

Komponentenbasiertes EAI-Framework unter Einsatz und Erweiterung von Web Services

Marten Schönherr, Björn Eric Gallas

Technische Universität Berlin

Zusammenfassung: Der Beitrag beschreibt die Möglichkeiten der im Konzept von Webservices verwendeten Standards und Architekturspezifikationen zur Unterstützung von Systemintegrationsprojekten. Probleme, die durch proprietäre EAI-Standardsoftware entstehen, können durch Webservice- und komponentenbasierte Architekturen teilweise oder ganz behoben werden. Dafür werden Grundlagen, fachliche Anforderungen und Ziele von EAI-Projekten kurz beschrieben. Den technischen Standards des Webservice-Konzeptes folgen notwendige Erweiterungen der Architektur, um integrative Aspekte unterstützen zu können. Der Beitrag schließt mit einem Ausblick auf notwendige Entwicklungen und zukünftige Potentiale, die durch eine umfassende Verwendung von Webservices in EAI-Projekten entstehen.

Schlüsselworte: EAI, Webservice, Component Ware

1 EAI – Enterprise Application Integration¹

Ziel des Beitrages ist die Beschreibung einer Systemarchitektur auf Basis von Webservices, die den komplexen Anforderungen einer Integrationsplattform entspricht. Zur Erarbeitung der wichtigsten Punkte werden zuerst die Grundlagen der sogenannten Enterprise Application Integration dargestellt.

1.1 Definition EAI

EAI steht derzeit für einen neuen Trend in der Informationsverarbeitung. Trends bedeuten nicht immer tatsächlich neue Inhalte. So auch in diesem Fall. Hinter dem Schlagwort EAI verstehen die Autoren dieses Beitrages Konzepte zur Integration

¹ EAI wird im Folgenden im Sinne von Enterprise Application Integration verwendet.

heterogener DV-Infrastrukturen. Systemintegration wird in der Wirtschaftsinformatik als „ingenieurwissenschaftlich orientierte Vorgehensweise“ [KuRa96, S. 277] beschrieben, um die betriebliche Realität möglichst genau über IT-Systemgrenzen hinaus abzubilden. Dabei umfasst der verwendete Begriff sowohl Methoden als auch Systeme [vgl. Mer97, S. 208]

KELLER beschreibt mit EAI Softwaresysteme, die es erlauben, verschiedene Applikationen eines Unternehmens zu integrieren. Er grenzt EAI von E-Commerce ab, da hier in der Regel mehrere Parteien beteiligt sind. Allerdings werden die Konzepte strukturell und technisch als identisch erklärt. Der einzige Unterschied besteht in der unternehmensübergreifenden Sichtweise des E-Commerce-Ansatzes und des unternehmensinternen Fokus von EAI [Kel02, S. 5 ff.].

Bezug nehmend auf MYERSON kann man EAI als umfassendes Konzept betrachten, dass folgende Schwerpunkte beinhaltet:

EAI “involves a complete system of business processes, managerial practices, organizational interactions and structural alignments... It is an all inclusive process designed to create relatively seamless and highly agile processes and organizational structures that are aligned with the strategic and financial objectives of the enterprise ... Systems integration represents a progressive and iterative cycle of melding technologies, human performance, knowledge and operational processes together.” [My02, S. 5]

Betrachtet man entgegen der Definitionen die Produkte, die derzeit am EAI-Markt angeboten werden, kommt man zu einem einfachen Verständnis des Begriffes: EAI sind Werkzeuge und Services, mit denen die mühsame Aufgabe der Integration heterogener Systeme erleichtert wird. [Gul02]

1.2 Grundlagen der Anwendungsintegration

Seit langem kennen vor allem große Unternehmen die Relevanz der Applikationsintegration. Besonders der Kostenfaktor in den Bereichen Systementwicklung und Systembetrieb spielt hier eine bedeutende Rolle. Die Herausforderung der Integration immer komplexer werdender IT-Systemstrukturen besteht bereits seit vielen Jahren für die IT-Abteilungen vieler Unternehmen. Umfangreiche Individualentwicklungen in Unternehmen waren die Folge: Es entstand die sogenannte Point-to-Point Integration. Die folgende Abbildung zeigt eine typische historisch gewachsene IT-Landschaft mit einer Vielzahl individueller Schnittstellen:

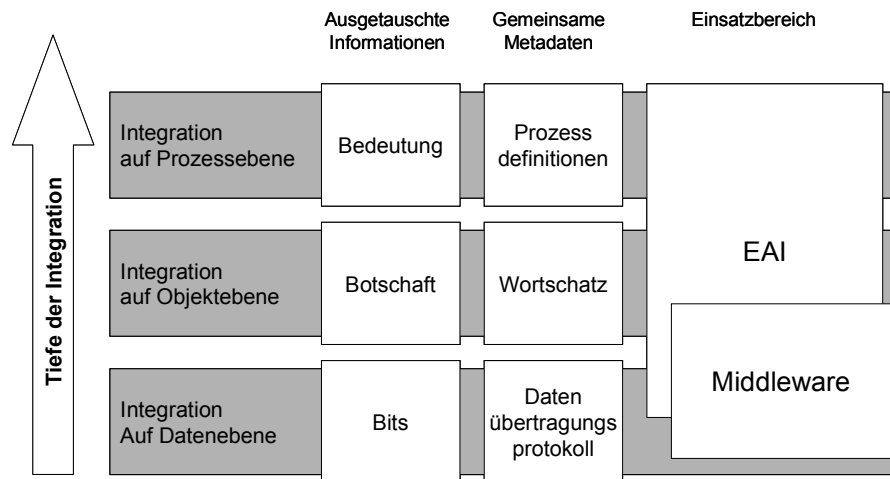


Abbildung 2: Integrationsarchitektur nach RING [Ring00, S. 26]

