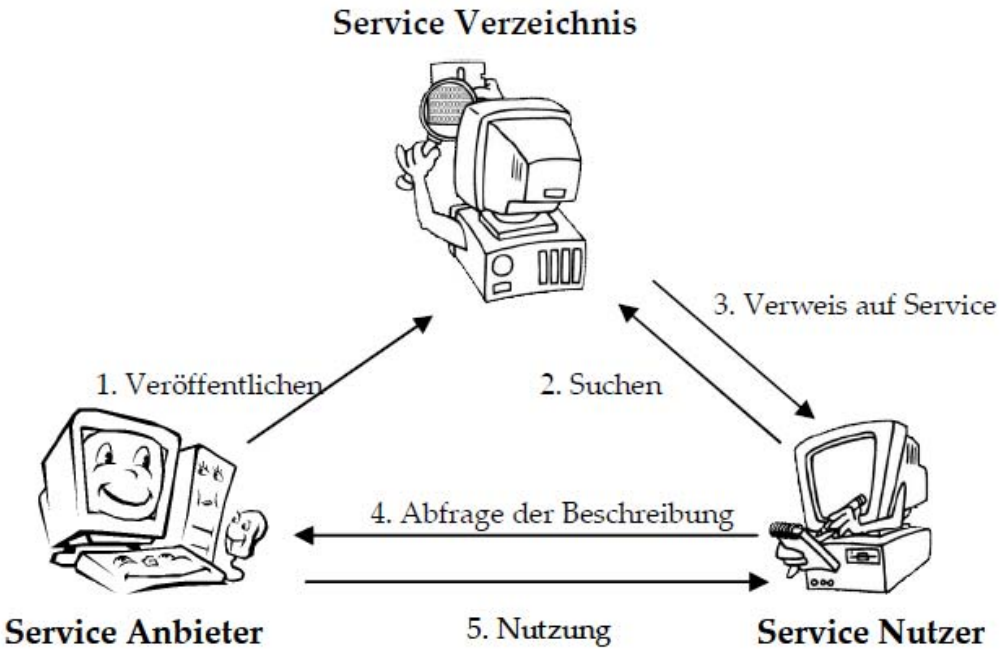

Abbildung 10: Das magische Dreieck einer SOA [DJMZ05]

Ein *Serviceanbieter* bietet einen Service an und erstellt von diesem eine Schnittstellenbeschreibung. Um diesen Service bekannt zu machen, muss diese Schnittstellenbeschreibung in einem *Serviceverzeichnis* publiziert werden. Der *Servicenutzer* sucht mit Hilfe der Servicebeschreibung einen geeigneten Service in dem Verzeichnis aus und bekommt als Ergebnis eine Referenz zum Service Anbieter. Mit dieser Referenz kann der Nutzer direkt den Service vom Anbieter aufrufen und verwenden.

2.3.2 Technische Umsetzungen

Grundsätzlich ist SOA ein Architekturkonzept, welches keine konkrete technische Realisierung oder bestimmte Methoden vorschreibt. So kann eine SOA mit CORBA, RMI, Web Services oder anderen Technologien umgesetzt werden. Im Rahmen dieser Masterarbeit werden beide Varianten, Web Services und die Enterprise Service-orientierte Architektur (eSOA), vorgestellt.

Web Services besitzen zurzeit die breitere Akzeptanz. Im Kontext Service-Orientierter Architekturen sind die verwendeten Protokolle sehr attraktiv und werden deshalb häufig eingesetzt.

Die eSOA ist das von SAP AG entwickelt Modell für eine service-orientierte Architektur. Die Grundbausteine der eSOA sind Enterprise Services.

2.3.2.1 Web Service als Implementierungstechnologie

Der Web Service steht im Mittelpunkt einer konkreten Implementierung des SOA-Konzeptes. Mit Web Services lässt sich SOA besonders leicht und wirkungsvoll umzusetzen, weil diese standardisiert und plattformunabhängig sind.

Ein Web Service ist ein Softwaresystem, das die Interaktion zwischen Anwendungen über das Netzwerk unterstützen kann [W3C01]. Auf dessen Funktionalitäten kann über das Internet mit einer Kombination von Protokollen wie HTTP, XML oder SMTP zugegriffen werden.

Die Web Services sind in der Lage, entfernte Funktionsaufrufe von beliebigen Plattformen und Programmiersprache zu dekodieren und an die angesprochene Anwendung weiterzureichen. Sie ermöglichen damit eine neue plattformunabhängige Kommunikation zwischen Applikationen.

Web Services sind also keine monolithischen Systeme, sondern eigenständige Anwendungen, welche mit Internettechnologien beschrieben beziehungsweise über das Internet veröffentlicht, gesucht und aufgerufen werden können. Sie stellen somit ein gutes Konzept zur Realisierung einer Architektur in verteilten Systemen dar und können überall angeboten und beliebig benutzt werden, wo Standard-Internettechnologien zum Einsatz kommen. Die Systemkomponenten eines Web Services sind in der Abbildung 11 dargestellt.

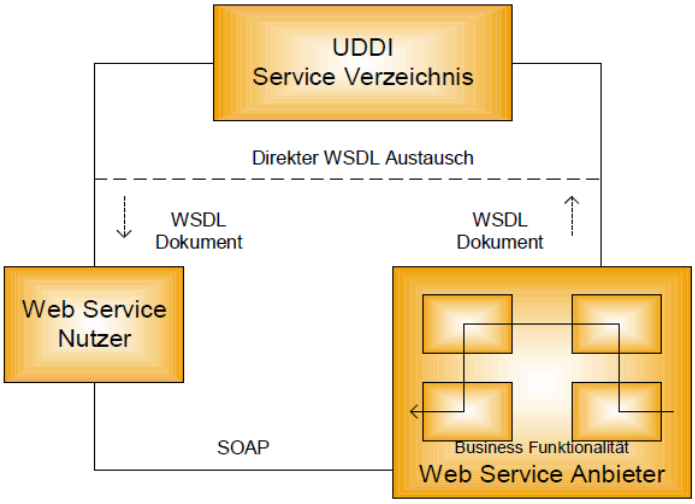

Abbildung 11: Basiskomponenten eines Web Services [WM05]

Der Web-Service-Anbieter stellt einen Web Service und die entsprechende Beschreibung in der *Web Service Definition Language* (WSDL) [W3C02] bereit. Diese Beschreibung ist eine der wichtigsten Elemente der Web-Service-Technologie. Sie erlaubt es unabhängigen Parteien, miteinander zu kommunizieren. Somit können Systeme flexibel aus unterschiedlichen Komponenten zusammengesetzt werden. Eine WSDL-Beschreibung ist ein XML-Dokument.

Die Kommunikation erfolgt durch SOAP-basierte Nachrichten [W3C03]. SOAP steht für *Simple Object Access Protocol* und beschreibt das XML-basierte Nachrichtenformat der Kommunikation. XML (Extensible Markup Language) [W3C04] ist ein Satz an Regeln für die Erstellung von Textformaten zur Datenstrukturierung [CGIH+2002]

2.3.2.2 Enterprise Service-Orientierte Architektur

Die Enterprise-Service-orientierte-Architektur (eSOA) ist ein von der Firma SAP AG entwickeltes SOA-Modell. Der Nutzen der eSOA liegt in der Unterstützung der Standardisierung und der Anpassungsfähigkeit in einer Unternehmungsumgebung. eSOA erweitert das Konzept der Web Services. Die zugrundeliegenden Enterprise Services sind aus technischer Sicht Web Services, die mit spezieller Geschäftslogik erweitert werden [HK07].

Nach der SAP-internen Servicebereitstellung werden Enterprise Services gemäß ihrer Semantik und gewissen technischen Attributen wie folgt unterschieden [BBBJ+06]:

- B2B (Business-to-Business): steht für die Kommunikation zwischen zwei Unternehmen, z.B. zwischen einem Großkunden und einem Lieferanten. Die B2B Services folgen internationalen Standards wie RosettaNET[2], UN/EDIFACT[3], usw.

- A2A (Application-to-Application): bezeichnet die Unternehmenskommunikation zwischen zwei Anwendungen. Hier sind die beiden Seiten der Kommunikation bekannt, z.B. von SAP-System zu SAP-System.

- A2X (Application-to-X): bezeichnet Services, bei denen der Servicenutzer nicht bekannt ist. Sie werden beispielweise für Benutzeroberflächen verwendet.

Die eSOA kann als eine Client-Server-Architektur betrachtet werden [WOOD04]. Die Rolle des Servers spielt die SAP NetWeaver [SB01] Plattform und der Client kann eine SAP oder Nicht-SAP Anwendung sein. Es gibt zwei Möglichkeiten für die Kommunikation zwischen dem Client und dem Server: direkt (Punkt-zu-Punkt) oder indirekt mit Hilfe der SAP Exchange Infrastrukture (SAP XI) [SB03]. SAP XI ist in die SAP NetWeaver Plattform integriert und wird zur Datenübertragung zwischen mehreren Systemen eingesetzt. Eine zentrale Aufgabe ist die Weiterleitung der eingehenden Nachrichten an das richtige System.

Da die Kommunikation über SAP XI keine Rolle für diese Arbeit spielt, wird darauf auch nicht näher eingegangen. Hier wird nur die direkte Kommunikation betrachtet.

Im Rahmen der direkten Kommunikation (Punkt-zu-Punkt) wird, um den Nachrichtaustausch zu ermöglichen, ein entsprechender Client-Proxy [SB02] im Anwendungssystem, in diesem Fall auch Servicenutzer genannt, generiert. Er enthält Informationen über die Aufrufmöglichkeiten des Enterprise Services. Durch diese Informationen wird später herausgefunden, wo und wie der benötigte Service aufgerufen werden kann.

2.4 Zusammenfassung

In diesem Kapitel wurden alle notwendigen Grundlagen und Begriffe vorgestellt, die für den weiteren Verlauf dieser Arbeit von Bedeutung sind:

- Anwendungsintegration
- EAI als neuer Ansatz zur Anwendungsintegration
- Datenintegration und die EAI-Lösung als spezifische Datenintegration
- Service-orientierte Architektur

[2] RosettaNet ist ein globales Konsortium, das sich damit beschäftigt bestehende offene EBusiness Standards, Richtlinien und Spezifikationen für plattformübergreifende Anwendungen und Netzwerkkommunikationen zu optimieren.

[3] UN/EDIFACT (*United Nations Electronic Data Interchange For Administration, Commerce and Transport*.) ist ein internationaler Standard für elektronischen Austausch von Handelsdokumenten und Geschäftsnachrichten.

Im folgenden Kapitel werden die vorgestellten Begriffe aus dem Bereich Datenintegration verwendet, um die zurzeit auf dem Markt existierenden Ansätze vorzustellen.

Kapitel 3

Aktuelle Ansätze des Marktes

Ein integriertes System kann sehr kompliziert sein, wenn es mit vielen heterogenen Datenquellen verbunden ist. Ein guter Entwurf für die grundlegende Architektur eines solchen Systems spielt dann eine wichtige Rolle. Die Architektur des Systems beschreibt die verschiedenen Komponenten, die für das System modelliert werden. In diesem Kapitel werden zwei verschiedene Architekturansätze zur Datenintegration vorgestellt.

3.1 Architekturen von Datenintegrationssystemen

Datenintegrationssysteme können in zwei wichtige Kategorien eingeteilt werden: *materialisierte Integration* und *virtuelle Integration* [LN06]. Der Unterschied besteht darin, dass bei der materialisierten Integration alle integrierten Daten an einer zentralen Stelle gespeichert und auf diese nur in der Anfragebearbeitung zugegriffen wird, während bei der virtuellen Integration alle Daten in der ursprünglichen Quelle bleiben und von dort anhand einer Anfrage zuerst ins zentrale System übertragen und dort weiter bearbeitet werden. Nach der Bearbeitung werden die übertragenen Daten wieder verworfen.

Im Folgenden werden diese Architekturen genauer vorgestellt.

3.2 Materialisierte Integration

Die materialisierte Integration ist eine physische Integration der Daten. Die Daten werden aus verschiedenen Quellen importiert, bereinigt und zentral abgelegt. Im Quellsystem werden dabei in der Regel keine Änderungen an den Daten vorgenommen.

Eine physische Materialisierung von Daten an einer zentralen Stelle hat folgende Vorteile [LN06]:

- Geschwindigkeitsvorteil bei der Anfragebearbeitung

  Bei der materialisierten Integration werden Daten direkt in der zentralen Stelle bearbeitet. Quellsysteme sind während der Anfrage nicht involviert. Dies bringt eine schnelle Bearbeitungszeit mit sich, da keine zusätzliche Kommunikation erforderlich ist.

- Verfügbarkeit von Datenquellen

  Im Fall des Ausfalls der Datenquellen wird das integrierte System davon nicht beeinflusst, da alle benötigen Daten schon im zentralen System liegen.

Eine typische Umsetzung der materialisierten Integration ist das Data Warehouse (DWH). Das Grundprinzip eines DWH ist die (synchrone oder asynchrone) Replikation von Daten aus heterogenen Quellsystemen in einer zentralen Datenbank. [LN06]

3.2.1 Data Warehouse

Architektonisch besteht ein Data Warehouse aus mehreren Komponenten (siehe Abbildung 12), die zur Aktualisierung, Überwachung und Benutzung des DWH verwendet werden.

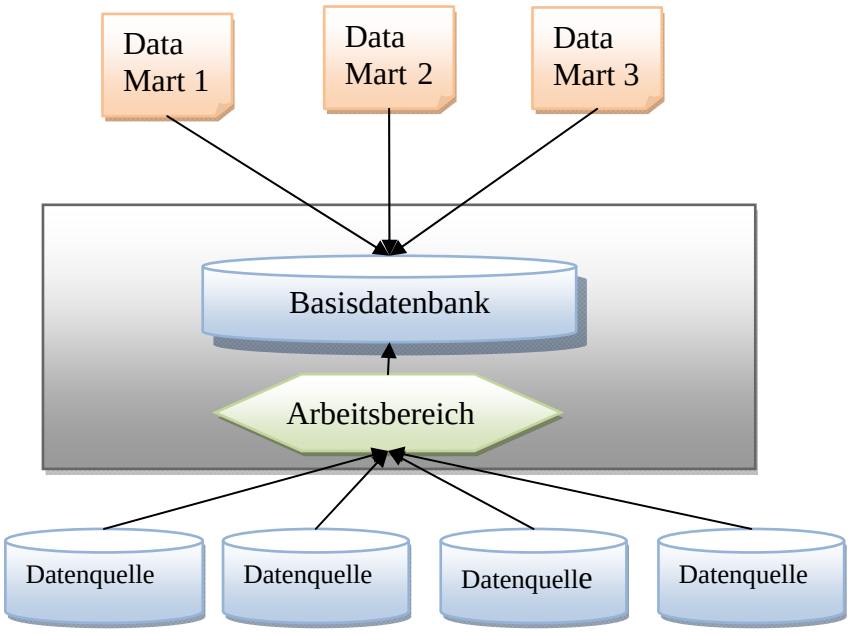

Abbildung 12: Architektur des Data Warehouse

Die in der Abbildung 12 dargestellte Architektur des DWH besteht aus den Datenquellen und einem DWH-Managersystem. Der Begriff des *Data Mart* bezeichnet die Datenbestände, die aus dem DWH abgeleitet werden. Die *Basisdatenbank* ist der physische Speicher der Daten. Er speichert alle zu integrierenden Daten und ist von zentraler Bedeutung im DWH. Zwischen den Quellsystemen und der Basisdatenbank gibt es die Komponente *Arbeitsbereich*, in der die

von verschiedenen Datenquellen kommenden Daten verarbeitet werden. Der Vorgang der Datenintegration wird im DWH System durch einen sogenannten ETL-Prozess [BG08] realisiert. **ETL** steht für **E**xtraktion, **T**ransformation und **L**aden. Alle nötigen Daten werden zuerst aus verschiedenen Quellsystemen extrahiert. Wegen ihres vielfältigen Ursprungs sind die Daten voneinander unterschiedlich in Bezug auf Format und Struktur. Sie müssen dann in das Format und die Struktur (Basisschema), die im DWH festgelegt sind, transformiert werden. Nach der Verarbeitung der Rohdaten im Arbeitsbereich können die Daten dann in dieses gemeinsame Basisschema des DWH geladen werden.

Um die Aktualität der Daten zu gewährleisten, finden periodische Update-Operationen, z.B. täglich zu bestimmten Zeitpunkten, statt. Bei Anfragen an das Anwendungssystem erfolgt die Datenbereitstellung *asynchron*. Hierauf wird im Abschnitt 5.2.2 näher eingegangen.

Data Warehouses sind heute die wichtigsten auf Datenintegration basierenden Systeme in Unternehmen. Da die Bearbeitung der Daten direkt auf der zentralen Datenbank stattfindet und Quellsysteme zur Anfragezeit nicht mehr involviert sind, bringt dieser Ansatz einen erheblichen Geschwindigkeitsvorteil. Außerdem ermöglicht er es, Daten im Data Warehouse getrennt von den Quellsystemen zu manipulieren.

Der Nachteil von DWH besteht darin, dass bei diesem Integrationsansatz nicht immer die aktuellsten Daten zur Verfügung stehen, da das Kopieren der Datenbestände nur in bestimmten Intervallen erfolgt. Es können auch Konsistenzprobleme auftreten, wenn Daten im Data Warehouse geändert wurden. [LN06]

3.3 Virtuelle Integration

Im Gegensatz zur materialisierten Integration werden bei der virtuellen Integration keine Daten zentral gespeichert. Alle zur Integration notwendigen Daten bleiben hier in der Datenquelle. Anders ausgedrückt, existiert bei der virtuellen Integration der integrierte Datenbestand nur virtuell. Aus Sicht des Nutzers gibt es zwar einen homogenen Bestand, die Daten werden aber erst bei einer Anfrage von den Datenquellen übertragen - die Bereitstellung der Daten ist *synchron*.

Im Folgenden wird eine wichtige Architektur, die zurzeit oft zur virtuellen Integration verwendet wird, vorgestellt.

3.3.1 Mediatorbasierte Informationssysteme

Ein mediatorbasiertes Informationssystem beinhaltet zwei wichtige Rollen: **Mediator** und **Wrapper**. Sie dienen als Vermittler zwischen Datenquellen und Anwendungen. Zu jeder autonomen, heterogenen Quelle wird ein spezieller Wrapper eingesetzt. Die Wrapper sind zuständig für den Zugriff auf die einzelnen Datenquellen. Ein Mediator kann auf einen oder

mehrere Wrapper zugreifen und führt bestimmte Aufgaben aus[FISC05]. In der Regel ist das die strukturelle und semantische Integration von Daten. Ein einfaches Mediator-Wrapper-System ist in der Abbildung 13 dargestellt.

