Innovativeness of open source software projects

Krzysztof Klincewicz

kklinc@poczta.onet.pl

School of Innovation Management
Tokyo Institute of Technology
W9-108, Tokyo, Meguro-ku, O-okayama 2-12-1, Japan

2nd affiliation: School of Management, Warsaw University, Poland

August 11, 2005

Abstract

The paper addresses an ongoing debate about the innovativeness of open source projects and critically evaluates the innovative potential of 500 most active projects registered by SourceForge.net. The analysis is based on a proposed framework, distinguishing between radical inventions, technology / platform modifications, and marketing innovations. Research findings include relatively low levels of technical newness in the studies sample, alongside high interest of developers and users in the innovative projects. The article discusses the underlying mechanisms, restricting innovativeness of community-driven open source efforts, and postulates the establishment of an institution of “idea brokers”, playing roles corresponding to venture capitalists in the commercial software domain.

1. Introduction

Imagine an inventor, who has a promising idea, which could be turned into a potentially successful software application, but this would require cooperation with experienced programmers. Can the *open source software* (*OSS*) community offer him compelling mechanisms, helping implement the innovation? If so, what are the prospects for the OSS to displace the current commercial efforts of established software companies – is it an alternative value creation mode, or is its current role restricted to commoditization of established technological designs?

Microsoft's infamous “Halloween documents” from 1998 analyzed the OSS movement, emphasizing its apparent problems with absorbing new ideas. The view was criticized by OSS activists, who were able to demonstrate numerous examples of innovative open source projects. Nowadays, Linux is an established computing environment, and nobody questions the attractiveness of many OSS applications. This seems to be the right moment to re-assess the innovativeness claims, looking at the original contributions of open source versus product imitations, identifying sources of innovative ideas and mechanisms of their promotion within the community.

The paper is an exploratory study, starting with a review of relevant existing studies and a definition of software innovations. Using the proposed innovation typology, it analyzes a large sample of OSS projects to verify, how many of them are based on truly new ideas, as well as whether innovative and imitative projects differ in certain characteristics, related to their popularity, involvement of developers and activity of users.

2. Open Source Software and innovation

The popular view of open source movement postulates that the OSS development model leads to faster incorporation of innovative ideas than the proprietary regime, pursued by commercial companies (Tuomi 2005: 434). The innovative opportunities are however not available to every open source project, and development efforts for many promising applications were suspended because of insufficient programmer contributions. Selection and mortality rates of OSS projects are higher than those of commercial initiatives, which are often maintained because of existing user bases, reputation effects, or sunk costs. Additionally, apart from community-initiated efforts, OSS model is nowadays widely used by commercial companies to fill in gaps in their technology platforms, or maintain legacy code, which has been handed over to the developer community, with both types of work characterized by a relatively low innovativeness.

The view of OSS innovation introduced in the beginning seems therefore idealistic or at least disputable – many projects end up generating yet another “me-too” product, focused on usability, support for a specific platform and commoditization of features, so far available from commercial vendors, and there are only few open source projects venturing into new, previously unexplored areas. Many researchers tend to classify every open source product as an innovation, thus using the terms “open source development” and “open source innovation” synonymously (comp. e.g. Hippel 2001; Johnson 2002; Dalle, Jullien 2003; Harhoff, Henkel, et al. 2003; Hippel, Krogh 2003; Krogh, Spaeth, et al. 2003). This terminological confusion results in a distorted image of the OSS community – the implicit assumption of the innovativeness of every OSS project leads to the use of value-bound statements in definitions of research problems, and the term “innovation” faces the risk of turning into a buzzword in the context of the software industry (Swanson 2000).

The biased interpretation of OSS innovativeness contrasts with other approaches, attempting to introduce a distinction between imitation and actual novelty – as Tuomi noticed, “there is nothing particularly innovative in projects such as Linux, which basically re-implements commercially available operating-system functionality" (Tuomi 2005: 436), and some authors prefers to enumerate certain innovative achievements of the OSS community (comp. Lerner, Tirole 2002: 201), intentionally not generalizing the term to cover all possible developments. The seminal analysis of the OSS movement - "*The Cathedral and the Bazaar*" - avoided the excessive use of the notion as well: in the final section of his essay, Raymond explicitly addressed the innovativeness question, suggesting that both the traditional development model of large corporations, as well as community-driven open source efforts cannot offer any guarantees that innovations will occur – the necessary element is always "one good idea in one person's head", usually originating outside of the observed setting, e.g. from university research or innovative ventures (Raymond 2000a).

3. Innovations in the software industry

The question of innovativeness calls for a better understanding of the term innovation, especially in the context of the software industry. Roger's seminal book on the diffusion of innovations defined an innovation based on its perceived newness to users (Rogers 2003: 12), and his followers emphasized that “as long as the idea is perceived as new to the people involved, it is an «innovation», even though it may appear to others to be an «imitation» of something that exists elsewhere” (Van de Ven 1986: 592). International statistics institutions look in turn for easier operationalizations: for example, Eurostat interprets innovations as new or significantly improved products or services, introduced to the market or within an enterprise and resulting from technology or knowledge development, emphasizing that “innovations should be new to the enterprise concerned; for product innovations they do not necessarily have to be new to the market” (Crowley 2004: 7). This definition – like many studies in the management of innovations area - dilute the Schumpeterian distinction between innovation (assumed to be the driving force of economic development) and imitation: paradoxically, products presented as innovative no longer need to be new and unique. This lack of differentiation between innovation and imitation has been identified as a general shortcoming of innovation studies, which fail to adopt Schumpeter's original dichotomy (Tanny, Derzko 1988: 227). Abernathy's and Utterback's analysis of process and product innovations emphasized the importance of incremental improvements and evolutionary as opposed to radical changes (Abernathy, Utterback 1978). Interestingly, some scholars seem to intentionally avoid the notion of innovation, using terms such as: breakthroughs, technological change or discontinuity instead (Tushman, Anderson 1986).

“New” turns out to be a broad term in marketing literature, referring to three categories of products: new to the world (with no comparable alternatives available), new to the firm, or improved (through changes to an existing product) (Kamel, Rochford, et al. 2003). Apart from this “market newness”, one can also identify technical product newness, when products are based on newly developed technologies, their new combinations, or original applications of a well-known technology (Loch 2000: 257; comp. also review of empirical studies in: Chandy, Tellis 1998: 476). Moreover, a seemingly insignificant recombination of existing technical components (*architectural innovation*) may have profound impact on the market and competence bases of industry players (Henderson, Clark 1990). Creative re-positioning of technically unchanged products (*marketing innovation*) is also an effective mechanisms for improving sales in the software industry (Klincewicz, Miyazaki 2004) – but aesthetic, usability or conceptual changes are strikingly different from the actual technology development. Innovative products encompass therefore categories as diverse as radical projects (with new technologies addressing new markets or user needs), line extensions (where established technical designs are used to serve new customer segments), and incremental projects (Loch 2000: 249) – the term “product innovation” equals therefore “product diversification”.

