Conceptual Modelling as a New Entry in the Bazaar: The Open Model Approach

Stefan Koch[1], Stefan Strecker[2], and Ulrich Frank[2]

[1] Institute for Information Business, Vienna University of Economics and BA
stefan.koch@wu-wien.ac.at
[2] Information Systems and Enterprise Modelling, University Duisburg-Essen
{stefan.strecker|ulrich.frank}@uni-due.de

Abstract. The present contribution proposes to transfer the main principles of open source software development to a new context: conceptual modelling; an activity closely related to software development. The goal of the proposed "open model" approach is to collaboratively develop reference models for everyone to copy, use and refine in a public process. We briefly introduce conceptual modelling and reference models, discuss the cornerstones of an open modelling process, and propose a procedure for initiating, growing and sustaining an open model project. The paper concludes with a discussion of potential benefits and pitfalls.

1 Introduction

Open source software development [5] is currently the prime example for collaborative development processes by geographically dispersed participants. Similar joint efforts have emerged in collaborative writing and publishing (i.e. open content [23]), and in other areas [32] such as open hardware, and open education [16]. Recent research on open source projects has identified fundamental principles common to many collaborative development processes [30], e.g. the named credit and anti-forking norm [35], which seem to carry over to collaborative processes with outcomes other than source code. However, further research is still required to determine possible boundaries for this, and the necessary preconditions that have to be met in an area to make this transfer successful.

The present contribution proposes to apply the main principles behind open source software development to *conceptual modelling*, an activity closely related to software development [9]. The goal of the proposed "open model" approach is to develop *reference models* for everyone to copy, distribute, use, and refine with the collaboration of a large number of participants in a public process. Its consequential objective is to encourage the development of software based on these models as well as the models' use for research and teaching purposes.

Transferring the principles of open source software development to conceptual modelling is of interest for both practical and scientific reasons. The use of tried and tested reference models promises several advantages over "reinventing the wheel"-approaches, e.g. (i) reduced time and effort in software design, (ii)

use of the knowledge of domain experts, and (iii) facilitation of integration and reuse (cf. Sec. 2). From a research point of view, an open model approach provides an opportunity to research whether and how the principles of open source software development processes carry over to other contexts in general [32] and to modelling in particular. Starting from the observation that the absence of modelling activities in open source software development has been recognized as problematic, e.g. [38, 24, 42], an open model approach also serves as a testbed for investigating the effects of conceptual modelling and open models on open source software development.

An ideological argument refers to the freedom of models: If it is accepted that information needs to be freely accessible [33, 23], this should also pertain to the models behind any software, even more so than the software's documentation, given that the models are of much higher importance. For example, problems with a large code base becoming effectively closed due to high complexity might be overcome at least to some degree when the underlying models are accessible. Even if SAP would release the source code of R/3, or Microsoft the code of Windows or Office, these large software systems would be difficult to understand without the underlying models. Releasing the appropriate models would be of even greater importance than the release of source code. Given a free and open model, alternative implementations of the same functionality will be easier to produce. Other examples are the Netscape/Mozilla or OpenOffice projects, which experienced difficulties in setting up a community.

In this paper, we briefly introduce conceptual modelling and reference models (Sec. 2), discuss the cornerstones of an open modelling process (Sec. 3.1), and propose a procedure for initiating, growing and sustaining an open model approach (Sec. 3.2). We will also discuss both benefits and pitfalls (Sec. 4), and conclude with a summary and future work (Sec. 5).

2 Prospects of conceptual modelling

2.1 Bridging the gap

On a conceptual level, models represent abstractions of real-world phenomena relevant to a certain modelling task (conceptual models) [9]. *Conceptual models* are aimed at providing representations of software systems that are accessible not only to modellers and software developers, but also to domain experts and prospective end users. For this reason, they focus on general concepts commonly used within a certain domain abstracting from technical aspects.

