Association for Information Systems
AIS Electronic Library (AISeL)

Wirtschaftsinformatik Proceedings 2007 | Wirtschaftsinformatik

February 2007

Management of Portal Evolution - Introducing Evolution Management for the Corporate Financial Portal

Hong Tuan Kiet Vo
Universität Karlsruhe (TH), vo@iism.uni-karlsruhe.de

Helmuth Elsner
Universität Karlsruhe (TH), elsner@iism.uni-karlsruhe.de

In: Oberweis, Andreas, u.a. (Hg.) 2007. *eOrganisation: Service-, Prozess-, Market-Engineering*; 8. Internationale Tagung Wirtschaftsinformatik 2007. Karlsruhe: Universitätsverlag Karlsruhe

ISBN: 978-3-86644-094-4 (Band 1)
ISBN: 978-3-86644-095-1 (Band 2)
ISBN: 978-3-86644-093-7 (set)

Management of Portal Evolution

Introducing Evolution Management for the Corporate Financial Portal

Hong Tuan Kiet Vo, Helmuth Elsner

Institut für Informationswirtschaft und -management
Universität Karlsruhe (TH)
76131 Karlsruhe
{vo,elsner}@iism.uni-karlsruhe.de

Abstract

Software evolution is an essential concept underlying the engineering of corporate portals. Due to the complexity of such systems, it requires great effort and is not advisable to design and implement a feature-complete corporate solution. The concept of evolutionary portal development helps portal engineers to cope with design complexity. While the technical perspective on component based software development and evolution is widely discussed, little work addresses the actual management of software evolution, let alone in the portal engineering context. In this paper we focus on the management of portal evolution. On the basis of a portal engineering process model we discuss methods and practices that facilitate the management of portal evolution cycles. We further outline the evolution management procedure and tools that we have established for the corporate financial portal at Bayer and point out key lessons we have learned so far.

1 Introduction

Software evolution is an essential concept underlying the engineering of corporate portals. Due to the complexity of corporate portal systems that may even rival that of corporate ERP systems [Remu06], it is virtually impossible and not advisable to design and implement a feature complete corporate portal solution. The concept of evolutionary portal development helps portal engineers to cope with the design complexity and to foster user acceptance as iterative portal releases will incorporate actual user requirements gathered during portal operation. While the

technical perspective on software development and reuse is widely discussed (cf. component based software development), only a handful of work address the actual management of software evolution cycles, let alone in the portal engineering context.

In this paper we address the management of portal evolution i.e. we outline the processes and activities that are of importance to manage evolution cycles. On the basis of a portal engineering framework we discuss activities that help to specify direction for the next evolution cycles. We discuss the evolution management procedure and tools that we have established at the corporate financial portal project at Bayer and point out key lessons we have learned so far.
The paper is structured as follows. In the following section we discuss approaches for software evolution with focus on web applications. Section 3 outlines evolution management in the context of a portal engineering process model as proposed by the authors. Section 4 then illustrates the application of computer aided evolution management in the context of the corporate financial portal project. The paper closes with a summary and an outlook on future work.

2 Management of Software Evolution

The technical perspective of evolutionary software development is extensively discussed especially in the context of component based software engineering [BrWa96]. While these discussions focus on the question of how to facilitate reuse of software components, we are more interested in the actual management process that follows software evolution cycles. Yet, this question is often addressed only on a side note while the focus is on the iterative or evolutionary development process. Sommerville points out the importance of feedback or change proposals as the basis of software evolution and describes a generic change identification and evolution process cycle (cf. figure 1). Again, the focus is on the development process and Sommerville further states that the actual software evolution process will vary considerably depending on the type of software being maintained [Somm04].

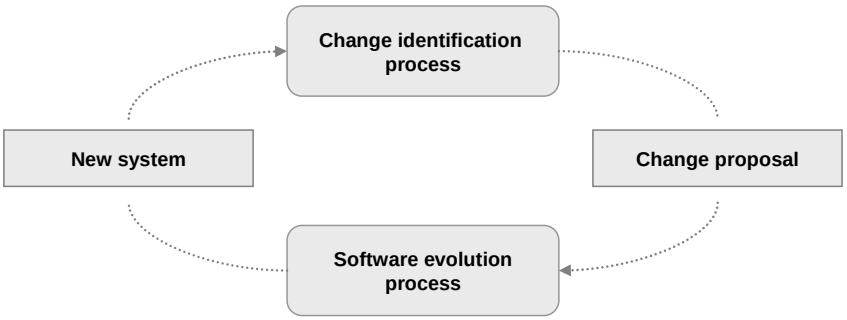

Figure 1: Change identification and evolution processes [Somm04]

