Learning and knowledge in FLOSS

Situated learning and organizational knowledge-conversion in community-based free/libre open source software development

Master Thesis by Sverre Helge Bolstad

Department of media and information studies, Faculty of social science, University of Bergen, Norway

2006

sverre.bolstad@student.uib.no

ABSTRACT:

In free/libre open source software development (FLOSS), groups of developers and users working in geographically dispersed settings are supported by a dense network of interactions. The participants are highly skilled in the use of information- and communication technologies, and build the software by relying on extensive peer production and through skillful use of communication tools available on the Internet. In building the software, explicit, formal and structured knowledge in the form of documents, objects, machines and external sources are communicated and stored in ways that make it available for others in the present and future. This knowledge make up an important resource for the members and developers of the community. Another kind, or aspect, of knowledge, often called tacit or soft knowledge, is informal, unstructured, resides *in* people, and are difficult, or maybe impossible, to articulate. The questions guiding this research is how knowledge, both explicit and tacit, is shared, and how a new member is able take part in the practice and knowledge of the community. The theory of legitimate peripheral participation in communities of practice describes an environment for people to develop knowledge through interaction with others in an environment where knowledge is created, nurtured and sustained. By taking part in the practice as a participant observer, through virtual ethnography, the author describes the practice and communication in this decentralized and knowledge-intensive process. Taking it a step further, the knowledge of the community, and how it is shared within the 'organization', is explored with a model for managing dynamic aspects of organizational knowledge-creation. The central theme here is that knowledge is created through a continuous dialogue between tacit and explicit knowledge. Logs from Internet Relay Chat (IRC) and interviews with core developers are analyzed, and the author argues that the Plone community is able to share both kinds of knowledge in a complex web of resources and interaction. The analysis further suggest that the FLOSS development-model facilitates access, transparency and participation on premisses that are important for learning.

Keywords: free/libre open source software; CSCW; CSCL; community of practice; organizational knowledge management

ACKNOWLEDGMENTS:

I would like to thank the following people for helping me compile and accomplish this thesis; my supervisor Bjørnar Tessem for constructive and thorough feedback. Jason Mcvetta for welcoming me to the Eventregistration project and letting me use our communications as data. Martin Aspeli for introducing me to some of the people in the Plone community and letting me use research-data he had previously collected. Kristin Svartveit, Svein Olav Norenes and Jan Erik Mandelid for discussions, proof-reading and professional input. I also want to thank the people and community behind free/libre open software tools I have used in this thesis; specially X-chat and Colloquy IRC clients, and the TAMS analyzer. Finally I would like to thank the the people in the Plone community for providing a vibrant and rich research setting which proved to be a heaven of data for social science student interested in technology.

Thank you !

TABLE OF CONTENTS:

1. INTRODUCTIONS

Free/libre open source software (FLOSS) development, with all its facets and implications, offers a wide field for research of current scientific and social interest. As an idealistic and social movement, Free/Libre Software emphasize freedom and sharing of both knowledge and content. In a more pragmatic view FLOSS development has proved to produce a multitude of high quality software such as the GNU/Linux Operating System, the MYSQL database and the Apache web-server. In Norway and in several other countries in the world studies are being conducted on the possibilities to use FLOSS in the public administration, and some countries are already using it. Many small and big companies base their business on FLOSS products and my assumption is that the FLOSS development model is going to be more and more used in the future. With **sourceforge.org´s** [1] over 100,000 registered FLOSS projects, as well as large corporations like Apple, IBM and Motorola, and various governmental organizations using FLOSS, it is necessary to study single successful projects and develop an understanding of the working practice of the communities behind such software. My suggestion is that one of the keys for understanding the success of FLOSS can be found in analyzing the process of communication and how knowledge is created and shared. A large growth in FLOSS projects demands that programmers know models and methods used in FLOSS development. Corporations and companies doing in-house FLOSS development, or cooperating with other FLOSS projects, need to be able to understand how the community functions, to be able to contribute back. If not understood, a typical result could be the creation of forks[2] of the source code, or that the new code is not contributed back to the community.

Actors from commercial-, public/state- and academic sectors are interested in how to learn, use and teach the methods of FLOSS development. The main goal of the project will be to understand more about learning in such a setting, and about the way knowledge is created and shared, but also to learn more about distributed software development in general. Hopefully this thesis will be of interest for the research fields of Computer Supported Cooperative Work (CSCW), Computer Supported Collaborative Learning (CSCL) and Computer Supported

1 Sourceforge.net is the world's largest Open Source software development web site, hosting more than 100,000 projects and over 1,000,000 registered users with a centralized resource for managing projects, issues, communications, and code.(http://sourceforge.net)

2 A project fork or branch happens when a developer (or a group of them) takes a copy of source code from one software package and starts to independently develop a new package.

Communication (CMC), as well as in the field of software development research, and specially free/libre open source software development. This first chapter is an short introduction to Free/Open Source Software development and to Plone Content Management System, which is the case in focus in this research. After the introductions I move on to existing research before I present the theories applied in this thesis. I will then describe the methodological framework used, before I do the analysis, and finally makes some concluding remarks.

1.1.Free/Libre Open Source Software Development (FOSS)

Developers in the 1970s frequently shared their software in a manner similar to the principles of free/libre open source software. In the late 1970s, companies started routinely imposing restrictions on users with the use of license agreements. In 1984, Richard Stallman started working on the GNU project, founding the Free Software Foundation[1] (FSF) one year later. Stallman introduced the concepts of "free software" and "copyleft", which he specifically devised to give users freedom, and to restrain the possibilities for proprietation[2] of software. The FSF has produced a specific free software definition, by which software is free in the sense that it grants four freedoms: *the freedom to run the program for any purpose* (freedom 0), *the freedom to study and modify the program* (freedom 1), *the freedom to copy the program so you can help your neighbor* (freedom 2) and *the freedom to improve the program, and to release your improvements to the public, so that the whole community benefits* (freedom 3). This implies important political, democratic and practical benefits for the user of the software, and the community at large, and it constitutes the basic idea that technical knowledge is free and should circulate freely.

The Open Source Software (OSS) movement is philosophically distinct from the free software movement. It began in 1998 with a group of people who formed the Open Source Initiative[3] (OSI). They sought to bring a higher profile to the practical benefits of sharing software source code, and to interest major software houses, and other high-tech industry companies, in

1 The Free Software Foundation (FSF) was established in 1985 and is dedicated to promoting computer users' rights to use, study, copy, modify, and redistribute computer programs. http://fsf.org

2 Proprietation is the act of imposing restrictions on the use and copying of the software, usually enforced by a proprietor.

3 The Open Source Initiative is a non-profit corporation dedicated to managing and promoting the “Open Source Definition” which is a list of 10 criteria that open-source software must comply with. http://www.opensource.org/

the concept. These advocates see the term open source as avoiding the ambiguity of the word "free" as in free software. Many people recognize a qualitative benefit to the software development process when a programs source code can be used, modified and redistributed by developers. The main difference between OSI and the Free software movement, is that the Free software movement places primary emphasis on the moral or ethical aspects of software, seeing technical excellence as a desirable by-product of its ethical standard. The Open Source movement sees technical excellence as the primary goal, regarding source-code sharing as a means to an end. Despite this difference, people from both camps often work together on same projects and the actual result will be the same; free software with open source code, with all the freedoms from the FSF and, sometimes, with the technical excellence emphasized by the OSI. In this paper I choose to use the term free/libre open source software (FLOSS) to include both camps and not to go further into this discussion.

The FLOSS developing model is used to build end-user applications like for example Internet browsers, programs for scientific analysis, games and communication tools, but also software for Internet infrastructure like web application servers, and also developer tools such as libraries, programming languages and even operating systems, which can be used and extended freely if one has the knowledge. This constitutes some solidarity principles connected to principles in the Internet's original constitution as an open place where knowledge can circulate freely. Skeptics have pointed out that the sharing principle predates the open source movement; for example, the free sharing of information has been institutionalized in the scientific enterprise at least since the 19th century.

Creation and innovation in FLOSS projects can be higher than in traditional development because more people are customizing the product to use it themselves to fit their needs, and also because of the special relation the user and developers have to the software and its community, and also because ideas can be freely copied from similar projects. The result is often widely used products with high quality code. It counters the, until now, widely held belief in the market based, or corporate development models, as the ideal way of creating high-quality software. FLOSS has been accused to be “elitist”, and its high prestige connected with being a central hacker[1] in a important FLOSS project. Since digital media and the Internet-based infrastructure services are becoming more and more important elements of our

1 The word hacker can have positive or negative connotations depending on the cultural context where it is used. I use it as a positive term to describe a respected programmer who knows a set of programming interfaces well enough to write software rapidly and expertly.

society, and how we live our lives, the technical questions concerning these issues, like how software and services are developed, are important for our lives. Open formats and alternative solutions are important to prevent monopoly and to ensure competition and free access to technological knowledge.

1.2.Plone CMS

I have used three criteria to find a suitable case for my research; 1) the project should have an extended base of users and developers, 2) it should be community driven, 3) it should produce an, for me, interesting software, which I was willing to learn how to use. I have examined different FLOSS projects and found Plone Content Management System (CMS) as a suitable case. Plone is described by Wikipedia in this way:

> *"Plone is an extensible content management system written in the Python programming language based on the Zope. It can be used as an intranet and extranet server, a document publishing system and a group ware tool for collaboration between separately located entities. The Plone project was started in 1999 by Alan Runyan, Alexander Limi, and Vidar Andersen. It has quickly grown into one of the most popular and powerful open source content management systems in the world. In 2004, the Plone Foundation was formed to protect and promote the use of Plone. It is built on top of the open source application server Zope and the accompanying Content Management Framework, which has thousands of developers around the world supporting it" (Wikipedia, 2005)*

Plone is registered at **sourceforge.net** with 90 developers and a files activity percent of 99,6. (Sourceforge.net, 20.05.06). A developer has write access to the subversion system (svn[1]) and often has decision powers. Although this number is the official **sourceforge.net** number, there are many more developers contributing to the project. Other work that is done by users, contributors and committers are filing bug-reports and doing bug-fixing, writing translations, providing documentation or doing maintenance of the infrastructure of the project. The Plone project uses its own software product as its collaborative platform to manage documentation, information and sub-projects. The development community is often also its own end-users and participants make their own system requirements. They often use the tools they develop as a platform for creating solutions for external clients. Many Plone developers make their livelihood from clients that pay independent programmers, or small companies based on Plone CMS, to write code that usually goes back into the Plone source code. Plone is widely

1 Subversion (svn) is an FLOSS application used for revision control, it keeps track of all work and all changes in a set of files in a software project, and allows several distributed developers to collaborate. http://subversion.tigris.org/

used all over the world and has more than 626,629[1] registered downloads from **sourceforge.net**. Companies, governments and NGO's that use Plone CMS includes NASA, eBay, Nokia, Oxfam America, Creative Commons, The Church of England, The Free Software foundation, Japan External Trade Organization, The Brazilian Parliament, the Governor of Texas and Hawaii, The Defense Academy of the United Kingdom, The Universities of Gothenburg, Bristol, Chicago and Utah and the City of Bern to name a few. Plone is made up of a variation of tools for collaboration and communication, and is highly customizable. Infrastructures for hosting FLOSS projects support and coordinate the development of software projects. It can be as simple as a mailing list or an ftp server, but in the Plone project it includes several elements such as websites, content management system, chat, discussion forum, group-ware and version control system. These tools are often called community tools and are used widely in the coordination of FLOSS projects. Plone CMS is on top of a technology stack with the FLOSS programing language Python in the bottom and the web application server Zope[2] and the Content Management Framework[3] (CMF) as important elements. Changes done on a lower level in the stack is reflected further up as it affects Plone's technology base.

1.2.1.The Plone community

Plone CMS is community driven, has a recognized name, and well defined, well known software. The opposite of "community driven" in FLOSS is "vendor driven", and even though Plone has some very active companies on the contributor list, the community plays the most important part in developing the software. The atmosphere is friendly, no one get their hands smacked if they do mistakes or commit code that breaks the software. The community has several hundred, or even thousands of members. In the center of the community you will find the initiators of the project, the core developers and people from the Plone foundation. Further out there is co-developers and active users that modifies the software and use it as an development platform, and further out there is active users, that installs and uses the application, sometimes without touching the code. Several times a year central people from

1 Numbers from statistics at sourceforge.net, 15.06.06. This does not include downloads directly from svn, other servers nor packages included in Linux distributions. http://sourceforge.net/project/stats/?group_id=47214&ugn=plone

2 The official home site of the Zope application server can be found at http://zope.org/

3 The Content Management Framework (CMF) from Zope Corporation provides a platform for building content management applications.

the Plone community meets at sprints[1] and conferences. The Plone community is on the top of a "community stack" with the Python[2] community in the bottom and the Zope community as Plone´s closest bordering community. Zope has a smaller community and some challenging issues. Some of the problems with Zope, according to Plone community members, is lack of a brand and identity (Panel debate, Europython 2005). A version of Zope is closed-source and is sold as a proprietary product on the side of the community. Python, which is the programming language that Zope and Plone is written in, has a large community of developers and supporters.

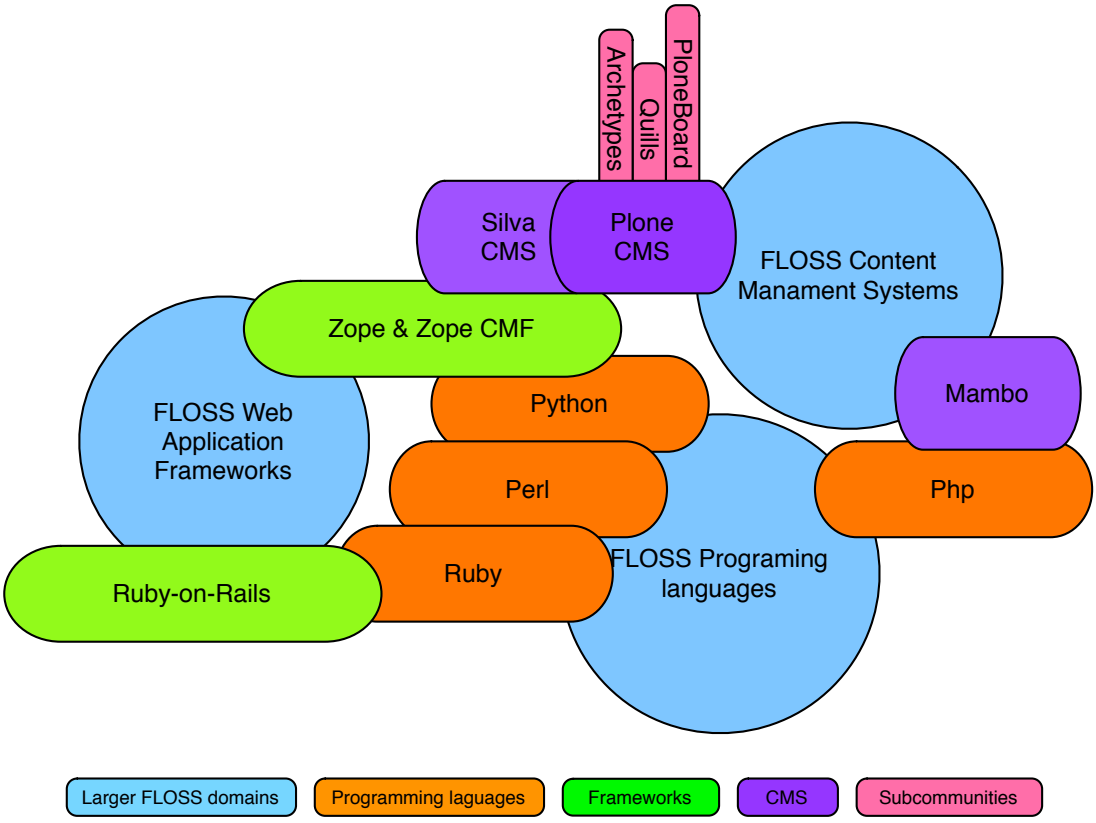

Illustration 1: Plone CMS and bordering technologies and communities

1. A sprint is a three to five day focused development session, in which developers pair in a room and focus on building a particular subsystem. Sprints are based on ideas from the extreme programming (XP) community http://plone.org/events/sprints/whatis

2 The official website for the Python programming language can be found at http://www.python.org/

Illustration 1 shows how the Plone project is part of a a complex network of overlapping and interacting communities and sub-communities. The overlapping figures show what other technologies and communities they are part of[1], and also their competitors.

1.2.2.The Plone foundation

The Plone foundation exists to "*...further the development, marketing, and legal affairs of Plone and the Plone community*" (plone.org/foundation.org, 2006). The foundation is modeled after similar ventures, such as the Apache Software Foundation, and is providing support for the development and marketing of Plone. The Foundation is also the legal owner of the Plone code, trademarks, and domain names. Another important function it has, is to act as the voice of Plone for official announcements, press releases, and other communications. The foundation also tries to get Plone contributors to transfer rights to the Foundation, so that there is a safe, consistent, and flexible organization that can handle the Plone commons[2]. In this way Plone stands juridically strong, something that is more and more important for FLOSS projects. The Software Freedom Law Center, that are financed by companies like NEC, Intel and HP has taken the Plone Foundation under its wing and contributes with juridical assistance. The licensing scheme is an important issue for commercial interests in Plone. The Foundation might pursue a dual-licensing (GPL[3] and non-GPL) scheme. The Foundation does not take technical decisions for Plone, the development team still leads this process.

1.3.Research question and area

With this thesis I hope to contribute to the small line of research that aims to understand how FLOSS works on an everyday basis. I want to describe the practice of the community, where knowledge is embedded, and especially how a "newbie" (newcomer) can enter the

---

1 Plone CMS is on top of Zope and CMF which are using the Python programing language. Eventregistration is a sub-community of Plone.

2 I define commons as resources belonging to or affecting the whole of the community.

3 The GNU General Public License (GNU GPL or simply GPL) is a widely-used free software license, originally written by Richard Stallman for the GNU project. The latest version of the license, version 2, was released in 1991, with a upcoming version 3 coming in 2006. The GPL grants the recipients of a computer program the right to run the program, for any desired purpose, the right to study how the program works, and modify it. (Access to the source code is a precondition for this), the right to redistribute copies, and the right to improve the program, and release the improvements to the public. (http://www.gnu.org/licenses/gpl.html)

community. By describing communication and activities, I aim to shed light on social aspects of FLOSS, from a learning/knowledge-sharing perspective.

Why is it that FLOSS is able to produce high quality software in a distributed agile process ? One hypothesis is that the participants in FLOSS project is able to communicate tacit knowledge which usually only is transferred in situations with physical presence. This kind of knowledge are often described in terms like informal, less structured, or soft. Its refers to what people know, but which cannot be articulated, abstracted, codified, captured or stored easily. The Plone community produce a large amount of externalized, or explicit, knowledge in the forms of how-tos, tutorials, instructions, notes, faqs and written procedures. This type of knowledge is often referred to as hard knowledge. I posit that FLOSS, as an online community of practice, also overcome the problem of tacit knowledge-transformation through dense interaction, collective reflection and accumulation of knowledge. I think that knowledge and knowledge-artifacts is produced in a way that resembles the way participants in a FLOSS development-process creates code by modification and reuse of existing code, in a open, collaborative setting.

In *The future of sociology of FLOSS* Yuwei Lin (2005) suggests that:

> *"methodologically, we need more grounded, ethnographic-oriented research for our understanding of the socio-technical practices of deployment, development and implementation of FOSS in different contexts."(Lin 2005)*

FLOSS is in many cases user-initiated, user-developed, user-driven and user-tested in a collaborative effort. Participants build and share knowledge and learn from each other across time and space. Instead of studying a system built for the purpose of collaboration and learning, my intention is to analyze how the practice of a community is formed by, and uses, the possibilities of the Internet for knowledge-sharing, collaboration, learning and communication. I assume that the participants are skillful users of these tools.

To examine the everyday practice in Plone and how knowledge is created, shared and nurtured amongst the participants in the project I focus on two main questions. First:

How does a new member learn in this setting, and how does he/she take part in the practice of the community ?

An important issue for a FLOSS project is to attract new users, developers and members to the community, and to integrate them in the project. The traditional power-relationship between experts and lay people, users and designers, might be challenged in a FLOSS project, given that "the influence of local knowledge and tacit skills is very much in evidence in the FLOSS innovation system" (Lin, 2004). I want to find out how a newcomer can take part in the technology of practice and gain the necessary knowledge to be able to act in the community.

The most used channels for communication in Plone are the mailing lists and IRC channels, but Internet technology is used in various ways to pool and archive knowledge resources. The design of those online platforms provides the frame in which knowledge is concentrated and activated as a resource for creation, and a participant has to learn how to use these resources.

I also suggest that FLOSS communities are able to share tacit knowledge. I want to find out if, and how, this aspect of knowledge can be shared in an online environment:

How and where is tacit knowledge shared among the participants in the Plone community ?

These two questions are reflected in the data-gathering process. I will first gain experience through virtual ethnography and then analyze communication and tools used by other members to see how knowledge, and more specific, how tacit knowledge, can be shared. The practice shared by the participants is what holds them together, and learning and knowledge-sharing are important parts of the socio-technical practice. This question calls for knowledge on the subject of learning, knowledge, communication and FLOSS projects, as well as computer supported cooperative/collaborative work and learning.

Having presented the setting for the research and the research question, I will move on to briefly review some existing research in the field. After that, I present the theories I will use in the analysis, before I describe the method I use for collecting data, and the design of the research. I will then proceed with the analysis, and finish with some conclusions.

2.EXISTING FLOSS RESEARCH

"Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information on it."
Samuel Johnson (1709 - 1784)

The last few years the phenomena of FLOSS has been subject for research from many different disciplines of academia. Most research projects look at the economic implications of FLOSS, motivational aspects, governing models or the software development model, but some of them also focus on learning and knowledge-sharing. I will first present a research that used Plone as its case, and then look at some more specified research on learning and knowledge in FLOSS. Many research-projects seems to treat FLOSS as one community, with one structure and one model for development. From the more than 100000 projects registered at **sourceforge.net** we can see that this is not the case; projects varies in number of developers and participants from one to many thousand, with a multitude of different ways of organizing, both socially and technically. Each project is unique, based on its technical base (the code), its participants and relations between them (the community), and the relations to other communities, users and external actors such as commercial companies and organizations (the world). It is no more sensible to talk of "one" mode of open source development then it is to talk about "one" approach to commercial software development, or indeed "one" way in which any task-oriented community may be organized (Aspeli, 2005:6).

2.1.Plone: a model of a mature open source project

When I started writing my thesis there were no FLOSS research projects using Plone CMS as a case. When I was halfway in the work with my thesis Martin Aspeli published a master thesis titled *Plone: a model of a mature open source project* (2005) at the London School of Economics. Aspeli is a core Plone developer and has been a member of the community for more than 2,5 years. I first met Aspeli on the Europython 2005 conference, where we briefly discussed our thesis' and he introduced me to some of the core participants in the community. In his dissertation Aspeli conducts a thorough literature review on existing FLOSS research. To give an overview I will sum up some of his points here. Aspeli (ibid) argues that his survey on the open source literature reveals "a strong bias towards positivistic articles, even among those which attempts to understand the social processes of of the community through case study research (e.g O'Mahoney and Ferroaro, 2004; Cummings, 2002, Giuri et al., 2004) few have been able to capture the everyday practice of FLOSS where the ideals of transparency, code sharing, peer review and open communication are important aspects." Tuomi (2005)

asserts that the ability to support multiple forms of motivation is a particular strength of the FLOSS model. Franck and Jungwirth (2002) theorizes that two types of contributors are found in FLOSS: those who contribute to "invest" in reputation and learning, and those who "donate" for altruistic or ego-gratification reasons, and that the novelty of FLOSS is, in particular the license that ensures that the software remains perpetually free (as in speech), to permit those two types of contributors to come together (Aspeli, 2005). von Hippel and von Krogh (2003) posit that open source development constitutes a "private-collective" innovation process, where innovation is achieved by users, not producers, and point to intangible personal rewards, such as social benefits, learning or a sense of ownership, as motivation for such innovation (Aspeli, 2005). The common conception of the organization of a FLOSS project is that of a strong "core" of developers, and a large bug-fixing and testing periphery which have given rise to "layers of the onion" models such as the one proposed in Crowston and Howison (2004).

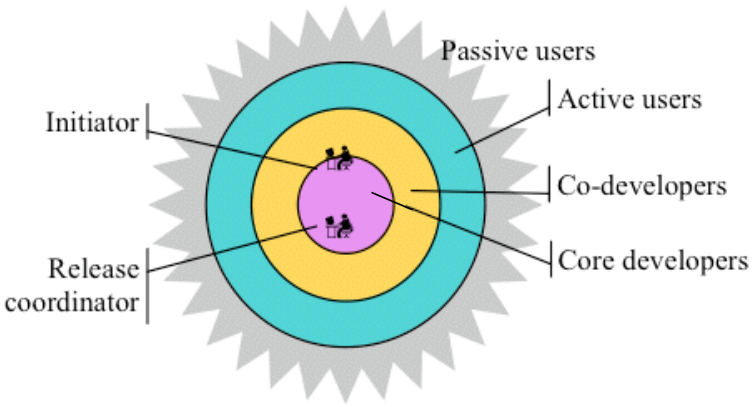

Illustration 2: A synthesised FLOSS development team structure. Crowston and Howison (2004)

Aspeli argues that early literature on computer-mediated communication and knowledge management suggests that text-based communication is "lean" and thus poorly suited for communicating tacit knowledge and social cues (see Kraut and Streeter, 1995), making virtual communities more action-oriented and less intimate (Yamauchi et. Al, 2000). Communication through mailing lists, chat, open source code and automatically generated modification-emails helps the developers stay aware of each others activities and share knowledge. Plone members also meet in person quite regularly as part of sprints, conferences or through personal friendship. Aspeli holds that:

> *"certainly learning is a common motivation for contributing, and learning through "apprenticeships" takes place whenever developers of different skills collaborate on the same piece of software". (Aspeli, 2005:22)*

Through interactions in cyberspace, FLOSS development is driven forward by knowledge-sharing and experimentation, through process of destruction, reflection-on-action and discourse (Hemetsberger and Reinhart, 2004). Knowledge constructed in this manner is intrinsically linked to the community and the identity contributors find there. Thus, meaning, as defined by the community and its culture, becomes embedded in the software artifact, recursively influencing future development and knowledge-sharing (Tuomi, 2000).
Wenger (2000) emphasizes that successful organizations must organize themselves as social learning systems, by giving primacy to informal learning mechanisms, meaningful participation and receptiveness to complexity. Aspeli (2005) argues that

> *"these concepts lies in the very core of Plone: Informal learning occurs daily through the mailing list and chatroom, participation is greatly valued and encouraged and the projects decentralized nature makes it well suited for dealing with complexity, with the ability to re-factor different parts of the software into add-on modules or lower level frameworks where necessary". (Aspeli, 2005:23)*

Aspeli's model of a mature Open Source model is based on Plone's community of practice and includes the elements of organization and governance, culture and dynamics, the FLOSS life-cycle, and user and business relationships of the project. Perhaps more important than the specifics parts of the model, however, is the demonstration that mature open source communities are not so different from communities in the offline world: they are socially complex, playing host to rich interactions between real human beings.

> *Any account of motivation, organization, governance, business models or other aspects of the open source movement which reduces participants to purely technically or economically rational actors inevitably betrays this complexity, ultimately limiting our ability to understand these communities (Aspeli, 2005:33)*

Aspeli concludes that the "community of practice" literature has shown to be highly relevant to open source communities and that a longitudinal case study could yield further insight into its social process and underpinning structures.

2.2.Sharing and creating knowledge in open source communities

In *Sharing and creating Knowledge in Open-Source communities: The case of KDE* Hemetsberger and Reinhardt (2004) conducts a research on KDE, a window manager for Linux. They start out by saying that research on open source communities of practice;

> *"haven't given any answers about how the knowledge sharing and creation process develops at the interface of technology and communal structures that efficiently exploit the advantages of Internet technology". (Hemetsberger and Reinhardt (2004:2)*

The questions guiding their research is how community members organize content that potentially can transform into knowledge for other members, how new members are enabled to accumulate the knowledge necessary for becoming a valued member and how members co-create and conceptualize new ideas. They claim that their research demonstrates that:

> *"...online communities of practice successfully overcome the problem of tacit knowledge transformation through technological tools, task-related features, collective reflection, stories and usage scenarios. Doing this, online communities of practice constantly support knowledge creation and dissemination not only for the current actors involved, but also for future generations." (Hemetsberger and Reinhardt (2004:6)*

The observation that the knowledge is available for others in the future is important, and is a powerful feature of open, digital, Internet-based communication.

2.3.The knowledge ecology of open source software projects

In *The knowledge ecology of open source software projects* Lanzara and Morner (2003) says that open-source software projects are evolutionary systems based on dense interactions between humans and technical artifacts within an electronic media. In such an environment knowledge processes develop by means of "variation, selection, and stabilization, gaining a distinctive ecological quality"(ibid). In their explorative approach, based on in-depth analysis of Linux and Apache, they use three arguments prior to entering their research: 1) FLOSS projects should be conceptualized as interactive systems, and that knowledge is the emergent outcome of interaction; it is a communication based process. 2) If we want to grasp how knowledge creation and dissemination happen in FLOSS, we should look at the technology. Interactive systems are deeply intertwined with a web of artifacts and tools that support programming, development and knowledge making activities at large. 3) The diversity of knowledge processes and practices in open-source software projects will be better appreciated if they focus on their distinctive ecological quality. FLOSS projects are intricate webs of

agents, practices, artifacts, tools, resources, problems and solutions that co-exists and interact in an ever evolving ecology, where the evolutionary mechanisms of variation, selection, and stabilization keep the activity system in a dynamic balance (ibid).

> *"In an ecological perspective, knowledge is created by leveraging the scattered and occasional contributions of many small agents. Knowledge is then the unplanned, evolutionary outcome of the complex interplay of a broad variety of heterogeneous elements rather than being a manufactured commodity or the product of smooth conversion processes." (Lanzara and Morner, 2003:3)*

As a methodological choice, they place artifacts and their critical role as dynamic holders and vehicles of knowledge in the focus of their observations. Artifacts play the role of media which human communicate and act through. They look at how these artifacts are affected by, and support, the mechanism of variation, selection and stabilization in the making of technical, organizational and institutional knowledge. They suggests that technology can be seen as inscriptions of human agency and knowledge. Follow Latour (1992) they use the concept of *inscription* as the act by which humans cast relevant components of their agency and knowledge into artifacts that become holders and dynamic vehicles of human agency. They hold that technology, rather than organization, embodies most of the conditions for governance in FLOSS projects, hence becoming a critical pathway to the understanding of collective task accomplishment, coordination and knowledge making processes. The idea of ecology applied to knowledge suggests that whatever we call 'knowledge' in FLOSS projects is the evolving outcome of the processual interplay of multiple contributions (Bateson, 1972, Sindig-Larsen,1987, Anderson & Laird, 1988 in Lanzara and Morner, 2003). This suggest that "knowledge comes out of bricolage, in which a lot of creative recombination and recycling of existing materials takes place" (Lanzara, 1999, Ciborra, 2002).

2.4.Community, joining, and specialization in open source software innovation

With *Community, joining, and specialization in open source software innovation* von Krogh, Spaeth and Lakhani (2003) intends to contribute to a theory of the open source software innovation process by examining joining behavior in a development community. They use the *private-collective innovation model* (von Krogh and von Hippels, 2003) to say that FLOSS represent a private-collective model of innovation where developers obtain private rewards from writing code for their own use, sharing their code, and collectively contributing to the development and improvement of the software. They claim that success of a project, in terms

of producing the software, relates to the growth in the size of the developer community. (Moody, 2001; Raymond, 1999, Sawhney and Prandelli, 2000, Waner, 2000 in von Krogh et al., 2003). They acknowledge that software development is a knowledge-intensive activity that often requires very high levels of domain knowledge, experience, and intensive learning by those contributing to it (Pliskin et al., 1991; Waterson et al., 1997 in von Krogh et al. 2003). As the technology grows more complex, only a few people who have been actively involved in its development over a certain period of time might fully understand the software architecture and efficiently contribute code to its development, and new contributors might find it too costly to join the project (Kohanski, 1998). von Krogh et al. (2003) holds that researchers have to understand how joiners become newcomers, that is, how they make their initial contribution to software. In particular, it is important to understand that what benefits newcomers, derive from belonging to an existing developer community. The literature on commercial software development suggests that "modularization" (Baldwin and Clark, 2000) of the software code may increase a projects transparency, lower barriers to contribute and allowing for specialization by enabling efficient use of knowledge. Their research attempts to contribute to a theory of open source software innovation process, by uncovering if newcomers specialize and what may cause this specialization. They hold that the transition from joiner to newcomer is achieved when a person is granted access to the developer community and to a privileged “source-code-commit-regime”. The author suggests that the sharing of knowledge among in-going, outgoing, and remaining developers needs more attention, and that solid theory building and empirical studies on the social aspects of software development is still lacking. They also suggests that their study shows how the FLOSS development process is transparent, both with regards to the social and the technical aspects.

