Can collaboration engineering help open source communities to structure their activities ?

Matthieu Lettry, Imed Boughzala, Aurélie Dudézert

Can Collaboration Engineering help Open Source communities to structure their activities?

Matthieu Lettry[†] Imed Boughzala[‡] Aurélie Dudézert[†]

[†]Ecole Centrale Paris, Laboratoire Génie Industriel
Grande Voie des Vignes Châtenay-Malabry 92295 France
aurelie.dudezert@ecp.fr
matthieu.lettry@ecp.student.fr

[‡] GET-INT, Department of Information System
9 rue Charles Fourier 91011 Evry France
Imed.Boughzala@int-edu.eu

Abstract

Collaborative work became an important stake for productivity, performance and innovation for companies. They bet more and more on communities of practice to support collaboration, sharing and creation of knowledge. The goal of this paper is to analyze the possible contribution of the Collaboration Engineering in the working of these communities, by taking as example, the Open Source communities.

Keywords: Collaboration Engineering, communities of practice, collaboration, Open Source communities.

1 Introduction

During the past decades, the need for collaboration has become more and more important for organizations. Collaboration enables people to achieve tasks that they could not achieve alone. It is easier to take several points of view, methods or strategies into account to resolve a problem.

The most used way of collaboration in organizations is the “project mode”. Although there is no doubt that the “project mode” is efficient and gives good results to achieve important and huge tasks, it is very formal and imposes a strong hierarchy which can creates some communication problems. Communication problems can sometimes lead to unproductive team work.

In order to avoid these problems and to make communication easy and efficient two research fields have become very interesting for organizations: First, there is the Collaboration Engineering (CE). This new research field aims to model and deploy repeatable collaborative processes to be executed by practitioners themselves of high-value recurring collaborative tasks [Kolfschoten, 2004]. CE initially deals with structured and formal processes and tries to improve them.

Second, there are the Communities of Practice (CoPs). CoPs are groups of people who communicate, create and share knowledge on a subject in an informal way generally. Since there is no hierarchical barrier. Communication in CoP is more spontaneous and efficient than it could be in a project group. That’s why organizations try to develop CoPs.

Although CE and CoPs both aim at improving group productivity and knowledge sharing, they can seem opposed because CE deals with formal and well-structured processes and CoPs deal with informal communication and ad-hoc processes within organizations.

This paper will examine these two ways of improving group productivity and knowledge sharing and will give some research tracks to check if Collaboration Engineering can help Open Source communities (as CoPs), which are a kind of communities of practice, to structure and to develop their collaboration activities

2 Collaboration Engineering

Although, there is no doubt that collaborative work can improve the performance of organizations. Team work sometimes faces difficulties

that generate a lack of productivity and the results may then be different from what was expected. Thus, team work is not an exact science and its results are not always predictable.

In order to make results more predictable, organizations sometimes rely on professional facilitators, especially when teams want to use Group Support Systems (GSS) tools [Kolfschoten, 2006]. But skilled facilitators are generally rare and expensive, many organizations cannot afford them, although it could improve their performances.

The goal of CE is to reduce the need for skilled facilitation expertise and to make a team lead a GSS session and manage its collaboration itself, without professional facilitator but with predictable results.

2.1 Definition

Collaborative Engineering is a design approach that models and deploys repeatable processes for recurring high-value collaborative tasks for execution by practitionners themselves using facilitation techniques and technology [Kolfschoten, 2006].

CE focuses on recurring rather than on ad-hoc processes because the benefits of designing a recurring process accrue each time the process is done and because the designs of recurring processes are intellectual capital for organizations [de Vreede, 2005] since practitionners must learn methods and then transfer them to other practitioners.

There are four facets in CE [Boughzala, 2007]:

- The a priori and a posteriori evaluation of collaboration, its tools and their use;
- The modelling of collaborative processes;
- The specification of collaborative technologies;
- The management of collaborative knowledge.

There are three key roles in Collaboration Engineering: the facilitator who designs and executes a collaborative process, the collaborative engineer who designs and transfers a process to practitioners, and the practitioner who learns a process from an engineer and who executes it [Kolfschoten, 2006].

