A Collective-Intelligence View on the Linux Kernel Developer Community

Haoxiang Xia
Dalian University of Technology, China

ABSTRACT
With the rapid proliferation of all sorts of online communities, the knowledge creation and dissemination in these online communities has become a prominent social phenomenon. In this paper, one typical Open Source Software community, i.e. the online community of Linux kernel developers, is studied from the perspective of collective intelligence, to explore the social dynamics behind the success of the Linux kernel project. The Linux kernel developer community is modeled as a supernetwork of triple interwoven networks, namely a technological media network, a collaboration network of the developers, and a knowledge network. The development of the LDC is then an evolutionary process through which the supernetwork expands and the collective intelligence of the community develops; and a bottom-up approach is attempted to unravel this evolutionary process.

Keywords: Online Community, Open Source Software (OSS), Knowledge Creation, Collaboration Network, Collective Intelligence

INTRODUCTION
In recent years, with the explosion of the Internet and the World Wide Web, online communities have become a prominent social phenomenon (Preece & Maloney-Krichmar, 2005); correspondingly, these online communities are playing an increasingly vital role in society-wide knowledge developments. Typical examples include the Wikipedian community which produces high-qualified encyclopedia (Giles 2005), the open source software (OSS) developer communities which build complex software systems like Apache and Mozilla (Mockus, et al., 2002), and “Science 2.0” communities for scientific collaboration (Shneiderman, 2008). The proliferation of such knowledge-intensive online communities may raise a critical research issue, i.e. to study how knowledge is created and diffused in the online communities, and how the communities themselves grow during the collective actions of the participants. This issue is akin...
to the well-discussed research field of “Knowledge Management” (e.g. Nonaka & Takeuchi, 1995; Alavi & Leidner, 2001), which is usually focused on the “management” of knowledge and knowledge processes in a formal organization. Nevertheless, it can be argued that the existing theories and models for organizational knowledge management cannot be simply transplanted to the situation of online communities, since the knowledge processes in the online communities are fundamentally different from those in a formal organization. The knowledge creation, dissemination and utilization in the online community are commonly accomplished by independent participants in a self-organizing and “autopoietic” (Varela, et al., 1974) fashion, while in the formal organization the knowledge processes usually take place under the centralized managerial control to achieve some well-defined organizational objectives. New theories and models to explain the knowledge processes in online communities are required.

The Linux kernel developer community is one of the most famous online OSS communities; and this community may provide a fine case to study the knowledge processes in the online communities. The Linux kernel is the operating system kernel that underpins all distributions of Linux operating systems, which was initiated in 1991 by Finnish programmer Linus Torvalds and has thereafter been developed by thousands of part-time voluntary programmers scattered across the Internet without formal organization or centralized control. Along with the development of the Linux kernel, the online community of the contributors, or the Linux Kernel developer community, rapidly grows. Due to the unbelievable success of Linux, it would be worthwhile to examine how the collective actions of the voluntary programmers develop such large-scaled and complex software of high quality in a spontaneous way, and to examine how the online community of the voluntary programmers self-grows. As software development can be regarded as one form of knowledge creation, this examination may from another aspect contribute to enrich our understandings of the knowledge processes in online communities.

Therefore, in this paper we try to explore the knowledge-intensive online communities by giving an analysis on the actual case of the Linux-kernel developer community (LDC for short). In an earlier effort, we suggested that many online communities manifest some degree of community intelligence (Xia, et al., 2008; Luo, et al., 2009). Based on this idea, we in this paper try to explore the underlying dynamics for the evolution of the LDC as well as the development of this community’s knowledge product, the Linux kernel.
A SHORT HISTORY OF THE LDC

To facilitate further discussion, the history of Linux is shortly introduced, with the focus being placed on the growth of the developer community in which the Linux kernel is collectively created and continually updated.

