Informationssysteme für E-Commerce, EMISA-2000, 8.-10. November 2000, Linz, Östereich, H. Jasper, J. Küng, G. Vossen (Hrsg.), Universitätsverlag Rudolf Trauner, Linz, Österreich, pp. 133-147

Ein EC-Portal als Integrationsplattform für Electronic Commerce Systeme

Matthias Book (WebService Haltern), Volker Gruhn (Universität Dortmund), Lothar Schöpe (ICD e.V.)

Abstract

In diesem Erfahrungsbericht wird die Konzeption und Realisierung eines Electronic-Commerce-Portals (ECP) als Integrationsplattform für Electronic-Commerce-Systeme beschrieben. Durch dieses Electronic-Commerce-Portal sollen insbesondere die Außendienstmitarbeiter (VAD) von Versicherungsunternehmen bei ihrer täglichen Arbeit unterstützt werden, wobei der Einsatz des Electronic-Commerce-Portals in einem Intranet erfolgt. Die Versicherungs-Außendienstmitarbeiter benötigen Zugriff auf Informationen über die privaten oder gewerblichen Kunden, die sie zu betreuen haben. Sie müssen aber auch bei der Organisation ihrer Arbeit durch Terminplaner mit Erinnerungsfunktion, Adressverwaltung etc. unterstützt werden. Das Versicherungsunternehmen hat das Bestreben, seine Mitarbeiter mit den aktuellsten Informationen über das Produktportfolio, Tarifen sowie Gesetzestexten und -kommentaren zu versorgen. Ein Electronic-Commerce-Portal muss daher eine Vielzahl von Funktionen bereitstellen. Der Vorteil eines Portals liegt in der Integration von verschiedenen Softwaresystemen, die diese Funktionen bereitstellen, so dass der Nutzer eines Portals zielgerichtet an die gewünschten Informationen herangeführt wird. Er muss einerseits nicht umständlich suchen und wird andererseits nicht mit sinnlosen Informationen versorgt. Die objektorientierte Konzeption mittels UML, die Realisierung von Adaptern zur Integration von Softwaresystemen in der Programmiersprache Java und die Verwendung einer Middleware zur Kommunikation innerhalb des Portals waren Gegenstand eines Projekts. Dieses Projekt wurde von der Universität Dortmund in Kooperation mit verschiedenen Versicherungs- und Softwareunternehmen durchgeführt. Das Electronic-Commerce-Portal wurde exemplarisch für ein Versicherungsunternehmen durch die Integration eines Legacy-Systems auf der Basis von XML realisiert.

1. Einleitung

Der konventionelle Geschäftsverkehr – der nicht elektronisch unterstützte Geschäftsverkehr – erfolgt heutzutage mittels verschiedener Medien wie Brief, Telefon und Fax in konventioneller Art und Weise. Der elektronische Geschäftsverkehr nutzt andere Medien wie z.B. Electronic Mail, EDI, WWW und Internet [1],[2],[3]. Die Partner beim Geschäftsverkehr – sei es elektronisch oder konventionell – sind abstrakt betrachtet zunächst einmal Anbieter und Kunde, jedoch können je nach Art des Geschäftsverkehrs die Partner auch im speziellen als Anbieter und Nachfrager, Empfänger und Lieferant oder Hersteller und Zulieferer, aber auch als Management und Mitarbeiter bezeichnet werden.

Diese Rollen, Anbieter und Kunde, können von Unternehmen oder Endverbrauchern eingenommen werden. Wenn sowohl die Rolle des Anbieters als auch die Rolle des Kunden von einem Unternehmen eingenommen wird, wird der Geschäftsverkehr als Business-to-Business (B2B) bezeichnet, wird die Rolle des Kunden dagegen von einem Endverbraucher eingenommen, so wird der Geschäftsverkehr als Business-to-Consumer (B2C) bezeichnet. Anstelle eines Unternehmens als Anbieter oder als Kunde kann auch eine Behörde treten. Dann wird der Geschäftsverkehr auch als Business-to-Administration (B2A), Administration-to-Administration (A2A) oder Administration-to-Consumer (A2C) bezeichnet [4]. Der Geschäftsverkehr, der nur intern innerhalb eines Unternehmens zwischen den Mitarbeitern (Geschäftsführung, Management, etc.) – d.h. ohne externen Partner – abläuft, wird als Business-to-Employee (B2E) bezeichnet [5]. Diese Bezeichnungen des Geschäftsverkehrs sind zwar erst mit den Begriffen Electronic Commerce (EC) und Electronic Business (EB) aufgekommen, können zunächst jedoch wertfrei auch für den konventionellen Geschäftsverkehr verwendet werden. Aus der Sicht eines Anbieters von Produkten bzw. Dienstleistungen umfasst Electronic Commerce die Verkaufsförderung, Verkaufsdurchführung, Distribution und Verkaufsnachbereitung. Electronic Commerce dient primär der Unterstützung des Business-to-Consumer (B2C) oder Administration-to-Consumer (A2C) Geschäftsverkehrs. Electronic Business dagegen orientiert sich nicht am Endkunden, sondern dient der Unterstützung des Geschäftsverkehrs zwischen Unternehmen, Behörden oder auch zwischen Unternehmen und Mitarbeitern [6]. Bei Electronic Commerce und Electronic Business verkehren Anbieter und Kunden auf elektronischem Weg über ein Datenkommunikationsnetz miteinander. Das Internet stellt mit seinen Diensten (TCP/IP, FTP, NNTP, SMTP, HTTP, etc.) ein solches Datenkommunikationsnetz dar.