By allowing for various abstractions, e.g. data abstraction, object abstraction, and process abstraction, they contribute to the reduction of complexity and risk. On the other hand, they take into account certain characteristics of implementation-level languages. Thus, conceptual models help to overcome the notorious cultural chasm between developers and end users [20]. At the same time, they support the communication among software developers, thus contributing to more efficient coordination in software development projects.

Furthermore, conceptual models are the instrument of choice to prepare for integrating applications by defining common concepts for a set of applications. Also, abstracting from technical details renders conceptual models better suited for reuse than source code.

2.2 Reference models as silver bullets

The design of high quality conceptual models suited to guide the development of large systems is a challenging task that requires outstanding expertise as well as a thorough and costly analysis. This motivates the development of *reference models*. A reference model is a conceptual model that comes with the claim to suit not just one system, but a whole range of systems, e.g. a generic process model for contract processing in the insurance industry. The claim pertains to two aspects. On the one hand, reference models are intended to provide appropriate generalisations of existing domains. On the other hand, reference models are aimed at delivering blueprints for good system design. Thus, reference models are descriptive and prescriptive at the same time. Reference models are a reification of a very attractive vision: They promise higher quality of information systems at less cost. However, adapting reference models for actual system implementation often requires significant adaptations for a specific application.

The development of reference models currently takes place mainly in academia and in large software companies. Reference models distributed as part of commercial packages, e.g. Enterprise Resource Planning (ERP) software such as SAP R/3, have been adopted in practice. Their development process is typically a closed-shop effort on part of a software or consulting company, e.g. SAP, with the respective copyright and patent issues attached.

Academic research has produced several modelling languages and associated reference models in recent years, e.g. [31, 10]. Conceptual models in general and reference models in particular have been a focus in information systems (IS) research [41]. Research on reference models and modelling languages is commonly subsumed in the field of enterprise modelling [4, 2].

With regard to the tremendous benefits to be expected from high quality reference models, it seems surprising that there is only a small number of reference models available [6]—despite the remarkable amount of work on reference models in academia. However, these models usually suffer from two deficiencies. Firstly, they remain in a prototypical state—due to limited resources available in single research projects. Secondly, they fail to be deployed in practice. While the second shortcoming can in part be contributed to the first one, it is also caused by the lack of effective mechanisms to disseminate research results.

A recent survey on internet-based reference modelling [39] has shown that only very little information on reference models is available on-line and that most models are either published in part or entirely in print publications if at all. The study implies that discussion about and construction of reference models hardly ever is an open process and concludes that the internet offers potential for further distributed, collaborative efforts to develop reference models.

Reference models seem to be an ideal subject for an open, community-driven development process. The modelling process necessitates a higher level of abstraction than programming. Its overall complexity allows for the involvement of a diversity of participants ranging from developers to users to domain experts and reviewers, among others. Following Raymond [30], a larger number and a greater diversity of eyeballs on a modelling task is required to conceive high quality conceptual models. Note, however, the differences between conceptual models and source code. It is likely that the number of eyeballs on models will be less than those on code if only due to the fact that evaluating a reference model to suggest improvements requires different skills and interests. The transparency of a conceptual model fosters the coordination of the various contributions. An open model project would not only allow for bundling academic resources. Rather, it could serve as a common medium for organizing the exchange between academia and practice, thus fostering its acceptance and deployment. With respect to the division of labour, a reference model could be used as a common reference in various disciplines. On a higher level of abstraction, for instance, business experts could analyse and eventually redesign business processes, while software experts could focus on the design of supporting information systems. Hence, reference models could support cross-disciplinary cooperation and contribute to the coherent integration of state-of-the-art knowledge from multiple disciplines.

3 Conceptual modelling as an open process

3.1 Cornerstones of the open model process

In the following, we assume that it is possible to initiate, grow and sustain collaborative processes with outcomes other than source code based on the fundamental principles behind open source software development. Distributed modelling processes are a particular instance of such collaborative processes, in particular, reference modelling processes in which stakeholders in the process collaborate to develop reference models. Therefore the following cornerstones of open source development need to be adopted to the open model approach:

Appropriate licence. An appropriate model licence is required to ensure that everyone is allowed to copy, distribute, use and modify the model (open model) [33, 29]. The licence should explicitly allow for the model's use in proprietary software development to promote its adoption and deployment in practice, while aiming for widest possible range of participants [34].