Linux was initially developed by Linus Torvalds in 1991, when he was a student in computer science at University of Helsinki. His initial motivation was to write programs in order to use some UNIX functions in his own PC with an 80386 processor; and he implemented a task-switching program, a disk driver and a small file system, which constituted Linux 0.01. On 25 August 1991, he announced this skeletal operating system in the newsgroup “comp.os.minix” and asked for suggestions for the preferable features. Then, his continuous efforts ended up to Linux 0.02, which came on October 5th. Together with the free release of the source code, he posted another message in the same newsgroup to seek feedbacks as well as possible contributors or co-developers. This was a critical event for Linux since it started the collective journey of Linux development. The response was instantly positive; of the first ten people to download Linux, five sent back bug fixes, code improvements, and new features. By the end of the year, when Linux finally became a stand-alone system in Version 0.11, more than a hundred people worldwide had joined the Linux newsgroup and the mailing list (Kurabawa, 2000). Since then, the Linux developer community rapidly expands, together with the rapid development of the Linux operation system.

One critical measure is the development of the Linux kernel in term of the source lines of code (SLOC). The actual SLOCs in some typical versions are listed in Table 1. This table shows that the Linux kernel rapidly expands in the past 18 years.

<table>
<thead>
<tr>
<th>Release Year</th>
<th>Kernel Version</th>
<th>Source Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1991</td>
<td>Linux 0.0.1</td>
<td>10,239</td>
</tr>
<tr>
<td>1994</td>
<td>Linux 1.0.0</td>
<td>176,250</td>
</tr>
<tr>
<td>1996</td>
<td>Linux 1.2.0</td>
<td>310,950</td>
</tr>
<tr>
<td>1999</td>
<td>Linux 2.2.0</td>
<td>1,800,847</td>
</tr>
<tr>
<td>2001</td>
<td>Linux 2.4.0</td>
<td>2,210,149</td>
</tr>
<tr>
<td>2003</td>
<td>Linux 2.6.0</td>
<td>5,929,913</td>
</tr>
<tr>
<td>2008</td>
<td>Linux 2.6.25</td>
<td>9,232,484</td>
</tr>
</tbody>
</table>

Another critical measure for the growth of the LDC is to count the community size. Since its creation in 1991, this open operating system has attracted increasing numbers of developers worldwide. In the year of 1993, there were over 100 developers worked on the Linux Kernel. More recently, as reported by Koah-Hatman et al. (2008), each release since version 2.6.11 generally contains the work of nearly 1000 developers. Since 2005, about 3700 individual developers have contributed to the kernel. Their report also shows that the number of developers gradually increases from 483 in Version 2.6.11 to 1057 in Version 2.6.24.

The previous description reveals much information for the LDC. Linux had a somewhat haphazard starting-up, since Torvalds himself was not even aware that he was writing an operating system when he began programming the first task-switching system in 1991. He did not anticipate at that time that Linux would catch such persistent enthusiasms from so many developers and users, nor could he imagine that Linux would become such huge and complex software and such a successful product. From this haphazard starting-up, the community rapidly expands together with the explosion of the Linux (kernel) product. Behind the success of the Linux operating system and the growth of the Linux developer community, two questions naturally come to the fore: what are the mechanism underlying the evolution of this Linux developer community; and why this complex software product can be successfully created in this largely open community without thorough planning or centralized control.

**SOME PREVIOUS VIEWS**

The intriguing phenomena of the Linux operating system and the corresponding developer community have attracted great attention in the last decade. Here a few typical explanations are shortly introduced.

One well-noted work is given by Raymond (1999), who distinguished two different modes of software development, i.e. the “cathedral” model of most of the commercial world and the “bazaar” model of the Linux world. The success of the Linux project is then attributed to the inherent openness in the bazaar model. On one hand, Raymond believes one success factor of Linux is the frequent releasing and updating so that the users can quickly detect the bugs; one the other, with a large developer base, the project leader Linus Torvalds can safely rely on others to write and test the code for him. The openness is for sure an important factor for enhancing Linux development; however, it is farfetched to use the openness to explain everything.
Kuwabara (2000), by contrast, argued that the bazaar-analogy is too simplistic, and the success of the Linux project should be understood from an evolution and complex-adaptive-system (CAS) point of view. To him, behind the Linux project is a bottom-up engineering approach that effectively challenges the top-down worldview entrenched in the monolithic software engineering approach. We largely agree with his CAS view; our further argument is that the Linux developer community is not merely an evolutionary system, but also an evolutionary intelligent system, as the knowledge-and-problem-solving aspect is less stressed in Kuwabara’s work. Another limitation of Kuwabara’s work is that he points out that the Linux project should be considered as a CAS, but the underlying mechanisms of the formation of this CAS is not well-addressed.

