# OPEN SOURCE PROGRAMMER'S STRATEGIES TO COPE WITH IDEOLOGICAL TENSIONS

Bertil Rolandsson[a], Magnus Bergquist[b] and Jan Ljungberg[b, 1]

[a] University of Borås
[b] University of Gothenburg

**Abstract**
In this paper we analyse how the increased use of open source software in companies affect employed programmers' work, which we theorize as part of a larger secularisation process. We have studied both companies based on a more traditional proprietary model who are becoming open source oriented, and SMEs built around open source business concepts. This change results in a need for professional programmers to re-interpret open source within a new business oriented context. We study what kind of strategies programmers develop to cope with these contradictory systems and how it changes work roles and programmers' approaches towards open source community work.

Keywords: open source, programmers, strategies, secularisation

## 1. INTRODUCTION

Open Source software has within a few years changed large parts of the global software industry by enabling radically new business models, organizational forms and commons based platforms as a new foundation for value creation. From being almost entirely a community issue, open source software has entered companies and public organizations as an alternative or complement to proprietary software. This broader use of open source software, methods and practices has resulted in that more programmers employed by IT-companies, are confronted with open source in their work. Sometimes this is a result of companies adopting an open development model, as a parallel to their traditional proprietary model of production (Fitzgerald, 2006), and sometimes it is the result of a more ad hoc practice of problem solving in various development projects. Regardless of what strategy companies choose, programmers face a new situation where they have to relate to the different norm systems interacting with each other: the more ideological open source community norm system and the norm system of a more traditional software development and business organisation model.

Most research about this development in software development has focused on firms and their concern for how company and community interests can be combined (Fitzgerald, 2006; Ven & Verelst, 2008; Dahlander & Magnusson, 2005). Few studies have addressed what this development means for the individual programmer. Studies on programmers have mainly focused on motivational issues and questions concerning why programmers choose to engage in community-based open source software development (Bates et. al 2002; Goosh et al., 2002; Hars & Ou, 2002; Hertel et al., 2003). In this paper we bring the discussion further by elaborating on professional employed programmers and their strategy for dealing with open source initiatives. We have studied how employed programmers, who are confronted with open source in their professional practice, cope with what will be referred to as *secular tensions*. This means

[1] Corresponding author: Jan Ljungberg, Department of Applied Information Technology, IT-university 412 96, University of Gothenburg, Sweden. Phone: +46(0)31-7862746. Fax +46(0)31-772 4899. Email: jan.ljungberg@ituniv.se

that we analyse how basic assumptions, ideological values, forms of organization and practices, associated with the open source movement, is challenged while professional programmers try to re-interpret them within a new business oriented context.

Our main questions guiding us through this analysis are:

1. How do programmers that face various demands at work, originating from the tensions between proprietary and open approaches, reinterpret the values that are associated with the open-source movement?

2. What kind of strategies do they use to cope with the ideological tensions that appear in their professional practice, when open source code becomes a component in their software development?

## 2. FROM FREE SOFTWARE COMMUNITY TO OPEN SOURCE FIRM

From being a community issue, were hackers are producing code for other hackers, open source software has entered companies and public organizations as an alternative to proprietary software. The emergence of open source communities has created important resources that companies can utilize, but rarely control. The development from a free software and open source movement to a situation where communities co-exist with firms that to different degrees engage in the communities provides a latent set of tensions emanating from the different interests and value bases that could be found in firms and communities. These tensions may be viewed as part of the employed open source programmers' everyday practices.

### 2.1 The Development of Free Software and Open Source

The roots of the free software and open source phenomenon go back to the hacker culture of the early sixties (Levy, 1984), and the communities related to the Unix operating system and the C programming language. Here programming skill, and the sharing of solutions and code were highly valued. These core values were further facilitated in the formation of the free software movement in the early eighties, with initiatives like the GNU project (Stallman, 1984; 1985), the Free Software definition (GNU Bulletin, 1986), and the Free Software Foundation. The GNU General Public License (GPL) was constructed to ensure the recipients of computer programs the rights of the free software definition, and that they are preserved even if the program is modified (i.e. an inscription of the free software definition in copyright law). This could encourage developers to contribute, since they may be confident that their work benefits the whole world and remains free, rather than being exploited by software companies that may not give anything back to the community.