Es werden die unterschiedlichsten Softwaresysteme eingesetzt, um den Geschäftsverkehr bei Electronic Commerce und Electronic Business zu unterstützen. Diese Softwaresysteme können individuell, einfach oder komplex sein und den Geschäftsverkehr vollständig oder teilweise unterstützen. So werden z.B. Shopsysteme mit oder ohne Integration eines Warenwirtschaftssystems unter Verwendung einer Web-Client/Server-Technik eingesetzt. Zur Unterstützung des Bestellwesens und der Lagerhaltung innerhalb des Electronic Business werden Systeme zum Austausch von Informationen mittels der Protokolle XML oder EDIFACT verwendet. Es interagieren zur Unterstützung des Geschäftsverkehrs immer mindestens zwei Systeme miteinander, die nicht notwendigerweise gleich sein müssen, z.B. ein Warenwirtschaftssystem mit einem Shopsystem und einem Web-Serversystem und einem Web-Clientsystem oder eine SAP R3-Procurementkomponente mit einem Warenwirtschaftssystem.

Ein Electronic-Commerce-Portal dient, zumal wenn es für eine spezielle Benutzergruppe in einem Intranet eingesetzt wird, der Unterstützung des Geschäftsverkehrs im Sinne des Business-to-Employee (B2E). Hierbei wird der Geschäftsverkehr zwischen dem Management und den verschiedenen Mitarbeitern, aber auch zwischen den Mitarbeitern untereinander unterstützt. Diese Unterstützung des Geschäftsverkehrs erfordert den Einsatz von verschiedenen Softwaresystemen (Legacy-System, Shopsystem, Web-System, Internet-System, Office-System). In diesem Sinne dient ein Electronic-Commerce-Portal als Integrationsplattform für die verschiedenen Softwaresysteme. Ziel ist es, den Geschäftsverkehr im Sinne des B2E optimal zu unterstützen.

2.Systemarchitektur

Die erste Phase im Softwarentwicklungsprozess war eine Analyse der inhaltlichen und funktionalen Anforderungen für das Electronic-Commerce-Portal. Dies erfolgte durch eine emiprische Betrachtung von sogenannten Außendienstsystemen, die bei Versicherungsunternehmen im Einsatz sind, um Möglichkeiten und Defizite zu erkennen. Darüber hinaus wurden durch Interviews und Diskussionen mit VADs die typischen Arbeitsabläufe eines VADs analysiert. Die Ergebnisse der Analyse wurden strukturiert, indem die umfassenden Arbeitsabläufe in einzelne Aktivitäten unterteilt, priorisiert und in Anforderungsformularen dokumentiert wurden. In dieser Phase des Projekts wurde erkannt, dass das Electronic-Commerce-Portal als Integrationsplattform für verschiedene heterogene Subsysteme dient. Ausgehend von einer 3-Ebenen-Architektur [7]bei der die Benutzungsoberfläche und die Datenverwaltung von der funktionalen Anwendungslogik

getrennt wird, wurden in der Ebene der funktionalen Anwendungslogik zunächst die folgenden Subsysteme eines Electronic-Commerce-Portals klassifiziert:

Office-System (OS):* Durch das Office-System werden sowohl Adressen resp. Kontakte als auch Termine verwaltet. Bei Adressen und Kontakten werden harte und weiche Daten unterschieden. Durch das Office-System werden nur weiche Daten verwaltet; dies sind Daten, die der VAD für seine persönlichen Zwecke erfasst (z.B. Besuchshistorie, Anmerkungen zu Verträgen, etc.)

Content-Management-System (CMS): Die Bereitstellung von Informationen durch das Versicherungsunternehmen für die VADs, aber auch durch die VADs für das Versicherungsunternehmen und andere VADs wird durch das CMS verwaltet. Auf der Basis eines Berechtigungskonzepts sind Innendienst- und Außendienstmitarbeiter in der Lage, größere Informationen (Prospekte, Handbücher, Gesetzestexte, Urteile, Marketingstrategien, etc.) zur Verfügung zu stellen, so dass der VAD auf diese Informationen zur Unterstützung seiner Arbeit zugreifen kann.

Procurement-System (PS): Die Beschaffung von Verbrauchsmaterial einerseits und die Inanspruchnahme von Dienstleistungen (Seminare, Kurse, Veranstaltungen, Schulungen, Immobilienvermittlung etc.) andererseits ist für Versicherungsunternehmen, wenn sie zentral erfolgt, ein wichtiger Kostenfaktor. Sämtliche VADs, aber auch Innendienstmitarbeiter können aus einem zentralen Katalog Produkte und Dienstleistungen auswählen.

Legacy-System (LS): Ein Partnermanagementsystem oder ein Provisionierungssystem liefert Informationen (Verträge, Risiken, etc.) über private und gewerbliche Kunden bzw. bietet die Möglichkeit, eine individuelle Tarifierung durchzuführen. Diese Systeme sind als Hostanwendungen realisiert und können über eine XML-Schnittstelle in das Electronic-Commerce-Portal integriert werden.

Kommunikationssystem (KS): Das Kommunikationssystem stellt die Schnittstelle zu den Techniken der Telekommunikation dar. Texte und Benachrichtigungen können in Form von e-Mails oder

---

* Die Verwaltung von Daten erfolgt einerseits traditionell durch ein Hostsystem, z.B. IBM MVS (für die sogenannten harten Daten) und zusätzlich durch ein Office-System, z.b. IBM Lotus oder MS Office (für die sogenannten weichen Daten). Der Zugriff auf die harten Daten erfolgt durch eine XML-Schnittstelle, wodurch auch eine Synchronisation der harten und weichen Daten möglich ist.

