Organisational Knowledge Creation in Open Source Communities

JOUNI JUNTUNEN
Oulu University of Applied Sciences, Information Technology

TEPPO RÄISÄNEN
Oulu University of Applied Sciences, Information Technology

Abstract: Open source based development has become increasingly popular way to make software. The use of open source has changed the principles of software engineering and industry. A theory of organisational knowledge creation describes how knowledge is created dynamically in organisations. The three layer model consists of SECI (socialisation, externalisation, combination, internalisation) model, ba and knowledge assets. The purpose of the study is to evaluate, how the dynamic theory of knowledge creation can be applied to open source software projects as an organisation in order to explain how knowledge is created in open source communities.

Keywords: Open source, knowledge-creation, SECI, software engineering.

1 Introduction

Open source based development has become increasingly popular way to make software. The use of open source has changed the principles of software engineering and industry. Developing Linux operating system, for example, is the largest project making use of collaboration in the history of computer science. Since 2005 there has been about 8000 developers creating about 15 million lines of code for Linux 35. Year by year more people are working with open source software and people use open source software even that they do not know or think of it.

Famous open source software are, for example, Linux and Android operating systems, LibreOffice office suite, Mozilla Firefox web browser, Apache HTTP server, MySQL database server and Moodle learning environment. Major part of the internet is powered by Linux operating system or uses other open source software 1736. Several social media services, such as Facebook, Twitter and Youtube, are based on open source software 35.

European commission has updated their strategy for internal use of open source software addressing that open source software should be assessed on an equal basis when purchasing new software solutions 7. On national level, for example, Finnish public organisations are advised to take open source into account on software acquisitions 12. Already in year 2007 22 billion euros were invested in open source software which is about one fifth of the total software investments in Europe 33. Open source and data are seen one of the key technologies increasing the growth of European economy in the future.

The key factor of open source software is the openness of source code and collaboration. Ingo describes that unlike many commercial and proprietary software the source code of open source software is available to all and without limitations. By doing this open source communities are turning the practices of the IT industry upside-down 11. Scharff 32 sees the availa-

bility of source code as a technical core of open source project whereas the community of sharing participants forms the social core.

The use of open source has also changed the principles of software development. More and more today's software is being developed using agile methods instead of making in depth specifications. Development is carried out in communities that utilise various communication channels like mailing lists, forums and tools of real-time communication 8. Linus Torvalds, the creator of Linux operating system, was the first person who learned how to play with the new rules that Internet access made possible 29.

Open source communities have been able to produce quality software neglecting methodology suggested by the information processing science 13. Quality of the open source software is seen equal or even better compared to proprietary software 4. Open source communities are also able to produce software in productive and efficient manner. For instance new version of Linux kernel is released approximately in every two or three months making the development process much faster compared to other proprietary operating system 35.

Open source project and community can be seen as an organisation, where theory of organisational knowledge creation can be applied to study how knowledge is created and shared. Dynamic process of creating knowledge consists of three layers interacting to together. The three layers are SECI (socialisation, externalisation, combination, internalisation) model, ba and knowledge assets 1820.

SECI is a spiral model describing the conversion between tacit and explicit knowledge. Tacit knowledge is highly personal form of knowledge and it is hard to express formally. Explicit knowledge can be expressed and documented formally and systematically. SECI identifies four modes of knowledge conversion between tacit and explicit: socialisation (tacit to tacit), externalisation (tacit to explicit), combination (explicit to explicit) and internalisation (explicit to tacit) 1821.

Ba can be understood as platforms or foundation for knowledge creation. It can be also defined as a context, where knowledge is created, shared and utilised. Ba does not mean necessarily physical place, it can also mean virtual or mental place, shared experiences ideas or ideals or any combination of these kinds of spaces 1821.

Inputs and outputs of the knowledge-creation process are the assets forming the basis for organisational knowledge creation and have also influence on how ba functions as a platform in the process. Knowledge assets can be categorised into four types: experimental knowledge assets, conceptual knowledge assets, systemic knowledge assets and routine knowledge assets 21.

By studying the knowledge-creation in open source communities it is possible to improve software development processes and methodologies in general. Principles of open source development could be also used in higher education while learning software engineering.

Rest of the paper is organised as follows. In section 2 open source software development is explained. Section 3 presents the knowledge management, ba and knowledge assets. Section 4 discusses open source software communities and the different aspects of knowledge creation. Finally section 5 concludes the paper.