Some developers found the ideological nature of FSF too zealous, i.e. viewing free software as an end in itself (Raymond, 1999). In order to avoid these connotations, the term open source was coined, indicating that open source is viewed mainly as a means to an end of producing software of high quality. The Open Source Initiative (OSI) was founded to support the new focus on technology rather than ideology. This more pragmatic nature of the movement, downplayed the most ideological parts of the value system, but also contributed in the diffusion of free and open source software. The movement grew substantially, and included both large traditional software companies (e.g. IBM, HP, Sun) and small companies that were founded on an open source business model (e.g. Red Hat, Mandrake). New more permissive licenses compared to GPL (e.g.

BSD - Berkley Software Distribution, Välimäki, 2005), were used to make it easier for open source and proprietary software to coexist. This diffusion of open source to a commercial context has been described as a transformation of open source software into a more mainstream and viable form (Fitzgerald, 2006).

Despite the differences, the open source and free software communities share many of the core values. They place a high value on freedom of speech, regarding programs as communal resources, free information sharing as a right, an ideal and a practical strategy (e.g. See Himanen, 2001; Szczepanska, Bergquist and Ljungberg, 2003). Other values that are held high are helping others so they may solve new problems instead of readdressing old ones; technical knowledge and skill, learning for its own sake (Raymond, 2003; Stewart & Gosain, 2006); voluntary cooperation (Stallman, 1992) and reputation. The status, fame and reputation that a contribution could give are understood as one of the main motivational forces within the developer-communities (e.g. see Raymond, 1997; Bergquist and Ljungberg, 2001; Bergquist, 2003; Zeitlyn, 2003).

### 2.2 The Relation between Open Source Communities and Firms

The rationale for open source firms and open source communities to engage in open source software is different (Dahlander & Magnusson, 2005). Firms are utmost driven by maximizing profits, while open source communities are driven by producing public goods. The motivations for firms to engage in open source communities are mainly of economic and technical nature, and do not correlate to the social motivations of individual programmers (Bonaccorsi & Rossi, 2006). Utilizing open source means to capture value created by others, while contributing back to the community is to take part in the value creation and cultivation of the community. The communities thus could be seen as resources possible to utilize by companies. This opens up for potential tensions in the relation between companies and communities. Dahlander & Magnusson (2005) describe three kinds of relationships between communities and companies: i) a parasitic relation were the firm will not give back but try to avoid direct conflicts; ii) a symbiotic relation were both firm and community will gain; and iii) a commensalistic relation in using the resources while keeping the involvement in developing the resources to a minimum without obviously braking the norms.

While a large portion of recent open source research has focused on how company and community interest can be combined (Fitzgerald, 2006; Ven & Verelst, 2008; Dahlander & Magnusson, 2005; O'Mahony, 2003), few studies have addressed what this development means for the individual employed programmer. Most studies on individual programmers have focused on motivational issues concerning why programmers choose to voluntarily engage in community-based open source software development (Bates et al., 2002; Goosh et al., 2002; Hars & Ou, 2002; Hertel et al, 2003). In this paper we focus on the professional programmers who deal with open source in their daily work practice, trying to make up strategies to handle the ideological tensions that may occur while working in between these different value systems.

## 3. SECULARIZATION AND TENSIONS – A THEORETICAL FRAMEWORK

We will analyse how ideological values and practices, associated with the open source movement, is challenged while programmers try to re-interpret them within a new business oriented context. More precise, professional open source programmers are understood as gradually secularized. This means that we touch upon a classic theme that usually refers to the shrinking impact of religion in modern society. Traditional values

and communities are then questioned due to a rationalization process, in which individuals are organized and more aware of possible consequences of their actions by the help of formal functions and routines. In accordance with Weber, we may describe it as a process in which work continuously gets more marked by bureaucracies, demanding that we engage less in faith and more in calculations of what type of technology we need in order to achieve planned goals (Weber, 1978). Ideals of instrumental expertise gains strength in connection to an increased emphasis on economical achievements and our capability to trade products and services on markets.

Weber stress that such bureaucratic hierarchies and markets give rise to an emphasis on impersonal relations, which, if they become a goal in themselves create indifference to shared values and value-related goals (Tambiah, 1990:12). An increased emphasis on formal hierarchies of work, knowledge and economy, may mean that we become instrumental specialists without spirit and heart, indifferent to collective values (Weber, 1978). Now, we could claim that our knowledge and how we value it, under such circumstance also will be constantly under scrutiny. We will keep on testing our standpoints in a pragmatic way, which according to Habermas also means that we instead of just devaluing our ideals will focus upon them. That is, we will face a process in which we reinterpret our value-based perspectives, so they fit with a rational view on life. Instead of a one-way process towards what could be described as a technocratic perception of work, secularization then means that we experience tensions between instrumental and value-based beliefs (Habermas, 2002:16).

