Realizing An Integrated Electronic Commerce Portal System

Matthias Book, WebService Haltern, mbook@ws-haltern.de
Volker Gruhn, Department of Computer Science, Dortmund University, gruhn@cs.uni-dortmund.de
Lothar Schöpe, ICD e.V., Dortmund, schoepe@icd.de

Abstract

This experience report[1] describes the design and implementation of an electronic commerce portal system for insurance companies which supports the insurance agents in their daily work. The electronic commerce portal, called IPSI, is used within an intranet. The insurance agents need information about their industrial and private customers and have to be supported in organizing their work by an organizer with reminder function, address management, etc. The insurance company has the goal to provide its employees with the most current information about its product portfolio and tariffs as well as wordings of the law and comments on it. An electronic commerce portal therefore has to provide a multitude of functions. The advantage of a portal lies in the integration of different software systems that provide these functions and thus leads the users directly to the desired information. They do not have to search awkwardly and are not served with irrelevant information. An object oriented design using UML, the realization of adequate adaptors for the integration of different heterogeneous software systems using the Java programming language, and the use of the middleware CORBA for communication within the portal were objectives of the project. This project was realized by the University of Dortmund in cooperation with different insurance and software companies. The electronic commerce portal was implemented as a prototype for a certain insurance company, including the integration of one of their legacy systems.

Introduction

Conventional business transactions – i.e. transactions not supported by information technology (IT) – are conducted nowadays by media like paper, telephone or fax (Zwass, 1996; Zwass, 1999). IT-supported business transactions use media like electronic mail, EDI, WWW and other Internet services (Chesher and Kaura, 1998; Conelly, 1997). On an abstract level, partners in business transactions – either electronic or conventional – are supplier and customer. In special businesses, however, they can be called supplier and consumer, addressee and provider, or producer and supplier but also management and employee.

These roles – supplier and customer – can be taken by companies, administrations or private persons. If the role of the supplier as well as the role of the customer is taken by a company, the business transaction is called business-to-business (B2B). If the role of the customer is taken by a private person, the business transaction is called business-to-consumer (B2C). Analogously, the roles can be taken by an administration. In that case, the business transactions are called administration-to-consumer (A2C), administration-to-administration (A2A) or business-to-administration (B2A). Business transactions within a company – between management and employees, without external partners – are called business-to-employee (B2E).

In electronic commerce as well as electronic business, suppliers and customers communicate electronically by means of a data communications network (Adam and Yesha, 1995). The Internet with its protocols (TCP/IP, FTP, NNTP, SMTP, HTTP, etc.) and services (Usenet, e-mail, WWW, etc.) represents such a data communications network.

The common aim of electronic business and electronic commerce is the electronic support of business transactions or market transactions. This is realized either by supporting the supply chain (ordering, billing, payment) between different suppliers or by supporting marketing, sales, distribution and after-sales support of products or services (Schmid and Lindemann, 1998) for customers. Not only services like stock exchange news, insurances or weather prognosis etc. can be supported by electronic commerce, but also communal administration services and tax declaration.

While electronic commerce primarily supports private customers, electronic business, on the contrary, does not involve private customers but supports electronic business transactions between companies, administrations or between management and employees. In this context, an employee is not seen as a private customer.

Any kind of electronical business transaction conducted between two partners is supported by one or more different software systems. Each partner of the electronic business transaction uses individual and specific, simple or complex software systems to support his own business transactions., e.g. SAP B2B-Procurement, EDI/EDIFACT for B2B or various shop systems for shopping or auctioning, which are partially based on Internet client/server techniques. The set of

specific software systems of all partners involved in an electronic commerce transaction form an electronic commerce system. To build such an electronic commerce system, these software systems can be integrated in a rather tight or more lose way. Thus, a shop system consisting of a web browser and a web server with heavyweight extensions is an electronic commerce / electronic business system, as are two connected EDI/WEB converters for commodity management systems.

In this context, an electronic commerce portal – i.e. an integration platform for different software systems like legacy, web, Internet or office systems – is also an electronic commerce / electronic business system. However, an electronic commerce portal which is used in an intranet supports only business-to-employee transactions (B2E). Communication between management and employees (e.g. agents of an insurance company), but also between different employees, is supported by providing information about the product portfolio, tariffs, customers and contacts within the electronic commerce portal and its subsystems. An additional feature for an intranet portal supporting business-to-employee transactions is the integration of the functionality of legacy systems. In contrast to internet portals, access to the services provided by the intranet portal is limited to a special user group (here: insurance agents).