With the current trend towards web application, software evolution is again becoming more important. One distinct feature of web applications is that it is no longer necessary to install the applications on the local machines, but all users access the current version of the software through their internet browser. Therefore aspects of software distribution and versioning are practically less important when developing for the web. That in return fosters iterative software evolution and development cycles. Software engineering for the web is discussed in the field of web engineering that emerged in the late 90s following the internet revolution and the increase of applications developed for this medium. In order to enable the development and maintenance of high quality Web-based system and applications Murugesan [MDHG99] proclaimed the need for web engineering that is *the establishment and use of sound scientific, engineering and management principles, and disciplined and systematic approaches to development, deployment and maintenance of Web-based systems* [MDHG99, p.6]. Web engineering distinguishes itself from traditional software engineering as aspects like software evolution, hardly predictable network behaviour, heterogeneous and usually unknown end-users with different cultural background are of greater importance when developing applications for the web [Powe98; Schw00]. With regard to software evolution, again current works primarily focus on the technical aspects of enabling the reuse thus evolution of web components [Gaed98]. Currently, dedicated discussions on the management of web application evolution are rare and often mentioned as a side note. That is surprising given the fact that the lifecycle of a web application not only comprises of develop, launch and maintain [Powe98] but also operation and iterative evolution. That strengthens our believe that for the engineering of web applications and especially web portals that are in a state of constant change, a structured management process that prepares and guides evolution cycles is necessary for a sustainable evolution.

3 Management of Portal Evolution

3.1 Management Evolution and Portal Engineering

As stated in section 2, research that particularly focuses on the management of evolution is rare both with regard to work on software engineering and web engineering. Following our literature review, this holds also true for current discussions on the engineering approaches for (corporate) portals.

Finkelstein and Aiken [FiAi00] present a framework for engineering enterprise portals. They especially focus on the design and implementation stage and illustrate how to implement a corporate portal using XML. The discussion centres around methods for data transformation and data modelling that facilitate the integration of structured and unstructured data. Portal evolution is discussed from a technical perspective only with regard on practices and architectures that enable future integration of data sources.

Alt et al. [ACLÖ04] outline a method for developing process portals. According to their definition, process portals are enterprise portals that especially focus on the integration of (external) customer processes and facilitate the collaboration between enterprises, customers and suppliers. Following the method engineering methodology, their method for developing process portals focuses on the portal strategy, portal design and portal technology stage. The notion of evolution is not within the scope of this work.

The Frauenhofer IAO portal analysis and design method – PADEM – follows the stages of a traditional software lifecycle: the definition of the strategy, subsequent requirement analysis, conceptual design, realisation and the introduction of the portal [GHKV04]. For each stage, the method defines checklists, questionnaire and offers practices that shall support companies in their portal selection and decision process and guide them through the portal implementation. PADEM focuses on providing guidance especially during the initial lifecycle of a portal implementation while taking a consulting and management level perspective on the heterogeneous decision problems. Evolution is not explicitly discussed although the authors state that there will be subsequent evolutionary portal releases.

Amberger et al. present a portal engineering process [AmRB04] that comprises of five stages: requirement analysis, analysis of profitability, detailed analysis of users, business processes and corporate IT, proof of concept prototyping and iterative evolution. However, the purpose of this

work is to give an overview of the field of portal engineering therefore the engineering stages – including evolution – lack a detailed further discussion.

In summary, existing frameworks on portal engineering agree on the importance and existence of iterative portal evolution. However, they lack an explicit discussion on how to prepare and guide such an evolutionary cycle. This is not surprising as portal engineering is quite a new field of research. Thus, in the next sections we specifically focus on the discussion of evolution management within the limits of a portal engineering process model that we outline in the following.

3.2 Evolution Cycles in Portal Engineering

Following traditional software engineering process models and lifecycles, figure 2 depicts a portal engineering process model. Dotted lines represent stage transitions while solid lines illustrate feedback relations among engineering stages.

