Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives

Kirk St.Amant
Texas Tech University, USA

Brian Still
Texas Tech University, USA

Information Science
REFERENCE

INFORMATION SCIENCE REFERENCE

Hershey • New York

Acquisitions Editor: Kristin Klinger
Development Editor: Kristin Roth
Senior Managing Editor: Jennifer Neidig
Managing Editor: Sara Reed
Assistant Managing Editor: Diane Huskinson
Copy Editor: Nicole Dean, Shanelle Ramelb, Ann Shaver, and Sue VanderHook
Typesetter: Diane Huskinson
Cover Design: Lisa Tosheff
Printed at: Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of Idea Group Inc.)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@idea-group.com
Web site: <http://www.info-sci-ref.com>

and in the United Kingdom by
Information Science Reference (an imprint of Idea Group Inc.)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: <http://www.eurospanonline.com>

Copyright © 2007 by IGI Global. All rights reserved. No part of this publication may be reproduced, stored or distributed in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher.

Product or company names used in this set are for identification purposes only. Inclusion of the names of the products or companies does not indicate a claim of ownership by IGI Global of the trademark or registered trademark.

Library of Congress Cataloging-in-Publication Data

Handbook of research on open source software : technological, economic and social perspectives / Kirk St. Amant and Brian Still, editors.
p. cm.

Summary: "This book examines how use of open source software (OSS) is affecting society, business, government, education, and law, including an overview of the culture from which OSS emerged and the process through which OSS is created and modified. Readers will gain an understanding of the complexities and the nuances related to OSS adoption and the range of its applications"--Provided by publisher.

Includes bibliographical references and index.

ISBN 978-1-59140-999-1 (hardcover) -- ISBN 978-1-59140-892-5 (ebook)

1. Shareware (Computer software)--Handbooks, manuals, etc. 2. Open source software--Handbooks, manuals, etc. I. St. Amant, Kirk, 1970- II. Still, Brian, 1968-

QA76.76.S46H35 2007

005.3--dc22

2006039844

British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this book set is new, previously-unpublished material. The views expressed in this book are those of the authors, but not necessarily of the publisher.

Chapter XIV

The Social Order of Open Source Software Production

Jochen Gläser

Australian National University, Australia

ABSTRACT

This chapter contributes to the sociological understanding of open source software (OSS) production by identifying the social mechanism that creates social order in OSS communities. OSS communities are identified as production communities whose mode of production employs autonomous decentralized decision making on contributions and autonomous production of contributions while maintaining the necessary order by adjustment to the common subject matter of work. Thus, OSS communities belong to the same type of collective production system as scientific communities. Both consist of members who not only work on a common product, but are also aware of this collective work and adjust their actions accordingly. Membership is based on the self-perception of working with the community's subject matter (software or respectively scientific knowledge). The major differences between the two are due to the different subject matters of work. Production communities are compared to the previously known collective production systems, namely, markets, organizations, and networks. They have a competitive advantage in the production under complete uncertainty, that is, when neither the nature of a problem, nor the way in which it can be solved, nor the skills required for its solution are known in advance.

INTRODUCTION

This chapter contributes to the sociological understanding of open source software (OSS) production by identifying the social mechanism that creates social order in OSS communities. The concept of social order is used here in its most basic sense as describing a situation in which actors have adjusted their actions to each other. This order is indeed very high in OSS communities, who produce large and highly complex

software from many independent contributions. It is even astonishingly high when we take into account how few of the most common tools for creating order—rules, commands, and negotiations—are used. Therefore, most analysts agree that OSS is produced in a distinct “new” mode that is qualitatively different from the “corporate way” of software production.

However, none of the four strands of literature on OSS production has produced a consistent explanation of the way in which this amazing order is

achieved. The *participant-observer* literature has proposed metaphors that emphasize the decentralized, democratic, open, and communal nature of OSS, notably the “cooking pot market” (Ghosh, 1998) and the “bazaar” (Raymond, 1999). These metaphors, while suggestive, are not grounded in social theory. *Economics* is still fascinated by the voluntary contributions to a public good, and has consequently focused on motivations to contribute to OSS (Dalle & Jullien, 2003; Lerner & Tirole, 2002; von Hippel & von Krogh, 2003). However, neither these investigations nor the generalized questions about transaction costs (Demil & Lecocq, 2003) or about the allocation of efforts to modules (Dalle, David, Ghosh, & Steinmueller, 2004) capture the specific ways in which an ill-defined group of people manages to produce a complex good. These ways have been looked at primarily in the context of *management and software engineering analyses*, which produced interesting case studies of the coordination of individual OSS projects such as Linux, Apache, Perl, Sendmail, Mozilla, and others (Holck & Jørgensen, 2005; Iannacci, 2003; Jørgensen, 2001; Koch & Schneider, 2002; Lanzara & Morner, 2003; Mockus, Fielding, & Herbsleb, 2002). Some analysts tried to compare OSS communities to “traditional organizations” (Sharma, Sugumeran, & Rajagopalan, 2002) or to catch the specific mode of OSS production with generalized concepts such as “virtual organization” (Gallivan, 2001) or “distributed collective practice” (Gasser & Ripoche, 2003). However, these concepts are similar to the metaphors in the observer-participant literature in that they are ad hoc generalizations that are not embedded in theories of social order or of collective production. Finally, sociological analyses have contributed the idea of a *gift economy* (Bergquist & Ljungberg, 2001; Zeitlyn, 2003), various concepts of community (Edwards, 2001), social movements (Hess, 2005; Holtgrewe & Werle, 2001), the hacker culture (Lin, in this volume), and applications of actor-network theory (Tuomi, 2001). These sociological accounts focus on the specificity of social relations in OSS

communities and more or less entirely disregard the specific mode of production employed by these communities.

Missing from the numerous case studies on OSS production is a description of the social mechanisms that create order by enabling the adjustment of actions. Following Mayntz (2004, p. 241), we define a social mechanism as *a sequence of causally linked events that occur repeatedly in reality if certain conditions are given and link specified initial conditions to a specific outcome* (for a similar definition, see Hedström, 2005, p. 11). Heroically simplifying, we can think of social mechanisms as *subroutines of the social* that are activated under certain conditions and produce specific results. Only by describing the social mechanism at work can we explain how a specific outcome is produced under certain conditions (Hedström). In order to explain how a well-ordered collective production of OSS is achieved under conditions of shifting membership, incomplete information, and autonomous decision making by contributors, we need to find the social mechanisms that create order under these conditions.

