ELSEVIER

Information and Software Technology 44 (2002) 891–901

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

Software processes for the development of electronic commerce systems☆

Volker Gruhn[a,*], Lothar Schöpe[b]

[a]Dortmund University, Baroper Str. 301, D-44227 Dortmund, Germany
[b]Informatik Centrum Dortmund e.V., Joseph-v.-Fraunhofer-Str. 20, D-44227 Dortmund, Germany

Abstract

The development of electronic commerce (EC) systems is subject to different conditions than that of conventional software systems. This includes the introduction of new activities to the development process and the removal of others. 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. This article describes the development of an EC system and it generalizes salient features of the software process used. The result is a process model which can be used for other highly integrative EC system development projects. The processes described are determined by short process lifecycles, by an orientation towards integration of legacy systems and by a strict role-based cooperation approach. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Electronic commerce system; Software process model; Software process

1. Introduction

In this paper, electronic commerce (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) [20–22]. The parties involved in EC transactions use information technology (IT) systems to automate their transactions [2,5]. The examples used in this paper are business-to-employee (B2E) EC systems. The EC portal system discussed in this article 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 [3,7,9,11]. 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.
- 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.

☆ This article is an extended version of the article presented at APAQS 2001 [31].

* Corresponding author.
E-mail addresses: volker.gruhn@uni-dortmund.de (V. Gruhn), lothar.schoepe@icd.de (L. Schöpe).

PII: S0950-5849(02)00109-X

These aspects are discussed in the following. 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. This challenge clearly indicates that paradigms known from component-based systems and architectures [23] are well-suited for EC systems as well. Components in the sense of component models like Enterprise JavaBeans or DCOM are the building blocks from which EC systems are built [25]. 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 [18]. 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. [13].

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 s [17] for a response, resulting in loss of revenue and image for the company associated with the site. Therefore, characterization of customer behavior, workload forecasting and performance modelling become very prominent activities [24]. Two characteristics of EC customer behavior aggravate workload characterization in EC settings: peak-like request bursts and high-volume data requests that are not typically found in conventional software systems [16].

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 JavaBeans, 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,

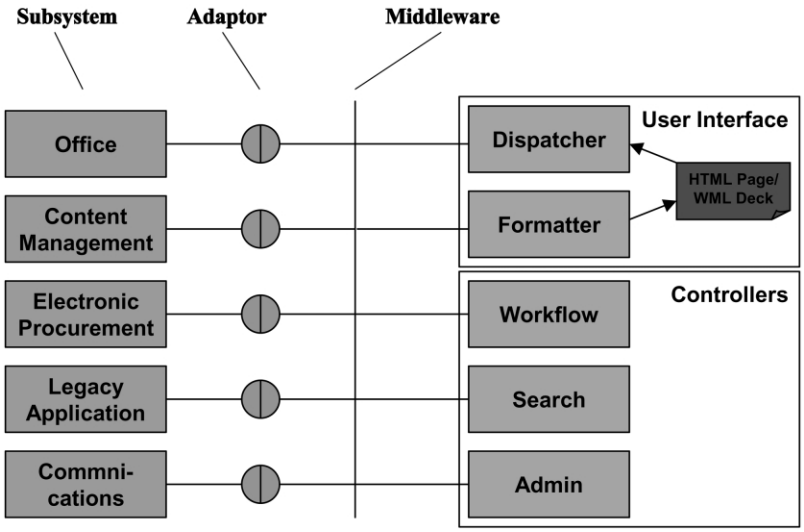

Fig. 1. System architecture.

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 validates the proposed process model and relates it to other work in the software process field. Finally, Section 5 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 [4]. 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 B2E processes [15]. 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. The process used for the development of IPSI is discussed in Section 3.

During the requirements analysis phase of the project, it was recognized that the EC portal serves as an integration platform for different heterogeneous subsystems [8]. Based on an *n*-tier-architecture, the user interface and data repository[1] are separated from the functional business logic [12] that resides in multiple application components (called subsystems). This highly integrative character of IPSI had a substantial impact onto the software process chosen (compare Section 3). 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

[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.

