Development of an Electronic Commerce Portal System using a Specific Software Development Process

Volker Gruhn[1], Martin Mocker[2], Lothar Schöpe[3]

[1] Universität Dortmund, Fachbereich Informatik, Lehrstuhl Software-Technologie
Baroper Str. 301, D-44227 Dortmund, Germany
volker.gruhn@uni-dortmund.de

[2] adesso AG
Stockholmer Allee 24, D-44269 Dortmund, Germany
martin.mocker@adesso.de

[3] Informatik Centrum Dortmund e.V., Abt. Software-Technik
Joseph-v.-Fraunhofer-Str. 20, D-44227 Dortmund, Germany
lothar.schoepe@icd.de

Abstract

The development of electronic commerce (EC) systems is subject to different conditions than that of conventional software systems. Consequently, software development processes used for conventional systems to date need to be adapted to these new conditions. This includes the introduction of new activities to the development process and the removal of others. In addition, the roles involved in the development process, their tasks, qualifications and the software tools used by them, are different to other processes. An adapted process must cope with important idiosyncrasies of EC system development: EC systems typically have a high degree of interaction, which makes factors like ergonomics, didactics and psychology especially important in the development of user interfaces. Typically, they also have a high degree of integration with existing software systems such as legacy or groupware systems. Integration techniques have to be selected systematically in order not to endanger the whole software development process. Furthermore, the approach to the development of EC systems should take into account the "time-to-market" factor and allow development time reduction while retaining quality. This paper introduces and describes an adapted software development process for EC systems and its special features using the development of an EC portal system as an example.

Keywords: Electronic commerce, software development process, application integration, component based development, distributed architecture design.

1 Introduction

In this paper, EC is defined as conducting transactions of any kind by means of electronic media, especially the Internet. The roles of suppliers and customers in these transactions can be adopted by different parties, such as consumers (C), administrations (A), businesses (B) or even their employees (E) (Shaw 2000). Depending on the type of parties participating in an EC transaction, one is talking about B2B, B2C, B2E, B2A, C2C EC etc. The parties involved in EC transactions use information technology (IT) systems to automate their transactions (Chesher, and Kaura 1998). The examples used in this paper are based on B2E EC, where an EC portal system is used for automation but can also be transferred to other types of EC. An EC portal system is an integration platform for different software systems: conventional (i.e. non EC) software systems such as legacy and office systems as well as other EC based systems such as shop systems.

In the same way that conventional application software systems are developed according to conventional software development processes, special software development processes are necessary to delineate the development of EC systems (Bayer, Junginger, and Kühn 2000, Harrison, Ossher, and Tarr 2000, Gruhn, and Schöpe 2001, Haire, Henderson-Sellers, and Lowe 2001). These software development processes for EC systems differ from those for conventional application software systems in the following aspects:

- They include provisions for new or adapted activities and roles performing them:
  - System integration is very important in EC settings, because often, many heterogeneous systems have to be integrated and these do not necessarily have a long lifetime. Therefore the add-on or replacement of components should be planned beforehand.
  - The need for attractive and user-friendly user interfaces is very pressing. One reason for this is that often, users are of various kinds with differing backgrounds, not known personally making it difficult to obtain feedback. Roles for graphical design activities are needed in order to provide these user interfaces.
  - Peculiarities in EC customer behaviour make workload hard to plan. Therefore, performance

planning has to be considered strongly for EC systems. Roles and activities that deal with these issues are needed.

  - Content is an integral part of most EC systems and important regarding quality, quantity and frequency of change. Roles and activities for managing this content are needed.

- They must take into account that EC software development is a very distributed process. That is, the roles mentioned above are often adopted by different parties: software companies develop software components, multimedia companies develop graphical features for the interface and specialized content providers supply the content.
- They have to cope with high time pressure expressed in a shorter time-to-market, yet at the same time preserving the required quality, especially regarding non-functional requirements such as extensibility.

These aspects are now discussed in more detail:

In EC system settings, many predefined building blocks, like shop systems, content management systems etc., are often developed by small software suppliers with special expertise in their field. Companies generally do not want to be dependent on these small suppliers. So, when designing an EC system, one should have in mind that some components might be replaced and others might be added at a later point in time. Thus, focus should be put on system integration activities during the software development from an early stage on. If an EC system needs to be integrated into an existing infrastructure, the requisite methods, concepts and software tools for the integration must be available (Noffsinger, Niedbalski, Blanks, and Emmart 1998). The methods, concepts and software tools used, as well as the software developers involved, depend on how the integration is undertaken. For example, security aspects (use of firewalls, cryptography etc.) might have to be taken into account. These security aspects then not only have to be considered during implementation, but also during the design of the EC system. Also, it must be decided if direct sales processes for products or services should be electronically supported by an EC system. If support for products is required, the EC system has to be integrated with an open or closed inventory control system of the merchant. Integration with conventional domain-specific, highly individual application software systems is also usually required when supporting direct sales processes for services. These individual application software systems, termed 'legacy systems', are used by all kinds of businesses such as insurance companies, governmental agencies, banks, power utilities etc. (Lamond, and Edelheit 1999).