Architecture of the IPSI portal system

During the information analysis of the IPSI project, it was recognized that the electronic commerce portal serves as an integration platform for different heterogeneous subsystems. Based on a 3-tier-architecture, the user interface and data repository are separated from the functional application logic (Lewandowski, 1998). On the level of the functional application logic, the following subsystems of an electronic commerce portal have been identified:

Office System:[2] The office system manages contact addresses and scheduled appointments. For addresses, remote data and local data are distinguished: While remote data is managed by the partner management system of the insurance company, local data is managed by an office system on the user's computer in order to satisfy his privacy requirements.

Content Management System: Information of any kind is supplied by the content management system. Each employee of a company (e.g. management, back office employees or agents of the insurance company) can

provide information for all others. Governed by individual access rights, every employee can get information from or put information into the content management system for every other employee (e.g. new product portfolio, handbooks, marketing materials, comments to the law, decisions in the context of insurances, etc.). The content management system will organize this information using different views and access rights.

Procurement System: The procurement system offers consumer goods (e.g. laser printers, toner, pencils, etc.) and services (e.g. courses, trainings, seminars, etc.). Every insurance agent can order consumer goods for his daily work. The management can monitor the orders and control the costs caused by the insurance agents.

Communications System: The communications system represents the interface to telecommunications media like mobile phones, fax and e-mail. The communications system is able to send documents, notifications or reminders by e-mail, Short Message Service (SMS) or fax. Notifications and reminders are sent at a pre-defined time set by the office system.

Portal Administration System: The portal administration system serves as the single point of login, i.e. the user of the electronic commerce portal does not need to authorize himself at each subsystem of the portal. The second purpose of the portal administration system is the analyzation and presentation of the log files of the subsystems.

Search System: The search system allows the user to search for information in the entire electronic commerce portal, based either on full text scan retrieval or predefined keywords. The results of a search request can be appointments, addresses of customers, information from the content management system, ordered goods or a combination of these elements.

Legacy System: A legacy system is an external system not included in, but connected to the electronic commerce portal. Legacy systems are realized as host applications.

The portal user interface consists of web pages written in Hypertext Markup Language (HTML). For data management, a relational database management system is used if the subsystems do not have their own repository. Now, let's take a closer look at the system architecture (Figure 1):

---

[2] The management of addresses is realized by a traditional host system like IBM MVS (for remote data) and additionally by a local office system like Lotus Organizer or Microsoft Outlook (for local data). Access to the remote data is provided by the electronic commerce portal via an XML interface. The synchronization of remote and local data is also guaranteed by the electronic commerce portal.

Figure 1. System Architecture
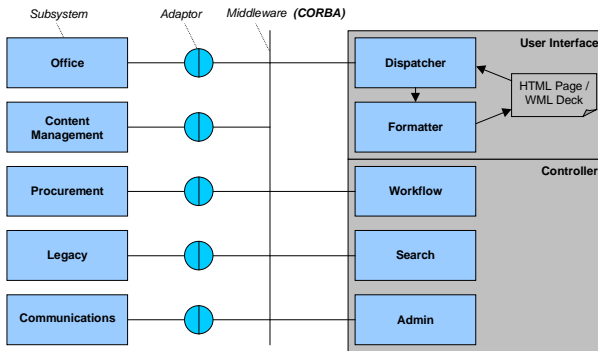
Office, content management, procurement, legacy and communications are all external systems. To avoid building these from scratch, it was decided to integrate existing solutions into the electronic commerce portal.

Since the interfaces used to access the external systems are very different, each one is connected to the central middleware „backbone" via an individual adaptor. Each adaptor provides a set of methods to the middleware that encapsulates the native interface of the external system. This way, the (possibly complicated) native interface does not need to be publicly known in order to access its functionality. Instead, other subsystems can simply use the methods provided by the adaptor. For example, to send an e-mail via the communications system, it is sufficient to call the respective method of the communications adaptor which will then take care of constructing a RFC822-compliant message (Crocker, 1982) from the supplied parameters, setting up a session with the SMTP server and sending the e-mail. Furthermore, the encapsulation allows for an easy change of external systems: If a system's native interface changes, only its own adaptor must be rewritten while all other subsystems remain untouched.