2.2 Modeling of a collaborative process

In this part, we will see that the modeling of a collaborative process relies on the use of packaged and reusable elementary bricks called ThinkLets.

2.2.1 The CE approach

First, the collaboration engineer identifies the sequence of steps that make the collaborative process. Each step is linked to a phase (evaluate alternative, choose alternatives, take action...) and the deliverables (a set of alternatives, a decision, a plan...) of this phase.

Each step of a process is achieved through activities that create patterns of collaboration among the members. Five patterns of collaboration characterize people's activity during a step. Each of these patterns is defined in terms of moving a group activity from an initial state from an end state [de Vreede, 2005]:

- **Diverge:** Move from having fewer to having more concepts;
- **Converge:** Move from having many concepts to focus on an understanding of a few deemed worthy of further attention;
- **Organize:** Move from less to more understanding of the relationships among concepts;
- **Evaluate:** Move from less to more understanding of the benefit of concepts toward attaining a goal relative to one or more criteria;
- **Build consensus:** Move from less to more agreement among stakeholders so that they can arrive at mutually acceptable commitments.

Finally, the collaboration engineer has to select existing building blocks and plug them in to specify how a given pattern should be realized when the process is run. Such building blocks are the ThinkLets and each of them encapsulates specifications for the repeatable activities it documents [de Vreede, 2005]. ThinkLets are generally defined as named, packaged facilitation interventions that create a predictable, repeatable pattern of collaboration among people working together toward a goal [Briggs, 2003].

Variations are often applied to existing ThinkLets to create predictable changes in the group behavior. These variations are called modifiers and are defined as a named, packaged set of modifications that can be applied to one or more ThinkLets to produce a predictable change within the pattern the ThinkLet produces [Kolfschoten, 2006].

If ThinkLets and modifiers are building blocks of collaborative processes, these blocks must be linked to each others to make the process consistent. The links are called ThinkLets transitions. A ThinkLet transition is defined as all that must transpire to move people from the end of one ThinkLet to the beginning of the next. A transition

deals with changes of data, changes of capabilities, changes of orientation [de Vreede, 2005].

Some sequences of ThinkLets and transition are frequently reused. These sequences have been amalgamated into a named, reusable compound ThinkLet called a module [Kolfschoten, 2006]. As with ThinkLets, modules must be linked together with transitions, called module transitions.

2.2.2 First conceptualization of ThinkLets

In its first conceptualization, a ThinkLet has three components: a tool, a configuration of that tool and a script [Kolfschoten, 2004]:

- The **tool** concern the specific technology used to create the expected pattern of collaboration among people (it can be anything from pencil and yellow stickers to technologies such as GSS).
- The **configuration** defines how the tool is prepared (projection on individual or common screen), set up (configured for anonymous communication or not) and loaded with initial data (a set of question for example).
- The **script** is the set of recommendations the practitioner will have to tell the group to make it move through the expected pattern of collaboration.

This conceptualization has many drawbacks. First, ThinkLets are dependent on technology, which is surprising since, for example, a brainstorming can be lead with GSS or stickers on a wall. Second, some important changes in a script have no impact whereas little changes in other ThinkLets have a big influence. Some aspects of a scripts have more impact on behavior than others. Finally, each change of tools or scripts creates a new ThinkLet. This could lead to an explosion of the number of ThinkLets and could be a hurdle to the transfer of knowledge across organizations.

These observations lead researchers to develop a new model of ThinkLets based on more fundametal and elementary elements.

2.2.3 Second conceptualization of ThinkLets

In a second conceptualization, ThinkLets are defined in terms of action, capability, rules, roles and parameters. More precisely a ThinkLet now describes how a participant executes an **action** dependent on the **role** he performs in the process in three manners [Kolfschoten, 2004]:

- It describes the technological **capabilities** (and not the tool)

required for the participants to execute his **action**.

- It describes the **rules** needed to constrain particular **actions** (add, delete, edit, relate, judge).
- It describes the **parameters** that are required to instantiate the effectuation of the participants' **action**.