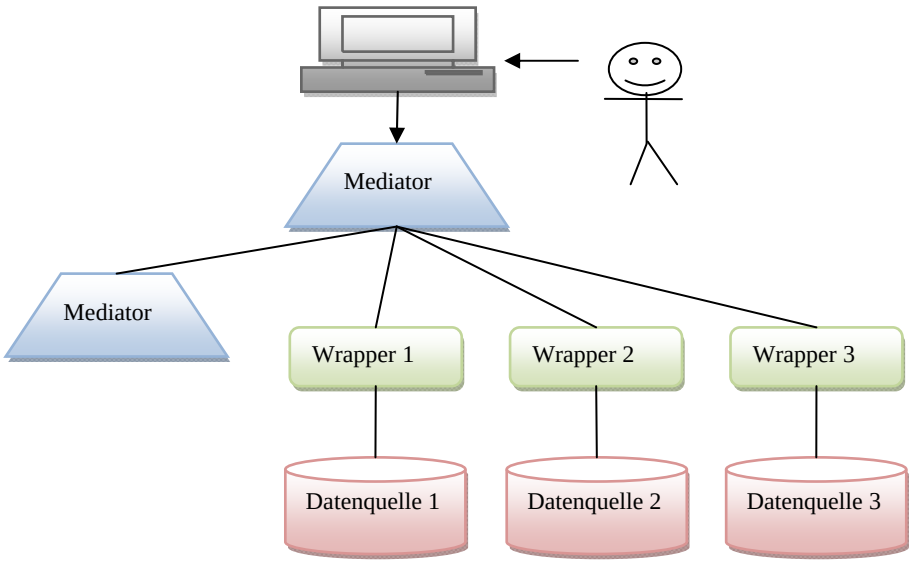

Abbildung 13: Mediator-Wrapper-System

Die genauen Definitionen von Mediator und Wrapper sind wie folgt:

Ein *Mediator* ist eine Softwarekomponente, die Wissen über bestimmte Daten nutzt, um Informationen für höherwertige Anwendungen zu erzeugen[LN06].
Die wichtigsten Aufgaben von Mediatoren sind:

- Suche und Auswahl relevanter Datenquellen
- Datentransformationen
- Konsistenzerhaltung
- Bereitstellung von Metadaten zur Weiterverarbeitung
- Integration von Daten mittels gemeinsamer Schlüssel

Wrapper sind Softwarekomponenten, die die Kommunikation und den Datenfluss zwischen Mediatoren und Datenquellen herstellen [LN06].

Wrapper sind jeweils spezialisiert auf eine Ausprägung autonomer, heterogener Quellen. Ihre wichtigsten Aufgaben umfassen:

- Überwindung von Schnittstellenheterogenität, also z.B. von SQL zu HTML-Formularen
- Herstellung von Datenmodelltransparenz durch Überführung der Daten in das kanonische Datenmodell

- Überwindung der schematischen Heterogenität durch eine geeignete Abbildung zwischen Quellschema und globalem Schema

Der Ablauf bei diesem Ansatz ist wie folgt: Der Mediator nimmt eine Anfrage von der Anwendung entgegen und beantwortet diese, indem er mit den benötigten Datenquellen kommuniziert. Dieser Ansatz stellt bereits eine konkrete Softwareausprägung von Middleware dar.

Bei diesem Ansatz bleiben alle Daten in den Quellen, wodurch es keine redundante Datenhaltung wie bei dem Data Warehouse gibt. Die Informationsquellen sind in diesem Fall autonom und wissen oft nichts von ihrer Integration. Ein anderer Vorteil besteht darin, dass bei diesem Ansatz die Aktualität der zu integrierenden Daten gewährleistet werden kann.

Ein wesentlicher Nachteil dieses Ansatzes sind mögliche Geschwindigkeitsprobleme. Da bei der virtuellen Integration alle Daten erst zum Zeitpunkt der Datenanfrage von den Quellsystemen geholt werden, kann es wegen einzelner langsamer Datenquellen zu einer insgesamt längeren Antwortzeit kommen.

3.4 Vergleich der beiden Ansätze

Anhand mehrerer Kriterien können die beiden Ansätze zur Datenintegration verglichen werden. Die wichtigsten Kriterien sind [LN06]:

- Aktualität

  Die Daten sind bei der virtuellen Integration immer aktuell, da sie für jede Anfrage direkt von den Datenquellen geholt werden. Bei der materialisierten Integration hängt die Aktualität der Daten von der Aktualisierungsfrequenz des Systems ab.

- Antwortzeit

  Bei der virtuellen Integration werden Daten erst zum Zeitpunkt der Anfrage von den Quellsystemen geholt. Dies bringt in der Regel einen Geschwindigkeitsnachteil mit sich; das integrierte System kann niemals schneller sein als die langsamste Datenquelle, die an einer konkreten Anfrage beteiligt ist. Bei der materialisierten Integration werden Daten direkt von der zentralen Datenbank geholt.

- Änderungen an den Daten

  Da bei der virtuellen Integration die Daten physisch nur in den Quellsystemen vorliegen, können meist keine Änderungen vorgenommen werden, die die Integration erleichtern könnten. Bei der materialisierten Integration hingegen können Änderungen an den Daten vorgenommen werden, ohne dass eine Quelle dies unterstützen muss. Dafür müssen

Vorkehrungen getroffen werden, die verhindern, dass Änderungen im integrierten System bei Updates aus den Quellen einfach überschrieben werden.

- Speicherbedarf

  Aufgrund der zentralen Speicherung sämtlicher Datenbestände ist der Speicherbedarf materialisiert integrierter Systeme groß. Bei der virtuellen Integration wird nur sehr wenig Speicher für die Metadaten und gegebenenfalls temporäre Anfrageergebnisse benötigt.

- Belastung der Quellen

  Bei der materialisierten Integration entsteht durch die Bereitstellung von Updates regelmäßig eine sehr hohe Last auf den Quellen, die jedoch planbar ist. Bei der virtuellen Integration entsteht dagegen unregelmäßig eine eher geringere Last, die sich allerdings in Zeiten intensiver Nutzung – meist unvorhersehbar – verstärken kann.

- Datenbereinigung

  Daten aus autonomen Datenquellen sind oft nicht von der benötigten Qualität; sie enthalten Datenfehler und Duplikate. Die Verfahren zur Datenbereinigung sind in der Regel so aufwändig, dass sie nicht im Zuge der Anfragebearbeitung, also innerhalb von Sekunden, durchgeführt werden können. Somit kommt Datenbereinigung nur für materialisiert integrierte Systeme in Frage.

Es gibt noch einige andere Kriterien, die zur Bewertung der beiden Ansätze dienen können, wie z.B. Komplexität der Anfrage und Anfragemöglichkeiten. Diese Kriterien sind für die vorliegende Arbeit allerdings nicht interessant, da diese hauptsächlich für andere Datenintegrationskonzepte verwendet werden. Weil die hier diskutierte Datenintegration ein Teil der EAI-Lösung und damit ein nachrichtenbasierter Ansatz ist, werden an dieser Stelle nur die dafür relevanten Bewertungskriterien behandelt.

3.5 Zusammenfassung

Es wurden mit der materialisierten und virtuellen Integration zwei Architekturen für Datenintegrationssysteme mit ihren Vor- und Nachteilen vorgestellt. Eine optimale Auswahl basiert dabei auf dem vollständigen Wissen über deren praktische Anwendungsgebiete. Wenn beispielsweise der zu integrierende Datenbestand einfach zu groß ist oder die Struktur der Daten in alle Quellsystemen zu stark voneinander abweicht, ist es nicht zu empfehlen, eine materialisierte Integration zu wählen. Wenn bei integrierten Systemen eine hohe Geschwindigkeit bei der Datenbearbeitung zwingend notwendig ist, dann ist wiederum der virtuelle Ansatz nicht ideal.

Im Hinblick auf die Datenintegration sind außer dem Entwurf der passenden Architektur auch die Infrastrukturen, die für die Realisierung dieser Architektur notwendig sind, wichtig. Möglichen Infrastrukturansätzen widmet sich das folgende Kapitel.

Kapitel 4

Vorstellung der service-basierten Datenintegration

Effiziente Datenintegration kann nur dann gelingen, wenn man mit den zugrundeliegenden Daten optimal arbeiten kann. Heutzutage setzen viele Unternehmen zunehmend auf moderne Datenintegrationsplattformen, die den Austausch von Informationen zwischen unterschiedlichen Systemen automatisieren und vordefinierte, wiederverwendbare an die Stelle aufwändiger eigener Programmierung setzen. Für die Realisierung der Datenintegration sind mächtige Integrationswerkzeuge erforderlich, damit auf die sich kontinuierlich ändernden Anforderungen flexibel reagiert werden kann. Die modernen Integrationstechniken wie Services bieten hier die Möglichkeit und Performance, um den Integrationsvorgang effizient abwickeln zu können.

Im Vergleich mit anderen Integrationstechnologien, wie z.B. RMI, RPC oder CORBA bietet der Service eine Integrationsmöglichkeit, die vollständig unabhängig von der Plattform und Programmiersprache der zu integrierenden Anwendungssysteme ist. Deswegen hat zurzeit die service-basierte Integration immer mehr an Bedeutung gewonnen. Im Zentrum der vorliegenden Arbeit stehen die detaillierte Analyse und die praktische Implementierung einer service-basierten Lösung für die Datenintegration.

Wie gesagt, ist SOA nur ein Architekturkonzept. Es gibt, wie schon im Abschnitt 2.3.2 erwähnt, verschiedene konkrete Implementierungen. Um die Idee der service-basierten Datenintegration besser erklären zu können, wird in diesem Kapitel der Web Service als ein konkretes Beispiel der SOA verwendet. Eine auf Web Services basierende Datenintegration wird vorgestellt. Zum Schluss werden deren Vor- und Nachteile analysiert.

4.1 Vorstellung einer auf Web Services basierenden Datenintegration

Der Gegenstand der Datenintegration sind vielfältige verteilte Datenbestände. Bei den involvierten Datenquellen kann es sich um Standardapplikationen von SAP, Oracle oder Peoplesoft handeln, um verschiedene relationale Datenbanken, Dateien aller Art, um die Daten, die über Standards wie IBM MQSeries, TIBCO Rendezvous, webMethods, ODBC oder XML, angesprochen werden, oder um hierarchische und multidimensionale Plattformen,

wie verschiedene Mainframe-Systeme, C-ISAM oder Adabas. Neben solchen standardisierten Datenformaten existieren aber auch unstrukturierte Daten wie E-Mails, Word-Dokumente, Präsentationen, Excel-Tabellen oder PDF-Dateien. Schätzungen zufolge machen diese unstrukturierten Daten bis zu 90 Prozent der in einem Unternehmen gespeicherten Informationen aus.

Die Zusammenführung von Daten aus heterogenen Datenquellen und ihre Auslieferung an den richtigen Ort zur richtigen Zeit sind mittlerweile zu einem erfolgsentscheidenden Faktor geworden. XML und Web Services spielen in diesem umfangreichen Markt eine wichtige Rolle.

Wie bereits in Kapitel 2 beschrieben, bietet ein Web Service eine XML-basierte Kommunikation zwischen Anwendungssystemen. Die Extensible Markup Language (XML) als universelles Datenaustauschformat ist die Grundlage für die Schaffung von systemunabhängigen Schnittstellen. Ein XML-Dokument kann in seiner Struktur frei definiert werden und Daten innerhalb dieser Struktur speichern bzw. transportieren. Genauer gesagt, werden die zu übertragenden Daten in Form eines XML-Dokuments gespeichert und über SOAP transportiert. Durch die Nutzung von XML als Datenaustauschformat stehen mächtige Datenbeschreibungssmöglichkeiten für lokale Datensysteme zur Verfügung. Es reduziert auch die Komplexität der Middleware-komponenten, denn in einer homogenen Umgebung muss keine Konvertierung zwischen unterschiedlichen, plattformspezifischen Binärformaten mehr durchgeführt werden.

Beim Einsatz von Web Services für die Datenintegration werden die in bestimmten Geschäftsprozessen erzeugten Daten in einen entsprechenden Service verpackt. Wenn die Daten benötigen werden, erfolgt direkt durch einen Service-Aufruf ihr Transport zwischen den verschiedenen Systemen.

4.2 Vor- und Nachteile der service-basierten Datenintegration

Der Vorteil der service-basierten Integration liegt darin, dass neben reinen Datenbanksystemen auch beliebige andere Datenspeichersysteme angebunden werden können. Diese Funktionalität wird z.B. über lokale Java-Konnektoren realisiert. Die Java-Konnektoren-Architektur bietet eine einheitliche Schnittstelle zu allgemeinen Datenspeichersystemen, ähnlich wie ODBC (Open Database Connectivity) und JDBC (Java Database Connectivity) bei relationalen Datenbanksystemen. Diese angeführten Eigenschaften decken sich besonders gut mit den Anforderungen, die an eine datenorientierte Integrationslösung mit verschiedenen, verteilten Systemen gestellt werden.

Ein weiterer Vorteil ist die konsequente Nutzung von Standards. Im Vergleich zu anderen Integrationslösungen werden hier keine selbst entwickelten und damit proprietären Technologien für den Zugriff auf integrierten Daten eingesetzt.

Besonders interessant sind jedoch die konzeptionellen Vorteile, die sich durch XML-Web Services ergeben: Durch die Nutzung von Technologien des World-Wide-Web in dem Bereich Middleware lassen sich weltweit verteilte Anwendungen auf einfache Art und Weise zusammenstellen.

Außer den oben genannten Vorteilen wurden noch folgende Aspekte betrachtet:

- **Wiederverwendbarkeit:**

  Alle Daten, die in einem Geschäftsprozess entstehen, können in Services verpackt werden. Services sind wiederverwendbar, wodurch auch die Möglichkeit der Mehrfachnutzung von vorhandenen Daten besteht.

- **Aktualität:**

  Daten, die über einen Service verfügbar sind, können jederzeit durch einen Service-Aufruf abgerufen werden. Dies gewährleistet die Aktualität der integrierten Daten.

Durch die neuen Möglichkeiten und den damit verbundenen Technologien ergeben sich aber auch eine Reihe an Nachteilen.

Im Vergleich zu älteren Middleware-Technologien wie CORBA und Java-RMI sind bei diesem Integrationsansatz insbesondere der deutlich höhere Bedarf an Rechenleistung sowie eine deutlich größere Netzwerkbelastung nachteilig. Beides sind Folgen des durchgängigen Einsatzes von XML, denn im Vergleich zu den binären Datenformaten älterer Middleware-Technologien ist hier der Aufwand beim Parsen und Übertragen des textbasierten XML-Formats deutlich größer.

Ein anderer Nachteil besteht in der Konfrontation mit neuen sicherheitsspezifischen Herausforderungen. Die Sicherheitsanforderungen an SOA-Infrastrukturen sind häufig hoch. Diese betrifft neben der Problematik verteilter Strukturen, realisiert durch offene Formate, auch Administrations- und Beherrschbarkeitsaspekte. Die Realisierung der Datenintegration über eine Vielzahl lose gekoppelter Services erfordert Authentisierung und die Sicherstellung der Vertraulichkeit.

Kapitel 5

Entwicklung eines Konzeptes zur service-basierten Datenintegration

Das Ziel der vorliegenden Arbeit ist die Entwicklung eines EAI-Konzeptes, das sowohl eine neue, datenorientierte Integrationslösung für heterogene, verteilte Systeme darstellt, als auch eine einfache, schnelle Integration ermöglicht. Die Kommunikation zwischen verschiedenen Systemen soll auf einer service-basierten Architektur aufbauen.

In diesem Kapitel werden zuerst wichtige Anforderungen, die ein EAI-Konzept, die auf dem Beispiel des Unternehmensszenarios beruht, erfüllen sollte, durch eine Vorüberlegung eingeführt. Es wird dann gezeigt, dass die entwickelte service-basierte Architektur mit den heutigen vorhandenen Standards die Kernanforderungen an eine moderne EAI-Plattform erfüllt.

5.1 Vorüberlegung

Um den geeignetsten Integrationsansatz bewerten zu können, muss bei der Vorbereitung für die Entwicklung eines EAI-Konzeptes in eine bestehenden IT-Infrastruktur die vorhandenen Geschäftsprozesse und die involvierten Daten auf den Integrationsbedarf hin analysiert werden. Wie Kapitel 2.2.4.2 schon deutlich gemacht hat, wird hier anhand praktischer, existierender Probleme eine datenorientierte Integration realisiert. Dabei werden zwei oder mehrere Anwendungen integriert, indem direkt auf die von ihnen erzeugten, verwalteten und gespeicherten Daten zugegriffen wird.

Im Folgenden werden anhand des Unternehmensszenarios, das am Anfang des zweiten Kapitels beschrieben wurde, die Anforderungen an das zu entwickelnde Integrationskonzept, die insbesondere für eine service-basierte Datenintegration benötigt werden, im Einzelnen vorgestellt.

5.1.1 Anforderungen an die Datenintegration

Den Aufgaben der Datenintegration liegt folgendes Problem zugrunde: Wie sind alle benötigen Daten auffindbar? Üblicherweise befinden sich die gewünschten Daten verteilt in den unterschiedlichsten Systemen. Ein Beispiel dafür ist das Unternehmen DIwS AG, das mehrere Standorte besitzt. Die Daten, die beispielsweise in der zentralen Stelle in Stuttgart benötigt werden, können aus einem beliebigen dezentralen Anwendungssystem kommen. Sie können in einer normalen Datei und/oder in einem Datenbanksystem abgelegt sein. Zusätzlich besteht die Möglichkeit, dass sie sich unter der Kontrolle von unterschiedlichen Betriebssystemen befinden. Das Ziel der Datenintegration ist es, die Daten, die von heterogenen Systemumgebungen kommen, zu vereinen, damit die entsprechende Anwendung auf diese einheitliche Datenbasis ohne Probleme zugreifen kann. Der Integrationsvorgang sollte somit vor dem Benutzer der Anwendung verborgen bleiben.

Wie im zweiten Kapitel schon kurz beschrieben wurde, kann dabei eine syntaktische oder logische Heterogenität vorliegen. Bei der syntaktischen Heterogenität bestehen technische Unterschiede zwischen den Datenbanksystemen, den Betriebssystemen oder den Zugriffsmethoden. Eine logische Heterogenität kann durch semantische, schematische oder strukturelle Unterschiede hervorgerufen werden. Semantische Unterschiede werden durch Synonyme und Homonyme bei der Bezeichnung von Daten verursacht. Schematische und strukturelle Differenzen entstehen bei gleicher Modellierungssprache des Datenmodells durch unterschiedliche Modellierungen. Das Finden eines einheitlichen Schemas, um die verschiedenen heterogenen Daten vereinen zu können, ist ein wichtiges zu lösendes Problem bei der Integration von Daten [LN06].

Außer dem oben genannten Integrationsproblem von Daten, müssen die zu integrierenden Daten selbst auch einige Anforderungen erfüllen. Die Datenintegration macht nur dann Sinn, wenn folgende Fragen geklärt sind:

- Welche Daten existieren?
- Ist bekannt, welche Daten benötigt werden?
- Sind die Daten aktuell?
- Stimmt die Qualität der Daten?
- Liegen die Daten im gewünschten Format vor? Wenn nicht, können sie konvertiert werden?
- Lassen sich die Daten aus verschiedenen Systemen so transformieren, dass sie mit einander verglichen werden können?

Die oben genannten Fragen sind nicht sehr ausführlich, verdeutlichen aber schon wichtige Aspekte von Daten, über die man sich vor dem Entwurf des Konzeptes klar sein sollte.

Das Sammeln von Daten ist nicht kompliziert und mit Abstand der einfachste Vorgang. Aber es ist erforderlich, dass die gesammelten Daten auch laufend gepflegt werden (z.B.

Aktualisieren, Auslagern von zurzeit nicht mehr benötigen Daten oder Löschen von alten Daten), um sie später auch sinnvoll aufbereiten und nutzen zu können.

