Towards an Open Source Development Process - Evaluating the Migration to an Open Source Project by Means of the Capability Maturity Model

Wolf-Gideon Bleek
University of Hamburg
Department for Informatics
Software Engineering Group
Hamburg, Germany
wbleek@acm.org

Matthias Finck
University of Hamburg
Department for Informatics
Applied and Social Informatics
Hamburg, Germany
finck@informatik.uni-hamburg.de

Bernd Pape
University of Hamburg
Department for Informatics
WissPro Research Group
Hamburg, Germany
pape@wisspro.de

***Abstract* – In this paper we review the ongoing development of a Web-based community system that has been migrated from a closed software development to an open source project. We identify three different phases in the migration process where the development process changed significantly. We analyse these phases by means of the Capability Maturity Model (CMM). The insights gained show the implications of such a migration process towards open source concerning the process quality of a development process. They also show underlying assumptions of the CMM that do not totally match with developments in this specific case study.**

As a helpful outcome, our reflection about the ongoing software development process helped identify two crucial factors: reflection about the process is possible even at lower levels and how to handle people's fluctuation to sustain a development project.

I. INTRODUCTION

For many researchers at university innovative software development is part of their work. As a result of one such development activity, a Web-based community system is being developed since 1999 at the Department for Informatics, University of Hamburg. Despite the innovative character of this software, it has always been necessary to find a common organisational frame for hosting this project due to normal staff fluctuation. Striving for a quality development process – and thus a quality software – has therefore continually been a challenge.

In 2003 the existing development project became an open source project. The characteristics of such a project seem to offer a suitable organisational frame for addressing the problem: recurring changes of general conditions and fluctuation of participants. However, changing the process fundamentally holds the danger of decreasing the quality of the development process. To further investigate the implications of such a migration process concerning the software development's quality we analyse the phases of the development process before and after being on the verge of becoming an open source project by means of the Capability Maturity Model (CMM). Based on our case study we will show how quality aspects of a software development processes will change when it is migrated from a closed software development to an open source software project.

Therefore we take a software engineering perspective on software development processes with an organisational point of view. While the CMM helps us to discover the positive and negative implications concerning the migration process, our case study gives indications about the suitability of CMM evaluating development processes with high fluctuation.

This paper is organised into six sections. Following this introduction, Section 2 motivates the need for an open source development process. Section 3 provides an abridged description of the Capability Maturity Model. Section 4 gives a detailed overview of the project's phases at the verge of becoming an open source project with regard to the CMM. It therefore first briefly describes the functional orientation of the Web-based community system, then depicts four categories the phases will be characterised by, and finally describes each phase in detail. Section 5 provides a discussion of the observed findings in that it first reviews the findings of the project analysis and then lists limitations of and differences from the underlying assumptions of the CMM. Section 6 concludes by summing up the lessons learned by applying the CMM and how these can be used for future work.

II. THE NEED FOR AN OPEN SOURCE DEVELOPMENT PROCESS

CommSy stands for community system. It is a Web-based system to support the communication and co-ordination in working and learning groups. The CommSy development founds on a participatory and evolutionary design process, which is based on the STEPS model [9, 17]. The design process is considered an integrated organisational and software development [20, 29]. User participation has taken place during the entire design process, from 1999 until today. The participatory design approach and the application of evaluation methods provide an environment for user participation. The CommSy design process builds on at least three strategies:

- The design process has been iterative, including system design, usage and evaluation tasks;
- These tasks have partly handled concurrently and are not divided in separate phases;
- By applying a mix of participatory design methods and feedback channels we guarantee close collaboration with CommSy users. We applied, among others, interviews, focus groups, log file analyses, questionnaires, prototyping (cf. [26], [27]). With these methods, we are able to receive feedback about CommSy features as well as about its usage.

In accordance to Floyd [10] we perceive software development projects as a knowledge project in which knowledge building is a shared activity among all project participants [11].

In contrast to classic software development projects that aim at developing a product, the development of software at the university happens in a different environment and therefore faces different challenges. A major difference to

Proceedings of the First International Conference on Open Source Systems
Genova, 11th-15th July 2005
Marco Scotto and Giancarlo Succi (Eds.), pp. 37-43

classic software development is the high fluctuation of students and researchers. People from different organisational frames (e.g. departments, research groups) may join the project. Moreover, the organisational frame for a software development project is more volatile and short-handed than in commercial contexts.