SMS-Nachrichten, aber auch als Fax versandt werden. Durch das Terminplanersystem gesteuert werden diese Texte und Benachrichtungen jeweils zu vorher bestimmten Zeitpunkten versandt.

Suchsystem (SS): Um im gesamten ECP nach Inhalten suchen zu können, bietet das Suchsystem Funktionen zur kontextsensitiven Suche an. Dies umfasst neben der Möglichkeit der Volltextsuche auch die Suche nach bestimmten Merkmalen von Informationen.

Portal-Administrationssystem (PAS): Durch das PAS werden Funktionen zur Verfügung gestellt, die das Auswerten von Protokolldateien und deren graphische Präsentation ermöglichen. Darüberhinaus wird durch das PAS sichergestellt, dass der Benutzer des Portals sich nur einmal dem Portal gegenüber identifizieren muss.

Die Benutzeroberfläche als Zugang zum Electronic-Commerce-Portal ist mit der Hypertext Markup Language (HTML) realisiert worden. Zur Datenverwaltung wird, sofern die einzelnen Subsysteme über keine eigene Datenverwaltung verfügen, ein relationales Datenbank-Managementsystem verwendet.

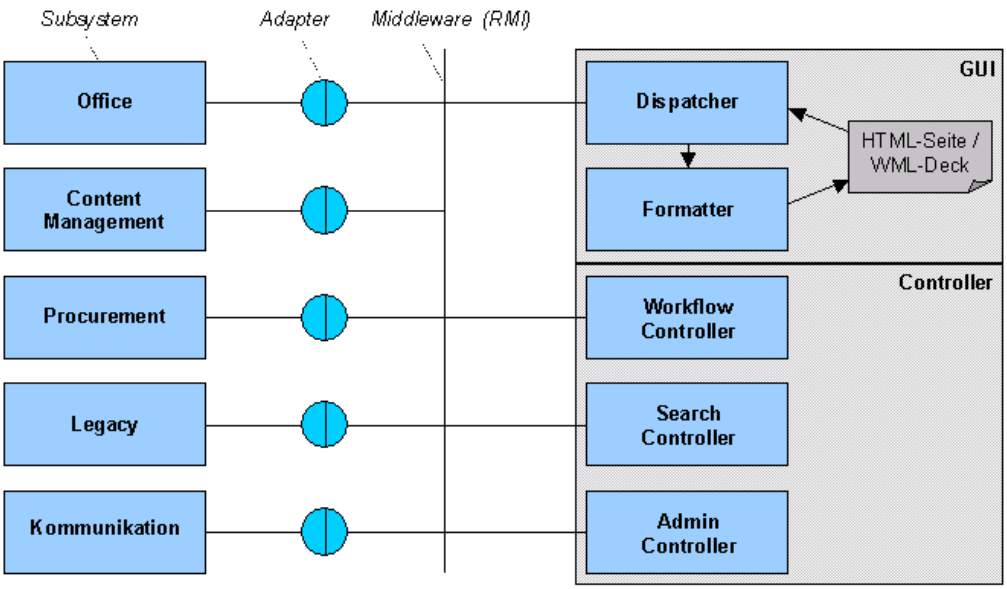

Abbildung 1: Systemarchitektur

Die Funktionalität der Subsysteme Office, Content Management, Procurement und Kommunikation wird durch bereits existierende, externe Systeme bereitgestellt, die über Adapter (Kreise in *Abb. 1*) in das Electronic-Commerce-Portal integriert werden. Durch die Kombination der Funktionalität verschiedener Subsysteme entstehen neue Funktionen des Gesamtsystems, die sich an den

Geschäftsprozessen der Benutzer orientieren: Um z.B. einen wichtigen Vertragsstichtag aus dem Legacy-System „Partnerdatenbank“ zu lesen, einen entsprechenden Eintrag in den Terminkalender des Office-Systems zu schreiben und das rechtzeitige Versenden einer Erinnerung über das Kommunikationssystem zu veranlassen, wäre bei getrennten Subsystemen eine Reihe von unterschiedlichen Tätigkeiten erforderlich. Im Electronic-Commerce-Portal kann der VAD diese Aufgabe hingegen in wenigen Schritten lösen.

Die dazu notwendige Verknüpfung von Tätigkeiten und Konsolidierung von Daten wird durch Controller geleistet, die als Gehirn des Electronic-Commerce-Portals betrachtet werden können: Sie erhalten vom Benutzer Aufträge zur Ausführung komplexer Geschäftsprozesse, steuern daraufhin die Subsysteme entsprechend an und liefern die konsolidierten Daten an den Anwender zurück.

Die Subsysteme werden nicht direkt vom Benutzer angesprochen und treten nicht selbst mit ihm in Kontakt. Aus diesem Grund wurden Such- und Administrationssystem nicht als externe Systeme, sondern als Controller klassifiziert, da sie aktiv Informationen von den anderen Systemen anfordern oder ihnen übergeben. Alle weiteren Geschäftsprozesse werden vom Workflow-Controller realisiert und gesteuert.