In this paper, similar tensions are in focus within the context of open source. Secularization is understood as a tensed process in which programmers struggle with open source norms, values and practices while trying to re-interpret them within a new business oriented context. It is of course impossible to set a start date for when a "pure" non-secularized state of open source can be found (or if it ever existed). Instead we will describe this process as a gradual reformation and reinterpretation of basic values, forms of organization and practices, associated with what is recognized as open source. We ask ourselves: what type of strategies do programmers use while trying to cope with the ideological tensions that appear in their professional practice, when open source code becomes a component in their software development?

To answer this question we will look at how the programmers describe the internal organization of software development; e.g. does their mode of work in any sense remind us of the bureaucracy Weber sketched (Weber, 1978). Likewise, we will look at how these programmers refer to external relations, e.g. concerning economical achievements and how market relations are associated with their work as programmers. These are aspects that to a certain extent remind us of concepts like hierarchies and markets, frequently pointed out as ideal typical components in rationally calculated modern organization of work (Ouchi, 1980; Tilly, 1999). In addition, the open source movement is associated with communities resembling a third ideal-typical organizational component described as an alternative to hierarchies and markets, namely networks (Podolny & Page, 1998). Networks are said to consist of relations that are personal, egalitarian and marked by informal power relations. Instead of formal bureaucrats, network members are described as engaged in cultivating and controlling their community activities together according to less formalized goals and values (cf. Tilly, 1999; Podolny & Page, 1998; Douglas & Wildavsky, 1983:138f). In our case, such a *cultivating* strategy could cause ideological tensions between collective values of openness and bureaucratic or market demands on instrumental proceedings of software development. However, we may also find programmers *utilizing* open source code

without being directly involved with open source communities and the ideological tensions that may appear.

## 4. METHOD

We have done qualitative interviews with 30 programmers working at two types of companies: traditional large proprietary software firms that gradually have incorporated open source software (twelve interviews) and so called pure-play firms that are formed around an open source business model, typically small entrepreneurial service oriented consultancy firms (eighteen interviews). Six of the interviews were made with female programmers professionally engaged in open source software development (from both kind of companies). When talking about ”programmer” we refer to a wide definition, including designers, coders, system developers, software engineers, software architects and to some extent project leaders.

Tams-analyzer was used to code the empirical data. Alternating between theory and content generated detailed codes that visualised how the relations between programmers and the open source movement were supposed to be understood (Ragin, 2000). Codes that were both theoretically anchored in concepts like hierarchies, markets and networks, and empirically grounded in our interviews, were in this way distinguished and used in the analysis (e.g. content of work, pragmatism, openness) (Miles & Huberman, 1994:271).

## 5. RESULTS

The coded data was developed into four distinct categories that will be presented in this section. Our results indicate that the programmers face different demands on being pragmatic, associated with different ways of reinterpreting their view upon open-source software development. Sharing was important for all interviewed programmers. They appreciated the possibility to download and study the source code, to report bugs and to be active in forums. Still differences coincided with whether the tensions were expressed by programmers from companies with a history in proprietary software that gradually have moved to include some open source code, or whether they came from SMEs that built their business models entirely on open source software. There were also distinct differences between SMEs that only made use of open source code and those that were in charge of or had a close collaboration with a community. Finally, we traced two different strategies for reinterpreting open source ideology within this new business context among the firms that had close collaboration with a community. In the following sections we will more thoroughly describe the content of these strategies.

### 5.1 Values of Openness, Technological Quality and Juridical Constrains

Programmers working at companies that had had a traditional proprietary approach to software development but gradually had included some open source software, described openness and access to the source code as something desirable. They saw themselves as researchers in a lab exploring new technological possibilities. In accordance with such values, they appreciated technological expertise as well as the idea of sharing expertise and code with each other. Against these pro-openness values stood a judicial procedure based on the company’s need to protect intellectual property rights, which sometimes was directly hindering openness. A tension identified related to openness and contradictions between the inside of the company and the outside with references to the open source community. An innovation that could lead to a business opportunity

resulted in a patent. When this was the case nothing could leak outside the organization. A GPL license would jeopardize such business opportunities. Most of the interviewed programmers were well aware of license problems with e.g. GPL and organisational constrains at work for sharing code with communities. This problem was regulated by a juridical document that had to be filled in by every programmer who was considering using an open alternative in the system. The document was cleared by the patent department, and if no risk was identified, the programmer could choose the open alternative..