To establish a permanent organisational frame, an agreeable software development model and a platform that allows people from different organisations to join the project, the transformation of the development project into an open source development process was seen as a viable solution.

Characteristics of Open Source Development Processes

Open source projects have become very popular in recent years. Even formerly clearly commercial software is developed in open source projects these days (e.g. [7, 12]).

Current literature presents different views on open source software development. Some e.g. give a conservative interpretation as classic software engineering projects [28], some view it as a special type of academic research [3], and others see open source software development as a highly political activity [25].

With respect to the development process, an open source project is usually managed by volunteers. Each person needs to find its own interest in participating and contribution to the project. Commonly, nobody is getting direct financial compensation from the open source development activity. Joining and leaving the project is at the own will of each person. Participation in the project is open to any outside person that qualifies for any contribution.

Due to the freedom of participation there exists no limitation in respect to spatial, temporal or institutional distribution. Most of the coordination and communication work is done with electronic media, e.g. e-mail, online discussions forums, and Web-based bug-tracking systems [5, 24]. Becoming a member of an open source project is based on the fact that one contributes actively to the project by e.g. fixing a known bug or implementing a feature presented on a "roadmap".

The process of an open source software development is commonly performed in a fashion similar to agile software development (cf. [1]). This is mainly because of four reasons: First, the number of documents produced in advance of and during development is relatively small. Code and prototypes need to speak for themselves. Second, the team structure is flat, and even though the number of participants may be large, the actual number of people working on the same area of the software is fairly small. Third, the assignment of tasks is ad-hoc (on a favour basis) and the tasks are small. Fourth, the release cycles are short (less than 3 months).

To characterise a software development as an open source process, literature agrees on three essential characteristics that must be fulfilled [8, 18, 21]:

- Openness: The project must be open to new participants, e.g. new developers can get involved; existing barriers are easy to overcome for them. Furthermore, openness means that anybody can use the product by simply installing it. It also means that the process itself is open to changes, which leads to the next point:
- Distributed: The participants of the development process are not necessarily all located in the same place or are members of different organisations.
- Agility [1]: The development process itself must be agile in the sense that the process is carried out in short cycles and may be changed as needed. Minor releases are common artefacts within the development process and publicly visible. Further development is driven by experiences made with minor releases.

Tailoring the CommSy Development Process

The existing development process already met a number of characteristics of the open source development processes aforementioned. All criteria of open source development activities were goals of the development team. It therefore seemed to be a good approach to transfer the development activities to an open source platform. By doing so the development team established a common basis for present developers that also signalled and allowed new developers to join the project. By referencing to an external open source development platform the team also secured a stable organisational frame even though developers may change their organisation and join or leave the project.

However, how the change of the process organisational frame affects the process quality remains an open question. As a proper means for evaluation we need a theoretical framework. In the next chapter we thus describe the selected theoretical framework, the Capability Maturity Model, we will use to further investigate the development process' situation before and after the transformation.

III. THEORETICAL FRAMEWORK: CAPABILITY MATURITY MODEL

Process Evolution: Five Levels of Maturity

The Capability Maturity Model (CMM) is a framework for characterising the software development process [15, 22, 23]. Its purpose is to help assess an understanding of the capabilities of the development team by providing five steps of process evolution: initial, repeatable, defined, managed, and optimising. Development teams capable of one level's features may evolve to the next level. However, they are likely to degrade to a lower level in a crisis situation [15, p. 75]. The five levels are characterised as follows:

- *Initial*: The process is mainly "ad hoc". The characteristics of the process change at will or with change of personnel. Controlling the process is hard (e.g. there is no data available how long it took to implement a certain feature). While organisations at the "initial process" level may have formal procedures for project control, there is no management mechanism to ensure they are used. "The best test is to observe how such organisations behave in a crisis. If it abandons established procedure and reverts to merely coding and testing, it is likely to be at the Initial Process level" [15, p. 75]. Organisations at the Initial Process level can improve their performance by instituting basic project controls. The most important are: project management, management oversight, quality assurance, and change control.
- *Repeatable*: The process is stable and the data collected in the process is available for reviewing.

Commitments, costs, schedule and changes are managed. The Repeatable Process has one important advantage over the Initial Process: It provides commitment control. New challenges likely to pose a risk for the process are: new tools and methods which affect how the process is performed, new kinds of products that must be developed, and major organisational changes. Key actions necessary to advance to the next level are: establishing a process group, establishing a software development process architecture, and introducing a family of software engineering methods and technologies.