2 Open source

2.1 Free and open source software

The roots of open source software can be traced to the hacker culture of 1970's where Richard Stallman was the key person. In 1984 he started the development of free operating system called GNU devoting his life to developing software in an open process 11. For supporting this development Stallman published the terms of distribution that gave users a permission to use, copy, modify and distribute the operating system freely. The only remarkable restriction was that it was not allowed to make one's own restrictions to the software. These rules were included in the GNU Manifesto originally written in 1983 [10].

Free software is defined by Richard Stallman as follows 9:

"*The users have the freedom to copy, distribute, study, change and improve the software. Thus, "free software" is a matter of liberty, not price*."

The free software definition describes the four essential freedoms for software users: 9

- The freedom to run the program as you wish, for any purpose (freedom 0).
- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbour (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

In 1997 Eric Raymond published an essay called "The Cathedral and the Bazaar" 29 where he analyses the hacker community and the principles of free software. The essay got a lot of publicity and in 1998 Netscape Communications, for example, decided to open the source code of their Netscape Communicator product. At the same time Raymond and Bruce Perens, the former Debian leader, founded an Open Source Initiative (OSI) that nowadays certifies open source licenses and maintains the Open Source Definition (OSD) 22.

The term "open source" was born in 1998. The people of OSI felt that the word "free" is problematic. "Free" was seen as a confusing word that is easily misunderstood. They also felt that "free" was ignored by corporate world and business people will never buy "free software" 8.

OSI defines open source software in short as follows 23:

"Open source software is software that can be freely used, changed, and shared (in modified on unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with open source definition".

OSI has released ten-point criteria for software and license it has to comply. Open source software is distributed under license approved by OSI and at the moment there are 71 different open source licenses listed in their home pages 23.

As a result there are two political camps or movements in free software community. Free software is more concerned about freedom to use software whereas open source software is more focused on practical issues and licenses 3334. Although free software and open source software are partially released under same licenses, open source initiative allows more permissive licenses restricting the freedom of usage according to Free Software Foundation 34. Stallman recommends to use term Free/Libre and Open Source Software (FLOSS) instead of Free and Open Source Software (FOSS) to explicitly avoid the preference between free soft-

ware and open source software when referring to both movements of free software community 33.

FLOSS is often considered as free or non-commercial software which is not true. Open source licenses do not restrict commercial usage of the software or source code as long as terms of used license are met. On the other hand closed-source software can be non-commercial, for example, a freeware application.

2.2 Software development process

Open source software project may last for years. Over time, new features are implemented and detected bugs are fixed or software is updated to as new technologies and tools emerge. Process can be seen as a continuous cycle repeating problem identification, finding volunteers, solution identification, implementation and review. New version of developed software is released periodically, see Figure 1.

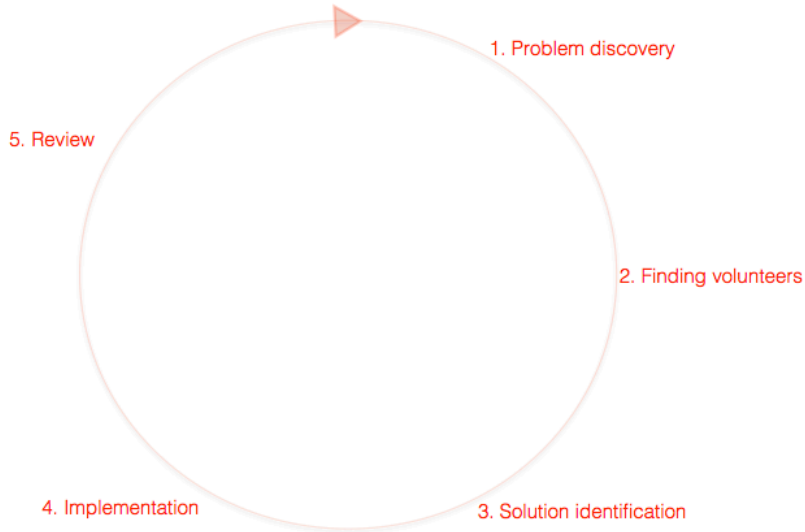

Fig. 1 Open source development cycle

Iteration starts with problem identification (phase 1 in Figure 1). For example, someone in a community detects a bug or has an idea for a new feature. Development task is published usually through issue list where volunteer(s) may pick it up and start developing (phase 2). Depending on the development task implementation may require work from multiple members of the community and sharing knowledge as well as studying documentation, software under development and other sources from internet. Usually the task descriptions on the issue list are not accurate and developer has free hands to come up with the solution (phase 3).