Iannacci (2005), in his PhD dissertation, gave a social epistemological analysis for the LDC. The major focus is on the coordination mechanisms that are emerged in the community. Three mechanisms are analyzed in his work, namely standardization, loose-coupling to form a “heterarchical” structure, and partisan mutual adjustment. Lee and Cole (2003), from the knowledge-creation perspective, attempted to develop a community-based model of knowledge creation, adopting the evolutionary framework suggested by Campbell (1960) and emphasizing the role of criticism and critical evaluation as a key driver in the evolutionary processes.

The prior contributions, among many others, are no doubt valuable for deepening our understanding of the LDC. However, their work just reflects the partial facts of the LDC from different facets; more endeavors are still needed to obtain a more comprehensive view.

TOWARDS A COMMUNITY-INTELLIGENCE VIEW ON THE LDC

Based on the prior observations, in this section we give our analysis on the LDC, basically regarding this community as an evolutionary intelligent system in which the participants collectively create knowledge and solve problems. The LDC largely reflects the phenomenon of “community intelligence” (Xia, et al., 2008; Luo, et al., 2009) or the “collective intelligence” (Levy, 1997) of online communities. Subsequently, we try to explain the evolution of the LDC from the aspect of community intelligence.

The LDC as an Example of Community Intelligence
Our central point is that the development of the LDC shows a typical example for the formation of community intelligence. Under this view, the LDC contains a “supernetwork” (Nagurney & Wakolbinger, 2005) structure that consists of triple interwoven networks, as illustrated in the following figure.

First, the LDC grows in the environment of computer networks, especially of the Internet. Thus there is a technological network which underpins the interactions of the LDC. The major role of this technological network in the LDC is to serve as communication media that enable information passing, knowledge diffusion, and interpersonal collaboration. Therefore, we can call this network “media network”. Second, intermediated by the “media network”, there is a “human network” of the participating programmers, in which the nodes are the programmers and the edges are the communications and collaborations between the programmers. The collaborations of the programmers on one hand create the product of Linux kernel, and on the other hand generate knowledge about the development of the Linux kernel. The knowledge about the Linux kernel development is logically structured in a networked form, and we can call it “knowledge network”. As discussed in our earlier work, these three networks are intertwined to form a supernetwork. This supernetwork can be termed as a “knowledge supernetwork” since it is characterized by the knowledge network and the collaborative knowledge work in the human network to form the knowledge network.

Furthermore, this knowledge supernetwork is “intelligent” in the sense of problem-solving through the collective action of Linux development. In other words, this online community of the independent programmers manifests higher capabilities in the collective software development than any individual programmer per se. The collective intelligence of the LDC can be exhibited in two aspects. On one hand, the community contains a knowledge base (i.e. the “knowledge
network” in the prior triad) superior to the member programmer’s; on the other hand, the community is self-organized to form a collaboration network (i.e. the “human network” in the prior triad) so that the collective action of software development is well-coordinated and effective.

Corresponding to the prior supernetwork view of the LDC, a further research topic is to examine how this knowledge supernetwork grows through the collective work in Linux kernel development, more specifically to examine the co-evolution of the human network and the knowledge network. To carry out this investigation, the starting point is that the overall evolution of the LDC is not a monolithic process, compatible with the “bottom-up engineering” of Linux development as Kuwabara contended. Linux is developed by thousands of independent programmers without complete goal-setting, thorough planning, or top-down task-assignment. When Torvalds initially announced Version 0.01, he himself had no clear intention for the future of Linux; and he just asked for feedbacks and suggestions on the features which Linux should contain. During the entire process of Linux development, he also avoids imposing long-term plans and visions on the community:

“That way I can more easily deal with anything new that comes up without having pre-conceptions of how I should deal with it. My only long-range plan has been and still is just the very general plan of making Linux better.” (Interview, cited from Kuwabara 2000)