The user interacts with the electronic commerce portal primarily via a web browser (other user agents such as mobile phones are also allowed by the system architecture). This has important implications for the control flow within the system: In traditional software systems, the dialog can be controlled by the system to a large extent: For example, the system can open a modal dialog box at any time, forcing the user to take some specific action before he can do anything else (Nielsen, 1997). On the web, however, all actions are initiated by the user. The server cannot push information to the browser that the user did not request.[3]

---

[3] This is true for a user interface built from plain HTML pages. Of course, one might conceive a client-side Java applet displaying information pushed to it by the server. However, this would require a Java-capable user agent, ruling out most of the currently available mobile agents like WAP phones, organizers etc. Plain HTML, on the other hand, makes the least assumptions about the target platform, and the subsystems producing it can easily be adapted to generate similar formats like Wireless Markup Language (WML).

Consequently, the external systems (office, content management etc.) of the electronic commerce portal remain passive and act only on user requests passed to them via the path depicted in Figure 2:

Figure 2. Communication within the Electronic Commerce Portal
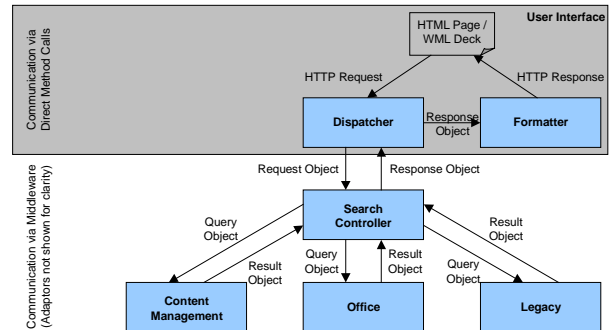
Every user action like clicking on a link or submitting a form generates an HTTP request (Fielding et al., 1999) which is received by a central dispatcher. The dispatcher parses the HTTP request string, builds a request object from its contents and passes it to the controller that is responsible for handling the requested task. The search controller and admin controller implement the functionality of the search and portal administration systems mentioned earlier; all other transactions involving the external systems are handled by the workflow controller.

The controllers might be considered the brains of the electronic commerce portal: They evaluate the request objects passed by the dispatcher. Depending on the type of request, they send commands to or query information from the external systems, consolidate the results and return them to the dispatcher. To achieve this, the specific workflow necessary to fulfill any given request is hard-coded into the respective controller. For example, upon receiving a request to search for a particular person in all the external systems, the search controller queries the office, content management and legacy systems and returns the combined results to the dispatcher.

The dispatcher then forwards the response object received from the controller to the formatter. This subsystem is responsible for converting the information contained in the response object into a format the user agent can render. In most situations, the preferred output format will be Hypertext Markup Language (HTML) (Pemberton et al., 2000) which is accessible with a wide range of user agents. For more exotic user agents such as WAP phones and organizers, other formatters can generate output formats like Wireless Markup Language (WML) (WAP Forum, 1999). This flexibility is one main advantage of the separation between formatters and controllers: Since the implementation of the user interface is concentrated in one dedicated system, the visual

presentation of information can be changed or expanded without touching any of the systems actually providing the information.

Because of performance considerations and special system requirements, most external subsystems and the web server run on separate computers. This distributed architecture requires a middleware like CORBA to coordinate the calling of methods and passing of objects among the different subsystems. Of course, using the middleware is not necesssary within single subsystems such as the user interface: For example, the dispatcher calls a method of the formatter directly to pass a response object received from a controller.

The dispatcher and the controllers, however, might run on different machines. Thus, they exchange objects via the middleware. Two models of communication were considered during the design phase of the project:

1. **Publisher/Subscriber Model:** The dispatcher publishes a request object via the middleware and announces its availability with an event that describes the type of request. Controllers can subscribe to events that are relevant to them and get a copy of the respective request object from the middleware.

2. **Explicit Call Model:** Based on the type of request, the dispatcher decides which controller(s) it must call to pass the request object to via the middleware.

