

Participatory Development Strategies for Open Source Content Management Systems

by Stefanie Panke, Christian Kohls, and Birgit Gaiser

For many software designers and developers the emergence of open source software (OSS) offers much promise for valuable innovations, particularly in the educational sector. Yet the occasional drawbacks of OSS—such as missing documentation, difficult organizational structures, limited usability, and a development process driven chiefly by technological affordances (Levesque 2004)—severely hinder its integration and adaptation and prompt general user frustration (Exhibit 1). The openness of code remains the strongest advantage of OSS insofar as it allows for greater adaptability to a specific area of application in any given case. Playing to this strength, however, requires an interdisciplinary collaboration of users and developers as well as a more comprehensive awareness of the social aspects of software design.

What does it mean to recognize software development as a distinctively *social* phenomenon? First and foremost, it means that we avoid viewing it within a narrow sphere in which technological decisions are made in a vacuum. Software development may entail a structured process of problem-solving, but it cannot simply be understood as a rational, objective, streamlined process leading directly from specification to prototypes and end-products. Rather, implicit values, personal estimations, political conditions, and individual goals all have a bearing on the evolution of technology (Schulz-Schaeffer 1996). Social informatics research, for example, has contributed much to this fuller understanding of technological development by emphasizing the interrelations between social agents and their institutional and cultural background as well as the interaction between information technologies, social acts, and social habits (Kling 1999; Lamb and Johnson 2004). In doing so, such research derives its key insights from the social, historical, and political context in which information and communication technologies (ICT) are deployed (Kling 1987).

In this article we provide an account of the difficulties we experienced in adapting and extending the open source [ZOPE/Plone](#) course management system (CMS) as a technological infrastructure for the educational Web portal [e-teaching.org](#). In doing so, we aim to characterize software design as a task that requires the active involvement of users throughout the whole process. To this same end, we also draw upon two theoretical models to analyze software design as a social phenomenon. First we employ the [web model](#) of Rob Kling (1992, 1999) as we reflect upon our experiences in the pilot phase of development; this model provides the framework for a qualitative analysis of ICT implementation, particularly with regard to the critical factors for effective collaboration. In turn, we employ the [STEPS model](#) of Floyd, Reisin, and Schmidt (1989) to illuminate the dynamic flow of decisions in more detail and to establish a foundation for a participatory software development approach. We will argue that socially focused models such as the web model as well as process-oriented models such as the STEPS model are most suitable as a framework for assessing and supporting effective collaboration in software development, design, and implementation.

Background on e-teaching.org

The portal [e-teaching.org](#) offers comprehensive information on didactical, technological, and organizational aspects of e-learning at universities and specifically targets university lecturers in German speaking areas who want to integrate digital media into their teaching (see [English demo version](#)). Since its launch in 2003, it has become a well-established information resource comprising approximately 1,000 Web pages, an extensive glossary, and diverse multimedia supplements. The portal can be flexibly integrated into hybrid formats to support the distinctive needs of various institutions. For a detailed description of the concept and structure of the portal, see Panke et al. (2004).

The construction of the portal has unfolded in roughly two phases: a pilot phase and a consolidation phase.

First, during a two-year pilot phase (2003–04) supported by the Bertelsmann and Nixdorf Foundation, a prototype of the portal was developed and tested at two universities. A major accomplishment during this time was the implementation of specific functions that allow cooperating universities to generate a localized version of the portal. So-called "university editors" can enhance the general content with location-specific information ([Exhibit 2](#)).

In turn, the main emphasis in the consolidation phase (2005-06) was on the development of a sustainable maintenance model that can support the creation of a broader e-teaching community. As the acquisition of new partner universities intensified, the requests for better usability by university editors began to carry more and more weight. An additional aim in this phase was to motivate users to visit the portal regularly by providing not only an information pool but also communication tools to foster professional collaboration. Currently, the Federal Ministry for Education and Research provides the support for the development and distribution of the portal to other universities.

