Shared mental models among open source software developers

Barbara Scozzi[1], Kevin Crowston[2], U. Yeliz Esereyel[2], Qing Li[2]
1 Il Facolta' di Ingegneria, Politecnico di Bari, Taranto, Italy
2 School of Information Studies, Syracuse University, Syracuse, NY 13210 USA

Abstract[1]

Shared understandings are important for software development as they guide to effective individual contributions to, and coordination of, the software development process. In this paper, we present the results of a preliminary analysis on shared mental models within Free/Libre Open Source Software (FLOSS) development teams. Based on structuration theory and by adopting cognitive mapping and process analysis, we represented and compared the mental models of some developers of the Lucene Java project. Our analysis suggests that there is a high-level of sharing among core developers but the sharing is not complete, with some differences related to tenure in the project.

1. Introduction

Distributed teams are groups of geographically dispersed individuals working together over time towards a common goal. Advances in information and communication technology are crucial enablers for recent development of this organizational form and as a result, distributed teams are becoming more popular. Distributed teams seem particularly attractive for software development because the code can be shared via the same systems used to support team interactions [1].

While distributed teams have many potential benefits, distributed workers face many real challenges. [2] suggest that distributed work is characterized by numerous discontinuities: a lack of coherence in some aspects of the work setting (e.g., organizational membership, business function, task, language or culture) that hinders members in making sense of the task and communications from others [3], or produces unintended information filtering [4] or misunderstandings [5]. The presence of discontinuities seems likely to be particularly problematic for software developers [3]. Numerous studies of the social aspects of software development teams [e.g., 6, 7] conclude that large system development requires knowledge from many domains, which is thinly spread among different developers [6]. As a result, large projects require a high degree of knowledge integration and the coordinated efforts of multiple developers [8]. The additional effort required for distributed work often translates into delays compared to traditional face-to-face teams [9].

In response to the problems created by discontinuities, studies of distributed teams stress the need for a significant amount of time spent learning how to communicate, interact and socialize using computer-supported communications tools [10]. In this study, we focus specifically on the role of shared mental models (e.g., common conceptions of the project, other team members, users, competitors or programming standards) that guide team members' behaviours and shape their actions.

The goals of the current study are finding evidence for the existence of shared mental models that shape teamwork practices. Specifically, the study addresses these research questions:

1. To what degree are the mental models of project team members shared? Which aspects are common and which unique to developers?
2. What factors (either project or individual) are related to the sharing of mental models? Can we predict which aspects are likely to be shared or not shared?
3. What are implications of shared mental models for team performance?

In this paper, we report on a preliminary analysis of the degree of sharing of mental models among developers in one project. The following section presents the theoretical basis for our study. Subsequent sections present the methodology for data elicitation and analysis and our findings. We conclude by discussing advantages and disadvantages of the adopted approach and plans for future research.

2. Theory

Shared mental models are defined by Cannon-Bowers & Salas [11] as:

> knowledge structures held by members of a team that enable them to form accurate explanations and expectations for the task, and in turn, to coordinate their actions and adapt their behavior to demands of the task and other team members (p. 228).

The issue is not so much whether team members have mental models, but rather the degree of similarity among the models of team members. Prior research suggests that the existence of accurate shared mental models that guide member actions are important for team effectiveness [11].

[1] Under review at the *Hawai'i International Conference on System Science*. Please do not cite or quote. This research was partially supported by NSF Grants 03-41475, 04–14468 and 05–27457.

To conceptualize what kinds of shared mental models will be important to effective action, we draw inspiration from structuration theory [12]. Structuration theory describes structure as the rules and resources that influence, guide or justify individual action and that are simultaneously created by those actions. We chose this framework because structure is "encoded in actors' stocks of practical knowledge" [13] and "instantiated in recurrent social practice" [14]. Therefore, consideration of how structure has been conceptualized in prior work will provide insight into the kinds of shared mental models that might be relevant for guiding group work. For this study, we consider three kinds of rules and resources identified in prior research: 1) interpretive schemes, 2) resources, and 3) norms [13, 15]. In the remainder of this section, we discuss these in turn.