In the publisher/subscriber model, the dispatcher is effectively reduced to a mechanism for converting HTTP request strings to request objects since it does not know which controller is responsible for which type of request. While this may at first seem like an elegant decoupling, there are some pitfalls: Although the "sending" part of the dispatcher does not need to be changed when a new controller is added to the subscriber list, the "receiving" part must still be prepared to accept result objects from the additional controller. Regarding the effort for defining interfaces between the dispatcher and the controllers, the publisher/subscriber model holds no advantage over the explicit call model: Both dispatcher and controllers need to know which attributes are defined for request objects of any type, regardless of the means by which the objects are transported. More problems arise from the multi-user environment of the electronic commerce portal: The dispatcher needs to know which result object returned by the controller corresponds to which request object passed to it. In the explicit call model, this mapping is implicitly provided by the call stack of the middleware. In the publisher/subscriber model, each request object (and the objects passed between controllers and subsystems) would have to be tagged with a unique identifier in order to track the incoming result objects – an unnecessary overhead.

Controllers and subsystems communicate by exchanging "business objects", i.e. entities that are central to the workflow in the electronic commerce portal. The following business objects are therefore known to all controllers and subsystems:

- User
- Contact
- Appointment
- Task
- Message
- Shop Item
- Order
- Order History
- Search Request
- Search Result

To schedule an appointment, for example, the workflow controller creates an appointment object from the data received by the dispatcher and passes it to a method of the office subsystem that adds the appointment to the user's calendar. If the user chooses to be reminded of the appointment by e-mail in time, the workflow controller additionally creates a message object, connects a copy of the appointment object to it and passes it to the communications system which will queue it for e-mail delivery at the time requested by the user.

Realization

The first phase in the process of realizing the electronic commerce portal was an analysis of the content and function requirements. To gain insight into the portal users' needs, the project team visited several insurance companies. Through demonstrations of systems currently used by insurance agents and discussions with developers, the team learned about the typical tasks an insurance agent performs in his daily work and how these can be supported by software solutions. The results of the analysis were organized by breaking the more comprehensive tasks down into singular activities which were then prioritized and documented in requirement forms.

Based on the requirement forms, the subsystems office, content management, procurement, communications, legacy, search and administration were identified. For each of these subsystems, a make-or-buy decision had to be made. After evaluating existing solutions and considering the effort for developing a subsystem from scratch vs. integrating the existing solution, the team chose the integrative approach for most systems, namely:

- **Office:** Outlook 98 by Microsoft Corporation (Byrne, 1999)
- **Content Management:** Pirobase 4.0 by PiroNet AG (Pironet, 2000)
- **Procurement:** SmartStore Standard Edition 2.0 by SmartStore AG (SmartStore, 2000)

- **Communications:**
  - *e-mail:* JavaMail by Sun Microsystems, Inc. (Sun, 2000)
  - *Fax:* sendfax – Freeware; included in Linux 6.3 (i368) by SuSE GmbH (SuSe, 1999)
  - *SMS:* yaps – Freeware; included in Linux 6.3 (i368) by SuSE GmbH (SuSe, 1999)
- **Legacy:** Sample partner database of the Continentale Versicherung

The search and administration systems were not classified as external systems but as controllers since they actively request or modify information of the other systems.

To test the feasibility of these decisions, the team programmed cut-through prototypes, i.e. "quick-and-dirty" implementations of the adaptors described in the system architecture. The goal of these prototypes was to prove that it is possible to encapsulate the native interfaces of the external systems and make their key features accessible via the adaptors. This goal was met for all subsystems, clearing the way for the next phase of the software development process.