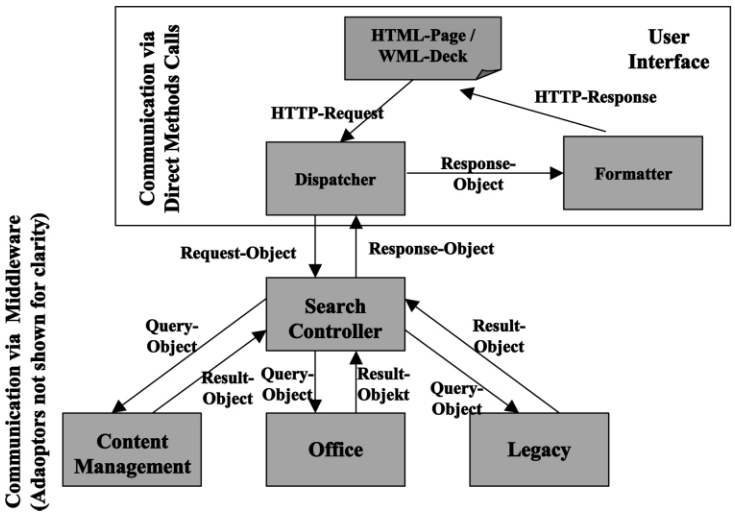

Fig. 2. Communication within the electronic commerce portal.

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 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 such as a partner management system storing contract data of people insured in the case of IPSI.

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

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 et al. [4] 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).

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 [17]. 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.[2]

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

Every user action like clicking on a link or submitting a form generates an HTTP request 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 actual 'work' of the system is done by the

[2] 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).

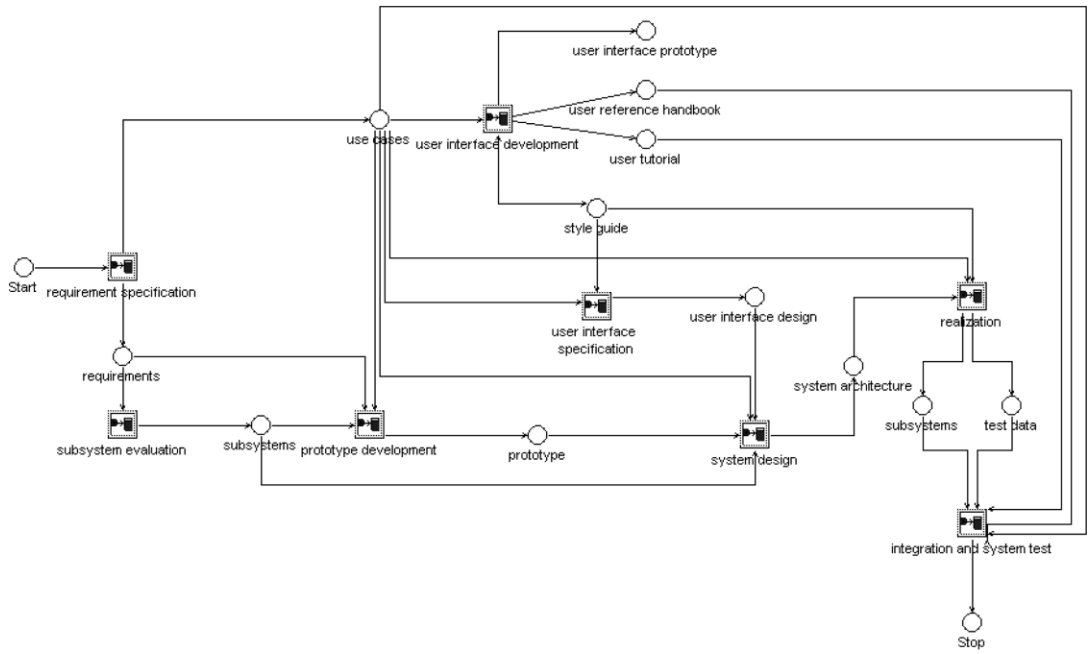

Fig. 3. Electronic commerce portal development process model.

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 [10], especially by interacting with the subsystems' adaptor interfaces. Controllers and subsystem adaptors communicate by exchanging business objects [1] 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

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

In this section we describe the software process chosen for the development of IPSI. This process is considered as a

good candidate for other highly integrative EC systems as well (compare Section 4).

In general, 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.