Instead, the stage-goals of development usually rise from collective efforts and discussions. Unlike a formally-organized software project, neither Torvalds nor any other sub-system maintainer authoritatively arranges a particular contributor to accomplish a particular task. From the very beginning, it has been left to each contributor to decide what to work on at the moment:

“In developing Linux, you have complete freedom to do whatever you feel like doing. There’s no specification anywhere of what the Linux kernel has to end up doing, and as such there is no requirement for anyone to do anything they are not interested in.” (Personal Interview, cited from Kuwabara 2000)

Thus, the entire project is carried out in a spontaneous mode with no global coordination. In the project, nobody can anticipate what will be added or modified in the next release of the Linux kernel. Associated with this spontaneous and bottom-up development of the Linux operating system is the endogenous growth of the corresponding community of the developers. This LDC again grows from bottom up as the result of the free choices of the contributors. No one can exactly anticipate who will join the community by submitting patches and bug-fits at the next
stage of work, or who will leave because of any reason. Without formal organization, up to now this community has somehow stabilized to be an effective community that efficiently creates complex software of high quality.

Accordingly, a “bottom-up” and generative approach (Epstein & Axtell, 1996) should be adopted to study the social dynamics of the LDC evolution. Following this approach, we need to identify the local actions that eventually propel the overall evolution of the community, and then to examine how the global evolutionary processes can emerge from the local action.

**An Exploration of the Underlying Dynamics of the LDC Evolution**

In the previous sub-section, we argue that behind the development of the LDC is a spontaneous process of the evolution of community intelligence. In this sub-section, a bottom-up investigation is given to explore the underlying mechanisms.

**Identifying the Local Force**

With global project-planning and top-down task-assignment being absent in the Linux kernel development, the programmers have complete freedom to do whatever they feel like to do. The individual programmers’ coding behavior is obviously one of the important local actions that are directly meaningful for the production of the Linux kernel. However, the coding behavior itself does not explain the social dynamics of the LDC since it ensures neither the interpersonal collaborations nor the assembly of different pieces of code to fulfill a complete system function. Then, the first question is: what is the key driven force to propel the evolution of the entire LDC and the creation of the Linux kernel?

Our answer to this question is that communication may serve as such local force. It can be easily observed in the Linux-kernel mailing list that there are dense communications within the Linux developer community. According to a report by Zack Brown (cited from Kuwabara 2000), in the later 1990s the kernel traffic has reached up to six megabytes of posts in a week. In another research, Lee & Cole (2003) counted the total number of emails sent to the mailing list during 1995 to 2000. They found that 14,535 people had sent at least one e-mail to the mailing list; and each person has averagely sent 14 emails over 5 years, since there were 199,374 emails archived as of August 26, 2000. These communications cover wide-range of topics like bug-reports and -fixes, announcements of patches, discussions on technical problems, and debates about the development choices. The community members who write posts are surely the direct
communicators and the communications are the key factor to link them together to accomplish their joint efforts in developing the Linux kernel code. In addition, these communications also attract more people who do not directly participate in the discussions on a particular thread. However, they read the posts, they read and test the programs, and they often learn a lot from these discussions and their future coding actions may be influenced. In this sense, they are the indirect participants of the communications. Consequently, these communications actually connect more people than it seems; and they play a critical role in the growth of the LDC as self-organization processes that eventually impels the global development of the entire community may be activated by the local communications.

**Self-organization Processes Impelled by Local Communications**

Coordination is doubtlessly critical to the quality of Linux as well as to the growth of the community itself. Without coordination, the integrity of the Linux product would not be possible, nor could the developer community grow to be a coherent or creative one. However, as previously stated, the top-down coordination is generally absent in this community. Instead, coordination in the LDC is an emergent property generated from bottom up, where the local communications serve as the driving force. One simplest case of coordination is the task “assignment” (tasks not assigned really, as the tasks are voluntarily done). For one contributor, the good performance in his past work reinforces his reputation and trustworthiness for the quality of his other contributions. Such coordination is indeed activated by the indirect communications. The contributor transfers knowledge by submitting the patches; and other community members simultaneously get “know-who” knowledge when obtaining the technical knowledge related to the submitted patch. As a result, such indirect knowledge communication impels possible future collective and collaborative work.