Ein anderer wichtiger Aspekt, der bei der Datenintegration berücksichtigt werden muss, ist die Art des Zugriffes auf die Datenbanksysteme. Bei dem Ansatz der virtuellen Datenintegration kann es passieren, dass die in der zentralen Stelle verarbeiteten Daten eventuell noch ins dezentrale System zurückgeliefert werden müssen. Es sollte deswegen nicht nur ein Lesezugriff erlaubt werden, sondern auch die Möglichkeit bestehen, Daten jederzeit in eine dezentrale Datenbank schreiben zu können. Es muss dabei sichergestellt sein, dass geschäftliche Abläufe niemals auf bereits überholten Informationen basieren. Es ist in vielen Fällen nicht bekannt, ob wichtige Daten bereits irgendwo vorhanden sind oder die Daten vorhanden, aber nicht in der gewünschten Form abgespeichert sind.

5.1.2 Anforderungen an die Softwarearchitektur

Bei einer Softwarearchitektur geht es in der Wirtschaftsinformatik um die Komponenten eines Softwaresystems und deren Interaktion. Die Entwicklung einer Integrationsarchitektur dient allgemein der Bestimmung eines unternehmensweiten Rahmenplanes, der die Beziehung aller beteiligten Systeme zueinander verdeutlicht und Regeln für die Benutzung von Daten definiert.

Eine gut entworfene Softwarearchitektur sollte die folgenden Punkte erfüllen:

- **Dienlichkeit**

  Ein genereller Maßstab für die Beurteilung einer zu entwickelnden Architektur ist deren Dienlichkeit für die Erreichung der allgemeinen Geschäftsziele des Unternehmens.

- **Langfristig gültig und flexibel**

  Zum Schutz der Investitionen in der Informationsverarbeitung sollte die Architektur langfristig gültig sein, gleichzeitig aber flexibel genug, um schnell auf sich verändernde Geschäftsanforderungen reagieren zu können.

Zudem gibt es noch einige Punkte, die man bei der Entwicklung einer Integrationsarchitektur beachten sollte, wie z.B. Performanz, Wartbarkeit oder deren Einfachheit.

Außerdem sollten bei der Definition einer EAI-Architektur im Sinne eines globalen Integrationskonzeptes auch Vorgaben für das Design der einzelnen Anwendungssysteme gegeben werden, die sich auf die benutzten Daten, Funktionen oder Kommunikations-mechanismen beziehen. In unserem Fall behandelt die zu entwickelnde Architektur hauptsächlich die Datenintegration in verteilten Systemen. Deswegen muss diese Architektur

sowohl für die zu integrierenden Daten geeignet sein, als auch die Regeln für die Kommunikation zwischen den Komponenten festlegen.

Von der Seite der zu integrierenden Daten aus muss diese Architektur weitere Aspekte beachten. So gibt es beispielsweise verschiedene Datenkategorien. Manche Daten ändern sich nicht so oft, andere hingegen regelmäßig. D.h. die zu entwickelnde Architektur sollte die beiden Varianten berücksichtigen, um sowohl Redundanz zu vermeiden als auch die Daten effizient integrieren zu können. Abschnitt 5.2.1 behandelt diesen Aspekt ausführlicher.

Die Seite der Kommunikationsmechanismen wird im Abschnitt 5.2.2 näher beleuchtet.

5.1.3 Anforderungen an die Kommunikationsinfrastruktur

Die zugrundeliegende Infrastruktur sollte dem Datenaustausch zwischen verschiedenen Systemen dienen. D.h. sie sollte die Übertragung der verteilten Daten und danach die Bearbeitung der gesammelten Daten ermöglichen. Wenn bei der Bearbeitung der Daten herausgefunden wurde, dass sie unvollständig oder fehlerhaft sind, sollte es auch möglich sein, eine Benachrichtigung zu senden, die den Quellsystemen dies mitteilt und eventuell eine neue Zustellung der Daten veranlasst.

Bei der Übertragung der Daten vom Sender zum Empfänger gibt es zwei mögliche Mechanismen, die sogenannte synchrone- und asynchrone Kommunikation Die beiden Begriffe werden im Abschnitt 5.2.2 näher behandelt.

Eine gute Kommunikationsinfrastruktur sollte beide Kommunikationsmechanismen unterstützen und dann je nach Dateneigenschaft und Systemanforderungen die Entscheidung des zu verwendenden Kommunikationsmechanismus, der vom konkreten Anwendungsfall abhängt, treffen.

5.1.4 Andere Anforderungen

Neben den oben aufgelisteten, spezifischen Anforderungen lassen sich noch einige zusätzliche Merkmale betrachten, die sinnvoll zur Beschreibung von Integrationskonzepten herangezogen werden können. Hierzu zählen sowohl die Unterstützung für Sicherheit, die Skalierbarkeit als auch die Leistungsfähigkeit der Lösung.

- Unterstützung für Sicherheit

  Ein sicheres Integrationskonzept muss die festgelegten Sicherheitsmechanismen durchsetzen. Entsprechend ist zu bewerten, inwieweit sich die Rechte der unterschiedlichen Benutzer, Systeme und Dienste, die auf die Integrationsinfrastruktur

zugreifen, verwalten und umsetzen lassen. Daneben sollten ein Aktivitäts-Logging sowie die sichere Datenkommunikation gewährleistet werden können [KAIB02].

- Leistungsfähigkeit und Skalierbarkeit der Laufzeitumgebung

  Die Leistungsfähigkeit umfasst die Zuverlässigkeit und die garantierte Übermittlung von Daten zwischen Systemen. Zusätzlich können Parameter wie die Latenz, der Durchsatz oder die Effizienz der Datenübertragung in die Bewertung mit einfließen. Bei der Übertragung von Daten zwischen verschiedenen Systemen gibt es dann eine typische Schwachstelle im Hinblick auf die gesamte Verarbeitungszeit: die Datentransformation. Die darauf verwendete Zeit ist abhängig von der Menge der zu verarbeitenden Daten sowie von der Komplexität der Transformation und der dabei anzuwendenden Regeln. Aufgrund der mit einer zentralen Integrationsinstanz verbundenen Gefahr, zum Engpass für den Datenaustausch zwischen allen integrierten Anwendungen zu werden, müssen diese Anforderungen bei der Entwickelung eines Konzeptes besonders beachtet werden [KAIB02].

Im Abschnitt 5.2 wird anhand der oben aufgelisteten Anforderungen an verschiedene Aspekte ein EAI-Konzept, das der Datenintegration in verteilten Systemen dient, entwickelt und beschrieben.

5.2 EAI-Konzept zur Datenintegration

Im Kapitel 3 wurden einige Architekturansätze zur Enterprise Application Integration bzw. Datenintegration vorgestellt. Nun muss eine Auswahl aus den verschiedenen Ansätzen getroffen werden, mit denen die im Abschnitt 5.1 vorgestellten Anforderungen zusammen erfüllt werden können. Dabei muss man jeweils noch beachten, für welchen Zweck bzw. welche Situation die Ansätze ursprünglich entwickelt wurden und wie sie im vorliegenden Fall angepasst werden müssen.

Es wird in diesem Abschnitt ein Konzept zur service-basierten Datenintegration entworfen. Eine datenorientierte Anwendungsintegration hat Vorteile in bestimmten Anwendungsgebieten, insbesondere in dem in Kapitel 2 beschriebenen Unternehmensszenario. Die Gründe für die Auswahl des Services als Infrastruktur wurde schon in Kapitel 4 erklärt. Eine dementsprechend entworfene Systemarchitektur wird im Abschnitt 5.2.4 vorgestellt.

Im Folgenden wird zuerst anhand von vier verschiedenen Gesichtspunkten die Umsetzung des zu entwickelnden EAI-Konzeptes betrachtet.

5.2.1 Integrationsmodell

Das in diesem Konzept verwendete Integrationsmodell basiert, im Unterschied zu den anderen möglichen Integrationsmodellen, auf der Datenebene. D.h. es werden mehrere Anwendungen aus verschiedenen Systemen integriert, indem direkt auf die von ihnen erzeugten, verwalteten und gespeicherten Daten(-banken) zugegriffen wird. Bei diesem Ansatz werden die benötigten Daten aus verschiedenen Datenbeständen, die in verteilten Anwendungssystemen liegen, ausgelesen, dann gegebenenfalls in passender Art und Weise modifiziert und schließlich in einem zentralen Zielsystem weiter bearbeitet.

Es wird in dem hier entwickelten Konzept ein Hybridansatz vorgestellt, der auf dem Data Warehouse und Mediator-basierten Integrationsansatz aufbaut. Das Data Warehouse wird hier verwendet, um die langfristig gültigen Daten zu speichern, während der Mediator-Ansatz für die Daten verwendet wird, die sich sehr oft ändern.

Eine Darstellung für diese Art der Integration zeigt das folgende Bild.

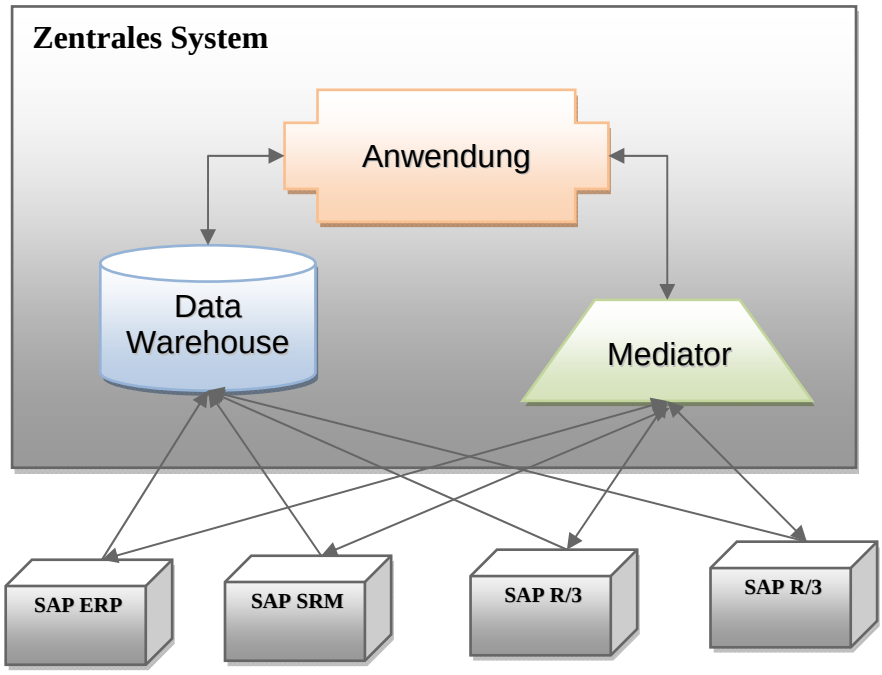

Abbildung 14: Hybridansatz für die Datenintegration

Wie in Abbildung 14 dargestellt, können die Daten, die in den externen/dezentralen Systemen langfristig gültig bleiben, im DWH des zentralen Systems redundant gespeichert werden. Es wird dafür nur ein regelmäßiger Update-Mechanismus benötigt, damit die eventuell aufgetretenen Datenänderungen auch ins zentrale System übernommen werden. Der Mediator wird verwendet, um die Daten, die in den externen/dezentralen Systemen sehr oft aktualisiert werden, auf Anfrage jederzeit ins zentrale System übertragen zu können.

In der Praxis besitzen die Daten aus den verschiedenen Systemen normalerweise jeweils eine sehr unterschiedliche Struktur. Ein Problem liegt darin, die Daten, die zwar eine andere

Struktur, aber gleiche Semantik haben, in ein einheitliches Schema zu bringen, damit sie im zentralen System einfach verwendet werden können. Um dieses Problem zu lösen, stehen mehrere Möglichkeiten zur Verfügung. Die in diesem Konzept verwendete Lösung wird im Abschnitt 5.2.3 näher beschrieben.

5.2.2 Kommunikationsmodell

Das Kommunikationsmodell liefert ein Konzept für den konkreten Nachrichtenaustausch zwischen den Applikationen. Durch das Kommunikationsmodell wird festgelegt, welche Art der Kommunikation zwischen zwei verteilten Anwendungen verwendet werden soll. Es gibt zwei grundsätzliche Ansätze dazu: *synchrone* und *asynchrone Kommunikation*.

Wenn der Sender auf die Antwort des Empfängers warten muss, bevor er mit seiner Verarbeitung fortfahren kann, dann ist dies eine synchrone Kommunikation. Abbildung 15 verdeutlicht diesen Zusammenhang.

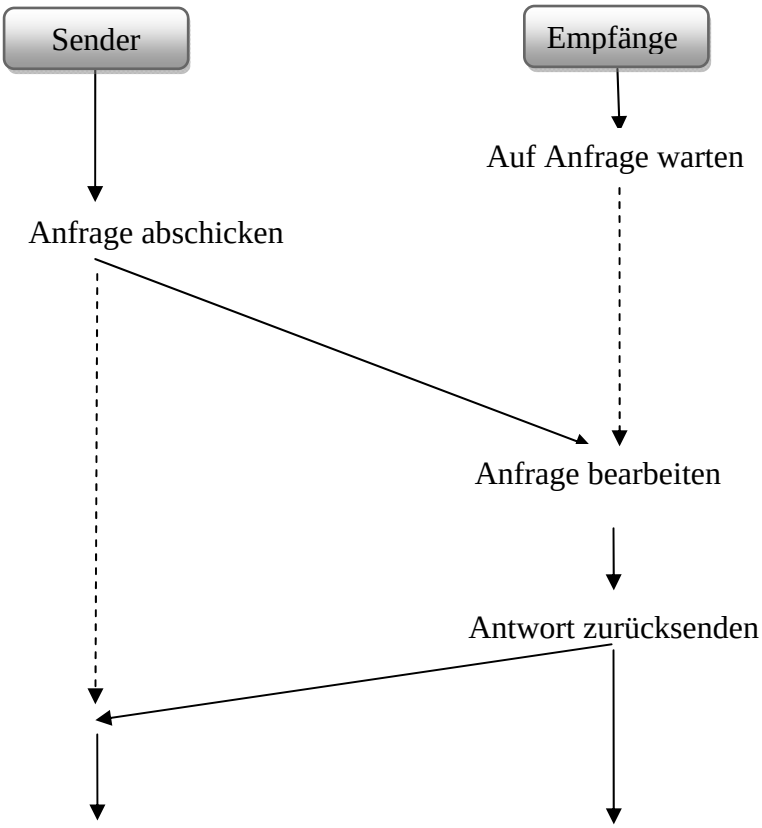

Abbildung 15: Synchrone Kommunikation

Hingegen wird bei der asynchronen Kommunikation der Sender direkt mit seiner Verarbeitung fortfahren, nachdem die Anfrage abgesetzt wurde. Da der Sender in diesem Fall nicht notwendigerweise eine Antwort auf seine Übertragung erwartet, kann der Empfänger die Anfrage entweder direkt verarbeiten oder aber in einem Puffer ablegen und bei Bedarf

verarbeiten. Dieser Kommunikationsmechanismus wird in der folgenden Abbildung verdeutlicht.

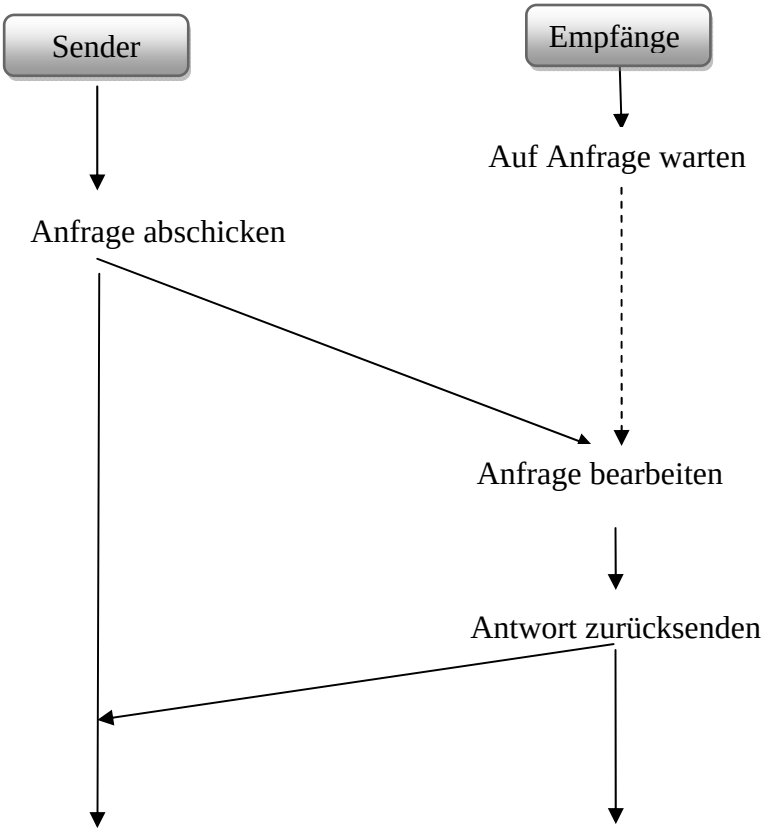

Abbildung 16: Asynchrone Kommunikation

In dem zu entwickelnden Integrationskonzept kommen die zu integrierenden Daten aus verschiedenen externen Systemen. Es können je nach Anforderung sowohl die synchrone als auch die asynchrone Kommunikation verwendet werden. Die Wahl der Art der Kommunikation wird dann je nach Situation getroffen. Im Folgenden werden diese beide Ansätze und deren Verwendungsszenarios beschrieben.

- Synchrone Kommunikation

  Synchrone Kommunikation eignet sich vor allem für das Kommunikationsszenario, in denen Sender und Empfänger ihre Handlungen koordinieren müssen. Das bedeutet, dass der Sender auf das Resultat des Empfängers warten muss, bevor er mit der Verarbeitung fortfahren kann.

  Bei synchroner Kommunikation sendet ein Anwendungssystem eine Anfrage an ein zweites Anwendungssystem und blockiert die weitere Verarbeitung solange, bis dieses eine Antwort gesendet hat. Diese Antwort kann als Ergebnis die im zweiten System verarbeiteten Daten enthalten, die das erste System durch die Anfrage angefordert hat. Alternativ kann es sich bei der Antwort auch nur um eine Bestätigung für den Eingang der Anfrage handeln. Sobald der Sender die Antwort erhalten hat, nimmt er seine Verarbeitung

wieder auf. Dieser Ansatz wird eingesetzt, wenn die Antwort Informationen beinhaltet, die für die Weiterverarbeitung auf Senderseite essentiell wichtig sind.

Synchrone Kommunikation eignet sich vor allem für die Abbildung von Echtzeit-Prozessen. Sie findet in interaktiven Systemen statt, in denen Sender und Empfänger ihre Handlungen koordinieren müssen.

Der Nachteil von diesem Kommunikationsmechanismus besteht darin, dass wenn die Verarbeitung auf der Seite des Empfängers eine längere Zeitspanne dauert, kann der Performanzverlust für das Gesamtsystem erheblich und möglicherweise inakzeptabel sein.

- Asynchrone Kommunikation

  Im Gegensatz zur synchronen Kommunikation bedeutet die asynchrone Kommunikation keine so starke Koppelung zwischen Sender und Empfänger, die jeweiligen Verarbeitungsprozesse wirken hier nicht blockierend. Der Sender überträgt seine Anfrage und setzt die Verarbeitung seiner Prozesse fort. Es ist dann irrelevant, wann die Anfrage das Empfängersystem erreicht oder ob sie überhaupt dort eintrifft. Wie die übertragenen Daten verarbeitet werden oder wie der Empfänger eine eventuelle Antwort kommuniziert, ist auch nicht von Belang.