For OSS products, an additional challenge is the distinction between inventions and innovations – market tests of new products commercialization are not feasible for free software, while download statistics do not verify the actual diffusion (comp. the process of innovating with IT, where initial adoption decisions need to be accompanied by successful implementation and assimilation phases – Ramiller, Swanson 2003: 8-9). It should therefore be no surprise that some authors do not distinguish between OSS development and OSS innovation: with the broad understanding of innovation, even insignificant changes to products appear innovative. The fascination with innovations poses also a more general problem, identified as “*pro-innovation bias*”, an implicit assumption that an innovation should be diffused and adopted by all potential users (Rogers 2003: 106), causing particular confusion in the software and IT services markets (Klincewicz 2005).

The present study tries to return to the intellectual roots of the concept of innovation by distinguishing between original and “me-too” products, the exploration of new possibilities and the exploitation of “old certainties” (March 1991). Some authors made attempts to identify criteria differentiating radical innovations from less important changes – a notable effort by Dahlin and Behrens (2005) offered operationalized criteria and verification techniques. The authors were intentionally analyzing inventions not innovations, justifying this by ambiguous relations between market success and technological radicalness:

”impact-based” definitions of radicalness lead to a biased selection, as only commercially successful developments are considered (Dahlin, Behrens 2005: 722-724). Radicalness results from technical contents of a product, not its diffusion (Dahlin, Behrens 2005: 718), and a radical invention was defined as (Dahlin, Behrens 2005: 725):

- novel (dissimilar from previously available inventions),
- unique (diverging from current interests of other inventors),
- having an impact on future technologies (encouraging imitation).

Radical inventions come from outside of the path-dependent “local search” within well-known technological domains (Rosenkopf, Nerkar 2001). The novelty and uniqueness criteria are satisfied only if no functionally comparable products exist at the launch date of a development project.

The application of the framework to software products calls for an additional modification: software applications are developed for specific underlying platforms, and in most cases, an application built for one platform cannot easily be ported to another system. The interdependences resulting from incompatible technological standards stimulate the formation of value networks (Christensen 2000: 36-47), industry value chains (Eselius, Gard, et al. 2002), alliance constellations (Gomes-Casseres 1997), platforms (Cusumano, Gawer 2002) or ecosystems (Iansiti, Levien 2004), where individual software firms do not compete directly against supporters of incompatible platforms. A software vendor, whose products complement Windows and Microsoft SQL Server database, will not face direct competitive threats from specialist vendors, developing comparable applications for Oracle database on Unix platform. Key players, controlling platforms, assemble coalitions of partners to develop specific solutions (Cusumano, Gawer 2002). They may also delegate this responsibility to end users (Hippel, Katz 2002), e.g. by releasing interface specifications to the public, or opening source code of certain modules.

These remarks are very important for the OSS community, as many of its development efforts are focused on improving and complementing the Linux platform. Even though certain applications may exist for Windows users, they need to be “re-invented” in the other operating system environment. These re-inventions are not radical technological breakthroughs, as similar benefits and functionality are already available for alternative platforms. If their functionality is new for a specific platform, they could be regarded as "local scale radical inventions" for a specific platform. These "re-inventions" (or "platform modifications", as they will later be referred to) differ additionally from certain OSS projects focused on commoditizing functionality, which is already available for the same platform from commercial vendors. Linux users were able to use commercial databases from Oracle, Inprise, Informix and Sybase already in 1998, but the OSS community worked on entirely open and free alternatives to become independent from large companies, be able to share and reuse the database source code – the initiative could neither be classified as radical invention, nor platform modification, as the OSS projects were intended to replace available products and commoditized the functionality, bringing databases "to the masses".

	New technology	New for a platform	Existing technology
New market	**Radical invention (breakthrough)**		**Marketing innovation**
Existing market	**Technology modification**	**Platform modification**	**No innovation**

Figure 1: Typology of product innovations in the software industry

In order to capture this complexity of innovations in the software industry, a modified classification is proposed (figure 1). The dimension, characterized as technological newness, refers to previously discussed technological discontinuities and radical inventions, while the remaining variable – market newness – explains, whether the user needs in question are already addressed by other solutions. The matrix identifies the following types of software innovation:

- radical inventions (breakthroughs) – products new to the world,
- technology modifications – significant incremental improvements of established technologies,
- platform modifications – products so far available only for competing

platforms,

- marketing innovations – resulting from original positioning and new uses of existing technology solutions.

Technology and platform modifications, as well as marketing innovations, fall jointly into a broad category of “creative imitations” (Drucker 1993: 220-223): their authors do not invent entirely new products, but perfect established designs, or offer new applications for yet unaddressed user segments.

The identification of software innovation is never a straightforward task. Most products have some original features - for example redesigned user interfaces, specific installation procedures, new ways of accessing certain functionality. Newness can be interpreted as a continuum between insignificant modifications and substantial changes, original features and unique products. Decision whether an addition of specific features constitutes overall product innovativeness is therefore subjective. The development of individual innovative features in OSS projects is an important component of innovativeness, but will be omitted in the present analysis. For example, Linux graphical environment KDE integrated a spell-checker as standard system layer, available to most applications; KDE and GNOME pioneered the transparency of program windows and other useful usability improvements; Red Hat established RPM format for easy distribution of executable programs; individual Linux distributions offer various features, simplifying the system configuration. These elements do not however constitute defining characteristics of the concerned projects, remaining “bells and whistles”, minor changes of merely incremental nature. These single innovative features of software products usually can easily be copied by competitors, as opposed to complex technological developments or new uses of technology. In order to eliminate potential selection biases, the present research focuses on the importance of modifications for the entire product, perceived by people developing the project. The following classification of OSS projects was based on product positioning statements, authored by project administrators. If specific original features are important for the overall innovativeness of a project, information about them would appear in the project positioning statement (short description on project website), indicating the uniqueness.

Apart from product-related innovations, OSS movement generates also numerous innovations in the areas of software development (process innovations) and business models (organizational innovations, including unique delivery and support schemes), which are not included in the proposed framework and the present study.

4. Research methods

The present research analyzes the innovativeness of OSS projects, registered in SourceForge.net - originally founded by VA Linux Systems, the most comprehensive portal for OSS projects, providing essential project management tools for software developer communities, including shared code repositories and discussion forums. In July 2005, the portal had over 100,000 projects and over 1 million registered users, although only minority of these projects and developers were active. SourceForge implemented additional tools to help users search for projects based on their profiles, and regularly updates usage statistics, generating lists of the most active projects based on downloads, page views, and volumes of posted messages.

Due to its comprehensive coverage of projects and querying potential, SourceForge was a popular source of data for research concerning the OSS community. Some studies were based on portal crawling (Weiss 2005; OSSmole 2005) – automated retrieval of data about all registered projects for the purpose of statistical analysis. Such downloads last for at least several days, and affect the portal's performance, while large scale, all-encompassing analyzes call for substantial processing power (more information about the method in: Hahsler, Koch 2005). The major challenge for such studies is to move beyond mere data collection and basic descriptive statistics towards addressing specific research questions and hypotheses testing. An alternative approach uses theoretical sampling (Glaser, Strauss 1975: 45), with projects intentionally selected based on criteria such as: high number of bugs (Crowston, Howison 2005), or high activity (Krishnamurthy 2002). Data can be collected manually or extracted as a subset of the crawling results. The approach seems useful for exploratory, in-depth studies – but relatively small sample sizes (100-120 projects) make generalization difficult. Nevertheless, SourceForge data is ideally suited for social network analyzes (based on data about developer code contributions and replies to other people's questions), code and community growth tendencies and technical project dependencies.