1.3 Integrationsformen und –merkmale

Sowohl individuelle als auch Standardsoftwaresysteme sollen so miteinander verbunden sein, dass betriebswirtschaftlich relevante Daten prozessorientiert ausgetauscht werden können. Die Syntax und Semantik der Daten in den operativen Systemen wird dabei nicht verändert. Zur Realisierung dieser Grundelemente werden neben den existierenden operativen Systemen zusätzliche Architekturen aufgebaut, die entweder individuell entwickelt oder als Produkt erworben und den individuellen Integrationsanforderungen angepasst werden. Neben diesen technisch anmutenden Infrastrukturmaßnahmen gehen Integrationsprojekte immer mit Prozessveränderungen einher. Die Betrachtung von Geschäftsprozessen, die systemübergreifend stattfinden, bildet die Grundvoraussetzung für EAI-Projekte.

In der Regel sind zumindest in komplexen DV-Systemwelten bereits viele Integrationsaspekte zu finden. In diesem Zusammenhang folgen kurz verschiedene Formen von Systemintegration und die Beschreibung des Integrationsgrades einer bereits bestehenden DV- Infrastruktur.

1.3.1 Ex-ante-Integration vs. Ex-post-Integration

Bei der allgemeinen Betrachtung von Systemintegration kann man drei grundlegende Unterscheidungen bezüglich des Zeithorizontes der Integration vornehmen: [KuRa96, S. 169]

- Ablösung mehrerer Altsysteme durch Entwicklung eines kompletten Neusystems
- Verwendung integrationsfähiger Software-Komponenten
- Nachträgliche Integration vorhandener Systeme

Der erste Ansatz entspricht einer so genannten ex-ante-Integration. Die Integration findet vor der Implementierung des Systems in das Unternehmen statt. Typisches Beispiel ist die Ablösung von Altsystemen durch die Einführung von ERP-Systemen³ wie SAP R/3. Die dritte Alternative wird als ex-post-Integration bezeichnet und beschreibt den Hintergrund für die meisten klassischen EAI-Projekte. [Kaib02, S. 20 f.]

Interessant und keinem der beiden Zeithorizonte eindeutig zuzuordnen ist die Integration durch den Einsatz komponentenbasierter Systeme. Die bereits seit langem diskutierte Entwicklung integrativer Einzelkomponenten⁴ lässt sowohl die Integration neuer Funktionen (ex-ante) als auch die integrierte Weiterverwendung existierender Systeme (ex-post) zu. Diese werden gekapselt (wrapping) und dann ebenfalls als Komponente betrachtet. [Wil99, S. 50 ff.]

1.3.2 Integrationsgrad heterogener Systemlandschaften

Betrachtet man die gesamte vorhandene IT-Infrastruktur und die kurzfristig geplanten neu einzuführenden Applikationen als ein System im Sinne der Systemtheorie, kann man diesem Eigenschaften zuordnen. Die herausragendste Eigenschaft derzeitiger Systemlandschaften ist die Heterogenität in Bezug auf die Vielzahl an unterschiedlichen zugrunde liegenden Betriebssystemen, Datenhaltungskonzepten, Entwicklungsumgebungen- und -sprachen, Netzwerkeigenschaften und Hardwarespezifikationen. [Kaib02, S. 13]

Man versucht, den Grad der bereits erreichten Integration eines Gesamtsystems zu ermitteln bzw. zu beschreiben. Der Ansatz zum Integrationszustand eines IT-Systems nach LINß lässt sich für ein betrachtetes Gesamtsystem anwenden. LINß benutzt basierend auf Betrachtungen mehrerer Autoren⁵ die drei Dimensionen Integrationsgegenstand, -reichweite und -richtung als Beschreibungskriterien für den Fokus bzw. den Grad der Integration. [Linß95, S. 18 ff.] Der Integrationsgegenstand aus dem Modell nach LINß überschneidet sich mit dem oft verwendeten Begriff der Integrationstiefe. Sie sind Maße für den Grad und die Ebene auf die sich die Integration erstreckt. Es ergeben sich die folgenden Integrationsebenen:⁶

³ ERP- Enterprise Resource Planning

⁴ dazu mehr u.a. bei: [Zei00, S. 60 ff.] und [Rau01, S. 47 ff.]

⁵ Hier werden vor allem Ferstl/Sinz, Schumann und Mertens genannt.

⁶ u.a. [Lin00, S. 23 ff.] und [Kaib02, S. 17 ff.]

- Daten
- Funktionen/ Objekte/ Komponenten
- Prozesse

Integration auf Datenebene beruht auf der Nutzung gemeinsamer Daten durch verschiedene Anwendungen. Dabei greifen die Systeme physikalisch nicht auf gemeinsame Daten zu, jede Anwendung benutzt proprietäre Datenquellen. Der Einsatz von Schnittstellen oder Datentransferprotokollen sorgt für die Übertragung der Daten. EAI auf Datenebene stellt die geringste Integrationstiefe dar. SCHEER unterscheidet vier Datenintegrationsgrade: manuelle Datenweitergabe, automatische Weitergabe über Schnittstellen, Zugriff auf eine definierte gemeinsame Datenbasis und übergreifende Abbildung aller Unternehmensdaten in einem Metadatenmodell [Scheer90, S. 164 ff.]. Der Integrationsgrad steigt, je mehr Daten unabhängig von Ihren Stammsystemen verwendet werden können.