One trend in the software process community is to promote processes which focus on components and on building systems from components. Examples are Catalysis [26] and SELECT [27]. Both describe in detail how components could be specified and how the integration of components into systems could look like. Some of the elements used in the process for developing IPSI are related to these process models (compare Section 4).

The process model for the development of the IPSI EC portal is presented schematically in Fig. 3, using the FUNSOFT net notation [6]. 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 evaluation, 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 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 subprocess models such as the user interface specification and development.

In the following, the development process of the IPSI EC 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 EC-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.

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 et al. [14] 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

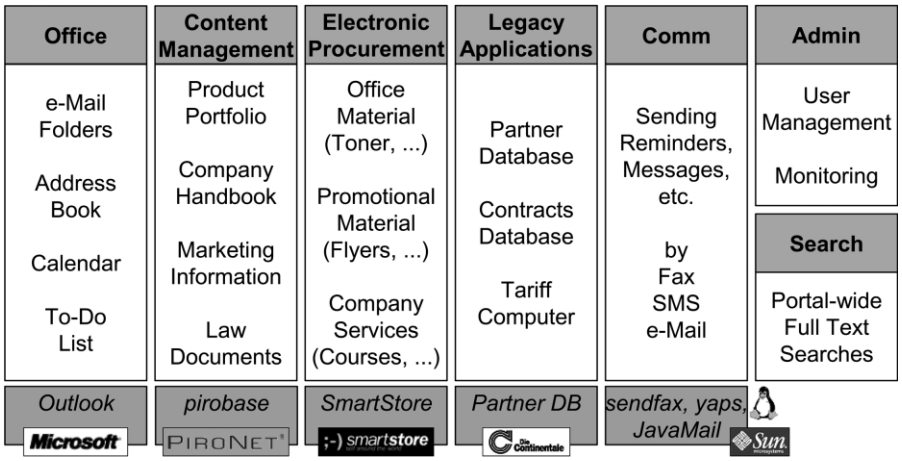

Fig. 4. Subsystems of the electronic commerce portal.

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 evaluation

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 EC 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.

For the IPSI EC 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 Section 3.3), 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 (Fig. 4).

3.3. Prototype development

In addition, it had to be determined if the software systems selected provided a programming interface (API),

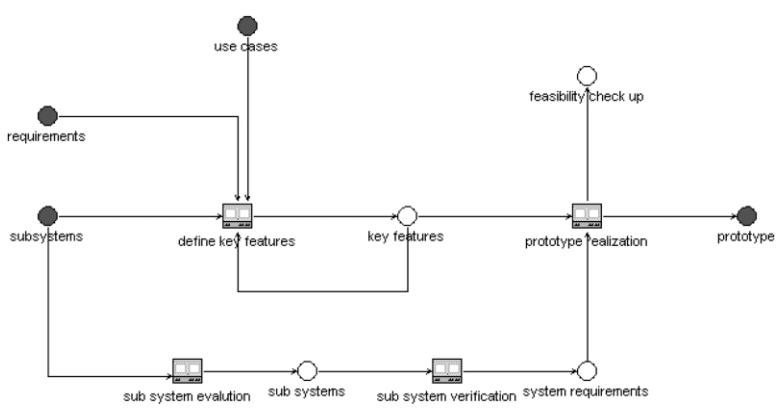

Fig. 5. Prototype development subprocess model.