Interpretive schemes and structures of signification. Individual actors' interpretive schemes create structures of *signification*. Research suggests that shared interpretive schemes help improve performance in face-to-face and distributed teams [16]. Shared interpretive schemes about tasks and actors' abilities can enable teams to coordinate their activities without the need for explicit communications [17, 18]. Research on software development in particular has identified the importance of shared understanding in the area of software development. Curtis et al. [19], note that, "a fundamental problem in building large systems is the development of a common understanding of the requirements and design across the project team." They go on to say that, "the transcripts of team meetings reveal the large amounts of time designers spend trying to develop a shared model of the design". The problem of developing shared interpretive schemes is likely to particularly affect FLOSS development, since FLOSS team members are distributed, have diverse backgrounds, and join FLOSS teams in different phases of the software development process [20]. In short, shared interpretive schemes are important as guides to effective individual contributions to, and coordination of the software development process.

Resources, roles and structures of domination. The control of resources is the basis for power and thus for structures of *domination*. For software development, material resources would seem to be less relevant, since the work is intellectual rather than physical and development tools are readily available, thanks to FLOSS development systems such as SourceForge (http://sourceforge.net/) and Savannah (http://savannah.gnu.org/). Furthermore, most FLOSS teams have a stated ethos of open contribution. However, team members face important differences in access to expertise and control over system source code in particular, which are encoded in the rights accorded to different roles.

Prior case studies have described how individuals move from role to role as their involvement with a project changes. For example, a common pattern is for active users to be invited to join the core development team in recognition of their contributions and ability. In some teams, this selection is an informal process managed by the project initiator, while others such as the Apache Project, have formal voting processes for new members. On the other hand, we are still learning how the privileges and responsibilities of these different roles are defined. Again, some projects seem to have formal role definitions, while in others, roles seem to be more emergent.

Rules and norms and structures of legitimation. Finally, actors' social norms and team rules embody structures of *legitimation*. [21] notes that rules allow FLOSS developers to form stable expectations of others' actions, thus promoting coordination. The importance of such rules have been documented in conventional software and FLOSS development teams [e.g., 22, 23]. For example, [24] describes a set of implicit and explicit rules for software development in the FreeBSD project (e.g., "Don't break the build"), while [25] notes implicit rules regarding project forking at the community level.

3. Data

We next describe how we obtained data for our study, covering in turn our data elicitation approach and subject selection.

3.1. Data elicitation

To elicit data to address our research questions, we interviewed developers active in FLOSS projects. Interviews followed a semi-structured protocol based on the theory above, designed to elicit information on how team members interpret their role and the other members' roles, how they act and the reasons for their behaviours, tacit norms and practices and the way such practices have arisen. Specifically, the interview protocol included:

- *Project rules and norms.* Any explicitly stated norms or rule as perceived by developers.
- *Project environment and constraints.* The environment in which the team operates, constraints that they have to deal with, customers and competitors.
- *Development strategy.* The overall approach to project development.
- *Development process.* Process by which the software is developed (activities, dependencies, coordination mechanisms), tools and technology used for software development, submit and handle bugs, patches and feature requests, and decision-making processes.
- *Team organization*. Team structure and specific team roles.

The decision to adopt a semi-structured protocol was driven by the techniques we decided to use for the analysis, as discussed in the next section.

3.2. Setting and subjects

The results presented in this paper are based on interviews carried out with four developers (of a total of seven committers) affiliated with the Apache Lucene Java pro-

ject. Lucene Java is a high-performance, full-featured text search engine library written in Java (http://lucene.apache.org), suitable for applications that require full-text search. It is a sub-project under the Apache Lucene top-level project.

One author interviewed the four Lucene Java developers at an Apache-sponsored conference (represented in the rest of the paper by code letters A to D). All four were male. Three of the interviewees were committers in Lucene Java. The committers had different seniority with the project, having been a committer for 2 years (A), 1 year (B) and 3 months respectively (C). The final interviewee (D) was a Project Management Committee (PMC) member for the Lucene top-level project, a contributor to Lucene Java and a committer for another Lucene sub-project (Nutch). The interviews were conducted separately, lasted between 30 and 75 minutes and were recorded and then transcribed for analysis.