With this model, ThinkLets are technology independent, the cognitive load is reduced and the concept will be easier to transfer to new collaboration engineers [Kolfschoten, 2006]. Moreover, this models allows researchers to begin thinking about families of ThinkLets that could be adapted to a specific situation thanks to the right modifier.
Even if this model seems interesting from a theoritical point of view, it must be tested to see if it is a real improvement in practice. Of course, it als has drawbacks: for example, the impact of facilitation intruction variations on motivation are not adressed.

2.3 ThinkLets classification

Since each variation in the components of a ThinkLet creates a new ThinkLet, it becomes really important to propose classifications that help engineers choose the right ThinkLets.

Several classifications have been made using the first conceptualization of ThinkLets[1]. The three classifications which will be presented [Kolfschoten, 2004] are based on the model of a group process presented by Nunamaker et al. [Nunamaker, 1991].

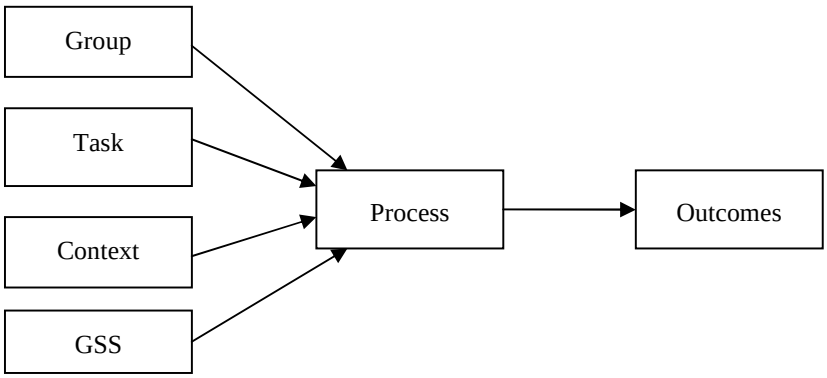

Figure 1: Model of group process [Nunamaker, 1991]

1 The second conceptualization is too recent, that is why no classification is based on it.

The **first classification** relies on the **phases** the group goes through. This classification seem logical since a ThinkLet is a building block that makes a group move through a phase but the problem is that a ThinkLet can fit in several phases.

A **second classification** is made with the **pattern of collaboration** created by a ThinkLet. Although it is interesting to move from the identification of a phase to the choice of a ThinkLet. This classification has two main drawbacks: patterns are interdependent and some ThinkLets can create several patterns.

The **last classification** is based on the **nature of the outcomes**: collection, structure, overview. It can be made more accurate with other attributes: judged/non-judged and clean/dirty. It corresponds to the reflection of the engineer who has in mind the deliverable while designing the process but the problem is that some ThinkLets can be used to several types of outcomes.

All these drawbacks suggest that there may be smaller units of collaboration than ThinkLets and that this model has reached its limits which strengthen the need for the new conceptualization presented above. Moreover, no current classification proposes a unambiguous classification. To this end, classifications based on the new model and research in other disciplines could offer interesting possibilities.

2.4 Deployment of Collaboration Engineering

Now that we have seen what thinkLets are and how they can be used to build processes, we will see the success criteria for the choice of a collaborative process and the different phases of a CE approach.

2.4.1 Success criteria

In order to make a CE intervention be a success, the most important point is to choose the right task. The chosen task must meet some requirements [Dean, 2006].

Clearly defined outcomes are important for collaborative activities because CE design process creates a plan to orchestrate a series of steps to achieve one more specific goal.

The best candidate tasks for a collaborative intervention are recurring **high-value** task running inefficiently. If the productivity of these tasks is improved, they can make organizations save time and money each time they are performed.

An **appropriate task type** for CE is a concerted effort collaboration where synchronicity and a high degree of interaction occur for at least part of the process. Moreover, the possibility of improving productivity of a task increases as the number of participants, the quantity of exchanged information and the use of technology.

In addition to identifying the desired outcome, the collaboration engineer should also attempt to determine whether the **goal is shared** among group members and whether the group **goal is congruent** with the individual goals of group members.

An appropriate task must have a **process champion**, i.e. a person in the organization with both the willingness and the ability to support the effort through to completion. He must have a good knowledge of both the process and the used technologies.

Finally, the success of a collaboration intervention is generally linked to the **budget** it was allowed.

2.4.2 The CE implementation

Once the process to be improved is chosen, several major phases must be followed in the right order to increase the chances that the collaboration intervention improves the process [Santanen, 2006].