The present study adopts the theoretical sampling approach to analyze a large-scale sample of 500 OSS projects. They have been selected from SourceForge list of projects with the highest activity all time (i.e. since the launch of the portal in 1999). This approach was preferred to the analysis of all SourceForge data - many registered projects are not active, and an initial analysis indicated that activity levels drop dramatically already in the second half of the sample. The research combines automated large-scale analysis and manual coding. The analysis was facilitated by a specialized "tech mining" software VantagePoint from SearchTechnologies, Inc., normally used for bibliometric studies of patents and scientific publications (Porter, Cunningham 2005).

	Population*	**Selected sample**
Number of projects	17,139 *(the remaining 81% of 89,557 registered projects not active)*	500 *(most active projects)*
% of all active projects	100%	3%
Number of developers *(estimate*)*	34,393	6,106
% of all developers	100%	18%

Table 1: SourceForge.net - comparison between the entire population and the selected sample of the most active projects (statistics for the entire population from: Weiss 2005)

**The estimated number of developers was calculated by multiplying the average number of developers per project by the number of projects – may be biased due to the parallel involvement of some developers in several projects.*

The manual coding was necessary, as an initial review of the data proved that automated application of technical radicalness criteria proposed by Dahlin and Behrens (2005) was not possible. While the criteria can be applied to patents thanks to their standardized classifications in patent databases, SourceForge project descriptions are incommensurable: administrators often use unique keywords so that two projects with almost identical functionality and technical specification are mistakenly assigned to very different topics and underlying technology platforms. Unlike automated crawling-based studies, the present research uses also qualitative data – the manual procedure involved the use of project positioning statements and additional descriptions on project web pages to classify every project within the previously presented software innovativeness framework.

Even though SourceForge is the most representative collection of OSS projects, it does not cover all relevant open source communities. Certain visible projects maintain independent developer websites – this is the case of Linux kernel, most Linux distributions, graphical user interfaces KDE and GNOME, web browser Mozilla, Internet server Apache, database MySQL, productivity software OpenOffice and many others. In addition, many highly active projects use SourceForge merely as a source code repository, redirecting all other traffic (including support requests) to own websites, what results in relatively low activity statistics registered by SourceForge (examples include popular content management platform Plone and e-learning system Moodle, not included in the 500 most active projects sample). Another potential bias refers to project registration dates, which do not necessarily correspond to actual project launches or beginnings of idea generation processes. A more general challenge for the OSS community study results from the fact that many small projects do not have the characteristics of the most successful cases (Crowston, Howison 2005) – but the present research question, concerning innovativeness, justifies a selection of the most visible ventures.

5. Research findings

Table 2 and figures 2-6 summarize descriptive statistics for the analyzed sample of open source projects. Windows and Linux are the most popular target operating systems, with C/C++, Java, PHP and Perl as the leading programming languages. Most of the projects rely on relatively small teams of developers: 9% of them are maintained by one person only, and 22% by 2-4 developers. The average number of developers per project is however 12.21, significantly more than the mean of 2.0067, computed for the entire SourceForge population (Weiss 2005: 31). As opposed to the findings of earlier studies (Krishnamurthy 2002), the most active projects do not have individualist character, and are maintained by relatively sizeable developer communities. Surprisingly, all of the analyzed projects were registered between years 1999 and 2001 – none of the projects started in the following four years gained enough popularity to join the most active group.

Year of project registration		
1999	58	11.6%
2000	351	70.2%
2001	91	18.2%

Development status		
Mature	55	11.0%
Production/Stable	317	63.4%
Beta	93	18.6%
Alpha	23	4.6%
Pre-Alpha	3	0.6%
Planning	1	0.2%
Inactive	3	0.6%

Receiving donations	204	40.8%

Intended audience*		
Developers	335	67.0%
End users	311	62.2%
System administrators	145	29.0%
Other	57	11.4%

License type*		
GNU GPL, GNU Library or LGPL	366	73.2%
BSD License	43	8.6%
MIT License	18	3.6%
Mozilla Public License	18	3.6%
Other	55	11.0%

Operating system*		
Windows	340	68.0%
Linux	332	66.4%
BSD	256	51.2%
Unix	223	44.6%
OS independent	134	26.8%
Mac	93	18.6%
BeOS	9	1.8%
PalmOS	6	1.2%
DOS	4	0.8%
IBM OS/2	4	0.8%
Windows CE	4	0.8%
Project is OS	3	0.6%
Other	17	3.4%

Programming language*		
C/C++	327	65.4%
Java	75	15.0%
PHP	53	10.6%
Perl	50	10.0%
Python	43	8.6%
Assembly	27	5.4%
Tcl	18	3.6%
Unix shell	16	3.2%
Delphi	14	2.8%
JavaScript	13	2.6%
Pascal	9	1.8%
VisualBasic/ASP	7	1.4%
Lisp	6	1.2%
C#	5	1.0%
Other	23	4.6%

Table 2: Selected statistics for 500 most active SourceForge.net projects

** Classification not exclusive (items do not add up to 100%)*

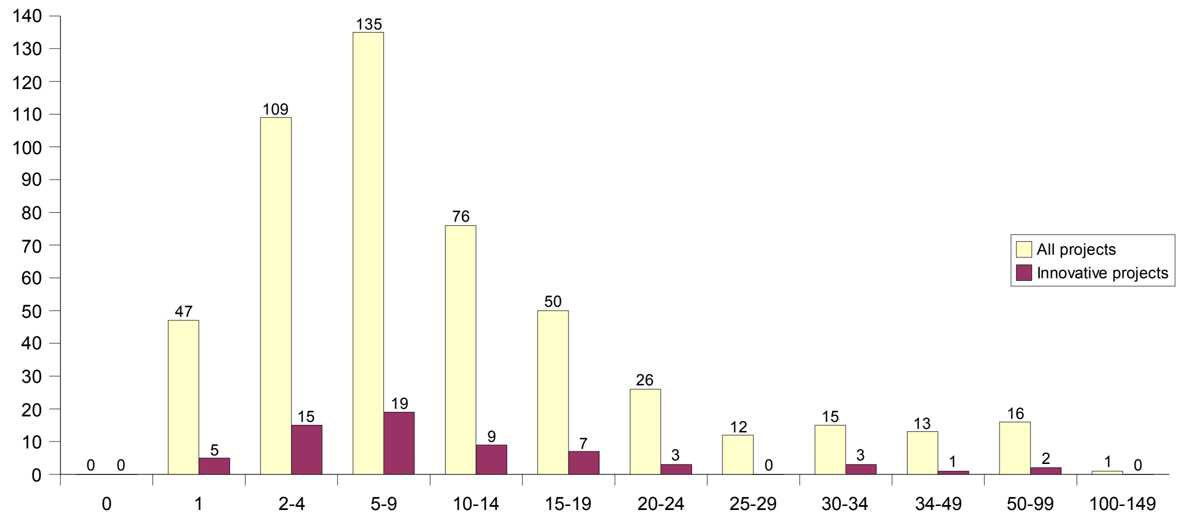

Figure 2: Distribution of the number of developers among all 500 and 64 innovative projects