Theoretical analyses of this kind are still rare. Only Benkler's (2002) proposal to regard OSS production as an instance of *commons-based peer production* comes close to describing a social mechanism of OSS production. According to Benkler, commons-based peer production “relies on decentralized information gathering and exchange to reduce the uncertainty of participants,” and “depends on very large aggregations of individuals independently scouring their information environment in search of opportunities to be creative in small or large increments. These individuals then self-identify for tasks and perform them for a variety of motivational reasons” (pp. 375-376).

The focus on information processes contributes an important insight in the process of OSS production. However, Benkler (2002) applies an extremely diffuse notion of production, which makes him subsume every personal communica-

tion, every electronic mailing list, and every online computer game to his model of commons-based peer production. Consequently, he is not able to describe the specific way in which the individual contributions are integrated into a common product. The distinctiveness of OSS production gets lost in his very general model.

Thus, while some important elements of the social mechanism that leads to OSS have been identified, we still do not have a consistent theoretical model of OSS production. Sociology can provide such a model if it supplements its analyses of social relations and cultures of OSS communities with an analysis of the social order of collective production. In this chapter, I provide a description of the distinct mechanism of collective production underlying OSS and compare it to the known mechanisms of markets, organizations, and networks.

Analyzing the social order of a collective production system requires delineating the producing collective and establishing how the actions of members are adjusted to each other. I will do this for OSS production by establishing how tasks for members of the producing collective emerge, how requirements of later integration affect the conduct of tasks, and how the integration of individual contributions into a common product is achieved. Thus, I will aim not at providing a complete description of OSS production, but rather at identifying the select social phenomena that enable the adjustment of actions in a dispersed collective whose members are only incompletely informed about each other but nevertheless manage to jointly produce a complex good. The “thin description” of OSS production that is provided in the following section enables the identification of features that make it a distinct type of collective production.¹ The description is based on published studies of OSS production from which I extracted the important elements of the social mechanism. I then use this description for two comparisons. First, I compare OSS production to its archetype, namely, the mode of production

employed by scientific communities, and introduce the concept *production community* for this mode of production. In the subsequent section, I compare production communities to markets, organizations, and networks, and tentatively discuss the specific efficiency of each mode. As a conclusion, I argue that OSS production is both a role model for much creative work of the future and a promising subject matter for studying production communities.

BACKGROUND

How Do Open Source Software Communities Produce?

The Emergence of Tasks

A crucial problem of all production processes is the definition of tasks for members of the producing collective. Each of these tasks must describe *utilizable contributions*, that is, contributions that are useful additions to the common product, can later be integrated into this product, and can be produced by the specific member of the collective. How is this achieved in OSS production?

One of the characteristic features of OSS production is that individual producers define their own tasks. Nobody is forced to produce a specific contribution. Instead, an individual producer perceives that something needs to be done, believes he or she is able to do it, and defines it as a task to do personally. Thus, individuals decide for themselves to produce the contributions they think are necessary, and offer them to the community. Necessary tasks are also publicly announced in a variety of ways. However, no mechanism exists that could force community members to solve one of these tasks (Bonaccorsi & Rossi, 2003; Mockus et al., 2002; Sharma et al., 2002). The creation of tasks for individual producers is essentially a decentralized, local activity by autonomous contributors. This has

been expressed by characterizing OSS production as a “distributed collective practice” (Gasser & Ripoché, 2003).

The decentralized autonomous decision making about tasks makes it likely that tasks match the abilities of producers. However, the autonomously defined contributions must also fit together and be integrated in a common product, which requires a mutual adjustment of individual actions. Since direct mutual adjustment between producers requires extensive information about participants, their current actions, and their plans, it is obviously impossible in OSS communities. OSS communities solve this problem by *mediated adjustment*, that is, by all producers adjusting to the common subject matter of work, which they observe and from which they derive their tasks. The common subject matter of work is a complex body of knowledge; at its core we find the current version of the source code, which often has a modular design in order to enable independent parallel work by many developers (Lanzara & Morner, 2003; Lerner & Tirole, 2002). This core is surrounded by a variety of corollary information that refers to the source code, such as bug reports or, more generally, “pending work tasks” (Holck & Jørgensen, 2005, pp. 7-8), documentation, other documents describing the software,² and discussions in mailing lists. Members of the producing collective draw on this knowledge when they formulate tasks for themselves. Based on these observations and their tests of the program, they perceive that additional code, a change in the existing code, or corollary work (e.g., on documentation) is needed for the software to function as planned.

Thus, the shared subject matter of work mediates the adjustment of producers’ actions by providing them with a common point of reference. This effect is implicitly described in numerous case studies, and has been explicitly discussed for the code (de Souza, Froehlich, & Dourish, 2005; Lanzara & Morner, 2003), and for the explicit and implicit descriptions of software requirements in

Web-based descriptions of the software (Scacchi, 2002). Since the subject matter of work orders the decentralized autonomous definition of tasks, OSS projects need to start with a significant initial submission of code, as has been observed by Raymond (1999) and by Lerner and Tirole (2002, p. 220): “The initial leader must also assemble a critical mass of code to which the programming community can react. Enough work must be done to show that the project is doable and has merit.” West and O’Mahony (2005) consider the amount of structured code initially available as a major advantage of the “spinout model,” where previously internally developed software is released under an open source software license, inviting the community to join the project.

Conduct of Work

While the production of code is the core task of OSS production, it is accompanied by a variety of other activities that are necessary for the production to succeed. Existing code must be tested and evaluated, which leads to information about bugs and suggestions for fixing them (new or changed code). Even the mere report of problems with the software is an important contribution. A documentation of the software must be produced, and users must be advised.

The conduct of these tasks is ordered in basically the same way as their emergence. Members of the producing collective are guided by the subject matter of work, that is, by the existing code and corollary information. The structure of the code that is used in the conduct of work poses highly specific requirements (de Souza et al., 2005; Mockus et al., 2002). Apart from these requirements inherent to the code, standards and general rules of good programming govern the conduct of work. Core developers of OSS projects setup guidelines, which are sometimes formalized (put in writing) and distributed (Bonaccorsi & Rossi, 2003). The fit of contributions is ensured by standard protocols, standardized interfaces,

guidelines for problem reports, and so forth (Bonaccorsi & Rossi; Iannacci, 2003; Jørgensen, 2001; Lanzara & Morner, 2003).