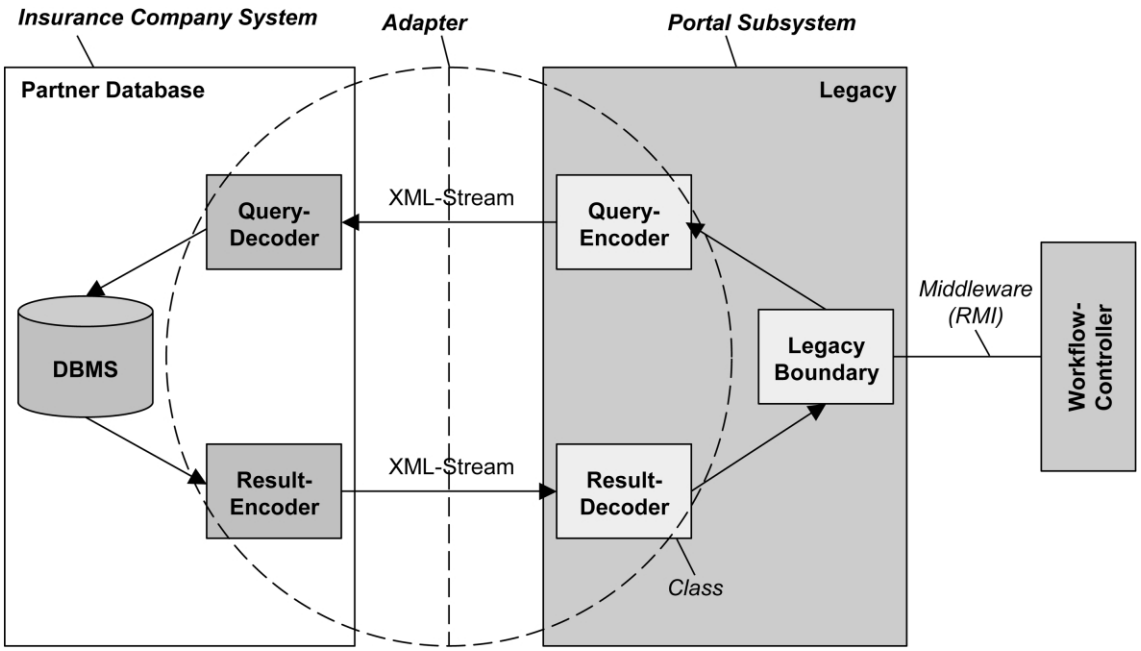

Fig. 6. Integration of legacy system.

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 (Fig. 5). 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.

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 EC 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. 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 EC portal, this was done for multiple access channels (WWW, WAP).

In addition to the portal's specific functionality in the insurance B2E application domain, its content is a significant

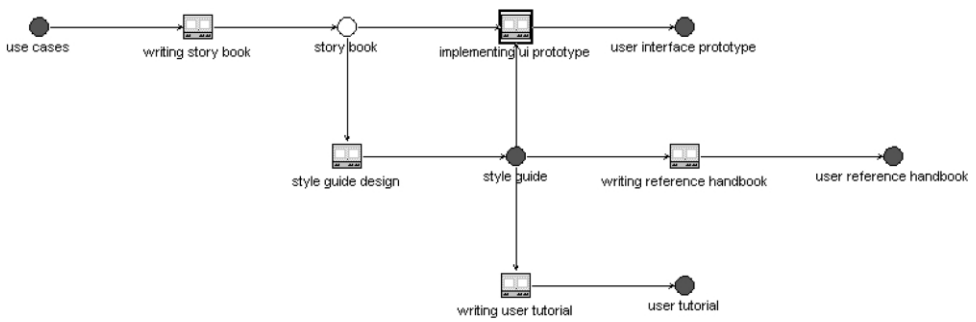

Fig. 7. User interface design subprocess model.

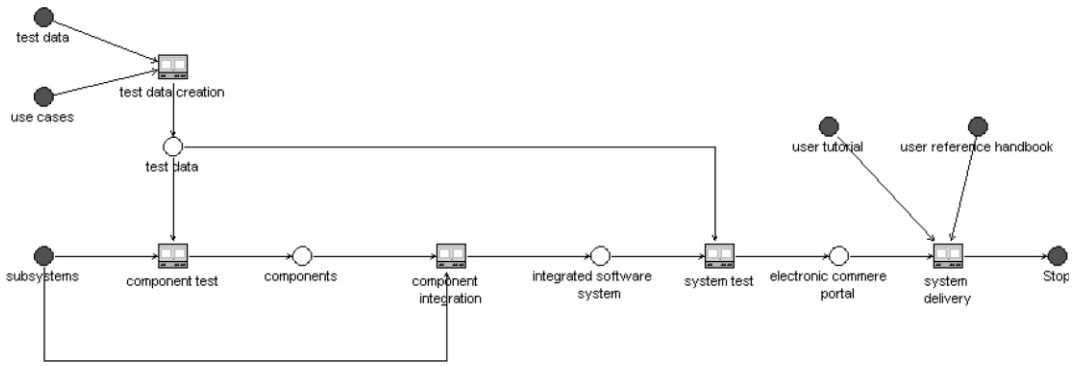

Fig. 8. Integration and system test subprocess model.

element. Content comprises all the information the EC portal provides, as well as its presentation within the user interface. Content often has multimedia characteristics, i.e. it comprises textual information, moving and still pictures and audio information. Consequently, a content manager responsible for multimedia 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 multimedia 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 (Fig. 7).