Bei der Integration auf Funktions- bzw. Objektebene kommunizieren Anwendungen kontrolliert miteinander. Es werden Funktionen bzw. Methoden in den Objekten anderer Systeme aufgerufen und übergreifend verwendet. Um diese Integrationsebene zu realisieren, werden u.a. APIs⁷ eingesetzt. Systeme müssen z.T. verändert werden, wenn keine standardisierter Interfaces zur Verfügung stehen. Auf einer höheren Abstraktionsebene bezeichnet man Programme bzw. Programmteile hier auch als Komponenten. Durch Kapselung und der standardisierten Kommunikation zwischen den gekapselten Komponenten kann eine sehr weitreichende Integration realisiert werden, die nicht mehr zwischen den originären Systemen unterscheidet, sondern die jeweils richtige Komponente verwendet. [MeGri00, S. 3] Je mehr Funktionen, Objekte bzw. Komponenten unabhängig von den DV-Systemen und deren Arbeitsumgebungen, in denen sie verankert sind, verwendet werden, desto höher ist der Integrationsgrad in dieser Ebene.

Bei der Prozessintegration liegen abgebildete Geschäftsprozesse nicht mehr in Einzelsystemen, vielmehr werden Prozesse unabhängig von operativen Systemen in einer EAI-Prozessebene unterstützt. Hieraus werden dann, gemäß dem definierten Workflow, die Dienste der einzelnen Anwendungen gesteuert und aufgerufen. Dabei steht die prozessorientierte Funktionsintegration und die durchgängige Unterstützung von Teilaufgaben eines Gesamtprozesses im Vordergrund. [Fer92, S. S. 14 ff.] Je durchgängiger Geschäftsprozesse unterstützt werden, ohne dass dabei die Verwendung spezifischer Systeme beachtet werden muss, sondern die Anforderungen des Prozesses, desto höher ist der Integrationsgrad.

⁷ API (Application Program Interface) - eine API dient der Kommunikation zwischen verschiedenen Programmen bzw. Programmteilen. Sie legt fest, wie die Programme miteinander zu kommunizieren haben. Standardisierte API's sind die Voraussetzung dafür, dass auch Produkte unterschiedlicher Hersteller miteinander arbeiten können.

Die Begriffe der Integrationsreichweite und der Integrationsbreite werden oft synonym verwendet. Die sogenannte Integrationsbreite bestimmt sich durch die Anzahl der Anwendungen, die integriert werden sollen. Sind es nur wenige, eng zusammengehörige Systeme, so spricht man von einer geringen Integrationsbreite. Wird eine ganze Wertschöpfungskette integriert, ist die Integrationsbreite groß. Mit der Integrationsbreite steigt die Komplexität des Integrationsprojektes. Diese Abgrenzung beachtet neben der rein quantitativen Betrachtung auch die Unternehmensgrenzen. Eine innerbetriebliche Application to Application Integration wird unterschieden von einer unternehmensübergreifenden Anwendungsintegration. [Linß95, S. 25]

Mit dem Begriff der Integrationsrichtung beschreibt KAIB den Grad der Integration bezogen auf die jeweilige Ausprägung der Integration in einer Organisationsstruktur. [Kaib02, S. 18 f.] Dabei wird zwischen horizontaler und vertikaler Integration differenziert. Der Grad der horizontalen Integration steigt mit der Anzahl der beteiligten Fachabteilungen bzw. Organisationseinheiten [Linß95, S. S. 23]. Bei der horizontalen Integration unterscheidet MERTENS zwischen der Anzahl der integrierten Planungs- und Kontrollsysteme aus den Administrations- und Dispositionssystemen [Mer00, S. 4].

1.4 Integrationsziele

Ziel von Integrationsprojekten ist vor allem die Steigerung des Nutzens beim Einsatz von IT. Diese Nutzenpotentiale werden in strategische und operative unterschieden. Der operative Nutzen von Systemintegration kann zum einen in den Fachabteilungen und zum anderen in der IT-Abteilung betrachtet werden. [Kaib02, S. 25f.] In der folgenden Tabelle werden diese Potentiale anhand der Kriterien Kosten, Zeit und Qualität aufgezählt:

Nutzen	Nutzenpotential in Fachbereichen	Nutzenpotentiale in IT-Abteilung
Kosten	<ul style="list-style-type: none"> • Daten werden einmal eingegeben • Papierverbrauch wird reduziert • geringere Übertragungskosten 	<ul style="list-style-type: none"> • Wiederverwendung von Komponenten • Punkt-zu-Punkt Schnittstellen verringern • Keine Veränderung von Altsystemen
Zeit	<ul style="list-style-type: none"> • Beschleunigung von Abläufen • Verringerung von Medienbrüchen • Online-Verfügbarkeit von Daten 	<ul style="list-style-type: none"> • Modifikation durch ex-post-Integration von Komponenten • Administrative Vorgänge werden beschleunigt
Qualität	<ul style="list-style-type: none"> • Redundanz durch Mehrfacheingabe • Erhöhung der Datenqualität 	<ul style="list-style-type: none"> • validierte Komponenten von Altsystemen können genutzt werden • durch Prozessredesign werden Fehler aufgedeckt

Tabelle 1: operative Nutzenpotentiale als Integrationsziele [Sche97, S. 7]

Neben den operativen Zielen steht vor allem die strategische Ebene bei der Systemintegration im Vordergrund. SCHILL beschreibt die folgenden Schwerpunkte [Schill00, url]:

- Flexibilisierung der Organisationsstrukturen durch nachhaltig anpassbare IT-Infrastrukturen
- Umfassende Kosteneinsparpotentiale bei Merger & Akquisition-Projekten
- Kosteneinsparungen durch Einführung von Standards
- Investitionsschutz in IT-Infrastruktur durch langfristige Verwendung von Systemen in einer Integrationsumgebung
- Steigerung der Integrationsfähigkeit für die Integration weiterer Systeme

Im folgenden soll untersucht werden, ob die beschriebenen Ziele durch den Einsatz von Webservices ganz oder z.T. erreicht bzw. unterstützt werden können.