Phases	Description
Field interviews	Requirements gathering from the problem owner and different stakeholders.
Design phase	It includes the identifications of the patterns of collaboration, the choice of ThinkLets and the validation of the process.
Transition phase	The collaboration engineer begins transferring the process to the organizational actors who test it on pilot programs.
	The collaboration engineer gathers feedback and improves the process to adapt it to the organization.
Practitioner implementation	The organizational actors, with the help of the collaboration engineer, begin to deploy the process across the organization.
Sustained organizational use	The process is deployed by the sole practitioners, it becomes part of the organization culture and it is transferred to a second generation of practitioners.

Table 1: The CE implementation phases [Santanen, 2006]

To help the collaboration engineer choose his ThinkLets and make his processes in the Way of Modelling, several documents and models have been created by collaboration researchers [de Vreede, 2005]:

- About 70 ThinkLets have been well documented in **ThinkLets Description Documents** (TDD).

- The **ThinkLets Notation Model** (TNM) is a formal textual method for documenting and communicating group process designs in a very synthetic way.
- The **Facilitation Process Model** (FPM) uses symbols to document the flow of a process from ThinkLet to ThinkLet.

The control of the CE intervention uses standard project management principles and techniques.

3 Communities of practice (CoPs)

3.1 Definition

CoPs are defined as groups of people who share a concern, a set of problems or a passion about a topic and who deepen their knowledge and expertise in this area by interacting on an ongoing basis [Wenger, 1998]. They appear when three important conditions are combined: a concerted entreprise, a shared repertory and a mutual commitment.

CoPs can be spontaneous, when they emerge through the initiative of some individuals, or sponsored when organizations deliberately create them [Boughzala, 2007]. Organizations are particularly interested in communities of practice since it can represent a powerful tool of Knowledge Management to create and transfer knowledge between peers [Lefebvre, 2004]. A particular type of communities of practice which is interesting for organizations is the Virtual Professional Communities (VPC), which are defined by ECOLEAD (European Collaborative networked Organizations LEADership initiative) as "*associations of individuals (being those employed by the company or individual professionals) explicitly pursuing an economic objective, identified by a specific knowledge scope and aimed at generating value through members' interaction, sharing and collaboration. This interaction is optimized by the synergic use of information and communication technology-mediated and face-to-face mechanisms*".

There are differents types of actors in a CoP (coordinator/facilitator, expert, sponsor, leader, administrators, passive members, lurker…) which can be classed according to their implication levels.

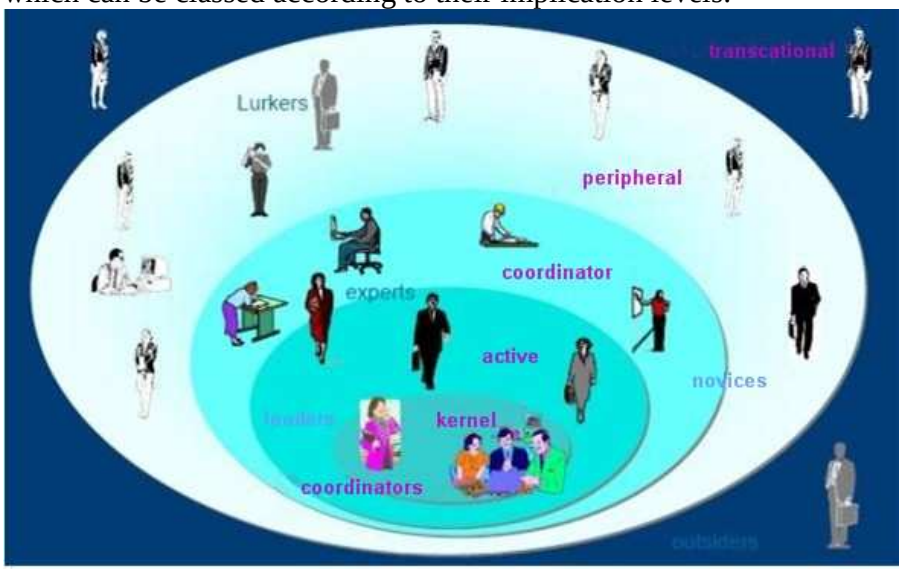

Figure 2: Levels of participation ([Soulier, 2004] adapted from [Wenger, 01])