Roles and stakeholders. The open modelling process should be designed to facilitate contributions from practitioners (e.g. domain experts, business analysts) and academics (e.g. researchers, students) alike. The role of practitioners is twofold: While they can and should participate in the modelling task itself, they serve as the most important form of quality assurance and review. Most

often, they will be in the best position to judge the relevance and correctness of business processes modelled against business requirements and practice. Based on common elements in open source team structures, we identify the following roles in an open modelling process:

- Maintainer: The maintainer is responsible for either the whole model or a distinct sub-model. Whether several maintainers are introduced, or become necessary, depends both on the size of the domain, and the success of the initiative. Depending on the organisational model chosen, this can be either an owner/maintainer, benevolent dictator, or trusted lieutenant [30], deciding on whether a submission is accepted, when a new official version is released etc., or, if a democratic structure is adopted, mostly an administrative position. These positions will be filled by people who have demonstrated long-term and high quality commitment, so that their authority is accepted by the others.
- Modeller: The position of a modeller is analogous to the commiter in open source software development, in that he has the right to perform changes to the model. The right to do this directly is normally linked to several prior submission that have successfully passed quality control.
- Contributor: Any person can fill the role of contributor, and propose changes to the model. These need to be passed over to a modeller or maintainer, in order to pass quality control and be accepted. If this is done several times, a contributor might advance to modeller position.
- Reviewers: As in software development, quality assurance is an important task in an open model project. Open source projects employ several mechanisms to this end [45], with extensive peer review as the most prominent example. In an open model project, an official position of reviewer might be established. Naturally, everyone filling up another role might become reviewer, e.g. any modeller could automatically be assigned this additional role. The most important task is to review any proposed changes to the model, and to decide according to relevance and quality. Practitioners are very much suited for this role in order to provide feedback from their experience.
- End users: Anybody can become an end user of an open model. Of special interest are those who become active participants, by either reporting problems or suggesting ideas, or by submitting changes to the model directly.

As empirical research on open source software development teams has shown, in most projects a small inner group forms [25, 19], surrounded by a larger number of contributors, and an even greater number of participants not directly involved in programming, but other tasks like bug reporting. A similar structure might appear in an open model project. It should also be noted that both structure and processes in open source software projects have been found to change over time in accordance with the needs and the evolution of the product, which in turn is of course shaped by the community [43]. In an open model initiative, both team organisation and processes should, therefore, be flexible enough to be adapted to changing needs should they arise.

Motivation and incentives. A key success factor pertains to establishing convincing incentives for participation in order to attract participants and to reach a critical mass of contributors. The question of motivation has been extensively researched in the area of open source software development [21, 12, 14, 15] showing that several different possible motivational factors both intrinsic and extrinsic are relevant. For an open community to work effectively, it is necessary to establish convincing incentives for all participants.

A key incentive to support open source projects originates from the joy of programming and the rewarding experience of creating an artefact that works and is recognized by peers. Conceptual models will usually not be executable, but peer-recognition as reputation mechanism still applies. In fact, most motivational factors are likely to carry over to open models, with the exception of those directly related to coding. On the other hand, people might also gain intrinsic motivation from modelling, though a common perception is that programmers do not like this activity. It remains to be seen whether and how developers perceive the value of open models and the participation in open modelling processes. Nevertheless, the development of models can be very appealing: It is a challenging task, hence, offering reputation for those who submit sustainable contributions. Also, as a blueprint for multiple systems, an open model is rewarding its designers with the practical relevance of their work. However, it is not sufficient to rely on these incentives only. There is need for additional incentives for all groups involved in the development of a reference model.

A researcher's contributions to a reference model could be acknowledged as a substantial academic achievement—similar to a publication. In order to evaluate such a contribution adequately, some sort of a review process would then be required, for example an adapted version of the democratic votes as used in the Apache project [7]. Incentives for practitioners seem hard to establish at first. However, the demand for system architectures and other forms of blueprints from practitioners points to their recognition of the value of reference models. It would also be possible for participants to pursue related business models, for example by providing related services like consulting or implementations.