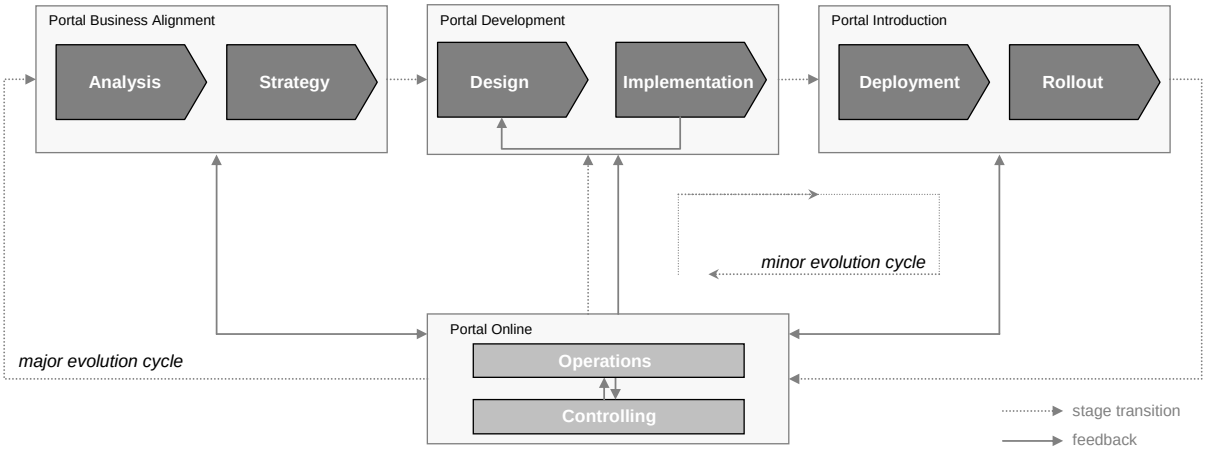

Figure 2: Portal engineering process model and evolution cycles

The portal engineering process model comprises of the *analysis* and *strategy* stages (portal business alignment), the *design* and *implementation* stages (development), the *deployment* and *rollout* stages (introduction) and finally the *operations* and *controlling* stages (online).

Following this process model, the notion of portal evolution is considered on two levels of magnitudes. On the one hand there are the major evolution cycles that require a thorough run through the engineering process from the refinement of the portal strategy to portal service rollout. Major evolution cycles are initiated according to the periodic project steering phases.

On the other hand there are minor evolution cycles nested within a major evolution cycles. A minor evolution is initiated whenever portal services are enhanced or improved in details only.

3.3 Evolution Management for Portal Engineering

According to the change identification and evolution process (cf. section 2, figure 1) every evolution process is triggered by feedback that comprises change proposals, bug reports or feature requests. For evolution management feedback has to be aggregated, categorized and evaluated to derive insights for the preparation of the next evolution cycle. While the majority of the feedback will be obtained during portal operation (i.e. user reported bugs or feature requests), feedback is also generated during portal development and of course following the rollout of a new portal release for example during training workshops. Note, that feedback should be regarded as desirable rather than as something that is unwanted. We believe that feedback is an indicator for actual system usage. In particular, feature requests can be regarded as indicators for the user's experience of the portal system as a *living* system that evolves over time according to the users need. Hence, to motivate the generation of user feedback, portal evolution management has to establish a process that first facilitates feedback reporting and second enables users to easily follow the status of their current requests.

Feedback that is accumulated over time has to be categorized and evaluated by the portal engineers. In general we distinguish between feature request for new applications, bug reports and change request. While we consider bug reports immediately in the next minor evolution cycle, feature and change requests must be evaluated in more detail. With regard to change requests, the developer of the application has to evaluate the redesign and implementation effort to decide on the priority of a change request and the initiator of this request should be notified on the decision. Small feature request are extensions to applications that depending on the request pipeline and the design and implementation effort can be addressed within minor evolution cycles. Major feature request often propose the development of completely new applications. The decision on the development and prioritisation is therefore a portal management level decision and handled respectively. Again the initiator of the request should be notified on the decisions. For the purpose of clear reference, in the following we use the term *issue* for any type of relevant feedback.

Figure 3 outlines the major stages of the portal evaluation management cycle with focus on the perspective of end-users as the primary source of relevant issues. Every evolution cycle starts

after the current batch of issues has been categorized and evaluated. Transparency as one key aspect to foster user participation in portal development requires the portal management to inform the portal users on the evaluation results of their reported issues. Relevant issues are processed in the subsequent development stage. During this stage, it is important to keep the users informed on the current processing status of their request. Once the requests have been implemented, the issue holder is involved in a limited introduction phase to evaluate the new developments. Upon approval, the issue is closed and the changes are introduced in a public release.