The functionality of inventory control systems is often covered by ERP systems of different software manufacturers (e.g. SAP, Oracle, Baan, Sage, PeopleSoft etc.). These ERP solutions provide interfaces (APIs) for integration with other software systems. For example, this makes it possible to offer integrated solutions between Intershop and SAP, OpenShop and Sage or Oracle. In some circumstances, the integration of several different inventory systems or individual application software systems may be necessary. This is usually the case for the implementation of EC malls or EC portals.

While conventional application software systems might win user acceptance mainly through their functionality and can be positioned against market competitors in that way, a special class of EC systems (e.g. shop systems) also have to win user (i.e. customer) acceptance via the user interface. The user interface not only presents content in a certain layout, but also guides and supports the user. The tasks concerning the selection of content and its presentation are not included in most conventional software development processes. The roles performing them are specialists for software ergonomics, didactics, graphical design and psychology.

Performance is a second factor influencing user acceptance of EC systems. This becomes more evident when looking at the negative effects of an EC system with poor performance: users tend to quit their visit to a site after waiting for 8-15 seconds (Nielsen 2000) for a response, resulting in loss of revenue and image for the company associated with the site. Therefore, characterization of customer behaviour, workload forecasting and performance modelling become very prominent activities. Two characteristics of EC customer behaviour aggravate workload characterization in EC settings: peak-like request bursts and high-volume data requests that are not typically found in conventional software systems (Menascé, and Almeida 2000).

Another major factor in acceptance of many EC systems (e.g. shop systems) is being up-to-date – not only regarding the content, but equally important, content presentation. In most conventional software application systems, data of different types and structures is managed and processed in different ways. The more data managed by the application software system, the more up-to-date it is. In addition, for a shop system to stay up-to-date, the presentation of its content must be kept up-to-date. This means that even if the data remains mostly unchanged, its presentation is subject to change over time. In the productive/maintenance phase, the functionality of a shop system may remain largely constant, while the presentation of the content is modified and adapted at certain time intervals by specialists for software ergonomics, didactics, graphic design and psychology. Extensive statistical testing permits the measurement of customer acceptance levels with time. And from these statistics, it can be deduced which parts of the presentation should be changed.

The roles involved in the development of an EC system are more specialized and more widely spread between participating suppliers than is normally the case with conventional software systems development. Some of the roles and their activities have already been mentioned: specialists for software ergonomics, didactics, screen design and psychology, performance engineers, content engineers and software developers with expertise in a multitude of technologies such as programming languages like Java, component models and other frameworks such as Enterprise Java Beans, Servlets or

Java Server Pages, or middleware at different levels such as XML, SOAP and RMI. In most cases, this variety of required skills is not found in one single supplier such as a software company. Collaboration between many suppliers with specialized skills such as multimedia design companies, software companies including freelancers as experts and content providers is far more likely to be the case. A process for the development of EC systems has to take this distribution into account, by considering contract settlement (legal and technical in terms of interface contracts) and means for easing communication between suppliers.

Depending on the course of action within the software development process, the different roles use different software tools, such as shop systems (Intershop, Openshop etc.), content management systems (Hyperwave, Firstspirit, Pirobase etc.) or software development/programming environments (JBuilder, Together J etc.).

As argued previously, when developing EC systems, a special software development process is needed to take into account these factors. This paper presents such a process that has been defined during the development of a B2E EC portal system: this portal system is presented in section 2 and demonstrates some of the above-mentioned features of EC systems stimulating the demand for adapted software development processes. Section 3 describes the actual process that dealt with these features and resulted in the portal system. Section 4 summarizes the main aspects and draws conclusions from the work on processes suited to support the development of EC systems.

2 The IPSI Electronic Commerce Portal

An EC portal for insurances was designed and implemented as part of a software engineering project (Book, Gruhn, and Schöpe 2000). This portal – called Internet Portal System for Insurances (IPSI) – is intended to provide support for insurance agents with their daily work. The main goal of the portal is to support business-to-employee (B2E) processes (Lincke, and Zimmermann 1999). Thus, the communication between management and employees (in this case between an insurance company and its agents), but also between employees themselves is supported by providing information about the product portfolio, tariffs and customer contacts via the EC portal and its subsystems. This portal system demonstrates some of the idiosyncrasies of EC systems that generate the demand for adapted software development processes.