4. Analysis

Three of the authors separately and then collectively analyzed the interview transcripts adopting an inductive approach. The text of the interviews was carefully read to identify instantiations of the following mental models:

- *Interpretative schemes*. Interpretive schemes include:
  1) definitions of key aspects of the projects;
  2) causal cognitive maps of project features (e.g., history, key aspects, norms and practices, and organization, in the view of the developers);
  3) processes carried out within the projects.

  The analysis process for these elements is described in more detail in subsequent sections.
- *Project roles and resources.* Roles, their responsibilities and privileges and the organization structure.
- *Adopted rules and norms.* Norms are accepted values or ways of behaving. Rules are explicit (hence written) norms.

Some aspects of these structures overlap, e.g., some topics are included in both the process and causal maps, and some project roles are also given in definitions.

4.1. Causal cognitive maps

Causal cognitive maps (hereafter referred to as causal maps) are graphic tools used to represent a person's views of a given issue. A causal map is composed of concepts and causal links among them [26]. Concepts represent ideas, opinions and key issues associated to the topic of the map. These are linked by causal relationships, which can be mainly distinguished in cause/effect (which do not imply intentionality) or means/end relationships. Concepts that represent the cause or the means to achieve a given goal are situated at the arrow's tail, concepts that represent the effect or the end at the arrow's head.

Causal maps can be used with different purposes. In this project, they have been adopted for an explicative purpose, i.e., finding evidences of the existence of shared mental models among FLOSS developers working on the same project. In particular, the maps are used to represent and compare the interpretative schemes of the developers so as to gauge the degree of common knowledge as well as to better understand the reasons that underlie team members actions and the dynamics based on which common interpretative schemes, if any, arise. The causal maps were developed using a technique called Documentary Coding Method [27], which involves the identification of the main concepts cited by the respondents during interviews and the relationships among them.

Once developed, different methodologies can be used to analyze and compare maps. In most studies quali-quantitative metrics, e.g., number of heads, tails, domain and centrality are used [28]. Some scholars have also defined ad hoc metrics to compare maps. [29]. Maps can be analyzed and compared by measuring the following quali-quantitative metrics:

- *Heads and Tails.* Map heads are those nodes that only have arrows going inside (no arrows go outside). Heads are representative of the developers' final end/goal and/or the effects of their perception. Tails are those nodes that only have arrows going outside (no arrows go inside). Tails explain/describe the causes of some perceptions and/or the means adopted to achieve goals.
- *Domain and Centrality.* Domain and centrality provide information about the importance of concepts. In particular, the domain score of a concept is given by the sum of direct links (both as input and output) the attendant node has. The centrality score of a concept is given by the sum of both direct and indirect links the attendant node has, so providing information on those concepts that are often unconsciously considered as the most relevant/central.
- *Sets*. Sets are groups of concepts that deal with a specific issue or topic. Topics were assigned to sets by the authors and again, disagreements about the assignment of topics to sets were resolved by discussion. By counting the number of concepts mentioned for each set it is possible to assess the importance/complexity associated with the topic of the set.

4.2. Process maps

A process is a set of activities that, by using different inputs, carries out an output [30]. Adopting a process view (or a process theory) means explaining how outcomes of interest develop through a sequence of events. A process map is a graphical representation of the process carried out within an organization. Many techniques have been proposed in the literature to map processes [31]. The main objective of such techniques is to gather the information necessary to analyze and, eventually, improve the process. In the paper, process maps were adopted to describe which tasks are accomplished within each project, how and by whom they are performed and which are the dependencies emerging among them. To this aim, by

carefully reading the interview texts, we first identified the processes mentioned by developers. Successively, we identified the task, the involved roles and dependencies among task per each process. Finally, we compared the maps of the processes as described by different developers so identifying eventual differences in the tasks, roles and/or sequences mentioned.

4.3. Reliability and cross-subject comparisons

To ensure reliability, the interpretative schemes, roles and resources, and rules and norms identified by each analyst for each interview were compared. When consistency was not achieved (as happened especially for causal cognitive maps), the authors reviewed the considered text together until they achieved agreement.