2.5.Keeping it going: the everyday practices of open source software

In *Keeping it going: the everyday practices of open source software* Monteiro, Østerlie, Rolland and Røyrvik (2004) argue that a key challenge in FLOSS projects is to cultivate and nurture a motivated community of developers. Through a case study of the operative system distribution Gentoo GNU/Linux they analyze three mechanisms that fosters continuity for keeping the project going. Methodologically they emphasize the everyday interactions, drawing predominantly on synchronous communication. They promote a process-oriented perspective on the social organization of FLOSS efforts, emphasizing the importance of small, seemingly mundane actions and gestures. “The everyday life – not heroic moments or “big” decisions – form the life-blood of FLOSS efforts and deserve closer

scrutiny" (Monteiro et al, 2004:3). By recognizing that work, learning, and innovating are interrelated and compatible and thus potentially complementary, not conflicting forces (Brown and Duguid, 1991) they embrace a stream of knowledge management and organizational learning literature that emphasize how innovation are part of daily "non-canonical" practices, unexpected twists and turns, and the "muddling through" of practical decision-making and knowing (Orlikowski, 2002). They argue that the implications that follow their study are of three different types: analytical, methodological and practical. Analytical they say that FLOSS projects need to be conceptualized as related to, not independent of, other software development projects. Methodological they observe that there is a need to focus more on the everyday interaction of FLOSS, including a closer scrutiny of interactive communication such as Internet Relay Chat. On the practical side their study has implications for the practice of project management. While conventional literature on project management typically advocates a logic of project control strongly emphasizing bureaucratic routines and follow-up of explicitly stated and agreed upon milestones, they argue that their study shows that different forms of mechanisms are important for keeping large and high-risk projects together. These are organizational mechanisms focusing on more loosely defined and situated divisions of labour combined with technological means for delegating and coordinating work. Ritual mechanisms in terms of the small and often "invisible" rituals of interaction, expressed in the ceremonial aspects of projects maintenance are important glue that prevents things from falling apart. Further, they argue that "enabling a culture of communication that cultivates, includes and integrates all these types of minor forms of interaction rituals, should be given due consideration ... " (Monteiro et al. 2000:29). Their study also suggests that in management of technology-infused projects one needs to take into account that the development and use of technologies also can be sources for uncertainties and surprises.

These existing research projects suggest that there is a need for looking into the everyday practice of the community and focus on how the technology supports large collaborative efforts. There is also a need for empirical studies on the social aspects of software development. I will now give an introduction of the literature on learning and knowledge-sharing that I use in my analysis.

3.THEORETICAL FRAMEWORK

"That one is learned who has reduced his learning to practice."
Hitopadesa

This chapter introduces the main theories to be used in my analysis. This thesis concerns both learning and knowledge-sharing and introduces two different theories, first a theory on learning and then a theory of organizational knowledge-conversion and how communities learn.

3.1.Legitimate peripheral participation in communities of practice

Social learning theory posits that people learn from observing others, and acting together with other people. By definition, such observations take place in a social setting (Merriam and Caffarella 1991: 134). Attending to a behavior; remembering it as a possible model or paradigm; and playing out how it may work for them in different situations (rehearsal) are key aspects of observational learning. *Situated learning: Learning through legitimate peripheral participation* (LPP) is an analytic framework based on this, and has been put forward by Jean Lave and Etienne Wenger (1991). Rather than seeing learning as the acquisition of certain forms of knowledge, they have tried to place it in social relationships – situations of co-participation. In this model it is not so much that learners acquire structures or models to understand the world, but they participate in frameworks that have structure. Learning is an integral part of generative social practice in the lived-in world. Lave and Wenger's problem – and central preoccupation of the theory - is to translate this into a specific analytic approach to learning, which they call *legitimate peripheral participation* (LPP). The background of LLP is to be found in the discussion around apprenticeship. The use of the word apprenticeship, in cognitive and educational research, were earlier largely metaphoric, even though apprenticeship has a long history as an educational form. By the concept of LPP, Lave and Wenger (1991) means to draw attention to the point that:

> *"...learners inevitably participate in communities of practitioners and that the mastery of knowledge and skill requires newcomers to move toward full participation in the socio-cultural practices of a community" (Lave and Wenger 1991: 29)*

Dividing the concept of legitimate peripheral participation into its constituent parts to understand them better might seem obvious, but Lave and Wenger emphasize that this in a large degree would destroy our possibility to understand learning, exactly because it is a

complex concept that combines several aspects of social processes, and where each aspect is important in the explanation of the others. However, a look at the different words, in relation to each other, is helpful for a better understanding of the concepts.

Peripherality suggests that there are multiple, varied, more or less-engaged and inclusive ways of being located in the fields of participation defined by a community. Peripheral participation is about being located in the social world. Changing locations and perspectives are part of actors learning trajectories, developing identities, and forms of membership. Legitimate peripherality is a complex notion, implicated in social structures involving relations of power. Peripherality can be a source of power or powerlessness, as in a place in which one is kept from participating more fully. The norms for who can talk, about what, with whom, and when offer either opportunities or constraints for how participants are able to negotiate their meanings. The ambiguous potentialities of LPP reflects the concept´s pivotal role in providing access to a nexus of relations otherwise not perceived as connected. The word peripherality is somehow misleading because Lave and Wenger says that there is no place in the community which is designated to "the periphery", and that it has no single core or center (physical, political or metaphorical). Full participation is what LPP leads to and is intended to do justice to the diversity of relations involved in varying forms of community membership. Lave and Wenger use peripherality as a positive term, whose most salient conceptual antonyms is un-relatedness or irrelevance to ongoing activities. The partial participation of newcomers is by no means disconnected from the practice of interest. In this sense, peripherality, when it is enabled, suggest an opening, a way of gaining access to sources for understanding through growing involvement. LPP offers a framework to talk about identity, activity, access, learning and power relationships between participants in the Plone community. The form that the legitimacy of participation takes is a defining characteristics of ways of belonging, and is therefore not only a crucial conditioning for learning, but a constitutive element of its content.

Lave and Wenger holds that it is not appropriate to treat LPP as a mere distillation of apprenticeship or an abstracting process of generalizing from examples of apprenticeship. They argue that it´s theoretical significance "*...derives from the richness of its interconnections: in historical terms, through time and across cultures.*" (ibid, 1991). LPP is not itself an educational form or, much less a pedagogical strategy or teaching technique. It is an analytic viewpoint, and a way of understanding learning. The theory does not take school learning into consideration, although they don't deny that learning can take place where there is teaching. School learning is predicated on claims that knowledge can be de-contextualized.

Their viewpoint makes a fundamental distinction between learning and intentional instruction. Lave and Wenger illustrate their theory by observations of different apprenticeships. Initially people join communities and learn at the periphery, as they become more competent they move more to the 'center' of the particular community. Learning is, thus, not seen as the acquisition of knowledge by individuals so much as a process of social participation:

> *"...the mastery of knowledge and skill requires newcomers to move toward full participation in the sociocultural practices of a community. Legitimate peripheral participation" provides a way to speak about the relations between newcomers and old-timers, and about activities, identities, artifacts, and communities of knowledge and practice. A person's intentions to learn are engaged and the meaning of learning is configured through the process of becoming a full participant in a sociocultural practice. This social process, includes, indeed it subsumes, the learning of knowledgeable skills." (Lave and Wenger 1991: 29)*

The basic argument made by Lave and Wenger is that communities of practice are everywhere and that we are generally involved in a number of them. Some communities of practice are quite formal in organization, others are very fluid and informal. However, members are brought together by joining common activities and by what they have learned through their mutual engagement in these activities (Wenger 1998). In this respect, a community of practice (CoP) is different from a community of interest or a geographical community in that it involves a shared practice.

According to Etienne Wenger (1998), a community of practice defines itself along three dimensions:

> *1) What it is about – its joint enterprise as understood and continually renegotiated by its members.*
>
> *2) How it functions - mutual engagement that bind members together into a social entity.*
>
> *3) What capability it has produced – the shared repertoire of communal resources (routines, sensibilities, artifacts, vocabulary, styles, etc.) that members have developed over time.*

A community of practice involves much more than the technical knowledge or skill associated with undertaking some task. Members are involved in a set of relationships over time and communities develop around things that matter to people. The Plone community is for

example organized around a particular area of knowledge and activity that gives the members a sense of joint enterprise and identity. For a community of practice to function it needs to generate and appropriate a shared repertoire of ideas, commitments and memories. It also needs to develop various resources such as tools, documents, routines, vocabulary and symbols that in some way carry the accumulated knowledge of the community.

Lave and Wenger examine five different cases of apprenticeship in different cultural contexts. The first case is an observation about the Vai and Gola tailor apprentices within an analysis addressing question on how apprentices might engage in a common structured pattern of learning experiences without being taught, examined, or reduced to mechanical copiers of everyday tailoring tasks, and how they became skilled and respected tailors. I will shortly summarize three other studies which I will refer to, and use as comparison later in my analysis. These examples illustrates different historical, geographical and social cases of apprenticeship, and are illuminating for a more specific understanding of the concept of LPP. The research they refer to are done by various researchers, and cited by Lave and Wenger (1991) in the creation of the theory of LPP.

Naval Quartermasters (Hutchins, as cited in Lave and Wenger, 1991)

Quartermasters begin their careers with rather limited duties and advance to more complicated procedures as they gain expertise. A newcomer have to learn how to plot the ship´s position either alone or as a collaborative activity with others. It takes about a year to learn it. There are many resources for learning. Some go to a specialized school to learn the basic before they enter the ship, but they do not gain experience doing this (they are trained, but have no experience). Some of the training aboard the ship is a bit like school with workbooks and exercises. In order to gain higher rank novice quartermasters participate in joint activity with more experienced colleagues. At sea, depending upon the level of experience, the novice may be asked to perform all of the duties of the quartermaster of the watch. While he does this his activities are closely monitored by the more experienced watch stander, and he helps out if the novice is not able to satisfy the requirement of the ships navigation. The novice does not do this before he has several months of experience, because a lot of knowledge is needed to do this. The task of the novice is to learn to organize his own behavior such that it produces a competent performance. The novices has to move through six different positions, mastering each before moving on to the next. Being in the presence of others who are working is not always enough by itself. The fact that the work was done in an interaction between members,

opened it to other members of the the team. In a similar way, the design of tools can affect their suitability for joint use, the interaction of a task performed with a tool may or may not be open to others, depending upon the nature of the tool itself.

Meat cutters (Marshall, as cited in Lave and Wenger, 1991)

Not all concrete realizations of apprenticeship are equally effective. Apprentices can be a form of cheap source of unskilled labour, put to work in ways that deny the learners access to activities in the mature practice. Gaining legitimacy may be so difficult that some fail to learn until considerably time has passed. The exchange of labour for opportunities to become part of a community of mature practice can be fraught with difficulties. The example of meat-cutters illustrates several of the potential ways in which particular forms of apprenticeship can prevent rather then facilitate learning. The butchers apprenticeship consists of a mix of trade school and on-the-job training. At school they learn things which are relevant/specific to the trade, but which they don't need in supermarkets. Because they are not in demand, few students bother to learn them. Sharpening the knife is the first thing learned, which is not necessary, because at the supermarket they use a company that delivers fresh knives. Because the meat department manager tries to achieve a cost effective business the apprentice is set to do most efficient work and the skilled journeymen specialize in short repetitive tasks. This prevents the apprentices from learning very much. The physical layout of an work setting is also important, because observation of more experienced workers and being observed themselves is important. At a supermarket the apprentice worked in a place out of contact with the experts, and he didn't dare to go in to them, because "they knew so much and he knew nothing about meat cutting". An apprentice is often trained to perform a task, and stays at this task untill another apprentice comes along. If none comes along, he might stay at this place for years. In shops in poor neighborhoods the chance is better for learning, because the cost error is lower, and with a higher number of workers, division of labour increases efficiency (Marshall 1972:42-6).

Non-drinking alcoholics (Cain as cited in Lave & Wenger, 1991)

A detailed view of the fashioning of identity may be found in the analysis of the process of becoming a non-drinking alcoholic through Alcoholic Anonymous (AA). An apprentice attend several meetings a week, spending that time in the company of near-peers and adepts, those whose practice and identity are the community of AA. Goals are made plain in the litany of the *12 steps to sobriety* which guide the process of moving from peripheral to full participation in AA. The initial step into the AA community is the silent gesture of taking a

white chip from the table, to indicate that you will not drink the next 24 hours. The last step is to visit an alcoholic and try to persuade them to join the AA. Cain argues that the main activity of AA is the reconstruction of identity, through the activity of constructing personal life-stories, and with them, the meaning of the teller's past and future action of the world. By identity he means the way a person understands and views himself, and is viewed by others, a perception of self which is fairly constant. The identity of an AA is made up of "Alcoholic", which is an irreversible state, and the fact that one is a member of AA, which is a negotiation of the behavior which made one qualified to join. The life story are told for the explicit stated purpose of providing a model for alcoholism, so that other drinkers may find so much of themselves in others lives, that they can not help to ask themselves if they are, too, alcoholics. This is a negotiated process between the drinker and those around her. Telling the story is not something one learns through explicit teaching. First the member must be exposed to AA models through literature, meetings and talking to individual old-timers. In addition to learning from the models, learning takes place through interaction. The members are encouraged to talk to others and to maintain friendship with other AA members. In discussion meetings specific topics are discussed, in bits and pieces, rather then the entire life. The next speaker builds upon what was said by the previous speaker, and if the interpretations and evidences is according to the AA beliefs it is not corrected. The propositions about alcohol must be applied to the members own life. One learns to perceive oneself and one´s problems from an AA perspective. Stories are tools for reinterpretation of the past, and to understand the self in terms of the AA identity.

3.1.1.Practice, person, social world and internalization of the cultural given

All theories of learning are based on fundamental assumptions about the person, the world, and their relations. Conventional explanations view learning as a process by which a learner internalizes knowledge, whether "discovered", "transmitted" from others, or "experienced in interaction" with others. Lave and Wenger hold that this view leaves the nature of the learner, the world, and of their relations unexplored, and it establishes a sharp dichotomy between inside and outside. It takes the learner as the non-problematic unit of analysis. They say that learning as internalization is too easily construed as an unproblematic process of absorbing the given, as a matter of transmission and assimilation. The psychologist Lev Vygotsky˙s theory of *Zone of Proximal Development* (ZDP) (1978) has been interpreted in various ways, which according to Lave and Wenger can be categorized in three different views. 1) ZDP is

characterized as the distance between problem solving abilities exhibited by a learner working alone and that learners' problem solving abilities when assisted by or collaborating with more-experienced people. 2) A "cultural" interpretation construes the ZPD as the distance between the cultural knowledge provided by the socio-historical context – usually made accessible through instruction – and the everyday experience of individuals. Hedegaard (1988) calls this the distance between understood knowledge, as provided by instruction, and active knowledge, as owned by individuals. This interpretation is based on scientific and everyday concepts and that a mature concept is achieved when scientific and everyday versions have merged. 3) The third interpretation of ZDP is in the context of contemporary development in the traditions of Soviet psychology and critical psychology and takes a collectivistic or societal perspective. Engestrøm (1987) defines the ZPD as "the distance between the everyday actions of individual and the historically new form of the societal activity that can be collectively generated as a solution to the double bind potentially embedded in ... everyday actions." Under such societal interpretations of the concept of ZDP researchers tend to concentrate on processes of societal transformation.

In contrast with learning as internalization, Lave and Wenger see learning as increasing participation in communities of practice that concerns the whole person acting in the world. Theorizing about social practice, praxis, activity and the development of human knowing through participation in an ongoing social world, is part of a long Marxist tradition in the social sciences. A theory of social practice emphasizes the relational interdependency of agent and world, activity, meaning, cognition, learning and knowing. Participation is always based on situated negotiation and renegotiating of meaning in the world. This implies that understanding and experience are in constant interaction. The notion of participation thus dissolves dichotomies between cerebral and embodied activity, between contemplation and involvement, between abstraction and experience: person's actions, and the world are implicated in all thought, speech, knowing and learning (Lave and Wenger, 1991).

3.1.2.The person and identity in learning

Participation in social practice suggests a very explicit focus on the person, but as a person-in-the-world, as member of a socio-cultural community. This focus in turn promotes a view of knowing as activity by specific people in specific circumstances. Activities, tasks , functions and understandings do not exists in isolation; they are part of a broader system of relations in which they have meaning. Viewing learning as LPP means that learning is not merely a

condition for membership, but is itself an evolving form of membership. Lave and Wenger conceive of identity as long-term living relations between persons and their place and participation in communities of practice. Thus identity, knowing, and social membership entail one another. One of the first things one associates with apprenticeship is the master-apprentice relation. In practice the roles of masters are surprisingly variable across time and place (ibid). Quartermasters does not learn in a master-apprentice relation, but in relationships with their more experienced peers. AA newcomers has special relations to specific old-timers who act as their sponsors. Lave and Wenger says that it should be clear that, in shaping the relation of masters to apprentices, the issue of conferring legitimacy is more important than the issue of providing teaching. In all of the examples of apprenticeship that Lave and Wenger gives, the researchers insist that there is very little observable teaching, the more basic phenomenon is learning. The practice of the community creates the potential "curriculum" in the broadest sense – that which may be learned by newcomers with legitimate peripheral access. There are strong goals for learning because learners, as peripheral participants, can develop a view of what the whole enterprise is about, and what there is to be learned. In apprenticeships opportunities for learning are, more often than not, given structure by work practices instead of by strongly asymmetrical master-apprentice relations. It seems typical of apprenticeship that apprentices learn mostly in relations with other apprentices. There is anecdotal evidence (Butler, personal communication with Hass, in Lave and Wenger, 1991) that where the circulation of knowledge among peers and near-peers is possible, it spreads exceedingly rapidly and effectively. The effectiveness of the circulation of information among peers suggest, to the contrary, that engaging in practice, rather then being its object, may well be a condition for the effectiveness of learning. Structuring resources for learning come from a variety of sources, not only from pedagogically activity. Lave and Wenger take a de-centered view of master-apprentice relations which they propose leads to an understanding that mastery resides not in the master but in the organization of the community of practice in which the master is part; the master as the locus of authority (in several senses) is as much a product of the conventional, centered theory of learning, as is the individual learner. Similarly, a de-centered view of the master as pedagogue, moves the focus of analysis away from teaching and onto the intricate structuring of a communities learning resources.

3.1.3.The place of knowledge - participation and learning curricula

The social relations of apprentices within a community change through their direct involvement in activities; in the process, the apprentices' understanding and knowledgeable skills develop. Conventional speculations about the nature of "informal learning" is apprentices are supposed to acquire the "specifics" of practice through observation and imitation. Lave and Wenger argues instead that the effect of peripheral participation on knowledge-in-practice are not properly understood; and that studies of apprenticeship have presumed to literal a coupling of work process and learning process. They say that LPP provides newcomers with more than an "observational" lookout post: it involves participation as a way of learning, of both absorb and being absorbed in - the "culture of practice." Apprentices gradually assemble a general idea of what constitutes the practice of the community. This uneven sketch of the enterprise (available if there is legitimate access) might include who is involved, what they do, what everyday life is like; how masters talk, walk, work, and generally conduct their lives; how people who are not part of the community of practice interact with it; what other learners are doing; and what learners need to learn to become full practitioners. It includes an increasing understanding of "how, when, and about what old-timers collaborate, collude, and collide, and what they enjoy, dislike, respect, and admire" (Lave and Wenger, 1991:95). In particular, it offers exemplars (which are grounds and motivation for learning activity) including masters, finished products, and more advanced apprentices in the process of becoming full practitioners. Viewpoints from which to understand the practice evolve through changing participation in the division of labour, changing relations to ongoing community practices, and changing social relations in the community if peripheral, less intense, less complex, less vital tasks are learned before more central aspects of practice.

A learning curriculum is a field of learning resources in everyday practice, viewed from the perspective of learners. A teaching curriculum, by contrast, is constructed for the instruction of newcomers. When a teaching curriculum supplies - and thereby limits - structuring resources for learning, the meaning of what is learned (and control of access to it) is mediated through an instructor´s participation, by an external view of what knowing is about. A learning curriculum is essentially situated. It is not something that can be considered in isolation, or analyzed apart from the social relations that shape LPP. A learning curriculum is thus a characteristic of a community. By the term "community" Lave and Wenger do not only mean the primordial culture-sharing entity. Nor does the term imply necessarily co-presence,

a well-defined, identifiable group, or socially visible boundaries. What it do imply is participation in an activity-system, about which participants share understandings concerning what they are doing and what that means in their lives and for their communities.

3.1.4.The problem of access and transparency

According to Lave and Wenger the key to LPP is access by newcomers to the community of practice and all that membership entails. To become a full member of a community of practice requires access to a wide range of ongoing activity, old-timers, and other members of the community; and to information, resources, and opportunities for participation. The artifacts employed in ongoing practice, the technology of practice, provide a good arena in which to discuss the problem of access to understanding. Participation involving technology is especially significant because the artifacts used within a cultural practice carry a substantial portion of that practices heritage. The artifacts may have evolved over a long time, like the alidade used by quartermasters which embodies calculations invented a long time ago. Thus, understanding the technology of the practice is more than learning to use tools; it is a way to connect with the history of the practice and to participate more directly in the cultural life. Transparency in its simplest form may just imply that the inner workings of an artifact are available for the learner's inspection (e.g. open source-code). The black box can be opened, it can become a glass box. The transparency of any technology always exists with respect to some purpose and is intricately tied to the cultural practice and social organization within which the technology is meant to function; it can not be viewed as a feature of an artifact in itself but as a process that involves specific forms of participation, in which the artifact fulfills a mediating function. The quartermaster apprentices' does not only participate in the physical activities, but they participate in information flows and conversations, where they can make sense of what they hear and observe.

3.1.5.Motivation and identity, contradictions and change

In the cases of apprenticeship mentioned above it is true that the initial, partial contributions by apprentices are useful. Even the AA newcomer, while reinterpreting his life, produces new material that contributes to the communal construction of the understanding of alcoholism. As opportunities for understanding how well or poorly one´s efforts contribute are evident in practice; legitimate participation of an peripheral kind provides an immediate ground for self-evaluation. Moving towards full participation in practice involves not just a greater

commitment of time, intensified effort, more and broader responsibilities within the community, and more difficult and risky tasks, but more significantly, an increasing sense of identity as a master practitioner (Lave and Wenger, 1991).

The continuity-displacement contradiction is present during apprenticeship, whether apprentice and master jointly have a stake in the increasingly knowledgeable skill of the apprentice, as among the tailors and midwives, or whether there is a conflict between the master's need for labour and the apprentice's desire to learn, as among the meat cutters. Newcomers are caught in a positive dilemma, they have to engage in the existing practice, which has developed over time, to understand it, participate in it, and become members in it. On the other hand they have a stake in its development as they begin to establish their own identity in its future.

Communities of practice have histories and developmental cycles, and reproduce themselves in such a way that the transformation of newcomers into old-timers becomes unremarkably integral to the practice. Knowing is inherent in the growth and transformation of identities and it is located in relations among practitioners, their practice, the artifacts of that practice, and the social organization and political economy of communities of practice.
For newcomers their shifting location as they move centripetally through a complex form of practice creates possibilities for understanding the world as experienced. Denying access and limiting the centripetal movement of newcomers and other practitioners change the learning curriculum. This raises questions in specific settings about what opportunities exists for knowing in practice: about the process of transparency for newcomers. All this takes place in a social world, dialectically constituted in social practices that are in the process of reproduction, transformation, and change.

Lave and Wenger˙s analytical framework will serve as a theory for discussing learning in the Plone community of practice. The concepts of LPP seem promising to point at issues that might arise when a newcomer tries to enter the practice. In this research I also want to look at the use of a specific medium of communication, Internet Relay Chat (IRC). I hold that this channel, together with the mailing-list and sprints, is where sharing of tacit knowledge is best facilitated.

In LPP there is little focus on different artifacts for learning and knowledge-sharing, and how the community organizes knowledge. To be able to talk about how and where knowledge is

created and shared, and to look at the duality of structured formal knowledge and more unstructured, tacit knowledge, I will refer to an article written by Paul M. Hildreth and Chris Kimble (2002), and a theory of organizational knowledge-conversion proposed by Ikujiro Nonaka (2002). I will now present these theories which will be used in the last part of my analysis. As I move on to this part of the theory the focus will change from individual learning to how the community, as an organization, learns and how knowledge circulates in the community.

3.2.Knowledge management (KM) and organizational knowledge

The term "knowledge" suffers from a high degree of terminological ambiguity and needs a well described definition, more than just "what people know". The word knowledge combined with the word management often makes people want to quantify, count, organize and measure knowledge in a way that make it seem like an object that can be captured and stored. In this perspective one can argue that knowledge is seen as only information, and earlier knowledge management (KM) projects was concerned with creating expert systems to organize the knowledge objects in databases, books and manuals. Kimble and Hildreth (2002) argues that the view of knowledge as an object continues to dominate the KM field, with some researchers still viewing the capture of knowledge as the main challenge (Alavi and Leidner, 1997). This view is technology-centered and the main role for technology is to create a repository of structured knowledge. Recently there has been a trend towards recognizing that there are aspects of knowledge which cannot be articulated, abstracted, codified, captured and stored. This is sometimes called less structured knowledge. There is an ongoing debate about what constitutes knowledge (Kimble and Hildreth, 2002). This paper will not engage in this struggle but will take the position that, knowledge, what people know, consists of both structured and less structured knowledge. I will use Hildreth and Kimble's (2002) terms "hard knowledge", as what can be articulated, and "soft knowledge" as what cannot be articulated. Many KM researchers see knowledge as a dichotomy. Conklin (1996) uses the terms formal and informal. Formal knowledge is found in books, manuals and documents and can be easily shared at training courses. Informal knowledge is the knowledge applied in the process of creating formal knowledge. Rulke, Zaheer and Anderson (1998) focus on the knowledge of an organization, which they term transactive knowledge (knowing what you know) and resource knowledge (knowing who knows what). Kogut and Zander (1992) make a distinction between information and know-how, while Brown and Duguid (2000) make a distinction

between know-how and know-what. All these views see knowledge as either of the two. Leonardo and Sensiper (1998), on the other side, describe knowledge as a continuum and say that it exists on a spectrum from almost completely tacit (semi- and unconscious knowledge held in peoples head and bodies) on the one extreme, and as almost completely explicit and codified on the other extreme. The dictionary definition of tacit knowledge is "that which is understood without being openly expressed; it is unvoiced or unspoken" (Oxford American dictionaries). An example might be the knowledge that a native speaker has of a language. Explicit knowledge is that which can be expressed clearly, fully and leaves nothing implied. An example might be knowledge that can be formally expressed through manuals, specifications, regulations, rules or procedures. Polanyi (1967) is often cited on tacit knowledge and he sees it as subtle conception rooted in cognitive schemata referred to as mental models. He proposed a concept of knowledge based on three main thesis:

- First, true discovery cannot be accounted for by a set of articulated rules or algorithms
- Second, knowledge is public but is also to a large extent personal (i.e. it is socially constructed)
- Third, the knowledge that underlies explicit knowledge is more fundamental; all knowledge is either tacit or rooted in tacit knowledge.

Thus, for Polanyi, tacit or implicit knowledge is knowledge that is known but cannot be told; we know more then we can tell. This kind of knowledge is internalized in our unconscious mind. To look further into the soft/hard, explicit/tacit dimension and how this view on knowledge can be applied in the Plone community I will use Nonaka´s *dynamic theory of organizational knowledge creation* (2002). To prevent misunderstandings I will underscore that the focus is moved from how individuals learn to how the the organizations learn.

3.2.1.Organizational knowledge-creation and knowledge-conversion

An interesting view on knowledge-creation and knowledge-sharing is provided by Nonaka (1991) in "*A dynamic theory of organizational knowledge creation*". At the heart of Nonaka´s work is the premise that there are two types of knowledge: tacit and explicit. Tacit knowledge is subjective and experience-based knowledge that can not be expressed in words, sentences, numbers or formulas, often because they are context specific. This also includes cognitive skills such as beliefs, images, intuition and mental modes as well as technical skills such as craft and know-how. Explicit knowledge is objective and rational knowledge that includes theoretical approaches, problem solving, manuals and databases. He sees tacit and explicit knowledge not as separate, but as mutually complementary entities. They interact with each other in the creative activities of human beings like in a community of practice. Nonaka calls

the interaction of these two forms of knowledge the “knowledge-conversion process”. This process consists of four stages: socialization, externalization, combination and internalization:

- *Socialization* transfers tacit knowledge between individuals through observation, imitation and practice. It is experimental, active and a “living thing”. This depends on having shared experience and results in acquired skills and common mental models. This stage is primarily a process between individuals and LPP and apprenticeship are examples of this stage.

- *Externalization* is triggered by dialogue or collective reflection and relies on analogy or metaphor to translate tacit knowledge into documents and procedures. One case is the articulation of one's own tacit knowledge such as ideas and images in words, metaphors and analogies. A second case is eliciting and translating the tacit knowledge of others, customer, experts for example – into a understandable form. Dialogue is an important means for both, during face to face communication people share beliefs and learn how to better articulate their thinking. Externalization is a process among individuals in a group.

- *Combination* consequently reconfigures bodies of explicit knowledge through sorting, adding, combining and categorizing processes and spreads it throughout an organization. A diversity of knowledge sources is combined to shape a new and enhanced conception. In this area information technology is most helpful, because explicit knowledge can be conveyed in documents, email, databases as well as through meetings and conferences. A diversity of combination allows knowledge transfer among groups across organizations.

- *Internalization* is the process of understanding and absorbing explicit knowledge into tacit knowledge held by the individual. Knowledge in the tacit form is actionable by the owner. It is experimental in order to actualize concepts and methods, either through the actual doing or through simulations. The internalization process “transfer” explicit knowledge into the individual.

Eventually, through a phenomenon that Nonaka calls the "knowledge spiral", knowledge creation and sharing become part of the culture of an organization. Nonaka states that true knowledge – actionable understanding – comes from a gut-level commitment and belief, therefore, building and conveying knowledge requires shared emotion, feeling, mental models, experiences, and what he calls “empathy space”.

3.2.2.Two dimensions of knowledge

Nonaka (1994) sees knowledge as a multifaceted concept with multilayered meanings. He points out that the history of philosophy since the classical Greek period can be regarded as a never-ending search for the meaning of knowledge. His *Dynamic Theory of organizational Knowledge Creation* (1994) follows traditional epistemology and adopts a definition of knowledge as "justified true beliefs". Nonaka is also interested in the difference of information and knowledge and states that information is a flow of messages, while knowledge is created and organized by the very flow of information, anchored on the commitment and beliefs of its holder. Relatively little attention has been paid to how knowledge is created and how the knowledge process can be managed. Following the assumption that knowledge is created through conversion between tacit and explicit knowledge, Nonaka postulates, the already mentioned, four modes of knowledge-conversion in an organization.