Any development task (phase 4), like coding a piece of software or doing documentation, translation or virtually any kind of work that contributes to the open source project and community is called contribution. Contributions are reviewed by assigned members of the community or manager of the project (phase 5). Depending on the product and project review process is taking time and has formal procedures 215. As a result of review contributions might be accepted, accepted after corrections or rejected. A good practice is to give feedback to contributors if project is willing to have them contributing also in the future 15.

Quality assurance is an essential part of any software development. Open source projects do not necessarily rely on formal testing 29. Some of the projects are relying on formal testing. Automated tests are used as part of the development process and can be used, for exam-

ple, when evaluating contributions 515. Some projects even apply test driven development (TDD) to in to their development process 14.

In open source software projects, it is important that the software is released early and often. For that project should establish release management process. It is important to notice that the release does not have to be perfect – it might include bugs or missing features. Release is planned and documented using issue tracker and possible roadmap is updated to contain issues for the next release. Communication with the community and handing out information to end-users is important, so people are able to know what to expect from the next release 25.

Project has to be planned carefully ahead. Feature requests have to be evaluated and scheduled for development. Roadmap might be used to describe development status and future development work 11. There is no specific format for roadmap. Sometimes roadmap describes only the time of next release or the release interval. WordPress, for example, has a dynamic roadmap so that release interval is 3 – 4 months with the features that are primarily driven by ideas voted on by users of WordPress 37. Sometimes roadmap is very detailed and more static like in the case of Mozilla Firefox 16.

2.3 Community

There are various reasons why open source products are developed. Commercial software is not adequate enough for some reason or there might be personal need or ambition to develop a software product. Sometimes there is no suitable software on the market and one has to be developed. There may be needs for faster development or long-term stability 28. At the beginning of the project software might be developed for personal use and later on when other parties take interest community starts expanding. Many projects have commercial significance and there may be commercial distributions or services related to product.

Product should be useful to developers and end-users. It should also be interesting enough people to participate on development process 32. There might be other similar products available and therefore project has to stand out from the mass. Software to be developed might compete with proprietary alternative and goal for the open source community is to make open source version of the software better and more attractive to end-users.

People tend to do what they feel is fun. Open source project should somehow motivate people to participate 29. Open source participants engage in personally meaningful activities 32. Participating in well-known open source project may provide a certain status in a hacker culture 3031. Active members of community can be also awarded by organisations such as Microsoft's Most Valued Professional (MVP) programme.

The development is carried out in collaboration within community. Like in the commercial software development there are several roles in open source communities. In most cases the majority of people are software developers, mainly programmers. If the product is well-known it is likely that it has been localised to many languages and the number of people who are doing localising is high because typically one person is doing localisation for only language. Moreover open source projects need, for example, testers, bug fixers, art workers, documentation writers and people who recommend the project to others 26. Important is to have somebody to steer and organise the project at all times. Also end-users have to be taken care of by providing necessary information and support.

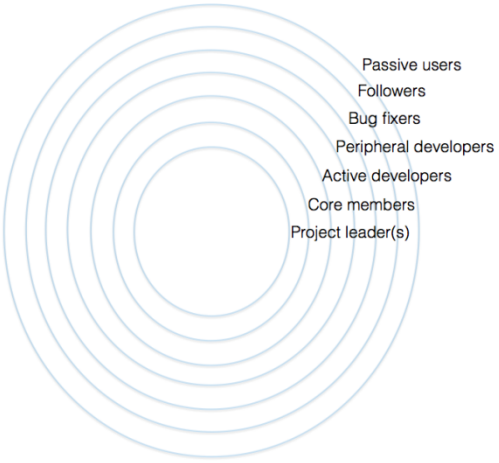

Fig. 2 Open source project member roles

Community of open source project can be organised in different ways (see e.g. Figure 2). Governance models and decision making vary from dictatorship to democracy and from centralised control to decentralised control depending on the objectives of the community. If centralised control is used there is one project leader or core team instead of one leader. The objective of centralised community is either to share innovation and knowledge or to provide stable service. In the case of decentralised control the objective is to satisfy individual needs and the structure of the community is based on many peripheral developers. Meritocracy is a decision-making model where contributors who have earned the respect of the community through frequent and useful contributions have a 'louder voice' or a veto in order to break deadlocks in decision-making 24.

2.4 Environment

Communication is important part of open source community to work efficiently. Tools like mailing lists, chats, FAQs, forums, wikis, social media or syndication tools help community to collaborate and are capturing project memory for later use. Proper tools should be chosen for the communication between developers and for the communication between developers and end-users.

Source code for the open source project is available for everybody through repository of version control system. If centralised revision control is used only some participants of the community usually have write access to version control system. Patches are used to supervise contributions in orderly fashion. Patch is a piece of software which can be easily merged to existing software. Some consideration should be taken when submitting a patch. To make merging easier patch should contain some limited new features or enhancements to software.