The success of a community can be endangered by power issue. The problem of authority and hierarchy in a community is very different from classical hierarchical organizations since the influence of a member relies on the legitimacy he acquires. In a community of practice "*the roles, the responsabilities and the shape of the community are never clearly a priori determined or definitively fixed. No actor is able to impose them because he has a lack of legitimacy and because the environment is complex and unpredictable*" [Benghozi, 2001]

3.2 Models of growth

The lifecycle of a community can be decomposed into six phases [Boughzala, 2005] (inspired from [Wenger 2002]):

- **Emergence**: a community emerges from an idea brought by a couple of people.
- **Structuring**: the objective becomes more precise, the community begins to grow and roles are clearer.
- **Maturation**: roles are well defined, members cooperate and develop new resources.
- **Officialization**: the community and its action are recognized by the organization(s).
- **Consolidation**: the community grows and is still active but

interactions intenstity decreases more and more because of identity, motivation or objectives issues.

- **Transformation**: a transformation must be done. It can be a change of objectives, structures, technologies, a union with other communities, the disassociation or even the winding-up of the community.

The structure of a community (rules, roles and processes) and the implication of the different types of actors are different from a phase to another.

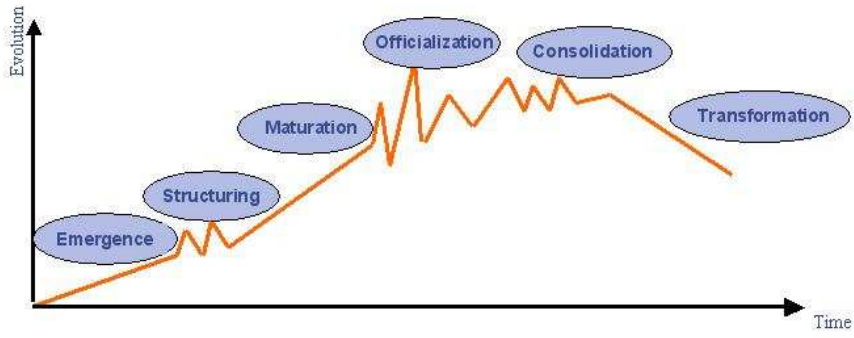

Figure 3: Lifecylce of communities [Boughzala, 2005]

From an economical point of view, Benghozi et al. propose a model of growth for two types of CoPs [Benghozi, 2001]:

- **Communities of services**, based on the use, by a group of persons, of the same set of on-line services.
- **Communities of crafts**, based on the exchange of similar professional practices.

The model is based on three points:

- The evolution of the relations structure.
- The level of equipment and use of communication and computer technologies.
- The evolution of resources within the community.

Although the two models seem to converge, they have really different dynamics of growth. For communities of services, the conception of new economic models implies the creation of news services, wheras for communities of crafts, the creation of news services leads to the research for new economic models.

4 Open Source communities

Open Source communities are communities who develop Open Source software. Open Source softwares are softwares whose source code is

published and made available to the public. It implies that anyone can modify, copy and distribute it freely. The code of programs evolves through the collaboration of the members of the community.
A good example is the Linux community which a Unix-like operating system kernel. This community is particularly interesting since it becomes more and more active [Dempsey, 2002] and it is composed of contributors from the whole world (even if the majority is European [Dempsey, 2002]). Even if it is said to be hard to use, if it suffers from a lack of visibility and if the community way of supporting the system can afraid some people, Linux will certainly go on growing. Its mains advantages are a low-cost, its scalability (it can be run on old and slow computers), its flexibility (thanks to its open source code), and its reliability. Moreover, Linux's modularity enables it to adapt easily to each culture [Lanier, 2005].
Regarding the points developped in the previous part, Open Source community developers are mutally comitted in proposing alternatives to softwares developped by big private firms (this is their concerted entreprise) and their shared repertory is computer knowledge.
Initially, the Linux community was spontaneous. It relied on a school project launched by a Finnish student called Linus Torvalds. Then, some Linux communities became sponsored by firms or rich individuals. Thus, the Linux universe covers both spontaneous and sponsored faces of CoPs.
Linux communities are also virtual professional communities (VPC) since contributors have the same professional background and cooperate through the Internet.
Finally, these communities have characteristics of both communities of services, since private investments enable them to develop new products, and of communities of crafts, because entreprises became interested in Linux through the quality of the services.