Integration of Contributions

The decentralized task definition leads to a significant redundancy. Many producers carry out the same pending task simultaneously, and thus submit reports of or solutions to the same problem (Bonaccorsi & Rossi, 2003; Holck & Jørgensen, 2005). It is important to notice that this is a redundancy only insofar as many offered contributions are solutions to the same problem. The solutions themselves differ from each other, and the best solution can be chosen from what is offered.

The integration of contributions is subject to an explicit decision process. Proposed code is published by submitting it to a mailing list, testing and evaluation by peers (other developers of code), and thereafter having it submitted to the software by one of the maintainers. The testing continues after submission because the members of the community who download the code and use it locally are testing it at the same time.

The quality of contributions to the common product is thus maintained by two different mechanisms. The first mechanism is a peer review procedure that is explicitly directed at establishing the quality of a proposed contribution. In the peer review process, a few authorized peers (some of the core developers) analyze the code, judge it explicitly, and decide on its integration into the software. The second mechanism is an implicit one, namely, quality control by use in a potentially infinite variety of different settings. It is possible because of the partial overlap of and close contact between the producer and user collectives. The common product is downloaded, used, and tested at every stage of its development by a potentially large audience. Since the local conditions (hardware and software environments) vary, the software is submitted to tests under a variety of conditions. The people who use the code,

encounter problems, and report these problems contribute to the production process.

Whenever the maintainers are of the opinion that a sufficiently advanced, comprehensive, and error-free code has been produced, they decide to release the code by assigning a version number to it and treating it as a product. These official releases are addressed to a wider audience of users and thus create a distinct user community that might not take part in any production activities.

Membership, Roles, and Social Structure

Students of OSS communities usually deal implicitly with the question of membership by applying a specific empirical method for identifying members. Depending on the method applied, a member is someone who has been identified as such by having an ID at source forge, being subscribed to a mailing list, having posted a message or a bug report, or having contributed code. For a theoretical model of OSS production, however, we need a theoretical answer to the membership question. What social phenomenon establishes membership in an OSS community? The obvious answer to that question is that a member is someone who participates in the collective production. However, this answer merely transforms the question. What constitutes participation in the production process?

While a variety of actions can be considered as participation in OSS production, none of them provides the opportunity of a clear-cut delineation that is both theoretically and empirically satisfying. Thus, while contributing code obviously constitutes participation in the production process, much of the offered code is not used, and the corollary work (testing, documentation, and other contributions) must be taken into account. Furthermore, some users report bugs but do not fix them, test the software but do not find bugs, or discuss the software but do not offer code. Some

of these activities are invisible to both the OSS community and the sociological observer.

From a theoretical point of view, the weakest form of participation is the conscious test of the software in its current version, which can be either an official release or a version in between releases. A conscious test means using the software while being aware of the existence of a community *out there* to whom problems can be communicated and from which advice could be received. While this description does not apply to all users of OSS, it includes the large number of them who never turn into active contributors of any sort because they never encounter problems with the software. In the communal production of OSS, successful tests of the software are rarely reported, the only systematic exception being the classification of reported bugs as invalid or “works for me” (Holck & Jørgensen, 2005, p. 11). Conducting tests, however, is a contribution to the communal production regardless of their outcome.

Thus, membership in OSS production communities is constituted by perception-based self-selection. One is a member of an OSS community if one perceives oneself as contributing, and the least possible contribution is a test of the software. This concept of membership is significantly wider than those that have been applied so far. It is consistent with empirical investigations of membership and social structure. Figure 1 provides an overview of the composition of an OSS community and the various roles in the production process. The major formalized distinction between work roles is that between the people who have permission to change the code and to release versions of the software (maintainers), and all other members of the community who may propose code but are not able to integrate it into the current version of the software themselves. OSS communities are basically meritocratic because membership to the core group of an OSS community depends on programming abilities as demonstrated in submitted code. However, the current activities of members need not coincide with the formalized

roles. Crowston, Wei, Li, and Howison (2006, p. 6) found that “the formal list of developers is not an accurate representation of contribution to the teams, at least as regards interactions around bug fixing” (see also de Souza et al., 2005, for transitions between positions in OSS communities).

The distribution of activities in OSS communities is highly skewed, with a very small proportion of members making most of the contributions. For example, a case study on the Apache project found that 3,060 people reported problems, 249 submitted new code, and 182 changed existing code (Mockus et al., 2002). This pattern repeats itself with regard to the lines of code submitted, number of problems reported, number of messages posted, and so forth (Crowston et al., 2006; Ghosh & Prakash, 2000; Koch & Schneider, 2002; Lerner & Tirole, 2002; von Krogh, Spaeth, & Lakhani, 2003). Furthermore, advanced projects appear to be surrounded by a larger user community whose members observe the production process, download and use the software, but become visible (turn into an active user) only if they choose to take part in discussions or report problems.

The perception-based membership is responsible for the fuzzy and fluid boundaries of OSS communities. In larger communities, no member knows all other members. One can easily become a member or fade out of a community. Simultaneous memberships in more than one OSS community are frequent (Robles, Scheider, Tretkowski, & Weber, 2001). Another important feature of OSS communities that is linked to the perception-based membership is the decoupling of subsistence and contributions. Even in the significant number of cases where producers are paid for participating in OSS production, payment is not received as a reward for a specific contribution (a problem report, code, etc.), but for the time that is devoted to the project. The payment is exogenous to the community, which creates its own nonmaterial reward (reputation). OSS communities are therefore based on a principal decoupling of offered and accepted contributions

Figure 1. Work roles and contributions in OSS communities (Source: Combination of information from Crowston et al., 2006, p. 1; Gacek & Arief, 2004, p. 36)

on the one hand, and the producers' subsistence on the other hand. This is in perfect agreement with the great variety of motives to contribute to OSS that has been empirically observed (Dalle & Jullien, 2003; Lerner & Tirole, 2002; von Hippel & von Krogh, 2003). The functioning of production communities is not affected by the specific motives of its members as long as enough people choose to contribute.