  Dieser Kommunikationsmechanismus wird dann eingesetzt, wenn Sender und Empfänger zeitlich unabhängig voneinander operieren.

  In dem entworfenen Konzept werden beide Kommunikationsarten zur Verfügung stehen. Es sollte jedoch konkret je nach Verwendungsfall festgelegt werden, welches Kommunikationsmodell eingesetzt wird.

5.2.3 Integrationsmechanismen

Der Integrationsmechanismus steht für die konkrete Technologie, mit dem die Kommunikation zwischen zwei Komponenten abgewickelt wird. Im hier entworfenen Konzept werden Services als Kommunikationsschnittstelle verwendet.

Im Kapitel 2 wurden bereits die Grundlagen der service-basierten Architektur vorgestellt. Darauf aufbauend wurde im Kapitel 4 weiter erklärt, wieso diese grundsätzlich zur Lösung von Integrationsaufgaben geeignet sind. Ein Service als Kommunikationsschnittstelle hat, zurzeit immer mehr Beachtung im Gebiet der Abwendungsintegration gefunden. Bei diesem Integrationsmechanismus kommuniziert der Sender mit dem Empfänger über die Service-Schnittstelle. Die zu übertragenden Daten werden je nach ihrer Geschäftsbedeutung in bestimmte Services gepackt und durch den entsprechenden Service-Aufruf transportiert. Durch die Nutzung von XML als Datenaustauschformat stehen mächtige Zugriffsmöglichkeiten auf lokale Datensysteme zur Verfügung.

Außerdem ist, wie im 5.1.3 schon erwähnt, um das Konzept flexibel für verschiedene Szenarios verwenden zu können, möglicherweise ein Integrationsmechanismus, der zwei verschiedene Kommunikationsarten unterstützt, vonnöten. Die Service-Architektur bietet diese Möglichkeit, da sie jeweils als synchron und asynchron definiert werden kann.

Bei dem synchronen Mechanismus wird ein synchroner Service so definiert, dass der Servicebenutzer zuerst eine Service-Anfrage zu dem Serviceanbieter schickt. Gleichzeitig wird der davon betroffene Prozessfluss gesperrt. Das Sendersystem wartet dann auf die Daten von dem Empfängersystem. Erst wenn das Sendersystem ein Antwort von diesem bekommen hat, kann der Prozessfluss wieder entsperrt werden.
Abbildung 17 beschreibt die Kommunikations-abläufe bei der synchronen Nutzung eines Services.

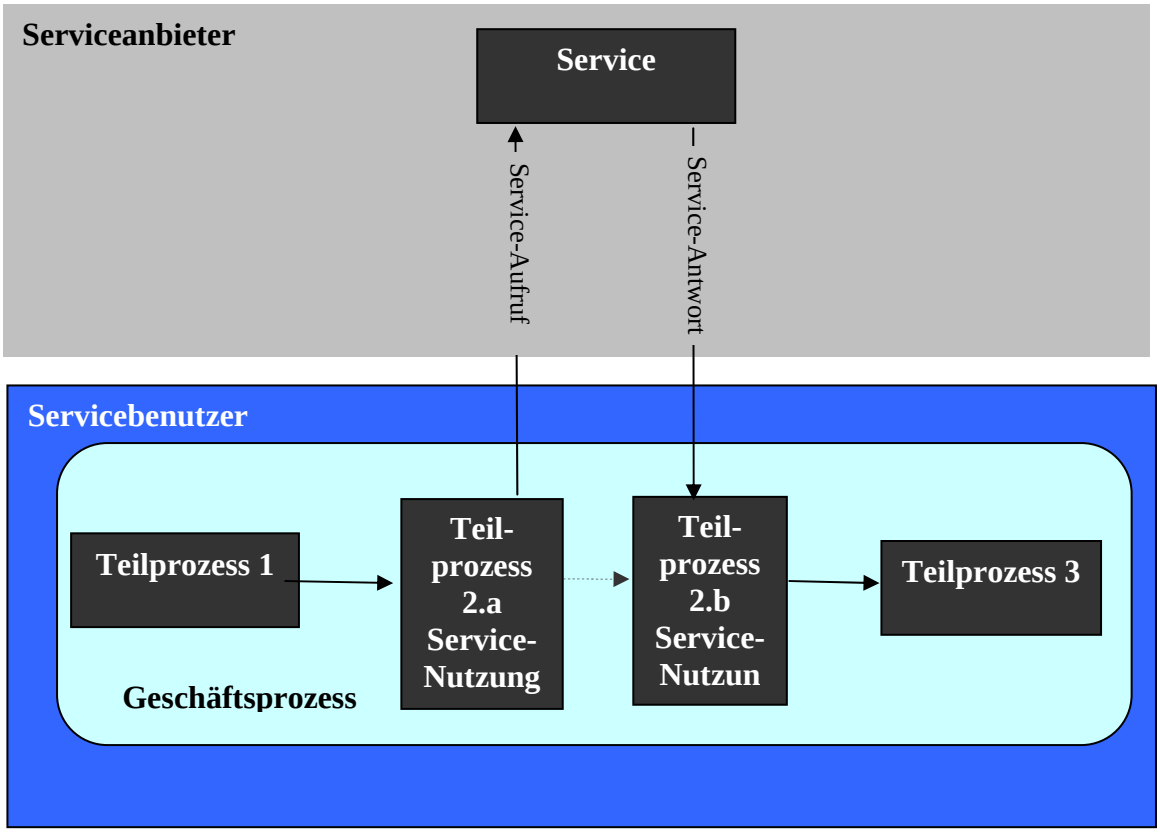

Abbildung 17: Kommunikationsabläufe bei der synchronen Nutzung eines Services

Bei dem asynchronen Mechanismus (Abbildung 18) sperrt das Sendersystem nicht den betroffenen Prozessfluss. Nachdem die Daten vom Service an den Servicenutzer übertragen wurden, können sie problemlos weiter bearbeitet werden.

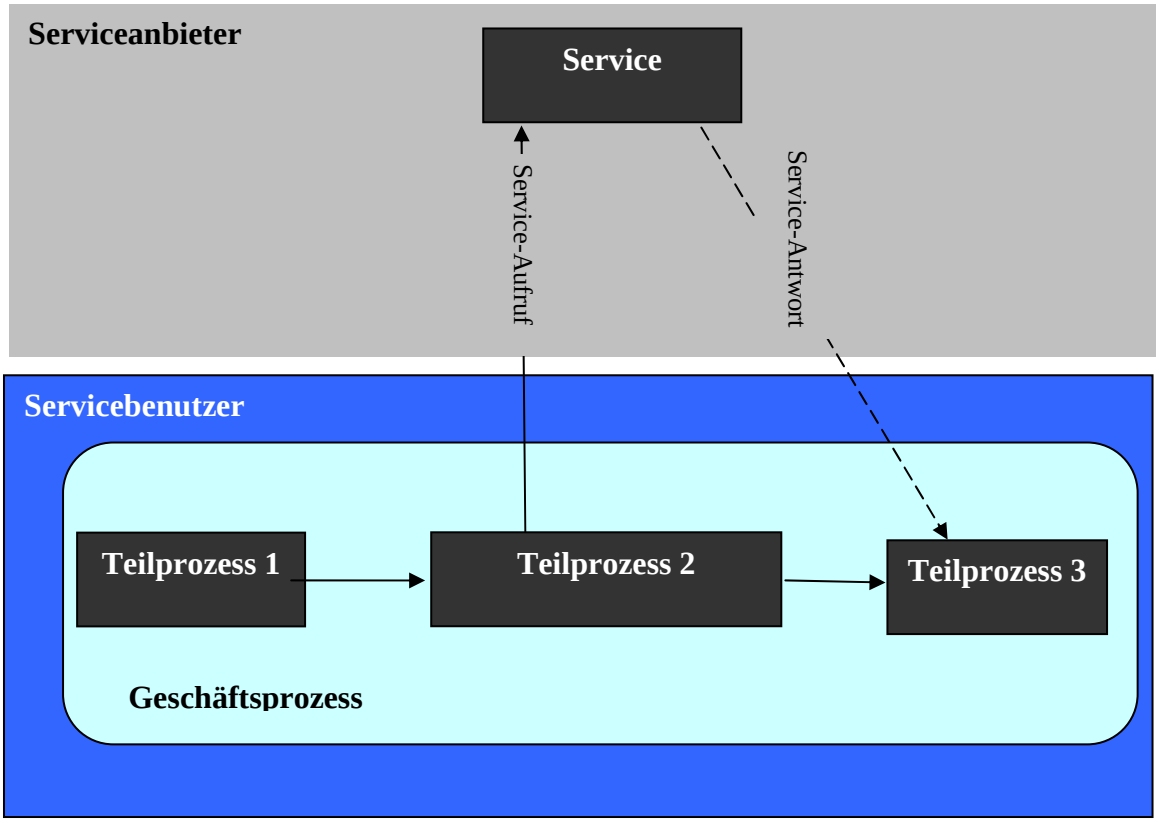

Abbildung 18: Kommunikationsabläufe bei der asynchronen Nutzung eines Services

In dem folgenden Abschnitt wird basierend auf der im letzten Abschnitt beschriebenen Gesichtspunkten eine passende Datenintegrationsarchitektur entworfen.

5.2.4 Architektur für die Datenintegration

Die Komponenten der Architektur für die Datenintegration befinden sich jeweils in zentralen und dezentralen Systemen. Abbildung 19 bietet dazu ein Überblick über die entwickelte Architektur.

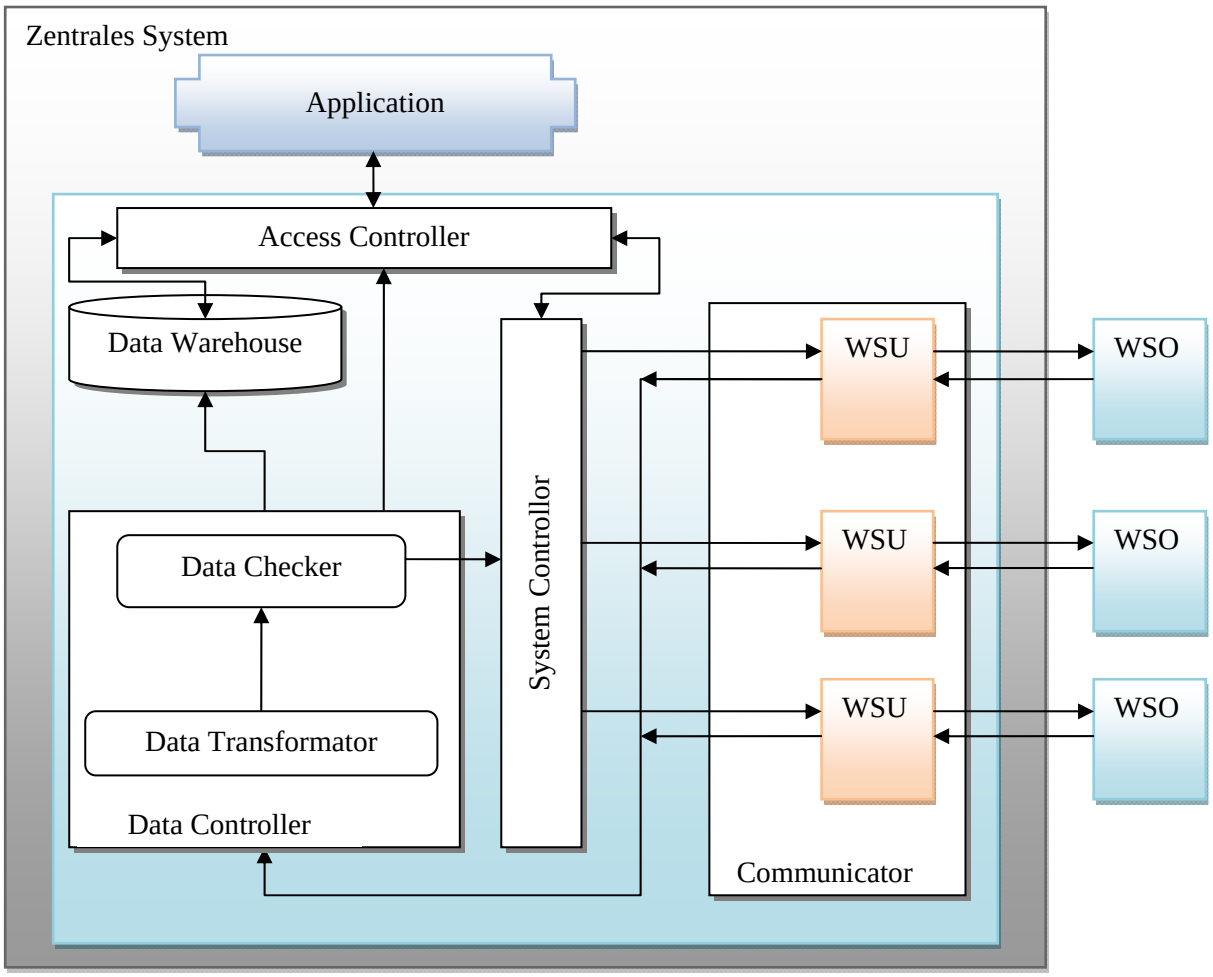

Abbildung 19: Systemarchitektur

Im zentralen System befinden sich fünf wichtige Komponenten: Access Controller, Data Warehouse, System Controller, Data Controller, und Communicator. Die Komponenten übernehmen die im Folgenden beschriebenen Aufgaben.

- Access Controller

  Es gibt zwei verschiedene Integrationsmechanismen in diesem Konzept und die Entscheidung des zu verwendenden Mechanismus wird von dieser Systemkomponente getroffen. Das heißt, die Anwendung braucht nur die Anfragen bezüglich der Daten an diese Komponente zu schicken und bekommt als Ergebnis die entsprechenden Daten zurück. Der dafür verwendete Integrationsmechanismus ist für diese Anwendung transparent.

- Data Warehouse

  Das Data Warehouse ist eine der Kernkomponenten der Integrationsarchitektur. Es ist die zentrale Stelle, an der sich die Datensammlung befindet. In den meisten Fällen wird es

durch eine Datenbank umgesetzt. Die Daten werden von den Datenquellen in das Data Warehouse geladen und dort vor allem für die Datenanalyse und zur betriebswirtschaftlichen Entscheidungshilfe in Unternehmen langfristig gespeichert.

Daten, deren Gültigkeit langfristig ist, werden einmalig von verteilten und unterschiedlichen strukturierten Datenbeständen aus in das zentrale System transportiert und im Data Warehouse (DWH) gespeichert. Davor müssen sie noch durch entsprechende Transformationsvorgänge bereinigt und vereinheitlicht werden, um damit im DWH eine globale Sicht auf die Quelldaten und damit übergreifende Auswertungen zu ermöglichen.

- System Controller

Der System Controller liegt auch im zentralen System und ist eine Komponente, mit deren Hilfe auf die externen Datenquellen einheitlich zugegriffen werden kann. Er funktioniert ähnlich wie ein Mediator im virtuellen Datenintegrationsansatz. Das Anwendungssystem kann durch diesen Controller die relevante Datenquelle finden, um von dort die angeforderten Daten zu holen.

- Data Controller

Der Data Controller beschäftigt sich hauptsächlich mit den zu integrierenden Daten. Da die im zentralen System eintreffenden Daten normalerweise noch nicht vereinigt sind und deren Korrektheit auch noch nicht geprüft ist, werden durch diese Komponente alle verwendeten Daten, die aus anderen Systemen kommen, in eine für das zentrale Anwendungssystem geeignete Form gebracht und geprüft. Hierzu müssen auch die durch die Prüfungen als fehlerhaft identifizierten Daten noch einmal von dem entsprechenden dezentralen System abgeholt werden.

Um die oben genannten Funktionen zu realisieren, wird diese Komponente wiederum in zwei Teilkomponenten unterteilt: *Data Transformator* und *Data Checker*. Der Data Transformator, übernimmt die Aufgabe der Datentransformation, während der Data Checker die Aufgabe hat, die ankommenden Daten auf ihre Korrektheit hin zu überprüfen.

- Communicator

Der Communicator ist die Verbindungskomponente zwischen dem Anwendungssystem und externen Systemen. Da hier die verwendete Kommunikationsinfrastruktur auf einem Service basiert, besteht der Communicator aus einem zweistufigen Web Service-Ansatz: Web Service Offer (WSO) und Web Service User (WSU). Für jedes zu integrierende externe System existiert ein WSO-WSU-Paar.

Der Web Service Offer wird vom Betreiber eines Datenbanksystems konfiguriert. Er beinhaltet mehrere von externen Systemen angebotenen Services. Der Betreiber des Datenbanksystems und des Web Service Offers hat damit die Kontrolle über die nach

außen freigegebenen Zugriffsmöglichkeiten. Der Web Service Offer kann ohne Programmieraufwand konfiguriert werden und läuft auf einer vom Betreiber des externen Systems bereitgestellten Plattform. Die Kommunikation zwischen diesem Web Service Offer und den externen Systemen kann mittels eines beliebigen, i. d. R. vom Datenbanksystem abhängigen, Protokolls erfolgen. Da der Web Service Offer seinen Dienst via SOAP anbietet, entfallen Konflikte mit der Firewall. Auch die sicherheitskritische Freischaltung weiterer Ports ist nicht notwendig.

Der im zentralen System installierte Web Service User (WSU) greift auf die einzelnen Web Service Offer (WSO) der externen Systeme zu. Diese Zugriffe erfolgen mit Hilfe von SOAP. Die aus den Anfragen an diesen Web Service Offer stammenden Daten, die durch das Web Service-Prinzip bereits in einem XML-Format vorliegen, werden durch den Web Service User in ein einheitliches XML-Format transformiert. Diese Daten werden dann in den Data Controller übertragen und dort je nach Anforderungen weiter umgeformt und überprüft.

Der detaillierte Ablauf des Datenintegrationsprozesses wird im folgenden Abschnitt beschrieben.

5.2.5 Ablauf der Datenintegration

Der Ablauf des Datenintegrationsprozesses wird durch die folgende Abbildung 20 veranschaulicht.