2. Komponentenbasierte Anwendungsintegration mittels Web Services

Die heutigen EAI Tools bzw. Architekturen weisen zum Teil erhebliche Nachteile auf. Sie sind meist teure, komplexe, proprietäre (Insel-)Lösungen. [Samt01] Eine Interoperationalität zwischen den EAI Systemen von verschiedenen Herstellern ist meist nicht gegeben. Die höchste Stufe der Integration, die Prozessintegration, wird zu meist über proprietäre Lösungen realisiert. Aspekte komponentenorientierter EAI Architekturen, die eine evolutionäre Integration sowie eine Front End Integration ermöglichen, werden bei den aktuellen Diskussionen um die Möglichkeiten der Systemintegration selten beachtet.

Im folgenden soll diskutiert werden, inwieweit Web Services für ein komponentenbasiertes EAI Framework geeignet sind, bzw. wo aus heutiger Sicht Entwicklungs- und Forschungspotential liegt, um die oben skizzierten Mängel zu umgehen.

Unternehmungen haben unterschiedliche Auffassungen bzw. Erwartungen an Web Services. Im folgenden sind einige Beispiele aufgeführt: [Mied01]

„Web-Services sind nicht mehr und nicht weniger als ein Weg, um die Anwendung-zu-Anwendung-Integration zu standardisieren.“ (Adam Bosworth, Bea)

„Web-Services sind ein reiner Schnittstellen-Mechanismus. Dadurch ist es möglich, eine einfache Schnittstelle für komplexe Geschäftsprozesse innerhalb eines Unternehmens bereitzustellen.“ (Mike Gilpin, Giga)

2.1 Definition Webservice

Web Services sind verteilte, lose gekoppelte und wiederverwendbare Software-Komponenten, auf die über Standard-Internetprotokolle zugegriffen werden kann [Cha⁺02].

Ein Web-Service ist demnach eine Komponente, die als Softwareeinheit eine definierte Schnittstelle implementiert. Bei lokalen Anwendungen bietet das Komponentenmodell Vorteile bezüglich Wiederverwendbarkeit und Flexibilität. Durch die Erweiterung auf die Netzwerkebene werden die Vorteile des Komponentenmodells mit denen der Verteilung, wie z.B. zentrale Verwaltung und Bereitstellung von Informationen, verbunden. Im Gegensatz zu Middleware Technologien findet bei dem Web Service Konzept eine lose Kopplung von Komponenten statt. Web Services bieten somit eine einfache und erweiterbare Integrationsschicht. Die Anwendung von Legacy-Systemen erfordert die Implementierung von Web Services Adaptern [Ham01].

Web Services sind über Schnittstellen zugänglich und bieten gewisse Funktionen (Service) an. Aufgrund der abgekapselten Art und der in Anlehnung an SZYPERSKI definierten Komponenteneigenschaften können sie als Komponente betrachtet werden [Szyp97, S. 34]. Ein Empfänger setzt neue Web Services oder komplexe Anwendungen aus einer Menge von einzelnen Web Services zusammen (Orchestration).

Die Basisarchitektur der Web Services ist ein Dreiecksverhältnis zwischen einem Dienste-Anbieter (Service Provider), einem Dienste-Nachfrager (Service Requester) und einem Dienste-Vermittler (Service Registry, oft auch als Service Broker bezeichnet) [Cha⁺02]⁸.

Der Service Provider veröffentlicht (publish) die Beschreibung des von ihm angebotenen Dienstes bei einem Service Registry. Bei diesem wird der Dienst in ein System von Kategorien eingeordnet. Mit Hilfe dieser Kategorien und weiterer Suchmechanismen macht ein Service-Requester einen benötigten Dienst ausfindig (find) und fordert von der Service-Registry die Informationen an, wie dieser Dienst zu nutzen ist. Schließlich kommuniziert der Service-Requester mit dem Service-Provider entsprechend den in der Dienstbeschreibung angegebenen Protokollen und Schnittstellen (bind).

⁸ vgl. hierzu Abbildung 3

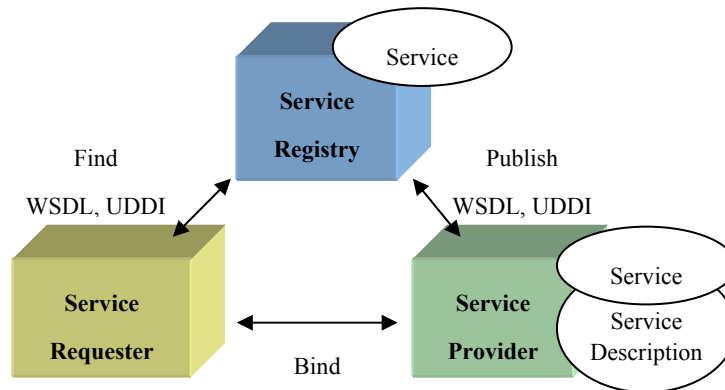


Abbildung 3: Service Orientierte Architektur (SOA) [Cha⁺02] [Kre⁺01]

Das dem Zusammenspiel zugrunde liegende Modell der Beteiligten wird auch als Service-Oriented Architecture (SOA) bezeichnet [Gis01] [Cha⁺02]. Dieses Zusammenspiel kommt unter Anwendung von Basisstandards zu Stande, d. h.:

- die Kommunikation von Anwendungen mit Webservices und Webservices untereinander,
- die standardisierte Beschreibung von Webservices
- und das globale Verzeichnis zum Auffinden von Services und Service-Providern.