MAIN FOCUS OF THE CHAPTER

Production Communities

OSS is produced by collectivities whose members are incompletely informed about each other,

decide autonomously about what task they are going to solve in which way, and propose contributions that are rarely ever used. Social order in the sense of adjusted individual actions emerges primarily because people adjust their actions to the community's common subject matter of work, that is, to the shared code and to the knowledge about that code. I have proposed the concept of *production community* for these identity-based producing collectivities, and *communal production* for the mode of production that employs autonomous decentralized decision making on contributions and the autonomous production of contributions while maintaining the necessary order by adjustment to the common subject matter of work (Gläser, 2001, 2006). The concept of a production community challenges traditional

community theory, which assigns communities features such as multiplex relationships, shared values, frequent face-to-face interactions, and emotional bonds (Brint, 2001). However, recent studies on interest communities suggest that it is useful to make theoretical room for communities that are based on quite specific identities and do not necessarily feature shared lives and emotional bonds (Gläser, 2001, 2006). The numerous relatively small collectivities that produce OSS without much communication in open forums (Healy & Schussman, 2003) are groups rather than communities. Whether an OSS project is conducted by a group appears to be due to chance or the attractiveness of the proposed software. The differences between community and group OSS projects have not yet been explored.

OSS development indeed appears to be a unique mode of production. However, it is not. It has been repeatedly mentioned in the literature that OSS communities are similar to scientific communities (Benkler, 2002; Bezroukov, 1999; Dalle et al., 2004; Dalle & Jullien, 2003; Seiferth, 1999). However, these comparisons have been metaphorical rather than analytical because they lack suitable theoretical frameworks and exact descriptions of both modes of production. The following comparison of OSS production communities and scientific communities is based on a larger study that identified scientific communities as production communities (Gläser, 2006).

Both OSS and scientific communities produce new knowledge and consist of members who not only work on a common product, but are also aware of this collective work and adjust their actions accordingly. Membership is based on the self-perception of working with the community's subject matter (software, respectively scientific knowledge). That is why both communities also have in common the fuzziness of their boundaries and incomplete information about members. Another similarity is the decoupling of contributions from contributors' subsistence. Research has been professionalized for a long time. However,

the organizations that employ scientists and pay them salaries are not the social contexts in which tasks are defined and contributions are used. The production of scientific knowledge takes place in the scientific communities that do not provide material rewards for offered or accepted contributions. Thus, payments to scientists are exogenous to the production community, as are payments for OSS developers.

In both OSS production and research, task creation is a decentralized and autonomous process. Both communities are similar in that the elite may (and indeed do) define tasks of special importance or urgency, but has no means to force other members to work on these tasks. The adjustment of individual, dispersed, local activities is achieved by individuals' reference to the joint subject matter of work. In order to become such a reference point for an open and unknown audience, the common subject matter of work must be available to all potential members of a production community. Since membership is unknown, the only way to guarantee availability to all is publication, which is realized by scientific books and journals (and increasingly via the Internet) in scientific communities, and by Internet publication in OSS communities.³ The common subject matter also guides the conduct of work, which is additionally informed by standards and rules of conduct. In scientific communities, standards and rules for experimentation have functions similar to the rules of conduct for OSS production.

Both OSS production and the production of scientific knowledge apply the same mechanism of quality control, namely, peer review as a first, explicit preliminary check, and a subsequent implicit and possibly infinite series of checks in a variety of settings.

The important differences between scientific and OSS communities can be summarized by stating that structures and processes that remain implicit and informal in science are explicit and partly formalized in OSS communities. Thus, while both scientific and OSS communities have

been described as meritocracies, the elite of a scientific community do not have a formal status and no formal right to decide on the current *true* state of a community's knowledge. While scientific elites exert significant influence on the opinions of their communities, they cannot make binding decisions or establish binding rules of conduct. In OSS production, there are official versions of the common product, and only formally recognized elite members have both the right and the opportunity to change these versions.

A similar difference exists between the ways in which contributions are integrated into the common product, which are more explicit in OSS communities. Peer review of contributions is ubiquitous but relatively unimportant for the integration of new contributions into the knowledge of scientific communities. The main way of integrating new contributions is implicit integration by using each others' results. When members of a scientific community use a contribution in their further production of knowledge, they implicitly accept it and integrate it into the community's common body of knowledge, which is the basis of further knowledge production. In the case of OSS production, members of the elite must decide explicitly about the integration of contributions.

Thus, we observe an explicitly defined elite who makes decisions by adhering to explicit and partly formalized procedures and creates formalized rules and standards in OSS communities, while the analogous processes in scientific communities remain more informal and implicit. This difference is due to the fact that OSS is not only knowledge, but also a technical product. While scientists can proceed regardless of gaps, inconsistencies, and contradictions in their shared body of knowledge, a computer program's functioning would be endangered. Garzarelli and Galoppini (2003) have argued that in OSS production, hierarchy is necessary to manage the interaction of modules in large OSS projects. The analogous process in science—the interaction of knowledge

from different fields—is left to the self-organization of research. Moreover, the OSS production is intended to lead to a standardized mass product at some stage, namely, when a new version of the software is released. The overlap of the production and use of OSS, albeit considerable, is not complete, and an unproblematic output must be created for users who are not able to change the product. Therefore, consistency and reliability are much more important in OSS production than in science, where the only further use is by the producers themselves in the very same activity of knowledge production.

The Competitive Advantage of Production Communities

If the production community is a distinct type of collective production system, it must be both comparable to and qualitatively different from the three known systems of collective production. Table 1 applies the questions answered for production communities to the other three types.⁴

It becomes apparent that the collective production systems can be grouped according to a distinction introduced by Hayek (1945, 1991), who observed that social order can be either made or spontaneous. Organizations and networks are *made orders* because they both employ dedicated actions (coordination) to create order. The characteristic forms of coordination in organizations and networks are hierarchical decisions and negotiations, respectively. Markets and production communities both rely on decentralized task definition and ex post selection of contributions. They are *spontaneous orders* and rely on "parametric adjustment" (Lindblom, 1965) to a situation rather than coordination. However, production communities differ from markets in their use of concrete information about the production system's subject matter of work rather than extremely reduced abstract information as provided by market prices. Another significant difference that is not shown in Table 1 is that markets are

characterized by feedback between contributions and producers' subsistence, feedback that does not exist in production communities.