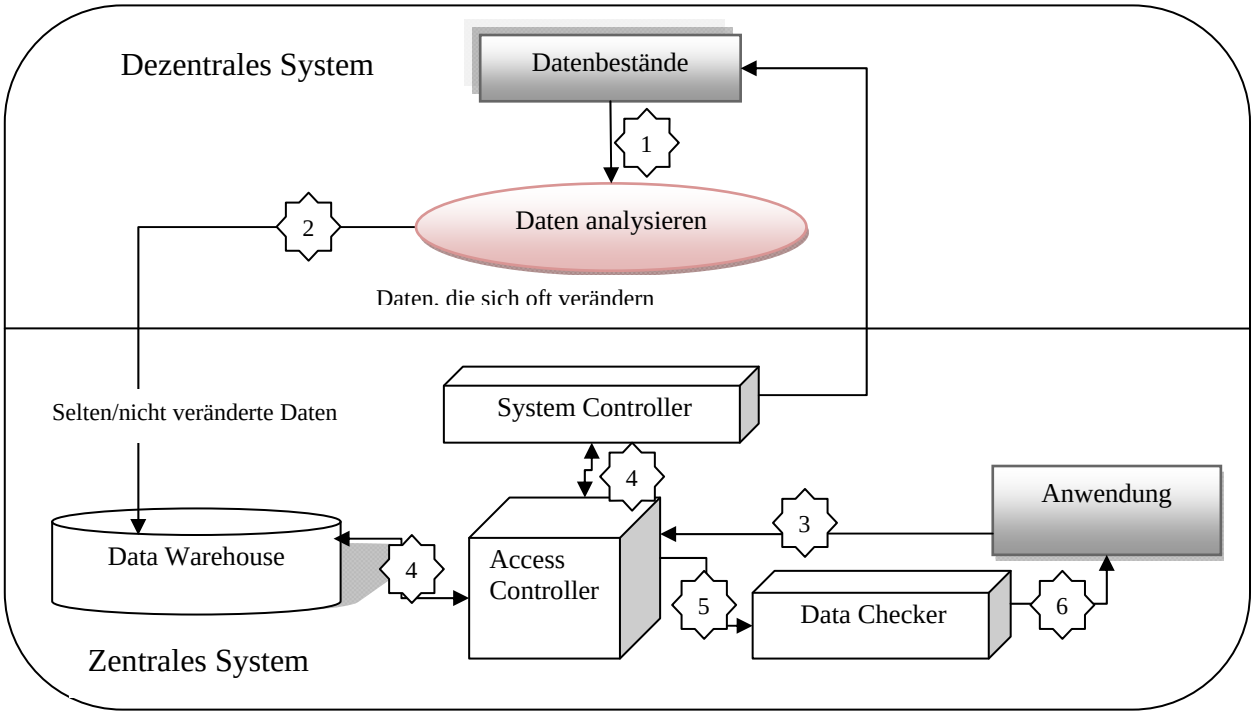

Abbildung 20: Ablauf des Datenintegrationsprozesses

Wie in der Abbildung 20 dargestellt ist, werden die zu integrierenden Daten zuerst in zwei Kategorien unterteilt:

- Daten, die sich nicht oder selten verändern
- Daten, die sich oft verändern

Anhand dieser Eigenschaft werden die Daten am Anfang der Integration analysiert und mit verschiedenen Integrationsmechanismen ins zentrale System transportiert.

Am Anfang des Integrationsprozesses werden alle Daten, die meistens als Stammdaten behandelt werden und sich deswegen nicht so oft ändern, einmalig ins zentrale System transportiert und dort im Data Warehouse als Replikat gespeichert. Dieser Vorgang kann durch einen entsprechenden asynchronen Service-Aufruf realisiert werden. Nach der Übertragung aller Daten ins zentrale System werden sie in die benötigte Form umstrukturiert und transformiert. Dieser Vorgang findet in der Komponente „Data Transformer“ statt. Danach werden entsprechende Datenprüfungsprozesse in der Komponente „Data Checker“ durchlaufen, damit mögliche Fehler bei der Datenübertragung rechtzeitig erkannt werden können. Im Fehlerfall werden die Daten nochmal vom entsprechenden externen System abgerufen. Die Wahl des entsprechenden externen Systems erledigt die Komponente „System Controller“. Korrekt übermittelte Daten werden dann direkt im zentralen System verwendet und gewartet. Bei Veränderungen oder Aktualisierungen an den Daten im dezentralen Systemen werden die modifizierten Daten wiederum durch asynchrone Services ins zentrale System gesendet und dort aktualisiert.

Im Gegensatz zu den sich selten ändernden Stammdaten werden die Daten, die öfters eine Aktualisierung erfordern, zur Laufzeit der Anwendung direkt vom entsprechenden externen System abgeholt. Die übertragenen Daten werden vor ihrer Weiterbearbeitung auch transformiert und auf Fehler überprüft. Wenn Fehler auftreten, müssen die Daten wie schon beschrieben neu abgeholt werden. Wenn keine Fehler mehr existieren, werden die Daten im zentralen System durch die Anwendung weiter bearbeitet.

Das Ergebnis der Datenbearbeitung im zentralen System kann durch einen asynchronen Service je nach Anwendungsanforderungen ins entsprechende dezentrale System zurück transportiert und dort gespeichert werden.

5.3 Zusammenfassung und Ausblick

Die service-basierte Datenintegration ermöglicht eine einfache und flexible Integration von Datenbeständen, die über das Internet verteilt sind. Durch die Bereitstellung der Daten im XML-Format ist deren einfache Weiterverarbeitung durch Anwendungen möglich.

Die entworfene Systemarchitektur bietet mehrere Zugriffsmöglichkeiten auf die Daten in verteilten Systemen. Durch den Data Warehouse-Ansatz wird die Performanz des Datenaustausches gewährleistet und durch den virtuellen Datenintegrationsmechanismus wird gleichzeitig auch die Aktualität der entsprechenden Daten sichergestellt.

Die Zusammenführung der Daten aus verschiedenen Datenbeständen ins Data Warehouse ist in zweifacher Hinsicht ein Bereich, in dem verstärkte Forschungstätigkeit notwendig ist. Zum einen ist die Erkennung der redundanten Daten wichtig, zum anderen ist die Frage zu klären, wie oft ein Datenupdate-Zyklus durchgeführt werden sollte, damit die Systemleistung nicht zu stark beeinträchtigt und dabei gleichzeitig die Aktualität der Daten im Data Warehouse gewährleistet wird.

Kapitel 6

Fallstudie

Um das im Kapitel 5 entworfene Konzept der service-basierten Datenintegration für verteilte Datenbestände zu evaluieren und die Funktionsweise zu demonstrieren, wird im Folgenden eine Fallstudie im Rahmen eines SAP-Anwendungssystems durchgeführt. Durch diese Fallstudie wird eine unternehmensspezifische Implementierung des Integrationskonzeptes in der Praxis vorgestellt.

Das Anwendungsgebiet, für das diese Fallstudie implementiert wird, ist der Geschäftsprozess der Rechnungsprüfung, der ein Teilprozess des Beschaffungsprozesses aus dem Modul des SAP ERP MM (Material Management) ist. Die Motivation entstammt der Idee, dass der Rechnungsprüfungsprozess nicht mehr in jedem lokalen System, sondern nur noch zentral durchgeführt wird. Der bisherige Rechnungsprüfungsprozess im SAP ERP System findet in jedem dezentralem logistischem System statt. Ein Logistiksystem steuert Transport- und Lagerungsprozesse. Zur Kostensenkung wird angestrebt, diesen Prozess in eine zentrale Stelle zu verschieben. Diese Idee entspricht dem im Kapitel 2.1.2 beschriebenen SSC-Konzept.

Die verwendete Integrationsinfrastruktur ist die service-orientierte Architektur. Im Unterschied zu der klassischen SOA wird von der SAP AG der Business Content eingeführt. Business Content bedeutet hier, dass nicht nur eine gemeinsame technologische Basis für die Prozessabwicklung eingesetzt, sondern auch ein gemeinsames Verständnis für die betriebswirtschaftlichen Inhalte gefordert wird [HEIL07]. Der Abschnitt 6.2 geht noch einmal genauer auf dieses Thema ein.

Bevor jedoch auf die konkrete Umsetzung des Konzeptes eingegangen wird, wird zuerst der Rechnungsprüfungsprozess im SAP ERP MM und der entsprechende betriebswirtschaftliche Hintergrund erläutert.

6.1 Betriebswirtschaftlicher Hintergrund

In diesem Abschnitt werden die nötigen betriebswirtschaftlichen Grundlagen zum Rechnungsprüfungsprozess erklärt.

Die für das Verständnis der Integrationsproblematik der Rechnungsprüfung wichtigste Komponente ist SAP ERP. Sie wird im Folgenden zuerst beschrieben.

6.1.1 SAP ERP

SAP ERP ist eine Komponente aus der SAP Business Suite. Sie unterstützt die Kern- und erweiterten Geschäftsprozesse sowohl von Mittelstandskunden als auch von großen Unternehmen. Sie besteht aus einer Software, die eine schnelle und effiziente Koordination, Planung und Ausführung aller Prozesse in einem anspruchsvollen Umfeld ermöglicht. SAP ERP fasst mehrere ERP-Funktionen zusammen und besteht aus fünf wichtigen Geschäftsprozessen: [FLTW06]

- Analystics
- Financials (Finanzwesen)
- Human Resources (Personalwesen)
- Corporate Services (Unternehmensdienstleistungen)
- Operations (Produktion und Logistik)

Die Komponente, die in dieser Arbeit näher betrachtet werden soll, ist SAP ERP Operations. Sie ermöglicht die Abbildung durchgängiger Logistikprozesse von der Bestellung / Beschaffung bis zur Bezahlung / Auslieferung.

In SAP ERP ist der Beschaffungsprozess, der auch den Rechnungsprüfungsprozess beinhaltet, ein Bereich der Materialwirtschaft (MM), die eine Komponente der ERP-Operations ist.

Ein Unternehmen kann mehrere ERP-Systeme installieren, wenn es z.B. mehrere Standorte hat. In jedem Standort finden dann eigene, selbständige Geschäftsprozesse (z.B. der Beschaffungsprozess) in einem eigenen SAP-System statt. Eine mögliche Unternehmenslandschaft kann mit folgendem Bild veranschaulicht werden.

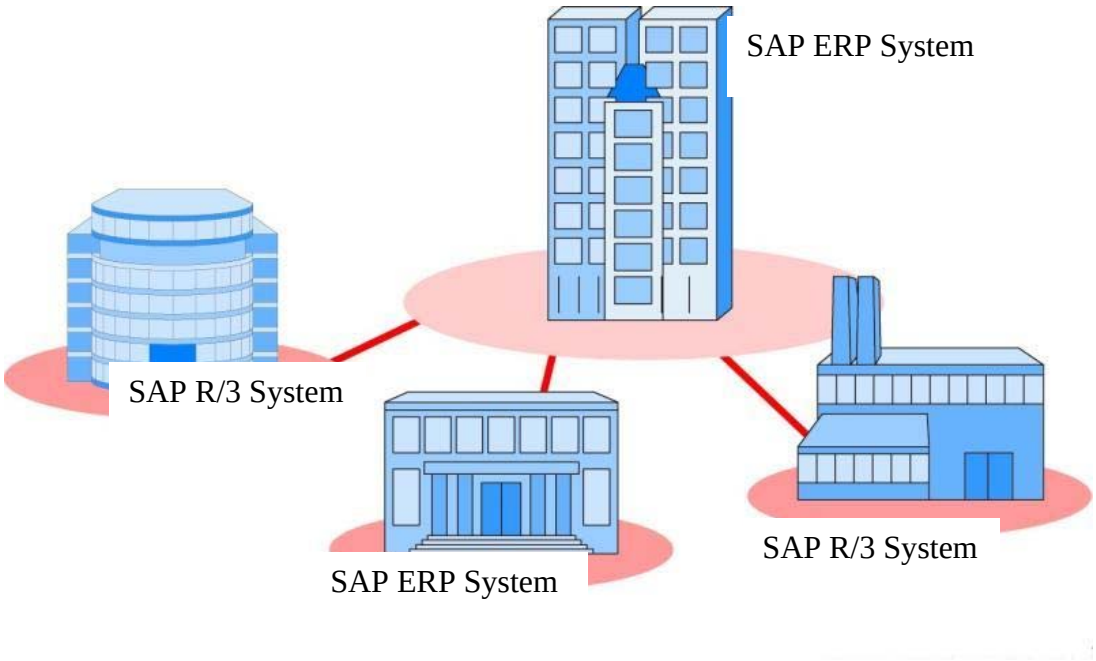

Abbildung 21: ERP-Landschaft

6.1.2 Geschäftsprozess der Beschaffung in SAP ERP

Der Beschaffungsprozess ist einer der grundlegenden Prozesse im Einkauf eines Unternehmens. Ein Beschaffungszyklus kann durch Abbildung 22 dargestellt werden:

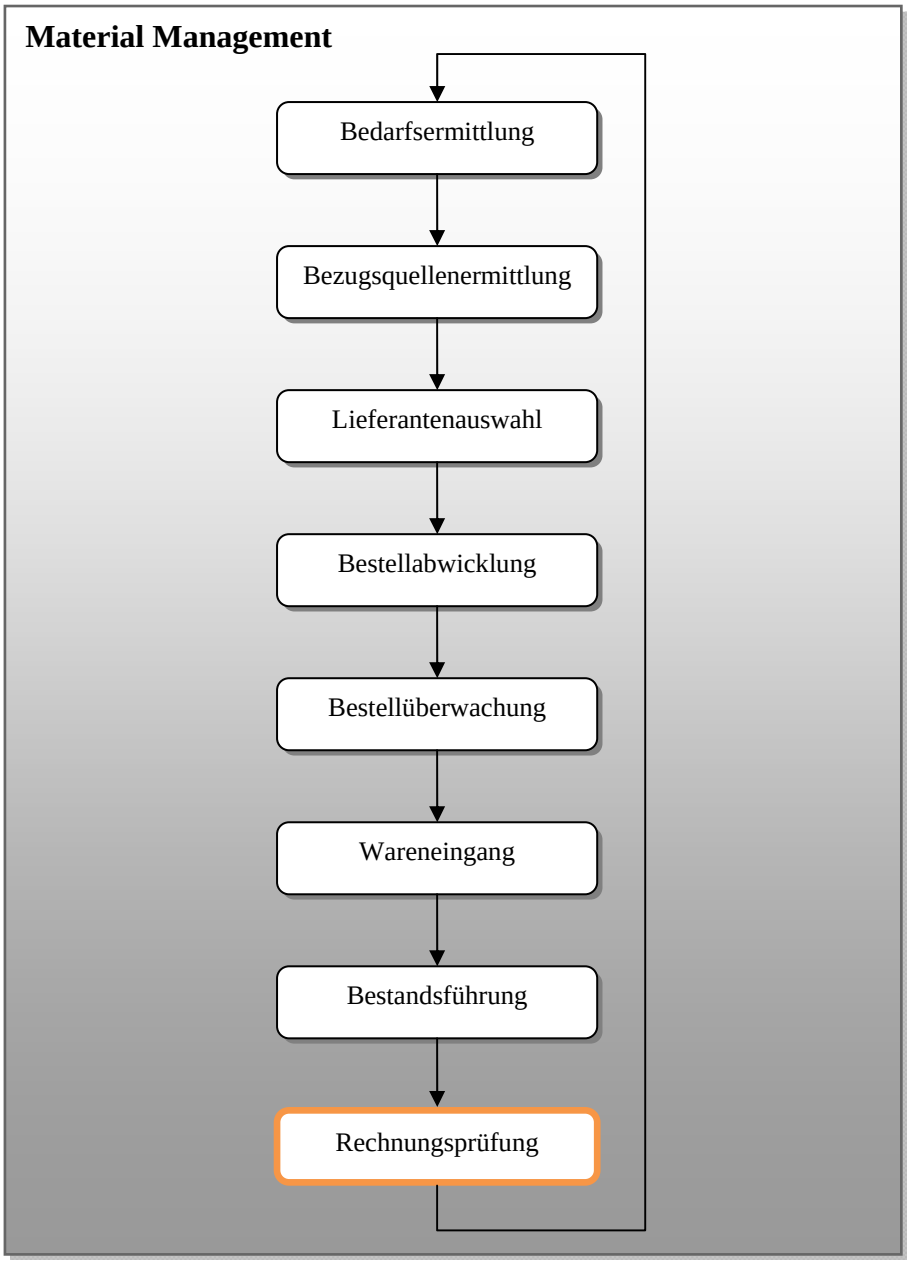

Abbildung 22: Beschaffungszyklus

Bei der Ermittlung des im Unternehmen aufgetretenen Bedarfs an Gütern oder Leistungen werden zuerst die möglichen Bezugsquellen unter Berücksichtigung vergangener Bestellungen oder bestehender Kontrakte im System ermittelt. Die geeigneten Lieferanten werden ausgewählt und die benötigten Bestellungen anhand dieser Informationen im System erzeugt. Jede Bestellung enthält sowohl Daten wie das Bestelldatum und den Namen des Lieferanten als auch Positionen mit den Informationen zu den angeforderten Materialien (z.B. Bezeichnung, Menge und Preis). Die Einhaltung der Lieferfrist wird vom System überwacht und sobald ein Wareneingang verzeichnet wird, kann dieser unter der Eingabe der entsprechenden Bestellnummer im System bestätigt werden. Nachdem ein Wareneingang im System verzeichnet wurde, wird der Bestand der eingelieferten Ware im Lager sofort

automatisch aktualisiert. Nebenbei wird die eingegangene Lieferantenrechnung auf ihre Richtigkeit und Vollständigkeit geprüft. Rechnungen werden anhand der entsprechend im System existierenden Bestellungen und Wareneingangsdaten auf ihre Korrektheit geprüft und anschließend bezahlt, wenn sie fehlerfrei sind.

Der Beschaffungsprozess kann, wie in der Abbildung 21: ERP-Landschaft dargestellt ist, in jedem lokalen Logistiksystem stattfinden. Wenn ein Unternehmen mehrere Standorte und dort jeweils ein lokales Logistiksystem installiert hat, muss der vollständige Beschaffungszyklus in jedem lokalen System ausgeführt werden.

6.1.3 Geschäftsprozess der Rechnungsprüfung

Wie in Abschnitt 6.1.2 bereits beschrieben, ist die Rechnungsprüfung ein Teil des Beschaffungsprozesses. Dieser Prozess steht am Ende der logistischen Kette von Einkauf und Bestandsführung.

Der Gegenstand der Rechnungsprüfung ist die eingehende Rechnung. Es wird zuerst im folgenden Abschnitt eine genaue Definition der Rechnung gegeben und die Informationen, die eine Rechnung enthält, werden erwähnt. Die Beziehung zwischen einer Rechnung und anderen Objekten des Einkaufs wird ebenfalls verdeutlicht.

6.1.3.1 Rechnung

Eine Rechnung ist die Aufstellung eines Rechnungsstellers über zu leistende Entgelte aufgrund vorangegangener Geschäftsvorgänge im Einkauf und in der Bestandsführung [SLRP01].

Seit dem 1. Januar 2004 sind die Rechnungssteller verpflichtet, neben Menge und handelsüblicher Bezeichnung der gelieferten Ware sowie der Art und dem Umfang erbrachter Dienstleistung noch folgende wichtige Angaben in einer Rechnung zu machen: [IHKS05]

- Vollständiger Name und Anschrift des leistenden Unternehmens (Rechnungsaussteller) und des Leistungsempfängers (Rechnungsempfänger).

- Das Ausstellungsdatum der Rechnung und der Zeitpunkt der Lieferung.

- Eine eindeutige Rechnungsnummer, die zur Identifizierung der Rechnung vergeben wird.

- Die Rechnungsbeträge, die nach Steuersätzen und einzelnen Steuerbefreiungen aufgeschlüsselt werden. Der anzuwendende Steuersatz und Steuerbezeichnung muss auch in jeder Rechnung klar genannt werden.

- Die Steuernummer oder eine Umsatzsteuer-Identifikationsnummer.

Außer den oben aufgelisteten Pflichtangaben einer Rechnung ist es bei einer ins SAP-System eingehenden Rechnung auch wichtig, worauf sich die Rechnung bezieht. Die Rechnung sollte eindeutig einer im System existierenden Bestellung, Leistung oder einem Wareneingang zuzuordnen sein, um die inhaltliche Richtigkeit der Rechnung sicherzustellen.

6.1.3.2 Rechnungsprüfungsprozess in SAP ERP

Die zentrale Aufgabe der Rechnungsprüfung ist es, eingehende Rechnungen auf sachliche, preisliche und rechnerische Richtigkeit zu prüfen.

Im SAP ERP System wird zuerst die eingehende Rechnung anhand der entsprechenden Bestellung oder des Wareneingangs geprüft. Korrekte Rechnungen werden gebucht und deren Daten im System gespeichert.