Critical Factors for Collaborative Development

Here we analyze the early pilot phase of the project (2003-04) by examining the requirement specifications, the timetables, the project reports, the mailing listserv, and our own experiences. In doing so we apply a variant of the [web model](#) of Rob Kling ([1992](#); [1999](#)) as the methodological frame for our analysis. This variant of the web model identifies four central dimensions that influence the evolution of technology:

- the history of decisions;
- the agents (e.g., users and developers);
- the available infrastructure; and
- the working context.

In the case of the e-teaching.org project, the social environment of the software development process may be mapped out according to these four dimensions ([Exhibit 3](#)). The development of the editorial system on the basis of ZOPE/Plone presented itself as a complex social negotiation process between the [Bertelsmann Foundation](#) (responsible for project management), the [Knowledge Media Research Center](#) (responsible for content and concept), and the [University of Bielefeld](#) (responsible for technological implementation). In the extended context, the partner universities—the [University of Duisberg-Essen](#) and the [University of Wuppertal](#)—played a role as end-users of the information resource e-teaching.org and as university editors, supplementing the portal with local information about their institution.

Throughout this section, we will highlight findings from our analysis and relate them to the four dimensions of the web model. The case study offers insights that are applicable to other projects in the sense that they can sensitize readers to probable troubles and restraints in interdisciplinary, complex development processes.

History

In the course of a project, early decisions on IT infrastructure and obligations to users and partners tend to be basic variables for the further developmental process. For instance, once chosen, a CMS will be a steady factor for all following implementations simply because the effort required for migration will rule out any other potentially better solution. Here we focus on two central decisions for the implementation of e-teaching.org: the choice of the open source product Plone and the distribution of responsibilities for content and IT development to two remotely located institutions.

Particularly at the beginning of the technological development, the Bertelsmann Foundation, forming the project management, was involved in decision-making processes concerning the infrastructure. It turned out that the project management faced difficulties in coming to a clear decision in favor of a technological solution. Multiple consultations resulted in lengthy procedures, which impeded the launch of the portal

e-teaching.org. A very detailed requirement specification was written to prescribe the trajectory of further development. However, this document was soon caught up and overhauled by the practical working context.

In retrospect, the selection of the ZOPE/Plone CMS was a post hoc rationalized decision based upon pragmatic considerations: Plone had a fairly good reputation, ZOPE was well-known in the media technology department of the Knowledge Media Research Center, and the project manager had already established contacts with a Plone developer at the University of Bielefeld.

The project management assigned the technical development to the University of Bielefeld and the content production and conceptual design to the Knowledge Media Research Center in Tuebingen. The institutions are located 600 kilometers apart from one another, which made regular face-to-face meetings rather difficult.

From the managerial perspective, this project architecture had the advantage that the risk of having a weak partner was spread out over two groups of developers. At the same time, possibilities of control and influence were apparently strengthened for the Bertelsmann Foundation. The fact that the project manager wanted to receive the entire e-mail traffic for the coordination of the technological development indicates the scope of managerial involvement. However, it gradually turned out that because of the immense amount of communication, the project management could not accompany the finetuning and adaptation of the functions ([Exhibit 4](#)).

The more specialized the questions, and the more granular the problems that are to be solved by the IT infrastructure, the fewer the number of agents who can be involved in the decision-making process. In the editorial team at Tuebingen, one editor was assigned the task to perform as a "communication interface" with the the technology team in Bielefeld; this editor chiefly coordinated the collaboration between the two teams. So instead of gaining more influence and transparency, the strategy of the project management resulted in costly arrangements and a concentration of influence and workload on individual project members.