- *Defined*: The organisation has defined the process to ensure consistent implementation and provide a basis for its better understanding. At this point, advanced technology can usefully be introduced. The Defined Process characterises a stable software developing organisation. Even in a situation of crisis, the development group will continue on its path. This opens up the possibility of systematically examining the process and decide on how to gradually improve it Key actions necessary to advance to the next level are: establishing a minimum, basic set of process measurements to identify the quality and cost parameters of each process step, establishing a process database with the resources to manage and maintain it, providing sufficient process resources to gather and maintain this data and advising project members on its use, and assessing the continuous quality improvement.
- *Managed*: The process is managed in a comprehensive manner. Significant quality improvements begin when these managements become effective. The Managed Process is about gathering data. Its problem are costs that emerge with related tasks. "There are an enormous number of potentially valuable measures of software development and support, but such data is expensive to gather and maintain" [15, p. 77]. Process data must not be used to compare projects or individuals. Its purpose is to illuminate the product being developed and to provide an informed basis for improving the process. Fundamental requirements to advance to the next and final level are the support of automated gathering of process data and the use of this data to analyse and modify the process to prevent problems and improve efficiency.
- *Optimising*: The organisation is able to discuss the process on a meta-level, which enables it to adapt the process to match its needs. The Optimising Process addresses not only the optimisation of the product, but the process itself. While on all other levels optimisation takes place to optimise the output (the product) here the process is the object to be optimised. "In the Optimising Process, the organisation has the means to identify the weakest elements of the process and fix them" [15, p. 78].

Applying the Theoretical Framework

The capability maturity model is an analytical framework that allows identifying the maturity of a software development process.

A helpful characteristic of the framework is that we can apply the CMM for evaluation in retrospect, enabling us to assess the quality of each phase of the development process with the data at hand. This will generate a rating of each phase allowing us to evaluate the effects against the development process after switching to an open source development model.

IV. EVALUATING THE COMMSY DEVELOPMENT PROCESS

In the last six years, CommSy has been used predominantly in university contexts to support project-based learning [19, 16, 13]. It supports communication (for example news and discussion forums) and the exchange of working materials (e.g. with file uploads and online documents) as well as project organisation (aided by dates and groups). CommSy consists of three parts:

- The project rooms are designed for closed learning groups of approximately 10 to 30 members and normally co-operate for a limited period of time.
- The CommSy community room is an archive "in progress" designed to support teaching and learning individuals and groups over a longer period of time.
- The CommSy portal serves as a single entry point and provides information for inexperienced users.

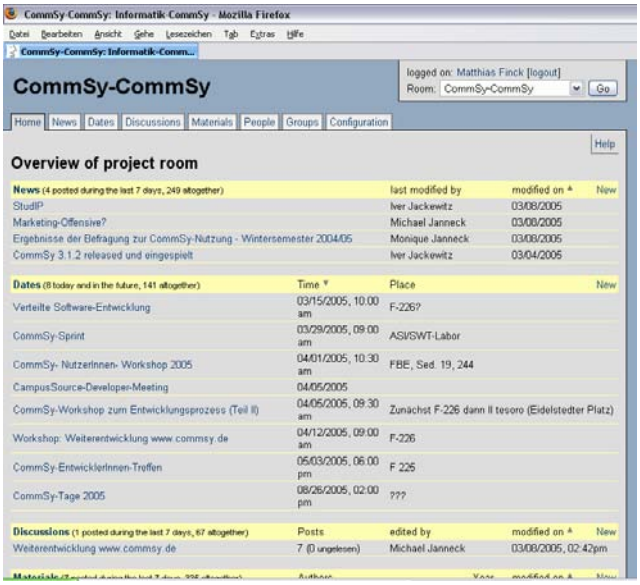

Fig. 1: Screenshot of a CommSy-project room

By applying CMM, we analyse the capability of the different phases and the variation in quality of the development process along the timeline. While the CMM has not been used to assess and improve CommSy development during the past process, we apply CMM in this paper in order to derive hints for possible improvements of such a transition.

To gain a systematic understanding, we describe each phase of the development process in reference to four key characteristics:

- *The Product* section describes what CommSy looked like in the different phases and for which target audience it was designed.
- *The Organisational Frame* section describes external conditions of the development process and institutional settings.
- *The process* section illustrates characteristics of the development process.
- *CMM Classification* provides an assessment using the CMM.