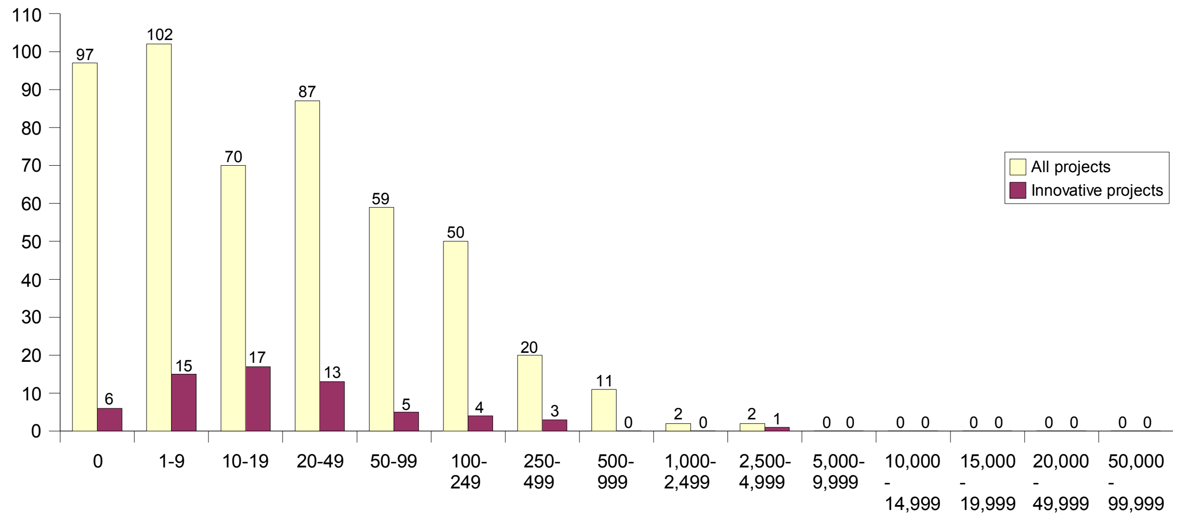

Figure 3: Distribution of new feature requests among all 500 and 64 innovative projects

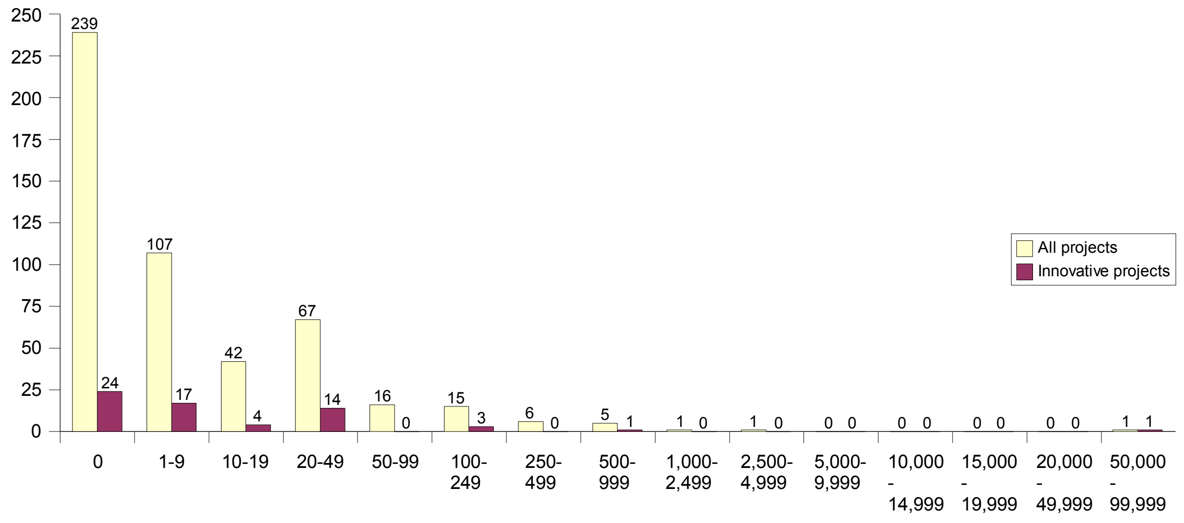

Figure 4: Distribution of support requests among all 500 and 64 innovative projects

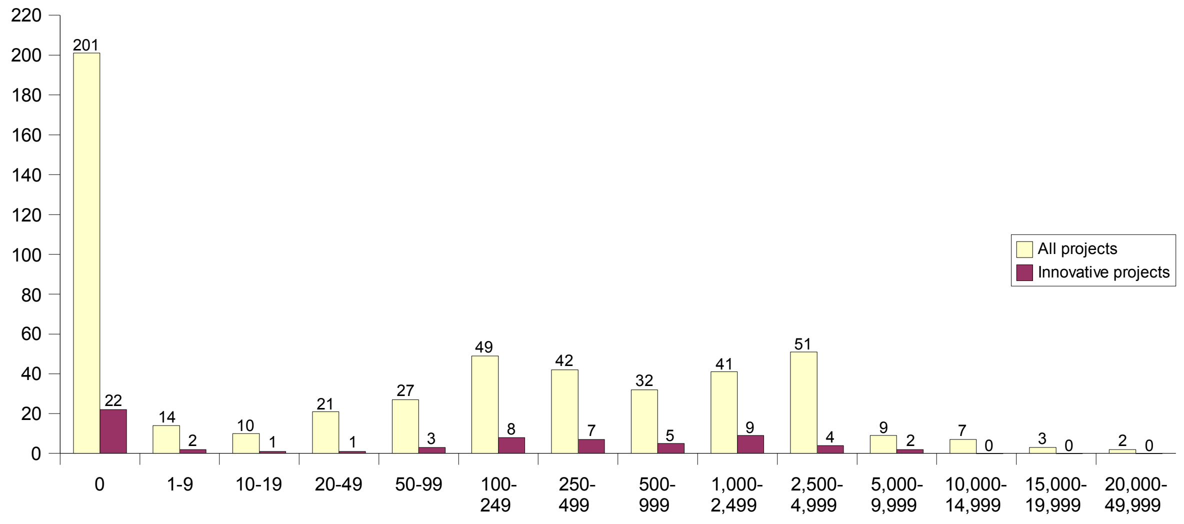

Figure 5: Distribution of forum messages among all 500 and 64 innovative projects