During the requirements analysis phase of the project, it was recognized that the EC portal serves as an integration platform for different heterogeneous subsystems (Hasselbring, Koschel, and Mester 2001). Based on an n-tier-architecture, the user interface and data repository[1] are separated from the functional business logic (Lewandowski 1998) that resides in multiple application components (called subsystems). At the functional business logic level, the following subsystems of an EC portal were identified, which show the need for focusing on integrating many different systems:

Office System: The office system manages any agent's customer contact addresses and scheduled appointments. For addresses, a distinction between remote and local data is made. While remote data is managed by the partner management system of the insurance company, local data is managed by an office system on the agent's computer, to satisfy his or her privacy requirements.

Content Management System: Information of any kind is supplied by the content management system. Each insurance company employee (e.g. management, back office employees or agents) can provide information for all other participants. Based on individual access rights, employees can retrieve information (e.g. new product portfolio, handbooks, marketing materials, comments on legal decisions in the context of insurances etc.) from or store information in the content management system for every other employee. The content management system organizes this information using different views and access rights.

Procurement System: The procurement system offers consumer goods (e.g. computer equipment, books or writing material) and services (e.g. training courses). Every insurance agent can order consumer goods for his daily work. Management can monitor the orders and control the costs generated by their agents.

Communications System: The communications system represents the interface to telecommunications media (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 any user-defined point of time set by the office system.

Portal Administration System: The portal administration system serves as the administration center and therefore provides functions to add, update or delete portal user data and other administrative features. The administration system allows for a single-sign-on, i.e. EC portal users do not need to authorize themselves at each subsystem of the portal separately. The second purpose of the portal administration system is the analysis and presentation of logging information provided by the subsystems.

Search System: The search system allows the user to search for information in the entire portal, based either on full text search or predefined keywords. The result of a search request can include appointments, customer addresses, documents from the content management

[1] The management of data (e.g personal addresses) is taken over by a traditional host system like IBM MVS (for remote data) and additionally by a local office system like Lotus Notes or Microsoft Outlook (for local data). Access to 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.

system, goods ordered or a combination of these elements.

Legacy System: A legacy system is any external system already existing at the provider's (in this example the insurance company) site, which has to be connected to the EC portal. Legacy systems are often implemented as host applications (Coyle 2000), such as a partner management system storing contract data of people insured in the case of IPSI.

The portal user interface consists of Web pages written in Hypertext Markup Language (HTML). For management of administrative data, a relational database management system is used in addition to the subsystems' own repositories.

The system architecture had to fulfil several non-functional requirements pivotal to most EC systems, but especially to portals integrating many different systems. Among the most important requirements were the following:

- Not being dependent on the output medium (HTML, WML etc.). In other words, switching from one medium to another should be possible without significant porting efforts. Additionally, when changing the user interface, business logic should remain untouched, allowing for the distribution of development between user interface specialists and software developers. Therefore, presentation logic had to be separated from business logic.
- Being extendable in functionality. Once the core system was developed, it had to be easy for developers to add portal functionality without deeper knowledge of the inner operation of the system. Therefore, implementation details of middleware technology had to be hidden from the application developers.
- Being able to integrate several existing systems seamlessly. Not only should developers be able to add portal functionality for already integrated systems later, but they should also be able to connect other systems completely unknown at the time of architecture design to provide access to these systems via the web through the portal. This aspect is also important when exchanging systems with equal functionality (e.g. when updating to a new version of a shop system or when switching to a different shop system manufacturer).

These requirements led to the development of the system architecture depicted in Figure 1.

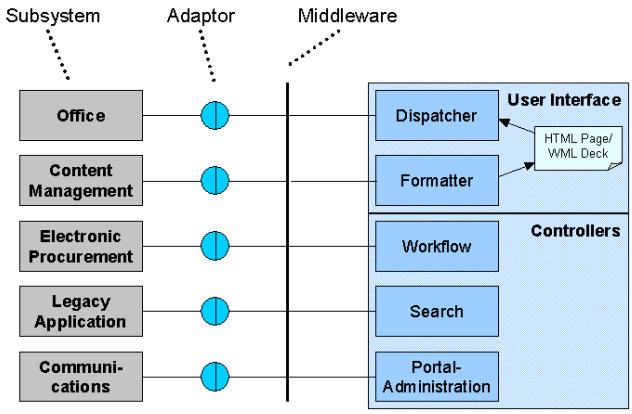

Figure 1: System Architecture

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 EC portal.

Book, Gruhn, and Schöpe (2000) describe the architecture of the portal system in detail. Only a short overview is given here, with special focus on the previously mentioned requirements of EC systems.