2.1.1 Simple Object Access Protocol (SOAP)

Das Simple Object Access Protocol (SOAP) findet in zwei Kommunikationsmodellen Verwendung. Die erste Variante ist die nachrichtenorientierte Kommunikation, die die Übertragung von beliebigen XML-Dokumenten (eXtensible Markup Language) ermöglicht [Bray00]. Darauf baut die zweite Variante auf, mit der sich ein entfernter Methodenaufruf (remote procedure call, RPC) realisieren lässt [SOAP01] [RPC92]. Hierfür wird festgelegt, dass eine XML-Nachricht bestimmten Formats die Anfrage darstellt, worauf eine Antwortnachricht folgt, die das Ergebnis oder eine Fehlermeldung enthält. Mit dem Ziel einer langfristigen und flexiblen Verwendung des SOAP-Protokolls, wird kein bestimmtes darunterliegendes Transportprotokoll vorgeschrieben, sondern es kann auf verschiedene Protokolle aufsetzen. Üblicherweise wird das Hypertext Transfer Protocol (HTTP) verwendet, aber es können auch alternativ andere Protokolle wie das Simple Mail Transfer Protocol (SMTP) oder Java Message Service (JMS) zum Einsatz kommen. SOAP bietet einen einfachen und erweiterbaren Mechanismus, um Daten zwischen verteilten Anwendungen auszutauschen. Dies erfolgt mit Hilfe so genannter SOAP-Nachrichten, die auf

XML basieren und einen elektronischen Austausch von Dokumenten ermöglichen. Eine Kommunikation zwischen Anwendungen mittels SOAP erfolgt plattformunabhängig. SOAP ist für den Austausch von Daten jeglicher Art verwendbar.

2.1.2 Web Service Description Language (WSDL)

Die Web Services Description Language (WSDL) ist ein XML basierter Standard [Chris⁺01]. Dieser Standard spezifiziert eine Syntax zum technischen Aufruf eines Web Services. Ein Anbieter von Web Services beschreibt einheitlich durch die Verwendung von WSDL, welche Art von Dienst er anbietet. Ein Interessent beschreibt ebenfalls mittels dieses Standards, wonach er sucht. WSDL beschreibt sowohl die Operationen, welche Web Services ausführen bzw. ermöglichen sollen als auch die einzelnen Parameter, die jede Operation akzeptieren und wiedergeben.

2.1.3 Universal Description, Discovery and Integration (UDDI)

Ein Standard für Verzeichnisse ist die Universal Description, Discovery, and Integration (UDDI) [Bell⁺01]. Er spezifiziert Schnittstellen, über die Informationen abgelegt und abgefragt werden können, sowie Formate zur Verwaltung dieser Informationen. UDDI dient zur Dienstbeschreibung und Auffindung als universeller Verzeichnisdienst für Dienstleistungen. Der Aufbau eines Web Services besteht aus den angesprochenen verschiedenen Schichten, die auch als Web Services Stack bezeichnet werden [Kre01].

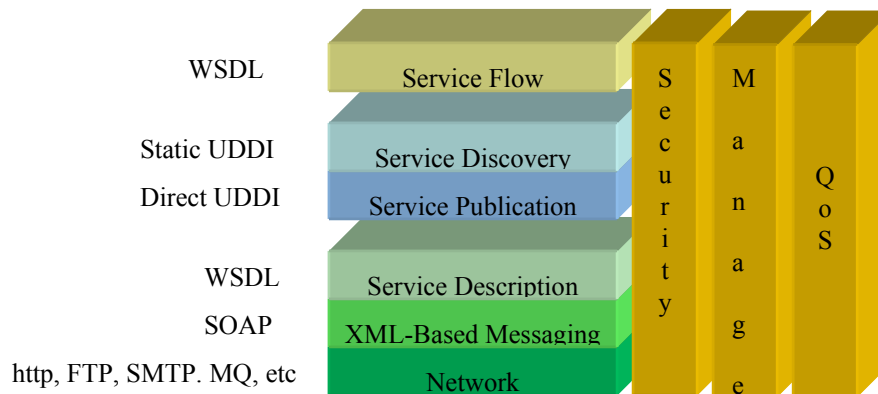


Abbildung 4: Der Web Service Stack

Dieser Stack beinhaltet die Basistechnologien und Anforderungen an die einzelnen Schichten (Security, Management und Quality of Services). Im folgenden soll nun dieser Stack um wesentliche Aspekte ergänzt werden, die zu einem

komponentenbasierten EAI Framework auf der Basis von Web Services führen können.

2.2 Web Services als Basis für ein EAI Framework

Webservicetechnologien bieten sich prinzipiell sowohl für unternehmensübergreifende B2B-Integration über das Internet an als auch für die unternehmensinterne EAI an. Die zum jetzigen Zeitpunkt verfügbaren Webservicetechniken ermöglichen nur eine Integration auf funktionaler Ebene. Neben den Basisdefinitionen entstehen mittlerweile viele ergänzende Festlegungen für Webservices. Nur durch eine Erweiterung des Web Services Stacks wird es in der Zukunft möglich sein, eine Integrationsstrategie unter Zuhilfenahme von Web Services für die Bereiche Anwendungen (Applikation-to-Application), Business-to-Business oder Business-to-Customer zu entwerfen. Im folgenden soll aber die Anwendungsintegration im Vordergrund stehen.

Konzepte für automatisierte Prozessabläufe fehlen, ebenso gibt es bisher noch kein standardisiertes Konzept für die Sicherheit, die Quality of Service und die Transaktionsunterstützung [Höf01] [Dun03, S.16-17]. Die Realisierung dieser Funktionalitäten sind eine wichtige Voraussetzung zur Erfüllung der Anforderungen eines EAI Frameworks. Hinzu kommt ein Komponentenmodell, für die Kapselung eines User Interfaces.