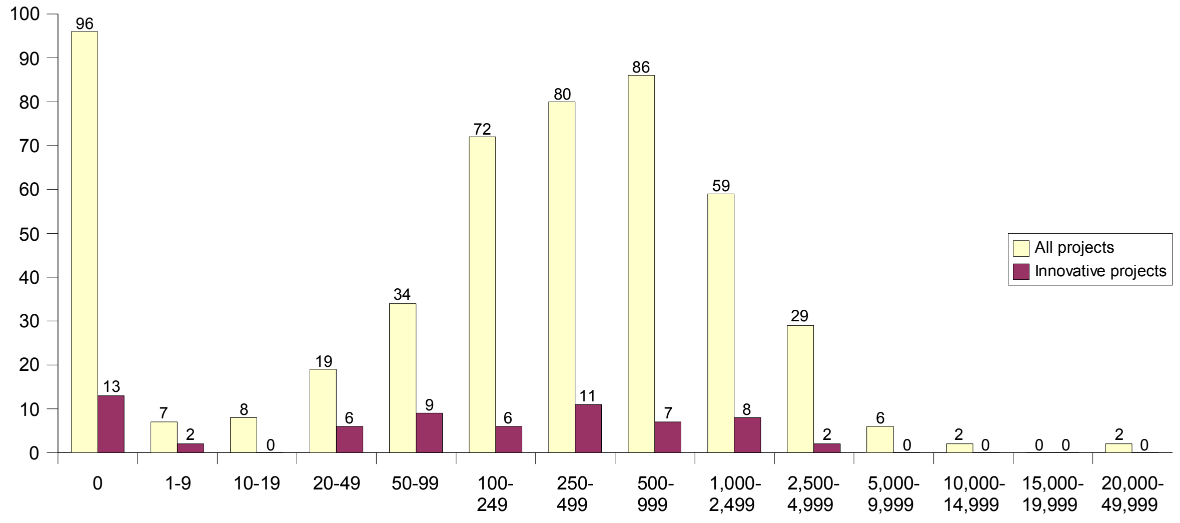

Figure 6: Distribution of the number of code additions to the Code Versioning System repositories (CVS adds) among all 500 and 64 innovative projects

	New technology	New for a platform	Existing technology
New market	**Radical invention (breakthrough) 5 (1.0%)**		**Marketing innovation 3 (0.6%)**
Existing market	**Technology modification 4 (0.8%)**	**Platform modification 52 (10.4%)**	**No innovation 436 (87.2%)**

Table 3: Types of product innovation among the 500 most active SourceForge.net projects

Table 3 presents the results of manual coding, which divided the 500 projects into distinctive groups based on the degree of newness. Only 64 projects (12.8%) were not direct imitations of existing solutions, and **Appendix A** presents their detailed list. Descriptions of the 436 remaining projects, classified as non-innovative, frequently use words such as "*similar*", "*one of*", "*enhancement of/based on* [another software]", "*yet another*", "*replacement*". Redundancy is a serious problem: "yet another" application offering similar functionality creates unnecessary competition for scarce resources within the OSS developer community, reducing their effective utilization. The observed multiplicity of comparable projects results often from *code base forking* (cases when some developers do not accept the directions of a project, and use its code to launch competing efforts - Weber 2004: 64), or ambitions to implement something better than others, instead of helping improve existing projects. In the OSS world, mergers and acquisitions do not occur, and joining forces by members of competing projects is a rare phenomenon – whereas among commercial software companies, similar product lines from several vendors may eventually merge into one system as a result of alliances, take-overs or technology licensing agreements.

Decision making processes in many OSS projects are highly formalized, what additionally discourages new concepts. It is easier to fork the code, than convince project decision makers to implement certain ideas. An example could be Samba TNG (The Next Generation), forked from Samba (file sharing and printer server for Linux environment, able to communicate with Windows computers). When innovative architectural suggestions were opposed by Samba project leaders, their proponent had no other choice than to launch an independent project in 1999 (Weber 2004: 169). After 6 years of development, Samba TNG remains a niche project, not included in known Linux distributions nor supported by commercial vendors. Decision making processes in open source projects can also be as complex as in established software companies, contrary to the assumptions of spontaneous "bazaar"-like organization. Apache uses e-mail voting system to make any significant decision (Fielding 1999: 42-43). Mozilla divides project code into modules, delegating responsibility for each module to a specific individual (Mockus, Fielding, et al. 2002: 332). OpenOffice has formal procedures for submitting proposals, involving voting, 6 months "probation period" and differentiation between "incubator" and "accepted" statuses, moreover any proponent needs to be already known to the community as a reliable contributor (OpenOffice 2005). The formalization may stifle innovation and prevent original ideas from being shared within existing projects. A possible solution addressing this concern is a *plug-in* architecture, enabling third-parties to easily complement existing platforms, and creation of portals to promote their applications. The openness of technology, facilitating development of complementary add-ins (Hippel, Katz 2002) is obvious for OSS projects (open by definition), but less obvious is the promotion of third-party software or offering additional tools to encourage developers. GNOME project maintains GnomeFiles, searchable Internet repository of compatible software, and Mozilla Foundation hosts a dedicated website, listing extensions for its products. These open source projects adopt partner management methods, traditionally used by established firms such as Microsoft, understanding the critical role of partner companies in the innovativeness and diffusion of a platform.

The findings reveal low overall share of unique OSS projects - as table 3 indicates, only 5 out of 500 SourceForge projects could be classified as technological breakthroughs, 4 were technology modifications, 3 - marketing innovations, and the remaining 52 projects were platform modifications, focused on implementing functionality new to a specific platform, but already available for other, incompatible systems. Among platform modifications, Linux is the leading environment (contrary to the high Windows' popularity, registered for the entire sample), and projects in this group port new functionality, popular among Windows users, or improve support for specific standards and hardware (e.g. USB, PMCIA, NTFS, IEEE 1394). Such projects are characterized by a “self-service” mode of development: end users “tweak the hardware support” for their own systems, what resembles the concept of *prosumption*, where consumers assume responsibility for the production of consumed goods (Toffler 1990). In spite of the disappointing results of the survey, truly innovative OSS projects exist, but many of them were implemented by individual programmers, and not listed by SourceForge. In the OSS community, there seem to be limited prospects of value co-production (Ramirez 1999), division of labor between inventors and implementors: if you have a new idea, or need specific complementary (and yet unavailable) functionality, you need to implement it by yourself. Pioneering new concepts is more difficult than implementing and commoditizing proven designs, developed for other platforms by commercial companies.

Project type	**No. of projects**	**Average no. of developers**	**Average no. of new feature requests**	**Average no. of support requests**	**Average no. of forum messages**
All	500	12.21	79.47	207.84	902.13
All *(excluding SourceForge.net)*	499	12.19	74.08	32.56	895.56
Non-innovative projects	431	12.30	77.98	33.42	921.86
Radical inventions	5	21.40	555.80	17542.20	1317.00
Radical inventions *(excluding SourceForge.net)*	4	85.00	10.00	39.00	2403.00
Technology modifications	4	38.50	236.25	0.00	0.00
Platform modifications	52	8.79	35.83	31.65	795.23
Marketing innovations	3	7.67	28.67	1.67	228.00

Table 4: Statistics for different types of projects

Table 4 presents comparative statistics for various types of projects. Innovative projects are significantly more popular among developers than “me-too” solutions. Development of underlying technology platforms attracts more attention than radical breakthroughs, benefits of which may sometimes be difficult to understand. The comparison of the numbers of active developers, feature and support requests supports Eric Raymond's reflections about the OSS developer community, where individual prestige seems to be an important factor, influencing the project involvement decisions: “One [...] gains more from projects that are strikingly innovative, as opposed to being 'me, too' incremental improvements on software that already exists. On the other hand, software that nobody but the author understands or has a need for is a non-starter in the reputation game, and it's often easier to attract good notice by contributing to an existing project than it is to get people to notice a new one” (Raymond 2000b).