The user interacts with the EC portal via a Web browser (system architecture also allows other user agents such as mobile phones). The actual "work" of the system is done by the subsystems: the office subsystem stores the agent's contacts, appointments, tasks etc., the content management system manages all the published data, the legacy systems handles contract data and so on. To connect the subsystems to the rest of the application while hiding the specifics of any subsystem, we used adaptors acting as a façade of a subsystem. If a subsystem is replaced or a new subsystem is added, only the adaptors have to be replaced.

To be able to add new functionality (which can be the case even if one is not changing subsystems), a highly configurable dispatcher-controller mechanism using the Java Reflection API was utilized. In this setting, the dispatcher is responsible for locating a controller able to handle the user's request. A controller implements the workflow necessary to fulfil one request (Hoffner, Ludwig, Grefen, and Aberer 2001), especially by interacting with the subsystems' adaptor interfaces. Controllers and subsystem adaptors communicate by exchanging business objects (Baker, and Geraghty 1998) i.e. entities that are central to the EC portal's workflow. 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

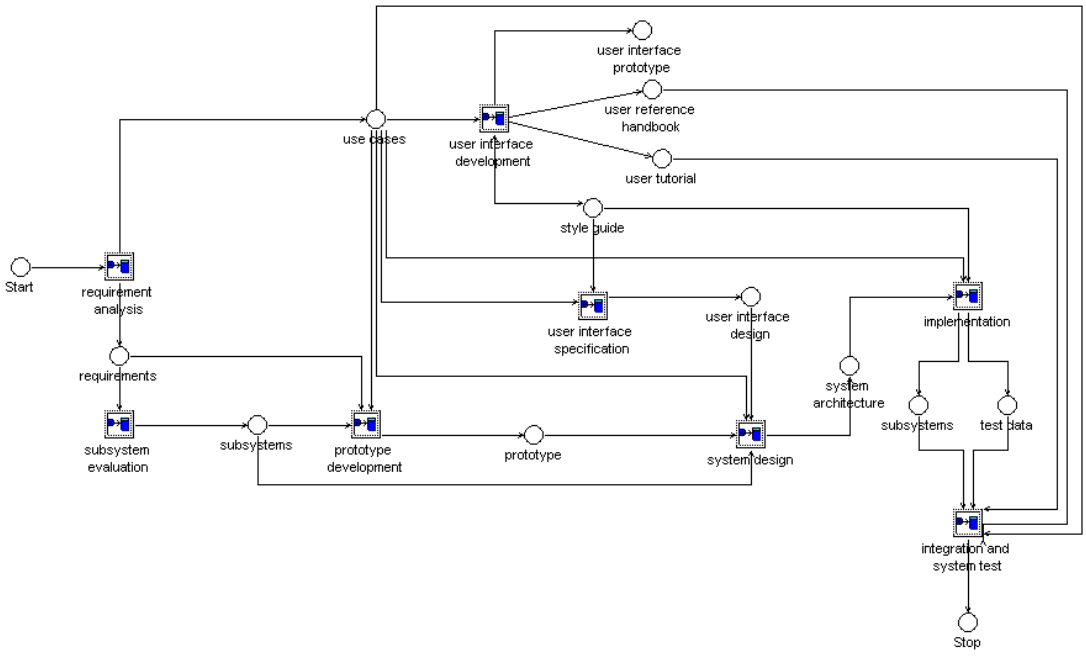

Figure 2: Electronic commerce portal development process model

To schedule an appointment, for example, the respective workflow controller creates an appointment object from data received by the dispatcher and passes it to a method of the office subsystem (or to be precise: the subsystem's adaptor) that adds the appointment to the user's calendar. If the user has chosen 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.

To separate the business logic contained in the controllers and maintained by software developers from the presentation logic maintained by user interface specialists, we employed a controller-formatter mechanism. The source of the user's request (e.g. a Web browser) determines the output medium and tells the dispatcher which formatter (e.g. a WML- vs. a HTML-formatter) to call after the controller finished its task. When changing the graphical user interface or adding a new output medium, only the formatters need to be modified by the design specialist, leaving the business logic completely unchanged.

To cope with performance considerations and other technical system requirements, most external subsystems and the Web server run on separate computers. This distributed architecture requires a middleware like RMI to coordinate the invocation of methods and passing of objects among the different components.

3 Process Description

The software development process for the development of a certain IT system is defined by a process model. A process model presents all the activities (in a certain order), the required tools and the created intermediate or final products necessary to achieve the process's purpose. A process model is usually tailored to a certain development project. A process, on the other hand, is the execution of a process model (in the object-oriented way of thinking, a process is an instance of a process model), i.e. the activities that are delineated in the process model are actually performed.