While this comparison does indeed prove that production communities can be introduced as a fourth distinct collective production system, it does not answer an important question. Which production system is the most efficient under which conditions? We have not yet a taxonomy that links types of production tasks to types of collective production systems. Therefore, only a few exploratory comments on this topic are possible.

One of production communities' striking features is their apparent waste of effort. In both OSS and scientific communities, many contributions are offered that will never be used. The decentralized task definition is liable to both misperceptions, that is, the production of contributions no one is interested in, and solutions of roughly the same problem by a multitude of contributors.

However, this apparent waste of effort and resources has major advantages for a collective production system that operates under *complete uncertainty*, which is the case for both OSS production communities and scientific communities. Complete uncertainty means that at any stage of the production process, the following are not clear:

- What exactly is the problem that needs solving (how should it be formulated)?

- Is there a solution to the problem at the current stage of knowledge?
- How could the problem be solved?
- What knowledge can be regarded as valid and reliable and should therefore be used for solving the problem?
- Who can solve the problem?

This is the case in science where each researcher formulates a problem in a particular way based on prior research and the current work environment, and in OSS production where the software informalisms are subject to idiosyncratic interpretations, and where the hardware and software environments of an OSS are constantly changing. Under these conditions, the apparent *waste of effort* is actually a very effective and probably even efficient use of resources. First, when nobody can say for sure what the problem is, decentralized task creation appears to be feasible because members of the concept of *production community* make as many independent attempts to formulate and solve problems as possible. While many (and sometimes most) of the attempts are bound to fail or to become redundant, the decentralized approach provides the highest likelihood that the problem is solved as quickly as possible.

The second advantage of production communities is that under the conditions of complete uncertainty, tasks are assigned to producers by "self-identification" (Benkler, 2002, pp. 414-415).

Table 1. Comparison of collective production systems

	Organization	Network	Market	Community
Membership constituted by	Formal rules	Negotiation	Exchange offer	Perception
Tasks created by	Ex ante division of labor		Decentralized autonomous decisions	
	Hierarchical	Negotiated		
Actions adjusted primarily by	Coordination by		Parametric adjustment to	
	Hierarchical decisions	Negotiations	Price	Subject matter of work
Integration of contributions based on	Preproduction decisions		Postproduction exchange	Postproduction peer review and use

The problem of who should do what, which would be impossible to solve by centralized decision making, is decided by every individual producer autonomously. The decentralized decision making on tasks guarantees the best possible fit of tasks and producers. Naturally, misperceptions and therefore a misallocation of tasks are still possible.

The third advantage of production communities that can outweigh the waste of effort is their mechanism of quality control. While it is often the case that peer review is seen as the major mechanism of quality control and an advantage of production communities, the quality control by use is much more important because it is more thorough in the long run. In production communities, contributions (code, respectively knowledge claims) are used in the further production process. Since production occurs in local settings, which vary from each other, each producer's work environment constitutes a specific test site for the community's common subject matter of work. The numerous tests of the common product that are permanently performed at these sites significantly enhance the quality of the product. In the case of scientific knowledge, this aspect of quality can be described without reference to truth as robustness, that is, as stability across different environments (Star, 1993). In the case of software, the major concern is the quality of the code. While there is still limited empirical evidence to back the claim that OSS is superior to proprietary software in terms of bug detection and fixing, one comparative study indicates that this advantage might indeed exist (Mockus et al., 2002).

FUTURE TRENDS AND CONCLUSION

OSS communities are production communities that apply a distinct mode of production that has so far been neglected by the literature. Production communities rely on decentralized task definition

that is ordered by the common subject matter of work, which is observed by all members. Decisions about the integration of contributions are made *ex post* by peer review and subsequent use. The use of contributions in a variety of local work environments is also the major mechanism of quality control.

The analysis has also revealed gaps in our knowledge about systems of collective production in general and about production communities in particular. Since OSS and scientific communities belong to the same type of collective production system, ideas and research questions could be exchanged. For example, strategies used in science studies for analysing knowledge structures and knowledge flows could be adopted for the investigation of OSS. In science, relationships between contributions are reflected in citations, which in turn enable the study of structures and flows of knowledge. Citation analyses have demonstrated that all scientific knowledge is interconnected and can be thought of as one body of knowledge, which is internally structured (Small & Griffith, 1974) and has fractal characteristics (Van Raan, 1990). The pendant to citation in science appears to be coupling, which reflects connections between modules. The analysis of coupling with techniques such as cross-referencing tools (Yu, Schach, Chen, & Offutt, 2004), design structure matrices (MacCormack, Rusnak, & Baldwin, 2004), and call-graph analysis (de Souza et al., 2005) should provide opportunities to study structures in OSS in a way similar to citation analysis.

Studies of production communities could benefit from the fact that OSS communities operate more explicitly than scientific communities. The explicit negotiations, decisions, and rules that characterize the production processes of OSS communities make them a suitable research object for studying the organizing functions of a production community's shared subject matter of work. For example, the social mechanisms that are at work in the branching out of knowledge production into different directions (*forking*) and

in the *dying* of a line of knowledge production (of an OSS project) can be assumed to be at least similar to those occurring in scientific communities, where they are less easy to identify and to observe because of the implicitness of knowledge structures and cognitive developments.

The introduction of the *production community* also challenges the theory of collective production systems. While the superiority of the communal mode of production for some types of software development tasks is felt by many observers, this point must also be made theoretically in the comparison of collective production systems. We need a generalized approach that relates types of production problems to types of collective production systems and enables a comparison of their advantages and disadvantages. Thus, in order to get more out of the many case studies on OSS production, we need more abstract and more comparative theory.