Ein typischer Ablauf des Rechnungsprüfungsprozesses in einem Unternehmen ist in der Abbildung 1 dargestellt.

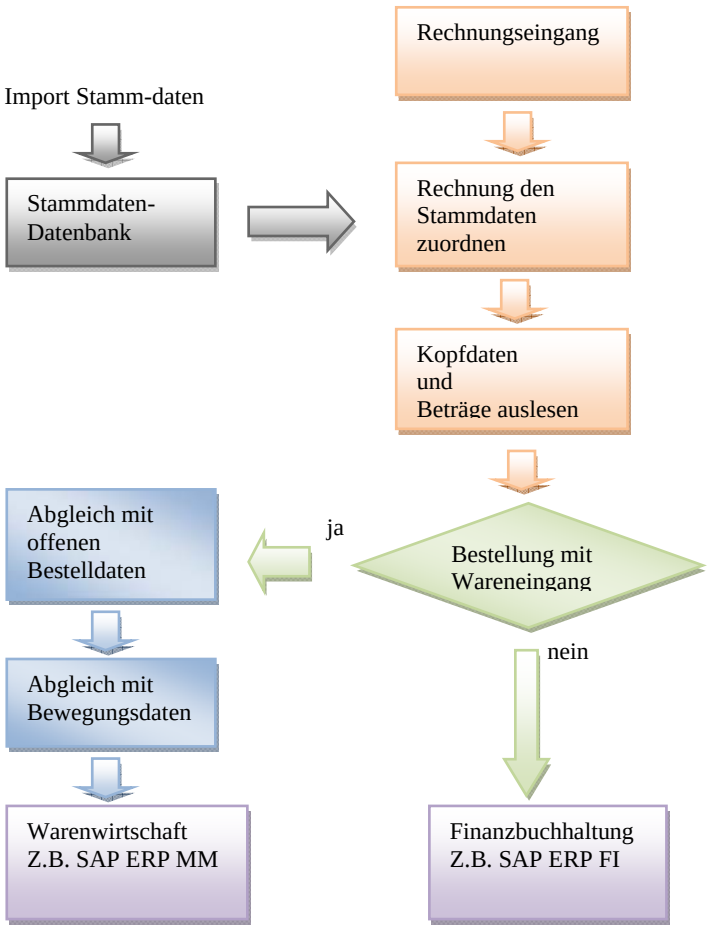

Abbildung 23: Rechnungsprüfungsprozess

Es gibt verschiedene Möglichkeiten, um Rechnungen an ein Unternehmen zu übermitteln, z.B. per Post oder auf elektronischem Weg. Papierrechnungen können entweder manuell von einem Sachbearbeiter in das System eingegeben oder durch OCR-Software eingescannt werden. Rechnungen können aber auch durch elektronischen Datenaustausch von einem anderen System aus übertragen werden. Die im ERP-System erfassten Rechnungen werden dann dahingehend überprüft, ob die Stammdaten, auf die in der Rechnung referenziert werden, existieren. Außerdem finden auch andere Prüfungen statt, wie z.B. auf Mengen-, Preis- oder Terminabweichungen. Die fehlerhaften Rechnungen können entweder von Sachbearbeitern korrigiert und gebucht oder zur Zahlung gesperrt werden.

Am Anfang dieses Abschnittes wurde schon erwähnt, dass um eine Rechnung auf ihre inhaltliche Richtigkeit zu prüfen, eine Beziehung zwischen der Rechnung und Bestellung bzw. Wareneingang hergestellt werden muss, damit in der Rechnungsprüfung die Daten, die in einer Rechnung stehen, mit den entsprechenden Bestell- und Wareneingangsangaben verglichen werden können.

Die Rechnungsprüfung ist kein isolierter Prozess im SAP-System. Sie setzt die Prozesse des Einkaufs und der Bestandsführung voraus. Nach der Prüfung und Buchung der eingehenden Rechnung können von der Rechnungsprüfung ein MM- und ein FI-Rechnungsbeleg erzeugt werden. Diese beiden Rechnungsbelege dienen der Fortschreibung in der Materialwirtschaft und in der Finanzbuchhaltung.

Da der Rechnungsprüfungsprozess dort stattfindet, wo auch der Beschaffungsprozess läuft, kann er problemlos von den lokalen Systemen alle wichtigen Information bekommen und die daraus erzeugten, neuen Informationen ins lokale System schreiben. Im folgenden Abschnitt wird eine neue Möglichkeit für die Rechnungsprüfung beschrieben, wo der Fall, dass auf alle benötigen Informationen einfach lokal zugegriffen werden kann, nicht mehr gilt.

6.1.4 Zentrale Rechnungsprüfung als neuer Ansatz

Der bisherige Rechnungsprüfungsprozess im SAP ERP-System findet lokal in jedem Logistiksystem statt. Es gibt in diesem Fall bei großen Unternehmen, die mehrere Standorte haben und deswegen mehrere Logistiksysteme besitzen, das gleiche Problem wie bei dem im Abschnitt 2.1.1 schon beschriebenen Szenario: Die Dezentralisierung der Verarbeitung des Geschäftsprozesses bringt mehr Wartungs- und Personalkosten mit sich. Des Weiteren ist es auch nicht optimal, wenn bei jedem lokalen System ein Update durchgeführt werden muss, wenn sich der Geschäftsprozess ändert. Eine gute Lösung dafür ist die Zentralisierung der Prozesse, wenn sie in jedem lokalem System gleich oder ähnlich ablaufen. Der Rechnungsprüfungsprozess ist ein solcher Prozess.

Die Verlagerung der Rechnungsprüfungsprozesse von jedem lokalen System zu einer zentralen Stelle bringt folgende Vorteile:

- Weniger Administrationskosten
- Weniger Personalkosten
- Niedrigere Fehlerraten
- Niedrigerer Wartungsaufwand

Ausgehend von dieser Idee soll ein Software-System, das als Rechnungseingangsbuch fungiert, in das zentrale System integriert werden, um sämtliche ins Unternehmen eingehenden Rechnungen zentral zu überwachen und zu bearbeiten.

Das Rechnungseingangsbuch sollte folgende Funktionalitäten beinhalten:

- Erfassung eingehender Lieferantenrechnungen
- Prüfung der Rechnungen auf Richtigkeit und Vollständigkeit
- Klassifizierung auftretender Probleme
- Fehlerspezifische Weiterbearbeitung
- Identifikation doppelter Rechnungen
- Freigabe der Rechnungen zur Buchung
- Kommunikation mit beteiligten Systemen
- Vollständige Protokollierung der Kommunikation zu einer späteren Nachverfolgung.

Das Design und die Implementierung eines solchen Software-Systems sind sehr umfangreich und komplex. Es ist unmöglich, alle Aspekte in einer Masterarbeit zu behandeln. Die Aufgabe dieser Arbeit ist die Entwicklung eines EAI-Konzeptes, das der service-basierten Datenintegration von verteilten Datenbeständen dient und die Überprüfung auf dessen Anwendbarkeit.

Eine wichtige Funktion des zu entwickelnden Rechnungseingangsbuches ist die Kommunikationsfähigkeit mit den beteiligten Systemen. Dies beinhaltet folgende Aspekte:

- Das Quellsystem finden, in dem die Bestellung liegt, auf die sich die zu prüfende Rechnung bezieht.
- Die Bestell- oder Wareneingangsdaten, die zur Prüfung der eingehenden Rechnung wichtig sind, von dem entsprechenden Quellsystem abholen.
- Die Daten, die von der Rechnungsprüfung erzeugt wurden, ins Quellsystem übertragen.

Die Implementierung dieser Funktionalität benötigt eine optimale Datenintegration der verteilten Systeme. Aus diesem Grund ist dies ein passender Anwendungsfall für das entwickelte Konzept.

Im Folgenden wird das im vorherigen Kapitel beschriebene Konzept für diesen Anwendungsfall umgesetzt. Ein Prototyp des Rechnungseingangsbuchs wurde implementiert.

6.2 Umsetzung und Implementierung des EAI-Konzeptes

In der Rechnungsprüfung – wie in anderen Anwendungen auch – muss das SAP System an vielen Stellen auf gespeicherte Daten zugreifen, diese Daten ggf. ändern oder neue Informationen hinzufügen. Die benötigten Daten sind aber im Fall des zentralen Bearbeitungsprozesses über mehrere Systeme verteilt. Die Daten, die aus den Lieferantenrechnungen kommen, liegen im zentralen Bearbeitungssystem. Andere wichtige Daten, wie z.B. Stammdaten und Bestell- oder Wareneingangsdaten, die zur Prüfung der Rechnung benötigt werden, befinden sich nur im dezentralen logistischen System.

Es stellt sich dann die Frage, wie die Daten von den dezentralen logistischen Systemen in das zentrale System transportieren werden, damit ein zentraler Rechnungsprüfungsprozess reibungslos ablaufen kann. Hierzu wird das in Kapitel 5 entwickelte Konzept verwendet, um die verteilten Datenbestände in einem zentralen Anwendungssystem zu integrieren.

6.2.1 Datenanalyse

Bevor der Integrationsprozess ausgeführt wird, sollten zuerst die zu integrierenden Daten und deren Verteilung im System detailliert analysiert werden. Die betroffenen Daten werden in zwei Kategorien unterteilt: Eingangsdaten und Ausgangsdaten.

6.2.1.1 Eingangsdaten

Um eine Rechnung erfassen, prüfen und schließlich erfolgreich buchen zu können, werden generell die folgenden Eingangsdaten vom zentralen Prüfungssystem benötigt.

- Rechnungsdaten

  Sie bestehen aus allen Informationen, die sich in einer Rechnung befinden, wie z.B. Name und Anschrift des Rechnungsstellers bzw. Rechnungsempfängers, Rechnungsbeträge und Steuersätze. Eine weitere wichtige Angabe in einer Rechnung ist deren Bezug zu der entsprechenden Bestellung.

- Stammdaten

  Stammdaten sind Daten über Objekte in einem SAP-System, die zur Abbildung der organisatorischen Einheiten der Konsolidierung dienen. Das System greift bei der Bearbeitung der Geschäftsprozesse auf die benötigen Stammdaten zu. Stammdaten werden normalerweise einmal zentral in jedem lokalen System gespeichert und in allen Modulen verwendet. Wichtige Stammdaten sind hier z.B. Materialstammdaten, die Informationen über Rohstoffe, Handelswaren, Halbfabrikate und Fertigerzeugnisse beinhalten, und die

Lieferantenstammdaten, die Angaben zu einem Lieferanten oder Kreditor umfassen [SLRP01].

- Bestelldaten

  Bestelldaten sind die Daten, die in Bezug zu einer Bestellung stehen, wie z.B. bestelltes Material, Menge und Preis pro Einheit. Es wird dann in dem Rechnungsprüfungsprozess ein Vergleich zwischen Rechnungsdaten und den entsprechenden Bestelldaten durchgeführt, um die inhaltliche Richtigkeit der Rechnung zu überprüfen. Bestelldaten werden in jedem lokalen logistischen System erzeugt.

- Bewegungsdaten

  Bewegungsdaten sind vorgangsbezogene Daten, die nur kurzlebig sind. Bei jeder Buchung – sei es bei der Erfassung einer Bestellung, eines Lieferplans, eines Wareneingangs – wird automatisch ein Beleg darüber erstellt [SLRP01]. Sie sind Ergänzungen zu bestimmten Stammdaten, d.h. sie dienen der Konkretisierung eines Geschäftsvorfalls (Menge, Termin, etc.).

  Bewegungsdaten ändern sich sehr schnell aufgrund der schnellen logistischen Vorgänge. Die Daten, die in einer Rechnung stehen, werden evtl. auch mit den aktuellen Bewegungsdaten verglichen, um ihre Genauigkeit zu prüfen. Ein Beispiel dafür ist die Menge der eingegangenen Waren.

Rechnungsdaten können über verschiedene Wege in das zentrale Prüfungssystem kommen. Dieser Vorgang wurde in Abschnitt 6.1.3.2 schon beschrieben. Stammdaten, Bestelldaten und Bewegungsdaten stehen anfänglich nur in den lokalen logistischen Systemen zur Verfügung. Sie sind aber die wichtigsten Daten, die später ins zentrale System integriert werden müssen.

6.2.1.2 Ausgangsdaten

Wie in Abschnitt 6.1.3.2 erwähnt, setzt die Rechnungsprüfung die Komponenten des Einkaufs und der Bestandsführung voraus. Bei Buchung der eingegangenen Rechnungen werden von der Rechnungsprüfung ein MM- und FI-Rechnungsbeleg erzeugt. Diese beiden Belege dienen der Fortschreibung in der Materialwirtschaft und in der Finanzbuchhaltung. Somit gibt die Rechnungsprüfung Informationen über die eingegangene Rechnung an die Finanzbuchhaltung, die Anlagenbuchhaltung und die Kostenrechnung weiter. Da sich die Materialwirtschaft und Finanzbuchhaltung wiederum im dezentralen logistischen System befinden, sollte der neue zu erzeugende Beleg auch im dezentralen System angelegt werden. Aus diesem Grund müssen die von der Rechnungsprüfung kommenden Daten wiederum ins dezentrale System zurück transportiert werden, damit die entsprechenden Belege dort dezentral erzeugt werden können.

6.2.2 Die konkrete Datenintegrationsstrategie

Nach der Analyse der zu integrierenden Daten wird in diesem Abschnitt die Umsetzung und Implementierung des im Kapitel 5 vorgestellten Konzeptes aufgezeigt.

6.2.2.1 Umsetzung des EAI-Konzeptes

Folgende Aspekte aus der Entwurfsphase des Konzeptes werden betrachtet:

- Datenintegrationsmechanismus

  Die *Rechnungsdaten* werden entweder durch manuelle Eingabe oder über ein elektronisches Format, z.B. mittels OCR-Software, im zentralen System angelegt. Alle wichtigen Informationen aus der Rechnung können im DWH gespeichert und später von der Anwendung direkt verwendet werden.

  Die *Stammdaten* werden im dezentralen logistischen System gepflegt. Die Daten werden am Anfang des Systeminitialisierungsprozesses einmalig ins zentrale System übertragen. Danach werden sie durch entsprechende Transformationen mit festgelegtem Format in das Data Warehouse repliziert. Ein wichtiger Aspekt, der hier betrachtet werden muss, ist, dass die Stammdaten, die aus den verschiedenen lokalen Systemen kommen, evtl. auf das gleiche Objekt referenzieren können, aber wegen der Heterogenität der Daten in den verschiedenen Systemen ist es möglich, dass sie eine unterschiedliche Struktur haben. In diesem Fall muss noch eine Erkennung stattfinden, damit die gleichen Informationen nicht doppelt im System gespeichert werden. Diese Aufgabe wird von der Komponente „Data Transformer“ übernommen. Die Aktualisierung der Stammdaten wird im dezentralen System erkannt und in einem Puffer gespeichert. Die gesammelten Änderungen werden periodisch ins zentrale System übertragen und die entsprechenden betroffenen Stammdaten aktualisiert.

  Jede Rechnung ist mit einer bestimmten Bestellung oder einem Wareneingang verknüpft. Da sich die *Bestelldaten* und *Bewegungsdaten* sehr schnell ändern können, ist es sehr wichtig, deren Aktualität sicherzustellen. Deshalb wird zum spätmöglichsten Zeitpunkt - wenn eine neue Rechnung geprüft werden soll - eine Datenanfrage an das dezentrale System gesendet. Dieses sucht dann die entsprechende Bestellung und die aktuellen Wareneingangs-informationen und sendet alle wichtigen Informationen an das zentrale System zurück. Die Daten werden in das vereinbarte Datenschema umgeformt und zur Rechnungsprüfung verwendet.

  Bevor alle Daten zur Rechnungsprüfung verwendet werden können, müssen sie noch einmal durch die zentrale Komponente „Data Checker“ kontrolliert werden, damit sichergestellt ist, dass alle benötigen Daten fehlerfrei sind. Die fehlenden oder fehlerhaften Daten müssen wiederholt vom lokalen System geholt werden.

Umgekehrt wird das Ergebnis der Rechnungsprüfung wiederum an das betroffene dezentrale System zurückgeliefert, da in dem entsprechenden lokalen System anhand solcher Informationen ein FI- und MM-Beleg erzeugt wird.

Unabhängig davon, ob die Daten von einem dezentralen System geholt oder zu einem dezentralen System geschickt werden, wird eine entsprechender Systemerkennungs-mechanismus benötigt, der das betroffene lokale System findet. Im Folgenden wird der in dieser Fallstudie verwendete Mechanismus erklärt.

- Systemerkennung

Eine SAP-Systemlandschaft kann aus mehreren Systemen bestehen, wobei jedes System eine eigene Systemnummer hat. Ein System besteht wiederum aus mehreren Mandanten (Clients), die für verschiedene Aktivitäten benötigt werden. Ein Mandant ist eine abgeschlossene Einheit innerhalb eines SAP-Systems mit eigenständigen Anwendungsdaten und ist durch eine Mandantenkennung gekennzeichnet.

Die von der Rechnungsprüfung benötigten Stamm-, Bestell- und Bewegungsdaten kommen immer von einen bestimmten Mandanten aus einem bestimmten SAP-System. Wenn eine eingehende Rechnung geprüft wird, muss zuerst der richtige Mandant bzw. das richtige lokale System gefunden werden, wo die referenzierten Daten liegen. Wie kann ein System nun anhand der bestehenden Informationen aus der zu prüfenden Rechnung erkannt werden? Dieses Problem wird von der Komponente „System Controller“ gelöst. Der in dieser Komponente verwendete Erkennungsmechanismus wird im Folgenden genauer beschrieben.

Jedes lokale SAP-System hat eine eigene Systemnummer und unter Umständen mehrere Mandantenkennungen. Durch die Kombination einer Systemnummer mit einer Mandantenkennung kann die Lage der Daten eindeutig bestimmt werden. Die Rechnungssteller haben aber normalerweise keine Kenntnis der Systemlandschaft beim Rechnungsempfänger. Sie wissen beispielsweise nur, wie die Anschrift des entsprechenden Bestellers lautet. Aus diesem Grund ist es notwendig, ein Mapping von Anschrift des Bestellers zu Systemnummer und Mandantenkennung durch Customizing im System zu erstellen, damit eine zentrale Anwendung durch die Anschrift des Bestellers, das System eindeutig bestimmen kann, aus dem die Bestellung stammt.

Ein Beipiel für dieses Mapping kann mit folgender Tabelle veranschaulicht werden.

Anschrift des Bestellers	Systemnmmer	Mandantenkennung
69117 Heidelberg	ERP01	000
69110 Heidelberg	ERP02	000
30625 Hannover	ERP03	001

Tabelle 2 : Beispiel für ein Mapping von Besteller auf Systemkennung

- Verwendete Services

  In diesem Anwendungsfall findet die Übertragung der Daten zwischen verschiedenen Systemen in einem Unternehmen statt. Aus diesem Grund werden A2A Services eingesetzt.

  Die verwendeten A2A Services sollen die Aufgabe der Übertragung der zu integrierenden Daten, z.B. Stamm- und Bewegungsdaten übernehmen.

  Die Prozessabläufe im dezentralen und zentralen System sind voneinander unabhängig, d.h. der zentrale Rechnungsprüfungsprozess kann nicht den lokalen logistischen Prozess beeinflussen. So kann der im lokalen System laufende Prozess fortfahren, ohne die zentrale Datenbearbeitung berücksichtigen zu müssen. Das Zurückliefern des Ergebnisses der Rechnungsprüfung ins dezentrale System kann deswegen mit asynchronen Services realisiert werden. Sie werden dann nur als Informationen im entsprechenden dezentralen System gespeichert, wo es keine Prozesse gibt, die zum Fortfahren auf solche Daten warten.