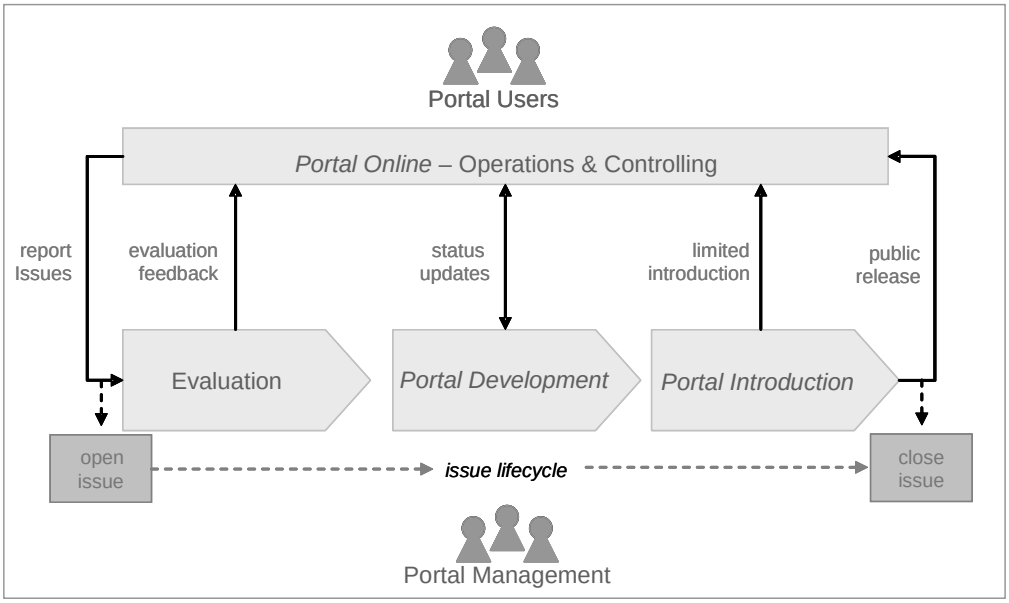

Figure 3: Portal evolution management cycle (issue life cycle)

In summary evolution management as proposed, centres around the management of an issue lifecycle starting with feedback collection, followed by issue categorization and evaluation to the final implementation of the requests. This process has to be supported by an issue management system that is tightly integrated with the issue processing stages. In the following sections, we present and discuss the implementation of the proposed evolution management process and the respective issue management system for Bayer's corporate financial portal.

4 Evolution and the Corporate Financial Portal

4.1 Overview: The Corporate Financial Portal Project

In this section we present the implementation of an issue management system in the Corporate Financial Portal (CoFiPot). The CoFiPot system is operated by the financial department of the Bayer AG. It was implemented with a few core financial services in 2001. Since then, the number of services provided by the system and the number of users has constantly grown. Today the portal is used by users from all over the world and is part of their daily work schedule. The goal for the following years is to extend the portal scope on further financial departments. For example the latest project focuses on providing services for the tax department partially based on their existing IT-solutions with the aim to greatly expand the range of services available to satisfy the information needs of the tax-accountants.

The portal system is based on Microsoft's .NET Framework and loosely on the IBuySpy[1] open source portal framework that at that time was introduced to demonstrate the development of personalized, component-based web applications using the just introduced ASP.NET technology. Portal development is coordinated by the financial department in cooperation with the IT-department. Right from the start the CoFiPot-project was accompanied by a research project offering the authors the possibility to follow the major stages of the corporate financial portal lifecycle.

4.2 The need for Evolution Management

In the initial development stage of the portal, the structure of the project and the number of users and developers were of manageable size. At maximum three developers were working simultaneously on the project and only a small number of services was offered to the users. The assignment of incoming development requests worked straightforward and did not require much administrative effort. The development itself took place on an ad-hoc basis. The communication between users and developers usually took place directly via phone or email. Due to the small development team there was no pressing necessity for a dedicated management of requests. Requests were simply stored and managed in an Excel file accessible for all team-members. New features and changes to the portal system were brought to attention of the development

[1] Current open source portal frameworks like DotNetNuke are derived from the IBuySpy framework, which currently is discontinued.

team via email or in meetings. However, a real documentation about the change proposals and their current status did not exist. Notifications about new features to the users were issued infrequently via mail and often only directly to the target user group.

When the portal system started to further extend its services the number of users, feature requests, change proposals and bug reports consequently began to rise. Due to the growth of the project the development team size increased as well. The previous Excel-based issue management system reached the limit of its scalability. The development-process required a more sophisticated approach that would allow handling a large-scale web application project with a growing number of users and developers. Last but not least with the new portal evolution management process we wanted to offer the portal users a new level of transparency with not only the possibility to report new change proposals but also to monitor the current status of their proposals. The key points for the process development thus were:

- a clearly defined process for the collection and management of requests
- a traceable log of all development steps
- transparency to the users

An issue-management system to support the process was designed and implemented in the portal. This system and the underlying process are presented in the following.