5 Discussion and conclusion

In this part, we will discuss why Open Source communities (as an example of CoPs) can be interesting to use CE.

As it has been said above, the success of a CE intervention depends on several criteria in the choice of the task. Open Source community tasks seem to gather several of these criteria since the outcomes are clearly defined and shared by all members. Budget could be provided by firms which sponsor the communities. The task type would be a meeting, even an e-meeting, activity with high value for the community. Finally,

finding a process champion should not be a problem since members of a community are used to working together and have a good knowledge of the practice. We conclude that all the success criteria of a task choice are gathered in a Open Source community like Linux.

The implementation of CE implies that there is someone within an organization who has enough authority to make people follow a formal and strict way of working (It is the role of the coordinators when the community becomes ripe in the maturation phase). Instead of telling people how to drive a given process, the idea is to propose them to use CE facilitation techniques for their ad-hoc processes. They would use it only if they want and would judge if it improves their productivity. In such a context, Collaboration Engineering could be used progressively to automate both occasional and recurrent processes. We can imagine a library of process placed at their disposal [Boughzala, 07].

Even if the CE tools are used on voluntary basis, there must be someone with enough authority who drives the process to make the group respect the instructions given by ThinkLets. But as for methods, it is really difficult to impose authority on members of a community. In a community, authority is replaced by legitimacy [Boughzala, 2007]. Thus, a person (generally the coordinator or an expert) who is recognized for having great legitimacy and who is interested in testing Collaboration Engineering methods could learn how to use these facilitation techniques by reading ThinkLets Documentation Models. He could also try to model a process with its steps, its patterns of collaboration and ThinkLets by using tools such as ThinkLet Notation Model and Facilitation Process Model. Then, he could suggest other members to use them to see if it can make them increase their productivity. We can imagine that such a person would be able to incite people to test CE tools.

Regarding the lifecycle of communities, Open Source communities which are as developed and structured as Linux could be expected to accept CE methods when they enter in their maturation phase since it is the phase in which roles are well defined and members really cooperate to develop news products.

If CE is accepted by the community and if it improves the productivity of some processes, we can also wonder about its long terms effects. May be, it can provide the community a better stability which would postpone the inevitable transformation phase of the community. In that way, CE could be a mean to pursue the knowledge sharing in Open

Source communities. This could be very useful to the maintenance of Open Source developments and to avoid that such Open Source development knowledge disappears in the transformation phase.

Regarding the economical growth, the model proposed in [Benghozi, 2001] for communities of craft suggest that the creation of news services leads to the research for new economic models. In Collaboration Engineering, the use of ThinkLets implies to choose technologies to support the process. We can wonder if these technologies can be an engine for the creation of new services and then imply the research for new economical models.

To conclude, although these suggestions offer interesting research perspective, naturally they must be empirically tested in the field in order to be validated. It could also be interesting to extend encouraging results to other types of CoPs.

References

[Benghozi, 2001] Benghozi, P.J., Bitouzet, C., Soulier, E., Zacklad, M. Le mode communautaire: vers une nouvelle gestion des connaissances, *Actes du 3e Colloque International sur les Usages et Services des Télécommunications (ICUST)*, 12-14 juin 2001.

[Boughzala, 2005] Boughzala, I., Kaouane, F. Vers un cadre méthodologique pour la conception des communautés professionnelles virtuelles, *10ème colloque de l'AIM*, 2005.

[Boughzala, 2006] Boughzala I., Dudezert A., Heibült H. VPC : A Strategic Instrument for KM and Organisational Learning, *3rd International Conference on Intellectual Capital, Knowledge Management an,d Organisational Learning*, 2006.

[Boughzala, 2007] Boughzala, I. Ingénierie de la collaboration, *Hermes*, 2007.