Commonly used version control systems are SVN, CVS, Git, Bazaar and Mercurial. Version control system enables parallel software development and efficient source code management through revisions, branching, merging and tagging. Codebase can be centralised or Distributed Revision Control (DRCS) may be used. Centralises management has canonical reference copy of the developed software and the source code. Contributions are merged into that using commits or patches. DRCS conducts synchronisation by exchanging patches from peer to peer.

Documentation is important in terms of project memory as people participating in project may change over time. Material may be targeted for developers, maintainers, instructors or

end-users. One of the most important tools for documentation and organising development tasks is issue tracker. Tracker might include feature requests, tasks and bug reports.

3 Knowledge-creation in organisations

Nonaka 18 (following the work of Polanyi 27) explicated two dimensions of knowledge in organisations: explicit and tacit. Explicit or codified knowledge refers to knowledge that is transmittable in formal, systematic language. Explicit knowledge can be "articulated, codified, and communicated in symbolic form and/or natural language" 1 and indeed, many times it is captured in the "records of the past such as libraries, archives, and databases" 18. Procedure manuals, product literature, and even computer software can be seen as explicit knowledge 19. New knowledge emerges through the interaction of tacit and explicit knowledge 18.

3.1 SECI-model

SECI-model 18 explains how new knowledge emerges within organisations. Knowledge creation is a "spiral" that takes place in two dimensions. These dimensions are the epistemological and ontological dimensions.

The epistemological dimension is the distinction between tacit and explicit knowledge. In the ontological dimension are the levels of knowledge creation entities. These are individual, group, organisation and inter-organisation. Strictly speaking all knowledge is created by individuals. Thus, "organisational knowledge creation should be understood as a process that 'organisationally' amplifies the knowledge created by individuals" 19. In essense "human knowledge is created and expanded through social interaction between tacit knowledge and explicit knowledge" 19.

The dynamic theory of organisational knowledge creation has four modes of knowledge conversions that are created when tacit and explicit knowledge interact. The modes are 18: 1) socialisation, 2) externalisation, 3) combination, and 4) internalisation.

Socialisation is a process of sharing experiences 18. It creates new tacit knowledge from existing tacit knowledge. For example by observing a colleague the observer can learn through imitation or practice. Typically the new tacit knowledge is in a form of shared mental models or technical skills.

Externalisation is a process of articulating tacit knowledge into explicit concepts 18. Externalisation is the key process in the theory as it is the process that creates new explicit concepts from the tacit knowledge. One example of this is writing. It can be seen as an act of converting tacit knowledge into articulable knowledge. The use of metaphors and analogies seem to be a key in externalisation as it is typically triggered by dialogue or collective reflection.

Combination is a process of systemising concepts into a knowledge system 18. It creates new explicit knowledge from existing explicit knowledge. It is the kind of knowledge creation that happens in formal education or training at schools. Also the use of large-scale databases could be seen as an example of combination.

Internalisation is a process of embodying explicit knowledge into tacit knowledge 18. Reading documentations or watching videos is an example of the kind of "re-experiencing" that internalisation requires. Also "learning by doing" can be seen as an example of internalisation.

On their own these knowledge conversion modes produce only a limited amount of knowledge creation. They must form a dynamic and continuous knowledge spiral for knowledge creation to truly happen. Typically this spiral starts at the individual level and moves up on the ontological dimension (i.e. from individual to group, from group to organisational, and from organisational to inter-organisational level).

3.2 Ba

Knowledge is created in context. For example social, cultural and historical context provide individuals the background in which information is interpreted to create meanings. Nonaka et al. 21 call the context where knowledge is create ba. They define ba as a shared context in which knowledge is shared, created and utilised. In addition ba can be divided into two dimensions of interaction. The first dimension is the type of interaction, whether it takes place *individually* or *collectively*. The other dimension is the media used in the interaction. This can be *face-to-face* or *virtual*. These interactions enable four types of ba. They are originating ba, dialoguing ba, systemising ba and exercising ba 21.

Originating ba refers to individuals and face-to-face interactions. Thus, it is the context for socialisation. Through face-to-face interaction and sharing of experiences originating ba fosters the emergence of care, love, trust and commitment 21.

Dialoguing ba is defined by collective and face-to-face interactions 21. In it people share their mental models skills. Thus, it is the context for externalisation.

Systemising ba is the place for collective and virtual interactions. It is mainly the context for externalisation 21. Today mobile tools and social media offer organisations great support for externalizing knowledge in various forms (e.g. text, picture and video).