2.2.1 Geschäftsprozessmodellierung

Durch die Anwendungsintegration soll das Zusammenspiel mehrerer Applikationen in einem Unternehmen an einem gemeinsamen Geschäftsprozess ohne den Eingriff des Benutzer ermöglicht werden. Hierzu ist es erforderlich, den entsprechenden Geschäftsprozess zu standardisieren und exakt zu beschreiben.

Dieser Zusammenhang wird als Orchestration von Webservices, also dem Zusammensetzen von einzelnen Webservicebausteinen zu einer komplexen Anwendung bezeichnet. Eine Geschäftsprozessbeschreibung kann als Ansammlung einzelner Aktivitäten betrachtet werden, die durchgeführt werden müssen, um das Ziel des Geschäftsprozesses zu erreichen. Die Beschreibung muss Auskunft darüber geben, welche Aktivitäten durchzuführen sind, in welcher Reihenfolge, in welchen Abhängigkeiten die einzelnen Aktivitäten zueinander stehen, und wer für welche Aktivität verantwortlich ist.

Es existiert kein gemeinsamer Standard für die Orchestration bzw. die Geschäftsprozessmodellierung. Es existieren zahlreiche Vorschläge zur Beschreibung von Geschäftsprozessen, so z.B. die Web Services Flow Language (WSFL). Die Web Service Flow Language ermöglicht die Komposition eines neuen Web Service aus einer Menge bereits vorhandener Web Services [Ley01].

Ein weiterer Ansatz ist die XLANG – Web Services for Business Process Design. XLANG läßt die Automation von auf Webservices basierenden Geschäftsprozessen durch Definition einer Notation für den Nachrichtenaustausch zu [That01]. Zuletzt ist noch die Electronic Business eXtensible Markup Language (ebXML) in diesem Kontext zu nennen. Sie soll eine offene, auf XML basierende Infrastruktur für interoperables Durchführen von Electronic Business zur Verfügung stellen [EbXML03].

2.2.2 Sicherheit

Die Sicherheit, die an dieser Stelle nur kurz skizziert werden soll, wird beispielsweise auf der Protokollschicht über eine sichere https Verbindung realisiert; auf der höheren Anwendungsschicht stehen Standards wie XML Signature und XML Encryption zur Verfügung [Rea02] [Bar⁺02].

2.2.3 Unterstützung von Transaktionen

Für die Unterstützung der Geschäftsprozessabwicklung unter Verwendung der Web Services Technologie sind auch Anforderungen an die Unterstützung von Transaktionen notwendig. Das *Business Transaction Protocol* [Btp02] ist eine Spezifikation des *Business Transaction Technical Committee*, das sich unter dem Dach der Organisation OASIS befindet. Zweck des BTP ist die Transaktions-Koordinierung in verteilten Anwendungen, an der mehrere Anwendungskomponenten unabhängiger Organisationen beteiligt sind. Als alternatives Protokoll ist hier die *Transaction Authority Markup Language* (XAML) zu nennen[Xaml00].

2.3 Erweitertes Komponentenmodell

Um eine Anwendungsintegration auf Basis von Web Services zu ermöglichen, ist es erforderlich, zusätzliche Informationen wie beispielsweise die Beschreibung des Benutzerinterfaces zusammen mit der Schnittstellenbeschreibung kapseln zu können.

Auch hier gibt es unterschiedliche Ansätze. Die WSUI-Spezifikation (Web Services User Interface) beschreibt die Web Services als komplexe Endnutzer-Anwendungen, die einfach in Webseiten eingebettet werden können [WSUI01]. Einen komplexen Ansatz bildet die Web Services Experience Language (WSXL), ein Komponentenmodell zur Entwicklungsunterstützung wiederverwendbarer Web-Applikationen, mit dessen Hilfe visuelle und interaktive Schnittstellen entworfen und rekombiniert werden können [WSXL01].

Die folgende Abbildung stellt eine Erweiterung des Web Service Stacks vor, in der Integrationsaspekte beachtet wurden:

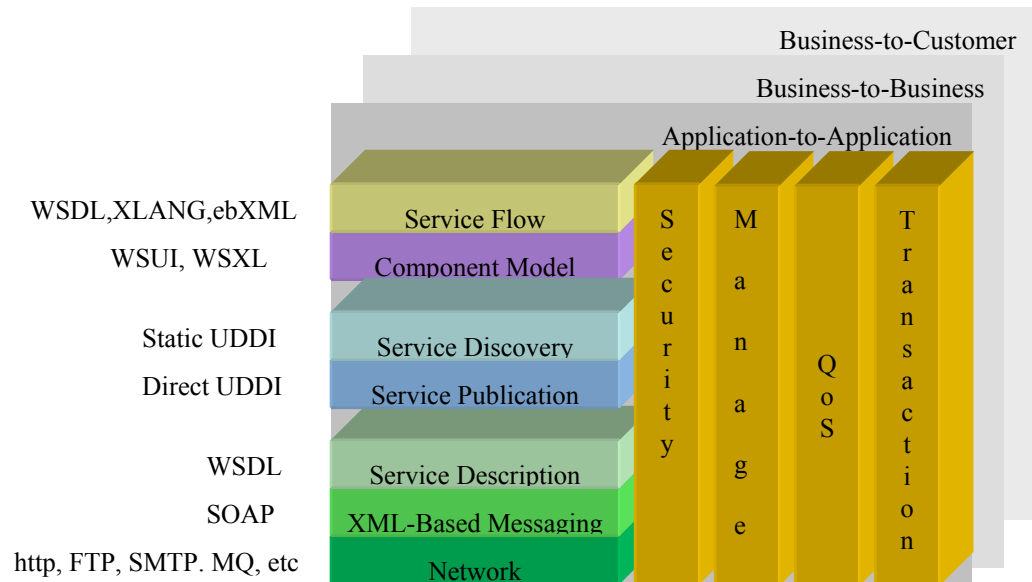


Abbildung 5: Der Web Services Stack erweitert zu einem EAI Framework

3 Fazit

Soll ein Web Services basiertes EAI Framework konzipiert werden, steht man vor der Entscheidung, bestehende Anwendungen in Komponenten umzuwandeln, die allein für sich existieren können und sich mit anderen Komponenten kombinieren lassen; oder eine Integrationsschicht aufzubauen, die den flexiblen Zugriff auf die Komponenten ermöglicht.