Although a formal specification of a software development process in the form of a process model simplifies its support by workflow systems, it is not mandatory for achieving a positive effect in software development. In order to reach consensus about the software development process among all those involved, an informal though structured and comprehensive description can be sufficient. A company's knowledge of best practices was and is often described in internal documents and development guidelines. For example, ISO 9000 (part 1-3) defines only the contents of the description of best practices and development guidelines, but not their notation. However, informal specifications relying on natural language bear the danger of misinterpretation because they usually have enormous volume, and some concepts, dependencies and prerequisites cannot always be formulated precisely.

The process model for the development of an actual EC system – namely the IPSI electronic commerce portal – is presented schematically in Figure 2, using the Funsoft net notation (Deiters, and Gruhn 1994). In order to reduce the complexity of presentation and increase the number of levels of abstraction, this notation allows distinction between elementary tasks (e.g. write story book, perform test data creation) and subprocess models (e.g. requirements analysis, subsystem selection, prototype development), which can again contain elementary tasks and subprocess models.

The object-oriented design using UML, prototype development, implementation of adaptors to integrate software systems as subsystems of the EC portal using the Java programming language and the use of a middleware (CORBA/RMI) for communication within the portal are all represented in this software development process. The development process also shows that the use cases described in UML are an important prerequisite for several sub process models such as the user interface specification and development.

In the following, the development process of the IPSI electronic commerce portal is described in reduced form with reference to the subprocess models, but not their internal details. The subprocess models for system design and implementation are not described, because in contrast to other activities, they did not show as many electronic commerce-specific deviations from the design and implementation activities of conventional development processes.

3.1 Requirements Analysis

Requirements analysis starts with a competition analysis, subsequent proposal and contract evaluation, and project initialization. After this, the functional and non-functional requirements for the EC software system are identified.

For the development of the EC portal for insurance agents, a competition analysis should determine if other software companies already offered a similar portal and which target groups those companies aimed at. Afterwards, the product idea was presented to several insurance companies, and one insurance company was won as a partner and potential client. During the proposal and contract evaluation, the feasibility of the client's requirements was clarified. The goal was a contract basis that was stable in every regard (content-wise, legal, mercantile) and the creation of a basis on which a software system could be developed that met the client's functional and technical requirements.

Identification and description of the portal's functionality and the priority-based structuring of these functions are very important tasks. Functionality must be sufficient to cover all client requirements, and yet must offer something unique compared to competitors' offers in order to gain competitive advantage. It must also contain opportunities for further development to ensure future competitive advantage.

High product functionality can be used to secure market advantage over competitor products. However, the realization of high functionality requires a certain effort, mirrored in the amount of time it takes to realize an EC system. Thus, advantage can also be gained by securing early market appearance of the EC system ("first mover advantage"). This means that, according to the "time-to-market" concept, EC software systems in particular need to be quickly developed and introduced to the market. The identification of requirements and the assignment of priorities to those requirements with attention to their impact on development time is a highly significant task when developing EC systems. To tackle the time-to-market problem, the bifocal approach proposed by Laartz, Scherdin, Cafarelli, and Hjartar (2000) can be used. This approach suggests building an EC system in two stages: first, requirements that are considered most critical regarding user acceptance (e.g. those requirements already covered by competitors identified during the competition analysis) are implemented in a first version of a system as quickly as possible, ignoring attributes such as reusability, scalability or flexibility. At the same time or shortly after the development of the system's first version is started, an architecture is designed which satisfies all functional and also non-functional requirements for a long-term system. The second system replaces the first once it is finished.

The initial list of requirements resulting from the competition analysis is the starting point for the creation of a requirements catalog for the entire EC portal to be developed. This requirements catalog is checked for contradictions, redundancy and completeness in several ways; for example, by interviewing users and providers. Users are people or groups of people who will actually use the portal, while providers are persons or groups of persons who will run the portal in order to provide its services to the users (in the context of this paper, users are insurance agents and the provider is the insurance company). Both users and providers have different, potentially competing requirements.

During the interviews for the IPSI portal, it became clear that some insurance companies already used supporting systems for their agents. These systems were examined in order to identify further requirements. After consolidating all requirements from the different sources, the requirements catalog was corrected and extended as required, and requirements re-checked for errors.

3.2 Subsystem Selection

In most cases, EC systems are not developed independently of an existing hardware and software infrastructure. Usually, the EC systems have to be integrated into the existing infrastructure by sharing data with its systems. However, the sharing of data between the EC system and existing software systems may not always be sufficient – sometimes, the use of existing functionality is necessary. Thus, the IPSI electronic commerce portal exchanges data with its subsystems as well as with the database systems of the insurance company (e.g. UDS for BS2000). In this way, the portal can supply the insurance agent with data of people insured and their contracts. Furthermore, the portal needs the functionality of a complex tariff computation module, e.g. for a life insurance. Existing software systems, such as the latter are termed legacy systems. In order to realize each subsystem, it must be decided if existing software systems fulfil the client's requirements, and if an existing software system can be integrated or if it is necessary to develop new software.