3.5. Integration and system test

In the implementation phase, the system architecture built during the design phase was implemented in Java. 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).

As an example, let's consider a search request handled by the legacy system (Fig. 6):

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 [19] 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.

All implemented subsystems subsequently went through a component test. Based on the use cases, test data sets were created to test the subsystems functionality. This test is a combined black box and white box test as described in Ref. [28]. While white box techniques are applied to all components for which the source code is available, black box techniques have to be applied to all components purchased from vendors, since these usually do not provide access to their source codes.

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 (Fig. 8).

4. Evaluation of the process applied

The process chosen for the development of IPSI provides some features which are closely related to features of EC systems. It is influenced by the software process work as described in Refs. [29,30]. The most important features of EC systems reflected by the chosen software process are:

1. The development process for the IPSI EC 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.
2. It is rather difficult to assess the feasibility of developing an EC system, because new, unproven technologies have usually to be used. 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.
3. Usability engineering is a crucial tasks. In EC systems users are a heterogeneous and usually not personally known set of people. To check whether are not navigation details and site structure suit them well is difficult and, in general, demands for usability engineering methods.
4. Testing urgently demands the integration of different types of testing techniques. This relates to the integration of black box and white box techniques as well as the combined use of component and system tests.

As mentioned in Refs. [7,10,30] these challenges are typical for EC system. Even though some of these challenges are properly addressed by traditional software process models, the process model applied to the development of IPSI concentrates on these challenges and therefore provides a lean solution to the problem of developing EC system.

5. Conclusion

Software processes for EC system are different from traditional software processes (as, for example, used in the development of information systems). Even though not a single of the identified challenges for the development of EC systems is completely new, process models which focus on EC systems (and which therefore are a natural choice for the development of such systems) are not available. Our approach was to start with a real problem (the development of IPSI), to model the process as it was carried out and to generalize the process in the process model discussed in Section 3.

The result is a rather lean software process model which covers most aspects of EC systems and which is flexible enough to be easily extended, if needed. All the activities mentioned in Section 3 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 [16,24] 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.

Our future work will be devoted to applying the proposed software process model to other types of EC systems (e.g. to a business-to-consumer system) and to integrated subprocesses which focus on activities like performance modeling and after-release monitoring which have not been appropriately considered yet.

References

[1] S. Baker, R. Geraghty, Java for business objects, in: A. Carmichel (Ed.), Developing Business Objects, SIGS, Cambridge University Press, Cambridge, 1998, pp. 225–237.
[2] N.R. Adam, Y. Yesha, Electronic commerce: an overview, in: N.R. Adam, Y. Yesha (Eds.), Electronic Commerce, LNCS 1028, Springer, Berlin, 1995, pp. 4–12.
[3] F. Bayer, S. Junginger, H. Kühn, A business process-oriented methodology for developing e-business applications, in: U. Baake, R. Zobel, M. Al-Akaidi (Eds.), Proceedings of the Seventh European Concurrent Engineering Conference, SCS Publishing House, 2000, pp. 123–132.
[4] M. Book, V. Gruhn, L. Schöpe, Realizing an integrated electronic commerce portal system, in: M. Chung (Ed.), Proceedings of the Americas Conference on Information Systems AMCIS 2000, Association for Information Systems, 2000, pp. 156–162.
[5] M. Chesher, R. Kaura, Electronic Commerce and Business Communications, Springer, Berlin, 1998.
[6] W. Deiters, V. Gruhn, The Funsoft net approach to software process management, International of Journal of Software Engineering and Knowledge Engineering 4 (2) (1994) 229–256.
[7] B. Haire, B. Henderson-Sellers, D. Lowe, Supporting Web development in the OPEN process: additional tasks, Proceedings of the 12th COMPSAC (2001) 383–389.