The engineers' main strategy to manage different tensions and demands was to contribute to a cutting edge technological development. Access to open source software and documentation made out an important resource in this development. By using open source code they as programmers got access to an environment where different software components were not just developed in a rather intensive pace, but also discussed and tested by a huge amount of people. It can be claimed that they saw their work as contributions to a wider technological development, which everyone would profit from in the long run, but not primarily by sharing code. The programmers never contributed to any community based software development. The reason put forward was that their potential contributions were so special that they would be useless for other programmers. By using open source code, they could as professional programmers identify themselves with something good, but they were mainly interested in problem-solving capabilities and to deliver high quality solutions.

Within this rather pragmatic approach to open source, we can sum up some of the ideological tensions. Generally the respondents wanted to be god programmers and supported an improved sharing of and access to the source code. But, such sharing of source code had to be strategically constrained by the company, which sometimes made the programmers complain about formal, slow and bureaucratic procedures of work. Many of them also expected the patent department to be reluctant to their use of open source code, due to fear of being sued for not following demands of licenses demanding access to the source code. That is, even if the programmers appeared to be focused upon solving problems in their work rather than dealing with the ideological problems of the open source movement, they had to relate to juridical tensions between what could and what could not become open source in the software development of their companies.

### 5.2 Recognising both Values of Openness and Customer Demands

Moving to SMEs that built their business models entirely on open source software a wider variety of tensions was found, due to the fact that the companies can be diversified based on their relationship to the open source movement and community driven software development. A common approach among such SMEs was to be relatively instrumental when it came to providing code to communities. They promoted the use of open source and often wanted to associate themselves with the principles behind the movement, but without necessarily being an active part in open source communities. The companies used available open source projects as a raw material for their offers to customers. The idea was to create services that include open source systems and applications, or part of applications, where the company added value to the project and built a business model around this.

Many of the programmers had a history as engaged open source contributors or advanced users, and were in some cases still active. Working for an open source oriented SME was a way to turn a previously private interest into a professional. These

programmers regularly expressed a direct relationship to the open source movement and it's ideology. In some cases, they had applied for their present job because they were attracted by the company's open source profile. Some of the programmers also participated in open source communities on a private basis. For these programmers, with an ideological background in the movement, a tension was created between a purist approach - pro open source in its most non-compromising form - and the kind of demands that were put on the company as an actor in a service and customer oriented competitive branch.

Compared to the respondents who worked at companies with a history in proprietary software that gradually had included open source software, these programmers were not so focused upon internal conditions or technological problem solving in their work groups. They rather tackled their ideological tensions by emphasising an identity that related to customer needs. Their work was conditioned by their capacity to come up with tailored software solutions and high quality support. For them the advantage with open solutions was described as improved opportunities to work closer with customers. It was seen as an affordable and efficient mode of producing high quality software together with their clients. Instead of starting a new project by writing lengthy specifications, open source made it easier to co-operate with the customer based on prototypes and iterations.

In this way, ideological issues of sharing, free access to source code and quality issues related to specific open source software development methods became intertwined with a client and business oriented rationality. By being able to use freely available open source code in a commercially driven customer oriented project, the developers would not primarily share code, but improve their co-operation with the customer and thereby focus directly on generating value from open source. In this context, absence of proprietary claims was seen as a reason to why trust could much easier be established between the company and the customers. As advocates of open source code they did not have to defend the solutions as had it been an in-house developed product. Hence, it provided them with opportunities for a more objective and critical role, and thereby achieved both higher quality in their software solutions and an honest face towards the customer. Furthermore, several respondents pointed to the fact that an open source environment would secure that the application or system could live on no matter what happened with the company.

Summing up, we can conclude that the programmers described an ideological position based on having an honest and pragmatic relation both to the world of community based open source and to the customer, and that this relation also appeared to be crucial for how they made sense of their work with open source code in a commercially driven company. In accordance, some of them claimed that one of the most important advantages with open source programming was that they could avoid being a salesperson and focus on advocating benefit and usefulness for the customer, which they saw as compatible with the more ideologically driven open source development where many of the interviewees had their background. Even if sharing was seen as something positive, and licenses demanding free access (such as GPL) were preferred, they declared that they also could choose proprietary alternatives if the customer opted for it.

### 5.3 Coping with Market Solutions and Community Contributions

In our study, we also found programmers who worked at SMEs that managed or actively took part in open source communities. These programmers considered themselves as highly capable of influencing the open source software development they engaged in. They saw it as a part of their job to contribute with solutions as well as to follow open source licenses, norms and values. The respondents described themselves as persons devoted to their work, with difficulties to separate their private interests from their professional role. In some cases they also had founded and remained project owners of the community on which they based their open source business.