Die externen Systeme verfügen über ein oder mehrere Schnittstellen (APIs), durch die die jeweilige Funktionalität genutzt werden kann. Jedes externe System wird daher über genau einen Adapter mit der zentralen Middleware verbunden. Jeder Adapter stellt der Middleware eine Menge von Methoden zur Verfügung, die die ursprüngliche Schnittstelle des externen Systems kapseln. Auf diese Weise muss das (möglicherweise komplizierte) ursprüngliche Interface nicht öffentlich bekannt sein, um die Funktionalität des Subsystems zu nutzen. Stattdessen können andere Subsysteme einfach die Methoden benutzen, die vom Adapter zur Verfügung gestellt werden. Um zum Beispiel eine e-Mail über das Kommunikationssystem zu versenden, genügt es, die entsprechende Methode des Kommunikations-Adapters aufzurufen, die sich dann darum kümmert, aus den übergebenen Parametern eine RFC822-konforme Nachricht [8] zu konstruieren, eine Sitzung mit dem SMTP-Server aufzubauen und die e-Mail zu versenden. Zudem erlaubt die Kapselung, externe Systeme einfach auszutauschen: Wenn das ursprüngliche Interface eines Systems sich ändert, muss nur dessen Adapter umgeschrieben werden, während alle anderen Subsysteme unberührt bleiben.

Der Benutzer interagiert hauptsächlich über einen Web-Browser mit dem Electronic-Commerce-Portal. Dies hat wichtige Auswirkungen auf den Kontrollfluss im System. In traditionellen Software-Systemen kann der Dialog-Ablauf zum Großteil vom System selbst bestimmt werden: Zum Beispiel muss nur eine modale Dialogbox geöffnet werden, um den Nutzer zur Ausführung einer bestimmten Aktion zu zwingen, bevor er irgendetwas anderes tun kann [9]. Im Web werden dagegen alle Aktionen vom Benutzer initiiert. Der Server kann keine Informationen zum Browser „pushen“, die der Benutzer nicht angefordert hat.†

Infolgedessen bleiben die externen Systeme (Office, Content Management etc.) des Electronic-Commerce-Portals passiv und reagieren nur auf Benutzeranforderungen, die ihnen auf dem in *Abb. 2* dargestellten Pfade übermittelt werden:

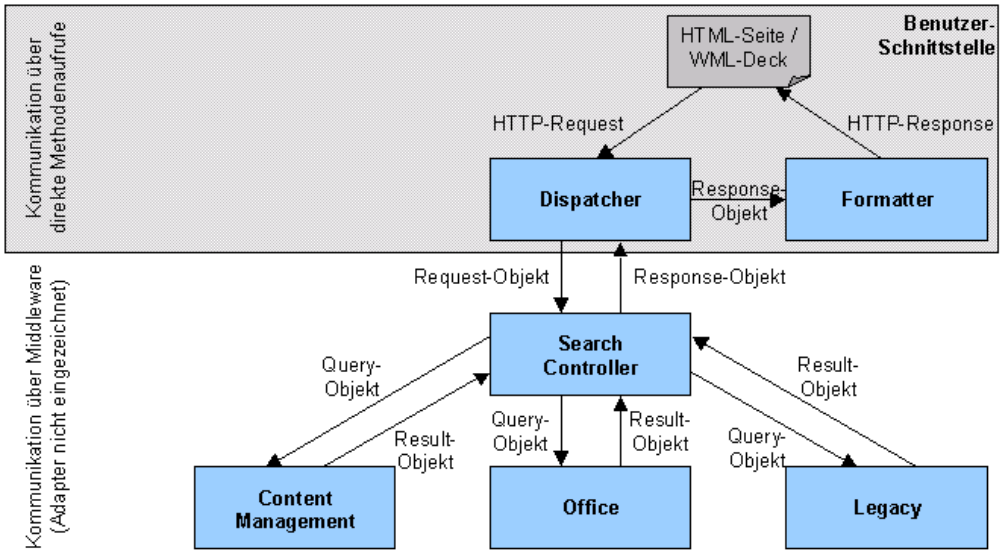

Abbildung 2: Kommunikation innerhalb des Electronic-Commerce-Portals

Jede Benutzeraktion wie das Anklicken eines Links or das Absenden eines Formulars erzeugt einen HTTP-Request 10, der von einem zentralen Dispatcher empfangen wird. Der Dispatcher liest den HTTP-Request-String, baut aus seinen Inhalten ein Request-Objekt und leitet es an den Controller

† Dies gilt für eine Benutzerschnittstelle aus einfachen HTML-Seiten. Natürlich wäre es möglich, auf der Client-Seite ein Java-Applet laufen zu lassen, das Informationen anzeigt, die ihm vom Server zugepushed werden. Dies würde jedoch einen Java-fähigen Browser erfordern und somit die meisten derzeit verfügbaren mobilen Endgeräte wie WAP-Telefone, Organizer etc. ausschließen. Einfaches HTML dagegen stellt die geringsten Anforderungen an die Zielplattform, und die Subsysteme, die es produzieren, können einfach modifiziert werden, um ähnliche Formate wie Wireless Markup Language (WML) zu erzeugen.

weiter, der für die Behandlung der angeforderten Aufgabe verantwortlich ist: Der Search-Controller und der Admin-Controller implementieren die Funktionalität der zuvor erwähnten Systeme Suche und Portal-Administration; alle anderen Transaktionen, an denen externe Systeme beteiligt sind, werden vom Workflow-Controller gehandhabt.

Die Controller werten die Request-Objekte aus, die ihnen vom Dispatcher übergeben werden. Je nach Art des Requests senden sie Befehle oder Anfragen an die externen Systeme, konsolidieren die Ergebnisse und schicken sie zum Dispatcher zurück. Um dies zu leisten, muss der spezifische Arbeitsablauf (Workflow) zur Erfüllung eines jeden Requests im jeweiligen Controller fest einprogrammiert sein. Wird zum Beispiel der Request empfangen, in allen externen Systemen nach einer bestimmten Person zu suchen, so fragt der Search-Controller das Office-, Content-Management- und Legacy-System ab und schickt die kombinierten Ergebnisse als Response-Objekt zum Dispatcher zurück.