On the basis of the characteristics concerning product, organisational frame and process, the migration process of CommSy development can be divided into three phases. In the following sections, we describe these phases in detail and assess them accordingly.

Phase 1: CommSy as a Closed Source Project -- Making Ends Meet

From summer 1999 to the end of 2000 CommSy's development was carried out as a students' and researchers' project.

Due to the growing use of the system, development activities were eclipsed by increased efforts for hosting and supporting.

Moreover, coaching tutors and students as well as offering services to the general users was a pressing need. To handle all necessary tasks, a co-operation of the CommSy team with a professional service provider was initiated. However, co-operation with the service provider came to an end as it did not satisfy the development team's expectations. Therefore, the development team tried to find other ways to ensure the development and provision process. As a result, in March 2001 a publicly funded research project by the Federal Ministry of Education and Research called “WissPro” was acquired to sustain CommSy development. The research project comprised of development, provision, evaluation, and research on basic concepts (e.g. pedagogical, usability). By the help of this project new versions of the system were frequently released.

In October 2002, a severe problem of the current system version pointed at some factors concerning the quality of the development process. The new version was implemented under high pressure to meet a deadline. This system version was slow due to its flawed architecture and data handling design. In order to meet the deadline basic principles of software process and quality management were abandoned. However, experiencing the results of that abandonment first-hand raised awareness of the need for a controlled software process. From that time onward, the development process became more coordinated and returned to its previous quality.

The Product: The community system became stable until the end of phase 1. The frequency of new prototype releases changed, depending on the focus of the development. The development focus was continuously shifting between technical issues, i.e. mechanisms to handle more than one project room were added to CommSy, or issues concerning the user interface. Some automated tests were introduced to cover the basic technical environment. Due to the aforementioned crisis, the user interface did not change significantly between October 2002 and March 2003. Consolidation of the architecture had the highest priority. By March 2003, the architecture was completely redesigned.

The Organisational Frame: When the publicly funded research project started, most of the developers were employed in the research project. This research project formed the organisational frame for CommSy development and its provision. Its hosting organisation was the Informatics Department at Hamburg University. The Federal Ministry of Education and Research provided funding for a limited period of three years. The WissPro project was a new institutional frame for the development process. All members of the development team joined the research project or were associated with it. Many developers were not directly paid for developing CommSy but rather had a job that included the development of CommSy. Commsy was no longer designed primarily in their free time.

The Development Process: With WissPro, the development team consisted of fully paid researchers and part-time student workers. The project manager of the research project was also the project manager of the development process. Since the research project deals with knowledge projects, the development team decided to design the process as a knowledge project [10]. This was the first time the development team typified its process. They organised the co-operative development process by arranging regular meetings, documenting the process and determining deadlines. The prototypes were planned in a strategic and goal-oriented way. Based on the experience of the project's crisis, the developers restructured their team. Two people full-time software developers had to manage the development process. Thus, the CommSy development team established a process group that organised the process explicitly and took care of software quality. The goal was a *defined* development process.

CMM Classification: Before WissPro, the process quality was at level 1. The development team did not have a formal procedure for project control and the characteristics of the process were strongly bound to the team members. Tools were neither well-integrated with the process nor uniformly applied. Management of the involved people was informal. Students and researchers interested in the development engaged themselves without formal commitment. The amount of time contributed was based on personal interest and availability.

Before the start of WissPro, first management structures were established. The process became more organised and some software-based tools were used to support the co-ordination of the development process. We espoused to follow the STEPS method. At that point, the process could be classified as level 2.

In WissPro, the development of the next software version became repeatable. A process management was established and process documentation started. The development team grew and the development process was not as dependent on a single person as before. Due to the crisis, the full-time software developers defined the development process in detail. It was the first time a process group focussing exclusively on improving the software development process or a software development process architecture that described the technical and management activities required for proper execution of the development process was established. Thus at the end of phase 1, the quality of the CommSy development process reached level 3.

Phase 2: CommSy Becomes Open Source

Foreseeing the end of the WissPro research project, actions were taken to sustain the development project. People developing CommSy would be scattered throughout different organisations or forced to develop CommSy in their free time. An open source development process seemed fitting as it provides a structure outside the project that allows to incorporate people from different organisational frames. Measures were taken to migrate the development to an open source process. As the platform

for open source development, SourceForge was chosen.