There are also several explanations for the viability and stability of open source software development, including a reputation-based gift culture [30, 44], a craftsman-model with programming as an immanent good [30, 36] or economic models [22] like the cooking-pot market [11], as an inverse tragedy of the commons [30] or as user innovation networks [40]. Again, all of these might be used to argue the stability of an open model initiative.

Parallelisation of work. Maybe the most important characteristic of open source software development is the strong parallelisation of work, especially software testing, using a large number of participants (*"Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone."* [30]). In order to reduce duplicate work, to ensure motivation and to keep the participants' interest, fast release cycles (*"Release eary, release often"* [30]) are necessary. For an open model initiative, this point

is also of relevance. As modelling involves creativity and a higher level of abstraction than programming, innovative contributions are even more required. The main question is whether the parallelisation of work is possible. To ensure this, the following preconditions need to be met: (i) appropriate tools for this cooperation, i.e. a model versioning system as described below, (ii) a modelling language supporting appropriate modularity as described below, and (iii) a modelling task extensive enough to bring several people to bear, which is why especially reference modelling is put to the center of this proposal.

Modularity. Achieving a modular design is seen as an important precondition to be able to parallelise large amounts of work on an artefact [26, 28, 8, 1]. Otherwise, costs for coordination and communication would grow exponentially and would negate benefits from higher headcount. Also in open modelling, this precondition is likely to exist. Therefore an appropriate modelling language is necessary that allows for modularity, especially on several levels of abstraction.

Collaboration tools. As most participants in open source software development teams are distributed around the globe without personal contact, communication and collaboration are achieved by appropriate tools, especially mailing lists, source code versioning systems, bug reporting and management and others. This also constitutes a precondition for the parallelisation of work. For an open model approach, comparable tools are needed. While for most communication needs the same tools like mailing lists can be employed, a substitute for source code versioning systems like CVS [8] or SVN might be needed. Although many models can be reduced to a text-based representation, for example using appropriate XML-schemas, models are by nature more visually oriented. Therefore a versioning system which explicitly supports visual inspection of models and especially changes to models would be important. We are not currently aware of a free product that fulfills these criteria, but such a tool should be implemented, probably in the context of a first such project.

3.2 Procedure for implementing an open model project

From having identified the cornerstones of an open model process as described above, several necessary decisions and steps can be derived for the implementation of such an initiative.

1. Choosing an appropriate licence: An appropriate licence should allow for several effects to take place. On the one hand, it should be as free and open as possible to ensure the highest possible number of participants [34], while avoiding ideological debates. On the other hand, using the model as a base for commercial implementations should not be impossible. Therefore, the licence would certainly need to conform to the Open Source Definition [29], while GPL-compatibility, i.e. being copyleft [33], might be problematic. Whether an existing licence from the field of documentation, e.g. creative commons, fulfills these prerequisites and could be adopted, or whether a new licence needs to be defined is still to be determined.