Bei der Konzeption steht man wiederum vor den bekannten Problemen der Komponenten- Technologie. Bei dem Entwurf eines Web Services basierten Frameworks muss differenziert werden, wie granular diese Schnittstellen sein sollen und welche Schnittstellen mit Web-Services bereitgestellt werden [Hop02].

Durch den Entwurf eines Web Services EAI Framework unter Zuhilfenahme des erweiterten Web Service Stacks⁹ kann mit einem Komponentenmodell die Front-End basierte Prozessautomation durch die Kopplung der Komponenten verbessert werden. Der Nachteil der herkömmlichen Integration auf Präsentationsebene ist, dass der Entwickler nur die Ausgaben der einbezogenen Systeme verarbeiten und nicht auf deren Logik zurückgreifen kann.

⁹ vgl. hierzu Abbildung 5

Eine wichtige Herausforderung, Voraussetzung und zugleich Chance wird es sein, offene, aber zugleich einheitliche Standards für die hier diskutierten Bereiche zu schaffen. Erst durch die konsequente Fortführung dieser Strategie, die mit der Standardisierung der Basisarchitektur begonnen hat, kann ein Web Services basierendes EAI Framework in der Zukunft umgesetzt werden.

Literatur

- [Anu⁺02] Anuff, E.; Chaston, M.; Moses, D.; Kropp, A.: Web Service User Interface (WSUI) 1.0; Working Draft - 11 February, 2002; <http://www.wsui.org/doc/20020211/WD-wsui-20020211.html>, (am 11. Februar 2003).
- [Arsa⁺02] Arsanjani, A.; Chamberlain, D.; Gisolfi, D.; Konuru, R.; Macnaught, J.; Maes, S.; Merrick, R.; Mundel, D.; Raman, T.; Ramaswamy, S.; Schaeck, T.; Thompson, R.; Diaz, A.; Lucassen, J.; Wiecha, C.: Web Services Experience Language (WSXL), <http://www-106.ibm.com/developerworks/webservices/library/ws-wsxl/>, IBM developerWorks, 2002, (am 02. Februar 2003).
- [Bar⁺02] Bartel, M.; Boyer, J.; Fox, B.; LaMacchia, B.; Simon, E.: XML-Signature Syntax and Processing W3C Recommendation, 2002 <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>, (am 10. Februar 2003)
- [Bell⁺01] Bellwood, T.; Clément, L.; Ehnebuske, D.; Hatley, A.; Hondo, M.; Husband, Y. L.; Januszewski, K.; Lee, S.; McKee, B.; Munter, J.; von Riegen, C.: UDDI Version 3.0; Published Specification, 19 July 2002; <http://uddi.org/pubs/uddi-v3.00-published-20020719.htm>, (am 02. Februar 2003).
- [Bray⁺00] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.: [Extensible Markup Language \(XML\) 1.0 \(Second Edition\)](#) W3C Recommendation, 2000, (am 05. Februar 2003)
- [Btp02] OASIS Business Transactions TC, <http://www.oasis-open.org/committees/business-transactions/>, (am 12. Februar 2003)
- [Cha⁺02] Champion, M.; Ferris, C.; Newcomer, E.; Orchard, D.: Web Services Architecture W3C Working Draft, 2002 <http://www.w3.org/TR/2002/WD-ws-arch-20021114/>, (am 04. Februar 2003).
- [Chris⁺01] Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S.: Web Services Description Language (WSDL) 1.1, W3C Note, 2001, <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, (am 12. Februar 2003).
- [Dun03] Dunn, B.: A Manager's Guide to Web Services , in eai Journal, S. 14-17, Ausgabe Januar 2003, in: [eaijournal.com](http://www.eaijournal.com/PDF/Dunn.pdf), <http://www.eaijournal.com/PDF/Dunn.pdf>, (am 09. Februar 2003).
- [EbXML03] N. N.: <http://www.ebxml.org/specs/index.htm#whitepapers>, (am 05. Februar 2003).
- [Fer92] Ferstl, O.: Integrationskonzepte betrieblicher Anwendungskonzepte. Fachbericht Informatik der Universität Koblenz-Landau, Nr. 1, 1992, S. 1-29.
- [Gis01] Gisolfi, D: The Web services architect, IBM developerWorks, 2001, <http://www-106.ibm.com/developerworks/library>, (am 12. Februar 2003).