The Product: During phase 2, an entry point to the system called campus was introduced, the interworking of campus and project rooms was established and CommSy's interface was redesigned based on the results of the evaluation.

The Organisational Frame: It was still the research project WissPro that provided the organisational frame, even as its end was foreseeable. During this phase almost every developer contributing to the open source project was still paid by WissPro.

The Development Process: CommSy became an *open source software* by placing it on SourceForge in March 2003. Due to the fact that anybody was able to download the source code of the system, the quality of the code and its documentation increased because developers were anxious about their reputation. Project management continued by facilitating the SourceForge platform (documentation and reviews), making the process publicly visible. In addition to the project manager a manager for interface design was introduced. Efforts were now measured to make outlays visible and use them for calculating future time estimation.

CMM Classification: During phase 2, process characteristics of the first phase were maintained. Placing the source code at SourceForge increases the process quality, because e.g. using the request trackers at SourceForge gave a better overview of the number of tasks and the average time the project team needs to handle it. The management of the process became more explicit. Calculations on the efforts spent and planned were established. The imminent end of the research project advanced solid models and calculations to sustain the project after the end of WissPro. The explicated process characteristics allow a process classification at level 3 with tendencies toward level 4.

Phase 3: CommSy -- an Open Source Project

In January 2004, funding of the research project ended. The project's organisational frame expired. To ensure sustainability in developing and providing CommSy, the development team looked for new opportunities to fund the open source software development of CommSy. As in 2002, a major release was planned for October 2004, but in contrast to the crisis in phase 1 the process management made sure that no second crisis arose.

The Product: For September 2004, a major release of CommSy was implemented completely redesigning the user interface. After September 2004, the consolidation of the architecture and the interface had the highest priority.

The Organisational Frame: Entering this phase, the organisational frame changed dramatically. There was no longer a single institution hosting the CommSy development. People formerly employed by the project were now forced to work in new jobs. However, everybody involved in the project was eager to continue developing the system because of the high team spirit and wide product acceptation. People were scattered in different organisational frames. This changed the structure of the team significantly. Moreover, not everybody was able to work primarily on the development of the system during work time.

To provide a permanent organisational frame, a development project was founded as a project within the HITeC association. This association resides on campus of the Informatics Department and aims at projects for technology transfer, plus functioning as an incubator for start-up companies.

People are able to join the project (paid and unpaid) by simply becoming members of the association's project. People working for other organisations join the project out of interest and the association may employ people to ensure constant care-taking of selected tasks and topics.

The Development Process: Due to the changing organisational frame, the development team had to open the development process to allow for people from different organisations to participate. The developers could not meet each other as often as before. Face-to-face communication decreased. New media are used to co-ordinate the development process.

CMM Classification: By transferring the research project's organisational frame to the association, some aspects of the process quality have been lost. A new process model has not been completely defined, but project management and oversight are carried out by a HITeC representative.

The remaining development team carries forward the experiences of the former development process and provides the necessary know-how in developing CommSy and organising the process through the years (e.g. [4]), but still there are challenges with the new situation. Tasks of process management and the corresponding roles have to be re-adjusted.

These are the reasons why the process no longer meets all criteria of level 3 and has fallen back to level 2. Level 3 could be reached again when the development team copes with the readjustment challenge of process management.

V. DISCUSSION

Migrating to an open source project poses a great challenge for the whole CommSy project. Due to its stability, the organisational frame of WissPro was an important factor that allows a classification of the process quality reaching level 4 of the CMM. This stability ended in 2004 and so the CMM assessment decreases significantly. This raises the question whether the decision to migrate CommSy development had negative effects concerning the process quality.

Taking a closer look at the migration process and its consequences for the development process, this can be negated. While the radical change of the organisational frame was inevitable, the migration to an open source project evoked two effects that have shown to increase process quality:

- A continuing organisational frame: Establishing an open source project provides a basis for continuity concerning the organisational frame which is rather unusual for development projects at university.
- Flexibility concerning fluctuation: The organisational frame of an open source project is designed to handle high fluctuation, e.g. team members and size. Due to this fact an open source project seems to be an ideal organisational frame for long term developments at university.