2. Choosing a suitable reference model domain: The domain of the reference model to be developed should also be chosen so as to attract a large number of participants, for whom the domain's problems are *"scratching an itch"* [30]. Also the scope should be large enough to allow for a sufficient number of people to work on the model.
3. Choosing appropriate abstractions: Models of business processes have shown to be a suitable abstraction for understanding a domain. They can be associated with further abstractions such as information models, e.g. object models or resource models. Therefore, it seems reasonable to focus on business process models as a common reference for all participants and as an instrument to integrate additional abstractions.
4. Choosing corresponding modelling languages and tools: Developing business process models, object models and other abstractions requires the selection of appropriate modelling languages. These decisions have to take into account the availability of corresponding tools, which are almost mandatory in order to cope with model complexity, to allow for automated syntax and integrity checks as well as for automated transformation into other representations such as implementation-level languages. The modelling languages themselves should support modularity and extensibility, e.g. to define business processes on several levels, which have been shown to be critical success factors in open source development [28]. Also, far spread knowledge in the chosen languages would increase the number of possible participants. In addition, storage and management of explanations, discussions and reasonings for the documented models and any change to them must be provided.
5. Design the appropriate processes: The necessary processes especially regarding decision making, i.e. new releases, conflict resolution [37] and the release management [17] should be designed. This also includes accounting for the participants' motivations by setting up appropriate incentive schemes.
6. Preparing the necessary infrastructure: As detailled above, the necessary infrastructure for coordination and communication needs to be set up. This includes standard tools like mailing lists or bug tracking, but especially versioning might need further enhancements to existing systems. A survey of reference models and reference modelling on the internet [39] has shown that the internet is hardly ever used to provide reference models. This reluctance is a problem, and will have to be overcome.
7. Delivering a plausible promise in form of a first prototype: To start the community building process, an initial set of open models needs to be released to the interested public. This prototype should give a plausible promise that an interesting initiative is starting, and that joining it would be wortwhile.
8. Continuously evaluating processes, products and community: During the lifetime of the initiative, all aspects will need to be monitored. This includes the processes and the community, where appropriate methods for analysing open source software projects e.g. regarding concentration measures or evolution could be adopted [13].

4 Discussion

From an academic point of view, reference models are appealing, because their claim for general validity makes them resemble scientific theories. Taken the complexity of some domains, reference models could serve as a medium to coordinate research in large teams. Thus, they could serve as object and objectivation of research in IS.

The evaluation of conceptual models is a challenging task - both with respect to quality assurance and from an epistemological perspective [9]. Due to their claim for excellence, this is even more the case for reference models. The concept of truth is only of limited use for evaluating them, since they are usually aimed at intended systems or future worlds. Hence, a discoursive evaluation is the only remaining option. This requires not only the participation of researchers, domain experts, prospective users, but also an open culture of critique and construction. An open model community could provide for that and hence contribute to a multi-perspective evaluation of reference models that is difficult to achieve as long as reference models are subject of single research projects only. Therefore, any model should be accompanied with reasonings about the model, changes to the model and discussions about these.

Reference models could also serve as a subject for teaching, e.g. in IS or Computer Science. Students could study and enhance reference models in order to get a differentiated, but still abstract imagination of application domains, of which a reference model provides the relevant concepts. Therefore, it could serve as a foundation for the development of application level standards ("business language") or enterprise level ontologies [3, 18]. A reference model represents the body of knowledge of the participating disciplines. It also includes best practices and therefore can be regarded as a blueprint for knowledge management as well.

Finally, open source software development itself might benefit from the establishment of open models. The absence of modelling activities has been a center of critique on open source software development, e.g. [38, 24, 42], and has been held responsible, among others, for insufficient documentation, lost possibilities for reuse or missing information for effort estimations. Therefore, open source software projects are prime candidates for experiencing positive effects of open model projects, and vice versa, as any open model project would benefit from one or more open implementations being pursued.

The main challenge for an open model initiative is to reach a critical mass of participants to start a sustainable open process. This will hinge mostly, besides the necessary infrastructure being in place to reduce transaction costs, on the motivation of potential participants. In this paper, we have discussed possible incentives for several groups, but if these fail in practice, the project might not get off the ground. While not the only factor, the question whether people can be found in large enough quantities for which modelling poses an interesting, challenging and therefore in itself rewarding activity remains to be seen.

5 Summary and future work

In this paper, we have proposed to adopt the principles of open source software development for the collaboration of geographically dispersed project participants and their joint efforts to another context: conceptual modelling. The goal of the proposed "open model" approach is to develop reference models for everyone to copy, use, refine and later implement with the collaboration of a large number of participants in a public process.