[Boughzala, 2007] Boughzala I, Les communautés professionnelles virtuelles et gestion des connaissances. Dans Boughzala I., Ermine J-L. (Eds). *Management des connaissances en entreprise 2ème édition*, 2007, pp. 167- 191.

[Briggs, 2003] Briggs R.O., de Vreede G.J., Numaker J.F. Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems. *Journal of Management Information Systems*, Vol 19, No 4, 2003, p 31-64.

[Dean, 2006] Dean D.L., Deokar A., Ter Bush R. Making the Collaboration Engineering Investment Decision, Proceedings of the 39 HICSS, Vol 1, 2006.

[Dempsey, 2002] Dempsey B.J., Weiss D., Jones P., Greeberg J. Who Is An Open Source Software Developer? Profiling a Community of Linux Developers, *Communications of the ACM*, Vol 45, No 2, 2002, p 62-67.

[Harder, 2005] Harder R.J., Keeter J.M., Woodcock B.W., Ferguson J.W., Wills F.W. Insights in Implementing Collaboration Engineering, *Proceedings of the 38th HICSS*, 2005.

[Kolfschoten, 2004] G.J., Kolfschoten, Briggs, R.O., Appelman, J.H., de Vreede, G.L. ThinkLets as Building Blocks for Collaboration Processes: A Further Conceptualization, G*roupware: Design, Implementation and Use*, 2004, p 137-152.

[Kolfschoten, 2006] Kolfschoten, G.L., Briggs, R.O., de Vreede, G.J., Jacobs, P.H.M., Appelman, J.H. A conceptual foundation of the ThinkLet concept for Collaboration Engineering, *International Journal of Human Computer* Studies, Vol 67, No 7, 2006, p 611-621.

[Lanier, 2005] Lanier C.R. Linux and the Appeal to Cultural Values, *IEEE technology and Society Magazine*, Vol 24, No 4, 2005, p 12-17+42.

[Lefebvre, 2004] Lefebvre P., Roos P., Sardas J-C. Les théories des communautés de pratique à l'épreuve: conditions d'émergence et organisations des communautés, *Systèmes d'Information et Management*, Vol 9, No 1, 2004, p 25-42.

[Leibovitch, 1999] Leibovitch E. (1999). The Business Case for Linux, *IEEE Software*, Vol 16, No 1, 1999, p 40-44.

[Nunamaker, 1991] Nunamaker, J.F., Dennis, A.R., Valacich, J.S., George, J.F., Electronic Meeting Systems to support Group Work, *Communications of the ACM*, Vol 34, No 7, 1991, p 40-61.

[Santanen, 2006] Santanen E., Kolfschoten G., Golla K. The Collaboration Engineering Maturity Model, *Proceedings of the 39th HICSS*, Vol 1, 2006

[Santos, 2004] Santos I.H.F., Valle C., Raposo A.B., Gattass M. A Multimedia Workflow-based Collaborative Engineering Environment, Integrating an Adaptative Workflow System with a Multimedia Collaboration System and a Collaboration Virtual

Environment for Petroleum Engineering, *Proceedings of the Sixth International Conference on Enterprise Information Systems*, 2004, p 259-262.

[Soulier, 2004] Soulier E., Les communautés de pratiques au coeur de l'organisation réelle des entreprises, *Systèmes d'information et Management*, Vol 9, No 1, 2004, pp 3-48.

[de Vreede, 2003] de Vreede G.J., Davison R.M., Briggs R.O. How a Silver Bullet Might Lose Its Shine, *Communication of the ACM*, Vol 46, No 8, 2003, pp 96-101.

[de Vreede, 2005] de Vreede, G.J., Briggs, R.O. Collaboration Engineering : Designing Repeatable Processes for High-Value Collaborative Tasks, *Proceedings of the 38th HICSS*, 2005.

[Wenger, 1998] Wenger, E. Communities of Practice: The social fabric of a learning organization, *cambridge U. Press*, New York, New York, 1998.

[Wenger, 2001] Wenger, E. Supporting communities of practice a survey of community-oriented technologies. Self-published report available at www.ewenger.com/tech , 2001

[Wenger, 2002] Wenger E., Mc Dermott R, Snyder W. (2002) "Cultivating communities of practice", Boston, MA, Harvard Business School Press.