Agents

System development is an interdisciplinary project that only succeeds if there is frequent exchange of information between its agents. The technicians have to know about the work processes in editing and the editors have to have a basic knowledge of the technological frame. In the case of the e-teaching.org project, the different backgrounds of the respective agents sometimes created challenges for collaboration. The project partners in Bielefeld saw themselves as members of the ZOPE community and were committed to the further development of the open source infrastructure (on the role of ideology for open source development, see Stewart and Gosain [2006](#)). In contrast, the colleagues in Tuebingen identified themselves mainly with contents and the overall concept of the portal e-teaching.org. As a result, the two teams implicitly pursued different goals in their work despite the coordination between them: the Bielefeld team placed a premium on the full implementation of features and functions afforded by the OSS technology whereas the Tuebingen team focused primarily on the usability and efficiency of the portal for partner institutions.

The different specialist backgrounds also had an impact on the styles of communication and work. Debugging was part of the everyday work of technicians whereas the editors' emphasis was on quality control of published contents. The different professional cultures consequently opened up the potential for conflicts between technology and content development. Communication differences between the two groups ranged from minor issues of netiquette ([Exhibit 5](#)) to more significant issues regarding the adequate description of technical difficulties ([Exhibit 6](#)). In such cases project partners should devote time to ensuring that they recognize the conventions and communicative norms of one another's field in order to guarantee efficient collaboration.

Infrastructure

The dimension of infrastructure includes the different artifacts with which the project members work; in this

particular case, the key elements of the IT infrastructure included the Plone CMS, the means of communication (an e-mail list and an online bulletin board), and the user documentations for the editorial system. The following collaboration difficulties arose from these three components of the infrastructure setting:

Content Management: The Plone CMS and the Content Management Framework (CMS) used for building up and operating the e-teaching.org portal is found on the application server ZOPE. While these components of the IT infrastructure support one another, they also result in a segmentation of the online working environment; consequently, the agents involved as editors, technicians, and project partners work at different interfaces. Editors and university editors each have their own views of the Plone CMS Web interface, while the technicians mainly interact with the system via the ZOPE Management Interface. The formation of a "common ground" necessary for the system's development was hampered by both aspects—the varying perspectives

- **Means of Communication:** An e-mail list served as a centralized means of communication between editors, university editors, and technicians. The number of 1,200 e-mails posted on the mailing list from June 2003 to December 2004 shows the high levels of communication required for the coordination of the project. However, in order to establish better-structured communication, particularly in the debugging process, a bulletin board called the "bug collector" was set up after almost one year of project time. Even though the bulletin board offered a clear layout and added to the transparency of the process, it was nevertheless used for only three months, after which there was a general return to the established form of the e-mail list. As this situation suggests, paths of communication are difficult to change once they have been established.
- **Documentation:** The problem of insufficient documentation or lack thereof is the downside of the high flexibility and adaptability of open source products. Manuals that are incomplete, missing, or unsuitable for the target group are an obstacle to the intended use of the technological infrastructure. Often the users are unaware of the spectrum of available functions and do not use these functions at all or use them in a manner different from their (intended or original) purpose; this situation typically results in the need for further development—which, in turn, would not have been necessary had there been precise documentation and instruction in the first place. Mistakes in particular appeared during insufficiently documented ad hoc decisions, which later often produced unwanted side effects ([Exhibit 7](#)).

Context

As indicated above, the evolution of e-teaching.org took place within a very complex network of cooperation. Here we focus further on the working context of the editorial team at the Knowledge Media Research Center to illuminate a mismatch of workflow models and the actual experience of project participants.

The internal workflow of editing played a special role in the further development of the editorial system. If, as is the case with e-teaching.org, several authors work on the same Web site, the setting of areas of responsibility and competence has to be ascertained. Editorial workflow systems often require a great deal of complexity to ensure effective collaboration throughout the course of a project. In our case the requirement specification document—the basis for the task assignments during technological development—reflected the priorities of the project management team and dictated the working context for project participants (e.g., the reliance on strict hierarchies of review to ensure quality control at each stage of the project). The predefined publication process of the Plone CMS thus corresponded to the highly formalized processes of large editing teams.