  Bei der Übertragung der Stammdaten ins zentrale System gibt es ebenfalls keine Zeitbegrenzung. Der Zeitpunkt, wann sie im Data Warehouse unbedingt vorliegen müssen, ist theoretisch unwichtig für das Fortfahren des Rechnungsprüfungsprozesses. Die Verzögerung der Daten bringt nur eine schlechtere Prüfungsqualität und evtl. ein fehlerhaftes Prüfungsergebnis mit sich. Da bei Fehlern die Prüfung erneut durchgeführt wird, kann auch in diesem Fall ein asynchroner Service verwendet werden.

  Anders ist die Situation bei der Übertragung der Bestell- und Bewegungsdaten. Sie müssen bei der Prüfung der Rechnungen rechtzeitig zur Verfügung stehen, denn sonst kann der Prüfungsprozess nicht fortgesetzt werden. Es wird in diesem Fall ein synchroner Service vorgeschlagen.

  Die übertragenen Daten werden aus der Perspektive des Services als Nachrichten betrachtet. Die Übertragung der Nachrichten erfolgt im XML-Format, wobei der Nachrichtenablauf in der Abbildung 24 dargestellt ist.

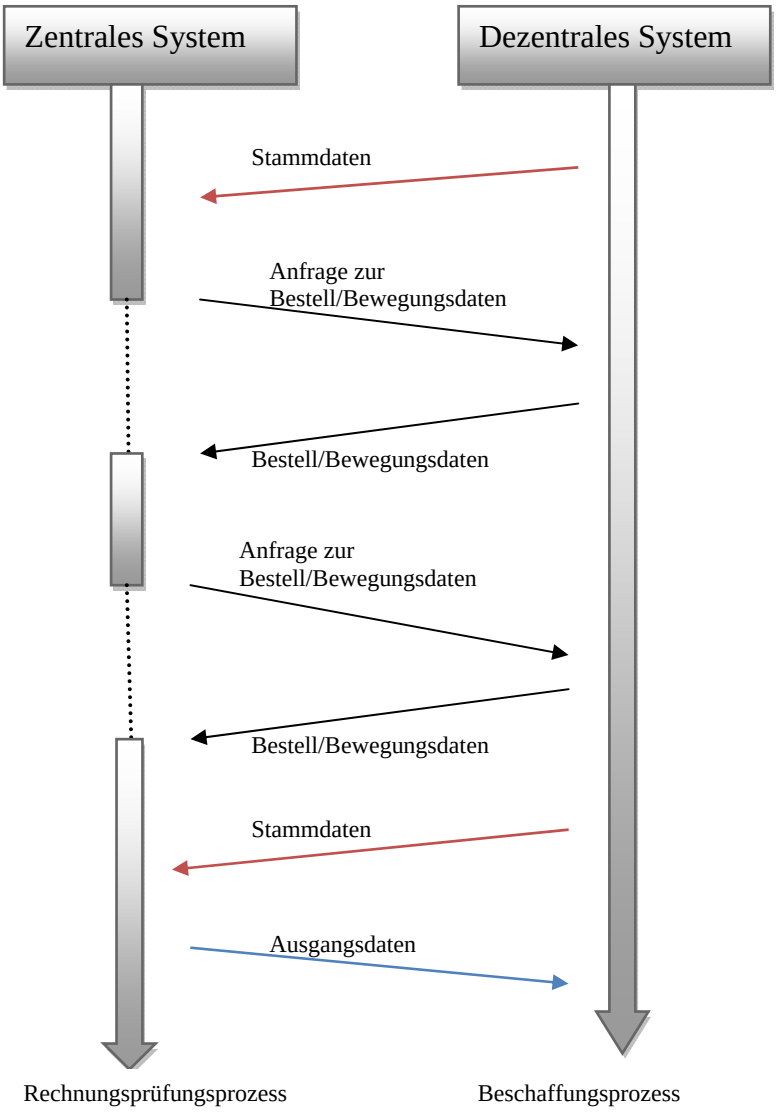

Abbildung 24: Nachrichtenflüsse im Rechnungsprüfungsprozess

6.2.2.2 Implementierung

Ein Prototyp, der einen zentralen Rechnungsprüfungsprozess umsetzt, wurde im Rahmen der Masterarbeit entwickelt und in der SAP-spezifischen Programmiersprache ABAP implementiert. Als Programmierumgebung wurde die ABAP Workbench, eine integrierte grafische Entwicklungsumgebung des SAP-Systems, genutzt. Das Hauptziel des Prototyps ist es zu prüfen, ob eine durch Services realisierte Datenintegration möglich ist, wie sie im entworfenen Integrationskonzept beschrieben wurde.

Es werden drei ERP-Systeme verwendet. Die bestehende Systemlandschaft wird durch die folgende Abbildung veranschaulicht:

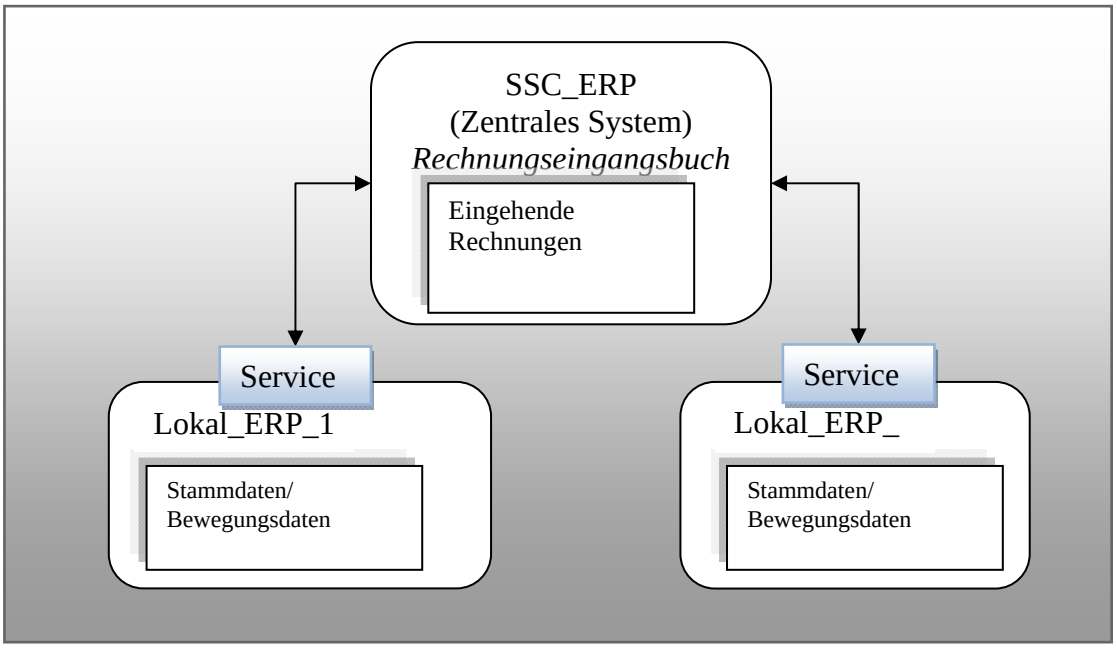

Abbildung 25: Systemlandschaft

SSC_ERP ist das zentrale System, wo der Rechnungsprüfungsprozess stattfindet. Die Systeme Lokal_ERP_1 und Lokal_ERP_2 sind dezentrale logistische Systeme, in denen sich die Bestell-, Bewegungs- und Stammdaten befinden. Der Prototyp wurde im SSC_ERP implementiert und die benötigten Services werden von Lokal_ERP_1 und Lokal_ERP_2 angeboten.

In diesem Prototyp sind die Integrationsmechanismen des zuvor beschriebenen Konzeptes nur teilweise implementiert. Die Übertragung der Stammdaten und die Fehlerbehandlung der übertragenen Daten konnten hier wegen der Einschränkungen des Systems nicht implementiert werden. Das Problem besteht darin, dass bis jetzt noch kein Service in diesem SAP-System existiert, der die Übertragung der Stammdaten übernehmen kann. Es gibt auch noch keinen Service, der zur Fehlerbenachrichtigung zur Verfügung steht. Bei SAP kann im System ein neuer Service jedoch erst nach einem speziellen Überprüfungsprozess verwendet werden, der die Anwendbarkeit des entwickelten Services prüft. Deswegen war es zeitlich nicht möglich, im Rahmen der Masterarbeit neue Services zu implementieren. Aus diesem Grund konnten im Prototyp nur die im System schon vorhandenen Services verwendet werden.

Deshalb wird angenommen, dass die Stammdaten schon vom dezentralen zum zentralen System übertragen wurden und direkt auf die im zentralen System vorliegenden (hier SSC_ERP) Stammdaten zugegriffen. Außerdem wird angenommen, dass es bei Übertragung der Daten vom dezentralen System (hier Lokal_ERP_1 und Lokal_ERP_2) keine Fehler gibt. Die durch Services übertragenen Daten werden direkt im zentralen System bei der Rechnungsprüfung verwendet, ohne zu berücksichtigen, ob ein Fehler auftrat.

Es gibt im ERP-System einen implementierten A2A–Service, dessen Aufgabe es ist, die vollständigen Bestell- und Bewegungsdaten anhand einer bestimmten Bestellnummer zurückzuliefern. Mit Hilfe dieser existierenden Services im System kann die Integration der Bestell- und Bewegungsdaten realisiert werden.

Unter Beachtung der oben genannten Annahmen und Einschränkungen wurde der Prototyp implementiert. Die Bearbeitungslogik des Prototyps wird in der Abbildung 26 dargestellt.

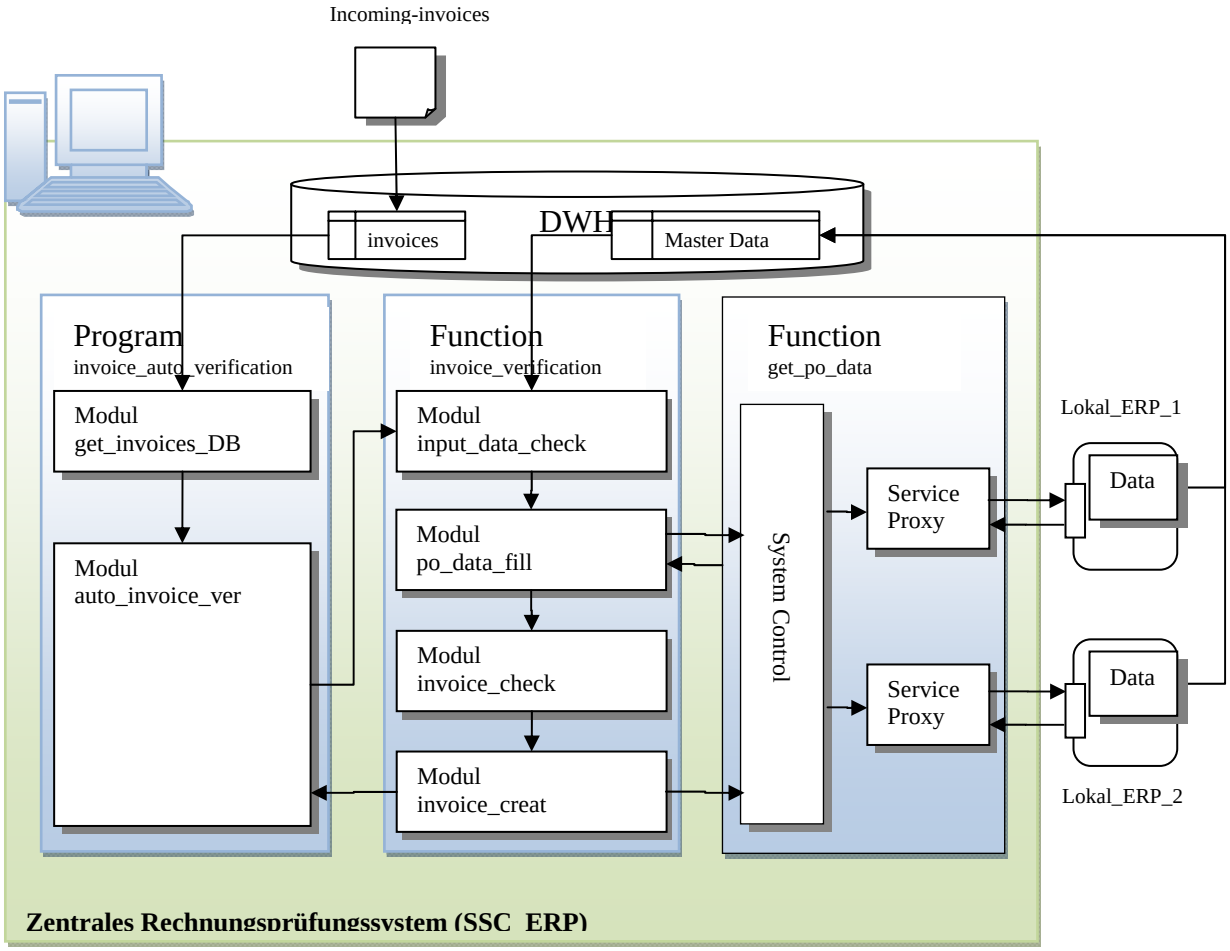

Abbildung 26: Bearbeitungslogik des Prototyps

Die wichtigsten Komponenten sind:

- *Function get_po_data*

  Diese Funktion übernimmt die Aufgaben von System Controller und Communicator. Um die Kommunikation zwischen verschiedenen Systemen zu ermöglichen, muss im zentralen System SSC_ERP für jedes dezentrale System Lokal_ERP_1 und Lokal_ERP_2 jeweils ein Client-Proxy(WSO) angelegt werden. Die Auswahl benötigten Client-Proxies bei der Anfragebearbeitung und der darauf folgende Aufruf des entsprechenden Services werden ebenfalls in dieser Funktion realisiert.

- *Function invoice_verification*

  Diese Funktion implementiert den Rechnungsprüfungsprozess und besteht aus den vier unten beschriebenen Unterprogrammen. Rechnungen werden hier auf ihre Richtigkeit geprüft.

  - *Modul input_data_check* prüft alle vom Data Warehouse importierten Daten auf ihre Existenz und Vollständigkeit.

  - *Modul po_data_fill* übernimmt alle Bestelldaten, die durch die Funktion get_po_data vom entsprechenden dezentralen System abgeholt wurden, und speichert sie temporär in einer internen Tabelle.

  - *Modul invoice_check* ist die Kernfunktion der Rechnungsprüfung. Es prüft die eingehenden Daten auf ihre Richtigkeit anhand der vom Data Warehouse ausgelesenen Stammdaten und den von dem dezentralen System abgeholten Bestell- / Bewegungsdaten.

- *Modul invoice_creat*

  In dem Modul *invoice_create* kann die fehlerfreie Rechnung erfolgreich im System gebucht werden. Fehlerhafte Rechnungen werden auch im System gespeichert und später durch spezielle Prozesse einzeln behandelt. Die Ausgangsdaten, die im Abschnitt 6.2.1.2 beschrieben sind, können ebenfalls durch dieses Modul mit Hilfe des System Controllers und der Services wieder ins entsprechende dezentrale System zurücktransportiert werden. Diese Funktionalität ist aber nicht in dem Prototyp implementiert, da entsprechende Services im SAP-System fehlten.

- *Programm invoice_auto_verification*

  Dieses Programm ist das Hauptprogramm des Prototyps. Es führt den automatisierten Rechnungsprüfungsprozess aus.

6.2.2.3 Bewertung und Erweiterungsmöglichkeiten der Fallstudie

Die Durchführung dieser Fallstudie hat gezeigt, dass das im Kapitel 5 entwickelte Konzept unter bestimmten Annahmen und Einschränkungen in einer realen Systemumgebung anwendbar ist. Die Fallstudie hat aber auch Probleme zutage gefördert, die betrachtet werden müssen.

- Das zentrale Duplikat der Stammdaten bringt zwar eine bessere Performance, da die große Datenmenge nicht oft zwischen Systemen übertragen werden muss, aber die

Aktualisierung aller Stammdaten kann nicht immer rechtzeitig im zentralen System erfolgen.

- Die rechtzeitige Übertragung von Bestell- und Bewegungsdaten liefert zwar immer die aktuellen Informationen, aber sie muss durch einen Service-Aufruf bei jeder Rechnungsprüfung stattfinden. Das führt zu einem großen Performanzproblem.

- Die angefallene Datenmenge war sehr groß.

Es wurde in dieser Fallstudie nur die Kommunikation zwischen drei SAP ERP-Systemen betrachtet. In Unternehmen existieren weitaus komplexere Landschaften. Das dezentrale System kann ein anderes SAP-System, wie z.B. SAP SRM, oder ein nicht-SAP System sein. Es ist aber auch möglich, dass das zentrale System nicht nur für ein Unternehmen verantwortlich ist, sondern als Shared Service Center fungiert, das diese zentrale Aufgabe für mehrere Unternehmen übernimmt. Zwei Erweiterungsmöglichkeiten:für dieses Konzept sind deshalb:

- Von einem ERP-System zu einem anderen SAP-System/nicht-SAP-System

- Von einem Unternehmen intern zu mehreren Unternehmen

6.3 Zusammenfassung

In diesem Kapitel wurde das im Kapitel 5 entwickelte Konzept anhand eines konkreten Anwendungsbeispiels umgesetzt. Die Anwendbarkeit wurde durch einen Prototyp getestet. Dieser Prototyp ist zwar nicht vollständig, aber es ist möglich, ihn in der Zukunft noch zu erweitern.

Durch Implementierung und Testen des Prototyps konnte gezeigt werden, dass das entwickelte Konzept in einem realen System anwendbar ist. Festgestellte Probleme dieses Konzeptes werden im nächsten Kapitel diskutiert und mögliche Verbesserungsvorschläge gegeben.

Kapitel 7

Bewertung des Konzeptes

Anhand der Fallstudie im vorherigen Kapitel wurde das in dieser Arbeit erstellte Konzept in eine reale Anwendung umgesetzt. In diesem Kapitel werden die dadurch aufgedeckten Stärken und Schwächen des Konzeptes gezielt betrachtet.

7.1 Rückblick auf den Entwicklungsprozess

Das Konzept wurde vor dem Hintergrund entwickelt, einen einheitlichen Bearbeitungsprozess in einer zentralen Stelle für die verteilten Datenbestände aus verschiedenen Backend-Systemen zu verwenden. Das vorgestellte Konzept sollte allgemein anwendbar für solche Szenarios sein.

Die Erstellung dieser Arbeit begann mit einer gezielten Analyse der aktuellen Situation des Unternehmens. Dabei haben sich die folgenden Probleme ergeben:

- Viele früher in jeder dezentralen Stelle eines Unternehmens durchgeführten Geschäftsprozesse sollen wegen des starken Wettbewerbsdrucks an eine zentrale Stelle verschoben werden. Aber die dafür benötigen Informationen können nur in den dezentralen Stellen erzeugt werden.

- Die dezentralen Stellen eines Unternehmens, wo die Daten liegen, sind aus historischen Gründen nicht mit einem einheitlichen Anwendungssystem ausgerüstet. Die unterschiedlichen Anwendungssysteme erhöhen die Schwierigkeit bei der Daten-integration.