Breakthrough projects tend to be more difficult to understand and adopt for potential users, what explains the high number of messages in discussion forums and support questions, alongside less new feature requests. The tendency is distorted by the SourceForge.net project metrics (development of the underlying SourceForge software is also an OSS project): continuous development of the portal is critical for thousands of its users, resulting in unproportionally high activity – table 4 includes therefore additional statistics, excluding this particular project.

Stimulation of innovation is an additional challenge for the OSS community. Follow-up content analysis of the documentation of the identified innovative projects revealed (table 5) that 40% of breakthroughs came from company-initiated projects, and 50% of technology modifications grew out of academic research, while community-driven initiatives were in turn more focused on platform modifications and marketing innovations.

Project initiated by:	Radical inventions	Technology modifications	Platform modifications	Marketing innovations	All innovative projects
Companies	2 (40.0%)	0	5 (9.5%)	0	7 (10.9%)
Academia	0	2 (50.0%)	1 (2.0%)	0	3 (4.7%)
Community	3 (60.0%)	2 (50.0%)	46 (88.5%)	3 (100.0%)	54 (84.4%)

Table 5: Initiators of innovative projects

New functionality arrives often as "*feature gifts*", original code developed by individuals or companies **outside** the OSS community and contributed upon joining a project (Krogh, Spaeth, et al. 2003: 1233) – but such inventions are not really generated by the OSS development model itself. Similarly, source code opening means that a commercial company decides to share its innovations and let the community maintain and modify them. Many companies use OSS licensing model and developer communities, while pursuing parallel commercialization strategies: examples include Linux vendor Red Hat, CRM provider SugarCRM, database company MySQL AB, or grid storage architecture pioneer HiSpread. Some companies do not even offer free products, or require users to pay for additional functionality - and their innovativeness is linked to clear financial motives, not present in typical "bazaar"-like OSS projects. One should therefore not confuse primarily commercial efforts, where open source licensing is incidentally used, with truly community-driven developments: they differ in technology control modes, decision making processes about development directions, and motivations to innovate.

"Real” open source projects often originate from university research or hobbies of individual programmers – like Linus Thorvalds' hobbyist development of Linux, creation of the first version of Linux image manipulation software GIMP as a term project at University of Berkeley (GIMP 1998), research of University of Cambridge, resulting in virtualization architecture XenSource, or creation of free data mining software Weka by the University of Waikato. SourceForge lists hundreds of applications, developed by university researchers as byproducts of their studies. Many of them implement novel ideas, but at the same time have limited practical uses outside narrowly defined research fields. Consequently, they have to be maintained by people from one research institution, and rank very low on the activity scale. In such cases, the application of the open source licensing does not need to be combined with a real community-driven development model, as the actual research is financed from other sources, and driven by objectives not related to the OSS community. This parallels the previously described case of commercial companies, incidentally opening code of parts of own applications, while pursuing independent objectives. Many ideas coming from academics are interesting, and could be inspiring for other developers – but sadly, they tend not to be, as there seem to be no “technology transfer” mechanisms, helping promote own ideas across the developer community. An example could be MusicMiner project, maintained by Databionics Research Group, University of Marburg. The software implements an algorithm to cluster files in music collections and search for similarities. If comparable research was conducted by companies such as Apple or Microsoft, the findings would probably quickly find their way into upcoming releases of media players as “intelligent” recommendations for users, enabling them to automatically create playlists of aurally similar songs. As an open source initiative, MusicMiner remains unnoticed, with very low activity levels registered by SourceForge and no integration with other multimedia-oriented projects. Open source licensing model was adopted by the software developers, but the project does not actually benefit from being part of the OSS community – to make things worse, the research progress and software development would probably be more impressive, if the university researchers decided to continue the project for proprietary technology companies. The case demonstrates the perils of open source: lack of effective project promotion mechanisms may stifle innovations and discourage inventors.

6. Discussion

Nobody keeps track of innovative open source projects, abandoned by their founders, who were not able to find sufficient support from other developers. A well-documented case is Freenet, which was miraculously revived, and is included among the most active technological breakthrough in the present research. The product idea was proposed by Ian Clarke in his master thesis, and published on Freshmeat.net – a website, listing new launches and significant changes to open source projects. Clarke established code repository on SourceForge, but failed to attract and motivate other programmers, and decided to leave the project by the end of 1999 (Krogh, Spaeth, et al. 2003: 1221). Later other people managed to turn the idea of Freenet into software – but numerous other concepts are less successful, and original ideas usually need to be implemented by the proponent herself. Lack of diffusion prospects for innovations may discourage also their generation, and pioneers are inclined to look for alternative ways to implement their ideas. The analyzed statistics of OSS projects indicated relatively low levels of innovativeness, with multiple "me-too" products. Large, established technology companies can afford the luxury of limited innovativeness (Christensen 2000), acquiring new ventures to extend own product portfolios – in contrast, community movement can only generate new value by own developments, or by convincing commercial vendors to open code of of their existing applications and donate "feature gifts".

Even though the role of first mover has certain drawbacks, such as high development costs and elevated risk, technological systems cannot function without a steady inflow of new concepts, and the open source community needs to innovate to maintain its relative technological self-sufficiency. Proprietary and open software are currently interdependent – technological change requires new ideas, which usually are outcomes of costly and time-consuming research, calling for money and dedication. It seems much easier to assemble a group of active developers to "fight with a joint enemy", Microsoft (Weber 2004: 139): projects concerning critical improvements of Linux platform are visible and popular. One could however observe an interesting paradox in the recent developments, best exemplified by Mono, an open source project intended to port Microsoft .Net technology to non-Microsoft operating system platforms. In the years 1994-99, Microsoft invested substantial funds into partner companies and third-party projects, intended to diffuse its COM architecture and establish it as a standard by offering integration kits for various computing platforms and tools for converting applications. 10 years later, with its next generation technology platform, .Net, Microsoft does not need to resort to such payments, as it can rely on the work of OSS community. The orientation of the OSS developers towards replicating established designs can actually lead to externalities, benefiting companies not involved in the movement (or even hostile to it). Mono is a very promising initiative, expected to significantly reduce the software development cycle for various platforms – but if most OSS projects focus merely on porting to Linux functionality, already available from commercial vendors, the role of open source will be restricted to commoditization, which stimulates the generation of new ideas by the affected companies. The unintended effect of OSS development is the intensification of activities of proprietary technology firms, focused on innovation and creation of customer lock-ins (West 2003: 1279) – one could even suggest, that these powerful commoditization mechanisms stimulate more innovation in proprietary than in open software domains.