However, an unsuitable workflow initially resulted from this infrastructural framework; the structure was too rigid and still not sufficiently designed to reflect the full range of decisions made during the editorial process. In response to this disconnect between structure and practice, project management adapted the requirement specification document to accommodate the workflow patterns of the editors. This transformation took place with varying success. By this time the editors had already gotten used to individual workarounds to bypass problems they encountered during different stages of development, and consequently, the new workflow model for the multiple stage publication process was largely unused (on the role of workarounds in software design, see Spinuzzi 2003). The integration of the workflow model into everyday editorial practice was further

hampered by other factors as well. Due to differences in background and professional experience, content editors and technology designers sometimes had disagreements based on their divergent models of workflow structure ([Exhibit 8](#)). Moreover, the aforementioned problems with insufficient or lacking documentation further led to the abandonment of the workflow model by many of the editors ([Exhibit 9](#)).

Lessons Learned

The analysis described above was conducted at the beginning of the consolidation phase (2005-06) and led to several changes in the project architecture and working routines.

Ensuring Effective Collaboration

Reflecting upon our experiences in the pilot project phase, we found that successful software development is not solely contingent upon the expertise of the project members working on technological and conceptual questions. It is equally important to create structures that are fruitful for a collaborative development: to offer frequent and noncostly occasions for exchanging ideas, to choose adequate technological means of communication, and to establish effective routines for testing and documentation.

We learned that different models of technology development are dependent on the subject matter, the location and professional background of participants, and the integration of work groups or institutions. Misunderstandings and conflicts occurred in the pilot phase because the agents did not verbalize and discuss their backgrounds but rather implicitly assumed their ideas of design quality were held by others. For a successful cooperation, project members need to have the opportunity to discuss and reflect upon their ideas as products of implicit models based on their own previous working experience. Hence, we aimed at harmonizing the design goals with the development group by providing more room for informal exchange and fostering a common identity.

As one step in this direction, we created a new job for a technician who would work at the same site as the editors and thereby improve communication between editors and technical staff. This step, in turn, offered greater opportunities to discuss intermediate steps and partial results of the development process in weekly editorial meetings, which significantly enhanced collaboration. Furthermore, in order to ensure transparency in the decision-making processes and the comprehensibility of negotiations at later stages, we replaced the e-mail list with a weblog to document all developmental steps of the project. The weblog is a shared artifact of all project members and contains not only technical documentation but also organizational information and a noteworthy share of informal communication as well (sporting results, photos of pets, jokes, etc.). This step has also helped participants to discuss and negotiate their respective models, anticipated expectations, and implicit assumptions regarding their roles in the development project.

The STEPS Model of Participatory Design

Upon reflecting on the pilot project phase, we also recognized that the development process in our case corresponded to the cyclical prototyping approach characteristic of other OSS development projects, and we concluded that having a shared conceptual model would be beneficial for interdisciplinary collaboration. In our search for such a model, we selected the STEPS model of Floyd, Reisen, and Schmidt (1989), which views software design as a learning process shared by developers and users and emphasizes intensive communication between these groups throughout the process.

The cyclical process description in this model provides a theoretical basis to plan the specification, development, implementation, and evaluation of software while assigning clear responsibilities to both the users (in our case, the editors) and the technicians. Both parties play an active role in the production and usage of the software, which thereby incorporates the experience of users directly within the design process; in effect, the users serve as co-designers of the system from one prototype to the next. The cyclical structure additionally allows for counteracting unrealistic expectations of quality when the prototype is first applied by

the users and thus allows for more specialized and focused feedback from users at each stage of the process. In short, the STEPS model makes clear that test-usage is an important part of establishing and maintaining a software development project. This factor is particularly crucial for the design of groupware, which requires an intensive amount of "on-road testing" within the production process (cf. Pankoke-Babatz et al. 2001).

We introduced the STEPS model by discussing the results of our web model analysis within the project team. In doing so, we recognized that our particular approach to development and design required adjustments, and we employed the model to give clearer definition to these adjustments. We eventually developed our own adapted variant of the STEPS model that takes the specific context of content management environments into account ([Exhibit 10](#)). The model reflects a focus shift from creating a core running technical infrastructure to improving the usability of the system and developing features in a more user-oriented way. Feature specification, testing, and documentation are now understood as common tasks for all project members.