REFERENCES

- Benkler, Y. (2002). Coase's penguin, or, Linux and the nature of the firm. *Yale Law Journal*, 112, 369-446.
- Bergquist, M., & Ljungberg, J. (2001). The power of gifts: Organizing social relationships in open source communities. *Information Systems Journal*, 11, 305-320.
- Bezroukov, N. (1999). Open source software development as a special type of academic research (critique of vulgar Raymondism). *First Monday*, 4. Retrieved from http://www.firstmonday.dk/issues/issue4_10/bezroukov/index.html
- Bonaccorsi, A., & Rossi, C. (2003). Why open source software can succeed. *Research Policy*, 32, 1243-1258.
- Brint, S. (2001). Gemeinschaft revisited: A critique and reconstruction of the community concept. *Sociological Theory*, 19, 1-23.
- Crowston, K., Wei, K., Li, Q., & Howison, J. (2006). *Core and periphery in free/libre and open source software team communications*. Paper presented at the 39th Annual Hawaii International Conference on System Sciences, Waikoloa, HI.
- Dalle, J.-M., David, P. A., Ghosh, R. A., & Steinmueller, W. E. (2004). *Advancing economic research on the free and open source software mode of production* (SIEPR Discussion Paper 04-03). Stanford: Stanford Institute for Economic Policy Research.
- Dalle, J.-M., & Jullien, N. (2003). "Libre" software: Turning fads into institutions? *Research Policy*, 32, 1-11.
- Demil, B., & Lecocq, X. (2003). *Neither market nor hierarchy or network: The emerging bazaar governance*. Retrieved from <http://opensource.mit.edu/papers/demillecocq.pdf>
- De Souza, C., Froehlich, J., & Dourish, P. (2005). Seeking the source: Software source code as a social and technical artifact. In *Proceedings of the ACM International Conference on Supporting Group Work (GROUP 2005)* (pp. 197-206).
- Edwards, K. (2001). *Epistemic communities, situated learning and open source software development*. Retrieved from <http://opensource.mit.edu/papers/kasperedwards-ec.pdf>
- Gacek, C., & Arief, B. (2004). The many meanings of open source. *IEEE Software*, 21, 34-40.
- Gallivan, M. J. (2001). Striking the balance between trust and control in a virtual organization: A content analysis of open source software case studies. *Information Systems Journal*, 11, 277-304.
- Garzarelli, G., & Galoppini, R. (2003). *Capability coordination in modular organization: Voluntary FS/OSS production and the case of Debian GNU/Linux*. In *Economics Working Paper Archive at WUST, Industrial Organization*. Retrieved February 23, 2006, from <http://opensource.mit.edu/papers/garzarelligaloppini.pdf>

- Gasser, L., & Ripoche, G. (2003). *Distributed collective practices and free/open-source software problem management: Perspectives and methods*. Paper presented at the 2003 Conference on Cooperation, Innovation & Technologies (CITE2003). Retrieved from <http://www.isrl.uiuc.edu/~gasser/papers/cite-gasser.pdf>
- Ghosh, R. A. (1998). Cooking pot markets: An economic model for the trade in free goods and services on the Internet. *First Monday*, 3. Retrieved from http://www.firstmonday.org/issues/issue3_3/ghosh/index.html
- Ghosh, R. A., & Prakash, V. V. (2000). *The Orbiten free software survey*. Retrieved from <http://www.orbiten.org/ofss/01.html>
- Gläser, J. (2001, December). "Producing communities" as a theoretical challenge. Paper presented at the TASA 2001 Conference, Sydney, Australia.
- Gläser, J. (2006). *Wissenschaftlich produktionsgemeinschaften: Die soziale ordnung der forschung*. Frankfurt am Main, Germany: Campus, im Erscheinen.
- Hayek, F. A. (1945). The use of knowledge in society. *The American Economic Review*, 35, 519-530.
- Hayek, F. A. (1991). Spontaneous ("grown") order and organized ("made") order. In G. Thompson, J. Frances, R. Levacic, & J. Mitchell (Eds.), *Markets, hierarchies and networks: The coordination of social life* (pp. 293-301). London: SAGE Publications Ltd.
- Healy, K., & Schussman, A. (2003). *The ecology of open-source software development*. Retrieved from <http://opensource.mit.edu/papers/healy-schussman.pdf>
- Hedström, P. (2005). *Dissecting the social: On the principles of analytical sociology*. Cambridge, UK: Cambridge University Press.
- Hess, D. J. (2005). Technology- and product-oriented movements: Approximating social movement studies and science and technology studies. *Science, Technology, and Human Values*, 30, 515-535.
- Holck, J., & Jørgensen, N. (2005). Do not check in on red: Control meets anarchy in two open source projects. In S. Koch (Ed.), *Free/open source software development* (pp. 1-25). Hershey, PA: Idea Group Publishing.
- Hollingsworth, J. R., & Boyer, R. (1997). Coordination of economic actors and social systems of production. In J. R. Hollingsworth & R. Boyer (Eds.), *Contemporary capitalism: The embeddedness of institutions* (pp. 1-47). Cambridge, UK: Cambridge University Press.
- Holtgrewe, U., & Werle, R. (2001). De-commodifying software? Open source software between business strategy and social movement. *Science Studies*, 14, 43-65.
- Iannacci, F. (2003). *The Linux management model*. Retrieved from <http://opensource.mit.edu/papers/iannacci2.pdf>
- Jørgensen, N. (2001). Putting it all in the trunk: Incremental software development in the FreeBSD open source project. *Information Systems Journal*, 11, 321-336.
- Knorrr-Cetina, K., & Merz, M. (1997). Floundering or frolicking: How does ethnography fare in theoretical physics? (And what sort of ethnography?) A reply to Gale and Pinnick. *Social Studies of Science*, 27, 123-131.
- Koch, S., & Schneider, G. (2002). Effort, cooperation and co-ordination in an open source software project: GNOME. *Information Systems Journal*, 12, 27-42.
- Lanzara, G. F., & Morner, M. (2003). *The knowledge ecology of open-source software projects*. Paper presented at the 19th EGOS Colloquium, Copenhagen, Denmark.

- Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics*, 50, 197-234.
- Lindblom, C. E. (1965). *The intelligence of democracy: Decision making through mutual adjustment*. New York: The Free Press.
- MacCormack, A., Rusnak, J., & Baldwin, C. (2004). *Exploring the structure of complex software designs: An empirical study of open source and proprietary code*. Unpublished manuscript.
- Mayntz, R. (1993). Policy-netzwerke und die logik von verhandlungssystemen. In A. Hritier (Ed.), *Policy-analyse: Kritik und neuorientierung* (pp. 39-56). Opladen: Westdeutscher Verlag.
- Mayntz, R. (2004). Mechanisms in the analysis of social macro-phenomena. *Philosophy of the Social Sciences*, 34, 237-259.
- Merz, M., & Knorr-Cetina, K. (1997). Deconstruction in a "thinking" science: Theoretical physicists at work. *Social Studies of Science*, 27, 73-111.
- Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11, 309-346.
- Powell, W. W. (1990). Neither market nor hierarchy: Network forms of organization. *Research in Organizational Behavior*, 12, 295-336.
- Raymond, E. (1999). The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12, 23-49.
- Robles, G., Scheider, H., Tretkowski, I., & Weber, N. (2001). *Who is doing it? A research on Libre software developers*. Retrieved from <http://widi.berlios.de/paper/study.html>
- Scacchi, W. (2002). Understanding the requirements for developing open source software systems. *IEEE Proceedings: Software*, 149, 24-39.
- Scharpf, F. W. (1997). *Games real actors play: Actor-centered institutionalism in policy research*. Boulder, CO: Westview Press.
- Scott, W. R. (1992). *Organizations: Rational, natural, and open systems*. Englewood Cliffs, NJ: Prentice-Hall.
- Seiferth, C. J. (1999). Open source and these United States. *Knowledge, Technology & Policy*, 12, 50-79.
- Sharma, S., Sugumeran, V., & Rajagopalan, B. (2002). A framework for creating hybrid-open source software communities. *Information Systems Journal*, 12, 7-25.
- Simon, H. A. (1991). Organizations and markets. *Journal of Economic Perspectives*, 5, 25-44.
- Small, H., & Griffith, B. C. (1974). The structure of scientific literatures I: Identifying and graphing specialities. *Science Studies*, 4, 17-40.
- Star, S. L. (1993). Cooperation without consensus in scientific problem solving: Dynamics of closure in open systems. S. Easterbrook (Ed.), *CSCW: Cooperation or conflict?* (pp. 93-106). London: Springer.
- Tuomi, I. (2001). Internet, innovation, and open source: Actors in the network. *First Monday*, 6. Retrieved from http://firstmonday.org/issues/issue6_1/tuomi/index.html
- Van Raan, A. F. J. (1990). Fractal dimension of co-citations. *Nature*, 347, 626.
- Von Hippel, E., & von Krogh, G. (2003). Open source software and the "private-collective" innovation model: Issues for organization science. *Organization Science*, 14, 209-223.
- Von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, 32, 1217-1241.

Von Krogh, G., & von Hippel, E. (2003). Editorial: Special issue on open source software development. *Research Policy*, 32, 1149-1157.

West, J., & O'Mahony, S. (2005). Contrasting community building in sponsored and community founded open source projects. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Waikoloa, HI. Retrieved from <http://opensource.mit.edu/papers/westmahony.pdf>

Yu, L., Schach, S. R., Chen, K., & Offutt, J. (2004). Categorization of common coupling and its application to the maintainability of the Linux kernel. *IEEE Transactions on Software Engineering*, 30, 694-706.

Zeitlyn, D. (2003). Gift economies in the development of open source software: Anthropological reflections. *Research Policy*, 32, 1287-1291.

KEY TERMS

Community: A group of actors who share a collective identity that is based on the perception of having something in common, and who adjust some of their actions because of this identity.

Peer Review: Process in which people engaged in the same kind of work judge the quality of one's work, comment on it, and make decisions on these judgements.

Production Community: A community whose members jointly produce a good by autonomously deciding about their contributions by adjusting their decisions to the common subject matter of work.

Social Mechanism: A sequence of causally linked events that occur repeatedly in reality if certain conditions are given and link specified initial conditions to a specific outcome (Mayntz, 2004, p. 241).

Social Order: A state of a group of actors that is characterized by a mutual adjustment of actions.

ENDNOTES

- ¹ The concept of "thin description" has been developed by Merz and Knorr-Cetina (Knorr-Cetina & Merz, 1997; Merz & Knorr-Cetina, 1997) in their analysis of the work of theoretical physicists.
- ² Each open source software is accompanied by a variety of Web-based descriptions that contain requirements for the software (the so-called "software informalisms," Scacchi, 2002; see also Gasser & Ripoché, 2003).
- ³ While the openness of source may be an ideology that is rooted in the Hacker movement (Holtgrewe & Werle, 2001, pp. 52-53; von Krogh & von Hippel, 2003, pp. 1150-1151), it is also a logistic prerequisite for the communal production of software.
- ⁴ As is the case with OSS production, the description of the other systems of collective production needs to be synthesized from a dispersed literature (Gläser, 2006). Key publications that support the interpretation applied here are Hayek (1945) for markets, Simon (1991) and Scott (1992) for organisations, Powell (1990) and Mayntz (1993) for networks, and Hollingsworth and Boyer (1997) and Scharpf (1997) for the comparative approach.

Table of Contents

Foreword	xxvi
Preface	xxx
Acknowledgment	xxxvi

Section I Culture, Society, and Open Source Software

Chapter I Free Software Philosophy and Open Source / <i>Niklas Vainio and Tere Vadén</i>	1
Chapter II Greasemonkey and a Challenge to Notions of Authorship / <i>Brian D. Ballentine</i>	12
Chapter III Morality and Pragmatism in Free Software and Open Source / <i>Dave Yeats</i>	23
Chapter IV Hacker Culture and the FLOSS Innovation / <i>Yu-Wei Lin</i>	34
Chapter V Social Technologies and the Digital Commons / <i>Francesca da Rimini</i>	47
Chapter VI ALBA Architecture as Proposal for OSS Collaborative Science / <i>Andrea Bosin, Nicoletta Dessì, Maria Grazia Fugini, Diego Liberati, and Barbara Pes</i>	68
Chapter VII Evaluating the Potential of Free and Open Source Software in the Developing World / <i>Victor van Reijswoud and Emmanuel Mulo</i>	79
Chapter VIII Open Source Software: A Developing Country View / <i>Jennifer Papin-Ramcharan and Frank Soodeen</i>	93

Chapter IX

The Social and Economical Impact of OSS in Developing Countries / *Alfreda Dudley-Sponaugle, Sungchul Hong, and Yuanqiong Wang*..... 102

Section II

Development Models and Methods for Open Source Software Production

Chapter X

Dependencies, Networks, and Priorities in an Open Source Project / *Juha Järvensivu, Nina Helander, and Tommi Mikkonen*..... 116

Chapter XI

Patchwork Prototyping with Open Source Software / *M. Cameron Jones, Inghbert R. Floyd, and Michael B. Twidale*..... 126

Chapter XII

An Agile Perspective on Open Source Software Engineering / *Sofiane Sahraoui, Noor Al-Nahas, and Rania Suleiman*..... 141