Der Dispatcher leitet das vom Controller erhaltene Response-Objekt an den Formatter weiter. Dieses Subsystem ist für die Konvertierung des Response-Objekts in ein Format verantwortlich, das vom Endgerät des Nutzers dargestellt werden kann. In den meisten Fällen wird das gewünschte Ausgabeformat Hypertext Markup Language (HTML) [11] sein, die von einem breiten Spektrum an Endgeräten verarbeitet werden kann. Für exotischere Endgeräte wie WAP-Telefone und Organizer können andere Formatter Ausgabeformate wie Wireless Markup Language (WML) erzeugen. Diese Flexibilität ist ein wesentlicher Vorteil der Trennung zwischen Formatter und Controller: Da die Implementierung der Benutzerschnittstelle in einem einzigen System konzentriert ist, kann die visuelle Darstellung von Informationen geändert oder ergänzt werden, ohne die Subsysteme ändern zu müssen, die diese Informationen zur Verfügung stellen.

Aufgrund von Performance-Abwägungen und besonderen Systemanforderungen laufen die meisten externen Subsysteme sowie der Web-Server auf getrennten Rechnern. Diese verteilte Architektur erfordert eine Middleware wie CORBA oder RMI, um Methodenaufrufe und Objektübergaben zwischen den verschiedenen Subsystemen zu koordinieren. Der Gebrauch der Middleware ist innerhalb einzelner Subsysteme wie der Benutzerschnittstelle natürlich nicht nötig: Der Dispatcher ruft zum Beispiel direkt eine Methode des Formatters auf, um ein Response-Objekt zu übergeben, das er von einem Controller erhalten hat.

Der Dispatcher und die Controller könnten jedoch auf verschiedenen Rechnern laufen. Aus diesem Grund tauschen sie Objekte über die Middleware aus. Zwei Modelle zur Kommunikation wurden dazu in der Entwurfsphase des Projekts in Betracht gezogen:

Publisher/Subscriber-Modell: Der Dispatcher veröffentlicht ein Request-Objekt über die Middleware und gibt seine Verfügbarkeit mit einem Event bekannt, das den Typ des Requests beschreibt. Die Controller können Events abonnieren, die für sie relevant sind, und eine Kopie der entsprechenden Request-Objekte von der Middleware erhalten.

Expliziter Controller-Aufruf: Der Dispatcher entscheidet in Abhängigkeit vom Typ des Requests, welche(n) Controller er aufrufen muss, um das Request-Objekt über die Middleware zu übergeben.

Im Publisher/Subscriber-Modell wird der Dispatcher effektiv auf einen Mechanismus zur Konvertierung von HTTP-Request-Strings in Request-Objekte reduziert, da er nicht weiß, welcher Controller für welchen Request-Typ verantwortlich ist. Dies scheint zunächst eine elegante Entkopplung darzustellen, bei genauerer Betrachtung werden jedoch einige Fallstricke offensichtlich: Obwohl der „sendende“ Teil des Dispatchers nicht geändert werden muss, wenn ein neuer Controller zur Abonnenten-Liste hinzugefügt wird, muss der „empfangende“ Dispatcher-Teil dennoch darauf vorbereitet werden, Response-Objekte von dem zusätzlichen Controller anzunehmen. Im Hinblick auf den Aufwand für die Definition von Schnittstellen zwischen dem Dispatcher und den Controllern bringt das Publisher/Subscriber-Modell keine Vorteile gegenüber dem expliziten Controller-Aufruf: Sowohl Dispatcher als auch Controller müssen wissen, welche Attribute für welche Request-Objekte definiert sind – unabhängig davon, wie die Objekte transportiert werden. Weitere Probleme entstehen aus der Multi-User-Umgebung des Electronic-Commerce-Portals: Der Dispatcher muss wissen, welches vom Controller zurückgelieferte Response-Objekt mit welchem ihm übergebenen Request-Objekt korrespondiert. Beim expliziten Controller-Aufruf wird diese Zuordnung implizit vom Aufruf-Stack der Middleware übernommen. Im Publisher/Subscriber-Modell müsste jedes Request-Objekt (und die zwischen Controllern und Subsystemen ausgetauschten Objekte) mit einem eindeutigen Schlüssel versehen werden, um die eingehenden Response-Objekte zuzuordnen – ein unnötiger Overhead.

Controller und Subsysteme kommunizieren durch den Austausch von Business-Objekten[12], d.h. Einheiten, die zentrale Bedeutung für den Workflow im Electronic-Commerce-Portal besitzen. Die folgenden Business-Objekte sind darum allen Controllern und Subsystemen bekannt:

- Benutzer
- Kontakt
- Termin
- Aufgabe
- Nachricht
- Shop-Artikel
- Bestellung
- Bestellhistorie
- Suchanfrage
- Suchergebnis

Um zum Beispiel einen Termin einzutragen, erzeugt der Workflow-Controller ein Termin-Objekt aus den vom Dispatcher empfangenen Daten und gibt es an eine Methode des Office-Subsystems weiter, die den Termin in den Kalender des Benutzers einträgt. Wenn der Benutzer angibt, dass er rechtzeitig per e-Mail an den Termin erinnert werden möchte, erzeugt der Workflow-Controller zusätzlich ein Nachricht-Objekt, verbindet es mit einer Kopie des Termin-Objekts und gibt es an das Kommunikationssystem weiter, wo es in die Warteschlange für e-Mail-Nachrichten eingereiht und zum gewünschten Zeitpunkt versandt wird.