For the IPSI electronic commerce portal, it was decided to integrate existing software systems for most subsystems. This decision was followed by market analysis to determine which existing systems should be used. The analysis also took into account non-functional

criteria such as price, availability, support, platform, and possibilities of integrating the system (discussed in the next section), and led to the selection of Microsoft Outlook 2000, Pirobase 4.0, SmartStore 2.0 and several freeware communication applications for the subsystems office, content management, procurement and communications respectively (see Figure 3).

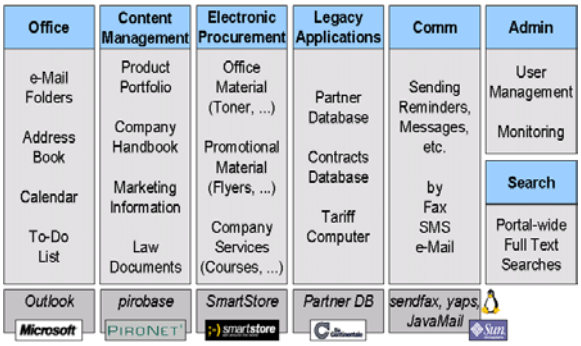

Figure 3: Subsystems of the electronic commerce portal

3.3 Prototype development

In addition, it had to be determined if the software systems selected provided a programming interface (API), or if an interface could be developed. This was achieved by developing prototypes on the basis of key features (major requirements in form of use cases), with the goal to identify opportunities for integrating the software systems with each other (Figure 4). For each software system, key features were defined, that had to be realized by the prototype. The prototypes should show if the features of the underlying software system could be accessed through its interface.

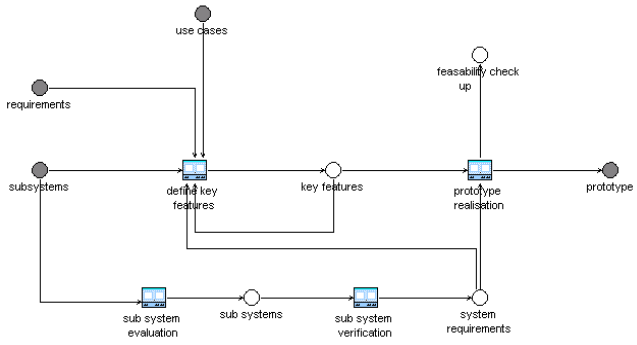

Figure 4: Prototype development subprocess model

Based on the prototypes, the effort, cost and time for the development of the whole EC system could be estimated. This estimate was used to verify the "time-to-market" aimed for by marketing, and to plan accompanying measures such as advertising etc. In the case of the IPSI electronic commerce portal, more resources were necessary for the development of an interface to integrate MS Outlook 2000 than for the development of the communications subsystem based on Java libraries. The effort required to integrate the partner database legacy system was relatively low since the adaptor could be implemented using XML (Haifi 2000). However, this is not always the case. Depending on the type of legacy system, integration may be more difficult. For example, under some conditions the integration of an SAP R/2 system with an EC system can only be achieved through the generation of batch input folders and could therefore require more attention in terms of resource capacity devoted to that integration task.

3.4 GUI Development

The graphical user interface for an EC system is developed in two steps. First, a user interface prototype is designed. This prototype is also used by marketing/sales to support accompanying advertising measures. The prototype development begins with writing a storybook based on use cases. This storybook is then used to define a style guide and, in a second step, to realize and implement the user interface for the EC system. For the IPSI electronic commerce portal, this was done for multiple access channels (WWW, WAP).

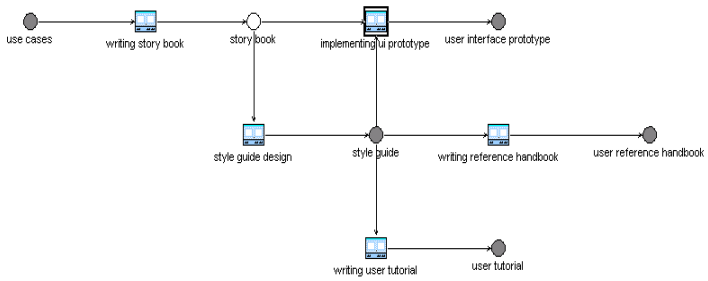

Figure 5: User interface design subprocess model

In addition to the portal's specific functionality in the insurance B2E application domain, its content is a significant element. Content comprises all the information the EC portal provides, as well as its presentation within the user interface. Content often has multi-media characteristics, i.e. it comprises textual information, moving and still pictures and audio information. Consequently, a content manager responsible for multi-media information plays an important role in the software development process. This is a new role that can consist of several other roles, such as the media author who collects textual information and reworks it for a consistent presentation; the media designer responsible for the audio-visual design of the user interface; and the media producer who researches available media, creates images, graphics, animations, audio and video sequences, and clarifies copyright issues. Media editors are responsible for quality assurance in the multi-media content part of the application.