To this end, the cornerstones of open source development need to be adopted, and in some cases adapted. This led to a list of decisions and steps to be considered for implementing such an initiative. The important next step would be to verify the viability of the open model process in the light of a real-world example, i.e. preparing the set-up of such a project. Following [27], it seems prudent to create a technological infrastructure which facilitates exchange of ideas and models among interested parties, i.e. to make discussions and models available to the open source community and the public at large. Especially for the first project, initial funding for preparing the infrastructure, especially an open "model versioning system", and also for developing a prototype is required. Also, it is necessary to educate relevant groups of prospective participants. We intend to pursue the proposed approach and found an open model initiative. After all, we are convinced that such an initiative would yield substantial benefits, both in itself, and as an academic field study.

References

1. Terry Bollinger, Russel Nelson, Karsten M. Self, and Stephen J. Turnbull. Open-source methods: Peering through the clutter. *IEEE Software*, 16(4):8–11, July/August 1999.
2. Nikunj P. Dalal, Manjunath Kamath, William J. Kolarik, and Eswar Sivaraman. Toward an integrated framework for modeling enterprise processes. *Communications of the ACM*, 47(3):83–87, 2004.
3. Jos de Bruijn, Dieter Fensel, Uwe Keller, and Rubn Lara. Using the web service modeling ontology to enable semantic e-business. *Communications of the ACM*, 48(12):43–47, 2005.
4. Dursun Delen, Nikunj P. Dalal, and Perakath C. Benjamin. Integrated modeling: the key to holistic understanding of the enterprise. *Communications of the ACM*, 48(4):107–112, 2005.
5. Joseph Feller and Brian Fitzgerald. *Understanding Open Source Software Development*. Addison-Wesley, London, 2002.
6. Peter Fettke and Peter Loos. Systematische Erhebung von Referenzmodellen - Ergebnisse einer Voruntersuchung. Working Papers of the Research Group Information Systems & Management 19, University of Mainz, Mainz, Germany, 2004.
7. Roy T. Fielding. Shared leadership in the Apache project. *Communications of the ACM*, 42(4):42–43, April 1999.
8. Karl Fogel. *Open Source Development with CVS*. CoriolisOpen Press, 1999.

9. Ulrich Frank. Conceptual Modelling as the Core of the Information Systems Discipline — Perspectives and Epistemological Challenges. In *Proceedings of the Fifth America's Conference on Information Systems (AMCIS 99)*, pages 695–697, Milwaukee, 1999. Association for Information Systems (AIS).
10. Ulrich Frank. Multi-Perspective Enterprise Models as a Conceptual Foundation for Knowledge Management. In *Proceedings of the Thirty-Third Annual Hawaii International Conference on System Sciences*. IEEE CS Press, 2000.
11. Rishab Aiyer Ghosh. Cooking pot markets: an economic model for the trade in free goods and services on the Internet. *First Monday*, 3(3), March 1998.
12. Rishab Aiyer Ghosh. Understanding free software developers: Findings from the floss study. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, *Perspectives on Free and Open Source Software*, pages 23–46. MIT Press, 2005.
13. Michael Hahsler and Stefan Koch. Discussion of a large-scale open source data collection methodology. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-38)*, Big Island, Hawaii, 2005.
14. Alexander Hars and Shaosong Ou. Working for Free? Motivations for Participating in Open-Source Projects. *International Journal of Electronic Commerce*, 6(3):25–39, 2002.
15. Guido Hertel, Sven Niedner, and Stefanie Hermann. Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel. *Research Policy*, 32(7):1159–1177, 2003.
16. Kei Ishii and Bernd Lutterbeck. Unexploited resources of online education for democracy - why the future should belong to OpenCourseWare. *First Monday*, 6(11), November 2001.
17. Niels Jorgensen. Putting it all in the trunk: Incremental software engineering in the FreeBSD project. *Information Systems Journal*, 11(4):321–336, 2001.
18. Ejub Kajan and Leonid Stoimenov. Toward an ontology-driven architectural framework for b2b. *Communications of the ACM*, 48(12):60–66, 2005.
19. Stefan Koch. Profiling an open source project ecology and its programmers. *Electronic Markets*, 14(2):77–88, 2004.
20. Sari Kujala. User involvement: a review of the benefits and challenges. *Behaviour & Information Technology*, 22(1):1–16, January–February 2003.
21. Karim R. Lakhani and Robert G. Wolf. Why hackers do what they do: Understanding motivation and effort in free/open source software projects. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, *Perspectives on Free and Open Source Software*, pages 3–22. MIT Press, 2005.
22. Josh Lerner and Jean Tirole. Economic perspectives on open source. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, *Perspectives on Free and Open Source Software*, pages 47–78. MIT Press, 2005.
23. Lawrence Lessig. *The Future of Ideas: The Fate of the Commons in a Connected World*. Random House, New York, 2001.
24. Steve McConnell. Open-source methodology: Ready for prime time? *IEEE Software*, 16(4):6–8, July/August 1999.
25. Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of Open Source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, 2002.
26. Alessandro Narduzzo and Alessandro Rossi. The role of modularity in free/open source software development. In Stefan Koch, editor, *Free/Open Source Software Development*, pages 84–102. Idea Group Publishing, 2004.