For the object oriented design phase, the team used the Unified Modeling Language (UML) (Booch et al., 1999). The key features of all subsystems were modeled in use cases in order to identify business objects and possible dependencies between the subsystems. Based on the insights gained in this step, concrete classes were defined for each subsystem. To ensure an easy consolidation of the results and allow for later changes to the subsystems without touching any dependent classes, each subsystem is represented at the "outside" by one boundary class. This class provides all methods other classes need to access the subsystem. As an example, let's consider a search request handled by the legacy system (Figure 3):

Figure 3. Integration of Legacy System
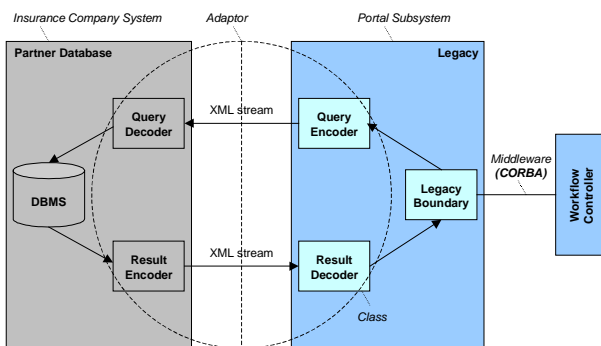
The large box in the middle is a view inside the legacy subsystem that we know from previous figures. The smaller boxes inside represent classes. Because only the legacy boundary class is connected to the workflow controller via the middleware, in our example the controller does not pass the search request object directly to the query encoder. Instead, the search request is passed to the legacy boundary class which then passes it to the query encoder. This class is a part of the adaptor that, as discussed earlier, hides the native interface of the external system from the portal subsystem: In the case of the legacy system, queries and results of the insurance company's partner database are XML-encoded (Bray et al., 1998) for maximum platform and transport independence. The XML-encoded search query is run against the insurance company's database, and the encoded result is returned to the legacy subsystem where the result decoder (another part of the adaptor) creates a search result object and passes it to the legacy boundary class, which returns it to the workflow controller.

After consolidation of the designs for subsystems, controllers and user interface, the team entered the implementation phase. Most classes were implemented in the Java programming language (Gosling et al., 1996), only the adaptor for the office system uses Microsoft Visual C++ (Kruglinski, 1997) code to access the Microsoft Outlook 98 API.

Figure 4 shows the homepage of the electronic commerce portal. After logging into the system, the insurance agent is presented with all information that is relevant to him that time: Personal messages, articles of interest from the content management system, scheduled appointments and due tasks from the office system, events and items from the procurement system. Legacy applications like the partner database and a provisioning system are accessible via links on the homepage. A search interface allows for meta searches in selected areas of the portal.

Figure 4. Electronic Commerce Portal Homepage
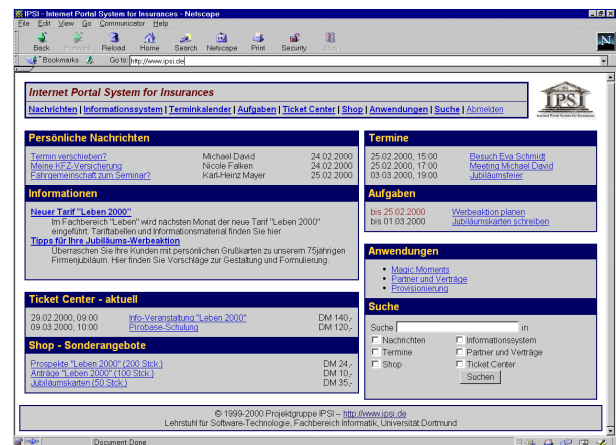
Conclusion

In building the IPSI system we had to recognize that the implementation of a portal system is an integration engineering task. This had an important impact onto the software process deployed. Backend integration is based

on middleware, frontend integration is based on a commonly used user interface which called for careful design.

Most requirements for IPSI were fulfilled by integrating standard tools. In order to effectively plan the software process for building IPSI, it was crucial to use prototypes (compare above). Only after implementing these prototypes we were able to assess the feasibilty of the architecture and only then we were able to calculate duration of the tasks identified and efforts needed for these tasks. The productive use of IPSI showed that the openness of the architecture is a crucial issue. Many further legacy systems had to be added after the initial release, standard tools were exchanged for individual customers. All these modifications depend on a clear and modular architecture. With hindsight, it would have been useful to develop IPSI as a component-based system on the basis of a standard component model like JavaBeans or COM.

Summing this up, the effort for implementing was lower than initially expected, simply because we were able to benefit from standard tools. The kind of tasks was different from what was initially planned, more tasks than initially planned were integration tasks. In the end only a few thousand lines of code were written, but this software was used as glue between existing systems and therefore required extremely detailed design and careful testing.