Once we had an agreed set of models for each interview, we compared the models across the individuals. We first listed and examined the items for which a definition was provided. The degree of similarity of the item definitions provided in different interviews was then qualitatively assessed. A similar approach was used to compare the view of developers on roles and resources, and norms and rules.

5. Findings

In this section, we discuss in broad terms our findings, covering in turn definitions, causal cognitive maps, processes, roles and rules and norms. Table 1 summarizes the number of concepts in each model identified per interviewee and identifies concepts that appeared in multiple interviews (the number of such concepts is reported in bracket).

5.1. Definitions

The analysis of the definitions provided by the interviewees are provided in Table 2. An important point is the high degree of sharing of key definitions, such as project goals, users and challenges. For example, both senior members (A and B) mentioned that the team does not have clearly stated goals, but yet the community works towards the same goals. As one of them put it; "It's really kind of a free flowing communal meeting of the minds". This is also evident in that all three members identified the goal as developing a search library. Only the non-committer member described the project as "information retrieval" project. Similarly, three interviewees described the intended users as people who want to incorporate search into their applications. When asked for challenges related to the project, all members were able to identify some challenges, yet, it seems that none of them take those as "problems". All three members clearly stated that "there aren't any big problems in the project".

An interesting difference is in the descriptions was in the area of the challenges that the new members face. The most senior committer A thought new members did not face any particular challenges, since they will have been part of the community for a while before becoming a committers. On the other hand, the second most senior committer, B, suggested the challenges of getting up to speed on Apache infrastructure, commit rights, and so on. The newest committer, C, identified four different challenges, such as the burden of suddenly being responsible and writing a good code, as well as trying to work with non-committers and encouraging them to submit patches. Finally, D thought the project is complex to jump in, so people need to go through quite bit learning. Meanwhile, he also has an opposite opinion: the barrier to enter is low.

5.2. Causal cognitive maps

The causal maps for the interviews included a large number of concepts (ranging from 63, in the case of D, to 153 in the case of B) but with only a relatively small number of causal connections. In our case, we found that the head concepts seemed to reflect the structure of the interview, making them less useful for comparing across interviewees. We have not attempted to summarize the large number of tail concepts.

We turn next to a comparison of the causal maps examining central concepts. Many concepts with the highest domain scores also have the highest centrality scores (Table 3 shows concepts with the highest domain score), suggesting which concepts are considered as the most relevant by each developer. Concepts that have the highest centrality scores (but not the highest domain scores) are associated to: the abilities of new members and new committers and the reasons to take part to the project for A; some project strengths and some aspects of the Apache project for B; some project strengths in the case of C. Finally, for D they are the fact Lucene Java is used in many projects, the project founder's contribution and the committers' mindset.

The success of Lucene Java is central (both in terms of centrality and domain), thus most relevant, in three maps. However, apart from it, relevant concepts differ from developer to developer. For example, A describes the abilities of another committer, and also mentions a step of the procedure to become a committer. B talks of a step of the new member hiring procedure, C mentions some project strengths and a step of the committing procedure, Finally, D talks of the different modes to contribute to the project, the extension of the community and new members' main issue and problems.

Finally, by examining the concepts presented in the maps, nine sets were identified, namely:

- Challenges: Challenges of the project in general and the challenges of new members
- Change in project: Change in the project over time.
- Community: Number of members, the roles of community members, how community gets along.
- Coordination: How coordination problems/issues are addressed.
- Goals: Goals of the project.

- History: How the project was initiated.
- Leadership: Who the leaders are and why they are mentioned as leaders.
- Membership: New member selection, skill and knowledge needed by members.
- Success/Strengths: Reasons for project success, strengths of the software and team.

The relevance of each set (i.e., the number of concepts per set) is an indication of the importance that different issues have. As shown in Table 4, issues related to community are the most cited by three developers. Challenges and success/strengths are the most important issues for the two of them.

5.3. Processes

We turn next to a consideration of the processes identified by the interviewees. Three of the interviewees described two-three processes whereas one described in detail seven different processes (Table 5). However, this difference might be attributed to the longer interview period. All of the members had the same understanding of how the project got initiated. Although all of the interviewees described the member selection process, they described it with a different detail level. All interviewees mentioned that one person nominates a candidate and then PMC votes on the membership. Three interviewees suggested multiple criteria for someone to be nominated that include high quality contribution over a long period of time and making positive comments on the mailing list. The release process and bug-fixing or feature-adding processes are also described by two of the interviewees.