3.Realisierung

In der Realisierungsphase musste für jedes Subsystem eine „Make-or-Buy"-Entscheidung getroffen werden. Nach einer Bewertung existierender Lösungen und einer Gegenüberstellung des geschätzten Aufwands für die Eigenentwicklung eines Subsystems vs. der Integration einer bestehenden Lösung wurden folgende Systeme für die Integration ausgewählt[13],[14],[15],[16],[17],[18]:

- *Office:* Outlook 98 der Microsoft Corporation
- *Content Management:* Pirobase 4.0 der PiroNet AG
- *Procurement:* SmartStore Standard Edition 2.0 der SmartStore AG
- Kommunikation:
    - e-mail:JavaMail von Sun Microsystems, Inc.
    - Fax: sendfax – Freeware; enthalten in Linux 6.3 (i368) der SuSE GmbH
    - SMS: yaps – Freeware; enthalten in Linux 6.3 (i368) der SuSE GmbH
- *Legacy:* Beispiel-Partnerdatenbank der Continentale Versicherung

Der Nachweis der Praktikabilität der Integrierbarkeit dieser Systeme erfolgte durch die Entwicklung von Prototypen, d.h. „Quick-and-Dirty"-Implementierungen der in der Systemarchitektur beschriebenen Adapter. Ziel dieser Prototypen war der Beweis, dass es möglich ist, die

ursprünglichen Interfaces der externen Systeme zu kapseln und ihre Schlüssel-Features über die Adapter zugänglich zu machen. Dieses Ziel wurde für alle Subsysteme erreicht, so dass der Weg für die nächste Phase des Softwareentwicklungsprozesses frei war.

Für die Phase des objekt-orientierten Entwurfs wurde die Unified Modeling Language (UML) [19],[20]verwendet. Die Schlüssel-Features aller Subsysteme wurde in Use Cases modelliert, um Business-Objekte und mögliche Abhängigkeiten zwischen den Subsystemen zu identifizieren. Aufbauend auf den Erkenntnissen aus diesem Schritt wurden anschließend konkrete Klassen für jedes Subsystem definiert. Um eine einfache Konsolidierung der Ergebnisse zu gewährleisten und spätere Änderungen an den Subsystemen zu ermöglichen, ohne davon abhängige Klassen zu berühren, wird jedes Subsystem nach außen durch eine Fassadenklasse repräsentiert. Diese Klasse stellt alle Methoden zur Verfügung, die andere Klassen benötigen, um mit dem Subsystem zu kommunizieren. Als Beispiel betrachten wir, wie eine Suchanfrage ans Legacy-System behandelt wird *(Abb. 3):*

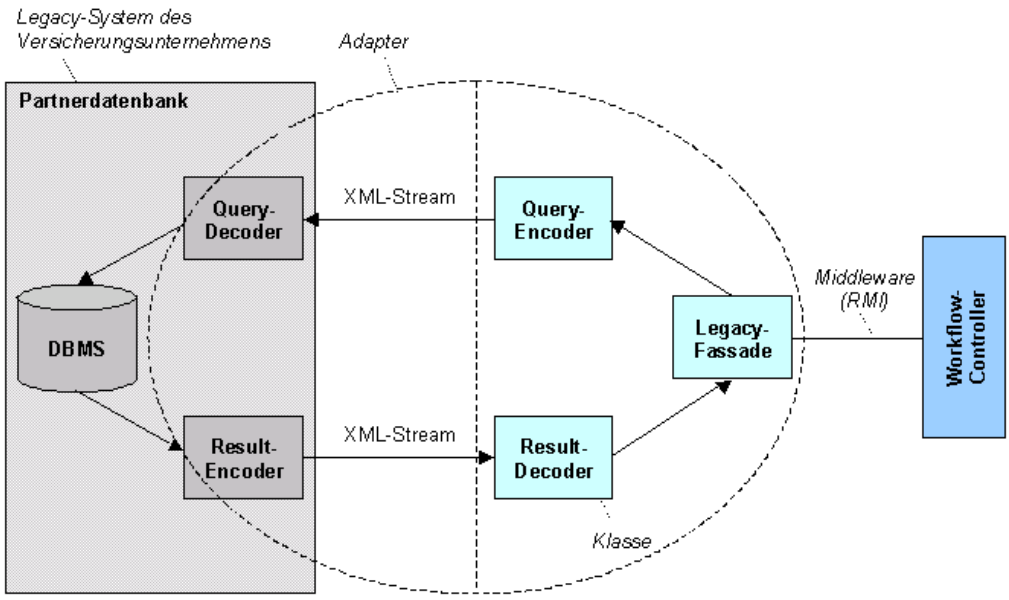

Abbildung 3: Integration des Legacy-Systems

Die Integration des Legacy-System erfolgt in diesem Fall durch eine in XML konvertierte Datenbankanfrage[21]. Das große Rechteck links in der Abb. 3 stellt einen Blick in das Legacy-Subsystem dar, das wir aus vorhergehenden Abbildungen kennen. Die kleineren Rechtecke repräsentieren Klassen. Da nur die Legacy-Fassadenklasse über die Middleware mit dem Controller verbunden ist, übergibt der Controller in unserem Beispiel das Suchanfragen-Objekt nicht direkt an

den Query-Encoder. Stattdessen wird die Suchanfrage der Legacy-Fassadenklasse übergeben, die es dann an den Query-Encoder weiterleitet. Diese Klasse ist ein Teil des Adapters, der das ursprüngliche Interface des externen Systems kapselt: Im Fall des Legacy-Systems erfolgen die Anfragen an die Partnerdatenbank sowie die Ergebnisse von der Partnerdatenbank auf der Basis von XML [22],[23],[24] um maximale Unabhängigkeit von Plattform- und Transportmedium zu wahren. Die XML-codierte Suchanfrage läuft über die Datenbank des Versicherungsunternehmens, und das codierte Ergebnis wird an den Adapter des Legacy-Subsystem zurückgeliefert. Dort erzeugt der Result-Decoder ein Suchergebnis-Objekt und übergibt es an die Legacy-Fassadenklasse, die es an den Workflow-Controller weiterleitet.