27. David M. Nichols and Michael B. Twidale. The Usability of Open Source software. *First Monday*, 8(1), January 2003.
28. Tim O'Reilly. Lessons from open-source software development. *Communications of the ACM*, 42(4):32–73, April 1999.
29. Bruce Perens. The open source definition. In Chris DiBona, Sam Ockman, and Mark Stone, editors, *Open Sources: Voices from the Open Source Revolution*. O'Reilly and Associates, 1999.
30. Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates, 1999.
31. August-Wilhelm Scheer. *Business Process Engineering: Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, Germany, 2nd edition, 1994.
32. Clay Shirky. Open source outside the domain of software. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, *Perspectives on Free and Open Source Software*, pages 483–488. MIT Press, 2005.
33. Richard M. Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. GNU Press, Boston, Massachusetts, 2002.
34. Katherine J. Stewart, Tony Ammeter, and Likoebe Maruping. A preliminary analysis of the influences of licensing and organizational sponsorship on success in open source projects. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-38)*, Big Island, Hawaii, 2005.
35. Katherine J. Stewart and Sanjay Gosain. The Impact of Ideology on Effectiveness in Open Source Software Development Teams. Working paper, Department of Decision and Information Technologies, University of Maryland, 2005. Forthcoming in MIS Quarterly. `http://www.smith.umd.edu/faculty/kstewart/ResearchInfo/KJSResearch.htm`.
36. Linus Torvalds. FM interview with Linus Torvalds: What motivates free software developers? *First Monday*, 3(3), March 1998.
37. Ruben van Wendel de Joode. Managing conflicts in open source communities. *Electronic Markets*, 14(2):104–113, 2004.
38. Paul Vixie. Software engineering. In Chris DiBona, Sam Ockman, and Mark Stone, editors, *Open Sources: Voices from the Open Source Revolution*. O'Reilly and Associates, 1999.
39. Jan vom Brocke. Internetbasierte Referenzmodellierung – State-of-the-Art und Entwicklungsperspektiven. *Wirtschaftsinformatik*, 46(5):390–404, 2004.
40. Eric von Hippel. Open source software projects as user innovation networks. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, *Perspectives on Free and Open Source Software*, pages 267–278. MIT Press, 2005.
41. Ron Weber. *Ontological Foundations of Information Systems*. Coopers & Lybrand, Melbourne, 1997.
42. Greg Wilson. Is the open-source community setting a bad example? *IEEE Software*, 16(1):23–25, January/February 1999.
43. Yunwen Ye, Kumiyo Nakakoji, Yasuhiro Yamamoto, and Kouichi Kishida. The co-evolution of systems and communities in free and open source software development. In Stefan Koch, editor, *Free/Open Source Software Development*, pages 59–82. Idea Group Publishing, 2004.
44. David Zeitlyn. Gift economies in the development of open source software: anthropological reflections. *Research Policy*, 32(7):1287–1291, 2003.
45. Luyin Zhao and Sebastian Elbaum. Quality assurance under the open source development model. *The Journal of Systems and Software*, 66:65–75, 2003.