References

Adam, N. and Yesha, Y. (eds.) "*Electronic Commerce*", LNCS (1028), Springer Verlag, Berlin, 1995

Adam, N. and Yesha, Y. "*Electronic Commerce: An Overview*" in *Electronic Commerce*, Adam, N. and Yesha. Y. (eds.) LNCS (1028), Springer Verlag, Berlin, 1995

Booch, G., Jacobson, I. and Rumbaugh, J. *The Unified Modeling Language User Guide,* Addison Wesley, Reading, MA, 1999

Bray, T., Paoli, J. and Sperberg-McQueen, C.M. *Extensible Markup Language (XML) 1.0, W3C Recommendation 10 February 1998,* http://www.w3.org/TR/1998/REC-xml-19980210 (current March 2000)

Byrne, R. *Building Applications With Microsoft Outlook 2000 Technical Reference,* Microsoft Press, 1999

Chesher, M. and Kaura, R. *Electronic Commerce and Business Communications,* Springer Verlag, Berlin, Germany, 1998

Connelly, D.W. "*An Evaluation of the World Wide Web as a Platform for Electronic Commerce*" in *Readings in Electronic Commerce*, Kalakota, R. and Whinston, A. (eds.), Addison Wesley, Reading, MA, 1999

Crocker, D.H. *RFC822: Standard for the Format of ARPA Internet Text Messages,* ftp://ftp.isi.edu/in-notes/rfc822.txt (current March 2000)

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T. *RFC2616: Hypertext Transfer Protocol – HTTP 1.1,* ftp://ftp.isi.edu/in-notes/rfc2616.txt (current March 2000)

Gosling, J., Joy, B. and Steele, G. *The Java Language Specification,* Addison Wesley, Reading, MA, 1996

Kalakota, R. and Whinston, A. (eds.) *Readings in Electronic Commerce,* Addison Wesley, Reading, MA, 1999

Kruglinski, D.J. *Inside Visual C++ Version 5,* Microsoft Press, 1997

Lewandowski, S. "*Frameworks for Component-Based Client/Server Computing*" in ACM Computing Surveys, (30:1), 1998

Nielsen, J. *The Difference Between Web Design and GUI Design,* Alertbox for May 1, 1997, http://www.useit.com/alertbox/9705a.html (current March 2000)

Pemberton, S. et al. *XHTML™ 1.0: The Extensible HyperText Markup Language. A Reformulation of HTML 4 in XML 1.0, W3C Recommendation 26 January 2000,* http://www.w3.org/TR/2000/REC-xhtml1-20000126 (current March 2000)

PiroNet AG. *pirobase® System Architecture,* http://www.pironet.com/servlet/IbMenu/ID=14210 (current March 2000)

Riggins, F. and Rhee, H. "*Toward a unified view of electronic commerce*" in CACM (41:10), 1998

Schmid, B. and Lindemann, M. "*Elements of a Reference Model for Electronic Markets*" in 31st HICSS, IEEE Press, 1998

SmartStore AG. *SmartStore Standard Edition 2.0: Eingesetzte Technologien,* http://www.smartstore.de/produkte/se20/techno.asp (current March 2000)

Sun Microsystems, Inc. *JavaMail 1.1.3™ Release,* http://java.sun.com/products/javamail/ (current March 2000)

SuSE GmbH. *SuSE Linux 6.3 (i386) – November 1999 "sendfax": sendfax part of mgetty,* http://www.suse.de/en/produkte/susesoft/linux/Pakete/paket_sendfax.html (current March 2000)

SuSE GmbH. *SuSE Linux 6.3 (i386) – November 1999 "yaps": Yet Another Pager Software,* http://www.suse.de/de/produkte/susesoft/linux/Pakete/paket_yaps.html (current March 2000)

WAP Forum. *Wireless Application Protocol: Wireless Markup Language Specification, Version 1.1, 16 June 1999,* http://www1.wapforum.org/tech/documents/SPEC-WML-19991104.pdf (current March 2000)

Zwass, V. "*Electronic Commerce: Structures and Issues*" in IJEC (1:1), 1996

Zwass, V. "*Structure and Macro-Level Impacts of Electronic Commerce: From Technological Infrastructure to Electronic Marketplaces*" in Kendall, K.E. *Emerging Information Technology,* Sage Publications, 1999