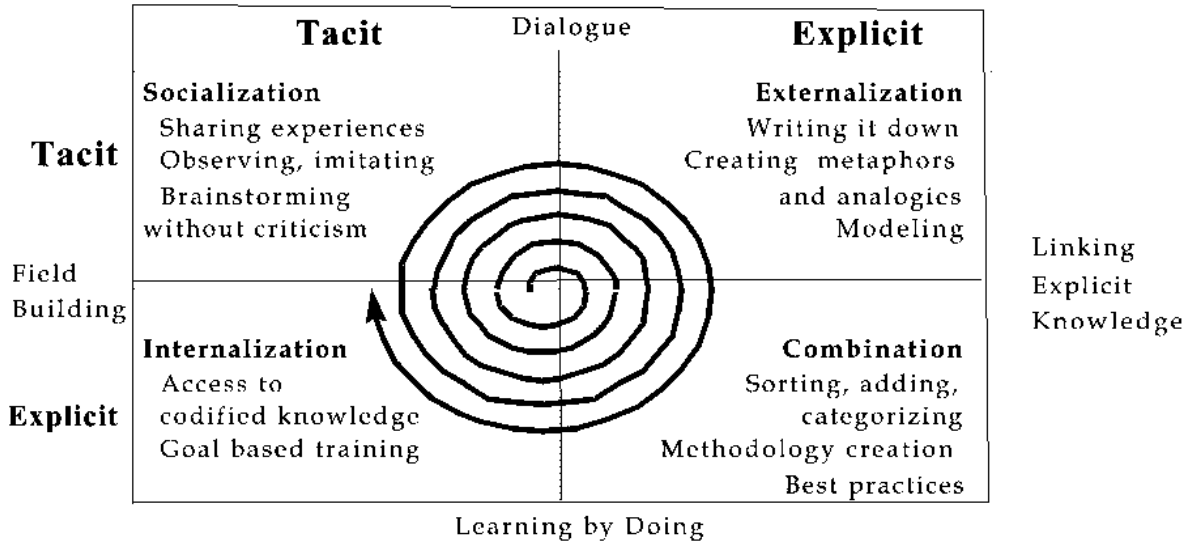

Illustration 3: Nonaka´s spiral of knowledge (Hildreth and Kimble, 2002)

Illustration 3 shows Nonaka´s four modes of knowledge-conversion. Knowledge in an organization is shared in the spiral described above.

This theory can be used when analyzing the Plone community by looking at how and where knowledge is created and shared, and to describe the knowledge-conversion that enables the community to convert tacit knowledge through interaction. The key to acquiring tacit knowledge is experience. Nonaka calls this process of creating tacit knowledge through

shared experience *socialization*. The second mode of knowledge-conversion involves the use of social process to combine different bodies of explicit knowledge held by individuals. In the Plone community individuals exchange and combine knowledge through such exchange mechanisms like virtual and physical meetings and electronic discussions. This process of creating explicit knowledge from explicit knowledge Nonaka calls *combination*. The third and fourth modes of knowledge-conversion relate to patterns of conversation involving both tacit and explicit knowledge and capture the idea that tacit and explicit knowledge are complementary and can expand over time through a process of mutual interaction. This interaction involves two different operations. One is the conversion of tacit knowledge into explicit, which Nonaka calls *externalization.* Tacit knowledge may be transformed into explicit knowledge by recognizing contradictions through metaphor like the prototypes of products or use-cases, and resolving them through analogy. The other is the conversion of explicit knowledge into tacit knowledge, which bears some similarity to the traditional notion of learning and is called *internalization*. Nonaka notes that action is deeply related to the this process.

3.2.3.Commitment - intention, autonomy, and fluctuation

Nonaka holds that individuals are continuously committed to recreating the world in accordance with their own perspectives. Polanyi (1967) noted that commitment underlies human knowledge-creating activities, thus commitment is one of the most important components for promoting the formation of new knowledge within an organization. Nonaka names three basic factors that induce individual commitment in an organizational setting:

- *Intention*. Intention is concerned with how individuals form their approach to the world and try to make sense of their environment. It is an action-oriented concept, and Husserl (1968, in Nonaka 1994) called this attitude on the part of the subject “intentionality”. He argued that consciousness arises when a subject pays attention to an object, that any consciousness is a consciousness of something. It arises, endures, and disappears with a subject's commitment to an object. (Nonaka, 2002)

- *Autonomy*. Every individual has his or her own personality. By allowing people to act autonomously, the organization may increase the possibility of introducing unexpected opportunities of the type that are sometimes associated with the so-called “garbage can” metaphor (Cohen et al., 1972). Individual autonomy widens the possibility that individuals

will motivate themselves to form new knowledge. Self-motivation based on deep emotions, for example in the poet's creation of new expressions, serves as a driving force for the creation of metaphors. Autonomy gives individuals freedom to absorb knowledge.

• *Fluctuation*. Knowledge creation at the individual level involves continuous interaction with the external world. In this connection, chaos or discontinuity can generate new patterns of interaction between individuals and their environment. These fluctuations differ from complete disorder and are characterized by “order without recursiveness” - which represents an order where the patterns is hard to predict in the beginning (Gleick, 1987). When breakdown occurs, individuals question the value of habits and routine tools, which might lead to a realignment of commitments. Environmental fluctuation often triggers this breakdown. When people face such a breakdown or contradiction, they have an opportunity to reconsider their fundamental thinking and perspectives.

3.2.4.The duality of soft and hard knowledge

A question could be asked about Nonaka’s tacit-explicit stage; if tacit knowledge is inarticulable how can it be transfered at all in socializing ? There are mainly two different views of the possibility of articulating tacit knowledge. The first, following Polanyi, states that it is not possible to externalize tacit knowledge. The other, following Teece (1998), holds that it is merely difficult to articulate. In social sciences this is known as the say-do problem; people know how to do a thing but it is not possible to explain it in words. In Nonaka's spiral of knowledge, tacit knowledge is 'shared' through interpersonal interaction. However, the tacit knowledge is not articulated and shared; the learner actually develops their own tacit knowledge by becoming immersed in the practice itself, under the guidance of a mentor and while situated in a particular environment. Hildreth and Kimble (2002) state that most researchers involved in the field of KM see knowledge in terms of opposites. In an attempt to move the debate forward, they explore the term soft knowledge as a precursor to their thesis that knowledge is in fact a duality. In Lave and Wenger´s theory of LPP (1991) newcomers learn the practice of the community by being situated in it, and learn from its established members and becomes established members themselves. LPP allows the development of both hard and soft knowledge. Hard knowledge can be articulated and may be exemplified by tasks the members of a community of practice perform. Soft knowledge is the kind of knowledge that the newcomer cannot learn simply by demonstration or instruction. It includes learning the language and unspoken conventions of the community. Soft knowledge is developed and

learnt through being socialized into the community and through interaction with the existing members. Hildreth and Kimble (2002) propose to view knowledge as a duality instead of a dichotomy. If we view knowledge as a duality then by implication, all knowledge is to some degree both hard and soft; it is simply that the balance between the two varies. When the harder aspects are abstracted in isolation, like in earlier KM projects, the representation is incomplete; softer aspects of knowledge must also be taken into account. Hargadon (1998) gives the example of a server holding past projects, but developers do not look there for solutions. As they put it, 'the important knowledge is all in people's heads', the solutions on the server only represent the harder aspects of the knowledge (ibid). For a complete picture, the softer aspects are also necessary. Critics have argued that Nonaka´s distinction between tacit and explicit knowledge is oversimplified, and even that the notion of explicit knowledge is self-contradictory. His theory proposes a paradigm for managing the dynamic aspects of the organizational knowledge creation processes, and I find the concepts useful.

3.2.5.Practice - participation and reification

Viewing knowledge as a soft-hard duality begs the question - how can this duality be managed? Finerty (1997) points out that technology has a role to play, but that the emphasis needs to move from trying to package knowledge as an object, to using technology as a way of sharing experience. This view is supported by Davenport and Prusak (1998) who emphasize the potential of technology as a means to create links between people. However, most technologies currently remain focused on the sharing of abstracted, harder aspects of knowledge in the form of reports and documents. Recent work by Wenger (1998) revisits his earlier work (Lave and Wenger, 1991) in the light of CoPs moving into the business environment. Wenger considers the practice of a CoP to be much more than the everyday practice of a community. He prefers to explore practice as meaning in particular context; "practice is about meaning as an experience of everyday life" (Wenger, 1998: 52). He states that meaning as an experience is located in a process he calls *the negotiation of meaning*. The negotiation of meaning involves the interaction of two processes, participation and reification, which forms a duality (Illustration 4). Having concentrated on the participation dimension of LPP, Wenger (1998) points out that this remains undefined without the other constituent process that makes up the negotiation of meaning: *reification* - giving concrete form to something that is abstract. Wenger emphasizes that participation and reification is analytically separable, but in reality is a single duality - one cannot replace the other. He uses the concept of reification:

> *...to refer to the process of giving form to our experience by producing objects that congeal this experience into 'thingness' ... With the term reification I mean to cover a wide range of processes that include making, designing, representing, naming, encoding and describing as well as perceiving, interpreting, using, reusing, decoding and recasting. (Wenger, 1998: 58-59)*

The participation/reification duality maps very closely to the earlier hard knowledge/soft knowledge duality proposed by Hildreth et al, (1999). Knowledge was said to be a soft/hard duality in that all knowledge was hard and soft - it is simply the proportions that differ. This is also true of participation and reification. If knowledge is predominantly soft then the participation proportion of the duality will be higher. Conversely; the harder the knowledge, the greater the proportion of reification.

> *Reificative connections can transcend the spatio-temporal limitations inherent in participation. We cannot be all over the world, but we can read the newspaper. We cannot live in the past, but we can wonder at monuments left behind by long-gone practices. In this respect reificative connections afford seemingly limitless possibilities. (Wenger, 1998: 110)*

An important aspect of the participation/reification duality is balance between each of the constituent processes. Each needs to be in its proper proportion so that each remains in equilibrium with the other:

> *If participation prevails - if most of what matters is left unreified - then there may not be enough material to anchor the specifities of coordination and to uncover diverging assumptions. This is why lawyers want everything in writing.(Wenger, 1998: 65.)*

While,

> *If reification prevails - if everything is reified but with little opportunity for shared experience and interactive negotiation - then there may not be enough overlap in participation to recover a coordinated relevant or generative meaning. This helps explain why putting everything in writing does not seem to solve all our problems. (Wenger, 1998: 65.)*

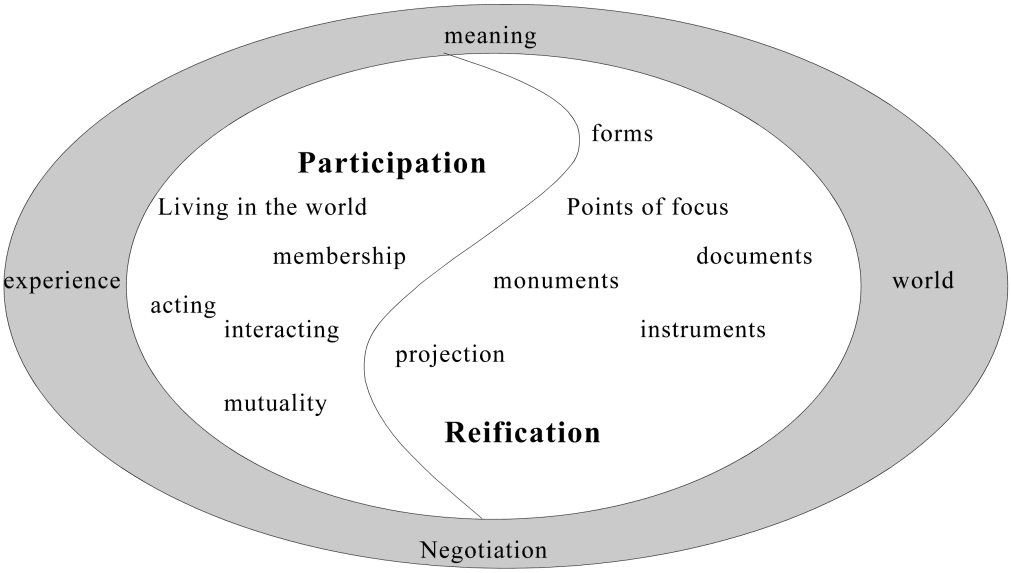

Illustration 4: The Duality of participation and reification (Wenger, 1998)

Wenger shows this idea of practice in illustration 4 where examples of participation and reification is placed in context of meaning, world, negotiation and experience.

The theory of LPP can provide useful concepts in the analysis of social aspecs of activities and process´ of learning in FLOSS. Nonaka`s "knowledge-conversion process" leads our focus to where and how knowledge is created and shared, throughout the community. I assume that the participation/reification duality is suitable to talk about the balance between types, or aspects, of knowledge in the Plone community of practice. I will now go on to explain the research method and how the research is designed, before I go on to the analysis.

4.METHOD AND RESEARCH DESIGN

"Where is the knowledge we have lost in information?"
—T. S. Eliot(1888-1965), The Rock

The theories I use in this thesis is reflected in my choice of method. Because I use socio-cultural learning theory, it is necessary to design the research in a way that might provide the best possible data for examining the concepts of the theories. Data collected by typical qualitative methods such as fieldwork and interviews gives us rich material to interpret and understand the practice of the community. In the previous chapter I showed that the keywords to understand learning and knowledge are participation, practice, collaboration, identity, relationships, artifacts and communicative processes. It is difficult to capture the meaning of the mentioned concepts by only observing the community from the outside. The method of virtual ethnography is used in an attempt to describe the community from the inside. The fieldwork lets me reflect on various concept of the theories at the same time as I participate. The IRC channels is the most used channels for the negotiation of meaning, and data from the **#plone** IRC channel is used as example of communication. Interviews with core developers; participants that may be said to have achieved full participation in the Plone community of practice, serves as rich reports from the center of the developer community

Choosing the right method

The social view of learning and knowledge creation promotes the idea that knowledge is deeply embedded in the technological and social context of a community (Nonaka and Konno 1998; von Krogh, Ichijo et al. 2000). In conceptualizing ways how to enable creation and sharing of knowledge on-line, I draw on the literature on communities of practice (Lave and Wenger 1990; Brown and Duguid 1991; Wenger 1998; Wenger 2000). My assumption is that in a successful open source project there are well established practices for handling decision making, enrolling/attracting new members, problem solving, knowledge-sharing and other organizational and collaborative issues. I assume that a FLOSS project of this size has its ways to handle creation and handling of knowledge and information. In its simplest form it is code commenting where the developer, by reflection-on-action, leaves messages in the code to other programmers and to him-/herself, or knowledge-sharing through agents, like auto-generated emails activated on submitting code to the svn. The other end of this scale is when end-users produce detailed flash video tutorials on how to do simple user actions like adding content to a page. The main discourse happens through discussions on mailing lists and through conversations on IRC. The everyday routine-work and practice is designed in a way

that stores information that other people can transform into knowledge later. By framing interaction, and looking at the channels of communication, I will focus on learning in the socio-technical network. I will analyze how the tools used for different kind of communication offers possibilities for learning. The gathering of empirical data to be used in the analysis, will be conducted according to the methodologies I will present in this chapter.

The gathering of empirical data for the project can be divided into three parts:

- Ethnographic fieldwork
- IRC logs
- Interviews

Although I use several different data I do not triangulate them. The main data source is my notes and experiences from doing fieldwork. I use IRC-logs to focus specifically at the stage of tacit-to-tacit knowledge-conversion in synchronous communication. Interviews are used to look at the participants' view of themselves in the community, and how they define themselves as part of it.

4.1.Virtual ethnography: participant observation

Ethnographic research originates from the discipline of social and cultural anthropology. A researcher doing ethnography is required to spend a significant amount of time in the field. Ethnographers immerse themselves in the lives of the people they study and seek to place the phenomena studied in their own social and cultural context (Lewis 1985). After early ground-breaking work by Wynn (1979), Suchman (1987) and Zuboff (1988), ethnography has now become more widely used in the study of information-systems in organizations, from the study of the development of information systems to the study of aspects of information technology management. Hammersly and Atkinson (1995) describe ethnography in this way:

> *In its most characteristics form it involves the ethnographer participating, overtly or covertly in peoples daily lives for an extended period of time, watching what happens, listen to what is said, asking questions - in fact, collecting whatever data are available to throw light on the issues that are the focus of the research. (Hammersly and Atkinson, 1995:1)*

The field-site for this research is challenging to both locate and define. Mailing list, IRC channels, web-pages, source code and code repositories, peers and participants, automated agents, my computer, conferences, the network and technologies are all part of the field. Instead of "field" it might be fruitful to talk about "connectivity" and "flow between nodes" (Hine, 2000). During the fieldwork I used different tools and moved between different protocols and channels on the network. Doing ethnography on the Internet can be viewed in different ways. Rich and sustained interaction could be viewed as constituting cultures in their own right. Internet can either be seen as a cultural artifact or a cultural product that is shaped in the ways it is marketed, thought and used. Early works in the field of CMC were mostly concerned with how the social cues where reduced and were basically comparing CMC to face-to-face communication. In my opinion CMC can be seen as a context for social relations in its own right. Between the poster of one message, on IRC or at the mailing-list, and the author of a response, a space opens, and that space is a cultural context (ibid). Communities which "live" on the Internet are often referred to as virtual communities:

> *virtual communities are social aggregations that emerge from the net when enough people carry on those public discussions long enough, with sufficient human feeling, to form webs of personal relationships in cyberspace. (Rheingold 1993:5)*

It can be discussed if the the Plone community is a virtual community, but the communication share many similarities with virtual communities. What separates the Plone from other virtual communities, such as BBS discussion groups, or online role-playing games, is the shared work-practice, and the knowledge connected to it.

4.1.1.Autoethnography

Autoethnography is an emergent ethnographic writing practice which involves highly personalized accounts where authors draw on their own experiences to extend understanding of a particular discipline or culture. Although I myself is not a major character in this paper, I place the writing from my LPP partly in the autoethnography genre. Autoethnography is a genre of writing and research that connects the personal to the cultural, placing the self within a social context (Reed-Danahay, 1997). This type of text are usually written in first person and feature dialogue, emotion, and self-consciousness, and the author use their own experiences in a culture, reflexively, to look more deeply at self-other interactions. By writing themselves into their own work, they challenge accepted views about silent authorship, where the voice of the researcher is not included in the presentations of findings (e.g. Charmaz and Mitchell,

1997). A problematic issue of autoethnography arises when one uses “the self” as the only data source. Autoethnographers have been criticized for being to self-indulgent and narcissistic (Coffey, 1999), and it has even been suggested that it is at the boundaries of academic research. Autoethnography, however, is not necessarily limited to the self because people do not accumulate their experience in a social vacuum. Social learning theory focuses on exactly the self in a social setting. In this method, as well as more general in ethnography two important issues concerning qualitative research can be mentioned; the dual crisis of representation and legitimation. The crisis of representation refers to the writing practice (i.e how researchers write and represent the social world). The crisis of legitimation questions traditional criteria used for evaluation and interpreting qualitative research, involving terms such as validity, reliability and objectivity (Holt, 2003).

Many FLOSS research projects seem to be heavily theoretical and in my opinion sometimes the practice (what is going on) seems to slip between the fingers of the researcher. The Plone community of practice is the main analytic entity of my research. I follow Wolcott (1994) who suggests that qualitative researchers need to be story-tellers, and story-telling should be one of their distinguishing attributes.

The main goal for my virtual ethnographic fieldwork is to take part in the material and intellectual artifacts used in the Plone community. Through hermeneutic interpretation I aim to gradually develop an understanding of what is going on. A difficulty of this way of working was to document my own discoveries in my loops in the hermeneutic circles. As I gradually developed a holistic understanding, it was very hard to see the single components I discovered and reflected over, and then used to adjust my views and presumptions. In the field one picks up a huge amount of information, some of it unconscious. Even though it is impossible to make account for all the information, these bits contribute to the holistic view of the situation and we can have a feeling of what is going on. This feeling can lead to a form of intuition which might be essential for the success of the research in that it influences what to look for and ask about, and what to examine closer, and might lead to a discovery we not yet know we are going to do (Polanyi 1966).

I use the methodological tradition of ethnography to study FLOSS as a social phenomena and try to link my findings back to relevant pedagogical theory such as LPP, and organizational knowledge theory.

4.1.2.Criticism of ethnography

A common criticism of ethnography has been that the ethnographers bring their own analytical frameworks with them and therefore fails to address the field site they visit on their own terms. Two ethnographers visiting the same place can come back with a different notion of the cultural reality of the place. Another issue is if electronic ethnography can be authentic. It is difficult to check if what the informant tell the researcher is true. A more specific issue concerning this, is if the correspondence between the real-world, or off-line, identity and the virtual identity matches. An example of a case where it does not match would be what is known as gender-swapping; that a male or female pretend to be of the opposite sex. This critique implies that there is *one* identity in the other end of the line. The theory of LLP argues that we go in and out of different communities and identities, and that identity is negotiated and changed after what community we act in. The identity in Plone is the actual identity of interest for this research project, so for me this does not pose a big problem. Another challenge for the virtual ethnographer is not to go native and take thing for granted, without reflection. At the same time the ethnographer should not only be a disengaged observer, but also share some of the concerns, emotions and commitments of the research subjects. This balance demands some reflections before the fieldwork starts. Another problem is that of the social construct of (knowledge) reality vs. the actual factual knowledge. In the case of ethnography it might be that ethnographic knowledge also is a cultural construct. This is certainly problematic for ethnographers of knowledge production, who might claim to produce objective descriptions; if what scientist thinks of as objective fact, turns out to be the upshot of social process. Hine (2000) suggests three ways this can be solved. The first is to include the views of the objects being studied. By including and focusing upon the ways people perceive, and define, the cultural space within which they exist, and their own in it, these studies therefore view distinctions between external and internal points of view, as processes of life that are contingent upon the particular context in which they are made. (Hastrup and Olwig, 1997: 11 in Hine, 2000) The second is to focus on the researcher and reflect on the particular perspective, history and standpoint that led this ethnographer to be giving their particular account of this setting. This view does not try to be objective, but tries to understand the role of the researcher (subjective). The third approach tries to incorporate a destabilization of ethnographic authority within the text itself. This is an "epistemological correctness" which entails making clear the constructed nature of accounts, and tries to make clear for the reader the constructed and contingent nature. There are various ways of doing

this, and new ways of ethnographic writing has occurred, based on recognition that writing is a constructive act, rather than a straightforward reflection of reality:

> *"[A]ll researchers interpret the world through some sort of conceptual lens formed by their beliefs, previous experiences, existing knowledge, assumptions about the world and theories about knowledge and how it is accrued. The researcher's conceptual lens acts as a filter: the importance placed on the huge range of observations made in the field (choosing to record or note some observations and not others, for example) is partly determined by this filter" (Carroll and Swatman, 2000:118-119, in Silverman, 1999).*

Internet can be said to be a site for social interaction or it can be seen as composed of bits of text. This means that Internet is a collection of text and using the Internet means to read and write text. The ethnographers' job then, is to see the underlying meaning.

4.1.3.Extended participant observation

Doing a study of participation in a social practice, suggests a very explicit focus on the person, but as person-in-the-world, as a member of a socio-cultural community. This focus in turn promotes a view of knowing as activity by specific people in specific circumstances. Activities, tasks, functionality and understanding exist not in isolation, they are part of broader systems of relations in which they have meaning (Lave and Wenger, 1991). My identity as researcher, and my identity as observing participant in another community, are overlapping and influence each other. LPP is more a concept to look at the world through, than an analytical framework, and it is the lens through which I will see the Plone community.

Within a period of 12 months I observed and participated in the Plone community. I participated through using and modifying the software and add-on products, contributing code to a sub-project called Eventregistration, read the OSCOM[1] and other mailing-list, made a Norwegian translation of the Calendar-X[2] product, read how-tos, source-code, readme.txt and other documents, reported bugs, posted and interacted on the mailing list, IRC channel and personal e-mail, and built four Plone sites. My goal was to take part in the practice of the community by learning the language, and by "thinking and learning" like them. This way I also was able to see what skills were needed, and what tools one is required to master to

1 OSCOM is the international association connecting users and developers of Open Source Content Management solutions. http://www.oscom.org/

2 CalendarX is a customizable calendar product for Plone, now i18n features. http://calendarx.org/

become an active user. I also attended the Europython[1] 2005 conference, which had a high number of participants from the Plone community. There I met several people I had seen and heard about on the **#plone** IRC channel. The role as a legitimate peripheral participant embodies the experience of being a new member of the community. The feeling of being an apprentice was funnily remarked by a participant when we were discussing the use of separate databases on one Plone instance and I used some unix commands; "*good boy, my young padowan!*"[2]

Learning is an important motivational factor for many of the participants that join FLOSS projects (von Hippel and von Krogh, 2003). To learn how to use Plone was one of my goals. To ensure the authenticity of my LPP I used Plone to build four different websites. For three of them I had real clients, and the last one I built as a tool for my thesis. By building them I had to learn how to use Plone, Zope and some Python programming. The site I built for my thesis was to help me with my writing process, and was used as a blog for my field notes. One of the other projects, "Kampkunst Portalen" I got involved in through my membership in another community of practice. The two last projects I landed through actively seeking clients, one of them is for a salsa school, and one is for a small company producing landscape models. I chose to do real-life cases to better understand the tools and have a actual situation where my search for knowledge will be as similar as as possible to that of the participants in the community. By building and using these sites I provided myself with an authentic and valid situation in which I had to learn Plone so that I actually could explore the concepts of legitimate peripheral participation.

4.2.Textual interaction/discourse analysis of IRC

Ethnographers are concerned with the *social organization* of documents, irrespective of whether they are accurate or inaccurate, true or biased (Silvermann, 1993). Ethnographers are more concerned with the processes through which texts depict reality, rather than with whether such texts contain true or false statements. The IRC channel is one of the most important communication channels in the Plone project. Because IRC facilitates informal, semi-synchronous communication it is important as a stage for tacit knowledge-sharing

1 EuroPython is an annual conference for the Python and Zope communities. It circulates between different sites in Europe and is run by volunteers. http://www.europython.org/

2 Padowan is the word used for an apprentice in the Jedi Knight Academy in the "Star Wars"-films. A padowan´s master is called Jedi, and this is what the padowan aims to be someday, a master to guard peace and justice in the universe.

through socialization. The IRC log represents a text written (or typed) by several actors in interaction. The document (IRC-conversation) is not the goal of the writing, as i.e. a wiki would be. The immediate feedback of other participants are its most important feature, and the ongoing discussions as well as the short sequences of problem-solving are the goal. Some typical ethnographer's questions about texts can be a starting point for analysis:

How are texts written?
How are they read?
Who writes them?
Who reads them?
For what purpose?
On what occations?
With what outcomes?
What is recorded?
What is omitted?
What is taken for granted?
What does the writer seem to take for granted about the readers?
What do readers need to know in order to make sense of them?
(Hammersly and Atkinson, 1983)

The goal of textual research is not to criticize or to assess particular texts in terms of apparently objective standards. It is rather to analyze how they work to achieve particular effects. The effect I am looking for is the sharing of experience and knowledge, and how learning can happen. The textual entity for my analysis was logged conversations from the **#plone** IRC channel. The logging was done with a dedicated computer that was logging 24 hours a day, 7 days a week.[1] The conversation was automatically generated in a textual format so no transcription was needed. I analyzed the text using categories and a qualitative research tool that use a mark-up system for text analysis[2]. The categories was partly created before I started reading my data, especially categories which I thought was useful for the theory, and partly as I read the data. I tagged the text according to the categories. Doing the categorization

1 In addition to this computer I had another computer that I used to interact on the channel with in the fieldwork. I had two different nicks and the nick I logged in with for logging purposes was never used for interaction.

2 TAMS Analyzer is an open source qualitative package for the analysis of textual themes. It can be used for transcribing digital media and for conducting discourse analysis in the social and cultural sciences. It can be found at http://tamsys.sourceforge.net/

was an iterative process. When a new category was created I had to read the text I already had categorized again, back and forth. Because I had logged several months of IRC conversation I had a huge amount of data, but was only able to read a few percent. The large amount of textual data could be useful for producing quantitative research results as well. Quantitative analysis provide a way of exploring the use of Internet in counting and correlating various features of posted messages. It plays an important role in providing for structured analysis and comparison over settings. Quantitative results is, however, not the aim of the thesis. The categories I used to analyze the text can be found in appendix 3. Content analysis is an accepted method of textual investigation (Silvermann, 1993). In content analysis the researcher establish a set of categories and then count the numbers of instances that fall into each category. I did this in addition to qualitatively analyze what the conversations were about, to be able to see patterns in the communication and what the main activities were.

4.2.1.Collecting textual data from IRC

From February 2005 - May 2006 I logged IRC session from the **#plone**, **#zope** and **#python** IRC channels. The channels are hosted by **freenode.net**[1], a peer-directed project center for open source projects. Two IRC-clients were used during the logging process, X-Chat and Colloquy. Collecting data from the Internet is considered a easy procedure. Since the **#plone** channel is using the IRC protocol, I had the ability to enable detailed logging with a time-stamp. This was important in my data gathering because I could run the IRC client, with logging enabled, without me being active by the computer. This type of logging can be called "single point" (Rintel, Mulholland et al. 2000), meaning that I did not use any other capture tools like video or audio equipment to record interactions outside of the channel. Common to most IRC research, public channel single point logs show the interaction processes in a way that closely resembles the interactions in which the users themselves are engaged. The representations on screen provide largely the same representations of relationships to researchers as they do to users. The search possibilities on the log makes it possible to search for certain persons, words or phrases. I can also easily find all the conversations I myself have joined. I have chosen to analyze natural chat conversation, I will not focus on gender or age, or look for a correct correspondence between the online-identities and the offline-identities, because I wish to look at the actors identities in the context. Because of my anonymity the participants does not change the way they communicate because of me being present. I have not collected

1 The freenode project can be found at http://freenode.net/

additional information about the chat participants in the channel, other than what comes naturally through what I read about them other places, like on **plone.org**, mailing lists, external sites, personal contact and through the products they write. This is related to my field work, where the network, and knowledge of the network I build, influences and constitutes my role as a LPP.

4.2.2.Email interviews

When I was half-way through my gathering of data, and after I had created interview questions and designed the interviews, including setting up an online interview solution with a Plone product and recruited interview objects, Aspeli published his thesis *Plone: A model of a mature FLOSS project*. His most important data was eight semi-structured email interviews with core Plone developers. Aspeli´s role as a core developer and his relation to the Plone community enabled him to ask meaningful on-the-point questions with a high degree of integration. This made the interviews very interesting, and a precise account of the participants involvement in the Plone project, and their reflections on the community's role. Several of the questions he asked was similar to mine, and I decided to actively use his interview data. I needed information about some of the participants, who they are, what they do, and in what ways they are involved in the Plone project, and the interviews gave me exactly what I needed. The main problem with using second hand interviews is the distance between the researcher and the interview objects. Follow-up emails, with corrections and elaborative questions, are difficult to perform. I also wanted to asks some more detailed questions about learning, but with my already huge amount of other data, I could not prioritize this. On the other side Aspeli's short distance to the objects and his integrity in the plone community enabled him to capture viewpoint about important issues from the Plone community, which would have been hard for me to do as a newcomer.

4.2.3.Ethical and practical framework

Doing virtual ethnography is not a question of asking what methods can I use online, and how I can justify the ethics of using them. Above all, the ethical priority of qualitative research is to protect the well being of participants. Because I entered the field covertly, and only a few people from the Plone community knew I was doing research in the field, I had little direct interference on the community. The main object of the fieldwork was my self and my position in the periphery. The communicative processes I got involved in was in focus. The "legitimate

peripheral participant" as object for analysis was myself. The "community of practice" as object for analysis was the publicly available Plone community. Regarding the textual data collected in the **#plone** IRC channel I anonymized all the nicks.

It is important what the participants themselves think of the publicity of what they write in **#plone**. There are probably as many meanings about this as there are participants, but one view of this is illustrated in this citation from **#plone**.

> *Nov 09 16:01:50 majalis wouldn't it be a good idea to have a bot running a logger on this channel and have these conversations indexed on the web?*
> *Nov 09 16:02:15 majalis I'm sure it would represent a good resource for finding solutions to problems*
> ..
> *Nov 09 16:03:54 rosa majalis: this has been discussed on plone-dev and I think the outcome was that although personal logging was okay, not everybody wants their informal conversations indexed so no archive was made*
> ..
> *Nov 09 16:08:44 cyhyb majalis: some of the people here spend considerable amounts of their time and social life here (man, that sounds a lot sadder than it is) and would prefer to not have it all as public information.*
> ..
> *Nov 09 16:10:07 r0bur maybe we should have more people writing documentation instead*
> *Nov 09 16:11:11 majalis there are lots of problems and questions that I can't imagine a time when the Plone universe could be fully documented*

The conversations in **#plone** are public in the sense that everybody can join the channel, and even log the conversation, if they want. I see no big problem in using the conversations, as long as I anonymize the participants. If it were sensitive, personal issues being discussed the project should be submitted to "Personvernombudet for forskning", but my opinion is that the open, technical, non-sensitive character of the information does not call for this. Because it is difficult to reach everybody with information about my research and to get a confirmation that I can use their name, I chose to anonymize all nicknames. Using nicknames provides a relative anonymity, but this is not enough to make the participants untraceable. Actually, many relationships in **#plone** are long term and a strong identity is connected to the nick which is the most vulnerable identifying factor in the community, and it is this identity that

must by anonymized. Regarding the persons I had a more extended relationship with, I contacted them after the data collecting was finished, to ask them if I could use their full name or nick, or if they wanted to be anonymous. Aspeli's interviews had already been clarified with the interview objects, and was published along with his dissertation. I contacted him and told him that I wanted to use his data and he confirmed that I could use them.