- [Gudg⁺01]Gudgin, M.; Hadley, M.; Mendelsohn, N.; Moreau, J.; Nielsen, H. F.: SOAP Version 1.2 Part 1: Messaging Framework, W3C Candidate Recommendation, 2002, <http://www.w3.org/TR/2002/CR-soap12-part1-20021219>, (am 02. Februar 2003).
- [Gud⁺02]Gudgin, M.; Hadley, M.; Mendelsohn, N.; Moreau, J.; Nielsen, H. F.: SOAP Version 1.2 Part 2: Adjuncts, W3C Candidate Recommendation, 2002, <http://www.w3.org/TR/2002/CR-soap12-part2-20021219>, (am 10. Februar 2003).
- [Gul02] Gulp Knowledge Base: Definition EAI.
<http://www.gulp.de/kb/pt/techexpert/mysap.html>, 2002 (am 12.12.2002)
- [Ham01] Hammer, K.: Web Services and Enterprise Integration, in: eaiJournal 11/2001, <http://www.eaijournal.com>, (am 02. Februar 2003).
- [Hop02] Hoppermann, J.: Die Realität hinter dem Web-Services-Hype, Artikel vom 07.06.2002, http://www.computerwoche.de/index.cfm?pageid=254&artid=36945&main_id=36945&category=84&currpage=1, (am 10. Februar 2003).
- [Höf01] Höfling, J.: Web-Tools: Dienste kommen aus dem Netz, Information Week, Ausgabe 10, Mai 2001, <http://www.informationweek.de/index.php3?/channels/channel02/011039.htm>, (am 12. Februar 2003).
- [Kaib02] Kaib, M.: Enterprise Application Integration, Wiesbaden, 2002.
- [Kel02] Keller, W.: Enterprise Application Integration. Heidelberg, 2002.
- [Klöck02] Klöckner, M.: Web Services: Die Lösung aller Integrationprobleme?, 2002, http://www.sigs.de/publications/os/2002/05/kloeckner_OS_05_02.pdf, (am 12. Februar 2003)
- [Kre01] Kreger, H.: Web Services Conceptual Architecture (WSCA 1.0), IBM Software Group, 2001, <http://www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf>, Link besucht am 9.02.2003
- [KuRa96] Kurbel, K.; Rautenstrauch, C.: Integration Engineering. in: Heilmann, H.; Heinrich, L.; Roithmayr, F.: Information Engineering, München, Wien 1996, S. 167-191.
- [Ley01] Leymann, F.: Web Services Flow Language (WSFL 1.0), IBM Software Group, Mai 2001 <http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>, (am 06. Februar 2003)
- [Lin895] Linß, H.: Integrationabhängige Nutzeffekte der Informationsverarbeitung: Vorgehensmodell und empirische Ergebnisse. Wiesbaden, 1995.
- [Lin00] Linthicum, D.S.: Enterprise Application Integration, München, 2000.
- [Mert97] Mertens, P.: Integrierte Informationsverarbeitung. in: Mertens, P. u.a.: Lexikon der Wirtschaftsinformatik. Berlin 1997, S. 208-209.
- [Mer00] Mertens, P.: Integrierte Informationsverarbeitung 1 Administrations- und Dispositionssysteme in der Industrie. Band 1, Wiesbaden, 2000.

- [MeGri00] Mertens, P.; Griese, J.: Integrierte Informationsverarbeitung 2 – Planungs- und Kontrollsysteme in der Industrie. Wiesbaden, 2000.
- [Mied01] Miedl, Wolfgang: Web-Services: Das Versprechen der Einfachheit, in Computerwoche Online, http://www.computerwoche.de/index.cfm?pageid=255&artid=34357&main_id=34357&category=40&currpage=1&type=detail, (am 03. Februar 2003).
- [My02] Myerson, M.J.: Enterprise Application Integration. Boca Raton, 2002.
- [Rau01] Rausch, A.: Componentware – Methodik des evolutionären Architekturentwurfs. Dissertation an der TU München, 2001
- [Rea02] Reagle, J.: XML Encryption Requirements 3C Note, 2002
<http://www.w3.org/TR/2002/NOTE-xml-encryption-req-20020304>, (am 06. Februar 2003)
- [Ren02] Ren, F.: The Marketplace of C, <http://www.public.asu.edu/~mbfr2047/eai.html>, (am 22.5.2002).
- [Ring00] Ring, K.: EAI - Making the right Connections; Boston, 2000.
- [RPC92] N.,N.: W3C, *Introduction to RPC*, http://www.w3.org/History/1992/nfs_dxcern_mirror/rpc/doc/Introduction/Abstract.html, 1992, Link besucht am 10. Februar 2003
- [Sche97] Scheckenbach, R.: Semantische Geschäftsprozessintegration. Wiesbaden, 1997.
- [Scheer90] Scheer, A.-W.: CIM. Berlin, 1990.
- [Schill00] Schill, A.: Middleware im Vergleich. [http://www.competence-site.de/eaisysteme.nsf/C937534DE4B6BC14C1256A14005E5D60/\\$File/middleware.pdf](http://www.competence-site.de/eaisysteme.nsf/C937534DE4B6BC14C1256A14005E5D60/$File/middleware.pdf), 2000, (am 12.12.2002).
- [Schu+01] Schullan, U.; Hefe, D.: Den Web-Services gehört die Zukunft, in: Computerwoche Online, <http://www1.computerwoche.de/index.cfm?pageid=255&artid=27159&type=detail&category=40>, (am 08. Februar 2003).
- [Szyp97] Szyperski, C.: Component Software – Beyond Object-Oriented Programming. Harlow, 1997.
- [That01] Thatte, S.: XLANG-Web Services for Business Process Design, Microsoft, 2001, http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, (am 06. Februar 2003)
- [Van00] Vander Hey, D.: One Customer, One View; in: Intelligent Enterprises Magazine, Heft 4; 1.3.2000.
- [Wil99] Wilkes, L.: Legacy Componentization and Wrapping: Reaping long-term rewards. in: Component Strategies, 1999, S. 50-57
- [Xaml00] Technology Reports: Transaction Authority Markup Language (XAML), <http://www.oasis-open.org/cover/xaml.html>, (am 02. Februar 2003).
- [Zei00] Zeidler, C.: Komponententechnik kritisch betrachtet. in: Computerwoche, Nr. 50, 2000, Bd. 27, S. 61-62.