Contrary to other companies where open source was seen as an opportunity to tailor software in accordance with customer demands outside the communities, these programmers stressed the importance of looking primarily at the development going on within their community and argued that the customers had to adapt their needs to the community agenda. This, it was argued, was best for the customers in the long run since it provided them with the best software quality.

Tensions reoccurred among programmers who worked at companies that were in charge of or had close collaboration with an open source community. When it came to actual contributions of code, programmers who collaborated closely with a community also had two different and distinct ways of understanding tensions between interests of the company and of the community, coinciding with how the company organized their engagement in their respective communities. This difference can be illustrated by referring to three different companies that had been formed around a community, functioning as project owners, and to which they made considerable contributions. These companies represent diametrical opposite positions versus open source.

The first example is a company that was formed around an open source initiative. Since the product was targeting a small specialised and globally diverse customer segment, open source became a business strategy to create a critical mass around the product, which was free to download and use. However, if the users wanted to contribute or in other ways affect the development of the product, they (preferably companies) paid a membership fee in order to get the opportunity to contribute with code. Besides having the possibility to affect the development of the product in favour of own interests, the customer could establish a good reputation through the community as a leading innovator. This community had attracted large multinational companies. The roles had been diversified in both contributors and their customers. The company that owned the project acted as a host and facilitated the different activities and interests from other members. The customers were part of the community. The programmers who worked at these companies had an ideological interest in open source, mainly concerning the spreading of code, and saw no major problem in the arrangement. They saw themselves as engaged in a community willing to share code, but also wanting to make money out of their engagement. They were well aware of the importance of choosing the right license so that viral effects of GPL was avoided, which implies that they tried to handle certain tensions between company interests and an ideology of openness.

This should be compared with the explicit ideological tensions found among respondents working in two other firms that were owners of communities organised around personal networks consisting of programmers. For these programmers the primary concern was to engage in a movement contributing to a wider change in society. Developing code in a community was for them a matter of taking part in a political project, in which it was important to show that they could both volunteer to

offer competence in the struggle for something good as well as profiting from doing that. They wanted to demonstrate that it was possible to combine both ideological and instrumental interests associated with taking part in a community based open source development. However, this ambition generated a tension between business and common good.

In general the tension between company and community interests became more pronounced among programmers working in companies engaged in network-based communities. In some cases, this tension also appeared to be associated with ideas that potentially could be recognised as somewhat contradicting. For instance, there were programmers who described both how they struggled to prove that investments of money and time in community based open source development were paying off, as well as how important it was for them to recognise the development going on in the community before looking closer at specific customer demands. Contrary to the programmers working at the firm engaged in a company based community that also included some of the customers, they described the customer as an external actor who had to be convinced about subordinating their interests in relation to the community development agenda. One of the respondents described how he repeatedly had to inform and establish agreements with customers to guarantee that their final work would become contributions to the community.

## 6. DISCUSSION

As argued before we use the term secularisation to focus on the general process by which open source code, development, values and norms are integrated into contexts that previously have not been based on open code and the community spirit of openness and collaboration that is recognised as characterising open source software development. From the perspective of open source programmers, secularisation then takes the form of different tensions between (more or less conscious) contradicting systems of belief.

If we analyze how programmers coped with ideological tensions created while they developed open source software within a business context, we can conclude that they expressed different strategies depending on differences in internal organization and external relations, as well as in what type of relationship with the open source community they referred to. These strategies do not necessarily exclude each other; firms may host a combination of them. However, they emanate from tensions created either in companies with a history in proprietary software that gradually have included open source code, or within SMEs that built its business models entirely on open source software. We have also found differences between SMEs that made use of open source code and those firms that had a close collaboration with a community. In addition, there were differences among firms that had close collaboration with a community.

In the following sections we will more closely analyze the external and internal organizational relations, and how these shape the use of community resources; i.e. open source code. Based on this we elaborate on what kind of strategies developers create. We will distinguish four different strategies (described in table 1).

| | | | |
|---|---|---|---|
| **Firm** | External relations | *Tailors on the market* | *Cultivating the community market* |
| | Internal organization | *Hierarchic utilizers* | *Cultivating the community and enlightening the customer* |
| | | Utilizing community resources | Cultivating community resources |
| | **Community** | | |

Table 1. Programmers' strategies in relation to organizational structure in firms versus community.