5.ANALYSIS

"There are two sides to every question.
Protagoras (485 BC - 421 BC)

The first part of this chapter is about the Plone community of practice and focuses on my own participation and learning, based on data collected from my field study. I analyze my findings according to Lave and Wenger's concepts of legitimate peripheral participation (1991). The second part focuses on the findings from episodes on IRC and the mailing list, and I use the concepts of soft/hard knowledge from Kimble and Hildreth (2004) to look at the duality of knowledge found in the Plone community. The last part looks at communication and knowledge-sharing, by looking at the community as an organization, in the light of Nonaka's knowledge-conversion process (Nonaka, 1994). Aspeli's interviews are used where appropriate throughout the analysis.

5.1.Learning as LPP in Plone

The concept of learning through *legitimate peripheral participation* enables us us to generate analytic terms and questions fundamental to the analysis of a community. As mentioned earlier a community of practice defines itself along three dimensions; what it is about, how it functions and what capability it has produced. By using the three dimensions I will give a short analytical description of the community:

What it is about – its joint enterprise as understood and continually renegotiated by its members.

The Plone community is engaged in producing a web based Content Management System, building on existing frameworks such as Zope and the CMF. The activities for the developers include creating solutions for clients based on the system. It also includes creating documentation and facilitating the use of the system. This is a continually ongoing process between the participants. Sometimes general goals can be stated, but the everyday practice is closely connected with sustaining a common view on what it is about. Different views and claims are negotiated on the mailing-list and at the IRC channel at all times, and the information on the **plone.org** website represents an official or general view.

A typical result of a failure of a common view, could be the occurrence of forks in the project. When someone disagrees in the direction a project takes, either because of technological

issues or because of the function and requirements of the program, a new community can start to grow and gradually change, or fork, the code-base so it ends up being two projects instead of one. This can in some cases be positive, i.e. as an ecological selection, but it is not wanted by the leaders of the project. Forking is often a result of to much disagreement among the developers. In Plone this have not happened yet. One reason may be that there still exists agreement among the developers on what Plone is, and the direction it is taking.

How it functions - mutual engagement that bind members together into a social entity.

Several of the interview subjects in Aspeli's (2005) research expressed that "the community is Plone". Although the technology used in the project might change, the participants will still be engaged in this social entity. The social character of FLOSS, despite the geographic distribution of the developers, is important. The long-term relationships demands a mutual personal engagement in the common tasks. The technology could be changed radically but it is the community of Plone developers, consultants, leaders and contributors that truly defines the project. The mutual engagement in the project makes people dependent on each other. Aspeli (2005) addresses this himself in a panel debate (ECM Panel) when says that "feeling wanted" is important, (mentioned by Paul Everitt when he says that Plone's greatest asset is its democracy). Alan Runyan, one of initiators of the Plone project, says that:

> *..." the community is what made me stick with plone. I have personal relationships in the community and even when i feel unhappy with the project -- the community members are there to reinforce my commitment" (Aspeli, 2005:45)*

This statement confirms that the mutual engagement, and the attributes as a social unit with personal relations as the key attribute, works as the glue that keeps the project together. The most important components of this 'social' glue is the collaboration and friendship that IRC, sprints and conferences facilitates. The developers are in different settings outside of Plone, and have different reasons for being involved. It is the the livelihood of some, and volunteer, unpaid work for some, and a mix of these two for many. One of the initiators of the Plone project, Alexander Limi, says that the work in Plone is split between commercial interests and volunteer work. He sees this as the reason why Plone is succeeding. :

> *It's pretty evenly split. There are strong commercial interests in Plone, but there is an equally strong force that is volunteer work. As I earlier mentioned I believe this combination is what makes up the Plone community, and is the reason why Plone is succeeding where other projects have failed. We never go too far in one or the other direction, we are specialized, but in a million directions at the same time. We have people*

> *that have an incredible amount of knowledge in a particular specialized area, and we have people that are generalists and architects. It's a very healthy composition, and we have very constructive dialogue in the community.*
>
> *(Aspeli:2005:44)*

This ability, that the participants have different motives and interests of joining, and that the community is able to satisfy and include different needs is an important characteristic of Plone.

What capability it has produced – the shared repertoire of communal resources (routines, sensibilities, artifacts, vocabulary, styles, etc.) that members have developed over time.

The main capability produced by the Plone community is the software itself; Plone CMS, with all its add-on products, tools and help-applications, as well as the code repository which contains the code and the application programming interface (API) which contains several hundreds megabytes of documentation. The home-page, **plone.org**, is the home of the project and is self-sustained in that it uses its own software to manage projects, files and documentation. The Plone community has produced 14 mailing-lists with 117 296 postings[1], which is searchable, and provides a history of the discourse of the project, from the day the list was created up-till today. It has also produced a large amount of online documents like how-tos, faqs and tutorials. In addition it has produced a certain style or flavor, sometimes referred to as 'the Plone Way™' or as 'plonish' by the developers. In the visual design of things this is easy to recognize, but it also has to do with the style of programming. Technical issues and style is closely connected, and it is this style a Plone developer, or plonista, identifies with as a plone developer.

An example of that Plone has produced a thing that can be used other places, is the use of elements of the Plone GUI in the Mediawiki[2] program. Mediawiki powers Wikipedia, an online encyclopedia and one of the top-20 most visited sites on the Internet. Mediawiki´s layout is based partially on the Plone style sheets (CSS). Plone has also produced a certain language, also closely connected with the technology. (e.g. the "*bad old days of bobo*", an unfortunate earlier name for Zope (Aspeli, 2005)). An attempt to collect important words

1 The lists were counted with Gmane 24.06.2006, doing searches with an empty search string, which returned all postings in the list. The name and numbers of the lists can be found in Appendix 4.

2 MediaWiki is a wiki software package licensed under the GNU General Public License. It can be found at http://www.mediawiki.org/.

from the language that might be useful for a newcomer, is done in a list of glossary entries in thea “glossary entries” in the documentation area of **plone.org**, where some 30 words are explained.[1] A language is of course more complex than its words, and does not only have to do with what people say, but also how they say it. The way of talking has to be learned through participation, but an example of more formal instructions is the “how to ask for help”[2] section at **plone.org.** An example of the increasing understanding of “how, when, and about what old-timers collaborate, collude, and collide, and what they enjoy, dislike, respect, and admire” (Lave and Wenger, 1991:95) is when a participant called Querk sent me a list of must-have application to use at my computer. The list was initially from a participant called zorb, and forwarded to me. The list gave me an overview of what types of programs they preferred and liked. The list was sent via email, when we sat face to face at the Europython 2005 conference. Plone conferences and sprints are important meeting places in the Plone community. There are annually international conferences and several regional conferences. Small and big sprints are held several times a years with between 5 - 40 developers. The Plone community has also produced a formal way of handling improvement proposals. Plone Improvement Proposal[3] (PLIP) are documents written by the Plone team to structure and organize proposals for the improvement of Plone. It has also produced an organization, The Plone Foundation, for providing support for the development and marketing of Plone.

5.1.1.The rise and fall of Eventregistration

By doing fieldwork and taking active part in the practice by putting myself in a situation where I had to learn and participate, I created a setting where I could reflect on my own learning. I will now describe the activities, relationships, tasks, tools and resources I used, and took part in as an participant. The practical work I have done with Plone is the basis for my learning curriculum and it can be summarized in four categories;

Layout and skinning: Plone comes with a default layout built with Cascading Style Sheets (CSS), graphical elements such as .png and .gif files and Dynamic Templating Markup

---

1 Another famous such collection of words is the “Jargon FIle”, a glossary of hacker slang from technical cultures including the MIT AI Lab, the Stanford AI Lab (SAIL), and others of the old ARPANET AI/LISP/PDP-10 communities. It was started by Raphael Finkel at Stanford in 1975. (http://en.wikipedia.org/wiki/Jargon_File, 2006)

2 “How to ask for help” is a description of how to best ask for help in the irc channel and at the mailinglist and how to write a semi-formal error report. It is written as an alternative to Eric Raymond's “How to Ask Questions the Smart Way”. It can be found at http://plone.org/documentation/how-to/ask-for-help

3 A list of Plone PLIPS can be found at http://plone.org/products/plone/roadmap/psc_improvements_listing

Language (DTML). All the content is separated from the visual presentation, and is controlled by several CSS files and DTML codes. To find out what piece of code controls what element a detailed knowledge of the structure of the visual styling and knowledge of CSS is acquired. Plone emphasize a well-formed graphical user interface (GUI) and visual design, and some of the work is to not mess this up, and to work within the Plone style.

Installing and upgrading Plone, Zope and Products: Plone comes with a default set of products that provides basic functionality for handling documents, news, events, files and links. To add extended functionality it is possible to install Products, which are packages written in Python and the Zope template language. These products must be downloaded and then uploaded to your server. Then the server must be restarted and the product must be installed. Some packages depend on the service provided by other packages or libraries on the server, and these also has to be installed in order to solve the dependencies. Some of the packages also need heavy configuration and customizing depending on the planned use and the state of the product.

Creating and modifying products: The ready-made products can be modified and provided with more or less functionality. If there is a general interest of the added functionality, and the responsible/maintainer of the product finds it appropriate, the new code can be included in the base code, and become a part of the product. If a large body of new functionality is needed, a new product with the wanted features can be created. In the Eventregistration product I added the functionality of sending mail to all the registered participants of an event, and the code I wrote is now part of the Eventregistration product. On a higher level Plone can be packaged with a set of products and a default configuration to serve certain needs. Examples of this is EduPlone[1] and Plone4Artists[2].

Adding and managing content and users: When the installation is complete and the products are set up and working, Plone is ready to fill its purpose; serve as a Content Management System. The next part of the work is to create the information structure of the site and add the actual content. Users are supposed to add and manage content themselves based on roles and

1 Eduplone is a version of Plone that handles Learning-Content Management and Web-supported Learning-Communities http://eduplone.net/

2 Plone4Artists is a project to develop a common global infrastructure for artist community websites that is freely available under an Open Source licence. http://plone4artists.org/

workflow. These users needs to learn how to use the system and I provided training and support for a group of users.

A reallife use-case - www.kampkunster.no

To obtain an authentic use-case to "force" me to gain knowledge and to put me in the situation as a learner, I needed a real life use-case scenario forcing me do real work with Plone. I aimed to get as close to the terms other participants in the community worked under. After searching for projects for a while I was lucky to get the opportunity to do several different sites in Plone. I will use examples from the work one of them here - Kampkunst Portalen.

In the spring of 2005 several cooperating martial arts schools wanted a common web portal/ solution for their schools. All the schools had existing sites on different servers, using different technology. The requirements and goals the schools had to the new site was:

- Share content like events and news without having to add it several places.
- A selected group of people should be able to add and manage content themselves and function as editors for content added by others.
- Members of the school should be able to register at the site and add content.
- Add and manage photo and video albums.
- Registration of the different courses given by a club.
- Possibility for the members to register at the class they want to attend.
- Different visual styling for the sites of the different schools.
- A discussion forum for the members.
- Reduce cost and administrative work by using one hosting service provider for all the schools.

As a board-member and web-master of one of the schools, I was chosen to sit in the project group for the portal called kampkunster.no. I was responsible for the technical solution, and this chance handed me a perfect way to work with Plone. Plone had support for all of the features asked for, except the handling of registration for courses. We bought space on a Linux server running Zope and operated it remotely using Secure Shell (ssh). I got a Plone instance up and running, did the necessary configurations, and installed the wanted products.

Scratching a small itch

As instructors in one of the schools I had to write down the name and email address of all of the attendants of a class at the start of each semester using pen and paper, and then write them down in a spreadsheet and send it to our treasurer so he could check who had paid the fee. This was time consuming and we really wanted to make the attendants register themselves in

a web-based schema, and that our treasurer could check the list on the web. The communication between the club and members was mainly mediated by email and we used a list of email addresses for each of the courses that the instructors handled. This was also an awkward solution, resulting in poor communication. What we wanted was for the instructor of a class to just visit the course' site and then email all the attendees who was registered for that course through the web. This work was clearly something a computer and the internet could do for us. One could say that my engagement in Eventregistration came from "scratching the developer´s personal itch" (Raymond, 2001) which is typical for many initial FLOSS projects. In opposition to conventional software development, the developers in FLOSS are often their own clients and creates their own requirement specifications.

After searching on Google and asking around in the different communication channels I found out that a tool with possibilities for registrable events was missing in Plone. I posted a request at the **plone.user** mailing list asking for a tool like this, with a registration and email feature, and at the same time I asked how to start writing such a product, if it didn't exist. After 6 days I got an answer from Jason McVetta, who were almost finished writing a product with parts of the wanted functionality. The product was called Eventregistration, and a 0.2 version was registered at **plone.org** 19 of April 2005[1] under the GPL license.

At **plone.org** a new product is handled by a software-project registration tool with information about motivation, improvement proposals, a roadmap, releases, contact address, link to svn and to the mailing list for support. The FLOSS product called Trac is used to manage the source code files of the projects which are stored in a svn repository. Eventregistration was written as an Archetype-based drop-in replacement for the default Plone Event-type. I downloaded the product and tried to install it. To install it I had to update and migrate other parts of my plone site to handle dependencies, which I had to use a install.txt, how-to.txt and readme.txt[2] file to do.

In my initial request at the mailing list for a product with the wanted functionality I mentioned the possibility to email all the registrant in an easy way. This functionality was not implemented in Eventregistraion 0.2, but McVetta thought it was a good idea, so it was listed

1 The homesite of the Product can be found here: http://plone.org/products/eventregistration/

2 Install.txt and Readme.txt are text files which are included in the Product package and are used as basic instruction on how to install and use the program. These files are usually always distributed with the source-code.

as improvement proposal #5, targeted at release 0.4 with me as the proposer on the product site at **plone.org**. The proposals are either in the status of being-discussed, in-progress, rejected, ready-for-merge or completed. In a mail McVetta asked me to post problems and bugs at the project site:

> *..And please, leave comments there; the more public the development process, the easier for more people to get involved. :)... (personal mail with McVetta, 2005)*

When I first started to communicate with McVetta I presented myself as a newbie and gave some information about my background. He encouraged me and invited me to write the feature I wanted, and this was an opportunity for participation :

> *.. Would you be willing/able to code this feature? ..(personal mail with McVetta, 2005)*

As I started asking questions, he gave me useful answers and helped me by sending me links to resources and hints on where to start. It seemed to me that he was clearly aware of that treating the users as co-developers is considered the least-hassle route to rapid code improvement and effective debugging (Raymond, 2001). More users find more bugs because more users adds different ways of stressing the program and also potentially brings up the need for other interesting features to be added to the application. This small scale loosely-coupled collaboration was hopefully going to bring a useful product to Plone. I will now describe three incidents that happened during my participation in Eventregistration. For the reader to be able to better understand the practice, I will be quite detailed and technical.

The feature was simple enough; I needed a form for the user to enter the message into, a script to get all the email addresses of the registered participants and a script to send the email. Because I didn't know DTML, Python or Zope I had to gain quite a lot of knowledge to be able to do a simple task. Searching for what solution to use was the most time consuming part of the work, but I learned a lot while reading about different solutions and I got an overview over the possibilities for sending mail on the Internet. After searching I came up with 3 different solutions on how to send the mail. I tested all of them but only one was working adequately. So I threw away the other alternatives, and concentrated on the ‘dtml-sendmail’ / ‘MailHost’ solution. I found examples written DTML about how to send out email from the site with MailHost, and modified them slightly. Rewriting and reusing code, and ideas, is the core benefit of FLOSS development, and I immediately understood why. For the actual form, I used the code from the "Send feedback" template in Plone. For the button "Email all

registrants" and its associated action, I copied pieces from McVetta's code that were used in the other functions, changed the name and added the appropriate action to it. We had a small discussion on how to do parts of the graphical user interface like where to put a button, on email, and McVetta suggested:

> *"..A tab (next to "export list of registrants") would be more Plonish..", (personal mail with McVetta, 2005)*

I agreed and learned something about Plone GUI. When I had a working "mail all registrants" function on my local server, and had tested it in different ways, I sent the patch on mail to McVetta. He read it and added it to the rest of the code. McVetta soon discovered a bug, and sent me a mail about it. I fixed the bug and mailed him the solution as a small patch. The bug was a 'shallow bug'; the mail server was set to 'localhost', but it should be set to the Plone site's MailHost setting, so it had to be changed to 'Mailhost'. Although McVetta is a more experienced Plone developer then me, he did not know exactly how to solve it:

> *...That's all it took? Excellent!... (Personal mail with McVetta, 2005)*

Because of my recently gained domain knowledge on sending mail from Plone, I was able to solve this quickly. Raymond (2001) suggests that using many testers is a good thing; it increases the probability that someone's toolkit will be matched to the problem in such a way that the bug is shallow to that person. He also suggests that the bugs are often discovered by someone, and fixed by someone else.

The 12. of April I was asked if I wanted a login at McVetta's svn server. Getting access to it can be seen as a act of confidence and as an invitation. The email with the offer ended with the words:

> *"Thanks for participating!", (Personal mail with McVetta, 2005)*

I realized I had made a small contribution to the Eventregistration project. This was followed by McVetta adding me to the credits.txt file which is distributed with the Eventregistration source-code. Together with the email with the question about what username I wanted on svn, was a link to 'Version Control with Subversion', which is the main online resource for svn.

The collaboration I had with McVetta is an good example that learning seen as LPP is not necessarily master-apprentice learning, but rather peer-to-peer learning. This way of learning is described by the concept of ZPD. His general knowledge was higher than mine and by imitating his code, asking questions and reading each others code we learned from each other in a relationship that can be explained by ZPD.

On the 3rd of august 2005 McVetta created a Google Groups mailing-list for Eventregistration [1]to make communication between the developers easier:

> *..This is better than the mess of Ccs[2] that we're working with now... (McVetta, Google Eventregistration Group, 2005)*

From that point all of the communication went through this list. The archive of the list is publicly available on the net, and googling "Eventregistration" will give results from the list It is now a searchable resource for others who might be interested in using the product. Before the list was created I wrote 9 messages to McVetta, including the initial mail to the **plone.users** mailing list and McVetta wrote 20 to me. The google list had 25 members (03.12.05), but not all of them were active. From the 8th of August till the 19th of October 150 messages was sent to the list in 20 different topics or threads. The biggest topic was called "Trouble uninstalling" and had 54 messages written by 4 different authors. I wrote 14 messages to the list. On the 5th of August 2005 a person on the list wanted to know how the product handles the email sending bit. Because I had written the feature I gave an answer with excerpts from the source code on how we used it. MailHost is a Zope object that uses the localhost for smtp by default, but it can be configured to use an arbitrary smtp server. He then asked for a specification and I gave an answer and pasted in the link to the documentation for the solution we used. Later that same day McVetta gave another specification of the answer I gave, with some code examples on how to use another host than the default localhost for sending mail. As other participants entered the list and asked question I found that the use-case scenario I initially posted at **plone.user** was something several others had in common with me. One might say that sharing use-cases also means sharing a learning curriculum and sharing the same need for tools. The need for certain tools and features to do a task, or solve a problem, is what brings programmers together in the first place. This common problem is initially the seed that the practice of the community grows from.

---

1 The group can be found at http://groups.google.no/group/Plone-EventRegistration

2 CC is short for 'Carboncopy' which is a list of recipient who gets a copy of the email, when sending it to someone.

From 19.03.06 to 18.08.06 there were five releases of Eventregistration. The 0.2 release fixed a problem with the packaging of version 0.1. The 0.3 was a clean-up release, and no new functionality was added. The 0.4 release added the "Mail all registrants" feature I had written, and also improved date display and flexibility. It also added an option to send confirmation and registry emails when a person registers. The 0.5 was a code cleanup, but also added a "how did you hear about this event?" field and a "Would you like to receive a newsletter?" checkbox field. It also gave a slightly nicer visual formatting. The upcoming release 0.6 is supposed to add Plone 2.1 compatibility and internationalization for different languages. The features was not only added by McVetta, but by several participants.

The FLOSS rule of "release early, release often" was followed in the Eventregistration project. Some Plone project release new versions everyday, sometimes referred to as “nightly builds”. By giving early releases, bugs are discovered in an early stage of the development and more users and developers will find more bugs. The possibility for proposing and adding functionality is also higher when the application is released in an immature state. In many projects there are usually two alternatives; the latest stable release or to checkout the latest code from the svn. Because of some start up problem with the svn, it was not easy to know what the latest branch in the svn was, and therefore it was difficult to know what branch to submit to.

> *..The actively developed branch has been jmcvetta-atct_and_location. That name is less than obvious, so I am renaming it to "0.6-devel"... (McVetta, Google Eventregistration Group, 2005)*

Naming conventions is part of the language, and often the use of file -, variable and function names, and here also the branch name in svn, has to be interpreted in the right way. By renaming it to 0.6-devel there is no doubt where to submit code that you want to go in to the 0.6 version.

Maximum participants

One of the features of Eventregistration was that it should be possible to set the maximum number of participants that could register for an event. This feature was proposed by me, and McVetta asked if I would write it into Eventregistration. The python script that checks maximum number of participants allowed, with the number of registered participants was simple, but because of Python/Zope’s heavy object orientation and the special acquisition

function I was not sure how to do this. Especially the naming of one variable was interesting. By reading code written by other developers I saw that some places ‘here’ was used and other places ‘context’ was used in the same way. Through discussing this in IRC I got the impression that ‘here’ was on the way out, and ‘context’ was the new standard. By asking if I should use ‘here’, I initiated a 50 line discussion between 4 other participants, and I learned a bit about ‘here’ and ‘context’ and which to use:

plmxn If iam inside a folder and use the tal:content="python:here/id.max()" it will return the result of the method max, right ? or do I have to use the absolute id ?

peer here/id.max() can't work

jokotan in a python: it'll divide...

plmxn aha

plmxn what about here.id.max()

peer plmxn: better :)

joix plmxn: absolute id ?

finro plmxn: why do you want to call max() on string ?

peer plmxn: however, here.id (or better: context.id) is a string in most cases

plmxn ok. Well I want the object .. ?

peer plmxn: I guess you just want to call context.max(), or even tal:content="context/max"

plmxn joix:absolute URl to the id of the object ?

peer that would be context/absolute_url

plmxn so the context is the object Iam inside ?

peer context ist the object youre template is called upon

peer s/youre/your

plmxn ok. I want to count objects in a folder and check that its not full, so I added a max() method to the folder and want to call it to check if its full... ?

finro plmxn: then use tal:content="here/max"

peer .oO(I thought "here" is evil and should be replaced with "context")

joix here == context

peer joix: really? :)

joix peer: yes

joix peer: context was invented for pythonscripts, here was invented for TALES

sonsw in fact, "here is context"

sonsw yeah, because John was an eedjut ;-)

joix peer: later someone realised this might cause confusion...

joix so standardisation is heading towards 'context' in TALES as well

peer joix: no, really? :) I know, that's why I said I thought it's evil

** finro does not find 'here' particularly evil*

** peer suggests to replace 'container' with 'there'*

kultut portal => 'somewhere_else' ?

peer kultut: yonder

kultut better

kultut if here == there: print yonder.location()

peer fix: if I wouldn't know better, I'd think 'here' is the script itself, (same as 'template' actually)

fix peer: yeah i agree with you... if template and container are used why not here is 'translated' to context?

peer kultut: :))

kultut Is there a minimum version of tales/zope/whatever for 'context' to work?

peer fix: it already is translated

fix peer: oh... since which version?

joix dunno, ask kultut :)

kultut Googling... found a message saying that zope3 'll use context in the page templates

pollokan Zope 2.7

pollokan and maybe late versions of Zope 2.6

fix pollokan: ok i haven't noted that... still using here :)

kultut Found something:

http://mail.zope.org/pipermail/zope-checkins/2003-October/024407.html

kultut Late 2003, added 'context' as an alias to 'here'.

kultut So should work, both of them.

kultut I like 'context' more so I'll be switching!

pollokan me, too

An interesting thing about this conversation was that it was me as a newcomer who initiated it with a simple question, but the conversation was carried on by more experienced participants and some of them actually conclude they are going to change their use of here/context at the end of the discussion. This example shows that newcomers can influence and contribute, by asking questions that initiates reflection on the established ways to do things.

My initial plan was to use python in-line script in the Event_view template to get the number of the participants. McVetta reminded me of the basic rule of web-programming; separating logic from presentation, and he recommended me to add a function to the Event itself, instead of in the template for Event. I agreed with him, and implemented it the way he advised me to. I had a working solution for the ER 0.3 release, but as the rest of the code was developing it

was difficult to know how it changed and how to implement it if the rest of the code was changed. I had to ask at the mailing-list about what code to base the function on.

In november 2005 the last commit to the svn was made by McVetta. On the 11th of Oct 2005 he wrote this to the Eventregistration mailinglist:

> *No firm release date for 0.6. If you know someone who wants to pay for its development, it would definitely move faster. As it is now, it's comming along on a "when I have time" basis.. which is not too fast right now, since all my time is being consumed with visits to family & old friends. It'll probably move faster once I'm back in California, with better net access and more free time. (McVetta, Google Eventregistration Group, 2005)*

After this only a few post was made to the mailing-list, Eventregistration 0.6 was not released, and the project is halted until someone eventually blows life in it again.

A visual challenge: skin per folder

One of the requirements for the kampkunster.no portal was that a the portal should have one common front-page, but the visual layout should change when a visitor entered the school´s site. To do this automatically, I had to learn how to set a different skin (visual theme) based on the URL of the site that was visited. I googled and read earlier postings at the mailing-list, and ended up spending the day learning how to create and use different skins in Plone. I installed two different skinning products, Moos´ Ruby Red and DIY Plone Style, and read installation instructions and a how-to. I managed to create a modification of Moos Ruby Red skin and also to change the skin based on URL, but I was not able to change it to Moos' Ruby Red. This solution used an external python method and an access rule. The solution was based on the how-to called "Selecting a skin based on URL"[1] and "Set skin for folder"[2]. This solution was complicated and would need a new skin for each school-site. When I read the mailing-list I also found another solution which supported making changes in the CSS on a per-folder basis. The advantage of that approach was that all the different styles were defined in one style-sheet, and no conditions in the registry was needed. I tried out this solution, but couldn't make it work. So I decided to write a question on the mailing-list about what the best solution for me might be with the given use case, and some specific questions on the selecting skin per-folder problem. My posting initiated a 9 post thread with 3 persons involved[3]. The person

1 http://plone.org/documentation/how-to/select-skin-by-url

2 http://plone.org/documentation/how-to/set-skin-for-folder/

3 http://article.gmane.org/gmane.comp.web.zope.plone.user/42512/

who originally came up with the change the "CSS per folder" solution answered me and gave me hints on how to make it work, and this time I was able to do it. This seemed the by far best solution for my case and I decided to use this solution. After reading more about this solution, I found the discussion where the hack[1] was created. It was a 8 posting thread with 3 persons involved and amongst them two of the core developers was involved. The work and communication was done in an interaction between members, and automatically stored at the mailing-list, so that it was accessible for me later.

The engagement in Eventregistration and Kampkunster Portalen, and other sites, involved learning a set of new technologies and programing languages. I was not familiar with any of these and had to search for knowledge in many different ways: tutorials, install.txt, readme.txt, the code itself, IRC, **plone.user** and the Plone Eventregistration mailing lists, websites with comments and more. I had to act in the form of writing source code, install products, configure and create scripts, discuss solutions, ask question, explain my problems to others as well as explain issues to others. By interacting with people relationships evolved. Finally the code I had written was merged with the other code and the proposal reached the completed status. Although my contribution was small, I had collected interesting data, and the goal of my fieldwork was achieved. Experiences from the fieldwork was written down in a weblog to be used in the analysis. This detailed description might seem trivial and unnecessary to include, but an ethnographic analysis of the practice of the community has to deal with the ongoing activities and what actually happens on a low level. I do not put an "equals to" mark between my learning and others learning, but according to LPP participants are peripherally located at different places in the social world, and it is individual what needs to be learned. Everybody's learning-trajectory are different, and this fieldwork was about my initial contribution and first relationships.

The work with the portal also included giving training and support for the users and editors of the kampkunster.no site. On the training sessions I had to explain concepts of Plone to the endusers. In this way I had to externalize the knowledge I had achieved in working with Plone. This is included in Nonaka's "externalization" as an example of eliciting and translating the tacit knowledge of others, customer, experts for example – into a understandable form. Dialogue is an important means; during direct communication people share beliefs and learn how to better articulate their thinking.

---

1 http://thread.gmane.org/gmane.comp.web.zope.plone.user/38764

5.1.2.The learning curriculum

According to Lave and Wenger (1991) a learning curriculum is a field of learning resources in everyday practice, viewed from the perspective of learners. Because the learning curriculum is essential situated, my learning curriculum is closely connected to the tasks I have to solve in the work with 'kampkunster.no'. This includes learning to install and upgrade Plone, Zope and additional products, skinning and visual configuration of the site and adding functionality to the Eventregistration product. Finding, sorting and using the actual learning resources is done through **plone.org** and **google.com** and through links and documents given from other participants. The activity of searching, reading and modifying the search keywords to get a a more precise result of the externalized resources, and formulating and asking questions on **#plone** and the mailing-list, was a major activity in my fieldwork. The formulation of google or Plone search-words as opposed to formulating "social" requests at **#plone** and the mailing-list, represents requests for two different types of knowledge. The first is a search for formal, externalized knowledge contained in documents, the latter is a request for active, soft and "living" knowledge held in peoples head´s. The learning curriculum is shaped by the need for knowledge to solve my problems, not by an instructors external view of what knowing is about. The learning curriculum has to be considered as a part of the social relations that shape LPP. Through my fieldwork I developed a view of what the whole project is about, and what there is to be learned. A learning curriculum is a characteristic of the community. Denying access and limiting the centripetal movement of newcomers and other practitioners changes the learning curriculum. In Plone, access is easily gained as learning resources and people with knowledge are accessible and available. The centripetal movement towards full participation is up to each participants to accomplish. In fact, both legitimacy, peripherality and participation is up to oneself, and the responsibility for the learning-process lays more on the individual participant than what might be the case in more formal communities of practice.

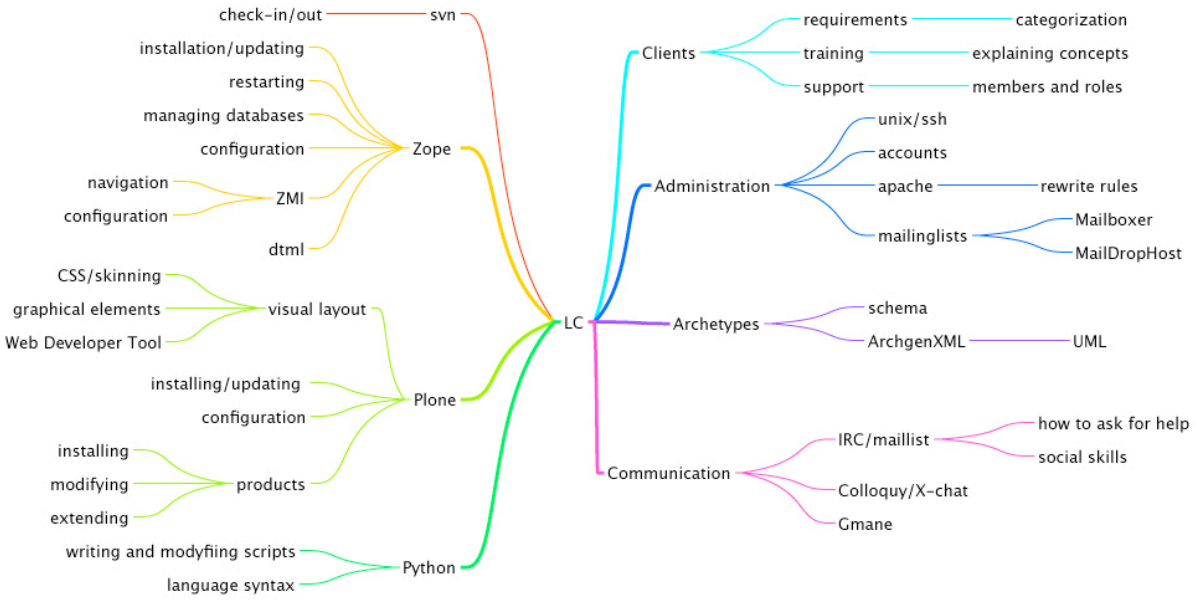

Illustration 5: Elements from my learning curriculum

The learning curriculum evolves and takes it form when the learners face problems that requires knowledge in certain domains. The practice of the community creates the curriculum.