5.4. Roles

We turn now to a consideration of the roles identified, in particular the role of the project founder and of project leaders.

All interviewees recognized the multiple level of community, in terms of internal people and external people, and also people working on coding and people working on answering question in email-lists. All of them emphasize the importance of community and consider community as one of project strengths and success factors. Compared to D and C, A and B provided more overall view of community. A said “it is viable sustainable community", because of the increasing interest and usage in both company and individual level. People leave, more new people come in. A also considers new committers as the core developers doing a large amount of codes at different time. B said “it is vibrant community", which contributes to project success, also builds on it. Same as A, B emphasized the importance of new people and he clearly connected the spike of group activity with new committers.

When defining the role of the community, B, C and D have similar opinions. B referred to the role of the community as “they provide feedback about what works and what doesn’t work”. When referring to the community role C said “users give feedback how they use Lucene. We answer the questions. They ask for new features”. D said “one big strength is the community. There are lot of people who know that Lucene very well and are kind of supporting everybody who is trying to use it”.

Interviewee D identified seven committers and PMC select members. C mentioned criteria of selecting members as “specialty in certain area, active for a while, submit good patches and responsible enough.” Also he identified the development skill as “knowledge in search, java and know how open source works”. B mentioned the “litmus test” as “contributing high quality stuff, for a time period”, as well as “cordial to each other”. He listed skill as “java, personality skill, specialty, know where to look in other parts of code, know how to ask, know what you know and don’t know”. A mentioned that to “become expert level users, contribute some patches, not be abrasive.” He also mentioned the PMC as “a step over committer” when “made a lot of contribution and sustained”.

A mentioned that people defer to others in certain part of the code because of their specialty in certain areas which earn the respect. B uses the same word “defer to others when it comes to some part of it”. C also briefly mentioned “specialists in certain component”.

Interviewees also had interesting comments that help us understand the leadership dynamics in Open Source. Two senior interviewees mentioned that “the project founder will always sort of be the head [leader] still”. The project founder is not as active in Lucene Java currently. Yet, “his opinion carries a lot of weight in the community” as stated by two interviewees. They also mentioned that he limits his comments not to influence the community. On the other hand, two interviewees identified other leadership dynamics in the team. The most senior committer suggested that there are either no leaders at this time or multiple leaders. He also suggested that there might be leadership in certain parts of the project, as well as perhaps a leader from the overall organizational perspective. He suggested that he might be one of the leaders due to the work he did the previous year as well as the organizational work (such as getting releases) that he did. In fact, the second senior committer also identified him as one of the two current leaders. According to these two interviewees, leadership seems to be correlated with sustained contribution. On the other hand, the third committer perceives leadership negatively, as almost being close to dictatorship, and thus identifies no leader. He says “there are no leaders in the team, everybody has a say and rights”.

5.5. Norms/rules

Finally, we consider some norms expressed in the interviewees. An important norm is that members be cordial to each other in interactions. A lists "nice enough" as an important criteria to select new members. Also, they try to avoid friction by really convincing people. B points out

the rule directly about "we all try to be cordial with each other." And he thinks “the project founder brings a lot in this area”. A emphasizes the importance of community, so they want to avoid frictions and convince people.

A mentioned “no much group members work on coordinating coding effort. Some users tried, but didn’t work. E.g. working on road map, pushing release”. He points out he does not want a release plan. B mentioned something there are plan for “big issue”, rather than routine work, but not explicitly task assignement. He also pointed out some rules about how to commit or release.

6. Discussion

Our findings, as reported above, do show a degree of sharing of mental models. As to the interpretive schemes, key definitions (e.g. project goals, users and challenges) have a high degree of sharing among developers. Some aspects of the cause maps are shared as well. For example, concepts related to project success/strengths, challenges and the role of community are central and/or relevant in most maps. As to roles, the importance of community is stressed by all members. Finally, norms and rule show a high degree of sharing.

However, some differences in the views of developers also emerge. Some of them seem to be related to tenure in the project.