Exercising ba defined by individual and virtual interactions 21. It is the context for internalisation. The vast amounts of information combined through the hypertext capabilities (i.e. web-links) of HTML provide the basis for exercising ba. In addition search tools and databases offer individuals new ways to find relevant bits of information 21.

Through the evolution of ICT it is the virtual interactions that are enhanced constantly. Technologies such as 3D-internet allow users to interact in new and novel ways. These technological advances can also be utilised in open source communities.

3.3 Knowledge assets

In addition to the SECI-model and ba, knowledge assets can be used to understand how organisations create knowledge. Knowledge assets can be defined as "firm-specific resources that are indispensable to create values for the firm" 21. If Ba is the place where knowledge creation happens, the knowledge assets are the inputs, outputs and moderating factors of the knowledge creation process. According to Nonaka et al. 21, there are four categories of such assets: 1) experiential knowledge assets, 2) conceptual knowledge assets, 3) routine knowledge assets, and 4) systemic knowledge assets. These are described more closely in Table 1.

Tab. 1 Knowledge assets

Knowledge asset	Description	Examples
Experiential knowledge assets	Tacit knowledge shared through common experiences	Skills and know-how of individuals

		Care, love, trust, security, passion
Conceptual knowledge assets	Explicit knowledge articulated through images, symbols and language	Product concepts, design, brand equity
Systemic knowledge assets	Systemised and packaged explicit knowledge	Documents, specifications, manuals
Routine knowledge assets	Tacit knowledge routinised and embedded in action and practices	Know-how in daily operations Organisational routines and culture

In the next chapter, the SECI-model, ba and the knowledge assets are discussed together with the open source communities. The question is does participation in the open source projects help community members generate new knowledge

4 Knowledge creation in open source communities

In context of open source tacit knowledge can be understood as personal skills or know-how related to software engineering and software itself being developed by the community. Besides that, there are practises, opinions, attitudes or even idealogical or political influence in the background. Explicit knowledge is known, published knowledge and information related to software engineering and the software under development. Since communities work through internet, knowledge is shared mostly virtually using modern online communication and collaboration tools.

Figure 3 illustrates the four states of SECI-model and open source development process, activities, collaboration tools and communication channels. Actions (discussion, studying, developing, contribution) are enabling knowledge to flow from state to the next one.