5.1.3.Legitimacy

The form that the legitimacy of participation takes, is defining characteristics of ways of belonging, and is therefore not only a crucial condition for learning, but a constitutive element of its contents (Lave and Wenger, 1991). At the mailing-list and at IRC, there is a high degree of exchange of information between totally newcomers and old-timers and everyone in between. Everybody engages in problem-solving; either the old-timers helps the newcomers or there is peer activity, and flow of information is always in many directions. The contributions from the newcomers are often wishes, bug-reports, or they need to do something which they have little, or no, idea on how to do. Certain requirements are demanded before one asks questions, and some rules are meant to make the communication easier. They are described on the web-page "Asking for Help with Errors & Problems". If these rules are not followed, i.e if someone does not search the mailing-list or google first, or do not write english or describe the problem and the context, or include error messages etceteras, they might be left without an answer on **#plone** or **plone.users**. Most of the time guidance and corrections are given. Legitimacy can also be seen in conjunction with meritocracy; if you do not contribute back in any way you are not so much wanted. Core developer Matt Lee puts it like this:

> *"I feel the Plone community welcomes everyone who can do anything to contribute back". (Aspeli, 2005).*

The risk for "free-riders"; people who use the software without contributing back, is high in FLOSS projects. The more a participant contribute to the project, the more legitimacy he gains. Contributing back also means to help others with their problems. The connection the newcomer has with the existing community is first very informal, and puts little or no demands on the newcomer, except from the just mentioned 'asking questions the smart way'. Later this connection is defined by personal relationships, skills and knowledge connected to various parts of the software. To get closer to the core of the community means taking on more responsibility and getting more knowledgeable. The legitimacy is thus not decided by formal contracts, but gained with skill, participation and involvement. The question of legitimacy will be discussed more later in connection with the term "lurking".

5.1.4.Peripherality

According to Lave and Wenger (1991) there may well be no such simple thing as 'central participation' in a community of practice. Although the Plone community has central actors, like core developers, peripherality suggests that there are multiple, varied, more or less engaged and inclusive ways of being located in the fields of participation. Peripherality is about being located in the social world. When peripherality is enabled, it suggests an opening, a way of gaining access to sources for understanding through growing involvement. The principle of openness, access and transparency in Plone indicates that there is a wide opening to gain legitimacy and change location in the peripherality. Furthermore, legitimate peripherality is a complex notion, implicated in social structures involving relations of power (ibid). As a place in which one moves towards more intensive participation, peripherality is an empowering position. To welcome people who wants to get involved, is in the nature of FLOSS, and is part of what defines a project as "free/libre" or "open". It is one of the pillars of the development model; keeping people out of the project is considered one of the quickest ways to kill a FLOSS project. As a place where one is kept from participating more fully, peripherality is a dis-empowering position. In Plone's meritocracy, where the legitimate peripherally is graded after contributions, engagement, involvement and degree of knowledge, knowledge enables these things in a circle. With poor knowledge, contribution and building relationships based on collaboration is difficult. To engage in collaboration initiates learning, which in the next round makes contributing possible. There are no built-in dis-empowering factors, like we saw in the meatcutters example or other hierarchical constructed obstructions.

The main challenge is the learners technical un-ableness of getting involved, which again, is closely connected to their knowledge. There is no structure to organize the increasing participation, and thus changing location in the peripherality.

Many FLOSS projects has a “get involved” section on their web-site, with links to the code repository, email lists or IRC chat rooms. Hegemony over resources for learning and alienation from full participation would mean the end for the Plone project. The old-timers and newcomers are both stake-holders in the project. The more participants Plone community are able to make into full participants, the healthier the project is. Lave and Wenger points out that the concept of “*full participation*” is intended to do justice to the diversity of relations involved in varying forms of community membership. Full participation is what partial participation is not, yet. The general transparency of the technology in that all the tools needed for joining is free, and the source-code of everything from the operating system to the graphical editor can be studied and modified, makes the tools of the practice open and accessible, and the knowledge needed to use them is the only obstacle. There seems to be a general attitude that attracting new members is seen as positive in the Plone project, there is a welcoming tone among the developers. They are helpful in providing tips, they act as hosts and willingly takes on the role as the experienced part of a ZPD relationship. Accessibility is therefore easily gained.

People move around in the peripherality of the practice depending on the knowledge domain they have and their use-cases. The different domains and sub-communities has their own practice and people. To ensure that the code that goes in to the core Plone source-code is high quality, not everyone can edit this code, some control mechanisms are required. This is not about restricting access to the practice, but to ensure high quality of the code, and involves power mechanism based on meritocracy, reputation, contribution and more formal status like the ‘release manager’ role. Access to the central svn repository is technically access to the central organizing artifact of the community. On the basis of what I learned through my legitimate peripheral participation I would say that what I have to do to learn more would be to take on more responsibility, this could i.e. be to take over the Eventregistration project when McVetta stopped being active. I could also get involved in other projects, but the main activity should be to participate more and to undertake more difficult tasks. This would also change my peripherality and my legitimacy, as I would become a knowledgeable co-participant with more personal relationships, and with whom the other participants could discuss ideas, undertake commitments and collaborate with.

5.1.5.Participation

By “watching” McVetta, reading his code, and acting together with him by discussing, writing and modifying code, I copied and imitated him. I got comments on my actions, discussed them with him, and changed them accordingly. Through collaboration we created and maintained a common view and idea of the desired result. The dense interaction, negotiation and socialization made differences in meanings visible, and made it possible to repair them. Participation, and reflection on action, is a key for being able to express oneself in the community through interpreting, communicating and learning. To read, produce and use text is a socio-cultural activity that involves practical and intellectual techniques.

The fact that the old-timers, and specially project leaders, acknowledge the importance of taking care of newcomers makes legitimacy and transparency high. In the butcher study (Lave and Wenger, 1991) the learners was seen as cheap labour, and the thing they could contribute with was doing the boring repetitive tasks. In FLOSS, taking care of the learners are seen as very important for the continuation of the software (Raymond, 2001). The newcomers contribute with error messages, reporting bugs, asking for features, provide use-cases and code, and by using the program they help spreading the word. The naive, unexperienced newcomer can help add a perspective to the community and might lead to contributions and changes. The old-timers reflects on their practice and activities by explaining how things are done around here, and thus define themselves in the meeting with newcomers. Raymond (2000) remarks that the last thing one should do as a initiator or leader, before leaving a community, is to find a competent successor. This successor should not only be technically competent, but should also have gone through a socialization process to become a legitimate near-the-core developer, with an understanding of how the practice of the community works and well developed relationships with other participants. This generative learning process is what one takes part in through participation, and is what the concepts of LPP tries to describe.

Joel Burton, chair of board in the Plone Foundation, says in the interview with Aspeli that the Plone community is a few things:

> *- a core audience of about 50 people, who are the "regulars" in IRC, conferences, etc.*
>
> *- a broader audience of a few hundred who have committed in any way to the project*

- a large audience of thousands who silently (and not so silently) cheer from the sidelines, advocate for us, implement us locally, etc.

Joel Burton (Aspeli, 2005:52)

After doing my fieldwork I place myself in the second and last group of participants. I have made a small contribution to a add-on product, and because of me, 8-10 persons are now using Plone as their web publishing solution reaching an audience of a couple of hundred regular members or so. To take a liking for what the practice evolves around, and to care about its artifacts, people and values is important for the participation. In my fieldwork I have entered the community, and in a small grade changed peripherality through getting involved in the Eventregistration product. If I where to learn more, a considerable amount of time would have to be invested in participation, and this would lead to a change in my "location" in the community. Let us move on to look at what some participants who have made such changes in peripherality, has to say on the subject.

5.1.6.Full participation in the practice of the community

Let us look at what some core developers who actually have reached "full participation", say about changing place and status in the community, when answering Aspeli`s question:

Do you feel that your status in the community has changed, or is changing, as a result of your contributions? If so, in what sense?

My status has changed quite a bit. Before I started contributing directly, I would write an occasional, hopefully helpful, post on the mailing list and submit a patch/bug report here and there; nobody within the community really knew or cared who I was. Now, apparently, I'm a Rock Star. ;) Such recognition isn't terribly important to me, but it's not at all a bad thing, and potentially a very good thing (as noted above).

(Alec Mitchell, Independent IT consultant and core Plone contributor)

Oh, yes! People are honoring my contributions and listen to my suggestions.

(Florian Schulze Student, independent IT consultant and core Plone contributor)

Definitly. First of all there are some people knowing my name now. As I am so far only communicating virtually with everybody my role is determined through my actions. As I havn'tdone much in the past, how should anybody had known me? Now I am working on the same project with everybody which includes communication, which is combined with ones actions the base of one's role in a group of people. So quite logically my status has changed.

(Hanno C. Schlichting, IT consultant and core Plone contributor)

(Aspeli, 2005:63,65,67)

These are examples of participants who have moved closer to 'full participation' in the community of practice. The citations show that their role, and maybe identity and location, in the community has changed toward the center. Schlichting points out that because he is only communicating virtually his role is determined through action. Schulze mentions that people are listening to his suggestion, and Mitchell mentions recognition as potentially good thing. To go further, it is interesting to look at how people who have reached 'full participation' define themselves as part of the community. Let us see what some central participants answered to Aspeli's question on how they relate to the community, and how the community influences their dealings with Plone:

Do you feel part of this community? To what extent does the community influence your own dealings with Plone?

*absolutely. the community is what made me stick with plone. I have personal relationships in the community and even when i feel unhappy with the project -- the community members are there to reinforce my commitment. I believe this is very important attribute behind cults. And is something we try to exploit in the dealing with people in the Plone project. *wink**

(Alan Runyan, Plone co-founder and architectural leader, owner of Enfold Systems LLC)

I do feel like part of the community. My role is a bit fuzzy--although I'm a programmer by trade, my contributions to Plone have tended to be less technical, as I was getting plenty of on-keyboard action in my consulting practice. As I've recently moved to a full-time position that's more managerial/sales than development, this may change, and I may start doing more technical stuff with Plone.

(Joel Burton, President of the Plone Foundation)

*Even though I'm not a developer, the Plone community is a tangible, warm, friendly, inviting thing to me. It is the lens through which I see Plone. In fact, it *is* Plone. The software comes and goes, changes, ebbs, etc. It's the people that make Plone what it is.I'm also quite interested in the businesses in Plone. I see Plone's democracy as a level playing field for a vast network of small businesses, each able to align their personal/technical ambitions with their professional/career ambitions. It's a powerful combination.*

(Paul Everitt, Chief executive of the Plone Foundation and founder of the Zope Europe Association)

Certainly. It's very nice to get immediate feedback on one's work, and discussions on IRC and through email can be a very effective way to work out a new feature or solve some other sort of problem.

(Alec Mitchell, Independent IT consultant and core Plone contributor)

Yes. I build my business on thsi networked working and have my small network arround me where contracts are made. arround this bigger non-formal networks are existing. all i release is FOSS - thats only possible with such a working network where other do release FOSS too.

(Jens Klein, IT consultant and Archetypes release manager)

As I have been around for a very long time now but only in a passive way, I would call myself part of the community but definitly not of the core group. Possibly you have to drink some beers in real live with other members to get that status ;) As I know of the history of Plone I'll have a certain respect for the people that are responsible for this. It's their work and I am only one guy using it, so the community is definitly influencing my behaviour.

(Hanno C. Schlichting, IT consultant and core Plone contributor)

Yes, I feel a strong part of the community. Whenever we go to develop something we look around to see what the community is doing, not just to find out if someone has done what we want before, but the general direction the community is going with various trends.

(Matt Hamilton, Technical Director of Netsight Internet Solutions Ltd.)

I have code in the Plone collective, and I can answer some questions in the IRC channel. I feel the Plone community welcomes everyone who can do anything to contribute back. I feel I am a part of the community. I also feel as I can bring my experiences of the free software world to the community.

(Matt Lee, Senior Developer at NHS Connecting for Health)

(Aspeli, 2005:45, 53, 50, 63, 70, 67, 71, 73)

The participants answering the questions all feel part of the community, but they use different arguments on why. Runyan mentions personal relationships and the “reinforcement of commitments” by the community. Burton says a bit about changing tasks and position in the community and about non-technical work. Everitt also has a non-technical involvement in Plone and talks in very positive terms of the atmosphere of the community. Mitchell mentions the feedback and discussions on mail and IRC as effective and positive. Klein says that he base his business on the network and also points out that this network is part of a larger network made possible because they are part of FLOSS. Schlichting differs between the community and the core group. He argues that his longtime (passive) involvement is a reason

why he call himself a member. He also mentions the social act of "*drink some beer in real life with other members*" as a criteria for being a member of the core group. Hamilton says that looking around in the community for what is going on, and trends for where its going, are important. Finally Lee mentions that he has code in the Collective[1] and that he can answer questions on the **#plone** IRC channel. He talks of the community in positive terms, in that it is welcoming to participants who bring something back. He also says that he can bring experience from the larger community of the free software world into the Plone community, which can be a important way to contribute back.

I have so far concentrated on my own and others learning through legitimate peripheral participation in the Plone community of practice. I will now move on to examine more specifically how the **#plone** IRC channel is used in participation.

5.2.Interaction analysis - dense interaction on IRC

Making sense of what you read in the IRC channel can be hard for a newcomer. The expert-language is a mix of IRC-slang, plone jargon, file format names, catalogue paths, hyperlinks, emoticons, abbreviations, programming code and regular written english. In expert communities like Plone, the messages are immediately dependent on meaning on the social setting that occasioned them. In IRC, the participants do not only take part in the activities, but they participate in information flows and conversations, where they can make sense of what they hear and observe. Situational awareness is important for effective decision-making and performance in any complex and dynamic environment. An important element for sharing tacit knowledge is sharing experiences. Because 'tacit knowledge' is tacit, it is not easy to recognize, and thus difficult to analyze. In this chapter I will focus on a communication channel where tacit knowledge can be shared through socialization, sharing experiences, observation and imitation, and through dialogue. The first part of this chapter is about characteristics of how IRC as a medium is used in Plone community. This includes how elements are used to structure the conversation, different awareness-functions and language. I will then look at some examples from **#plone** to focus on what actually is talked about, and how it is connected with learning and knowledge-sharing. At the end of the chapter I will examine how other communication channels are used and locate types and aspects of

1 The Collective is the central code repository for a collection of Products for use with the Plone and/or Zope CMF Frameworks. Developers of new Plone Products are encouraged to share their code via the Collective so that others can easily find it, use it, and contribute fixes and improvements.

knowledge and focus on how the knowledge can be converted between tacit/soft and explicit/ hard in interaction. Learning how to use IRC, both technically and socially, is important to be able to participate in the community.

5.2.1.General use of IRC in #plone

There are between 100 - 170 participants in the **#plone** IRC channel 24 hours a day. The "room"-metaphor is used about a channel in IRC, and it suggests a gathering on a location, which implies a social meeting. The participants use nicknames to identify themselves; some use their real name and some create new ones. The nicks are often used outside the channel as well, and some participants use their nick on the mailing list, in code, and when posting to **plone.org**. The nick identifies the participant throughout the project and sometimes also in other projects. For some people, their whole 'virtual life' is behind one nick, while others might use different nicks for different communities, or just use their birth name. Greetings are sometimes used when a participant enters the channel and see people they know. A regular greeting is to type the name of the person you want to greet and then a smiling emoticon or "!". This is a way to recognize the personal relationship, and it is possible to see who knows each other from whom they greet.

On **#plone,** several "threads" of discussions, involving different topics, often take place at the same time and this can be confusing. The way to address the messages to a certain person is by typing his nick first in the sentence followed by a colon. The participant often engage in more than one conversations at the same time. My experience from **#plone** suggests that a newcomer typically hangs on to the first who gives an answer. I gradually learned more and more nicks of people who I knew had answered my questions earlier. After a while I also knew who answered what kind of questions. The old-timers serve the role as hosts, are often well known to each other, and have in many cases met in person. For the experienced users who take part in several threads at the same time, there are less defined starts and stops of the interaction. Because they stay in the channel for several hours there is no need to do a self-presentation, then ask a question, and then a "*thanks*" or "*ok*" to end the particular message.

Sometimes a participant wants to check if a certain person is present at his computer. This is sometimes done by using the 'ping' command in a social way, i.e. *fred: ping john* means that fred wants contact with fred. Ping (Packet Internet Groper) is originally a computer networking command that let you verify that a particular IP address exists and can accept

requests, but on **#plone** it is used as a metaphor to check if a person is available. This mix of regular language and computer language can be seen on several occasions. The technical activities of the practice clearly colors the language and provides meaningful metaphors understandable by the participants.

Sometimes participants change their nick as an awareness function by adding another word after a pipe symbol (i.e. *coolj | lunsj* or *cyhyb | train*). In this way the participant indicates something about what they are doing outside the channel. If someone changes their nick to *plmxn | phone* the participants know that person is on the phone, and that he is not available at the moment. The variation in messages is big, and sometimes it is used in the same way as emoting (see later in this chapter).

For larger amount of code or tracebacks[1], a paste-bin exists at **paste.plone.org**. The paste bin allows the participants to paste code to a web-page and use the short url to reference to the code they are talking about. This functions as an referential anchor and is important for the discussion. Participants does not paste more than a line or two of code in the IRC, because it reduces the readability of the text. If someone paste a chunk of code in the channel they are likely to be corrected and informed about the use of **paste.plone.org**.

In the IRC channel there are certain fixed roles. ChanServ[2] is an IRC service which maintains channel registration and access information. If a channel is registered with ChanServ, its owners, and those they have designated, can use ChanServ to obtain control over the channel, and gain channel operator privileges. An operator has the possibility to throw out or ban participants. By looking at who the operators or halfops (users with certain privileges) are, it is possible to see who the old timers of the community are and who has responsibility and possibility to set the topic of the channel. The topic is displayed on entering the channel and gives information about what is going on and links to useful resources:

> *Bug day in #bugday | Plone 2.5 beta released! | Latest stable release: Plone 2.1.2: http://* ***plone.org****/download | Development tracker: http://trac.****plone.org****/plone/milestone/2.1.x | Plone blogs: http://planet.****plone.org*** *| Have a question? Read http://****plone.org****/documentation/howto/ask-for-help first! | Use http://paste.****plone.org*** *for code or tracebacks. | API docs: http://api.****plone.org***

1 Traceback is an error message generated by the software that addresses the root of the specific problem.

2 More information about ChanServ can be found at http://en.wikipedia.org/wiki/ChanServ

*(topic for **#plone**, 07.04.06)*

Abbreviations like *IMHO* (in my humble opinion), *IIRC* (if I recall correctly), *FYI* (for your information) and *BRB* (be right back), saves the participant typing time and is an important part of the language. To learn what each of them means can be done by asking. An advanced communication technique in IRC is "emoting". The asterisk sign is used as the first letter in a sentence to signalize that a participant talks of himself in third person, and it often describes an action. The nick of the person who says something will then be the grammatical object of the sentence and the nick is not displayed in front of the sentence. These examples are from **#plone**:

*~~O~~ct 28 14:28:11 * kutaCEA sings o/~ I wish I was in Tiajuana, eating barbecued iguana o/*
*Dec 10 10:35:48 * Jean_l nods*
*Nov 04 02:14:10 * amaryll exports from iphoto directly into a dav folder :)*
*Dec 10 14:39:27 * Vinfan gives Sylves editor rights to fix them*
*Nov 18 22:59:15 * amaryll flings poo at miMal*

Emoting has it roots in textual role-playing games called MUD, invented in 1979 and popular in the 80's. Through emoting one can describe fictional or actual feelings, actions and thoughts, and it is a very rich, playful activity that includes features which does not exist in regular verbal communication.

Emoticons, also called smilies, is a sequence of ordinary printable ASCII characters, such as ":-)", ";o)", "^_^" or ":-(" are intended to represent a human facial expression and convey an emotion. Emoticons are a part of emoting and can give social cues and enriches the expression. One may see them as poor substitute for social cues like facial expressions and vocal intonations which takes place in face-to-face communication, or one can say that they constitutes their own sign-system to support the expressions of emotions.

The Internet Service Provider (ISP) information that is displayed in IRC when you enter or quit the room gives geographical and institutional clues. When I log in at the University of Bergen everyone can see that I am from the University of Bergen, and that I am located in Norway from the address ".uib.no". This is an awareness function that is not directly very useful or important, but adds a bit of information about the person behind the nick. The use of

bots[1] on the channels, is a way of letting agents do the job of sharing information. An example of this is the zopebot in **#zope** that reports news from planetzope.org, with a link to the site containing the news-item. A "leave message" set by the participant in his or here IRC-client to be showed when the participant exits the room, says something about the participants world and his social context. Examples of such messages can be "*client quit*", "*error*", "*goes to bed*", "*off to work*" and so on.

Creating a new IRC channel is easy, and is often done when there is need for a specified room for communication. For example when its bugday[2] in Plone, the channel **#bugday** is created. Other sub-communities of Plone use their own channel for more specific discussions. The structure of the access learners have to ongoing activity is important for the participation and potential learning; everyone using an IRC client with Internet connection can join the channel and the access to **#plone** is open.

Communication on IRC is synchron and makes it possible for immediate corrections of misinterpretations and misunderstanding. It also gives the possibility to give specifications and additional information to a question or answer. The concept of 'common ground' is frequently used in the CSCW/CSCL literature and refers to a state in interaction where a mutual understanding is established. Common ground can also be described as a mutual state of common knowledge, meaning and presumptions, between the participants in a conversation (McCaathy, Milse, Monk, 1991). The concept of 'common ground' is created by Herbert Clark, and is related to the concept of 'grounding', which refers to the process where a common mutual understanding is established and sustained. In other words, grounding can be said to be the process that maintains effective communication, and secures a common internal understanding, mainly on a linguistic level.

By lurking[3] or reading occasionally what other participants write, it is possible to pick up information about ongoing activities and what is talked about, without being active in the conversation. An example of this is when I found out about the Plone Foundation Board

---

1 A bot is common parlance on the Internet for a software program that is a software agent. A bot interacts with other network services intended for people as if it were a real person. One typical use of bots is to gather information. The term is derived from the word "robot," reflecting the autonomous character in the virtual robot-ness of the concept.

2 Bugday is a regular effort to get rid of bugs from the collector. http://plone.org/development/info/participation/

3 In Internet culture, a lurker is a person who reads discussions on a message board, newsgroup, chat-room or other interactive system, but rarely participates.

election, which I did not know about before I saw someone talk about it in **#plone**. Another example is the conversations about the Europython conference, when people discussed who were going, where to stay and other practical issues. On IRC you are in a social setting, where things happens and are discussed. With the high amount of participants logged in to the **#plone** channel and the low amount of participants talking at the same time, I presume that a large number of the participants are lurking. Lurkers are reported to make up a sizable proportion of many online communities, yet little is known about their reasons for lurking, who they are, and how they lurk (Macdonald et al. 2003). Lurking is an interesting phenomena seen in light of legitimate peripheral participation. Seely Brown implies lurking as a positive role:

> *The culture of the Internet allows you to link, lurk, and learn. You can move from the periphery to the centre safely asking a question—sometimes more safely virtually than physically, and then back out again. It has provided a platform for perhaps the most successful form of learning that civilization has ever seen. (McDonald et al. 2003:2)*

This idea of lurking is picked up in the CoP literature as a legitimate role for members. Wenger (1998) sees lurking as legitimate, therefore it can be classified as legitimate peripheral participation. In some situations, e.g. over a long time without any involvement, lurking is not legitimate, and can be seen as selfish and non-reciprocating. Being only a spectator is often not enough, but lurking can represent various ways of participating, and the context of the CoP decides whether lurking are legitimate.

In the Plone community, lurking is proposed as a way to enter the community, and observing before engaging is advised. Lurking is a form of cognitive apprenticeship which can be seen as LPP (Macdonald et.al. 2003). Macdonald et. al. speaks of an optimal situation for a CoP which describes the Plone community very well:

> *An optimal situation for a CoP would be to create an environment where the participants are motivated perhaps by a problem to be solved, to pull specific information to themselves, but at the same time have some "outsiders" lurking in the background providing information that may come from a different view. When this new and different information is inserted into the CoP, the participants will analyze it and determine its usefulness. This may be an important role that the often under-appreciated "lurker" can play. (Macdonald et. al 2003:7)*

Lurkers are eyeballs in a sense, from whom you rarely hear anything. Lurking as a way of being newcomer can also be seen in comparison as the newcomers in Anonymous Alcoholics where listening to others life stories was an important way of shaping the identity and learn.

IRC is a powerful tool and rich medium that allows for complex communication that is well suited for discussing technical issues where text in various formats, such as code or error messages, are important. The multitude of others ways of expression in IRC, make communicating different types of knowledge possible. By the use of different awareness-functions, emoting and the copy/paste function, problems can be discussed simultaneous with ongoing activity. The feeling of presence in synchron communication is quite fun, especial around releases or when things happen. The feeling of taking part in something, together, at high speed, that requires concentration, skill and effort can be thrilling.

I will now look at some typical activities of the Plone community of practice, before I will analyze some specific episodes from the **#plone** IRC channel.

5.2.2.Typical activities of a community of practice - from #plone

Wenger (1998) uses some examples which he holds as typical activities of a community of practice. By providing some examples from **#plone,** I will show that these activities can be found in the Plone community. The examples are not a coherent sequence of text. The box titles are the names of Wenger's categories of typical activities :

Problem solving:
- Oct 28 16:05:53 Lynger Oerland, are you saying it is a client side issue when you say local proxy *- Oct 28 16:16:23 DCFRS Hi I'm having a problem with Archetypes actions. The action is not rendering instead it is defaulting to base_view - however the page template I call in actions appears as a tab next to edit tab* *- Nov 07 23:20:49 red_shades I have a problem where every access to a plone site is requiring HTTP authentication. Anyone got an idea what could be going on?*

Request for information:
*- Oct 28 17:49:42 Celestrus Any alternatives to this http://**plone.org**/documentation/faq/importing-2.0-content-into-2.1 for converting Plone Folders to ATFolders? and I'm pretty sure that there is a very simple solution to my problem. Can someone indicate to me if this has already been done, any pointer to documentation, please?* *- Nov 03 15:50:12 Sylves Vinfan: Is there a documentation somewhere, how to implement that? RichDocument?* *- Nov 08 10:28:13 Meni hi, i'm looking for a good tutorial on actions / validators creation (using metadata files and python scripts). Can someone point me to such a document ? thanx*

Seeking experience:
- Oct 28 14:43:00 Carmen hey guys, I was wondering if anyone was using ubuntu breezy badger and was having problems getting /etc/init.d/zope2.8 to run *Nov 24 02:35:17 zorb anybody here with Python 2.3.5/PIL/readline and DarwinPorts experience?* *Dec 06 05:37:57 WhiPer zorb, do you know how to translate tabs from a 3rd party product*

Reusing assets:
- Nov 07 22:31:25 araguta Asalea: you can reuse your view class with multiple templates *- Nov 18 22:22:06 amaryll maybe components from quills can be reused in easyblog?* *- Nov 22 17:19:59 2PDF2 r0bur: you could create a custom (arche)type that referenced an arbitrary number of weblogs, sucking out their WeblogEntry objects. You should even be able to reuse the weblog_view template, and also the syndication stuff* *- Dec 09 18:35:13 chen so it's always nice to see some reuse of code that's written for such reuse. :)*

Coordination and synergy:
*- Oct 28 17:53:35 *U_8897u_8897 and cyhyb|train shake hands in agreement* *- Nov 15 09:55:07 araguta I think our understanding is pretty close, but it would be good to have some coordination probably* *- Nov 15 09:56:17 epithet as a community, it's my goal that our rel mgmt process for "next version" (kill the "2.2" meme!) be much more open (ie, the schedule news comes only from infraspec, there's a page listing progress, and he understand he has authority)*

Discussing developments:
- Nov 02 14:53:00 cyhyb smlgfrmmars: theoretically , yes - and i think U_8897 is planning to make something like that *- Nov 02 15:32:05 smlgfrmmars byte2b: but thats a thing where discussion is needed* *- Nov 02 19:41:10 byte2b frutiella: is the current plan for plone 2.2 to run on zope 2.9 ? and if that is the case ... will plone officially support py2.4 ?*

Documentation projects:
- Apr 15 15:24:04 Querk physodes: if you are interested in getting more people onto APE, you have to campaign for it, and not at least provide working examples and good documentation. *- Nov 18 14:02:02 valhall pr0gg: mind putting that into a short mail to the archetypes-users mailinglist? Might be handy for the google-findable documentation.*

Visits (or physical meetings):
- Nov 15 20:57:55 Tim I hope that we can arrange a sprint and get things in better shape *Nov 20 01:18:35 smlgfrmmars epithet: we have to push AGX. I want to organize a sprint (max 15 people) for AGX developement 24th to 29th March 2006 in innsbruck after "Modellierung 2006" a conference on modeeling from german society of computer science (GI)* *Nov 19 00:14:48 Frodo epithet: might you be attending the CalSprint Dec 2-5?*

Mapping knowledge and identifying gaps:
- Nov 05 22:01:49 cyhyb smlgfrmmars: but Oerland is the mastermind *- Nov 07 23:06:31 araguta search for xliff in the collective* *Oct 29 02:26:17 epithetepithet python runs in my blood, but uml just occupies a teeny tiny wedge of my brain* *- Nov 24 18:04:56 majalis Olsen, if you need to get dirty, python will not be the obstacle, but the knowledge of the framework*

As we can see, all activities Wenger mentions as typical for a community are found in the Plone community, in fact the categories can be used to describe the main activities of the community. It is important to mention that **#plone** sometimes also is a place for off-topic conversations, jokes and fun, like discussing music, personal interests, trips, and other real-life issues not related to Plone, at least for the regulars. This adds a social dimension to the channel that is important for the socialization and sharing of experiences and meanings and also for development of identity and a common understanding of the culture.

Unit, team or network ?

The Plone community of practice is different from a business or functional unit in that it defines itself in the doing, as members develop among themselves their own understanding of what their practice is about. The **#plone** channel is an important tool to develop this understanding through discussion. The membership involves whoever participates in, and contributes to, the practice. People can participate in different ways and on different levels. This permeable periphery creates many opportunities for learning, as outsiders and newcomers learn the practice in concrete terms, and core members gain new insights from contacts with less-engaged participants (Wenger, 1998).

The Plone community of practice is different from a team in that the shared learning and interests of its members is what keeps it together. Sometimes teams are created in Plone for certain tasks, like to develop a product, but the general community is defined by knowledge rather than by task, and exists because participation has value to its members.

The Plone community of practice is different from a network in the sense that it is "about" something, it is not just a set of relationships. It has an identity as a community, and thus shapes the identities of its members. To learn how to engage in **#plone** includes developing an identity. The plone community of practice exists because it produces a shared practice as members engage in a collective process of learning and collaboration.

I will now go more in detail and analyze some episodes from **#plone**. The content of the conversations are coded according to certain categories (see Appendix 3) using a qualitative research tool that allows for tagging and searching the tags in the texts.

5.2.3.IRC Episode 1 - SQL, storage and Ape[1]

Following Silvermann (1993), I identify sequences of related talk, and try to examine how speakers take on certain roles or identities through their talk. I also look for particular outcomes of the talk where learning might happen, or more specific; when a problem is solved. These are often marked by confirmations such as "*ok, so that is how*" or "*aha, thanks*". I describe the communication and try to say something about types of knowledge, negotiation of meaning, and how knowledge is created in the light of the earlier mentioned theories.