6.1. Benefit and drawbacks of causal maps

The main benefit that derives from the adoption of the maps is the ease of the analysis of different perspectives. The graphical representation facilitates identification of the key issues and the differences among different positions. Moreover, the adopted metrics facilitate the understanding of concepts or relationships not perfectly clear or conscious to individuals. These relationships can be more easily stressed than is the case when other qualitative tools (such as case studies or simple interviews) are used.

Of course, causal maps also present some drawbacks. In particular, the stage of the knowledge elicitation (interviews and codification of collected data) is the most critical. As most of the qualitative research methodologies, the knowledge schemes of the interviewer (i.e., the researcher) can strongly influence the findings. By knowledge scheme we mean the culture, interests and experiences of the interviewer. The researcher’s knowledge scheme can influence the way questions are asked (so influencing the answers) and, above all, the way data are analyzed. As already mentioned, there exist some techniques that try to reduce the subjectivity, but they introduce other sources of error [29]. For example, by providing an ex-ante defined list of possible constructs and concepts (though in some cases they can be extended by respondents) the answer possibility of the respondents is limited and can be biased. Based on our previous experience, we have decided to adopt semi-structured interviews so trying to minimize the effects of biases. Despite the drawbacks, we argue that causal maps can be effectively used to characterize the interpretative schemes of the FLOSS team members as well to assess if such schemes are shared and how they affect work practices.

6.2. Limitations

Structure of the causal maps is influenced by the way the interview was structured. It was difficult to conduct semi-structured interviews while also ensuring respondents talked of the same items.

7. Conclusions

We have presented a preliminary analysis of the degree of sharing of mental models among developers in a FLOSS development project. Our analysis suggests that there is a high-level of sharing among core developers but the sharing is not complete, with some differences related to tenure in the project. Our future work will extend these results in several directions:

1. Include more developers from the Lucene Java project, including those with different degrees of participation in the project in order to assess the degree of sharing and how the sharing relates to individual characteristics.
2. Include developers from other projects. We have completed interviews with developers from several other Apache projects that we will analyze to assess the degree of sharing between different projects. We anticipate finding certain concepts in common but others that are unique to the particular project. We would also like to add non-Apache projects to see the influence of the Apache Software Foundation on sharing of mental models.
3. Include developers from less successful projects. The projects selected so far have all been rather successful. We would like to interview developers of a failing project to determine if there is a relation between the degree of shared mental models and project effectiveness.

References

[1] B. A. Nejmeh, "Internet: A strategic tool for the software enterprise," *Communications of the ACM*, vol. 37, pp. 23–27, 1994.

[2] M. B. Watson-Manheim, K. M. Chudoba, and K. Crowston, "Discontinuities and continuities: A new way to understand virtual work," *Information, Technology and People*, vol. 15, pp. 191–209, 2002.

[3] P. C. van Fenema, "Coordination and control of globally distributed software projects," in *Erasmus Research Institute of Management*. Rotterdam, The Netherlands: Erasmus University, 2002, pp. 572.

[4] P. S. de Souza, "Asynchronous Organizations for Multi-Algorithm Problems," in *Department of Electrical and Computer Engineering,* : Carnegie-Mellon University, 1993.

[5] D. J. Armstrong and P. Cole, "Managing distance and differences in geographically distributed work groups," in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 167–186.

[6] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Communications of the ACM*, vol. 31, pp. 1268–1287, 1988.

[7] S. Sawyer and P. J. Guinan, "Software development: Processes and performance," *IBM Systems Journal*, vol. 37, pp. 552–568, 1998.

[8] F. P. Brooks, Jr., *The Mythical Man-month: Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1975.

[9] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "An empirical study of global software development: Distance and speed," presented at Proceedings of the International Conference on Software Engineering (ICSE 2001), Toronto, Canada, 2001.

[10] B. Butler, L. Sproull, S. Kiesler, and R. Kraut, "Community effort in online groups: Who does the work and why?," in *Leadership at a Distance*, S. Weisband and L. Atwater, Eds. Mahwah, NJ: Lawrence Erlbaum, 2002.