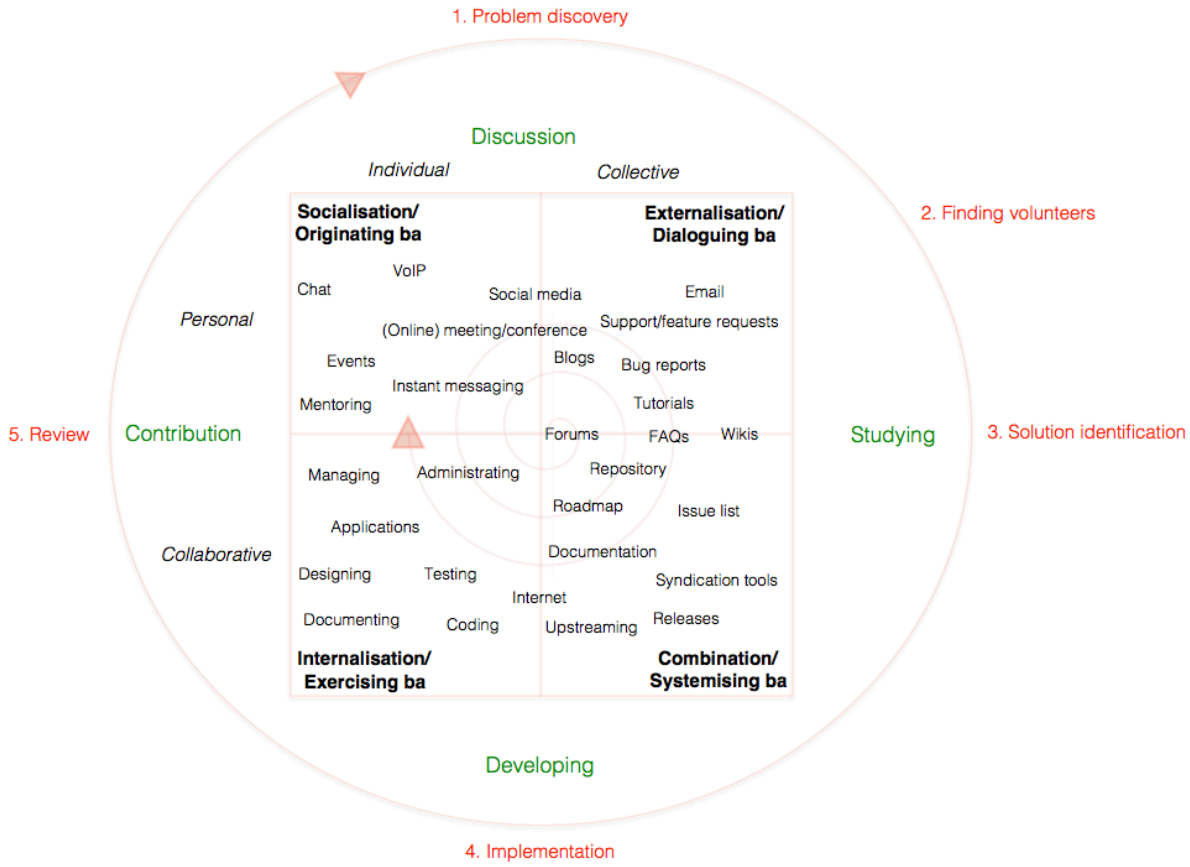

Fig. 3 SECI-model and open source development process

Knowledge creation starts with discussion including socialisation and externalisation. Issues related to developed software, technologies and development methodologies and other related issues are discussed. Online messaging and communication tools are used to share tacit knowledge. Based on discussion and personal findings new development tasks, like bug fixes, new features or technological updates, are decided to be taken under development (problem identification). Development tasks are recorded on issue list and suitable workforce is assigned to the task (finding volunteers).

Creating a solution requires studying of existing explicit knowledge. By studying explicit and combined knowledge solution for the development task is designed (solution identification). Both internal knowledge from the community and external knowledge from internet or syndication tools are used.

Explicit knowledge is absorbed while developing a contribution for the project (implementation). Contribution might be design, development, testing, using the developed application, documenting, doing translations or some other work that benefits the project. During the development new tacit knowledge may be created and made explicit later on.

Contribution includes tacit and explicit knowledge turned into practical solution for the development task. Review process varies but usually project leader or senior member of the community is reviewing the contribution and deciding whether it is accepted or not and some projects have practise to give feedback to contributors (review). As software evolves it stimulates to new discussions and knowledge-creation generating new development tasks.

Table 2 describes how different aspect of SECI model, ba's and knowledge assets are occurring in open source project and community.

Tab. 2 SECI, ba and knowledge assets in open source projects

Aspect	How occurs in open source project and community
Socialisation	Sharing tacit knowledge using online (instant) messaging tools and virtual or contact meetings. Mentoring is also used by some communities to coach and encourage new members to take part of the development. Special events, like Hatkathons, may be organised to promote socialisation and learning from each other. Extreme programming techniques, like pair programming, are also used to support learning and knowledge sharing. Project leader(s) and core members of the community can be seen as opinion leaders leading the process of knowledge creation.
Externalisation	Tacit knowledge is made explicit using various online communications tools. Tools are preserving and encapsulating knowledge for later use and enabling dialog within community. In addition to written media, online videos and other digital material may be also used. In open source project end-users should be treated as co-developers and valuable tacit knowledge may be gained from them by getting support of feature requests.
Combination	Explicit knowledge is combined using collaboration tools such as code repository and version control, online documenting tools and syndications tools. Knowledge is combined both from internal and external resources in structured form to preserve project memory. If project is making use of other open source software, modifications may be upstreamed back to original project.
Internalisation	Explicit knowledge is absorbed to tacit while contributing to the open source project, utilising the developed software or doing supportive and administrative tasks. Various applications and internet are used as a tool to get explicit knowledge and during the development process some of the knowledge becomes tacit.
Originating (messaging) ba	Operation of open source project and sharing knowledge is based on trust, respect and commitment to the project. Created between individuals in personal manner.
Dialoguing (communicating) ba	Different type of online communication tools enabling dialog between member of the community. Personal messages and other knowledge is published for collective use.
Systemising (collaborating) ba	Collaboration tools to create platform to preserve explicit knowledge for collective use. Knowledge is produced in collaboration.
Exercising (workspace) ba	Environment and tools creating a workspace that is used to contribute. Contribution may be done individually or collaboratively. Even though the level of collaboration varies, contribution is reviewed and utilised by other members of the community.
Experiential knowledge assets	Skills and know-how of community members. Trust and respect between community members. Commitment to the project. Innovativeness of the community. Size of the community (members) and amount of end-users.