Claims are what people think, and by reading and stating claims one engages in making meanings visible. The first episode I will look at is a discussion about different products providing ways to store data in external databases outside Zope. The discussion is between old-timers. The role as "questioner" and "answerer" changes during the conversation, but some of them do not ask any questions. Through the example we will see that gaining domain knowledge is seen as a cost that can be more or less expensive depending on how well documented and accessible the product/domain is. Learning Plone/Zope and the different modules as a framework for development can be seen as an investment or cost where one pays with time and effort and gains knowledge and skill. The discussion is also interesting because a typical issue of FLOSS development is brought up; when to start a new project and when contribute to an existing project. All the excerpts given in the first episode are part of the same sequence of talk. The discussion contains 38 inquires, 12 abbreviations, (6 technical and 6 social), 11 acknowledgments, 14 emotes and 13 referential anchors.

---

1 The complete conversation can be found in Appendix 1, page 114.

The discussion starts with a participant called Querk, who has written a new Archetype-based storage-solution, and needs a place to put it on the web. Querk asks if he can put it in the Archetypes subversion repository (AT svn). cyhyb, netaual and Querk then discuss how to use the version control system (svn). They talk about how to create directories and naming conventions, and who to tell about the creation of the new folders. This is an excerpt of the first part of the discussion:

Apr 15 14:58:17 <Querk> guys, I have a new AT storage, and I need some place to put it
Apr 15 14:58:21 <Querk> can i put it in AT svn?
Apr 15 14:59:00 <cyhyb> Querk: there should probably be an area for storages like there is for fields/widdgets
Apr 15 14:59:07 <Querk> cyhyb: I can make one
Apr 15 14:59:13 <cyhyb> Querk: do that :)
Apr 15 14:59:19 <Querk> actually
Apr 15 14:59:25 <Querk> i can't... how do I make a new directory server side?
Apr 15 14:59:39 <cyhyb> :)
Apr 15 14:59:45 <netaual> Querk: create a local dir and add it to svn
Apr 15 14:59:47 <cyhyb> svn mkdir
*Apr 15 14:59:58 * Querk likes cyhyb's suggestion*
Apr 15 15:00:00 <cyhyb> svn mkdir full path
Apr 15 15:00:00 <Querk> MoreStorages?
Apr 15 15:00:12 <Querk> also... there is already a ReviewStorage in the root of the repo
*Apr 15 15:00:27 * cyhyb really likes the -more-something adressing scheme*
Apr 15 15:00:30 <Querk> shall I just make my path, then mail the list?
Apr 15 15:00:42 <Querk> cyhyb: ?
Apr 15 15:00:44 <cyhyb> (probably just because i came up with it)
Apr 15 15:00:47 <Querk> lol
Apr 15 15:00:56 <cyhyb> Querk: yeah, do that.
Apr 15 15:01:11 <cyhyb> and suggest to whoever is doing ReviewStorage to move it
Apr 15 15:01:37 <Querk> cyhyb: what's the correct way of importing a new product?
Apr 15 15:01:46 <Querk> i guess I want to have branches, tags, trunk directories?
Apr 15 15:01:56 <Querk> do I just mkdir each of those?
Apr 15 15:02:11 <cyhyb> yes. mkdir them all
*Apr 15 15:02:32 <cyhyb> then check out trunk. put your stuff in it, and svn add **
Apr 15 15:02:44 <cyhyb> that is the way i like it, at least
Apr 15 15:02:46 <cyhyb> :)

Apr 15 15:02:55 <cyhyb> gives you total predictability
Apr 15 15:03:17 <Querk> cyhyb: ok

The issues mentioned here are: where to put files on the web, commands for using svn, naming convention and who to inform about the changes. The naming convention "-more-something" that cyhyb mentions, is a way of naming the folders that the software will be stored in. The convention has to make sense to others, and learning the convention can be seen as an important part of the practice. Wenger (1998) suggests that a CoP will produce artifacts such as tools, procedures, stories and language which reify some of the practice. The naming hierarchies of a folder structure in a software application with its classes, variables and functions, is a good example of aspects of the expert language. Knowing the structure and naming conventions is something that one learns through reading, and using other participants´s code.

The discussion continues about different storage alternatives for using relational data base management systems (RDBMS) with Plone. The discussion involves 4 participants and concerns best practice, what type of person one is (self-definition), where one should put the time and effort and of course what storage product to use. Querk thinks that Ape, a Plone product, is badly documented and to complex for his use-case and therefore writes his own solution; a product called sharedSQLStorage. The module use components from SQLStorage, which is another storage application. physodes, who has been writing on Ape, wants Querk to contribute to the Ape product instead of starting a new project. Because of the deadline, he has to make it work today, and he explains that he does not have time nor knowledge to use the Ape product. The essence of the discussion is whether one should write new applications, or spend time fixing already existing applications. To use the older application one needs knowledge of the framework which it provides. Getting this knowledge is seen as a cost. I will illustrate the discussion with some excerpts.

Apr 15 15:06:15 <jacqulin>Ape seems too complex
Apr 15 15:06:32 <jacqulin>and it cannot really make my existing SQL records appear as objects
Apr 15 15:06:35 <jacqulin>or can it?
Apr 15 15:06:38 <Querk> physodes: it may well be, but i don't know about APE, so poo on that

Querk do not have enough knowledge of the Ape product, so he doesn't even consider using it to solve his problem. physodes insists that he should use it:

> *Apr 15 15:15:32 <physodes>i think Ape could be refined quite a bit to be made simpler... but i think its an amazing framework for connecting plone to an rdbms*
> *Apr 15 15:15:34 <Querk> I'm sure it's great. It's just not accessible enough right now to me*
> *Apr 15 15:15:58 <Querk> physodes: then get some nice examples + simplfying layers together and make the world a better place :)*
> *Apr 15 15:16:01 <physodes>Querk: well then, rather than write your own sql thing... why not work on doing docs and making ape more mainstream?*
> *Apr 15 15:16:03 <Querk> it's something we need to get better at*
> *Apr 15 15:16:14 <Querk> physodes: because deadline is today*

Querk has a deadline the same day, and has to deliver a working application with the required functionality. The lack of documentation in Ape makes it inaccessible for him. Knowledge enables the developers to do their job, but if the knowledge in one domain (ie. a product) is to hard to access, because of lack of documentation and examples, its easier to choose another solution. The main reason for frameworks to exists is to enable the developers to do rapid software development (RSD). If the framework hinders the rapidity (i.e it to hard to learn it), then its not a good framework for the specific use-case. This is expressed quite clearly by Querk. This leads to some interesting self-defining statements in the discussion:

> *Apr 15 15:06:54 <jacqulin>i am a bottom-up guy*
> *Apr 15 15:07:23 <physodes>i am a make-existing-stuff-work-rather-than-build-new-stuff guy*
> *Apr 15 15:07:43 <Querk> physodes: me too. except when that old stuff is poorly documented or overly complex for my needs :)*
> *Apr 15 15:07:49 <Querk> and I *do* build on SQLStorage :)*

The asterix in *do* is used to put emphasize on the word, Querk really does build on existing software, he does not start from scratch. The participants seems to agree on the basic principles that it is better to modify existing code, rather than to ‘reinvent the wheel’, but several factors are present: time, domain knowledge, and the design of the software. If a programmer does not like a piece of software, they might not want to write on it, but start a new project instead.

The participants use emoting to express what they feel or think. This is done several places in the discussion to express feelings or actions. The following excerpts describes some of the aspects of the discussion where the developers talks about themselves in this way:

> *Apr 15 15:10:27 * physodes dislikes that there's a dozen different products with the expression 'SQL' in its description all aiming to do the same sort of thing*
>
> *Apr 15 15:12:08 * cyhyb likes products to do as little as possible*
>
> *Apr 15 15:15:19 * Querk will use APE when ti's mainstream enough that he doesn't have to hunt for documentation and examples and venture into a minefield of missed deadlines because something that is really simple becomes too complex with a big and mysterious framework*
>
> *Apr 15 15:21:16 * physodes is just very frustrating watching people overlook ape because it seems too complicated or because SQL databases are too slow... if any of that is truly the problem, then fix it rather than create some other niche product*
>
> *Apr 15 15:29:14 * jacqulin hates SVN*

In this discussion I found 14 cases of emoting. Most of them started with personal preferences like "likes", “dislike”, “will use”, “is inclined”, “frustrated”, “hates” and “thinks”. This way of sharing emotions is linguistically different from expressing it directly, because it makes fictional actions, thoughts or mental states visible. In face-to-face communication, body language and facial expression is included in the communication and is an important part of the identity of the participant. (is he/she skeptic, positive, big-mouthed, humble .. ?) Emoting in IRC makes this part of presentation, or identity-making possible.

Querk then sums up his problem of time, knowledge, access and money in clear words to physodes:

> *Apr 15 15:23:29 <Querk> physodes: you asked me why I don't spend my paid for, limited time fixing your "I think this is so great" technology. The reason is that it's not accessible to me in the time I have, and I won't risk total project failure (and thus not getting paid) if I'm uncomfortable about the technology. I don't know what I'm getting myself into.*

Another domain-specific, or rather product specific, knowledge can be seen in the continuation of the discussion. Querk and zorb does not know that SQLStorage, which

sharedSQLStoreage is built upon, is caching content. Because jacqulin, who just worked on the cache of SQLstorage, is present in the chat, this get cleared up.:

Apr 15 15:31:54 <zorb|nearby>SQLStorage is also slow as hell
...
Apr 15 15:32:09 <Querk> zorb|nearby: yes, because it doesn't cache.
Apr 15 15:32:11 <jacqulin>zorb|nearby: but it has caching...
Apr 15 15:32:15 <Querk> it should, and it wouldn't be too hard
Apr 15 15:32:16 <jacqulin>Querk: it does.
Apr 15 15:32:20 <Querk> jacqulin: are you sure?
Apr 15 15:32:23 <Querk> where?
Apr 15 15:32:27 <jacqulin>yeah, spent all night on the code.
Apr 15 15:32:29 <jacqulin>hold on.
Apr 15 15:32:42 <jacqulin>SQLMethod
Apr 15 15:33:11 <Querk> ah, right
Apr 15 15:33:19 <Querk> that's lower level, didn't think of that
Apr 15 15:33:24 <Querk> any idea how it invalidates?
Apr 15 15:33:26 <jacqulin>i am using that now.
Apr 15 15:33:34 <jacqulin>Querk: same as ZSQL -- time based.
Apr 15 15:33:51 <Querk> right
Apr 15 15:33:56 <jacqulin>and it has to be configured, it's at 0 by default.
Apr 15 15:34:04 <jacqulin>zorb|nearby: that's why it's slow... it can get faster.

This excerpt is about a function, its location in the code and the possibility to configure it. zorb thought that SQLStorage was slow because it didn't have cache, but jacqulin who "spent all night on the code", knows about it and inform the others what function provides the caching, and that it is set to 0 at default and that that might be the reason why its slow. Jacqulin's sharing of his domain specific experience corrects Querk's and zorb's understanding of SQLStorage, and they learn why its slow and how to change it. In this case jacqulin was a resource because of his knowledge of the code of SQLStorage. The modularization of the code-base facilitates domain-specific knowledge held by different participants.

The conversation continues and the participants discuss Ape more in detail. Jacqulin installs Ape at the same time as he talks on **#plone**, so the problems he meets is directly brought up and discussed. This allows simultaneously experimentation and problem-solving, as well as

collaborative reflection on his owns actions. When you log into **#plone** you are not alone at your desktop.

The participants use humor, irony and sarcasm. At some points the discussion is heated. The lack of social cues makes it difficult to see the affection of the participants, but some feelings can be seen in textual outburst such as "*see, so htf am i supposed to get anywhere???*". Rude language and the repetition of the question-marks indicates some of the frustration. In the interview with Aspeli, Limi sees the possibility for heated discussions online in Plone as a positive thing:

> *"Another thing that separates the Plone community from a lot of other communities is the amount of real face-to-face communication we have. We organize (sic) workshops, conferences, informal gatherings - and most people have a lot of close friends in the Plone community. One of the strengths of this approach is that we can have really heated discussions about things related to Plone - but people are very seldomly (sic) offended, since they most likely have met the person on the other end in person - and know that he's a human being, that he's a nice guy." (Aspeli, 2005:21)*

This indicates that "socialization in the real world strengthens ties in cyberspace" (ibid).

Through the discussion the participants learn who knows what, where files are stored and the features of certain programs and functions. They exchange opinions and likes/dislikes of things, and they discuss each others actions. Through emoting there is self-presentation and identity building. What to learn is individual, and different from newcomers to core developers. A newcomer who has not yet developed the appropriate knowledge will not have the same level of understanding as an old-timer.

5.2.4.IRC Episode 2 – Rockstars and UML Design[1]

This episode involves several old-timers and a plone newcomer with a lot of experience from another FLOSS project. It contains 15 inquires, 5 abbreviations (3 technical and 2 social), 8 acknowledgments, 8 emotes and two uses of referential anchors. The plone community has establish the word rockstar for old-timers, or core developers. In FLOSS in general the word newbie or n00b is used for a newcomer, while often 'guru' or 'hax0r' are used for old-timers or masters. In Plone old-timers, or participants that have made a considerable contribution, are

1 The complete conversation can be found in Appendix 2, page 121.

called ‘rockstars’. The episode starts with a discussion about who the rockstars are and continues with a discussion between one rockstar and the mentioned newbie.

Oct 29 02:22:00 byte2b *caneose: man you lucked out tonight ... got zorb, epithet, and Tim all helping you out... 3 plone rockstars!*
Oct 29 02:22:11 caneose *byte2b: :-)*
Oct 29 02:22:13 zorb *caneose is a rock star in his area ;)*

The newcomers might have different backgrounds, some might come from other projects where they are old-timers themselves, or they might be new to web development, and literally code-illiterates. Lave and Wenger (1998) suggests that we go out and in of different CoPs during a day. In the conversation, zorb points out that caneose is a ‘rockstars’ in his area, but in **#plone** he is a newcomer. The developers are often part of several projects and have a different peripherality in different projects. The reason why he gets so much help (from three ‘rockstars’) might be because they know about his background, and, as we will see later, they are interested in him as a potential participant. A discussion on who the old-timers or ‘rockstars’ are, and the start of another discussion, can be found in the continuation:

Oct 29 02:22:14 epithet *byte2b: stop being so modest. you're a plone rockstar.*
Oct 29 02:22:24 zorb *byte2b: ever heard of SVN? ;)*
Oct 29 02:22:30 byte2b *epithet: nah, i'm just a choir boy ;)*
Oct 29 02:22:33 epithet *Tim is the rock star. everyone else is merely but a bass player with ambitions*
Oct 29 02:22:34 zorb *he wrote it (not alone, but... ;))*
Oct 29 02:22:41 byte2b *zorb: hehe, yep, i know where caneose is from :)*
Oct 29 02:22:51 caneose *[not from MIT!]*
Oct 29 02:22:55 byte2b *lol*
*Oct 29 02:23:06 * Tim blushes*

Epithet says that byte2b and Tim are both ‘rockstars’. By negotiating this, they get a feeling of how others perceive them, which is a important part of their identity. Identity is made up of who you are and who other say you are. Zorb says, by emoting, that he wants to involve caneose in an svn project for CMF, and amaryll would also like to involve him:

*Oct 29 02:23:12 * zorb wants to involve caneose in the SVN-backend-for-CMFEditions*

Oct 29 02:23:15 zorb ;)
*Oct 29 02:23:21 * caneose tries to parse epithet's pattern description... hold on*
Oct 29 02:23:32 zorb I'm sure caneose would love to have his Plone content backed by SVN ;)
Oct 29 02:23:35 byte2b whoa, CMFEditions-on-svn would be nice!
Oct 29 02:23:38 caneose woo!
Oct 29 02:23:51 zorb frutiella and WhiPer had a working prototype at one point
*Oct 29 02:23:57 * amaryll might like to involve caneose in svn tracker and repo control integration for plone :)*
Oct 29 02:24:01 zorb not sure where it went
Oct 29 02:24:10 caneose svn tracker, eh?
Oct 29 02:24:15 zorb amaryll: I'm sure caneose has enough SVN duties ;)
Oct 29 02:24:23 amaryll like trac.
Oct 29 02:24:29 caneose well, not anymore, purely voluntary svn work now

Caneose is one of the core svn developers, and is a old-timer in the svn project. Further down in the conversation we get to know that he is now only doing voluntary work for svn, and this might indicate a move in the peripherally for caneose in the svn project.

Then a discussion about a recipe program follows with the same participants. They discuss if a ‘food’ category should be made as a class of its own or not, and other system design issues. Caneose is guided by epithet in a apprentice/master relationship, and the conversation ends with epithet sending caneose a tarball[1] of a recipe product. The email address is exchanged through private messaging in IRC. This is a function that let two or more participants talk privately.

In the conversation, epithet gives Tim an interesting comment concerning documentation:

Oct 29 02:28:59 epithet Tim, others: some docs on how to turn your needs into an AT plan would be a fantastic piece of docs, tho' probably hard to write

The externalization of the knowledge on how to "turn you needs into an AT (Archetypes) plan" is an example of knowledge which is both soft and hard. Tim has the knowledge, but unfortunately the knowledge is not externalized, or made available in a document. Why it is hard to write we can only speculate in, but it might be that the knowledge is of a soft character

1 Tar is a file compression format, in this case a package of source code files

or that it is very complex. Handling knowledge in big time collaboration needs to balance the existence of knowledge in external and internal forms. Knowledge resides in the head of people, but has to ‘come out’ and manifest itself in documents and tools. My opinion is that IRC makes knowledge held in peoples heads more accessible, and meanings more visible.

I will now leave LPP and individual learning for a while and focus more on how the community learns, and look the balance between soft and hard knowledge in the next chapter.

5.3.Knowledge-sharing and conversion

This chapter examines the way knowledge is created and shared by looking at various knowledge-holding resources and by looking specially at places where tacit knowledge can be shared. I am using Nonaka's knowledge-conversion spiral to explain aspects of how knowledge is shared in Plone. It is important to point out that this is about organizational knowledge, about how the Plone community, viewed as an organization, learns and manages different types of knowledge.

5.3.1.A duality - locating soft and hard knowledge

Despite of the uniqueness of each FLOSS project there is often a common way of structuring knowledge assets and resources in many FLOSS projects. Sourceforge and Savannah are the largest FLOSS project management sites on the Internet, and many projects use these, or similar services, as a management system, in addition to the projects own web-sites. The projects web-site serve as an entry point, and face outward, for the project. Many FLOSS web-sites has a 'get involved' / 'Join the community' section with links to wiki, IRC, mailing list and a bug trackers, as well as links to externalized knowledge resources. The artifacts in Plone are produced by the participants themselves, and they use Plone CMS to organize knowledge-artifacts such as procedures, instructions, tutorials, and how-tos. Wikis are used to create a collaborative online document where all the participants are invited to share their own knowledge and experience. The svn is used to store the code base, but also contains comments from the developers about what is submitted. The documentation section on **plone.org** is divided into "faqs", "how-tos", "tutorials", "reference manuals", "error reference", links", "glossary entries", "migration instructions" and a list of books about Plone. In addition to this documentation, a huge amount of documents can be found on other sites related to Plone in some way. There is a lot of documentation online, and it can be really hard for a newcomer to find what is needed.

External resources are the explicit side of knowledge created through reification. Hildreth et al's (2002) participation/reification duality maps very closely to the hard knowledge/soft knowledge duality. Knowledge was said to be a soft/hard duality in that all knowledge was hard and soft - it is simply the proportions that differ. This means that knowledge concerning a certain domain is both soft (knowledge held by participants) and hard (knowledge externalized in documents). This is also true for participation and reification. If knowledge is

predominantly soft, then the participation proportion of the duality will be higher. Conversely; the harder the knowledge, the greater the proportion of reification.

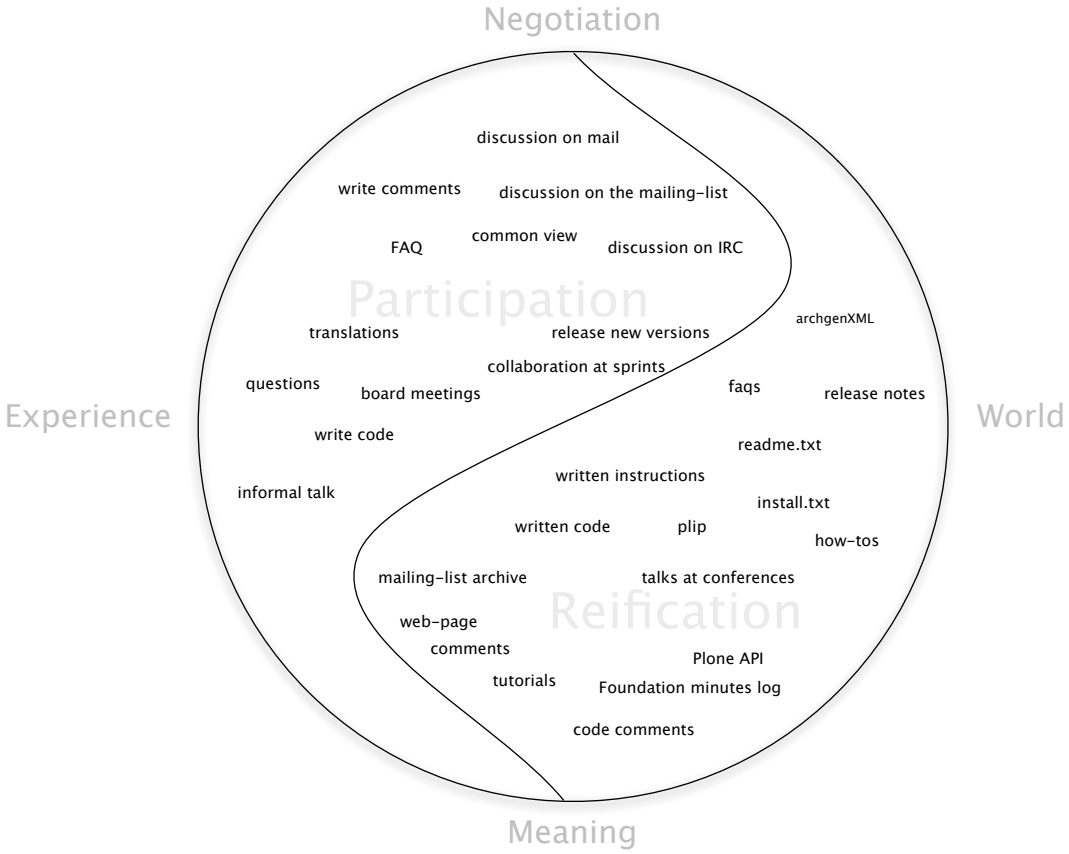

Illustration 6: The duality of participation and reification in the plone community of practice (original figure from Wenger 1998: 63, adapted by the author)

Illustration 6 shows how this is manifested in Plone. By looking at the learning curriculum and activities it is possible to map different knowledge-domains and identify participation and reification. In dynamic interaction like the mailing-list the differences in meaning is easily made visible. It is possible to enter the meaning exchange, not only the result, and see how the knowledge was created, and the process that created it. This is different from reifications made intentionally by members, like in how-tos or tutorials. Technology is used to store this knowledge-creation process, so it is available for others, now, and in the future. Illustration 6 shows Wenger´s figure (1998) of the duality of participation and reification, with the activities from the plone community of practice and the authors´ learning curriculum and use of resources. Plone´s high amount of reification makes it possible to be "all over" the project while sitting by the desktop. The reificative connections transcend the spatio-temporal limitations and afford possibilities for learning. Knowledge is not given by simple

institutional commitments, the ascription of identity, or assumed sources of power and authority. Rather, it is an outcome of interactions, negotiations, and interfaces between different participants.

5.3.2.Knowledge-conversion

Nonaka sees tacit and explicit knowledge as mutually complementary entities. They interact with each other in the creative activities of human beings. Nonaka calls the interaction of these two forms of knowledge the "knowledge-conversion process". This process consists of four modes: socialization, externalization, combination and internalization, which I have described in detail in chapter 3.2.1. I will now look at the activities in Plone in the light of these modes.

From tacit to tacit – socialization

Sharing experiences through collaboration on code, engaging in common problem solving and discussions of real-world use-cases are examples of 'tacit-to-tacit' knowledge-conversion. By observing and imitating in a social setting one engages in the common practice. In the Plone project, examples of the socialization stage is found in the places where technology, or physical co-location of people, directly links peoples together. This can be the IRC channel and at the mailing-list, or at sprints and conferences. Tacit knowledge is shared through interpersonal interaction. As we saw earlier in "Episodes from IRC", sharing experiences is an important activity through synchronous communication. Another example of this, which is presumably a better example of this stage, is collaboration and socialization at sprints with face-to-face interaction. At sprints, the developers team up in smaller units, and newcomers are paired with more experienced programmers to increase the learning. An example of social problem-solving can be the corrections of errors in the code, often called bugs. Testing a program, finding bugs, reporting bugs, locating the source of the bug and finally fix the bug is in many cases a process with several actors interacting around a problem mediated through different technology. End-users might discover a bug and get an error message, another might rapport it through a bug-collector, the developer who wrote the code might pick it up in the bug-collector and locate the source of the error, and another developer might write the hack to fix it. Severe bugs might even require larger parts of the program to be rewritten. Every friday is bug-day in Plone, and those who want to engage in bug fixing joins the **#bugday** channel on IRC, and grabs a bug at the collector. Bug-fixing is a process between different tools, both human and non-human, in a social effort (Østerlie, 2004).

Tacit knowledge is often context specific and at #IRC the one who asks a question has to precisely formulate the context of the problem. A well-phrased and well-thought-out question that demonstrates that you have already attempted to research the topic, and precisely explains the context of your problem, will get you more helpful replies. This stage depends on having shared experience and results in acquired skills and common mental models (Nonaka, 1994).

From tacit to explicit - externalization

Writing documentation is the main activity in this stage. As discussed earlier documentation can appear in many forms, but they are all resources made explicit from the individual or group who made them. Tacit knowledge is used when creating explicit knowledge, and all knowledge is rooted in tacit knowledge (Polanyi, 1967). There are a multitude of examples of externalized resources in the Plone project; books, talks, web pages, videos, podcasts, how-tos, tutorials, faqs, API and procedures. The effort made in creating these resources involves the participants own personal, actionable knowledge. Informal knowledge is the knowledge applied in the process of creating formal knowledge (Conklin, 1996). These are often explanations on how to solve a certain problem or perform a certain task. The one who writes the documentation has to know how to solve these problems to be able to write the documentation. Most of the resources are available on the net, but sometimes they can only be accessible live through physical presence, like at a talk. Sometimes even these events are made accessible through the net by video-streaming from conferences or sprints, or by publishing the recording of the talk after the conference[1]. An interesting question to raise at this stage is the motivation for creating documentation and external learning resources, which one doesn't need oneself. According to Nonaka, this stage comes in two cases. One case is the articulation of one's own tacit knowledge such as ideas and images in words, metaphors and analogies. A second case is eliciting and translating the tacit knowledge of others, customer, experts for example – into a understandable form. The training and support I did in my field study might be an example of this stage. I had to explain concept and ideas from Plone to the users of the publishing application. Another example which might show the difference between participation and reification, or tacit and explicit knowledge, is a post from the Eventregistration mailing list. It illustrates that different channels, or mediums, are used to share different parts of the knowledge:

---

1 Plone has got its own Internet-TV channel at http://mrtopf.tv/vlog/

> *corey, thanks for the offer. I'd highly appreciate any help having both 2.0 & 2.1 code inside a single release if possible, rather than releasing two versions. A single release was my original plan anyway, but I was not confident it's feasible enough. How about discussing this on IRC, at #geolocation@IRC.freenode.net? I tend to hang around on the channel during GMT working hours just in case someone wants to drop in to discuss the package. As a result of the effort, we could also compile a how-to on the topic. I would guess that'd be of some interest to the community at large as well. Thanks, mike-e (Eventregistration mailing list, 2005)*

Here the IRC channel is seen as a suitable place to discuss the 2.0/2.1 version problem, but he suggests that the outcome of the discussion should be compiled in a how-to document. The IRC discussion facilitates socializing and participation, while the how-to represents a reification or externalization of the same knowledge.

From explicit to explicit - combination

Finding, sorting, adding and categorizing externalizations of knowledge-resources is the main activity in this stage. Different tools such as **Google** search, web browser favorites, collections of documentation such as the **plone.org** documentation section, help locate and organize resources in different formats such as electronic books or code examples, web pages, videos, procedures and programs. **Google** is the leading search tool in the Internet. At its peak in early 2004, **Google** handled upwards of 80 percent of all search requests on the world wide web through its Web site and through its partnerships with other internet clients like Yahoo!, AOL, and CNN (Wikipedia.org, 2005). **Google** is the most important search-tool for finding the appropriate information on web pages, forums, tutorials, how toes, faqs and other external knowledge-resources published on the net. In my fieldwork, I used **Google** to search for keywords, then looked through the most interesting search results. If I didn't find what I was looking for I modified the query keywords and refined the search in accordance with the information I read at the previous results. Because of the importance **Google** has as a tool in the "combination"-phase of Nonaka´s knowledge-conversion theory, I will explain shortly how it functions. **Google** uses the PageRank system, which is a family of algorithms for assigning numerical weightings to hyper-linked documents to help determine a page's relevance or importance. Shortly explained, this means that a page is weighted after how many other pages links to this page, and it weights the importance of the word on the referencing page. This means that the relevance of the site is partly decided of how many other sites who links to it. A typical response for a simple newbie question at the **#plone** channel is "did you Google it ?", and this suggests that "googling" something is the first step to locate externalized knowledge. The **google** engine crawls both the plone mailing-lists and web-pages, as well as all other resources on the world wide web.

The most important activity for the project members for this stage is the spreading of resources in the form of "hard" knowledge throughout the project. Information technology like the Plone CMS is very important for organizing and publishing these resources. Everbody registered at **plone.org** can post documents to the site. Plone CMS has a search function that lets the user search all of **plone.org** or certain sections of it, such as the documentation part of the site. It is not only the the main project site who use Plone CMS for this job, many other important sources of information, like sub-projects, uses the Plone CMS application to serve their home-pages. Examples of this is the "My site" tutorial which uses Plone to organize an online book. In this stage *information*, not knowledge, is an important keyword. Plone is almost self-supporting in this way, except for using technology like **svn** and the program Trac[1] for organizing the code itself.

The use of indexed hyper-linked text as a way of organizing a distributed, large amount of information about a certain domain, is to complicated to start discussing here. But most of available externalized resources in Plone is available in this format, and **Google** or other search engines, is this most important tool at this stage.

From explicit to tacit - internalization

This is the process of understanding and absorbing explicit knowledge into tacit knowledge held by the individual. Knowledge in the tacit form is actionable by the owner. What I am able to to with Plone now is my actionable knowledge. The process of understanding, and thus making the knowledge mine, is not easily to point at or describe, but the outcome of my legitimate peripheral participation is my personal knowledge, that I might externalize through giving training and support, writing documentation or explain concepts to others. Much of this knowledge is gained through experimentation based on the information gained from explicit knowledge resources. A good example on this from my fieldwork was the use of the "My site" tutorial. The "My site" tutorial is created by Raphael Ritz and provides an introduction to file-system based product-development for Plone site-managers. The tutorial consists of two main parts: 1) a Plone product called Mysite[2] with source code that one installs locally on a server and 2) a book which explains the purpose of the product as well as all the individual files in detail. The book contains 17 exercises that help the reader with

[1] Trac is an enhanced wiki and issue tracking system for software development projects and can be found at http://projects.edgewall.com/trac/

[2] Mysite can be found at http://www.neuroinf.de/LabTools/MySite

testing and deepening his/here understanding within the learning-by-doing paradigm. An appendix provides solutions to most exercises and summarizes major resources and tools for developers. There is an online version of the book with possibilities to write comments, as well as a printable version. The online “back-talk” book contain useful reader-comments such as hints, errors, platform specific patches and hacks, criticism and questions. The book is a collaboration between the author and the readers. The interactivity of the tutorial, that you have to install it, and modify it through writing code makes it a good example of explicit-to-tacit knowledge-conversion as one learn through guided experimentation.