The above discussion should not lead to a conclusion, that the open source movement is entirely stripped of innovative capabilities. There are truly innovative projects initiated by the OSS community, but the actual problem is not the inflow of ideas, inputs to new product development. It is the ability to promote new ideas, gain support from other community members and stimulate the diffusion of new applications. While OSS developers have excellent tools for software engineering, technical support groups and mechanisms, stimulating code re-use to shorten development cycles, they lack efficient project promotion frameworks. Community portal Freshmeat.net was founded to endorse new software-related ideas – but nowadays, it resembles SourceForge (established and owned by the same company), listing new releases of software alongside project search engine and some specialist articles. Commercial software developers can resort to venture capitalists to receive funding, as well as marketing and networking support. The OSS community desperately needs corresponding “idea brokers”, helping launch innovative initiatives, not necessarily fitting into existing mainstream developments. Due to the nature of open source movement, funding is not as important as an independent evaluation of concepts, business consultancy (including advices on user needs, project positioning and potential partnerships with other projects), and promotion of promising projects to create distributed developer communities. This role could be taken by a dedicated open source foundation, actively seeking sponsors for these activities. Alternatively, one of large IT companies, showing its commitment to the OSS community by opening often low-end, obsolete technologies and expecting to have their code maintained and improved, could instead demonstrate support for truly innovative and open initiatives.

Appendix A. List of the identified innovative OSS projects

	Acti-vity ran-king	Project name	Description
Radical inventions	1	SourceForge.net	Portal for open source developers with dedicated software engineering tools
	30	The Freenet Project	Software surpassing the restrictions of Internet censorship
	286	Winfingerprint	Security tool for analyzing (or hacking) Windows networks
	461	Virtual Linux	Portable Linux distribution, running directly from CD-ROM drive without installation
	487	OpenCyc	Knowledge base and commonsense reasoning engine
Technology mod.	10	Python	Scripting language
	33	Tcl	Scripting language
	103	Dscaler Deinterlacer/ Scaler	Software converting PC into a professional line doubler/scaler
	451	curl and libcurl	Tool and library for transferring data via Internet
Platform modifications	8	AFPL Ghostscript	Interpreter and converter of PostScript and PDF
	9	Linux PCMCIA Card Services	Support for PCMCIA hardware in Linux platform
	12	Exult	Implementation of Ultima7 game engine for modern operating systems
	13	Numerical Python	Numerical extensions to Python language
	23	Netatalk	Unix tool for sharing printers with Apple Macintosh computers
	45	PhpWiki	Wiki (web encyclopedia) clone using PHP language
	87	Jetty	HTTP Servlet Server written in Java
	88	Bastille-Linux	Hardening and reporting/auditing program, strengthening Linux security
	94	AMaVis – A Mail Virus Scanner	Interface between mail programs and virus scanners
	95	Zapping	TV viewer for GNOME graphical environment
	102	CppUnit	Junit testing for C++
	119	Linux-USB Project	Support for USB hardware in Linux platform
	137	Nunit.Net	Junit testing for .Net languages
	151	Linux NTFS file system support	Support for NTFS disk file format in Linux platform
	154	Windrop	Windows version of Eggdrop IRC
	157	Python/XML	XML language support in Python
	177	Jython	Java implementation of Python
	184	Basilisk II	Mac II emulator
	187	IEEE 1394 for Linux	Support for IEE 1394 (i.Link) hardware in Linux platform
	193	Art of Illusion	3D modelling, rendering and animation studio
	194	RIMPS	Software converting Apache web server into MP3 server
	200	LIRC	Support for infrared remote controller in Linux platform

	206	QuickCam Express Driver	Linux driver for Logitec QuickCam webcams
	218	PyOpenGL	Binding layer between Python and OpenGL
	221	NFS	Linux Network File System
	238	Eteria IRC Client	Java-based IRC client
	240	KDE on Cygwin	Emulator enabling running Linux-KDE applications on Windows
	247	OpenSLP	Implementation of Service Location Protocol
	251	LSTP	Linux terminal server environment
	253	MySQL for Python	Support for Python in MySQL database
	261	Trinux	Floppy disk distribution of Linux for monitoring TCP/IP networks
	280	pam-mysql	Integration between MySQL database and Pluggable Authentication Modules
	288	TkCVS	Tcl/Tk based graphical interface to Code Versioning System
	303	rdesktop	Client for Windows Terminal Services
	312	CGI:IRC	Program for using IRC from web browser
	315	Prosper	Support for slides in LaTeX
	326	HW+/DXR3 driver for Linux	Linux drivers for DVD playback cards
	346	Snoopy	PHP class simulating web browser
	379	phpESP (php Easy Survey Package)	Package for creating web surveys using PHP and MySQL
	384	Exuberant Ctags	Multilanguage implementation of Unix ctags utility
	391	leJOS	Replacement firmware for Lego Mindstorms RCX bricks to build Java-programmed robots
	396	Software Process Dashboard	Software for collection and analysis of software development process metrics
	408	FreeMind	Java-based mind mapping software
	417	MM.MySQL	Java driver for MySQL database
	425	Small Linux Project	Utilities to run Linux on machines with small memory
	429	TiLP – Ti Linking Program	Software transferring data between Texas Instruments scientific calculators and PCs
	432	AdvanceMAME	Port of arcade game consoles MAME and MESS for PCs
	435	Linux driver for CpiA webcams	Support for CpiA webcams in Linux platform
	452	Linux Hotplug	Support for plug-and-play hardware in Linux platform
	489	Xamba Network Integration Project	Porting Linux Samba package to Mac OS X
	491	WinLIRC	Windows implementation of Linux infrared remote control software
	495	MDB Tools	Utilities reading and exporting Microsoft Access database files
Marketing innov.	347	Tunez	Music jukebox with network users votes influencing the choices
	350	Mars Simulation Project	Simulation of future human settlement on Mars
	356	Astro Info	Astronomical almanac for PalmOS

Literature

Abernathy, W.J., Utterback, J.M. (1978) Patterns of Industrial Innovation. *MIT Technology Review*, 80, 7, pp. 40-47

Chandy, R.K., Tellis, G.J. (1998) Organizing for Radical Product Innovation: The Overlooked Role of Willingness to Cannibalize. *Journal of Marketing Research*, 35, 4, pp. 474-487

Christensen, C. (2000) *The Innovator's Dilemma*. HarperBusiness, New York

Crowley, P. (2004) Sources and resources for EU innovation. *Eurostat, Statistics in Focus, Science and Technology,* 9-5

Crowston, K., Howison, J. (2005) The social structure of free and open source development. *First Monday*, 10, 2, *www.firstmonday.org/issues/issue10_2/crowston/index.html*

Cusumano, M.A., Gawer, A. (2002) The Elements of Platform Leadership. *MIT Sloan Review*, 43, 3, pp. 51-58

Dahlin, K.B., Behrens, D.M. (2005) When is an invention really radical? Defining and measuring technological radicalness. *Research Policy*, 34, pp. 717-737