An diesem Beispiel wird nochmals der Zweck der Business-Objekte deutlich: Das Gesamtsystem ist unabhängig vom ursprünglichen Datenformat, mit dem ein Subsystem intern arbeitet (im Fall des Legacy-Systems: XML). Die Fassadenklassen erzeugen stattdessen Business-Objekte, die allen Controllern bekannt sind.

Nach der Konsolidierung der Entwürfe für die Subsysteme, die Controller und die Benutzerschnittstelle ging das Team zur Implementierungsphase über. Die meisten Klassen wurden in der Programmiersprache Java [25] implementiert, lediglich der Adapter für das Office-System verwendet Microsoft Visual C++ [26], um die Microsoft Outlook 98 API anzusprechen.

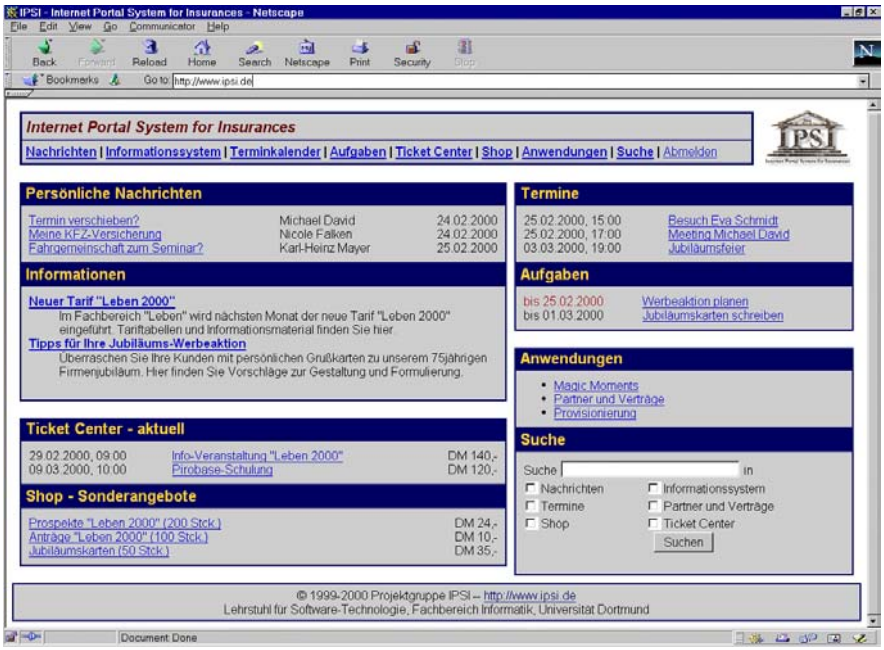

Abbildung 4: Homepages des Electronic Commerce Portals

Abb. 4 zeigt die Homepage des Electronic-Commerce-Portals. Nach der Authentifizierung erhält der VAD auf einen Blick alle Informationen, die zu diesem Zeitpunkt für ihn relevant sind: Persönliche Nachrichten, interessante Dokumente aus dem Content-Management-System, aktuelle Termine und fällige Aufgaben aus dem Office-System, Veranstaltungen und Artikel aus dem Procurement-System. Legacy-Anwendungen wie die Partnerdatenbank und ein Provisionierungssystem sind über Links auf der Homepage zugänglich. Eine Suchmaske erlaubt Meta-Suchen über ausgewählte Bereiche des Portals.

4.Zusammenfassung

Die Implementierung eines Portals für eine gezielte Benutzergruppe ist im wesentlichen eine Integrationsaufgabe. Dies hatte wichtige Konsequenzen für den verwendeten Softwareprozess, so war z.B. der Aufwand für die Entwicklung von Prototypen im Projektplan nicht vorgesehen und eingeplant. Die Backend-Integration basiert auf einer Middleware, die Frontend-Integration basiert auf einer weit verbreiteten Benutzerschnittstelle, die sorgfältiges Design erforderte.

Die meisten generellen Anforderungen (resultierend aus der Anforderungsanalyse) an das Portal konnten durch Integration von externen Subsystemen erfüllt werden. Um den Softwareprozess für die weitere Entwicklung des Portals effektiv planen zu können, war der Einsatz von Prototypen (s.o.) unabdingbar. Erst nach der Implementierung dieser Prototypen konnte die Realisierbarkeit der Architektur bewertet werden und der konkrete Aufwand und die Dauer der identifizierten Aufgaben abgeschätzt werden. Für den produktiven Einsatz des Portals zeigte sich, dass die Offenheit der Architektur ein entscheidender Faktor ist. Viele weitere Legacy-Systeme mussten nach dem ersten Release hinzugefügt werden; weitere externe Subsysteme wurden für individuelle Anwendergruppen ausgetauscht. Alle diese Modifikationen hängen von einer klaren und modularen Architektur ab. Rückblickend scheint es, dass es hilfreich gewesen wäre, das Portal als komponentenbasiertes System auf Basis eines standardisierten Komponentenmodells wie JavaBeans oder COM zu entwickeln.