4.3 Corporate Financial Portal Issue Management

The main goal of the issue management system is to facilitate the process to collect and process incoming requests. On the one hand this requirement should enable the developers to keep an overview of all open issues and to minimize the effort to administrate them. Moreover, in order to facilitate planning for the whole project, the system should enable the team to categorize issues by their priority and impact. On the other hand the users of the portal should have the option to easily report issues. With the previous system at least they had to write an email or to call the developers to describe their issue. With the new system they are able to report an issue directly out of the portal system with little effort.

An important topic while designing the fundamentals of the issue management process was the question how to manage the status of a request throughout the development stages as shown in figure 3.

One of the initial questions when designing the issue management process for the Bayer system was how to handle multiple issues that refer to the same topic. In case of a bug in a component many users might report it as an issue. Therefore there is the requirement to link issues to each other so that the processing of multiple issues is combined in one process. The chosen solution contained the step to differentiate between incoming requests as issues and tickets as actual working orders. A ticket thereby combines any number of issues and has a clear key for identification. Issues only identify the original request and provide initial information about the task but they don't change any more, once they have been assigned to a ticket. While any new issue might deal with already recorded problems, each ticket handles a unique topic. If required the assignment of an issue to a ticket can change to another ticket.

The application of the core sequence as presented in figure 3 to the process for managing the CoFiPot system is shown in figure 4.

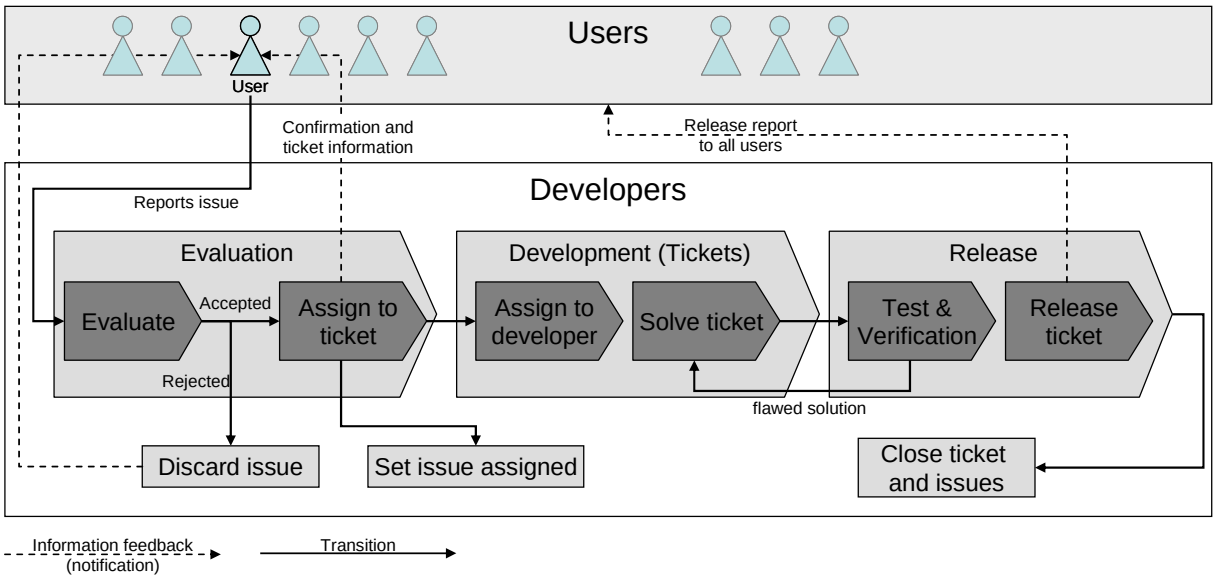

Figure 4: Issue life cycle with ticket handling

At the initial stage an issue is reported and stored in the database. The developers evaluate the request and decide whether to create a new ticket based on this issue, to assign it to an already existing ticket or to reject the topic. When the issue is accepted the resulting ticket will be categorized by priority and the required effort to solve it. In the development stage the ticket is assigned to a developer who will solve the case. When the developer sets the status of the ticket to *solved* the solution is to be tested and verified. Upon a successful evaluation the solution is released to the public and the ticket's status is set to *resolved* thus taking the ticket and all

related issues off the development list. Throughout all these stages the user who reported the initial issue will be informed upon major progresses.
In section 4.3.2. we will show the life-cycle of an issue as it is handled by the CoFiPot-portal in more detail.