More direct technical communications are also pervasive in the LDC, basically via the Linux-kernel mailing-list. Randomly picking up a thread from the mailing-list archive, we often see intensive discussions on some particular technical problem. For example, Michael Zick reported a bug-fix on May 22nd, 2009 by posting the following message:

```
*Found in the bit-rot for 32-bit, x86, Uni-processor builds:

diff --git a/arch/x86/include/asm/alternative.h b/arch/x86/include/asm/alternative.h
index f6aa18e..3c790ef 100644
--- a/arch/x86/include/asm/alternative.h
+++ b/arch/x86/include/asm/alternative.h
```

"Found in the bit-rot for 32-bit, x86, Uni-processor builds:

diff --git a/arch/x86/include/asm/alternative.h b/arch/x86/include/asm/alternative.h
index f6aa18e..3c790ef 100644
--- a/arch/x86/include/asm/alternative.h
+++ b/arch/x86/include/asm/alternative.h"
Immediately, this message causes intensive discussion. Under this thread, there were 31 postings on that same day. At least 6 persons participated. Among them, Zick posted 13 messages, Andi Klein posted 4, Ingo Molnar posted 1, Peter Anvin posted 4, Samuel Thibault posted 4, Dreier posted 2, and there are 3 other anonymous postings (probably posted by the above-listed authors too). This is a common scenario of the technical discussions in the LDC. It is not unusual that such technical discussions give rise to the emergent collaborations among the participants. Iannacci (2005) terms this kind of coordination as “partisan mutual adjustment” and argues that collaboration teams or “social networks” may appear from such communications. These social networks are open to contributions from everybody. However, they usually contain strong ties among a limited number of participants because of their frequent interactions; and a stable core may form around the strongly-tied participants. From the view of social-network-analysis (SNA), such stable cores can usually be termed “cliques”.

Besides facilitating the formation and adjustments of the collaborative teams or “cliques”, the coordination through communications also plays a critical role in the advancement of the overall Linux kernel project. Kuwabara mentioned an example of the row I/O patch, submitted by Stephen Tweedle in 1998 and rejected by Torvalds several times before the final version was last applied into the kernel. In the example, an iteratively-improved technical solution was achieved by collective work of the involved developers coordinated through communications.

From the prior description, it can be perceived that the emergent coordination is a key element that makes order grow from chaos in the LDC, as through such coordination the structuring of collaborative teams or cliques takes place and the patches with good quality are often the result of the coordinated work.
Closely related to the emergent coordination is the evolution of an implicit reputation system underlying the LDC. It is the common case that the highly-reputed or trusted persons play a central role in the communications and collaborations. They often become the primary coordinators (e.g. sub-system maintainers) in many collective efforts. However, their reputations are, in turn, the result of the performances of their previous contributions in submitting patches and in the discussions.

Kuwabara (2000) describes a feedback process for the self-reinforcement of the individual reputation, as shown in the following figure.

![Fig.2. Feedback Process of a Contributor’s Implicit Reputation in the LDC](image)

In this process, the reputation of an individual increases according to the contributions he or she makes; and in turn, the increasing of his or her reputation may foster the further contributions from the same individual. Kuwabara uses such reinforcement process to explain the motivations for a community member to contribute. We argue that this process also has influences to the evolution of the community itself. With the increasing reputation, an individual may play an increasingly-important role in the development actions; thus the evolution of the implicit reputation system is a critical factor for the formation and variation of the overall structure of the community.

From another aspect, the communications also stimulate “stigmergic” processes (Theraulaz & Bonabeau, 1999). The stigmergic processes basically function through the technical building of the Linux system. As previously cited from the project leader Linus Torvalds’ assertion, “my only long-range plan has been and still is just the very general plan of making Linux better”, the global goal-setting and long-range planning are absent in developing Linux. Moreover, the development of Linux is the collective work of massive autonomous contributors who also lack global sight of the project. With these two essential features, the development of Linux is the collective architecting without blueprint, resembling the nest-construction activities in the insect societies such as the termite colonies (Theraulaz & Bonabeau, 1995). Thus, it is natural to suppose that the stigmergic processes may exist in the Linux developer community, analogous to the collective nest-construction in a termite colony. Following this analogy, the general situation
of Linux development is that the contributors decide what to do in terms of what they have done. Therefore, no one, including the defacto project leader Linus Torvalds, knows in advance what would be added or modified in the next release; and the overall development is always open-ended. But during this open-ended process, their product is increasingly complex, increasingly powerful, and increasingly fitted to the environment (i.e. to better satisfy user needs and to cope with the technology advances outside the Linux project).