Chapter XIII

A Model for the Successful Migration to Desktop OSS / *Daniel Brink, Llewellyn Roos, Jean-Paul Van Belle, and James Weller*..... 154

Chapter XIV

The Social Order of Open Source Software Production / *Jochen Gläser*..... 168

Section III

Evaluating Open Source Software Products and Uses

Chapter XV

Open Source Software: Strengths and Weaknesses / *Zippy Erlich and Reuven Aviv*..... 184

Chapter XVI

Open Source Software Evaluation / *Karin van den Berg*..... 197

Chapter XVII

Open Source Web Portals / *Vanessa P. Braganholo, Bernardo Miranda, and Marta Mattoso*..... 211

Chapter XVIII

Curious Exceptions? Open Source Software and “Open” Technology / *Alessandro Nuvolari and Francesco Rullani*..... 227

Chapter XIX

Reducing Transaction Costs with GLW Infrastructure / *Marcus Vinicius Brandão Soares*..... 240

Chapter XX	
Issues to Consider when Choosing Open Source Content Management Systems (CMSs) /	
<i>Beatrice A. Boateng and Kwasi Boateng</i>	255
Chapter XXI	
Evaluating Open Source Software through Prototyping / <i>Ralf Carbon, Marcus Ciolkowski,</i>	
<i>Jens Heidrich, Isabel John, and Dirk Muthig</i>	269
Chapter XXII	
Motives and Methods for Quantitative FLOSS Research / <i>Megan Conklin</i>	282
Chapter XXIII	
A Generalized Comparison of Open Source and Commercial Database Management Systems /	
<i>Theodoros Evdoridis and Theodoros Tzouramanis</i>	294
Chapter XXIV	
Evaluation of a Migration to Open Source Software / <i>Bruno Rossi, Barbara Russo, and</i>	
<i>Giancarlo Succi</i>	309
Section IV	
Laws and Licensing Practices Affecting Open Source Software Uses	
Chapter XXV	
Legal and Economic Justification for Software Protection / <i>Bruno de Vyyst and Alea Fairchild</i>	328
Chapter XXVI	
OSS Adoption in the Legal Services Community / <i>Ray Agostinelli</i>	340
Chapter XXVII	
The Road of Computer Code Featuring the Political Economy of Copyleft and Legal Analysis of	
the General Public License / <i>Robert Cunningham</i>	348
Chapter XXVIII	
The Evolution of Free Software / <i>Mathias Klang</i>	363
Chapter XXIX	
Free Access to Law and Open Source Software / <i>Daniel Poulin, Andrew Mowbray, and</i>	
<i>Pierre-Paul Lemyre</i>	373
Chapter XXX	
Examining Open Source Software Licenses through the Creative Commons Licensing Model /	
<i>Kwei-Jay Lin, Yi-Hsuan Lin, and Tung-Mei Ko</i>	382
Chapter XXXI	
FLOSS Legal and Engineering Terms and a License Taxonomy / <i>Darren Skidmore</i>	394

Section V

Public Policy, the Public Sector, and Government Perspectives on Open Source Software

Chapter XXXII

On the Role of Public Policies Supporting Free/Open Source Software / *Stefano Comino, Fabio M. Manenti, and Alessandro Rossi* 412

Chapter XXXIII

Use of OSS by Local E-Administration: The French Situation / *Laurence Favier, Joël Mekhantar, and Marie-Noëlle Terrasse* 428

Chapter XXXIV

Issues and Aspects of Open Source Software Usage and Adoption in the Public Sector / *Gabor Laszlo* 445

Chapter XXXV

The Labor Politics of Scratching an Itch / *Casey O'Donnell* 460

Chapter XXXVI

Open Source Technology and Ideology in the Nonprofit Context / *Jonathan Peizer* 468

Chapter XXXVII

Governance and the Open Source Repository / *R. Todd Stephens* 480

Section VI

Business Approaches and Applications Involving Open Source Software

Chapter XXXVIII

Analyzing Firm Participation in Open Source Communities / *Wouter Stam and Ruben van Wendel de Joode* 495

Chapter XXXIX

Community Customers / *Jeroen Hoppenbrouwers* 510

Chapter XL

Open Source Software Business Models and Customer Involvement Economics / *Christoph Schlueter Langdon and Alexander Hars* 522

Chapter XLI

Investing in Open Source Software Companies: Deal Making from a Venture Capitalist's Perspective / *Mikko Puhakka, Hannu Jungman, and Marko Seppänen* 532

Chapter XLII

Revenue Models in the Open Source Software Business / *Risto Rajala, Jussi Niissilä, and Mika Westerlund* 541

Chapter XLIII	
Open Source for Accounting and Enterprise Systems / <i>Thomas Tribunella and James Baroody</i>	555

Chapter XLIV	
Open Source Software and the Corporate World / <i>Sigrid Kelsey</i>	570

Chapter XLV	
Business Models in Open Source Software Value Creation / <i>Marko Seppänen, Nina Helander, and Saku Mäkinen</i>	578

Chapter XLVI	
Novell's Open Source Evolution / <i>Jacobus Andries du Preez</i>	590

Section VII

Educational Perspectives and Practices Related to Open Source Software

Chapter XLVII	
Communities of Practice for Open Source Software / <i>Leila Lage Humes and Nicolau Reinhard</i>	610

Chapter XLVIII	
Selecting Open Source Software for Use in Schools / <i>Kathryn Moyle</i>	624

Chapter XLIX	
Open Source E-Learning Systems: Evaluation of Features and Functionality / <i>Phillip Olla</i>	638

Chapter L	
The Role of Open Source Software in Open Access Publishing / <i>David J. Solomon</i>	649

Chapter LI	
An Innovative Desktop OSS Implementation in a School / <i>James Weller and Jean-Paul Van Belle</i>	659

Chapter LII	
Rapid Insertion of Leading Edge Industrial Strength Software into University Classrooms / <i>Dick B. Simmons, William Lively, Chris Nelson, and Joseph E. Urban</i>	670

Chapter LIII	
Wikis as an Exemplary Model of Open Source Learning / <i>Robert Fitzgerald</i>	681

Chapter LIV	
A Perspective on Software Engineering Education with Open Source Software / <i>Pankaj Kamthan</i>	690

About the Contributors	703
-------------------------------------	-----

Index	720
--------------------	-----