Tickets are the actual foundation for the work of the development team. New tickets have the flag "open" and can be assigned to a responsible person. Throughout the working process, the information contained in a ticket can grow through the addition of comments and attachments. Once the developer marks it as resolved, preferably another member of the team verifies the solution and either reassigns the ticket for further work or finalizes the process by setting the ticket's status to *verified*. Every new release automatically will contain all verified tickets.

4.3.1 End user integration

A further goal of the issue management system was to further integrate the users in the development process. First of all they should easily have the option to report bugs or propose changes to the system. Further more they should be able to follow the ongoing development process and learn about the progress of any issues that they have reported or that caught their attention.

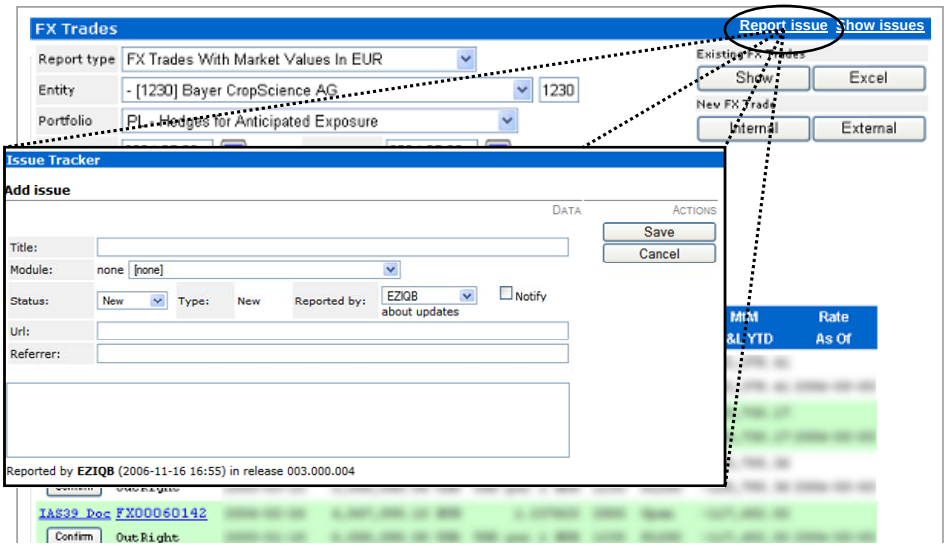

Figure 5: Report issue and module issues functionality on the CoFiPot live pages

The issue management system allows any user to directly report an issue through the portal system. Each module offers the option to directly report an issue or to view all current issues that apply to the module. Figure 5 shows a module with the described functionality and the form that is displayed to the user for reporting an issue.

Also, in case that an error occurs, an error report is automatically generated and stored as an issue. It is displayed to the user and provides input fields to add further information. For each reported issue the user can choose to be notified via email about its progress.

Notifications are sent upon the assignment of an issue to a ticket and when a resolved ticket is included in a release as shown in the process in figure 4. Through this notification service the users do not need to check the progress manually.

Further more the users have the option to view all Issues and Tickets of each component of the portal (see previous figure). Thus they can on the one hand learn which changes are planned and on the other hand follow up the past updates to the component. Thus the understanding of the components can increase. The users might even want to contribute more proposals for further improvements because they can see that issues are followed up.

4.3.2 Example case of an issue live-cycle

In this section we shortly want to follow the stages of an issue and the resulting ticket as they are handled with the issue management system.

Once an issue has been entered as described in the previous section, it will appear in the overview list shown in figure 6. One member of the development team evaluates the issue and assigns to a new or an existing ticket. For the ticket various fields can be set to categorize it.