Generally, there are two sources to stimulate the stigmergic interactions. One situation is the amplification of some personal ideas and suggestions inside the community. Another situation is the adaptation to environmental or external changes, especially technological innovations outside the Linux project. Responding to the internal and the external “stimuli”, some member(s) may trigger a stigmergic process in the community, acting like a termite emits pheromones onto a site to call for other termites to continue building there. Such a stigmergic process is generally a self-reinforcement process. If one ongoing topic is active and intriguing, and if it brings many open challenges, many contributors would be attracted to this topic; and more progresses may be made, bringing even more amount of further open issues…Gradually, when the “constructing” on this topic is going to complete, the contributors may step away because there is not so much work left.

Bringing the previous discussions together, we can conclude that the self-organizing processes around the emergent coordination, the evolving reputation system and the stigmergic interactions in constructing the software system cause the global effect that the overall process of the Linux kernel development has become an adaptive process of the evolution of community intelligence. The global property of the community intelligence of the LDC will be examined in the next sub-section.

**Collective Intelligence of the LDC as Emergent Global Property**

The previously-discussed self-organizing processes results in the global process of the development of collective intelligence in the LDC. In this sub-section, we take a look at such collective intelligence at the communal level.

The “intelligence” of the LDC is firstly and most-remarkably represented by the Linux operating system itself. Today the Linux kernel has become an extremely complex software system containing over 9 million source lines of code. Building such a system looks like a mission impossible to a collectivity of part-time hackers without any formal organization, as it is difficult task even for the largest corporations. The efficiency and effectiveness in the building of the
Linux system have unquestionably proved the high “intelligence” of this online community. Kroah-Hatman et al. (2008) showed the rate of change of Linux kernel from Version 2.6.11 to 2.6.24, as depicted below.

![Fig.3. Rate of Change of Linux Kernel: Version 2.6.11 -2.6.24 (Year 2005-2008)](chart)

According to Figure 3, on average within 100 days the community announces a new stable release, which contains thousands of pieces of changes from the previous release. This figure convincingly illustrates the highly-efficient work done by the Linux developer community. For the effectiveness of the Linux contributors’ work, it is difficult to get direct quantitative measurement; but the complexity of the system on one hand, and the adoption rate\(^1\) on the other hand have illustrated the effectiveness of their development.

In addition to the external measurement of the efficiency and effectiveness of the development work, the global regularity of the LDC can also been seen internally. As the internal measure, two properties are shortly discussed here, i.e. the global structuring of the community and the emergence of the norms of code quality and coding style.

One global consequence of the previously-discussed self-organization processes is the formation and evolution of the community structure. Most remarkably, a hierarchical structure gradually forms during the evolution of the community. This structure was described by Kroah-Hatman in his 2008 presentation at Google, as shown in Figure 4.

\(^{1}\) e.g. IDC’s report for Q1 2007 says that Linux now holds 12.7% of the overall server market, source: Linux Watch, at http://www.linux-watch.com/news/NS5369154346.html
This hierarchical structure is actually a core-periphery structure. In this structure, the project leader Torvalds works closely with a limited number of lieutenants. They are core developers and subsystem maintainers and they constitute the “inner circle” of the community. The peripheral participants are the ad hoc patch-submitters and bug-reporters. Torvalds himself explained this structure in a post:

“The fact is, we've had "patch penguins" pretty much forever, and they are called subsystem maintainers. They maintain their own subsystem, ie people like David Miller (networking), Kai Germaschewski (ISDN), Greg KH (USB), Ben Collins (firewire), Al Viro (VFS), Andrew Morton (ext3), Ingo Molnar (scheduler), Jeff Garzik (network drivers) etc etc. …