In addition to the role of a content manager, with its many tasks and responsibilities, the role of an ergonomics advisor has to be taken on by a team member. The ergonomics advisor's task is to ensure that the user interface of the EC portal meets ergonomic criteria, i.e.

- it is suited to the tasks the user has to accomplish
- it guides the user by being self-explanatory and gives additional help on request

- it lets the user decide how to use the system without forcing him or her to follow a predefined set of procedures
- it signals and describes user errors and allows their correction with little effort
- it can be adapted to the user's level of experience

User manuals can be differentiated into tutorials and references. For the creation of the user manuals, a style guide is used that describes what the complete user documentation should look like. The storybook already used for the user interface prototype was also used to create the tutorial. (Figure 5).

3.5 Integration and System Test

In the implementation phase, the system architecture built during the design phase was implemented in the Java programming language. In this phase, elementary parts of the system architecture (the controllers, adaptors, formatters and business objects mentioned in section 2) were incrementally implemented, class tests were performed and classes were combined to form subsystems (or components). All implemented subsystems subsequently went through a component test. Based on the use cases, test data sets were created to test the subsystems functionality.

In the integration phase, the tested components were then integrated into the EC portal. The complete integrated system was then subject to system and integration tests. To do this, the test data sets used for the component test were extended, and new sets were created. After a successful system test, the EC portal was delivered to the customer, together with the user tutorial in the system delivery phase (Figure 6).

4 Conclusion

Several conclusions can be drawn from the IPSI development process and are described in this section:

The development process for the IPSI electronic commerce portal is characterized by the high effort necessary to integrate the subsystems. This experience can be transferred to the development of other EC systems, because an EC system usually has to be integrated into a pre-existing software and hardware infrastructure. The integration effort comprises not only the design and implementation of interfaces (APIs), but also testing of those interfaces. The more complex the subsystems are, the more effort is required for the interface test since the necessary test drivers and stubs have to be equally complex.

Every introduction of an EC system to the market should happen within an adequate "time-to-market". Consequently, an early estimate of the feasibility, required effort and duration of the development project has to be made. This is a particularly difficult task in the EC context, because many new technologies (such as new Java libraries and XML) are used and not every developer is skilled in these technologies. What makes estimates even more complex is the fact that in some cases the effort needed to implement a specific component depends on implementation details (like the side effects of using RMI). These details can only be clarified by developing (vertical) prototypes. Only after implementing these prototypes we were able to assess the feasibility of the architecture and to calculate the effort and duration needed for the implementation tasks.

Productive use of IPSI showed that architecture openness 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 Enterprise Java Beans or DCOM.

As in every software system, features supporting the user (e.g. a self-explanatory user interface and online help) also should not be neglected in EC systems. It is important that the user-support features are tailored to the intended EC system target group. For example, in e-government, with its very heterogeneous target group, user-supporting features are mandatory. The same is true for EC systems used in an intra- or extranet, such as the EC portal for insurance agents. Consequently, the way the user interface of an EC system is designed significantly contributes to user acceptance of the system. This means that the software development process must include the creation of a user interface prototype that can form the basis for discussions with ergonomics specialists and also serve as a marketing tool.

All the activities mentioned above have been included in the IPSI development process. Nevertheless, there are some more aspects to be kept in mind when developing EC systems, not included adequately in the IPSI development process to date. For example, consideration of performance issues is extremely important, especially when using highly layered object-oriented architectures for Web applications. Thus, performance modeling and testing (Menascé, and Almeida 2000) should be a central activity in any software development process for EC systems. In general, quality-assuring activities of any kind are often victims of the "time-to-market" philosophy. Here, the goal must be to construct software development processes that ensure a consistent high quality of EC systems, despite the changed and dynamic conditions, and take into account the shorter development time for these systems.

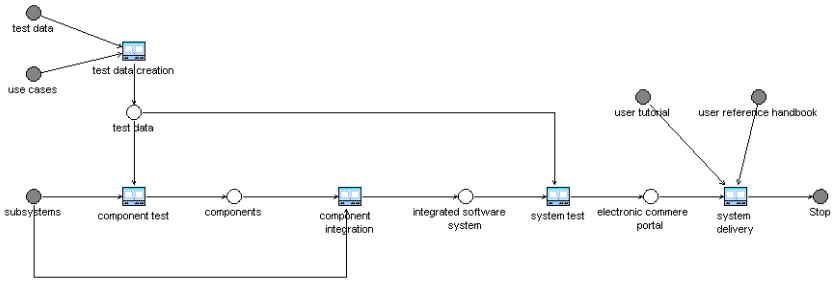

Figure 6: Integration and system test subprocess model

5 References

BAKER, S. and GERAGHTY, R. (1998): Java for Business Objects. In *Developing Business Objects*. 225-237. CARMICHEL, A. (ed). SIGS Cambridge University Press.