One of the first common tasks assigned to one of the editors and the new technician was that of rewriting the user documentation for the university editors in order to give the document a task-oriented structure. This assignment revealed a high amount of usability problems that had been neglected during the rapid, feature-centered development of the initial prototype ([Exhibit 11](#)). The documentation review and revision process eventually resulted in a complete redesign of the localization feature of the portal; redundant, unnecessary, or otherwise problematic functions of the system were stripped away in order to make the portal much more user-friendly than in its prior version. Due to the reduced and concise functionality, the university editors use the CMS much more frequently and the portal enlists a growing number of [partner universities](#); currently more than 40 universities are cooperating with e-teaching.org to make the portal a vital part of their e-learning strategy.

Conclusion

We cannot present foolproof recipes for other projects to follow; as a complex social activity, the software development process depends greatly on the specific context of the project in question. However, our experience does have broader implications for similar collaborative projects, particularly those that entail OSS development to serve the needs of diverse groups.

Like other open source systems, the [Zope/Plone](#) architecture allows rapid prototyping and an efficient development of new features. However, as our experience has revealed, having a large number of features is no end in itself—neither for developers nor for end-users. A feature-centered approach to development may be satisfying for the technicians in the beginning, but as soon as they are confronted with negative feedback from real users, enthusiasm can change to frustration. Users do not necessarily appreciate a wide range of features because this only complicates the learning process. In many cases the quality of the interface design and the technical documentation influence user satisfaction more effectively than coding add-on functionality. The time saved in rapid application development therefore should be invested into collaboratively reflecting upon the expectations and working context of the users, and this requires a participatory development strategy that gives users a stronger role as co-designers throughout the process.

In turn, a participatory development strategy is only valuable if it provides sufficient opportunities for all project members to discuss and reflect upon the different models they have come to follow as a result of their previous experience as editors, technical designers, or project managers in other professional environments. Whether the issue involves workflow patterns or approaches to troubleshooting and problem-solving, participants may often assume that their implicit expectations are shared by others only to discover that they need to negotiate these matters explicitly with their colleagues in order to avoid misunderstanding. Maximum transparency in communication throughout the whole course of the project will thus remain a key foundation for ensuring a productive social environment for collaborative work.

Finally, a comprehensive preliminary analysis of the project—a holistic, Web-based assessment of its history,

agents, infrastructure, and working context—as well as the introduction of effective participatory process models can help reconcile many differences in background among participants (in our case, the debugging practices of technology designers versus the quality control practices of editors). Cyclical process-oriented models of development such as the STEPS model can help provide a clear foundation for structured collaboration among the entire project team and thereby ensure that participants are fully aware of what stage of maturity and development the software has reached at any given point in the cycle. If participatory design will always remain a challenge in practice, such models will allow planners and participants to minimize the obstacles to collaboration while ensuring that their work consistently serves the needs of their intended users.

References

Breuer. 2004. Anwender-das dicke problem dei open source. Posting from the German Weblog "Notizen aus der Provinz." http://notizen.typepad.com/aus_der_provinz/2004/08/anwender_das_di.html (accessed November 28, 2006).

Fink, M., M. Janneck, and H. Oberquelle. 2004. Benutzergerechte Gestaltung von CSCL Systemen. In *Wissensprojekte. Gemeinschaftliches Lernen aus didaktischer, softwaretechnischer und organisatorischer Sicht*, ed. B. Pape, D. Krause and H. Oberquelle, 203-219. Münster: Waxmann.

Floyd, C., F. M. Reisin, and G. Schmidt. 1989. STEPS to Software Development with Users. In *ESEC '89, Lecture Notes in Computer Science*, ed. C. Grezzi and J. A. McDermid, 48-64. Berlin: Springer.