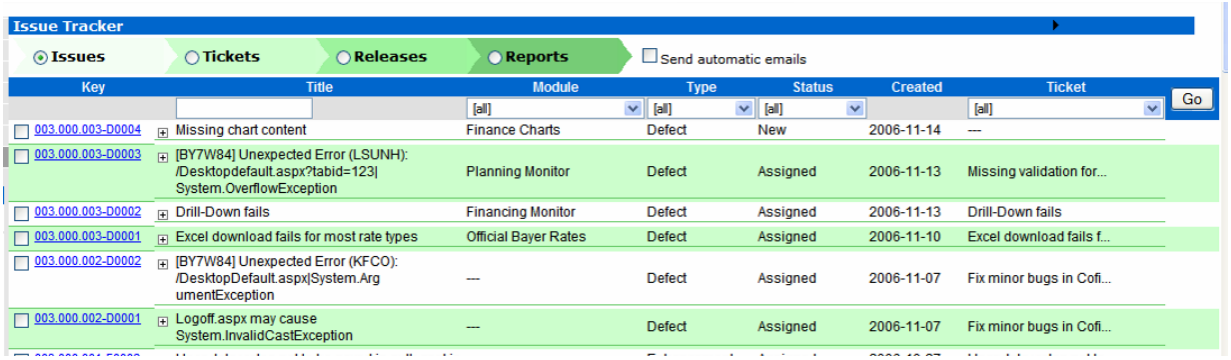

Figure 6: Issue overview for the developer-team

These fields define the priority, severity and the type of the ticket. With these settings, the next development steps of the portal can be identified. While tickets that describe major changes that lead to a major development cycle, minor issues such as bug fixes can be handled directly.

After a ticket has been assigned to a developer and it has been fixed and the solution has been tested and approved by another person, its status is changed to *verified* and it can be included in the next release. When a new release is planned the responsible administrator prepares it using the issue management system. Each release is stored in the system as a collection of all verified tickets. Upon creation a list of all these tickets is displayed as show in figure 7 including information about changes that need to be taken care of before the portal code can be released. Once the release has been published a list of all changes can be obtained through the system and is also accessible for all users of the portal.

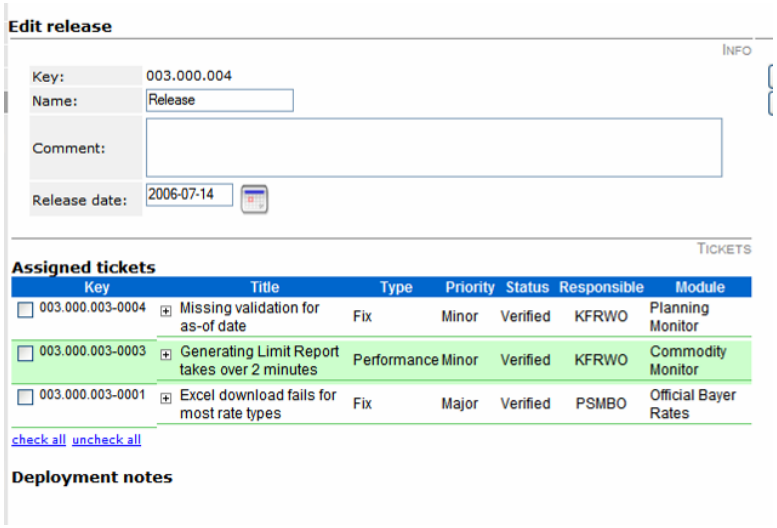

Figure 7: Release preparation form with assigned tickets

4.4 Lessons Learned

In the first months following the introduction of the issue management system, only a few selected users were allowed to access the features. But despite the restricted number, the new working process already had some significant influence on the development process.

Starting right from the beginning, the users filed all kinds of issues that they often had already had in mind for a longer time. Previously they would have needed to send an email or to directly tell it to one developer or even to several developers individually and than hope that one of them accepted or at least noted down the issue. Now they can add it to the list of open issues and know that the idea is at least in the list of open topics. Thus the barrier for reporting issues

turns out to be significantly lower than with the previous system. With the system, each issue can clearly be identified including information about its reporter and the creation date. This clear identification facilitated many talks between users and developers. They can refer to the issue and don't need to spend much effort on initially identify the problem and whether it was subject to discussion before.

Although the centralized collection of all issues significantly increased the number of open topics, it still had a positive effect for the work of the developers. First of all they do not need to enter each issue manually by themselves as they had to do with the previous Excel-based approach. Further more every user is now responsible for adding an issue to the system. The developers can ask every user that approaches them with a new issue, to report the issue in the issue management system themselves.

Comparing the number of resolved issues between the new system and the Excel-file, there is a significant difference. The old system required the developer to open the file and enter issue when it was resolved. Especially for minor changes the effort was seldom taken. There was also no immediate reason to enter each change. The new system requires each issue and resulting ticket to be processed. Thus each update is documented. For example if the developer forgets to set the status of a ticket to resolved, it will remain on his list of open tickets. Clearly he will try to reduce the number of tickets on the list and thus take care that he records each progress.