A word of warning: good maintainers are hard to find. Getting more of them helps, but at some point it can actually be more useful to help the _existing_ ones. I've got about ten-twenty people I really trust, and quite frankly, the way people work is hardcoded in our DNA. Nobody "really trusts" hundreds of people. The way to make these things scale out more is to increase the network of trust not by trying to push it on me, but by making it more of a _network_, not a star-topology around me.” (Source: Linux Kernel Mailing List Archive, at: http://lkml.indiana.edu/hypermail/linux/kernel/0201.3/1070.html)

In this post, Torvalds clearly stated that the structuring of the community relies on a “network of trust”. He himself trusts a small number people in the “inner circle” or the subsystem maintainers, and each maintainer trusts a small number of other developers, and so on… This network of trust
is in fact formed under the implicit reputation system underlying the community. Correspondingly, the evolution of this reputation system generally directs the adjustment of this working structure of the community. In short, the overall structure of the Linux developer community is emergent instead of organized top-down; and the subsystem maintainers grow, instead of being officially assigned. This structure is furthermore flexible. An active contributor who performs well in his previous contributions may play an increasingly vital role in the community; by contrast, if one contributor becomes less active, he might gradually become the peripheral participants.

The third measure is the formation of the global norms in the community. With the growth of the community, the formats of submitting patches and reporting-bugs become standardized. The standards on the code quality and the coding styles are also enforced. An example given by Kuwabara (2000) is the Italian programmer Andreas Arcangeli’s patches for the printer code:

“[Arcangeli] made substantial improvements, then branched out, but tended to do some pretty sloppy things - to the point where Linus said "go away." Andreas refused to go away, and eventually had major changes to the kernel accepted. All Linus did was enforce coding standards.” (Personal Interview 1999, cited from Kuwabara 2000)

At the first glimpse, it looks like that this standardization attributes to the personal efforts of Torvalds and his close co-developers. But factually it is the result of the coordination of the entire community. Without the abundant base of contributors, this standardization is impossible; without the adoption of the standards by the contributors, this standardization is either impossible. In turn, this standardization is a means of the emergent coordination of the whole community; and the accepted norms or standards are actually become a proportion of the communal knowledge.

To sum up, in the prior analysis, a rough picture has been drawn to explain the social dynamics that underlie the LDC evolution. Generally, the LDC that develops the Linux software can somehow be analogous to a mound-building termite colony. In both cases, the building activities are accomplished by the individual agents without global planning or task-assignment; and the coordination is emergent from the “stigmergic” information exchange. The “stigmergic” information exchange in the LDC is realized by the direct and indirect communications and these communications eventually propel the overall evolution of the community. The overall evolution of the LDC is then a dualistic process in which the collaboration network of the human beings grows on one hand and the knowledge network about the Linux development expands on the
other hand. The expansion of the knowledge network is partially manifested by the improvements of the Linux product itself.

CONCLUSION

In this paper, the social dynamics underlying the LDC evolution is explored from the aspect of community intelligence. It can be concluded from this investigation that the online community of the Linux kernel developers grows through a spontaneous self-organization process. In this process, the personal activities of programming and the interpersonal communications that stimulate knowledge transfer and coordination are the most fundamental local “force” that boosts the global evolution of the community. As a result, the LDC evolves as an “intelligent” social system that effectively and efficiently “solves” a complex problem, i.e. the development of the Linux kernel. The development of this collective intelligence of the LDC is embodied as the expansion of a supernetwork of three interwoven networks, where a knowledge network is embedded within a human network supported by a media network. This analysis of the LDC may, in turn, facilitate to enrich our understandings on the development of community intelligence in general. Subsequent to the present work, more elaborate exploration of the social mechanisms that underpin community intelligence will be a key subject of future research. On the other hand, this work may have practical implications for developing computer-support systems to enhance the knowledge creation and dissemination in the online communities like the Linux kernel developer community.

ACKNOWLEDGMENT

This work is partly supported by National Natural Science Foundation of China under Grant No. 70871016. Part of this work was presented in the 6th International Conference on Knowledge Management (ICKM 2009) jointly with the 10th International Symposium on Knowledge and Systems Sciences (KSS 2009).

REFERENCES