### 6.1 Hierarchic utilizers

The lower left quadrant in the table above mirrors the data from programmers who were employed by companies that used to have a traditional and proprietary approach to software development. They appeared to be skilled, had a developed interest in technology, and believed that open source could improve technical quality. However, the programmer freedom to make use of it was constrained by the company's internal organization and juridical considerations crucial while defining the programmer's technical needs (cf. Ouchi, 1980). As a consequence, they developed a strategy that concerned on the one hand achieving the possibility to make autonomous design choices based on technical preferences, and on the other hand coping with the internal structure of the company and it's juridical constraints. Their strategies concerned opportunities to fulfil personal needs and goals in the process of innovating their own work in relation to locally defined peers (at work), which also meant that they utilized resources provided by the community, rather than cultivating open source communities. Most of the programmers did recognise open source ideals as admirable, but even without open source they would see themselves as problem solving experts contributing to a common good technological development. Open source provided them with high quality code, but they were not devoted to the idea of openness as such. Any environment offering high quality code and similar technical benefits would support their strategies.

The programmers made use of open source code because it suited their demands on handling predefined problems and tasks, not because they wanted to change the world. Neither did they consider customer demands or to contribute back to open source communities. They were utilizers aware of their functions within a rather formalized hierarchy (cf. Podolny & Page, 1998; Tilly, 1999). In addition, they knew that they only shared code indirectly, by the help of a consultancy firm acting as a buffer towards external demands on the source code, and that potential clashes between claims on open access and their companies business model, involved formal procedures of the patent department. The programmers referred to FOSS-documents used for deciding what open source components could and could not be used. In this way, they distanced themselves from ideological demands embodied in e.g. licenses. But, the formal and hierarchically organised procedures also reminded them of ideological issues of openness and free access to the source code (cf. Luanne, 2002).

### 6.2 Tailors on the market

The upper left quadrant contains strategies among programmers working in SMEs that build their business models entirely on open source software. These programmers did not refer to patent departments or FOSS-documents. A more extrovert approach could be found, in which they were less occupied with problems distributed to them within a formal and hierarchic organization. Reminding us of Raymond (1998), we might say that they were parts of a bazaar. Instead of relying on in-house procedures for defining tasks and controlling code, they engaged in constant negotiations with the customers about the content and quality of the code, and had a pragmatic view upon choices of licenses that were more or less exclusive to proprietary code. The business idea for the company harmonized with the strategies of the programmer who had a clear customer focus. In comparison with programmers who worked at companies dominated by proprietary approaches, they were externally oriented consultants tailoring software and business propositions close to the customer, on an informal and individual basis.

Several of them made ideological statements e.g. about freedom of information as a common good (Castells, 1996). Nevertheless, for these firms open source was an important strategic asset in order to compete on the market, and the programmers seemed to have made this approach their own strategy. But, there was a tension between open source ideals and commercial demands, which they handled by claiming that the customer became less dependent on one supplier of software. They justified their use of open source by referring to the importance of sharing and how they by the help of open source could achieve trustworthy customer relations; i.e. they expressed a market strategy (Podolny & Page, 1998), based on close relationships with customers. If we consider the programmers' relation to open source communities, their strategies also remind us of the programmers employed by companies with a traditional proprietary approach. They mainly viewed open source code as a resource to utilize; open source was perceived as a high quality input helping them to solve problems defined by customers, rather than as an environment to cultivate in its own right.

### 6.3 Cultivating the community market

Turning to the right column in the table we find two strategies articulated by the programmers working for SMEs that actively took part in open source communities. In both cases content and quality of the code is something that was negotiated and guaranteed by the community developing software. Contrary to former cases where programmers utilized and certified the quality of open source code within an internal hierarchy, or as a strategic advantage to communicate to customers, these programmers were actively cultivating open source arenas by sharing and contributing back to their community (Raymond, 1999; Bergquist & Ljungberg 2001; Zeitlyn, 2003). However, there were differences among these programmers as well. For instance, in the upper right quadrant the programmers expressed strategies taming ideological tensions between commercial and business interest, by building an exclusive community of developers and customers who paid a fee enabling them to contribute to the software development. The community in this case became a mean for marketing and attracting customers to take part in the software development process. Even if the programmers can be described as cultivators of an open source community, this strategy thereby included market logic and a clear customer focus. In accordance, they preferred licenses facilitating combinations of open source and proprietary code. Their strategy could be described as a matter of cultivating an open consortium or a community market (Ouchi, 1980; Podolny & Page, 1998).