Not only did the quality of change-documentation improve but also could the work of the developers better be identified as for each ticket the actual time required to solve it is noted down. This also grants the possibility to analyse the development efforts and helps creating better estimates for new tickets.

The new process also improved the collaboration of the development team. Through the centralized system every team member can follow up all changes and open issues. The information gap between the members decreased. Through the clear assignment of tickets responsibilities became a more seldom topic of discussion. Each change and open task can now be triggered to one person. Time-consuming research to identify the right contact person is in most cases obsolete.

Future improvements will focus on further increasing the transparency of the development actions. The goal is to further encourage the users to observe ongoing issue developments. Thus they will learn more about features of the portal that might contribute to their work. Further on

they might be motivated to think about possible improvements to the system and thus contribute to its quality.
Another interesting topic for further analysis is the quality of issues who are added by the users through the portal. The question is what types of requests are made – whether they mainly describe minor issues or if they also include major change proposals. So far most reported issues did not contain the later but it must be considered, that the number of users was limited and that time for adaptation might be necessary before the users feel confident about adding more advanced issues.

In addition further analysis tools will be added to the issue management system to create reports about the development work that is performed. A long-term approach could lead to the integration of the ticket planning data into a project management application in order to improve the coordination of the developer team.

5 Conclusion

In this work we addressed evolution management in the context of portal engineering. On the basis of a portal engineering process model, that is derived from the traditional software lifecycle process we identify two different evolution cycles: minor evolution cycles that are nested in major evolution cycles. While major evolution cycles require a run through the complete portal engineering process, minor evolution cycles take places within two major cycles and by nature address smaller changes in the current portal iteration (e.g. bug fixes or the introduction of on breaking changes). Both, minor and major evolution cycles are based on feature requests, change proposals and bug reports, which must be managed and evaluated within the scope of a portal evolution management concept. We discuss the core concepts underlying an evolution management approach in theory and prove the feasibility of our claims in practice on the basis of the evolution management procedure established for the corporate financial portal at Bayer. Of course, the validity of our statements is subject to the general limitation of a case study. We cannot prove whether the framework will also be applicable on other corporate portal initiatives. Still, we believe that the core insights can also be transferred to other cases as the evolution management process and the supporting tool was not designed with any functional or organizational specifics in mind.

References

[ACLÖ04] *Alt, Rainer; Cäsar, Marc A; Leser, Florian, Österle, Hubert; Puschmann, Thomas; Reichmayr, Christian; Zurmühlen, Rudolf:*. Methode zur Entwicklung von Prozessportalen, Lösungen, Bausteine und Potenziale des Business Networking. Real-time Business. R. Alt und H. Österle., Berlin, Heidelberg, Springer 2004. p. 258-280.

[AmRB04] *Amberger, Michael; Remus, Ulrich, Böhm, Martin:* Entwicklung von Unternehmensportalen. WISU 33(5), 2004, p. 658-665.

[BrWa04] *Brown, Allen W.; Wallnau, Kurt C.*: Engineering of Component Based Systems. In: 2nd IEEE International Conference on Engineering of Complex Systems (ICECCS96), IEEE 1996.

[FiAi00] *Finkelstein, Clive; Aiken, Peter:* Building Corporate Portals with XML. New York, McCraw-Hill 2000.

[Gaed98] *Gaedke, Martin:* Web Composition: Ein Unterstützungsystem für das Web Engineering. In: GI Softwaretechnik-Trends 18(3) 1998, p. 20-25.

[GHKV04] *Gurzki, Thorsten; Hinderer, Henning; Kirchhof, Anja; Vlachakis, Joannis*: Die Fraunhofer Portal Analyse und Design Methode (PADEM) - Der effiziente Weg vom Prozess zum Portal, Fraunhofer IAO 2004.

[MDHG99] *Murugesan, San; Deshpande, Yogesh; Hansen, Steve; Ginige, Athula:* Web engineering: A New Discipline for Development of Web-Based Systems. In: First ICSE Workshop on Web Engineering, Los Angeles, Springer 1999, p. 1-9.

[Powe98] *Powell, Thomas A.:* Web Site Engineering, Prentice Hall, 1998.

[Remu06] *Remus, Ulrich*: Critical Success Factors of Implementing Enterprise Portals. In: 39th Hawaii International Conference on System Sciences 2006.

[Schw00] *Schwickert, Axel C.*: Web Site Engineering. Stuttgart, Teubner, 2000.

[Somm04] *Sommerville, Ian*: Software Engineering, Addison-Wesley 2004.