[8] W. Hasselbring, A. Koschel, A. Mester, in: A. Heuer, F. Leymann, D. Priebe (Eds.), Basistechnologien für die Entwicklung von Internet-Portalen, Datenbanksysteme für Büro, Technik und Wissenschaft (BTW), 2001, pp. 517–526.
[9] W. Harrison, H. Ossher, P. Tarr, Software engineering tools and environments, in: A. Finkelstein (Ed.), Proceedings of the 22nd International Conference on Software Engineering, 2000, pp. 263–277.
[10] Y. Hoffner, H. Ludwig, P. Grefen, K. Aberer, Crossflow: integration workflow management and electronic commerce, SIGecom Exchanges 1 (2) (2001) 1–10. ACM Press.
[11] V. Gruhn, L. Schöpe, A software process for an integrated electronic commerce portal, in: V. Ambriola (Ed.), Proceedings of the Eighth EWSPT, Springer, Berlin, 2001, pp. 90–101.
[12] S. Lewandowski, Frameworks for computer-based client/server computing, ACM Computing Surveys 30 (1) (1998) 3–27. ACM Press.
[13] K. Lamond, J. Edelheit, Electronic commerce back-office integration, BT Technology Journal 17 (3) (1999) 87–96. Kluwer Academic Press.
[14] J. Laartz, A. Scherdin, D. Carfarell, K. Hjartar, Evolve your architecture, CIO Magazine 15 (September) (2000).
[15] D. Lincke, H. Zimmermann, Integrierte standardanwendungen für electronic commerce-anforderungen und evaluationskriterien, in: A. Hermanns, M. Sauter (Eds.), Managementhandbuch Electronic Commerce, Verlag Franz Vahlen München, Berlin, 1999, pp. 197–210.
[16] D.A. Menasce, V. Almeida, Scaling for e-Business, Models, Performance, and Capacity Planning, Prentice Hall, Englewood Cliffs, NJ, 2000.
[17] J. Nielsen, Designing Web Usability: The Practice of Simplicity, Riders Publishers, 2000.
[18] W. Noffsinger, R. Niedbalski, M. Blanks, N. Emmart, Legacy object modeling speeds software integration, CACM 41 (12) (1998) 80–89. ACM Press.
[19] L. Haifi, XML and industrial standards for electronic commerce, Knowledge and Information Systems 2 (4) (2000) 487–497. Springer Verlag, London.
[20] M.J. Shaw, Electronic commerce: state of the art, in: M. Shaw, R. Blanning, T. Stader, A. Whinston (Eds.), Handbook on Electronic Commerce, Springer, Berlin, 2000, pp. 3–24.
[21] V. Zwass, Electronic commerce: structures and issues, International Journal of Electronic Commerce 1 (1) (1996) 3–23.
[22] V. Zwass, Structure and macro-level impacts of electronic commerce: from technological infrastructure to electronic marketplaces, in: K.E. Kendall (Ed.), Emerging Information Technology, Sage Publishers, Beverly Hills, CA, 1999, pp. 517–526.
[23] A.W. Brown, K.C. Wallnau, The current state of CBSE, IEEE Software 9/10 (1998).
[24] D. Menasce, V. Almeida, Capacity Planning for Web Performance—Metrics, Models and Methods, Prentice Hall, Englewood Cliffs, NJ, 1998.
[25] C. Szyperski, Component Software—Beyond Object-Oriented Programming, Addison-Wesley, Reading, MA, 1998.
[26] D. D'Souza, A. Wills, Objects, Components, and Frameworks with UML—The Catalysis Approach, Addison-Wesley, Reading, MA, 1998.
[27] P. Allen, S. Frost, Component Based Development for Enterprise Systems, Cambridge University Press, Cambridge, 1998.
[28] S. Beydeda, V. Gruhn, A graphical representation of classes for integrated black- and white-box testing, Proceedings of the International Conference on Software Maintenance 2001 (2001) 706–715.
[29] I. Alloui, S. Cimpan, F. Oquendo, Monitoring software process interactions: a logic-based approach, Proceedings of the Eighth European Software Process Workshop (2001) 39–46. LNCS 2077, Springer.
[30] G. Cignoni, Reporting about the Mod software process, Proceedings of the Eighth European Software Process Workshop (2001) 242–245. LNCS 2077, Springer.
[31] M. Book, V. Gruhn, L. Schöpe, A specific software development process for an electronic commerce portal, in: Y.T. Yu, T.Y. Chen (Eds.), Proceedings of the Second Asia-Pacific Conference on Quality Software, 2001, pp. 406–414.