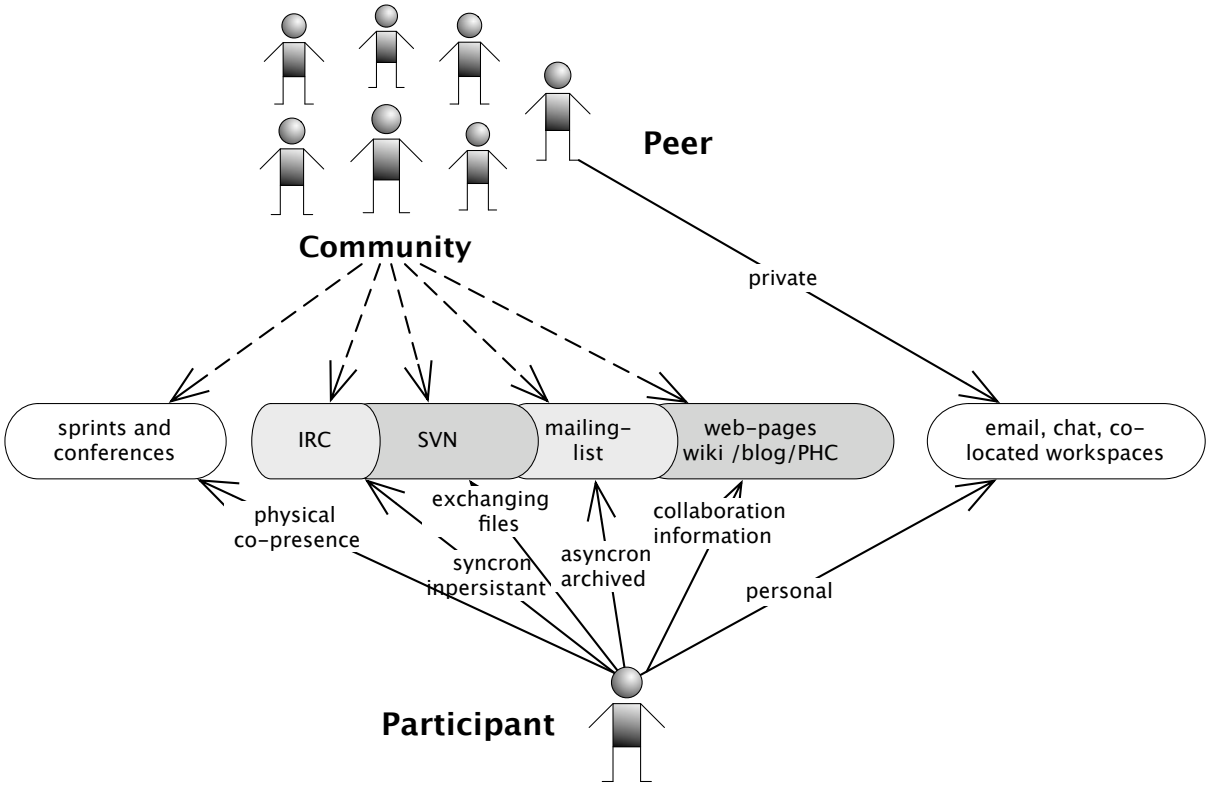

Illustration 7 : Communication in various formats and channels

Through IRC, svn, mailing-lists, web-pages, sprints and conferences, co-located workspaces[1], and private mail the community shares knowledge and learns from each other in various relationships. Illustration 7 shows the different computer-mediated and face-to-face communication between the participants, their peers and the community. These are interfaces where meanings are exchanged and both soft and hard knowledge is communicated and

1 Developers sometimes work within the same companies. Examples of such companies are Enfold Systems, Plone Solutions or Zest. Plone's "ecosystem" was build up by the founders with such small companies in mind.

shared. The various tools facilitates synchronous and asynchronous-, archived and temporal-, oral and typed-, personal and group communication through various media and channels. The outcome of the communication in the gray boxes in the middle is all open and accessible through the Internet, which says a lot about the accessibility to information and ongoing activity.

5.4.Analysis Summary

Project members rely heavily on network-enabled, asynchronous communication methods like email and discussion forums for both ad hoc and structured communication. The centralized storage of these communications significantly aids new project members in understanding the history of the project so they can contribute more swiftly. As a participant I can verify this, and these storages of past communication was a big resource for me. The mailing-list has a special place in the duality of soft and hard knowledge, as it is both a place for participation and reification. It enables many of the features necessary for tacit knowledge-sharing, and at the same time offers a searchable externalized knowledge resource for future learners. The use of it fits into all of Nonaka's stages. It is the fora of participation through negotiation and meaning exchange, and at the same time this "soft" knowledge found in participation, is created and reified by digital archiving, as a knowledge-resource for others in the future.

The distance between the communication (IRC, mailing-list) and the activity (e.g writing code) in Plone is very short. When experimenting and writing code at your desktop, the whole community is literally a key-press away. The activity and communication overlaps in two ways: in one way the programming-language and terminology from the activities becomes part of the natural language. On the other side the developers can influence the communication drastically by creating applications for communication. In Plone an example of the latter is the Plone Help Center[1]. The fact that the developers master programming-tools for developing communication tools gives them great power over their own setting, and enables them to do changes that increases the value of the ICT based communication. A programming-language is a tool for instructing machines to solve problems, and this power can enable engagement and a urge for the knowledge to solve these problems. The developers work in the field of communication technology, and are able to create and modify tools. The

1 Plone Help Center is an application designed to aid the documentation of Plone projects, and is used to organize and keep documentation up to date. http://plone.org/products/plonehelpcenter

fact that they know what is needed for their own communication, makes it an economic process, where they implement just the necessary functionality.

In Plone, tacit knowledge is shared through socialization, observation and imitation, on IRC and at sprints, but without individual efforts in externalizing knowledge, a newcomer would be helpless. The variation in learning-resources facilitates both sharing of embodied knowledge, and knowledge held in formalized instructional artifacts. All the modes of Nonaka ´s knowledge-conversion spiral are somehow covered in the Plone CoP. In many cases externalized documents (hard knowledge) allows for reflection and comments from the user, and becomes a collaborative effort. The channels for communication that facilitates soft aspects of knowledge (ie. IRC) also supports sharing externalized resources like code snippets and documents, that are often used as referential anchors that help the conversation, and reifies the subject in focus. The participants in Plone are highly skilled in computer-mediated communication, and participate in a practice where transparency and access to knowledge is high. The high skill can be connected with the fact that they are involved in a practice where the main activity is to develop solutions for computer-mediated communication, and in this way create the tools they use. Learning is an important aspect of the social practice of the Plone CoP and informal learning mechanisms, meaningful participation and receptiveness to complexity (Wenger, 2000) are facilitated.

6.CONCLUSIONS

"No one has ever completed their apprenticeship."
Johann Wolfgang Von Goethe(1749-1832)

The concepts found in the literature on LPP and CoP are general, and can be used to examine learning in very different settings. I used these theories as a lens through which I saw the ongoing activities in Plone. Hopefully I managed to give an account of the components of this lense, as well as what I saw through it. The theories are useful for looking at the world from a general "social learning"-theoretic perspective, but in my opinion they lack a notation on artifacts and instructional efforts, focusing only on informal learning. LPP is useful for understanding the activities in Nonaka´s first mode of knowledge-conversion where tacit knowledge is transferred through interaction. In FLOSS, knowledge is the key to both legitimacy, peripherality and participation. The most important point of LPP is that learning is an integrated part of all social activity and is promoted by participation. This might not be enough, however, to give an account of all aspects of learning and knowledge-sharing in a community of practice, and specially not for the technology of practice and Nonaka´s combination and internalization-mode of knowledge-conversion. The internalization-mode is what resembles traditional learning through instructions, and schooling most, and is not discussed in the theory of LPP.

We go in and out of many CoP each day, and we can not aim to be full members of all of them. The theory of LPP seems to take for granted that the aim for learning is always full membership, as it might be in more traditional master-apprentice relationships, and where the aim is to become master of a domain. LPP combined with the theory of organizational knowledge-conversion and the concepts of soft/hard knowledge and participation/reification, enables us to talk about aspects of knowledge and learning not explored in LPP. The externalization/internalization and soft/hard concepts may be too simplified, but by using them I was able to point at different aspects of knowledge, even though they might need a further discussion. The scope of this project does not permit me to go into the epistemological discussion of the internalization/externalization of knowledge, but this is clearly an issue to examine more closely in both the theory of LPP and the knowledge-conversion process. Other arenas with physical co-presence, such as sprints and conferences, might be important for tacit knowledge-sharing. Face-to-face communication was not in topic for this thesis, but it might be an interesting, and underestimated, element of learning in FLOSS as a subject for further studies.

In my own learning-process in Plone, the documents and artifacts containing externalized knowledge, and sometimes instructions through procedures, was important. If I was going to continue my learning process, it would mean participating more by taking on more responsibility and getting involved in more relationships. Understanding the technology of the practice is more than learning to use tools; it is a way to connect with the history of the practice and to participate in the cultural life of the community. The high level of knowledge that is already an implicit condition for joining, can make it hard for people with poor skills to join. Access to the tools and communication are open, but to actually get inner access to the practice, and become one of "them", is difficult. The skills of the participants at the core of the developer-community are high, and I assume that most of the participants have university degrees in computer science, and/or works with software development full time. To know the programming language, and other developer-tools, as well as to master the different forms of communication, requires a high level of technological knowledge, which the newcomer preferably should posses when joining. This might scare unskilled people away, but on the other side it might attract skillful participants in search of challenges. Another challenge for FLOSS is to keep the participants in the project after they have joined. In Plone there is no structuring elements around the increasing participation, like for example "the twelve steps to sobriety" an alcoholic has to go through in the AA community, or the six different positions the Naval quartermaster has to move through. In Plone the is self-regulatory, and the newcomer do not get any help in organizing the learning process from such a structure.

FLOSS projects is sometimes a second priority for the participants, after paid work. As we saw with the Eventregistration project, the project was abandoned when McVetta did not have enough time do develop it. This implies some insecurity, change and fluctuation connected to projects, but it also provides the participants who do have time, with possibilities to take on more responsibility. The projects are constantly changing and either evolving towards a new and, hopefully, better version, or put on hold if there are no participants with enough time and knowledge to keep contributing.

Nonaka´s theory of knowledge-sharing provides concepts to discuss important issues in an organizations knowledge. By reflecting on these modes of knowledge-conversion it is possible to reveal holes in the knowledge-sharing process of an organization, and attain an image of how knowledge is created and shared, and how different technologies supports these processes. The mailing-lists used in Plone has a special place in this knowledge-sharing, in that it is both a place for participation and for reification, and makes knowledge explicit, for

other participants now, and in the future. This is unique, and made possible by digital archiving possibilities, where the interaction is automatically stored in a searchable structure. What intentionally was participation - interaction and negotiation of meaning between some participants - ends up being an important, open knowledge-resource. Faqs (frequently asked questions) is also an example of a knowledge-resource that is meant to answer exactly what participant have questions about. The ecological qualities of code-selection, can also be plausible for knowledge-sharing and documentation; one does not create more documentation then necessary, and instead, several innovative ways to communicate knowledge of the code is used. The documentation is not created by one entity but is a mixed effort by users and developers. The knowledge is created in a way that resembles the way that the software itself is developed, and its closely connected to the ideas of openness, access and collaboration that is found in FLOSS.

The main activity in the Plone community of practice is writing code. In my fieldwork, I barely scratched the surface of this activity, and my focus was more on learning how to use the software, configuration and customization, and on problem-solving. In Plone there exists a method, or best-practice, of problem-solving. This includes reading the error rapport, googling, reading a tutorial or faqs, searching the mailing-list, formulating a question for IRC or the mailing-list, engaging in conversation, posting the solution, and so on. It could be interesting to formulate a theory of FLOSS problem-solving, based on comparative studies of different FLOSS projects. To extract problem-solving patterns from a "successful" community of practice could tell us a lot about the practice itself, and on learning, from a users perspective, The findings could be applied to other software-development projects in e.g. an action study.

Generalizations based on interpretive case study are not very sound to make, thus further case studies and empirical data are needed. The methodology I have used in this thesis can be used to compare the Plone community to other FLOSS communities or to conventional software development projects, or even to other project-related communities with a different practice than software-development. The descriptive character of this thesis also makes it potentially useful as a pre-project to other research into the domain of knowledge and FLOSS.

Finerty (1997) points out that technology has a role to play in sharing knowledge, but that the emphasis needs to move from trying to package knowledge as an object, to using technology as a way of sharing experience. Several communication-channels in Plone has this function,

as it facilitates access to other participants embodied knowledge, through frequent communication and socialization. Right from the start of my fieldwork, I got access to the old-timers in the community. If knowledge is, like suggested by Kimble and Hildreth, both soft and hard, then I would say that the different aspects of knowledge are both included, and that they are are in some kind of balance in the Plone project. As we saw in episode 1 from IRC, the lack of externalized knowledge in the documentation of Ape makes it too "expensive" to learn to use, and a solution would be to document it better and thus make it more user-friendly. Gaining knowledge is a cost that might be too high, depending on what the knowledge you gain enables you to do. To keep the cost low through good documentation, can increase the possibility for other participants to get involved in the project.

In Plone there is extensive use of peer-to-peer collaboration. Lave and Wenger holds that it is typical that apprentices learn mostly in relation with other apprentices. The effectiveness of the circulation of information among peers suggests that engaging in practice may well be a condition for the effectiveness of learning. Open communication among the participants is the core of the the FLOSS development model, and peer-to-peer review of code is one of its most important features. Learning, as explained by ZPD relationships, takes place wherever this happens. In FLOSS openness and access is made possible by the freedom-enabling restrictions of the GNU GPL license, and the evolutionary selection of code in a universe of free software. The FLOSS way of learning and sharing knowledge could be used in different task-solving processes where the communication is mediated by computers. One of the limitation of this thesis is that it does not include offline learning and collaboration such as sprints and conferences, which might be very important for LPP and knowledge-sharing, and I suggest that this part of the practice should be included in further studies of the subject.

Nonaka states that 'true knowledge' – actionable understanding – comes from a gut-level commitment and belief. The Plone participants build and share knowledge on the grounds of shared emotion, feeling, mental models and experiences. The participants care about the community, and thus each other. All participants are voluntarily part of the community, they spend a considerable amount of time, and a lot of energy, on something they believe in and have some kind of ownership-feeling for. This is a very important feature of FLOSS. If 'true knowledge' comes from this type of commitment and belief, this might explain why participants mention learning as a main motivation to join FLOSS projects. In Plone access to a wide range of ongoing activity, old-timers and other members of the community, information, resources, and opportunities for participation, is open. The artifacts employed in

ongoing practice, the technology of practice, provide a good arena in which to discuss the problem of access to understanding. The open source-code, which is the key to access to understanding, is truly a unique ‘glass-box’ feature of learning in FLOSS.

Writing about knowledge and learning is inspiring and stimulates reflection on my own learning. The FLOSS development-model can in some ways resemble scientific research; developers/researchers distributed over the globe, each with specific knowledge of one part of existing software/academic discipline studies the work of others and take in consideration how their own talent can improve software/science. This method ensures in a practical way the biggest possible achievements and assembles best the systematic character of science, and this is the way self-coordination through mutual adjustments works. Writing a thesis is not only a work of imparting gained knowledge; we don’t write finished thoughts, but engage in the manifestation of thoughts in a concrete form through reflection. Writing this theses has been a creative activity that has influenced both my understanding of the topic in focus, and my understanding of the academic practice.

REFERENCES

Alavi, M., Leidner, D. E. (1997). "Knowledge management systems: emerging views and practices from the field." INSEAD R&D working papers; 97/97/TM: 31 s.

Aspeli, M. (2005). Plone - a model of a mature open source project. MSc Analysis Design and Management of Information Systems. London, London School of Economics. Master.

Baldwin, C. Y. and K. B. Clark (2000). Design Rules, vol. 1. The Power of Modularity, MIT Press, Cambridge, MA.

Bateson, G. (1972). Steps to an ecology of mind. New York, Ballantine Books.

Bezroukov, N. (1999). Open Source Software Development as a special type of Academic Resource (Critique of Vulgar Raymondism). Firstmonday, http://firstmonday.org/issues/issue4_10/bezroukov/.

Bezroukov, N. (1999). A second look at the Cathedral and the Bazaar. FirstMonday. First Monday, http://firstmonday.org/issues/issue4_12/bezroukov/.

Brown, J. S. (1999). Tape transcription of the talk "Learning, Working & Playing in the Digital Age". 1999 Conference on Higher Education, http://serendip.brynmawr.edu/sci_edu/seelybrown/.

Brown, J. S. and P. Duguid (2000). The social life of information. Boston, Mass., Harvard Business School Press.

Brown, J. S. and D. P. (1991). "Organizational learning and communities of practice." Organization Science, 2, no. 1, reprinted in M.D. Cohen & L.S. Sproull (eds.), 1996.

Chris Mann, F. S. (2000). Internet Communication and Qualitative Research A Handbook for Researching Online. London, Sage.

Ciborra, C. U. and K. Nygaard (2002). The labyrinths of information: challenging the wisdom of systems. Oxford, Oxford University Press.

Clark, H. H. and C. R. Marshall (1981). Definile reference and mutual knowledge. Cambridge, Cambridge University Press.

Conklin, E. J. (1996). Designing organizational memory: preserving intellectual assets in a knowledge economy, Glebe Creek, MD, CogNexus Institute.

Crowston, K. and J. Howison (2004). The Social Structure of Free and Open Source Software Development. First Monday, http://www.firstmonday.org/issues/issue10_2/crowston/. 10.

Cummings, J., B. Butler, et al. (2002). "The Quality of Online Social Relationships." Communications of the ACM, Volume 45, Issue 7, ACM Press, New York, NY, USA.

Davenport, T. and L. Prusak (1998). Working knowledge. How organizations manage what they know. Cambridge, MA, Harvard Business School Press.

Dehlin, E. (2004). Reflection note Nonaka & Takeuchi. Trondheim, Høgskolen i Sør-Trøndelag, Avdeling Trondheim økonomiske høgskole.

Economist, T. (Mar 16th 2006). "Open, but not as usual." The Economist.

Europyhon (2005). Various talks and presentations. Chalmers University, Gothenburg, Sweden.

Finerty, T. (1997). "Integrating learning and knowledge infrastructure." Journal of Knowledge Management(1): 98-104.

Franck, E. and C. Jungwirth (2002). "Working paper. Reconciling Investors and Donators - The Governance Structure of Open Source." Lehrstuhl für Unternehmensführung und -politik Universität Zürich(No. 8).

Giuri, P., M. Ploner, et al. (2004). "Skills and Openness of OSS Projects: Implications for Performance." Paper provided by Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies, Pisa, Italy in its series LEM Papers Series with number 2004/19.

Hammersley, M. and P. Atkinson (1995). Ethnography: principles in practice. London, Routledge.

Hargadon, A. B. (1998). "Firms as knowledge brokers: lessons in pursuing continuous innovation." California Management Review 40: 209-227.

Hemetsberger, A. and C. Reinhardt (2004). Sharing and Creating Knowledge in Open-Source Communities The case of KDE. The Fifth European Conference on Organizational Knowledge, Learning, and Capabilities. Innsbruck, Austria, 2004.

Hildreth, P. M. and C. Kimble (2002). "The duality of knowledge." Information Research, Vol. 8 No 1.

Hine, C. (2000). Virtual ethnography. London; Thousand Oaks, Calif., SAGE.

Holt, N. L. (2003). "Representation, legitimation, and autoethnography: An autoethnographic writing story." International Journal of Qualitative Methods, 2(1). Article 2 http://www.uofaweb.ualberta.ca/iiqm/.

Infomedia (2004). Demokrati og åpne kildekoder, Application for project-funding for a study concerning deliberative democracy and system devlopment in open-source software, and also ideology, economt and user interests in political work for open source-code. Personal mail, Department of media and information studies, Faculty of social science, University of Bergen, Norway.

Kimble, C., P. M. Hildreth, et al. (2004). Knowledge networks: innovation through communities of practice. Hershey, Pa., Idea Group Publishing.

Kohanski, D. (1998). Moths in the Machine, St. Martin's Press, New York.

Kraut, R. E. and L. A. Streeter (1995). "Coordination in Software Development." Communications of the ACM ACM Press, New York, NY, USA Volume 38(3).

Kvale, S. (1996). Inter Views: An introduction to qualitative research interviewing. Thousand Oaks, CA, SAGE.

Lakhani, K. R. and E. von Hippel (2003). "How open source software works: "free" user-to-user assistance." Research Policy(32): 923–943.

Lanzara, G. F. (1999). "Between transient constructs and persistent structures: designing systems in action." Journal of Strategic Information Systems 8: 331-349.

Lanzara, G. F. and M. Morner (2003). The Knowledge Ecology of Open-Source Software Projects. 19th EGOS Colloquium. Copenhagen.

Latour, B. (1992). "Technology is society made durable." Sociology of monsters: Essays on power, technology and domination, in: Law, J. (ed., 1992: 103-131.

Lave, J. and S. Chaiklin (1993). Understanding practice: perspectives on activity and context. Cambridge, Cambridge University Press.

Lave, J. and E. Wenger (1991). Situated learning: legitimate peripheral participation. Cambridge, Cambridge University Press.

Leonard, D. and S. Sensiper (1998). "The role of tacit knowledge in group innovation." California Management Review: 40 (3) 112-132.

Lesser, E. L. and J. Storck (2001). "Communities of practice and organizational performance." IBM SYSTEMS JOURNAL 40 (nr 4).

Limi, A. (2005). Talk: Plone - Growing up. Europython 2005. Chalmers University, Gothenburg, Sweden.

Lin, Y. (2004). Contextualizing knowledge-making in Linux user groups. Firstmonday, http://firstmonday.org/issues/issue9_11/lin/index.html.

Lin, Y. (2005). "The future of sociology of FLOSS." FirstMonday. http://firstmonday.org/issues/special10_10/lin/index.html.

MacDonald, J., W. Atkin, et al. (2003). Let's get more positive about the term 'lurker', CPsquare Foundations of Communities of Practice Workshop, http://www.cpsquare.org/edu/foundations/index.htm.

McVetta, J. (2005). Personal mail.

Merriam, S. B. and R. S. Caffarella (1991). "Learning in adulthood: a comprehensive guide." The Jossey-Bass higher and adult education series, Jossey-Bass: xx, 376 s.

Monteiro, E., T. Østerlie, et al. (2004). Keeping it going: The Everyday Practices of Open Source Software, www.idi.ntnu.no/~thomasos/paper/keeping_it_going.pdf.

Nonaka, I. (1994). "A Dynamic Theory of organizational Knowledge Creation." Organization science 5(1): 14-37.

Nonaka, I. and T. Nishiguchi (2001). Knowledge emergence: social, technical, and evolutionary dimensions of knowledge creation. Oxford, Oxford University Press.

Nonaka, I. and H. Takeuchi (1995). The knowledge-creating company: how Japanese companies create the dynamics of innovation. New York, Oxford University Press.

Nonaka, I., H. Takeuchi, et al. (2004). Hitotsubashi on knowledge management. Singapore, Wiley.

Orlikowski, W. (2002). "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing." Organization Science 13(3).

Orthmann, C. (2000). "Analysing the Communication in Chat Rooms—Problems of Data Collection." Qualitative Social Research 1(3): 6.

Oxford (2006). American Dictionaries, Computer program.

Paccagnella, L. (1997). "Getting the Seats of YourPants Dirty: Strategies for Ethnographic Research on Virtual Communities." Journal of Computer-Mediated Communication(3).

Patton, M. Q. (1990). Qualitative evaluation and research methods (2nd ed.). Newbury Park, CA, Sage.

Pliskin, N., I. Balaila, et al. (1991). "The knowledge contribution of engineers to software development: a case study." IEEE Transactions on Engineering Management 38 (4) 38.

Plone (2005/2006). The homepage of the Plone project, http://plone.org.

Polanyi, M. (1958). Personal knowledge. London, Routledge and Kegan Paul.

Polanyi, M. (1967). The tacit dimension. London, Routledge & Kegan Paul.

Polanyi, M. (2000). Den tause dimensjonen: en innføring i taus kunnskap. Oslo, Spartacus.

Polanyi, M. and M. Grene (1969). Knowing and being: essays. [Chicago], University of Chicago Press.

QuoteDB (2006). Interactive Database of Famous Quotations. http://www.quotedb.com/.

Raymond, E. S. (2001). The cathedral and the bazaar: musings on linux and open source by an accidental revolutionary. Cambridge, Mass., O'Reilly.

Rintel, E. S., J. Mulholland, et al. (2000). "First Things First: Internet Relay Chat Openings." Journal of Computer Mediated Communication 6(3).

Ritz, R. Programming Plone - The MySite Tutorial, http://www.neuroinf.de/LabTools/MySite.

Silvermann, D. (1993). Interpreting qualitative data. London, SAGE Publications.

Sourceforge (2006). Home page of Sourceforge http://sourceforge.net.

Spencer, B. (1997). Organizational Knowledge Creation, summary of presentation given by Ikujiro Nonaka. Knowledge Advantage Conference.

Stallmann, R. M. and J. Gay (2002). Free Software, Free Society: Selected Esseys of Richard M. Stallman. Boston, MA, USA, GNU Press.

Stoll, B. L. (2005). Fun and Software Development, Free / Open Source Research Community, http://opensource.mit.edu/online_papers.php.

Suchman, L. A. (1987). Plans and situated actions: the problem of human-machine communication. Cambridge, Cambridge University Press.

Sveiby, K. E. (1997). Tacit Knowledge, http://www.sveiby.com/articles/Polanyi.html.

Tuomi, I. (2000). "Internet, Innovation and Open Source: Actors in the Network." First Monday, http://www.firstmonday.org/issues/issue6_1/tuomi/.

Tuomi, I. (2005). "The Future of Open Source." Wynants, M. & J. Cornelis (eds.) How Open is the Future? VUB Brussels University Press, pp. 429-59.

von Hippel, E., von Krogh, G (2003). " Open source software and the private-collective innovation model: issues for organization science." Organization Science 14 (2) 14.

von Krogh, G., S. Spaeth, et al. (2003). "Community, joining, and specialization in opensource software innovation: a case study." Research Policy 32: 1217–1241.

von Krogh, G. and E. von Hippel (2003). "Special issue on open source." Research Policy 32: 1149–1157.

Vygotsky, L. S. (1978). Mind in society: The development of higher psychological processes. Cambridge, MA, Harvard University Press.

Waterson, P. E., C. W. Clegg, et al. (1997). "The dynamics of work organization, knowledge and technology during software development." International Journal of Human-Computer Studies 46 (1).

Wenger, E. (1998). Communities of practice. Learning, meaning and identity. Cambridge, Cambridge University Press.

Wenger, E. (2000). En social teori om læring. I: Illeris, K.(red): Tekster om læring. Roskilde Universitetsforlag.

Wenger, E., R. McDermott, et al. (2002). Cultivating communities of practice: a guide to managing knowledge. Boston, Harvard Business School Press.

Wertsch, J. V. (1985). Culture, communication, and cognition: Vygotskian perspectives. Cambridge, Cambridge University Press.

Wikipedia.org (2005/2006). The Free Encyclopedia, http://wikipedia.org.

Wynn, E. H. (1979). Office Conversation as an Information Medium, University of California, Berkeley (dissertation).

Yamauchi, Y., M. Yokozawa, et al. (2000). Collaboration with Lean Media: How Open-Soruce Software Suceeds, Proceedings of ACM CSCW.

Zuboff, S. (1988). In the age of the smart machine: the future of work and power. New York, Basic Books.

Østerlie, T. (2004). In the network: Distributed control in Gentoo Linux, http://www.idi.ntnu.no/~thomasos/.

APPENDIX

Appendix 1: Chat-log transcript, episode 1 - Storage

Apr 15 14:58:17 <Querk> guys, I have a new AT storage, and I need some place to put it
Apr 15 14:58:21 <Querk> can i put it in AT svn?
Apr 15 14:59:00 <cyhyb> Querk: there should probably be an area for storages like there is for fields/widdgets
Apr 15 14:59:07 <Querk> cyhyb: I can make one
Apr 15 14:59:13 <cyhyb> Querk: do that :)
Apr 15 14:59:19 <Querk> actually
Apr 15 14:59:25 <Querk> i can't... how do I make a new directory server side?
Apr 15 14:59:39 <cyhyb> :)
Apr 15 14:59:45 <netaual> Querk: create a local dir and add it to svn
Apr 15 14:59:47 <cyhyb> svn mkdir
*Apr 15 14:59:58 * Querk likes cyhyb's suggestion*
Apr 15 15:00:00 <cyhyb> svn mkdir full path
Apr 15 15:00:00 <Querk> MoreStorages?
Apr 15 15:00:12 <Querk> also... there is already a ReviewStorage in the root of the repo
*Apr 15 15:00:27 * cyhyb really likes the -more-something adressing scheme*
Apr 15 15:00:30 <Querk> shall I just make my path, then mail the list?
Apr 15 15:00:42 <Querk> cyhyb: ?
Apr 15 15:00:44 <cyhyb> (probably just because i came up with it)
Apr 15 15:00:47 <Querk> lol
Apr 15 15:00:56 <cyhyb> Querk: yeah, do that.
Apr 15 15:01:11 <cyhyb> and suggest to whoever is doing ReviewStorage to move it
Apr 15 15:01:37 <Querk> cyhyb: what's the correct way of importing a new product?
Apr 15 15:01:46 <Querk> i guess I want to have branches, tags, trunk directories?
Apr 15 15:01:56 <Querk> do I just mkdir each of those?
Apr 15 15:02:11 <cyhyb> yes. mkdir them all
*Apr 15 15:02:32 <cyhyb> then check out trunk. put your stuff in it, and svn add **
Apr 15 15:02:44 <cyhyb> that is the way i like it, at least
Apr 15 15:02:46 <cyhyb> :)
Apr 15 15:02:55 <cyhyb> gives you total predictability
Apr 15 15:03:17 <Querk> cyhyb: ok
Apr 15 15:03:56 <jacqulin> Querk: i am starting SQLWindowStorage v2
Apr 15 15:04:03 <jacqulin> better approach this time.
Apr 15 15:04:11 <Querk> jacqulin: what's the difference?
Apr 15 15:04:21 <physodes> what is sqlwindowstorage ?
Apr 15 15:04:23 <Querk> jacqulin: in a minute, you can check out my approach; it's similar, but simpler (simpler use case)
Apr 15 15:04:51 <jacqulin> Querk: ok.
Apr 15 15:05:03 <jacqulin> physodes: AT with data from an existing SQL DB
Apr 15 15:05:13 <jacqulin> Querk: http://arch.jacqulin.net/cgi-bin/archzoom.cgi/jacqulin@arch.jacqulin.net--2005-zope/AT-SQLWindowStorage?expand -- but i need to get archzoom fixed first
Apr 15 15:05:23 <physodes> jacqulin: does it rely on Ape or anything like that?
Apr 15 15:05:29 <jacqulin> physodes: no.
Apr 15 15:05:56 <jacqulin> physodes: you can check out the arch archive if you want.