Reasons that lead to an unfavourable assessment of phase 3 cannot be found in the migration to an open source project. While the changing organisational frame was inevitable one result of the CMM evaluation is that

becoming an open source process has no negative long term consequences concerning the process quality for this type of project. The main reason why the CMM assessment of phase 3 was less favourable than before is that the migration process is not complete yet, but the problems of phase 3 seem to be results of the changing organisational frame and will probably be solved after the transition phase – when new roles and processes will be established.

Even though the CMM provides a detailed characterisation of each phase during the project's development, it is necessary to critically review these characterisations. There are a number of characteristics of the actual development process, e.g. flexibility, which are of high value for the development team members. Especially a highly dynamic organisational frame is not specifically addressed by the CMM. It is a quality of the analysed development process to sustain quickly changing organisational conditions. This condition and the characteristics are not taken into account by the CMM. Moreover, some of these characteristics even lead to lower levels in the maturity model while they are considered a strength.

Another underlying assumption of the CMM is that reflecting about and optimising the process (level 5) can only be reached when all preceding levels have been reached. Based on our project observation, we do not concur with this statement. We rather think that by keeping project management activities at level 2 and still reflecting about the process we ensure flexibility (and even agility in the sense of [14, 2, 6]) in a way that other processes compliant to CMM levels higher than 2 cannot achieve.

VI. CONCLUSION

In this paper, we have evaluated a long-term development project by applying the Capability Maturity Model. Each of the different phases has been briefly characterised and classified according to the CMM. By doing so, not only we have learned about the ups and downs of a project becoming an open source project, but also laid open certain differences and limitations of the CMM. Given the special needs of innovative research projects, we have learned about the model that processes regarded high in terms of the CMM lack the necessary flexibility.

Furthermore, the model's underlying assumption that reflective knowledge and action about the process can only be gained at the topmost level is not valid in our case.

With regard to the concrete project, we have also learned that the extreme changes in the organisational frame prevent the project's quality improvement. These can be interpreted as setbacks from which the project needs to recover. As a result, the careful, continuous development of the organisational frame for the project is necessary to gradually improve the project's process and the product's quality. With this precondition fulfilled, the development project is able to cope with a slow fluctuation of people.

The analysis of our case study reveals that the migration of a long-term development project at university to an open source project seems to be an appropriate solution to sustain high process quality. Becoming an open source project is an adequate option for such developments to handle fluctuation and offer a continuous organisational frame.

VII. REFERENCES

[1] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile software development methods. review and analysis*, Technical Report 478, VTT Electronic, Espoo, 2002.

[2] K. Beck, *Extreme programming explained: embrace change*, Addison-Wesley, Reading, Mass., 2000.

[3] N. Bezroukov, "Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism)", *First Monday*, vol. 4, No. 10, 1999.

[4] W. G. Bleek, and Finck, M., "Migrating a Development Project to Open Source Software Development," in J. Feller, B. Fitzgerald, S. Hissam, K. Lakhani, editors: *Collaboration, Conflict and Control: Proceedings of the 4th Workshop on Open Source Software Engineering*, 04, Edinburgh, Schottland, pp.9-13.

[5] Bugzilla. http://www.bugzilla.net/. last visited 8-Mar-2004.

[6] A. Cockburn. *Agile Software Development*, The Agile Software Development Series. Pearson Education, Inc, Boston, 2001.

[7] Legal resources for eclipse.org. http://www.eclipse.org/legal. last visited 8-Mar-2004.

[8] J. Feller and B. Fitzgerald, "A framework analysis of the open source software development paradigm," in W. Orlikowski, P. Weill, S. Ang, and H. Krcmar, editors, *21st Annual International Conference on Information Systems*, Brisbane, Australia, December 2000, pp. 58-69.

[9] C. Floyd, W.-M. Mehl, F.-M. Reisin, G. Schmidt, and G. Wolf, "Out of scandinavia: Alternative approaches to software design and system development," *Human-Computer Interaction*, 4(4): 1989, 253–350.

[10] C. Floyd, Towards Knowledge Co-construction, in C. Floyd, G. Kelkar, S. Klein-Franke, C. Kramarae, C. Limpangog, editors, *Feminist Challenges in the Information Age*, Opladen, 2002, pp. 203-222..