- Die Datenbestände werden auch auf vielfältige Art und Weise im verteilten System gespeichert, z.B. in Datenbanken oder im Dateisystem.

- Daten ändern sich in unterschiedlichen Zeiträumen. Manchen Daten, wie Stammdaten in SAP-Systemen, bleiben langfristig im System. Manche Daten, wie Bewegungsdaten, werden sehr oft aktualisiert.

Die service-basierte Datenintegration ist ein passender Ansatz, um solche Probleme zu lösen. Die mit diesem Konzept entwickelte Systemarchitektur wurde in einer Fallstudie umgesetzt und deren Verwendbarkeit dadurch geprüft.

7.2 Stärken des erstellten Konzeptes

Eine wesentliche Stärke des Konzeptes liegt in der Verwendung des Services als Integrationsmechanismus. Durch diesen Ansatz können zwei wichtige Vorteile realisiert werden.

- Multi-Backend-fähig
  D.h. die zu integrierenden Datenbestände können aus verschiedenen Systemen kommen, deren zugrundeliegende Plattform und verwendeten Technologien sich unterscheiden. Die lose Kopplung der Services ermöglicht diesen Vorteil.

- Geschäftsprozessorientiert
  Da Services selbst entlang von Geschäftsprozessen „orchestriert“ werden, sind die in Services verpackten Daten auch geschäftsprozessorientiert. Der Vorteil liegt dann darin, dass sie je nach Anforderungen des Geschäftsprozesses einfach verschoben oder zusammengestellt werden können.

Eine weitere Stärke des vorgestellten Konzeptes besteht in der Verwendung des Hybridansatzes für die Datenintegration. Die Kombination des Data Warehouse- und Mediator-Ansatzes ermöglicht eine effiziente und effektive Integration verschiedener Datenbestände. Die Performance des Systems wird durch das Data Warehouse gewährleistet, während der Mediator-Ansatz die Aktualität der Daten sicherstellt. Die zu integrierenden Daten werden in zwei Kategorien unterteilt und je nach ihren Eigenschaften entsprechend behandelt.

Die im Konzept entstandene Systemarchitektur bringt auch einige Vorteile mit sich. Durch die Verwendung der Systemkomponente „Access Controller“ wird die Entscheidung des zu verwendenden Integrationsmechanismus nicht direkt von der Anwendung getroffen. Der Zugriff auf die Daten erfolgt somit transparent. Sie braucht nur die Komponente „Access Controller“ aufzurufen und bekommt dann die benötigen Daten, unabhängig davon, ob sie sich im Data Warehouse befinden oder durch einen Mediator erst vom entsprechenden dezentralen System geholt werden müssen. Diese Architektur ermöglicht ebenfalls eine saubere Trennung von Anwendung und den integrierten Informationssystem und erlaubt somit eine Mehrfachverwendung des integrierten Informationssystems.

7.3 Schwächen des erstellten Konzeptes

Neben den oben aufgelisteten Stärken gibt es auch Schwächen im entwickelten Konzept. Die konzeptbezogenen Schwächen lassen sich hierbei in folgende Punkte zusammenfassen:

- Bei der Verwendung des Data Warehouse-Ansatzes wurden asynchrone Services benutzt. Im Fall des Ausfalls des zentralen Systems ergibt sich das Problem, wohin die Nachrichten dann geschickt werden sollen. In diesem Konzept wurde eine direkte Verbindung zwischen dezentralem System und zentralem System hergestellt, was zu einem möglichen Verlust der zu schickenden Nachrichten führen kann.

- In realen integrierten Systemen sollten normalerweise beim Einsatz von A2A oder B2B keine synchronen Services verwendet werden, da eine zu enge Systemkoppelung entsteht. Wenn ein an der Integration beteiligtes System ausfällt, dann brechen die entsprechenden Geschäftsprozesse ab und viele Ausnahmebehandlungen werden nötig.

Um die erste Schwäche zu überwinden, ist die Verwendung einer Middleware, wie z.B. SAP XI, eine mögliche Lösung. Hier werden die von den dezentralen Systemen verschickten Nachrichten temporär gespeichert und zum passenden Zeitpunkt noch einmal verschickt.

Die Probleme, die bei der Verwendung von synchronen Services auftreten, sind schwer zu beseitigen. Eine mögliche Lösung ist die Ersetzung durch einen entsprechenden asynchronen Service. Die normalerweise durch den synchronen Service abgeholten Daten werden durch den asynchronen Service in einem Puffer gespeichert. Die Anwendung greift dann direkt auf diesen Puffer zu und verwendet die dort abgelegten Daten.

Kapitel 8

Fazit und Ausblick

Das Ergebnis dieser Masterarbeit wird in diesem Kapitel zusammengefasst und ein Ausblick auf mögliche Ansatzpunkte für Weiterentwicklungen gegeben.

8.1 Fazit

Ziel dieser Masterarbeit war es, ein Konzept für die service-orientierte Datenintegration von verteilten Datenbeständen zu entwickeln und in einer praktischen Fallstudie umzusetzen. Dadurch wurde die Umsetzbarkeit dieses Konzeptes geprüft.

Es wurde zuerst anhand eines Unternehmensszenarios der Trend von Unternehmen aufgezeigt, gleichartige standardisierbare Administrationsprozesse in eine übergreifende Organisationseinheit zu verlagern. Ein möglicher Ansatz, um diese Szenarios zu realisieren, basiert auf dem Konzept von Shared-Service-Centern (SSC). Die Einrichtung eines Shared Service Centers ist eine Form der Zentralisierung. Aufgaben, Funktionen oder Tätigkeiten, die bislang in gleicher oder ähnlicher Form an mehreren Stellen im Unternehmen durchgeführt wurden, werden damit an einer zentralen Stelle zusammengefasst.

Eine wichtige Grundlage für die Realisierung eines Shared Service Centers ist die EAI-Technologie. EAI ermöglicht es, Unternehmen mit verschiedensten Plattformen, die entweder auf ERP oder kundenspezifischer Software basieren, eine einheitliche Plattform, wie z.B. SAP ERP Systeme, im Shared Service Center einzusetzten, ohne die bestehenden Plattformen von mehreren Abteilungen und Standorten ändern zu müssen. Der EAI-Ansatz unterstützt die Integration heterogener betrieblicher Anwendungen auf Daten-, Programm- und Prozessebene innerhalb eines Unternehmens und über Unternehmensgrenzen hinweg. Diese Arbeit konzentrierte sich auf die Integration im Sinn der Daten als Integrationsgegenstand. Ein wichtiger Grund dafür ist, dass die Datenintegration einen einfachen aber effizienten Integrationsansatz darstellt, der für das vorgestellte Unternehmensszenario sehr gut geeignet ist.

Das Kernproblem dieser Masterarbeit war die Auswahl und Erweiterung eines geeigneten Ansatzes für die Datenintegration. Der Data Warehouse- und Mediatoransatz sind die zurzeit am meisten verwendeten Datenintegrationsansätze, die beide ihre Vor- und Nachteile haben. Es wurde in dieser Arbeit versucht, eine mögliche Kombination dieser beiden Ansätze zu finden, um den Integrationsvorgang effizienter und effektiver zu machen. Ein hybrider Ansatz wurde dann vor diesem Hintergrund entwickelt. Durch den Data Warehouse-Ansatz wird die Performanz des Datenaustausches gewährleistet und durch den virtuellen Datenintegrations-mechanismus (Mediator-Ansatz) wird gleichzeitig auch die Aktualität der entsprechenden Daten sichergestellt. Anwendungsdaten wurden aus diesem Grund in zwei Kategorien unterteilt. Daten, die sich nicht oft ändern, werden regelmäßig ins zentrale Data Warehouse übertragen und dort als Duplikat gespeichert. Die Anwendungen greifen einfach auf das Data Warehouse zu, um die benötigen Daten abzuholen. Auf der anderen Seite, wenn die Daten sehr oft aktualisiert werden müssen, werden sie nicht im zentralen System physisch gespeichert, sondern durch einen virtuellen Integrationsmechanismus direkt von der entsprechenden externen Datenquelle angefordert. Ein großes Problem war jetzt, wie das System den richtigen Integrationsmechanismus ermitteln kann, um an der richtigen Stelle die benötigten Daten zu finden. Eine Lösung in dieser Masterarbeit besteht darin, dass die Unterteilung der Daten am Anfang durch Customizing schon im System festgelegt wurde.

Eine andere wichtige Aufgabe in dieser Masterarbeit war die Auswahl der passenden Integrationstechnologie, die zur Realisierung der Datenintegration in verteilten Systemen benötigt wird. Die Wahl der Integrationstechnologie fiel hier auf die service-orientierte Architektur. Services bieten eine neue Integrationsmöglichkeit, bei der heterogene Anwendungssysteme integriert werden können, ohne deren verwendeten Betriebssysteme und Programmiersprachen berücksichtigen zu müssen. Die service-basierte Datenintegration ermöglicht eine einfache und flexible Integration von Datenbeständen, die über das Internet verteilt sind. Durch die Bereitstellung der Daten im XML-Format ist deren einfache Weiterverarbeitung durch Anwendungen möglich.

Basierend auf den oben genannten Aspekten wurde als Ergebnis der Masterarbeit ein Konzept entwickelt, das die service-orientierte Integration von Daten aus verteilten Datenbeständen betrachtet. Eine mögliche Systemarchitektur wurde entworfen und die Aufgaben jeder Komponente in dem System genau definiert. Die in diesem Konzept verwendeten Kommunikations- und Integrationsmechanismen wurden ebenfalls festgelegt. Das Konzept sollte allgemein für ähnliche Situationen, wie im Unternehmensszenario beschrieben, geeignet sein.

Die Umsetzbarkeit des im Rahmen dieser Arbeit entwickelten Konzeptes wurde im Unternehmen SAP AG mit einer praktischen Fallstudie untersucht. Dabei wurde für eine zu entwickelnde Anwendung eines Rechnungsprüfungsmonitors im Bereich SAP ERP MM ein Prototyp erstellt. Der Prototyp wurde in der SAP spezifischen Programmiersprache ABAP implementiert. Als Programmierumgebung wurde ABAP Workbench, eine integrierte grafische Entwicklungsumgebung des SAP-Systems, genutzt. Die im Konzept entwickelte Systemarchitektur wurde in diesem Prototyp eingesetzt und die im SAP ERP-System schon

existierenden Services verwendet. Nach Implementierung und Testen des Prototyps konnte gezeigt werden, dass das in Kapitel 5 entwickelte Konzept unter bestimmten Annahmen in einer realen Systemumgebung anwendbar ist. Die Fallstudie hat aber auch Probleme zutage gefördert, die betrachtet werden müssen. Die synchrone Kommunikation zwischen verschiedenen Systemen ist immer als kritisch anzusehen, da der Ablauf der Geschäftsprozesse stark beeinflusst wird. Die Verzögerung oder der Ausfall eines der beteiligen Systeme ist immer der Flaschenhals für den gesamten Prozess. Außerdem ist die Zusammenführung der Daten aus verschiedenen Datenbeständen ins Data Warehouse in zweifacher Hinsicht ein Bereich, in dem verstärkte Forschungstätigkeit notwendig ist. Zum einen ist die Erkennung der redundanten Daten wichtig, zum anderen ist die Frage zu klären, wie oft ein Datenupdate-Zyklus im DWH stattfinden sollte, damit die Systemperformanz nicht zu stark beeinträchtigt und dabei gleichzeitig die Aktualität der Daten im Data Warehouse gewährleistet wird.

8.2 Ausblick

Es existieren einige Erweiterungs- und Verbesserungsmöglichkeiten für das in dieser Masterarbeit entwickelte Konzept zur Datenintegration.

Das Konzept verwendet eine direkte Verbindung zwischen dezentralem und zentralem System. Die durch den Service übertragenen Daten kommen entweder unmittelbar im zentralen System an oder können im Fall, dass das zentrale System ausgefallen ist, möglichweise verloren gehen. Um dieses Problem zu vermeiden, kann eine Middleware verwendet werden. Ein Beispiel dafür ist SAP XI, welche die vom dezentralen System verschickten Nachrichten temporär im Exchange Server speichert und später zu einem passenden Zeitpunkt noch einmal versucht, sie ins richtige System zu verschicken.

Bei der Benutzung von synchronen Services gibt es auch das Problem, dass der zentrale Geschäftsprozess durch fehlgeschlagene Serviceaufrufe stark behindert werden kann. Eine mögliche Lösung ist der Ersatz durch einen entsprechenden asynchronen Service. Die normalerweise durch den synchronen Service abgeholten Daten werden durch den asynchronen Service in einem Puffer gespeichert. Die Anwendung verwendet diesen Puffer und greift direkt auf die dort abgelegten Daten zu. Dieser Ansatz ist ähnlich wie das Data Warehouse. Der Unterschied besteht darin, dass die Daten nicht permanent im zentralen System gespeichert werden, sondern nur temporär im Puffer liegen. Das erfordert in diesem Fall keinen großen Speicherungsaufwand, aber es muss ein Mechanismus festgelegt werden, der entscheidet, wie lange solche temporären Daten im zentralen System bleiben und wie das zentrale System reagiert, wenn die benötigen Daten fehlen.
Außerdem wurden die Mechanismen zur Fehlerbehandlung in dieser Arbeit nicht behandelt. Mögliche Fehlerquellen sind die Datenübertragung, inkonsistente Daten oder Systemausfälle Lösungen dafür sollten in zukünftigen Arbeiten weiter untersucht werden.

Abbildungsverzeichnis

Literaturverzeichnis

[BBBJ+06] Boeder, J., Brendle, R., Buck-Emden, R. Jordt, N. et al.
Enterprise SOA Architecture Positioning
SAP AG (2006)

[BG08] Bauer, A., Günzel, H.
Data-Warehouse-Systeme – Architektur, Entwicklung, Anwendung,
Dpunkt Verlage (2008)

[BARK06] Bark, M.; Wulf, W.
SOA - ein pragmatischer Einstieg.
http://www.evodion.de/opencms/export/evodionIT/Infocenter/resources/evodion_SOA_WhitepaperSOA.pdf
Erstellungsdatum vom 04.04.2006.
letzter Zugriff: Dez. 2008

[CGIH+2002] Cable, S., Galbraith, B., Irani, R., Hendricks, M
Professional Java Web Services
Wrox Press, Chicago, 2001

[DJ04] Dostal, W., Jeckle, M.
Semantik, Odem einer Service-orientierten Architektur
In Java Spektrum 9 (2004) 1, S. 53-56.

[DJMZ05] Dostal, W., Jeckle, M., Melzer, I., Zengler, B.
Service-orientierte Architekturen mit Web Services. Konzept-Standards-Praxis
Verlag: Spektrum

[ESSC03] Enterprise application integration (EAI) in SSC
http://www.sharedxpertise.org/file/916/enterprise-application-integration-eai-in-ssc.html
letzter Zugriff: Jan. 2009

[FISC05]: Fischer, A.
Systematische Übersicht zu aktuellen kommerziellen Softwareprodukten für Datenintegration
Diplomarbiet(2005), Institut für Informatik, Universität Zürich

[FLTW06] Forndron, F., Liebermann, T., Thurner, M., Widmayer, P.
mySAP ERP: Geschäftsprozesse, Funktionalität, Upgrade-Strategie
Verlag: Galileo Press; Auflage: 1 (2006)

[HAMM05] Hammerschall U.
Verteilte Systeme und Anwendungen
Pearson Studium, München 2005.

[HANS05] Hansen H.; Neumann G.
Wirtschaftsinformatik 2. Informationstechnik
9. Aufl., Lucius und Lucius, Stuttgart 2005.

[HEIL07] Heilig L., Karch S.
SAP NetWieaver Master Data Management
Verlag: Galileo Press (2007)

[HK07] Heilig, L., Karch, S.
SAP NetWeaver Master Data Management
Verlag: Galileo Press

[HL04] Häuser, T., Löwer, U.M.
Web Services
Die Standards, Galileo Press, 2004

[HSB+08] Hagedorn, T., Schmid, J., Blume, P. et al
Wissens- und Informationsmanagement in der Praxis – Einführung einer Wissensdatenbank beim Aufbau eines Shared-Service-Centers bei E.ON Energie. In: Keuper, F. und Neumann, F. (Hrsg.), Wissens- und Informationsmanagement
Gabler Verlag, Dezember 2008

[HUL97] Hull, R.
Managing Semantic Heterogeneity in Databases: A Theoretical; Perspective
Proceedings of the Sixteenth ACM SIGACT-SIGMODSIGART; Symposium on Principles of Database Systems,
Tucson, Arizona, 12.-14. Mai 1997.

[IHKS05] IHK Saarland (2005)
Rechnung. Erklärung der allgemeinen Rechtsfragen der IHK Saarland.

[KAIB02] Kaib, M.
Enterprise Application Integration: Grundlagen, Integrationsprodukte, Anwendungsbeispiele
Deutscher Universitäts-Verlag; Auflage: 1. Aufl. (November 2002)

[LN06] Leser, U. und Naumann, F.
Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen
Dpunkt Verlage; Auflage: 1. A. (Oktober 2006)

[NIEW05] Nieweglowski, K.
eSOA Introduction: Web Service handover to IMS
Vortrag (2005) SAP AG

[SB01] SAP Bibliothek (2006)
SAP NetWeaver
http://help.sap.com/saphelp_glossary/de/04/baeb76337cbd48b9e0a5ecb7f47549/content.htm
letzter Zugriff: Jan.2009

[SB02] SAP Bibliothek (2008)
Proxy
http://help.sap.com/saphelp_glossary/de/d5/4f2686a6034c4d913f1621975226e6/content.htm
letzter Zugriff: Jan.2009

[SB03] SAP Bibliothek (2008)
SAP Exchange Infrastruktur
http://help.sap.com/saphelp_nw04/helpdata/de/0f/80243b4a66ae0ce10000000a11402f/content.htm
letzter Zugriff: Nov.2008

[SLRP01] SAP AG (2001)
Logistik-Rechnungsprüfung.pdf

[SCHA03] Schahram D., et al.:
Software-Architekturen für Verteilte Systeme
Springer, Berlin 2003

[SCHM05] *Web Service-basierte Referenzarchitektur für Enterprise Application Integration*
Wissenschaftlicher Verlag Berlin; Auflage: 1 (August 2005)

[STRN06] Strnadl, C.
Was bedeutet der SOA-Hype in der Praxis für das Unternehmen
http://www.bpm-guide.de/articles/40
letzter Zugriff: Oct.2008

[WIEH04] Wiehler G.
Mobility, Security und Web-Services
Publicis Corporate Publishing, Erlangen 2004.

[WM05] Woods, D., Mattern, T
Enterprise SOA
O'Reilly, 2006.

[WOOD04] Woods, D.
Enterprise Service Architecture: SAPs Bauplan für Geschäftsapplikationen der nächsten Generation
Verlag: Galileo Press 2004

[W3C01] *W3C: Web Services Glossary*
http://www.w3.org/TR/ws-gloss/#defs
Version: 2004.
letzter Zugriff: 11.2008

[W3C02] *W3C: Web Services Description Language (WSDL) 1.1.*
http://www.w3.org/TR/wsdl
Version: März 2001.
letzter Zugriff: 11.2008

[W3C03] *W3C: Simple Object Access Protocol (SOAP) 1.2.*
http://www.w3.org/TR/soap/
Version: April 2007
letzter Zugriff: 11.2008

[W3C04] *W3C: Extensible Markup Language (XML) 1.0.*
http://www.w3.org/TR/xml
Version: November 2006
letzter Zugriff: 11.2008