Zusammenfassend lässt sich sagen, dass der Aufwand für die Implementierung geringer war als ursprünglich erwartet, da von der Verfügbarkeit externer Subsysteme profitiert werden konnte. Die gelösten Arbeitsaufgaben waren andere als zunächst geplant: Mehr Aufgaben als geplant waren Integrationsaufgaben. Schließlich wurden nur einige tausend Zeilen Programmcode geschrieben,

diese Software fungierte jedoch als Bindeglied zwischen existierenden Systemen und erforderte daher einen extrem detaillierten Entwurf sowie sorgfältige Tests.

---

[1] ZWASS, V., Electronic Commerce: Structures and Issues, in: International Journal of Electronic Commerce, Vol. 1, No. 1, M.E. Sharpe, Armounk, NY, USA, 1996

[2] CONNELLY, D.W., An Evaluation of the World Wide Web as a Platform for Electronic Commerce, in: R. Kalakota, A. Whinston (ed.) Readings in Electronic Commerce, Addison Wesley,. Reading, MA, USA, 1996.

[3] ILLIK, J.A., Electronic Commerce – Eine systematische Bestandsaufnahme, in: HMD, 35. Jahrgang, Heft 199, Hüthig Verlag, Heidelberg, 1998

[4] MERZ, M; TU, T.; LAMERSDORF, W., Electronic Commerce – Technologische und organisatorische Grundlagen, in: Informatik Spektrum, Band 22, Heft 5, Springer Verlag, Heidelberg, 1999

[5] LINCKE, D.; ZIMMERMANN, H., Integrierte Standardanwendungen für Electronic Commerce, in: A. Hermanns, M. Sauter (ed.) Management Handbuch Electronic Commerce, Verlag Franz Vahlen,, München, 1999

[6] GEHMEYR, A., Electronic Commerce – Ein Überblick, in: Objekspektrum, 2/99, SIGS Conferences GMBH, 1999

[7] LEWANDOWSKI, S., Frameworks for Component-Based Client/Server Computing, in: ACM Computing Surveys, Vol. 30, No. 1, ACM Press, 1998

[8] CROCKER, D.H., RFC822: Standard for the Format of ARPA Internet Text Messages, ftp://ftp.isi.edu/in-notes/rfc822.txt (zitiert Mai 2000)

[9] NIELSON, J., The Difference Between Web Design and GUI Design, http://www.usit.com/alertbox/9705a.html (zitiert Mai 2000)

[10] FIELDING, R., et.al., RFC2616: Hypertext Transfer Protocol – HTTP 1.1, ftp://ftp.isi.edu/in-notes/rfc2616.txt (zitiert Mai 2000)

[11] SCHWICKERT, A., HTML - Hypertext Markup Language, in: Informatik Spektrum, Band 20, Heft 3, Springer Verlag Heidelberg, 1997

12 HALE, J. Business Object Modeling with aBCd, in: A. Carmichel, Developing Business Objects, SIGS Books, Cambridge University Press, UK, 1998

[13] BYRNE, R., Building Applications With Microsoft Outlook 2000, Technical Reference, Microsoft Press, 2000

[14] PIRONET, Pirobase System Architecture, http://www.pironet.com/servlet/Ib/Menu/ID=14210 (zitiert Mai 2000)

[15] SMARTSTORE, Smart Store Standar Edition 2.0: Eingesetzte Technologien, http.//www.smartstore.de/produkte/se20/techno.asp (zitiert Mai 2000)

[16] SUN, JavaMail 1.1.3, http.//java.sun.com/products/javamail (zitiert : Mai 2000)

[17] SUSE, SuSE Linux 6.3 (i386) – sendfax: sedfax part of mgetty, http://www.suse.de/de/produkte/susesoft/linux/Pakete (zitiert:Mai 2000)

[18] SUSE, SuSE Linux 6.3 (i386) – yaps: Yet Another Pager Software, http://www.suse.de/de/produkte/susesoft/linux/Pakete (zitiert:Mai 2000)

[19] BOOCH, G.; JACOBSON,I., RUMBAUGH, J.; The Unified Modeling Language User Guide, Addison Wesley, Reading, MA, USA, 1999

[20] BOOCH, G.; JACOBSON,I., RUMBAUGH, J.; The Unified Modeling Language Reference Manual, Addison Wesley, Reading, MA, USA, 1999

[21] COYLE, F., Legacy Integration – Changing Perspectives, in: IEEE Software, Vol. 17, No. 2, IEEE Computer Society Press, 2000

[22] FARSI,R. XML, in: Informatik Spektrum, Band 22, Heft 6, Springer Verlag, Heidelberg, 1999

[23] TOLKSDORF, R., XML und darauf basierende Standards: Die neuen Auszeichnungssprachen des Web, in: Informatik Spektrum Band 22, Heft 6, Springer Verlag, Heidelberg, 1999

[24] BÖHNLEIN, M.; ENDE, A.U.; XML – Extensible Markup Language, in: Wirtschaftsinformatik, 41. Jahrgang, Heft 3, Vieweg Verlag Wiesbaden 1999

[25] GOSLING, J.; JOY, B.; STEELE, G., The Java Language Specification, Addison Wesley, Reading, MA, USA, 1999

[26] KRUGLINSKI, D:J:, Inside Visual C++, Version 5, Microsoft Press, 1997