### 6.4 Cultivating the community and enlightening the customer

Finally in the lower right quadrant we find programmers expressing a strategy based on the expertise of the community. This strategy was generally founded on conventional open source ideology and ideas of freedom of information (O'Mahony, 2003; Castells, 1996), and that this focus would guarantee a quality of code that everyone profited from in the long run. In order to achieve something that is a common good, demands of specific customers had to be secondary. In accordance, tensions between community and business interests were handled by prioritising the community and it's capability to develop high quality software that as many as possible would profit from in the long term (including the firm). Within this strategy a hierarchy was created where the community overarched the customer. Cultivating the community was the ultimate goal, and business and customers were means to achieve this. Therefore, the quality of open source software should first of all be guaranteed by the review process and the informal hierarchy that existed within the open source community, and then be used to solve customer problems. This was also a community strategy in which personal networks of developers played a key role (Podolny & Page, 1998). However, the priority of the community meant that the programmers had to persuade customers that they would achieve more if they saw software development as a long-term project owned by them. The programmers had to convince the customer to trust the expertise within their open source community.

## 7. CONCLUSIONS

We have used the term secularisation to describe how open source values and norms are questioned due to demands of an instrumental and business minded approach; conventional open source ideology is assumed to have a shrinking impact (Fitzgerald, 2006). Following a Weberian logic, programmers will then engage less in ideological concerns about what is good or bad software development, and more in calculations of what type of technology they need in order to achieve preferably economical goals (Weber, 1978). Ideals of instrumental expertise will gain strength in connection to an increased emphasis on hierarchically organised capabilities, and on trading products and services on the market (cf. Podolny & Page, 1998).

In accordance, we may conclude that our respondents referred to open source code as a resource that could be utilized in an instrumental manner. We may then also confirm previous research stressing that the development process is becoming less bazaar like as strategic planning becomes paramount (Fitzgerald, 2006; Ven & Verelst 2008). Even if the programmers working for SMEs producing tailor-made software could be part of a bazaar, they also illustrated a secular approach supporting claims that focus in companies starting to profit from community based software development easily change from developing products together with open source communities to selling it to customers (Dahlander & Magnusson, 2005). Thus, our findings support the identification of secularisation in how open source software development is understood and practiced by programmers in the studied companies.

However, this secularisation appears to be a tensed process. Individual programmers still struggled with open source norms, values and practices, while trying to re-interpret them within a new business oriented context. Many of the programmers were testing ideological standpoints in a rather pragmatic way; instead of just devaluing their ideals they focused upon open source ideology (cf. Habermas 2002:16). They also appreciated the freedom to download and study the source code, to report bugs and to be active in forums, asking and answering questions (Raymond, 1999; Bergquist and Ljungberg,

2001; Bergquist, 2003). Some of the programmers cultivated conventional open source communities or what can be described as community markets. We may also point out that to the programmers working for companies that had a proprietary approach to software, open source ideology cannot be described as vanishing. In their case it was something new, and even if they had a distant relationship to ideological standpoints on contribution, freedom of information, common good etc., the administrative procedures constraining their use of open source reminded them of open source norms and values.

Hence, instead of a one-way secularisation process towards a technocratic perception of software development, the programmers dealt with tensions between instrumental and value-based standpoints. It can also be stressed that by handling these tensions the programmers developed new ways of organising work with software development. The practices of software development, which were identified and termed "hierarchic utilizers", "market tailors", "market cultivators" and "community cultivators", illustrate innovative ways of coordinating work with open source code in a company context (cf. Bonaccorsi & Rossi, 2006).

By saying that, we point out once again that much previous research has focused open strategies on an organizational level in firms, and research about individuals has focused motivation for contributing to communities. In this paper we show how the relationship between individual and organizational arrangements affect how professional programmers within a business context engage in developing open source, and how a secular approach to open source can be understood as motivating and innovative for the individual programmer.

**REFERENCES**

Bates, J., Di Bona C., Lakhani, K.and Wolf, B. (2002) *The Boston Consulting Group Hacker Survey.*

Bergquist, M. (2003) 'Open Source Software Development as Gift Culture: Work and Identity Formation in an Internet Community', in Garsten, C. & Wolff, H. (eds.), *New Technologies at Work. People, Screens and Social Virtuality*. Oxford: Berg Publishers.

Bergquist, M. and Ljungberg, J. (2001) The power of gifts: organizing social relationships in Open Source communities. *Information Systems Journal*, 11 (4), 305-320 pp.

Bonaccorsi A. and Rossi C. (2006) "Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement: From Community to Business", *Knowledge, Technology & Policy*, Winter 2006, vol. 18 nr. 4, pp. 40-64.