Conceptual knowledge assets	Problems and new ideas based on discussions and tacit knowledge published using various communication tools. Brand equity, attraction and reputation of developed software.
Systemic knowledge assets	Code repository, releases (published versions of software), documentation, other online material containing systemised and packaged explicit knowledge created within community.
Routine knowledge assets	Practises and guidelines, development process, culture and routines used by the community.

5 Conclusions

In this paper we have studied how the dynamic theory of knowledge creation can be applied to explain how knowledge is created in open source communities. The three layers of theory seem to be adequate to describe and explain knowledge-creation.

Original SECI model emphasises face-to-face communication in socialisation. Due technical development modern online tools, such as instant messaging tools and social media, are replacing face-to-face communication and even changing the way people communicate nowadays. Open source communities are operating using internet and naturally most of the communication and socialisation is done virtually instead of face-to-face contact.

Figure 3 illustrates communication channels that are used by open source communities. Some of the channels are hard to categorise according to SECI-model. For example, social media, like Facebook, has features that are suitable for socialisation as well as publishing knowledge in more formal manner. Therefore it could be used in socialisation or externalisation depending how it is used.

Overall working in open source communities seems to be good way for knowledge-creation. SECI-modes and open source development process seem to link well together as open source community members socialise, share knowledge and work together to make a contribution to the open source project. In addition, the open source project provides the shared context (ba) for the users allowing community members to utilise their various knowledge assets to the benefit of the open source project.

Importance of open source software development both in economical and methodological sense is growing. By studying knowledge creation in open source communities processes of software engineering can be improved. Also higher education and learning processes can be developed to take processes and practises of open source communities into account.

One limitation of the study is that it is purely conceptual. We did not perform interviews or collect any data from open source communities. As such we need more research on how open source communities foster knowledge creation. As a future research this is natural follow-up study. Also open source communities provide good opportunity to perform ethnographic studies as we could take part in development of open source software as community members. This could provide new insights in how knowledge is created by the community. For example, maybe the role of certain community members is emphasised in the knowledge creation process.

Literature

1. Alavi M & Leidner D.E. 2001. Review: Knowledge management and Knowledge Management Systems: Conceptual Frameworks and Research Issues. *MIS Quarterly* 25(1): 107–136.
2. Android open source project. 2012. *Welcome to Android*. [online]. Available at: <http://source.android.com/index.html> [Accessed 14 September 2015].
3. Chatti, M., Klamma, R., Jarke, M. and Naeve, A., 2007. *The Web 2.0 Driven SECI Model Based Learning Process.* Available at: <http://dbis.rwth-aachen.de/cms/publications/publication-684> [Accessed 31 August 2015].
4. Coverity inc. 2015. *Coverity Scan Report Finds Open Source Software Quality Outpaces Proprietary Code for the First Time*. [online]. Available at: <http://www.coverity.com/press-releases/coverity-scan-report-finds-open-source-software-quality-outpaces-proprietary-code-for-the-first-time/> [Accessed 14 September 2015].
5. Dokeos. 2012. *Dokeos*. [online]. Available at: <http://www.dokeos.com> [Accessed 15 September 2015].
6. European Commission. 2007. *The impact of Free/Libre/Open Source Software on innovation and competitiveness of the European Union.* [online]. Available at: <http://flossimpact.eu> [Accessed 15 September 2015].
7. European Commission. 2015. *Open Source Strategy in the European Commission.* Available at: <http://ec.europa.eu/dgs/informatics/oss_tech/index_en.htm> [Accessed 14 September 2015].
8. Fogel, K., 2010. *Producing Open Source Software.* [online] Available at: <http://producingoss.com/en/index.html> [Accessed 9 September 2015].
9. Free Software Foundation. 2015. *What is free software? The free software definition.* [online]. Available at: <http://www.gnu.org/philosophy/free-sw.html> [Accessed 9 September 2015].
10. GNU Manifesto. 2015. [online]. Available at: <http://www.gnu.org/gnu/manifesto.html> [Accessed 9 September 2015].
11. Ingo, H. 2006. *Open Life. The Philosophy of Open source*. [online]. Available at: <http://www.openlife.cc/files/OpenLife-aa.pdf> [Accessed 9 September 2015]
12. JUHTA - Julkisen hallinnon tietohallinnon neuvottelukunta. 2012. *JHS 169 Avoimen lähdekoodin ohjelmien käyttö julkisessa hallinnossa*. [online]. Available at: <http://docs.jhs-suositukset.fi/jhs-suositukset/JHS169/JHS169.html#H24> [Accessed 14 September 2015].
13. Luoma, I. 2006. *Software Engineering in Open Source Software. In Essays of OSS Practises and Sustainability.* <http://www.coss.fi/sites/default/files/documents/presentations>.
14. Martin, R.C. 2007. *Professionalism and test-driven development.* IEEE Software. May/June 2007. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumb er=4163026>.
15. Mozilla Developer Network. 2012. *Developer Guide*. [online]. Available at: <https://developer.mozilla.org/En/Developer_Guide> [Accessed 14 September 2015].
16. MozillaWiki. 2012. Firefox/Roadmap. <https://wiki.mozilla.org/Firefox/Roadmap>.
17. Netcraft. 2015. *January 2015 Web Server Survey*. [online] Available at: <http://news.netcraft.com/archives/2015/01/15/january-2015-web-server-survey.html> [Accessed 9 September 2015].
18. Nonaka I. 1994. A Dynamic Theory of Organizational Knowledge Creation. *Organizational Science* 5(1): 14-37.