[11] C. Floyd and B. Pape, u.M.v.: W.-G. Bleek, I. Jackewitz, I., M. Jeenicke, „Softwareentwicklung als Wissensprojekt - am Beispiel der CommSy-Entwicklung," in B. Pape, D. Krause, and H. Oberquelle, editors, *Wissensprojekte. Gemeinschaftliches Lernen aus didaktischer, softwaretechnischer und organisatorischer Sicht*. Waxman, 2004, pp. 389-411.

[12] Gnu public license. http://www.linux.org/ info/ gnu.html. last visited 4-Mar-2004.

[13] O. Hankel, I. Jackewitz, and M. Strauss, "Technical and didactical scenarios of student-centered teaching and learning," in M. Kerres and B. Voß, editors, *Digitaler Campus. Vom Medienprojekt zum nachhaltigen Medieneinsatz in der Hochschule*, Waxmann, Münster, 2003, pp. 411–419.

[14] J. A. Highsmith III, *Adaptive Software Development – A Collaborative Approach to Managing Complex Systems*. Dorset House Publishing, New York, 1999.

[15] W. S. Humphrey, "Characterizing the software process: A maturity framework," *IEEE Software*, 5(2):73–79, 1988.

[16] I. Jackewitz, M. Janneck, D. Krause, B. Pape, and

M. Strauss, “Teaching social informatics as a knowledge project,” in T. van Weert and R. Munro, editors, *Informatics and the Digital Society: Social, Ethical and Cognitive Issues*, Boston, Kluwer, 2003, pp. 261–268.

[17] K. Kilberth, G. Gryczan, and H. Z¨ullighoven, *Objektorientierte Anwendungsentwicklung*. Vieweg Verlag, 2. edition, 1994.

[18] T. O’Reilly, “Lessons from open source software development,” *Communications of the ACM*, 42(4): 1999, pp. 32–37.

[19] B. Pape, W.-G. Bleek, I. Jackewitz, and M. Janneck., “Software requirements for project-based learning –commsy as an exemplary approach,” in *Proceedings of the 35th Hawaii International Conference on System Sciences*, Big Island, Hawaii, January 2002. IEEE Computer Society.

[20] B. Pape and A. Rolf, „Integrierte Organisations- und Softwareentwicklung für kooperative Lernplattformen in der Hochschule,“ in B. Pape, D. Krause, and H. Oberquelle, editors, *Wissensprojekte. Gemeinschaft-liches Lernen aus didaktischer, softwaretechnischer und organisatorischer Sicht*, no. 27 in Medien in der Wissenschaft, Waxman, Münster, New York, München, Berlin, 2004, pp. 287–310.

[21] E. S. Raymond, *The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary*, O’Reilly, Bejing, 1st edition, 1999.

[22] Software Engineering Institute, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, 1995.

[23] Software Engineering Institute, Capability maturity model integration, v1.1., www.sei.cmu.edu/cmmi/cmmi.html, 2001.

[24] Sourceforge, http://sourceforge.net/, last visited 8-Mar-2004.

[25] R. Stallmann, “The GNU Operating System and the Free Software Movement,” in C. DiBona, S. Ockmann and M. Stone, *Open Sources: Voices of the Open Source Revolution*, O’Reilly, Sebastopol, 1999, pp. 53-70.

[26] Strauss, M., Pape, B., Adam, F., Klein, M. and Reinecke, L., *Commsy-Evaluationsbericht 2003: Softwareunter-unterstützung für selbstständiges und kooperatives Lernen*, FBI-HH-B-251/03, 2003.

[27] M. Strauss and B. Pape, „Eine methodische Expedition zur formativen Evaluation kooperativer Lernplattformen,“ in B. Pape, D. Krause, and H. Oberquelle, editors, *Wissensprojekte. Gemeinschaftliches Lernen aus didaktischer, softwaretechnischer und organisatorischer Sicht*, no. 27 in Medien in der Wissenschaft, Waxman, Münster, New York, München, Berlin, 2004, pp. 373–388.

[28] P. Vixie, “Software Engineering,” in C. DiBona, S. Ockmann and M. Stone, *Open Sources: Voices of the Open Source Revolution*, O’Reilly, Sebastopol, 1999, pp. 91-100.

[29] V. Wulf and M. Rohde, “Towards an integrated organisation and technology development,” in *Proceedings of the Symposium on Designing Interactive Systems*, pp. 55–64, 1995.

Proceedings of the First International Conference on Open Source Systems
Genova, 11th-15th July 2005
Marco Scotto and Giancarlo Succi (Eds.), pp. 37-43