[11] J. A. Cannon-Bowers and E. Salas, "Shared mental models in expert decision making," in *Individual and Group Decision Making*, N. J. Castellan, Ed. Hillsdale, NJ: Lawrence Erlbaum Associates, 1993, pp. 221-246.

[12] A. Giddens, *The Constitution of Society: Outline of the Theory of Structuration*. Berkeley: University of California, 1984.

[13] S. R. Barley and P. S. Tolbert, "Institutionalization and structuration: Studying the links between action and institution," *Organization Studies*, vol. 18, pp. 93–117, 1997.

[14] W. J. Orlikowski, "Using technology and constituting structures: A practice lens for studying technology in organizations," *Organization Science*, vol. 11, pp. 404-428, 2000.

[15] E. W. Stein and B. Vandenbosch, "Organizational learning during advanced system development: Opportunities and obstacles," *Journal of Management Information Systems*, vol. 13, pp. 115–136, 1996.

[16] J. Sutanto, A. Kankanhalli, and B. C. Y. Tan, "Task coordination in global virtual teams," presented at Twenty-Fifth International Conference on Information Systems, Washington, DC, 2004.

[17] J. A. Espinosa, F. J. Lerch, and R. E. Kraut, "Explicit versus implicit coordination mechanisms and task dependencies: One size does not fit all," in *Team cognition: Understanding the factors that drive process and performance*, E. Salas and S. M. Fiore, Eds. Washington, DC: APA, 2004, pp. 107-129.

[18] K. Crowston and E. Kammerer, "Coordination and collective mind in software requirements development," *IBM Systems Journal*, vol. 37, pp. 227–245, 1998.

[19] B. Curtis, D. Walz, and J. J. Elam, "Studying the process of software design teams," in *Proceedings of the 5th International Software Process Workshop On Experience With Software Process Models*. Kennebunkport, Maine, United States, 1990, pp. 52–53.

[20] L. Gasser and G. Ripoche, "Distributed Collective Practices and F/OSS Problem Management: Perspective and Methods," presented at Conference on Cooperation, Innovation & Technologie (CITE2003), University de Technologie de Troyes, France,, 2003.

[21] M. A. Rossi, "Decoding the "Free/Open Source (F/OSS) Software Puzzle": A survey of theoretical and empirical contributions," Università degli Studi di Siena, Dipartimento Di Economia Politica, Working paper 424, 2004.

[22] S. Sawyer, "A Social Analysis of Software Development Teams: Three Models and their Differences," presented at The 2000 Americas Conference on Information Systems (AMCIS 2000), 2000.

[23] K. J. Stewart and S. Gosain, "Impacts of ideology, trust, and communication on effectivness in open source software development teams," presented at Twenty-Second International Conference on Information Systems, New Orleans, LA, 2001.

[24] N. Jørgensen, "Putting it all in the trunk: incremental software development in the FreeBSD open source project," *Information Systems Journal*, vol. 11, pp. 321–336, 2001.

[25] E. S. Raymond, "Homesteading the noosphere," *First Monday*, vol. 3, 1998.

[26] M. Pidd, *Tools for thinking modeling management science*. Chichester: John Wiley and Sons, 1996.

[27] M. T. Wrightson, "The documentary coding method," in *Structure of Decision*, R. Axelrod, Ed. Princeton: Princeton, NJ, 1976, pp. 291–332.

[28] P. Cossette and M. Audet, "Mapping of an idiosyncratic schema," *Journal of Management Studies*, vol. 29, pp. 325–347, 1992.

[29] L. Markoczy and J. Goldberg, "A method of eliciting and comparing causal maps," *Journal of Management*, vol. 21, pp. 305–333, 1995.

[30] H. J. Harrington, *Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness*. New York: McGraw-Hill, 1991.

[31] V. Grover and W. J. Kettinger, "Business Process Change: Concepts, Methodologies and Technologies." Harrisburg: Idea Group, 1995.

Table 1. Concepts and shared concepts in mental models.