19. Nonaka I, & Takeuchi H. 1995. *The knowledge-creating company*. New York: Oxford University Press.
20. Nonaka, I, Toyama, R. and Byosiére, P. 1998. *A Theory of Organizational Knowledge Creation: Understanding the Dynamic Process of Creating Knowledge*. [online]. Available at: <https://ai.wu.ac.at/~kaiser/birgit/Nonaka-Papers/A-theory-of-organizational-knowledge-creation.pdf> [Accessed 15 September 2015].
21. Nonaka, I., Toyama, R. & Konno, N. 2000. SECI, Ba and Leardership: a Unified Model of Dynamic Knowledge Creation. *Long Range Planning* 33, pp. 5-34.
22. Open Source Initiative. 2012. *The History of the OSI*. [online]. Available at: <http://opensource.org/history> [Accessed 9 September 2015].
23. Open Source Initiative. 2015. *The Open Source Definition*. [online]. <http://opensource.org/osd> [Accessed 9 September 2015].
24. Oss Watch. 2013. *Governance models*. [online]. Available at: <http://oss-watch.ac.uk/resources/governancemodels> [Accessed 15 September 2015].
25. Oss Watch. 2013. *Release management in open source software projects*. [online]. Available at: <http://oss-watch.ac.uk/resources/releasemanagement> [Accessed 15 September 2015].
26. Oss Watch. 2013. *Roles in open source projects*. [online]. Available at: <http://oss-watch.ac.uk/resources/rolesinopensource> [Accessed 15 September 2015].
27. Polanyi M.1966. *The Tacit Dimension*. London: Routledge & Kegan Paul.
28. Rahtz. S. 2005. *What is an open source community?* OSS Watch Conference. July 2005. Edinburgh. <http://www.oss-watch.ac.uk/events/2005-07- 04/index.pdf>.
29. Raymond, E. S., 2000. *The Cathedral and the Bazaar*. [online] Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12 0.4974&rep=rep1&type=pdf> [Accessed 9 September 2015].
30. Raymond, E. S. 2000. *Homesteading the Noosphere.* http://www.catb.org/~esr/writings/homesteading/homest eading/index.html.
31. Raymond, E. S. 2001. *How to Become a Hacker*. <http://www.catb.org/~esr/faqs/hacker-howto.html>.
32. Scharff, E. 2002. *Applying Open Source Principles to Collaborative Learning Environments*. [online]. Available at: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.6755> [Accessed 9 September 2015].
33. Stallman, R. 2015. *FLOSS and FOSS*. [online]. Available at: <http://www.gnu.org/philosophy/floss-and-foss.html> [Accessed 9 September 2015].
34. Stallman, R., 2015. *Why Open Source misses the point of Free Software*. [online]. Available at: <http://www.gnu.org/philosophy/open-source-misses-the-point.html> [Accessed 9 September 2015].
35. TheLinuxFoundation. 2012. *How Linux is built*. [online]. Available at: <https://www.youtube.com/watch?v=yVpbFMhOAwE> [Accessed 9 September 2015].
36. W3Techs. 2015. *Usage of web server for websites*. [online] Available at: <http://w3techs.com/technologies/overview/web_server/all> [Accessed 9 September 2015].
37. WordPress.org. 2012. *Roadmap.* <http://wordpress.org/about/roadmap>.

Contact data:

Jouni, Juntunen, Senior Lecturer

Oulu University of Applied Sciences, Information Technology, Teuvo Pakkalan katu 19, 90130 Oulu, Finland

Jouni.Juntunen@oamk.fi

Teppo, Räisänen, Principal Lecturer

Oulu University of Applied Sciences, Information Technology, Teuvo Pakkalan katu 19, 90130 Oulu, Finland

Teppo.Raisanen@oamk.fi