*Apr 15 15:06:01 * physodes still thinks the future of using SQL with Plone should be with Ape*
Apr 15 15:06:15 <jacqulin> Ape seems too complex
Apr 15 15:06:32 <jacqulin> and it cannot really make my existing SQL records appear as objects
Apr 15 15:06:35 <jacqulin> or can it?
Apr 15 15:06:38 <Querk>physodes: it may well be, but i don't know about APE, so poo on that
Apr 15 15:06:39 <physodes> jacqulin: then make it simpler
*Apr 15 15:06:45 <Querk>i need *simple* and *yesterday**
Apr 15 15:06:46 <physodes> sure it can
Apr 15 15:06:54 <jacqulin> i am a bottom-up guy
Apr 15 15:06:57 <jacqulin> physodes: you have an example?
Apr 15 15:07:23 <physodes> i am a make-existing-stuff-work-rather-than-build-new-stuff guy
*Apr 15 15:07:25 * Querk had a brieft stint with LDAP before doing this storage and still hurts*
Apr 15 15:07:43 <Querk>physodes: me too. except when that old stuff is poorly documented or overly complex for my needs :)
*Apr 15 15:07:49 <Querk>and I *do* build on SQLStorage :)*
Apr 15 15:08:11 <jacqulin> physodes: http://hathawaymix.org/Software/Ape says that Ape can store python objects in an SQL database. it does not say it can instantiate records in a database as "virtual objects"
Apr 15 15:08:13 <physodes> Querk: i extended Ape a while back to work on MS SQL Server and Sybase... it ain't overly complex
Apr 15 15:08:31 <jacqulin> physodes: how does it store an object in the DB? do you have an example?
Apr 15 15:08:53 <physodes> jacqulin: i had ZClass-based objects living in a sql database quite sensibly a while back
*Apr 15 15:09:08 * Querk shudders at mention of ZClasses*
Apr 15 15:09:12 <jacqulin> yeah
Apr 15 15:09:26 <jacqulin> physodes: AT-SQLStorage can also store in an SQL database, but it's not what I want.
Apr 15 15:09:28 <physodes> jacqulin: the last ape stuff i did was about 1.5 years ago and at my last job where i don't have access to the code (nor would i be able to display it anyhow cuz it was proprietary)
Apr 15 15:09:56 <physodes> i only mentioned zclasses to show that even the worse type of objects can live in a rdbms
Apr 15 15:10:00 <Querk>:p
*Apr 15 15:10:10 * Querk shudders at yet another mention of the Z word*
*Apr 15 15:10:27 * physodes dislikes that there's a dozen different products with the expression 'SQL' in its description all aiming to do the same sort of thing*
Apr 15 15:10:38 <Querk>jacqulin: you may want to see my code. i think it's simpler than what you're doing, but it works for us :)
Apr 15 15:10:54 <jacqulin> Querk: sure.
Apr 15 15:10:56 <Querk>physodes: well, SharedSQLStorage is just SQLStorage overriden to behave a little differently
Apr 15 15:11:44 <physodes> Querk: why not extend SQLStorage to be able to behave two different ways?
Apr 15 15:12:00 <jacqulin> physodes: i will.
Apr 15 15:12:07 <Querk>physodes: different use cases.
*Apr 15 15:12:08 * cyhyb likes products to do as little as possible*
Apr 15 15:12:29 <Querk>SQLStorage = auto-create tables, make sure they correspond to chema, store values there
Apr 15 15:13:00 <physodes> cyhyb: i do too... but i dislike having multiple products that do kind of the same thing even more

Apr 15 15:13:12 <Querk>SharedSQLStorage = you have a table (in our case with memberdata) you want to share between two plone instances, and you don't want to share every single field, nor can you guarantee that SQL table column names == AT field names
Apr 15 15:13:21 <cyhyb> depends on how same it is, but in principle i agree w you
Apr 15 15:13:47 <jacqulin> Querk: with write support?
Apr 15 15:13:48 <Querk>physodes: not when one is a dependency of another. Want SQLStorage - get that. Want the SharedSQLStorage extension - get that too
Apr 15 15:13:52 <Querk>jacqulin: yep
Apr 15 15:14:06 <jacqulin> Querk: and support for n:1 and n:m relations?
Apr 15 15:14:08 <Querk>it is simple, however - it doesn't do things like joins
Apr 15 15:14:13 <Querk>no
Apr 15 15:14:15 <Querk>nor references
Apr 15 15:14:19 <jacqulin> those are the difficult ones...
Apr 15 15:14:25 <jacqulin> the rest i hacked up in a day. :)
Apr 15 15:14:27 <Querk>yep. also they are things we don't need :)
Apr 15 15:14:30 <Querk>yes, me too
Apr 15 15:14:31 <physodes> bah, use ape and write custom marshallers
Apr 15 15:14:35 <jacqulin> anyway, zope2.7 won't even start with APE
Apr 15 15:14:38 <jacqulin> Error: unknown type name: 'ape-db'
*Apr 15 15:15:19 * Querk will use APE when ti's mainstream enough that he doesn't have to hunt for documentation and examples and venture into a minefield of missed deadlines because something that is really simple becomes too complex with a big and mysterious framework*
Apr 15 15:15:29 <jacqulin> forgot %import
Apr 15 15:15:32 <physodes> i think Ape could be refined quite a bit to be made simpler... but i think its an amazing framework for connecting plone to an rdbms
Apr 15 15:15:34 <Querk>I'm sure it's great. It's just not accessible enough right now to me
Apr 15 15:15:58 <Querk>physodes: then get some nice examples + simplfying layers together and make the world a better place :)
Apr 15 15:16:01 <physodes> Querk: well then, rather than write your own sql thing... why not work on doing docs and making ape more mainstream?
Apr 15 15:16:03 <Querk>it's something we need to get better at
Apr 15 15:16:14 <Querk>physodes: because deadline is today
Apr 15 15:16:26 <Querk>i was going to do just that with LDAP (this is about memberdata, LDAP would've been better)
Apr 15 15:16:41 <Querk>and it was just so incredibly complex + undocumented that I would've needed a week to do what now took me a day
Apr 15 15:16:43 <jacqulin> LDAP sucks
Apr 15 15:16:45 <Querk>time is finite, unfortunately
*Apr 15 15:16:52 * Querk is inclined to agree*
Apr 15 15:16:55 <physodes> writing an ldap module for Ape shouldn't be too difficult
*Apr 15 15:17:17 <Querk>physodes: sigh... same problem - requires understanding *two* alient technologies*
Apr 15 15:17:25 <Querk>neither of which are readily acessible to me
Apr 15 15:17:45 <Querk>in the real world, people tend to minimise risk by doing things they know how long will take, even if another solution may be "better" (at any rate, our code works now and we're happy)
Apr 15 15:19:33 <physodes> Querk: in that case, you should move over to perl and postnuke... development will definately move faster
Apr 15 15:19:42 <jacqulin> urks

Apr 15 15:19:57 <Querk>physodes: don't be silly.
*Apr 15 15:21:16 * physodes is just very frustrating watching people overlook ape because it seems too complicated or because SQL databases are too slow... if any of that is truly the problem, then fix it rather than create some other niche product*
Apr 15 15:21:52 <jacqulin> physodes: trying it now.
Apr 15 15:23:29 <Querk>physodes: you asked me why I don't spend my paid for, limited time fixing your "I think this is so great" technology. The reason is that it's not accessible to me in the time I have, and I won't risk total project failure (and thus not getting paid) if I'm uncomfortable about the technology. I don't know what I'm getting myself into.
Apr 15 15:24:04 <Querk>physodes: if you are interested in getting more people onto APE, you have to campaign for it, and not at least provide working examples and good documentation.
Apr 15 15:24:05 <sashav> Querk and how did ldap stuff go? :)
Apr 15 15:24:21 <Querk>dropped it in favour of SQL, way too complex and things breaking I had no idea why
Apr 15 15:24:35 <Querk>see above comment to physodes - would be nice, but d.e.a.d.l.i.n.e.
Apr 15 15:24:40 <physodes> Querk: that is a very intelligent attitude to have, and i don't disagree with you on that point... you probably made the right decision for this project
Apr 15 15:24:55 <Querk>physodes: but you're right, it is unfortunate
Apr 15 15:24:59 <sashav> Querk, cmfmember + sql?
Apr 15 15:25:21 <Querk>sashav: yes, wrote a custom storage which meets our needs
Apr 15 15:25:28 <jacqulin> physodes: can you show me an example of a custom type mapped to APE?
Apr 15 15:25:34 <jacqulin> e.g. PloneArticle?
Apr 15 15:25:38 <sashav> can you show it to me?
Apr 15 15:25:40 <jacqulin> or better yet: an Arechtype?
*Apr 15 15:25:52 <Querk>physodes: this is why the people who *do* take the plunge and develop APE need to spend the time to make docs, examples, easy resources to convince those on the fence that it can work for them. Then they'll get a lot more help.*
Apr 15 15:26:15 <Querk>sashav: storage is in AT SVN, under MoreStorages
Apr 15 15:26:27 <physodes> Querk: you strike me as the kind of guy who should be taking plunges and not hanging out on fences ;)
Apr 15 15:26:34 <Querk>hehe :)
Apr 15 15:26:36 <Querk>when time permits
Apr 15 15:26:51 <physodes> jacqulin: unfortunately i haven't touched ape in over a year, but in the next few weeks i will be diving back in
Apr 15 15:26:51 <sashav> ok, any speciall configuration except storage=YourStorage in schema for the member?
Apr 15 15:28:56 <jacqulin> Querk: where is the code?
Apr 15 15:28:59 <jacqulin> 1.4?
*Apr 15 15:29:14 * jacqulin hates SVN*
Apr 15 15:29:31 <Querk>jacqulin: svn co http://svn.plone.org/archetypes/MoreStorages/SharedSQLStorage/trunk
Apr 15 15:30:17 <Querk>sashav: no, no special configuration, except a) the table must exist in the db already (by design) and b) you must give the table name (and optionally column name if you don't use the same as the AT field name) in the constructor of the storage
*Apr 15 15:30:33 <Querk>sashav: the biggest thing it's missing atm is cacheing of result sets; it makes a *lot* of trips to the db atm*
Apr 15 15:31:20 <sashav> Querk, sounds like sqlstorage :)

Apr 15 15:31:43 <Querk>sashav: it's an extension thereof, except SQLStorage stores by UID and auto-creates tables and hardlinks column names to AT fifeld names
Apr 15 15:31:54 <zorb|nearby> SQLStorage is also slow as hell
Apr 15 15:32:01 <Querk>i.e. it doesn't allow you to use existing tables, nor share the same table between two things
Apr 15 15:32:03 <jacqulin> Querk: i am working on that.
Apr 15 15:32:09 <Querk>zorb|nearby: yes, because it doesn't cache.
Apr 15 15:32:11 <jacqulin> zorb|nearby: but it has caching...
Apr 15 15:32:15 <Querk>it should, and it wouldn't be too hard
Apr 15 15:32:16 <jacqulin> Querk: it does.
Apr 15 15:32:20 <Querk>jacqulin: are you sure?
Apr 15 15:32:23 <Querk>where?
Apr 15 15:32:27 <jacqulin> yeah, spent all night on the code.
Apr 15 15:32:29 <jacqulin> hold on.
Apr 15 15:32:42 <jacqulin> SQLMethod
Apr 15 15:33:11 <Querk> ah, right
Apr 15 15:33:19 <Querk>that's lower level, didn't think of that
Apr 15 15:33:24 <Querk>any idea how it invalidates?
Apr 15 15:33:26 <jacqulin> i am using that now.
Apr 15 15:33:34 <jacqulin> Querk: same as ZSQL -- time based.
Apr 15 15:33:51 <Querk>right
Apr 15 15:33:56 <jacqulin> and it has to be configured, it's at 0 by default.
Apr 15 15:34:04 <jacqulin> zorb|nearby: that's why it's slow... it can get faster.
Apr 15 15:34:27 <jacqulin> Querk: what i want to do is create a framework which actually generates the schema for you.
Apr 15 15:34:35 <jacqulin> because my major problem is type maps.
Apr 15 15:35:36 <jacqulin> physodes, Querk: http://plone.org/events/sprints/castlesprint/wiki/ApeSupport
Apr 15 15:57:28 <jacqulin> physodes, Querk: I cannot even get DBTab to work as it seems to be simply broken.
Apr 15 15:57:47 <Querk>jacqulin: is this APE? don't ask me :)
Apr 15 15:57:52 <physodes> last time i used ape 1.0 i thought it didn't require dbtab anymore
Apr 15 15:57:57 <jacqulin> Querk: just fyi
Apr 15 15:58:02 <Querk>thanks
Apr 15 15:58:06 <jacqulin> physodes: ah, okay.
Apr 15 15:58:16 <jacqulin> physodes: so how do i mount/create the mount point?
Apr 15 15:58:26 <physodes> jacqulin: in the ape.conf file or something like that
Apr 15 15:58:35 <jacqulin> zope.conf?
Apr 15 15:58:42 <physodes> hm, i don't recall atm
Apr 15 15:58:52 <jacqulin> see, so htf am i supposed to get anywhere???
Apr 15 15:58:58 <jacqulin> there is no documentation.
Apr 15 15:59:27 <__gotcha> jacqulin, read the code ;-0
Apr 15 15:59:33 <davconvent> jacqulin, couldn't you find doc in the zope book or in the doc forlder of zope source ?
Apr 15 15:59:37 <__gotcha> I meant ;-)
*Apr 15 15:59:40 * jacqulin whacks __gotcha over the head with a large trout (non-violently, of course) [http://www.nwrc.usgs.gov/world/images/trout.jpg].*

Apr 15 16:14:25 <jacqulin> physodes: ping?
Apr 15 16:14:43 <physodes> yep?
Apr 15 16:14:53 <jacqulin> so now i have ape working but i can't figure it out at all.
Apr 15 16:14:58 <jacqulin> i added a Link object
Apr 15 16:15:06 <jacqulin> and it turns up in atlink_properties with an ID
Apr 15 16:15:16 <jacqulin> but i can't tell where the data are stored...
Apr 15 16:15:25 <jacqulin> http://rafb.net/paste/results/MXfREP28.txt
*Apr 15 16:15:29 <jacqulin> list of relations ^^^*
Apr 15 16:15:49 <jacqulin> oh yeah, it's a pickle in remainder...
Apr 15 16:15:54 <jacqulin> let's see if i can define a mapping.
Apr 15 16:16:17 <jacqulin> i find it quite annoying that the mount point has to be a Zope Folder, i.e. not a PortalFolder
Apr 15 16:16:24 <jacqulin> so plone can't really use it.
Apr 15 16:16:30 <jacqulin> other than through subfolders.
Apr 15 16:20:47 <physodes> jacqulin: btw, what i remember is that the mount point can be any type of folderish zope object... but there used to have to be some trickery to get it to work
Apr 15 16:21:03 <jacqulin> yeah well.
*Apr 15 16:21:59 <Vinfan> hum, a plonefolder *is* a zope folder, deep down inside...*
Apr 15 16:22:52 <jacqulin> http://mail.zope.org/pipermail/zope/2005-February/156574.html
Apr 15 16:22:58 <jacqulin> Vinfan: but not vice versa
Apr 15 16:24:57 <jacqulin> next project. this one will use my SQLWindowStorage.
Apr 15 16:25:34 <jacqulin> I can't seem to figure out how to make it e.g. write from a query and store to a table, or how to make it read different items for get and getRaw
Apr 15 16:27:12 <jacqulin> Querk: what kind of licence is that with SharedSQLStorage?
Apr 15 16:27:28 <Querk> BSD, same as AT
Apr 15 16:27:32 <Querk> see LICENSE.txt
Apr 15 16:27:39 <jacqulin> Querk: now i see. you could have given me credit. :)
Apr 15 16:27:44 <jacqulin> Querk: (don't worry... :>)
Apr 15 16:27:46 <Querk> jacqulin: i didn't use your code at all
Apr 15 16:27:54 <Querk> i used SQLStorage code quite a lot
Apr 15 16:28:00 <Querk> and should've given credit there
Apr 15 16:28:09 <Querk> but the fact that you rely on custom field types meant it was unsuitable for us
Apr 15 16:28:22 <jacqulin> ah, okay.
Apr 15 16:28:35 <jacqulin> i am going to drop this requirement now. it was really just convenience.
Apr 15 16:28:50 <Querk> "convenience"? :)
*Apr 15 16:28:56 * Querk is in debug hell*
Apr 15 16:29:08 <jacqulin> Querk: convenience as in not having to set sql_index=1 for every field.
Apr 15 16:29:19 <jacqulin> Querk: btw: Archetypes.ApeSupport :)
Apr 15 16:29:25 <Querk> yeah, i know
Apr 15 16:29:27 <jacqulin> it is moving in the right direction. :)
Apr 15 16:56:51 <Querk> well, I have a user in acl_users, but no memberdata object in portal_memberdata
Apr 15 16:56:57 <Querk> when I first log in, it creates such an object
Apr 15 16:57:03 <Querk> i need to know where that code is
Apr 15 16:57:38 <Querk> because it ends up resetting some fields to AT default values, when in fact I dont' want it to do that
Apr 15 16:58:51 <Vinfan> Querk: probably in wrapUser
Apr 15 16:59:18 <Querk> Vinfan: that looks promising

Apr 15 16:59:21 <Querk> i think you're right
Apr 15 16:59:31 <Querk> however, it's not giving me the answer i want
Apr 15 16:59:35 <Querk> i guess the problem is in AT
Apr 15 16:59:46 <Vinfan> or your schema
Apr 15 16:59:55 <Querk> - when a an object is created, it goes through and conveniently sets everything to default values
Apr 15 17:00:20 <Querk> Vinfan: what's happening is that I have two objects, in two different plone instances, talking to the same DB backend for storage
Apr 15 17:00:40 <Querk> which works fine, except when the object is first craeted, it resets all the fields to default values
Apr 15 17:00:51 <Querk> even though there's already something there in the table
Apr 15 17:01:04 <Vinfan> well, how would it know?
Apr 15 17:02:51 <Querk> Vinfan: i basically need to know if an object is in the process of being created or not
Apr 15 17:02:56 <Querk> age-old problem
Apr 15 17:08:57 <Vinfan> Querk: may it work to use a default_method in the schmema that does the right thing (or nothing)?
Apr 15 17:09:55 <Querk> Vinfan: possibly; except I'm trying to solve this at the storage level and having to write a default_method for each one would be a major pain

Appendix 2: Chat-log transcript, episode 2 - Rockstars and UML design

Oct 29 02:17:55 caneose *can zorb, epithet or someone help me with a design question?*
Oct 29 02:18:22 epithet *the way it works here, caneose, is that you ask your question, and then we decide ;)*
Oct 29 02:18:44 caneose *my main class is "Recipe", which is a cookbook recipe. I made another class called "category", and created a (* to 1..*) relationship*
Oct 29 02:19:05 caneose *that is, a instance of a Recipe can belong to many food categories, at least one.*
Oct 29 02:19:16 caneose *and a category can be assigned to 0 or more Recipes*
Oct 29 02:19:22 byte2b *categories are sooooo web 1.0, be a buzzword pro and go web2.0 with tags!*
Oct 29 02:19:32 caneose *but now I'm wondering if this is too lame, yeah*
Oct 29 02:19:34 Tim *maybe simpler as keywords unless category has its own behavior?*
Oct 29 02:19:41 caneose *maybe I want so sort of dynamic vocabulary thing?*
Oct 29 02:19:52 epithet *exactly; do you really need to have a separate object type for categories?*
Oct 29 02:20:03 caneose *'category' is really nothing more than a set of dynamic vocabulary, or keywords, yeah. no special behavior.*
Oct 29 02:20:13 epithet *then it's overengineering to have that as a class*
Oct 29 02:20:21 caneose *categories might be "desserts", "entrees", "pastas", etc.*
Oct 29 02:20:23 zorb *caneose: ArchAddOn has a nice way to do that, look at how PHC does it*
Oct 29 02:20:26 caneose *great, I thought so.*
Oct 29 02:20:34 zorb *just as a delimited field*
Oct 29 02:20:39 caneose *what's PHC?*
Oct 29 02:20:44 zorb *PloneHelpCenter*
Oct 29 02:20:47 zorb *product :)*
Oct 29 02:20:48 caneose *ah*
Oct 29 02:20:52 caneose *whoa*
Oct 29 02:21:15 zorb *it would look like this:*
Oct 29 02:21:29 epithet *the common design pattern here is often: you have a folder, "recipes". you put a property on it (or, if AT-based, have in its schema) the list of categories*
Oct 29 02:21:39 zorb *pasta | Pasta | Italian dish made of whatever-you-call-that-in-English*
Oct 29 02:21:41 epithet *and then recipes just have a multiselection field for which categories they fall under*
Oct 29 02:22:00 byte2b *caneose: man you lucked out tonight ... got zorb, epithet, and Tim all helping you out... 3 plone rockstars!*
Oct 29 02:22:04 epithet *the PHC way give you a simple way to do that, assuming the "recipes" folder is ATBased; there's a very, very simple data-grid-like widget in it.*
Oct 29 02:22:11 caneose *byte2b: :-)*
Oct 29 02:22:13 zorb *caneose is a rock star in his area ;)*
Oct 29 02:22:14 epithet *byte2b: stop being so modest. you're a plone rockstar.*
Oct 29 02:22:24 zorb *byte2b: ever heard of SVN? ;)*
Oct 29 02:22:30 byte2b *epithet: nah, i'm just a choir boy ;)*
Oct 29 02:22:33 epithet *Tim is the rock star. everyone else is merely but a bass player with ambitions*
Oct 29 02:22:34 zorb *he wrote it (not alone, but... ;))*
Oct 29 02:22:41 byte2b *zorb: hehe, yep, i know where caneose is from :)*
Oct 29 02:22:51 caneose *[not from MIT!]*
Oct 29 02:22:55 byte2b *lol*
*Oct 29 02:23:06 ** *Tim blushes*
*Oct 29 02:23:12 ** *zorb wants to involve caneose in the SVN-backend-for-CMFEditions*

Oct 29 02:23:15 zorb *;)*
*Oct 29 02:23:21 ** *caneose tries to parse epithet's pattern description... hold on*
Oct 29 02:23:32 zorb *I'm sure caneose would love to have his Plone content backed by SVN ;)*
Oct 29 02:23:35 byte2b *whoa, CMFEditions-on-svn would be nice!*
Oct 29 02:23:38 caneose *woo!*
Oct 29 02:23:51 zorb *frutiella and WhiPer had a working prototype at one point*
*Oct 29 02:23:57 ** *amaryll might like to involve caneose in svn tracker and repo control integration for plone :)*
Oct 29 02:24:01 zorb *not sure where it went*
Oct 29 02:24:10 caneose *svn tracker, eh?*
Oct 29 02:24:15 zorb *amaryll: I'm sure caneose has enough SVN duties ;)*
Oct 29 02:24:23 amaryll *like trac.*
Oct 29 02:24:29 caneose *well, not anymore, purely voluntary svn work now*
Oct 29 02:24:36 caneose *and trac is mighty impressive, yeah*
Oct 29 02:24:41 epithet *caneose: i have the recipe product i wrote to 5aday.gov, if you want that to look at*
Oct 29 02:24:43 zorb *caneose: can't kick the habit, eh ;)*
Oct 29 02:24:52 epithet *it's a straightforward recipe type-of-thing, with categories and such*
Oct 29 02:25:05 caneose *yes, is there a .xmi file for it that I can see?*
Oct 29 02:25:22 caneose *my question is assuming that I'm coming at this problem from UML*
Oct 29 02:25:45 amaryll *caneose: i was thinking of pulling some code in from trac, but i dunno.. they could probably benefit from zope interfaces either way ;d*
Oct 29 02:26:02 epithet *i didn't use UML modeling for that, caneose, sorry. it's AT-based, but not from UML.*
Oct 29 02:26:04 caneose *trac just donated their python-unittest system to svn, they're a great bunch*
Oct 29 02:26:12 Tim *amaryll: have you seen Kapil's SVNBrowser?*
Oct 29 02:26:17 epithet *python runs in my blood, but uml just occupies a teeny tiny wedge of my brain*
Oct 29 02:26:18 caneose *epithet: that's okay, I somewhat understand AT stuff*
Oct 29 02:26:23 amaryll *Tim: not since nola 1.*
Oct 29 02:26:41 Tim *amaryll: its evolved a little since then, it umm... works*
Oct 29 02:26:49 amaryll *i am talking to someone who has backe-end for creating and admin-ing svn repos from plone but i'm betting it's messy*
Oct 29 02:26:50 byte2b *lol*
Oct 29 02:26:54 amaryll *Tim: cool.*
Oct 29 02:27:00 amaryll *it worked then. ;)*
Oct 29 02:27:11 Tim *mostly, yeah*
Oct 29 02:28:15 epithet *mind you, caneose: there's nothing wrong with modeling recipes so that categories are full objects, the questions in cases like that usually become; will categories undergo workflow? have attached behavior? have a view of their own?, etc*
Oct 29 02:28:26 epithet *if not (& that's not usually the case for things like that), then it's overkill.*
Oct 29 02:28:29 amaryll *i really want an interface which is more agnostic, like trac and gforce have..*
Oct 29 02:28:33 Tim *amaryll: you might also want to look at https://svn.objectrealms.net/svn/public/plonecollab/trunk/PloneCollab/ based on what you seem to be up to*
Oct 29 02:28:34 amaryll *gforge, sorry*
Oct 29 02:28:58 amaryll *woah.*
Oct 29 02:28:59 epithet *Tim, others: some docs on how to turn your needs into an AT plan would be a fantastic piece of docs, tho' probably hard to write*
*Oct 29 02:29:19 ** *amaryll needs to figure out if membrane and teamspace can do business*
Oct 29 02:29:41 epithet *iirc, teamspace is pretty darn interwoven with cmfmember, no?*

Oct 29 02:29:59 caneose *epithet: no, I had a realization that it *was* overkill to make a food category into a class. it's really needs to be nothing but a list of properties attached to a recipe instance*
Oct 29 02:30:01 epithet *perhaps spanky|food can speak to that*
Oct 29 02:30:03 amaryll *epithet; it depends on cmfmember because it needs members to be content objects :)*
Oct 29 02:30:17 caneose *epithet: can you point me to an RTFM that demonstrates how I'd do this?*
*Oct 29 02:30:21 ** *spanky|food is now known as spanky|ate*
Oct 29 02:30:26 epithet *:)*
Oct 29 02:30:28 amaryll *so the model should work with membrane, but who knows how much of that interface is different..*
*Oct 29 02:30:32 ** *epithet is now known as joel|hungry*
*Oct 29 02:30:39 ** *joel|hungry is now known as epithet*
Oct 29 02:31:16 epithet *caneose: privmsg me your email addr and i'll send you a tarball of the product. you can see it in action at http://5aday.gov*
Oct 29 02:32:40 caneose *wow, you guys were contracted for 5aday.gov? how cool*
Oct 29 02:33:21 Tim *there is alot more Plone in the wild than you'd think*
Oct 29 02:33:34 amaryll *:)*
Oct 29 02:34:42 caneose *I can't believe that site is plone, that's some incredible branding*
Oct 29 02:38:48 epithet *caneose: sent*
Oct 29 02:38:57 caneose *thx*

Appendix 3: Categories applied on IRC conversations (explanations inside the codes) used with the TAMS Analyzer:

{Abbrivation}Abbreviations{/Abbrivation}
{Abbrivation>social}Abbreviations in normal language, like htf, iirc, fyi{/Abbrivation>social}
{Abbrivation>tech}Technical abbreviations like ZMI, SQL, AT ...{/Abbrivation>tech}
{Acknowledgment}A message to acknowledge a privous message. Like ok, yes, thanks, ...{/Acknowledgment}
{Acknowledgment>Greeting}A message to greet someone. Like hi, bye, cya{/Acknowledgment>Greeting}
{Acknowledgment>Negative}A negative message to acknowledge a previous message. Like yeah right, forget it, get lost!{/Acknowledgment>Negative}
{Acknowledgment>Positive}A positive message to acknowledge a previous message. Like Thanks, cool, yeah sure, ...{/Acknowledgment>Positive}
{by}{/by}
{Emote}Emoting: when putting * in front of the message{/Emote}
{Emoteicon}Emoting: textual signs, smilies, :) ;(´-) O-){/Emoteicon}
{envelope}{/envelope}
{event}{/event}
{Inquery}This is an Inquery{/Inquery}
{Inquery>Q}This is an Inquery in the form of a question{/Inquery>Q}
{Inquery>Q>code}Inquery where the question includes code{/Inquery>Q>code}
{Inquery>Q>Crct}This is an Inquery in the form of a correction of a question{/Inquery>Q>Crct}
{Inquery>Q>Ermsg}This is an Inquery in the form of a question including an error message{/Inquery>Q>Ermsg}
{Inquery>Q>Rsrc}This is an Inquery for a resource. It is *not* a question containing a resource or RA{/Inquery>Q>Rsrc}
{Inquery>Q>Spc}A specification of a question{/Inquery>Q>Spc}
{KD>CMF}Knowldge Domain of Content Management Framework{/KD>CMF}
{KD>CSS}Knowldge Domain of CSS{/KD>CSS}
{KD>ExStorage}Knowldge Domain of external storages like mysql or postgre{/KD>ExStorage}
{KD>Plone}Knowldge Domain of Plone{/KD>Plone}
{KD>Product}Knowldge Domain of a Certain Product{/KD>Product}
{KD>Product>Archetypes}Knowldge Domain of a the Archetypes{/KD>Product>Archetypes}
{KD>Python}Knowldge Domain of the Python programming language{/KD>Python}
{KD>Zope>zodb}Knowldge Domain of Zope : zodb{/KD>Zope>zodb}
{message}A message on IRC or mailinglist{/message}
{message>Date}An email is sendt on a certain date and time{/message>Date}
{message>Subject}An email usually have a subject to describe the message{/message>Subject}
{message>To}An email have one or more receivers{/message>To}
{Negotiation}Negotiation of meaning. (base for other levels){/Negotiation}
{Negotiation>Agreement}Negotiation of meaning. There is an agreemint in an argumentation or negotiation.{/Negotiation>Agreement}
{Negotiation>Realization}Negotiation of meaning. Someone realizes something, which makes something clear.{/Negotiation>Realization}
{Noresponse}Posts with no response{/Noresponse}
{Offering}An offer, in the form of a resource, code or similar{/Offering}
{OffTopic}Issues not concerning plone, zope, python or other related stuff.{/OffTopic}
{Opinion}An opinion or view{/Opinion}
{Opinion>Claim}An opinion or view in the form of a claim{/Opinion>Claim}
{Opinion>Sgst}An opinion or view in the form of an suggestion{/Opinion>Sgst}
{ProblemSolved}Here a question brought up is solved. Often marked with an acknowledgment, and then not mentioned again in the same sequence of text.{/ProblemSolved}
{ProblemSolved>Sub}Here a subquestion of a larger problem is solved. Often marked with an acknowledgment, and then not mentioned again in the same sequence of text.{/ProblemSolved>Sub}
{Quote}Quoting an earlier sendt message{/Quote}
{RA}An external reference{/RA}
{RA>API}An external reference: Application Programming Interface{/RA>API}
{RA>Code}An external reference: sourcecode{/RA>Code}
{RA>File}An external reference: file{/RA>File}
{RA>Image}An external reference: image{/RA>Image}
{RA>Maillist}An external reference: mailinglist{/RA>Maillist}
{RA>Paste}An external reference: pastebin (usually code or error message){/RA>Paste}
{RA>SVN}An external reference: subversion repistory containing files{/RA>SVN}

{RA>Webpage}An external reference: webpage{/RA>Webpage}
{Response}An response to a utterance{/Response}
{Response>A}An response to a utterance, an answer{/Response>A}
{Response>A>code}An response to a utterance, an answer including code{/Response>A>code}
{Response>A>Rsrc}An response to a utterance, a resource{/Response>A>Rsrc}
{Response>A>Sgst}An response to a utterance as an answer in form of a suggestion{/Response>A>Sgst}
{Response>A>Spc}An response to a utterance, a spescication or additional information to a given answer{/Response>A>Spc}
{Response>Q}A question as an response{/Response>Q}
{Self}Messages concerning oneself{/Self}
{Self>Action}Messages concerning an action one has done, is doing or is going to do{/Self>Action}
{Self>Experience}Messages concerning experiences done by the user. Both newly made experience (.. I cannot even get DBTab to work as it seems to be simply broken..) or more wider experiences (...i had ZClass-based objects living in a sql database quite sensibly a while back...){/Self>Experience}
{Self>Preference}Messages presenting oneself, dislikes/likes, what one preferes, and have positive/negative meanings about.{/Self>Preference}
{Self>Presentation}Messages presenting oneself, or giving views on oneself, skills abillity or history. Also initial presentations (..im a n00b...).{/Self>Presentation}
{sender}The sender of an message.{/sender}
{span}The person the message is meant for - john: "poipa: hi" means john says hi to poipa{/span}
{Typo>Correction}Correction of typographical error{/Typo>Correction}
{Usecase}Description of a use-case or scenario{/Usecase}

Appendix 4: Plone mailing-lists, with number of postings

Counted 24.06.2006

Searched gmane.comp.web.zope.plone.setup for articles 1,012 matching articles.
Searched gmane.comp.web.zope.plone.user for articles 53,648 matching articles.
Searched gmane.comp.web.zope.plone.announce for articles 32 matching articles.
Searched gmane.comp.web.zope.plone.documentation for articles 1,746 matching articles.
Searched gmane.comp.web.zope.plone.devel for articles 11,761 matching articles.
Searched gmane.comp.web.zope.plone.ui for articles 157 matching articles.
Searched gmane.comp.web.zope.plone.website for articles 528 matching articles.
Searched gmane.comp.web.zope.plone.internationalization for articles 2,577 matching articles.
Searched gmane.comp.web.zope.plone.archetypes.general for articles 3,405 matching articles.
Searched gmane.comp.web.zope.plone.archetypes.devel for articles 6,074 matching articles.
Searched gmane.comp.web.zope.plone.ngo for articles 48 matching articles
Searched gmane.comp.web.zope.eduplone.general for articles 152 matching articles.
Searched gmane.comp.web.zope.plone.cvs for articles 9,659 matching articles.
Searched gmane.comp.web.zope.plone.collective.cvs for articles 26,497 matching articles.

Total: 117 296 postings
(does not include The Foundation mailing-lists)