Dalle, J.M., Jullien, N. (2003) ‘Libre’ software: turning fads into institutions? *Research Policy*, 32, pp. 1–11

Drucker, P. (1993) *Innovation and Entrepreneurship*. HarperBusiness, New York

Eselius J.L., Garg, D., Hugher, G., Kelly, J., et al. (2002) The Wireless Value Chain and Infrastructure. [in:] Gulati, R., Sawhney, M., Paoni, A. (eds.) *Kellogg on Technology & Innovation*. John Wiley & Sons, Inc. Hoboken, New Jersey, pp. 79-154

Fielding, R.T. (1999) Shared Leadership in the Apache Project. *Communications of the ACM*, 42, 4, pp. 42-43

GIMP (1998) *A Brief History of GIMP, www.gimp.org/about/ancient_history.html*

Glaser B.G., Strauss A.L., (1975) *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing, Chicago

Gomes-Casseres, B. (1997) *The Alliance Revolution. The New Shape of Business Rivalry*. Harvard University Press, Cambridge, Massachusetts

Hahsler, M., Koch, S. (2005) Discussion of a Large-Scale Open Source Data Collection Methodology. *Proceedings of the 38th Hawaii International Conference on System Sciences*, *csdl2.computer.org/comp/proceedings/hicss/2005/2268/07/22680197b.pdf*

Harhoff, D., Henkel, J., Hippel, E. von (2003) Profiting from voluntary information spillovers: how users benefit by freely revealing their innovations. *Research Policy*, 32, pp. 1753–1769

Henderson, R.M., Clark, K.B. (1990) Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. *Administrative Science Quarterly*, 35, pp. 9-30

Hippel, E. von (2001) Innovation by User Communities: Learning from Open-Source Software. *MIT Sloan Review*, 42, 4, pp. 82-86

Hippel, E. von, Katz, R. (2002) Shifting Innovation to Users via Toolkits. *Management Science*, 48, 7, pp. 821-833

Hippel, E. von, Krogh, G. von (2003) “Private-Collective” Innovation Model: Issues for Organization Science. *Organization Science*, 14, 2, pp. 209-223

Iansiti, M., Levien, R. (2004) Strategy as Ecology. *Harvard Business Review*, 82, 3, pp. 69-78

Johnson, J.P. (2002) Open Source Software: Private Provision of a Public Good. *Journal of Economics & Management Strategy*, 11, 4, pp. 637–662

Kamel, M., Rochford, L., Wotruba, T.R. (2003) How New Product Introductions Affect Sales Management Strategy: The Impact of Type of “Newness” of the New Product. *The Journal of Product Innovation Management*, 20, pp. 270-283

Klincewicz, K., Miyazaki, K. (2004) Dilemma in Innovation. The Case of Product Innovations versus Marketing Innovations in the Software Industry. *The Japan Society for Science Policy and Research Management Yearbook*, Tokyo, pp. 107-110

Klincewicz, K. (2005) *Management fashions. Turning Bestselling Ideas into Objects and Institutions*. Transaction Publishers, New Jersey

Krishnamurthy, S. (2002) *Cave or Community? Empirical Examination of 100 Mature Open Source Projects*. Working Paper, University of Washington, *opensource.mit.edu/papers/krishnamurthy.pdf*

Krogh, G. von, Spaeth, S., Lakhani, K.R. (2003) Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32, pp. 1217-1241

Lerner, J., Tirole, J. (2002) Some Simple Economics of Open Source. *The Journal of Industrial Economics*, 50, 2, pp. 197-234

Loch, Ch. (2000) Tailoring Product Development to Strategy: Case of a European Technology Manufacturer. *European Management Journal*, 18, 3, pp. 246-258

March, J.G. (1991) Exploration and Exploitation in Organizational Learning. *Organization Science*, 2, 1, pp. 71-87

Mockus, A., Fielding, R.T., Herbsleb, J.D. (2002) Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11, 3, pp. 1-38

OpenOffice (2005) *Protocols for Project Proposal*, *www.openoffice.org/about_us/protocols_proposing.html*

OSSmole (2005) *Ossmole Project*, *ossmole.sourceforge.net*

Porter, A., Cunningham, S.W. (2005) *Tech Mining. Exploiting New Technologies for Competitive Advantage*. Wiley-Interscience, Hoboken, New Jersey

Ramiller, N.C., Swanson, E.B. (2003) *Whether, When, and How to Innovate with Information Technology: What Do Empirical Studies Tell Us?* Information Systems Working Papers, The Anderson School at UCLA, Los Angeles, 2-03

Ramirez, R. (1999), Value Co-production: Intellectual Origins and Implications for Practice and Research. *Strategic Management Journal*, 20, 1, pp. 49-65

Raymond, E.S. (2000a) *The Cathedral and the Bazaar*, www.carborg/~esr/writings/cathedral-bazaar/cathedral-bazaar

Raymond, E.S. (2000b) *Homesteading the Noosphere*, www.catb.org/~esr/writings/cathedral-bazaar/homesteading

Rogers, E.M. (2003) *Diffusion of innovations*. Free Press, New York

Rosenkopf, L., Nerkar, A. (2001) Beyond Local Search: Boundary-spanning, Exploration, and Impact in the Optical Disk Industry. *Strategic Management Journal*, 22, pp. 287-306

Swanson E. B., 2000, *Information Systems as Buzz*. Information Systems Working Papers, The Anderson School at UCLA, Los Angeles, 1-00

Tanny, S.M., Derzko, N.A. (1988) Innovators and Imitators in Innovation Diffusion Modelling. *Journal of Forecasting*, 7, pp. 225-234

Toffler, A. (1990) *The Third Wave*. Bantam Book, New York

Tuomi, I. (2005) The Future of Open Source, [in:] Wynants, M., Cornelis, J (2005) *How Open is the Future?* VUB Brussels University Press, Brussels, pp. 429-459

Tushman, M.L., Anderson, P. (1986) Technological Discontinuities and Organizational Environments. *Administrative Science Quarterly*, 31, pp. 439-465

Van de Ven, A.H. (1986) Central Problems in the Management of Innovation. *Management Science*, 32, 5, pp. 590-607

Weber, S. (2004) *The Success of Open Source*. Harvard University Press, Cambridge, Massachusetts

Weiss, D. (2005) *A Large Crawl and Quantitative Analysis of Open Source Projects Hosted on SourceForge*. Research Report of the Institute of Computing Science, Poznań University of Technology, Poland, RA-001/05, www.cs.put.poznan.pl/dweiss/site/publications/download/weiss-2005-large-crawl-of-sourceforge.pdf

West, J. (2003) How open is open enough? Melding proprietary and open source platform strategies. *Research Policy*, 32, pp. 1259-1285

Access date for all quoted Internet pages: August 10, 2005

Krzysztof Klincewicz is assistant professor at the School of Management, Warsaw University and researcher at the School of Innovation Management, Tokyo Institute of Technology. His research interests combine strategic management, organization theory and new technologies, with particular focus on strategies of high-tech companies. Dr Klincewicz is certified chartered marketer of the British Chartered Institute of Marketing, and spent numerous years managing business development for IT companies in Poland, Finland and in the UK.