Greene, J. 2005. Interview statement on the problems of applying scientific methods and models in software development. <http://press.oreilly.com/pub/pr/1475> (accessed November 28, 2006).

Kling, R. 1987. Defining the boundaries of computing across complex organizations. In *Critical Issues in information systems research*, ed. R. J. B., Jr. and R. A. Hirschheim, 307-362. New York: John Wiley.

Kling, R. 1992. Behind the terminal: The critical role of computing infrastructure in effective information systems' development and use. In *Challenges and strategies for research in systems*, ed. W. Cotterman and J. Senn, 365-413. New York: John Wiley. <http://rkcsi.indiana.edu/archive/kling/pubs/webinfra.html> (accessed November 28, 2006).

Kling, R. 1999. What is social informatics and why does it matter? *D-Lib Magazine* 5 (1): 217-232. http://rkcsi.indiana.edu/archive/kling/pubs/kling99_01.pdf (accessed November 28, 2006).

Lamb, R., and S. Johnson. 2004. Social aspects of digital information in perspective: Introduction to a special issue. *Journal of Digital Information* 5 (4). <http://jodi.tamu.edu/Articles/v05/i04/editorial/> (accessed November 28, 2006).

Levesque, M. 2004. Fundamental issues with open source software development. *First Monday* 9 (4). http://firstmonday.org/issues/issue9_4/levesque/index.html (accessed November 28, 2006).

Otwell, A. 2001. Comment to a blog-posting on eleganthack.com. http://eleganthack.com/archives/holding_the_vision.php#001487 (accessed November 28, 2006).

Panke, S., J. Wedekind, J. Reinhardt and B. Gaiser. 2004. www.e-teaching.org—Qualifying Academic Teachers for the E-University. In *Proceedings of ECEL 2004, European Conference on e-Learning*, ed. D. Remenyi, 297-306. Reading, UK: Academic Conferences International.

Pankoke-Babatz, U., W. Prinz, V. Wulf, and M. Rohde. 2001. Spezifika des CSCW-Designs. In *CSCW Kompendium - Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten*, ed. G. Schwabe, N.

Streitz and R. Unland, 373-394. Berlin: Springer.

Schulz-Schaeffer, I. 1996. Software-Entwicklung zwischen Ingenieur- und Designwissenschaft. Überzeugungskraft und nützliche Widersprüchlichkeit von Software-Engineering und Software-Gestaltung. In *Technikleitbilder auf dem Prüfstand. Leitbild-Assessment aus Sicht der Informatik- und Computergeschichte*, ed. H. D. Hellige, 115-140. Berlin: Edition Sigma.

Spinuzzi, C. 2003. Tracing genres through organisations: A sociocultural approach to information design. Cambridge, MA: MIT Press.

Stewart, K. J., and S. Gosain. 2006. The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *MIS Quarterly* 30 (2): Forthcoming.

<http://opensource.mit.edu/papers/stewartgosain2.pdf> (accessed November 28, 2006).

Veen, J. 2004. Making a better Open Source CMS. Blog-post on his personal Weblog.

<http://www.veen.com/jeff/archives/000622.html> (accessed November 28, 2006).

COPYRIGHT AND CITATION INFORMATION FOR THIS ARTICLE

This article may be reproduced and distributed for educational purposes if the following attribution is included in the document:

Note: This article was originally published in *Innovate* (<http://www.innovateonline.info/>) as: Panke, S., C. Kohls, and B. Gaiser. 2006. Participatory Development Strategies for Open Source Content Management Systems. *Innovate* 3 (2). <http://www.innovateonline.info/index.php?view=article&id=326> (accessed April 24, 2008). The article is reprinted here with permission of the publisher, [The Fischler School of Education and Human Services](#) at [Nova Southeastern University](#).

To find related articles, view the webcast, or comment publically on this article in the discussion forums, please go to <http://www.innovateonline.info/index.php?view=article&id=326> and select the appropriate function from the sidebar.