# of identified elements / Elements	A	B	C	D
Definition concepts	40	34	37	10
Definition concepts repeated	It is written in Java (3); [Goal is to] provide search functionality (3); [Users are] people who want to add search functionality into their applications (3)			
Causal cognitive map concepts	119	153	72	63
Processes	3	7	3	2
Processes repeated	Process Initiation (4); member selection (4), bug-fixing-feature adding (2); release kick off (2).			
Unwritten rule (norms) concepts	5	14	5	4
Unwritten rule concepts repeated	Community is important (3); Communication is done on public mailing list (3); [New member selection criteria] (3) No formal goals, yet same direction (2); not much planning (2)			
Written rule concepts	2	4	3	3
Written rule concepts repeated	Voting for new member selection (4); Accepting patches (3); PMC Role (2); no formal roles (2)			

Table 2. Concepts per definitions identified

Identified # of elements / Definition Elements	A	B	C	D
Project description/characteristics	7	4	6	1
Project description/characteristics repeated	Written in Java (3); search library (2); easy to use (2)			
Project goals	3	2	2	2
Project goals repeated	Provide search functionality (3); there aren't specifically stated goals, yet we're driving in the same direction (2)			
Intended users	2	1	1	1
Intended users repeated	People who want to add search functionality into applications (3); developers (2)			
Project success and strengths	5	10	7	4
Project success and strengths repeated	Number of users (2), companies using it (2); community (2); developers who support it (2); interest at the Apache Conference (2); software performance (2)			
Problems faced by the project	2	2	2	1
Problems faced by the project repeated	There aren't big problems (3)			
Challenges faced by new members	1	1	4	0
Challenges faced by new members repeated	None			
Skills/knowledge needed for development of Lucene Java	2	6	3	1
Skills/knowledge repeated	Information search/retrieval (2); java (2)			
Roles	8	6	10	N/A
Roles repeated	Specialists (2); active committers (2); new committers (2); people who support users by answering emails (2)			
Leadership	10	2	2	N/A
Leadership repeated	The project founder will always be the leader (2); Yannik is a leader (2); leadership equals sustained contribution (2)			

Table 3. Central concepts per map.

Map	Concepts (domain scores)	Summary of Concept Areas
A	Erik is good for a lot of projects (4) [another project leader] is important to get others to use Lucene (4) [New members from IBM] can handle lot of these things (4) They know how to fit in (3) Somebody ends up	Boundary Spanning Selection of & contribution

Map	Concepts (domain scores)	Summary of Concept Areas
	nominating them (3); I have slacked up (3) You want to work on a project (3)	by new members. Motivation for project
B	One of the most successful open source projects there is (19); there's one of two ways to select new members (5)	Project success New member selection
C	And then they commit it (3); I don't see any problems (2); It is very successful currently (2); Main focus should always be on simplicity (2); That's why I learnt Open Source (2); You can trust (2); So that the committers have a good feeling that the code is good and it's robust (2); Sometimes people of the other [sub]projects get involved in discussions and ask us to implement new features in a way to keep files and docs compatible (2); It's very simple to get basic search working but it also offers more sophisticated stuff for who are familiar with info retrieval and search (2)	Project success, strengths and challenges Process for committing code Motivation for project
D	I would say [the project is a successful one] (2); There is a great number of people involved in the community (2); It either requires you to go through lots of learning to get to a level where you are actually not able to improve it (2); So, it is kind of complex to start (2); The barrier to enter is kind of low (2); It is not a project that you can just jump in and do all the hard stuff (2); Contribution to positive group atmosphere, resolving conflicts, things like that (2); There are so many ways you can contribute (2); So even if you're very entry-level Lucene guy, you can contribute by helping others (2)	Project success & strengths Community & contribution Challenges for new members Group Maintenance

Table 4. Number of concepts for each set per map.

# of concepts / Sets	A	B	C	D
Challenges	1	6	10	4
Change in Project	6	2	4	3
Community	55	23	7	7
Coordination	4	5	9	2
Goals	5	2	4	1
History	7	3	2	3
Leadership	16	2	3	3
Membership	20	19	9	3
Success/Strengths	2	29	17	3

Table 5. Activities per processes described.

# of activities / Processes	A	B	C	D
Initiation	4	4	3	4
Rulemaking	-	8	-	-
Bug fixing- Feature adding	-	10	3	-
Member Selection	6	5	4	2
Planning	-	4	-	-
Release kick off	2	3	-	-
Apache Incubation Process	-	4	-	-