Dahlander, L. & Magnusson, M.G. (2005), "Relationships between open source software companies and communities: Observations from Nordic firms", *Research Policy*, vol. 34 nr. 4, pp.481-493.

Douglas, M. & Wildavsky, A. (1983) *Risk and Culture – An Essay on the Selection of Technological and Environmental Dangers*, Berkeley, CA: University of California Press.

Fitzgerald, B. (2006), "The Transformation of open source software", M*IS-Quarterly*, vol. 30 no. 3, pp. 587-598.

*GNU's Bulletin*, Volume 1 Number 1, page 8, February 1986. Available at http://www.gnu.org/bulletins/bull1.txt.

Ghosh R., Glott R., Krieger B., Robles G. (2002) *Survey of Developers. Free/Libre and Open Source Software: Survey and Study*. FLOSS. Final Report. Intemational Institute of lnfonomics. Berlecom Research GmbH.

Habermas, J. (2002) *Religion and Rationality - Essays on Reason, God and Modernity*, Cambridge, Oxford: Polity Press.

Hars A.. and Ou S. (2002) "Working for free? Motivations for participating in Open Source projects". *Internatiotial Journal of Electroiiic Commerce* vol. 6,pp. 25-39.

Hertel G.. Niedner S.. Herman S. (2003) "Motivation of software developers in Open Source projects:an Internet based survey". *Research Policy*, vol. 32nr.7.

Himanen, P. (2001) *The Hacker Ethic and the Spirit of the Information Age*. London: Secker and Warburg.

Klein, H.K. & Myers M.D (1999) "A set of principles for conducting and evaluating interpretive field studies in informations systems", *MIS-Quarterly*, vol. 23 no. 1, pp. 67-94.

Levy, S. (1984) *Hackers: heroes of the computer revolution*. Harmondsworth, Middlesex: Penguin Books.

Miles, M. B., & Huberman, M. A. (1994). *Qualitative data analysis: An expanded sourcebook* (2nd ed.). Thousand Oaks: Sage.

O'Mahony S. (2003) "Guarding the commons: how community managed software projects protect their work". *Research Policy*, vol. 32nr.7, pp. 1179-1198.

Ouchi, W.G. (1980) "Markets, Bureaucracies, and Clans", *Administrative Science Quarterly*, vol. 25, pp 129-141.

Podolny J.L & Page K.L. (1998), "Network forms of organization", *Annual Review of Sociology*, vol. 24, pp. 57-76.

Ragin. C. C. (2000). *Fuzzy-set social science*. Chicago: University of Chicago Press.

Raymond, E.S. (1999) T*he Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates, Sebastopol, CA.

Raymond, E. S. (ed.). "The Jargon File," December 29, 2003. Available online at http://www.catb.orgijargon/; version 4.4.7.

Stallman, R. (1985) "The GNU Manifesto", *Dr. Dobb's Journal of Software Tools*. vol. 10 nr. 3.

Stallman, R. (1992) "Why Software Should Be Free,"' April 24,1992 Available online at http://www.gnu.org/philosophy/shouldbefree.html.

Stewart,K. J. and Gosain, S. (2006) “The Impact of Ideology on Effectiveness in Open Source Software Development Teams”, *MIS Quarterly*, vol. 30 no. 2, pp. 291-314.

Szczepanska, A.M., Bergquist, M. and Ljungberg ‚J. (2005) ‘High Noon at OS Corral - Duels and Shoot-Outs in Open Source Discourse’, in Feller, J., Fitzgerald, B., Hissam,S. A. and Lakhani, K. R. (eds.), *Perspectives on Free and Open Source Software*, Cambridge, Mass.: MIT Press.

Tambiah, S.J. (1990), *Magic, Science, Religion, and the Scope of Rationality*, Cambridge: Cambridge University Press.

Tilly, C. (1999) *Durable inequality*, Los Angeles, London: University of California Press.

Välimäki, M. (2005) *The Rise of Open Source Licensing: A Challenge to the Use of Intellectual Property in the Software Industry*, Helsinki: Turre Publishing.

Weber, M. (1922/1978), *Economy and Society*. London: University of California Press.

Ven, K. & Verelst, J. (2008), "The impact of ideology on the organizational adoption of open source software", *Journal of Database Management*, vol. 19 nr. 2, pp. 58-72.

Zeitlyn, D. (2003) “Gift economies in the development of open source software: anthropological reflections”. *Research Policy*, vol. 32 nr. 7, pp. 1287-1291.