BAYER, F., JUNGINGER, S. and KÜHN, H. (2000): A Business Process-Oriented Methodology for Developing E-Business Applications. In *Proc. 7th European Concurrent Engineering Conference*. 123-132. BAAKE, U., ZOBEL, R. and AL-AKAIDI, M. (eds). SCS Publishing House.

BOOK, M., GRUHN, V. and SCHÖPE, L. (2000): Realizing An Integrated Electronic Commerce Portal System. In *Proc. of the Americas Conf. on Information Systems*. 156-162. CHUNG, M. (ed). Association for Information Systems.

CHESHER, M. AND KAURA, R. (1998): *Electronic Commerce and Business Communications*. Springer, Berlin, Heidelberg, New York.

COYLE, F. (2000): Legacy Integration – Changing Perspectives. *IEEE Software* **17**(2): 37-41.

DEITERS, W. and GRUHN, V. (1994): The Funsoft Net Approach to Software Process Management. *Int. Journal of Software Engineering and Knowledge Engineering* **4**(2): 229-256. World Scientific Publ. Company.

GRUHN, V. and SCHÖPE, L. (2001): A Software Process for an Integrated Electronic Commerce Portal System. In *Proc. 8th European Workshop on Software Process Technology*. 90-101. AMBRIOLA, V. (ed). Springer, Berlin.

HAIRE, B., HENDERSON-SELLERS, B. and LOWE, D. (2001): Supporting Web Development in the OPEN process: Additional Tasks. In *Proc. 12th COMPSAC 2001*. 383-389. IEEE Computer Society.

HAIFI, L. (2000): XML and Industrial Standards for Electronic Commerce. *Knowledge and Information Systems* **2**(4): 487-497. Springer, London.

HASSELBRING, W., KOSCHEL, A. and MESTER, A. (2001): Basistechnologien für die Entwicklung von Internet-Portalen. In *Datenbanksysteme für Büro, Technik und Wissenschaft*. 517-526. HEUER, A., LEYMANN, F. and PRIEBE, D. (eds). Springer, Berlin, Heidelberg, New York.

HARRISON, W., OSSHER, H. and TARR, P. (2000): Software Engineering Tools and Environments. In *Proc. 22nd Int. Conf. on Software Engineering*. 263-277. FINKELSTEIN, A. (ed). ACM Press.

HOFFNER, Y., LUDWIG, H., GREFEN, P. and ABERER, K. (2001): CrossFlow: Integration Workflow Management and Electronic Commerce. *SIGecom Echanges, Newsletter of the ACM SIG on Electronic Commerce* **2**(1): 1-10. ACM Press.

LAMOND, K. and EDELHEIT, J. (1999): Electronic Commerce Back-Office Integration. *BT Technology Journal* **17**(3): 87-96. Kluwer Academic Press.

LAARTZ, J., SCHERDIN, A., CAFARELLI, D. and HJARTAR, K. (2000): Evolve your architecture. *CIO Magazine*, Issue September 15, 2000.

LEWANDOWSKI, S. (1998): Frameworks for Computer-Based Client/Server Computing. *ACM Computing Surveys*, **30**(1): 3-27. ACM Press.

LINCKE, D. and ZIMMERMANN, H. (1999): Integrierte Standardanwendungen für Electronic Commerce – Anforderungen und Evaluationskriterien. In *Managementhandbuch Electronic Commerce*. 197-210. HERMANNS, A. and SAUTER, M. (eds). Verlag Franz Vahlen, Munich.

MENASCÉ, D.A. and ALMEIDA, V.A.F. (2000): *Scaling for e-Business – Technologies, Models, Performance, and Capacity Planning*. Prentice Hall.

NIELSEN, J. (2000): *Designing Web Usability: The Practice of Simplicity*. Riders Publishing, Indianapolis.

NOFFSINGER, W.B., NIEDBALKSI, R., BLANKS, M. and EMMART, N. (1998): Legacy Object Modeling speeds Software Integration. *Communications of the ACM* **41**(12): 80-89. ACM Press.

SHAW, M.J. (2000): Electronic Commerce: State of the Art. In *Handbook on Electronic Commerce*, 3-24. SHAW, M., BLANNING, R., STADER, T. and WHINSTON, A. (eds). Springer, Berlin, Heidelberg, New York.