Chapter 2
Opensourcing

In this chapter we discuss and illustrate opensourcing with case studies at IONA Technologies, Philips Healthcare and Telefonica. The chapter draws on the study reported by Ågerfalk and Fitzgerald (2008) and uses the set of company and community cues derived in that study (in the original publication, these were referred to as obligations). In the study, we asked both company and community interviewees to discuss their perceptions of their own obligations, and also the obligations they would expect from each other. It quickly emerged that many of these obligations were symmetrical. The study represents an important step towards both (a) elaborating the software sourcing research agenda to incorporate also this novel and unconventional approach to global sourcing and co-opetition, and (b) bringing the under-explored area of company-led open source projects onto the open source research agenda, in particular the liberation of hitherto proprietary software. As noted above, most research on outsourcing has adopted a single perspective: the customer or the supplier (but most often focusing on the customer), while this study considered both the customer (in this case, the company) and the community obligations. This is important since although the cues are symmetrical to a large extent, they are also complementary, and there are differing emphases from each perspective, both sides of which must be fulfilled to achieve a successful opensourcing arrangement. The study also identifies the significant ways in which opensourcing differs from conventional outsourcing—the lack of a formal contract and requirements specification driven by the customer as well as the absence of payment in the conventional sense, for example.

Seeking both theoretical and practical implications (Ågerfalk 2015), the study followed a mixed methods research design (Ågerfalk 2013; Venkatesh et al. 2013) in which qualitative interviews in three case companies were followed by a survey across several opensourcing projects. Further details on the research design can be found in an earlier publication on this study (Ågerfalk and Fitzgerald 2008).

The chapter continues as follows. Sect. 2.1 provides some background to the projects studied. Sects. 2.2 and 2.3 presents the company and community cues that

P.J. Ågerfalk et al., *Software Sourcing in the Age of Open*, SpringerBriefs in Computer Science, DOI 10.1007/978-3-319-17266-8_2

were identified and Sect. 2.4 elaborates on differences in perception of obligations between company and community. Finally, Sect. 2.5 summarizes the identified opensourcing cues and Sect. 2.6 elaborates on implications for practice.

2.1 Background to Case Study Projects

IONA Technologies—The Celtix Project: IONA Technologies, at the time of the study a NASDAQ-quoted company headquartered in Dublin, Ireland, with U.S. headquarters in Waltham, Massachusetts and offices worldwide. IONA was founded as a campus company at Trinity College Dublin in 1991, and provided products and services to help organizations build B2B enterprise portals. IONA was rated as the leading provider of standards-based platform middleware technology, with more than 4,500 blue-chip enterprise customers worldwide, who used IONA products to address large, complex application integration and achieve interoperability by means of a standards-based, service-oriented architecture. In June 2005, Iona extended its business model to incorporate open source by leading a community project to develop Celtix, an open source Java Enterprise Service Bus that was to co-exist with Artix, the company's flagship integration product. The Celtix project was hosted by an established open source community, ObjectWeb, who specialized in developing open source middleware products. Most of ObjectWeb's members were small to medium-sized enterprises based in continental Europe. The Celtix project achieved an impressive development productivity schedule, proceeding through four significant development milestones, a beta release to a fully stable 1.0 release in just over 10 months.

Philips Healthcare—The DVTk Project: Philips Healthcare (at the time of the study, Philips Medical Systems) specializes in medical devices and is part of the global technology organization Philips, headquartered in Eindhoven, The Netherlands. DICOM (Digital Imaging and Communication in Medicine), initiated in 1993, is a global standard for storing and processing medical images and is used in virtually all hospitals worldwide. The DVTk product provides extra functionality for problem solving and improving the quality of the DICOM interface. It begun as a collaboration between Philips and AGFA, and was released as open source under the GNU LGPL license in June 2005. Since then a community has grown up around the product with contributions from independent developers worldwide as well as developers in AGFA and Philips.

Telefonica—the Morfeo Project: Telefonica I+D, the R&D division of Spanish telecom operator Telefonica, initiated the Morfeo project which operated in the area of Service Oriented Architectures and aimed to speed up the development of software standards in this area. Telefonica I+D was the customer "engine," releasing proprietary software components and injecting resources into the community. The project set up its own Morfeo implementation of a SourceForge-like portal, where the development base of the sub-projects integrated in Morfeo were hosted, including source repositories, binaries, mailing lists, bug and changes trackers, and documentation.

2.2 Company Cues

2.2.1 Do Not Seek to Dominate and Control Process

Conventional software engineering and outsourcing wisdom suggest that explicit and comprehensive requirements specifications are critical to project success. Apparently this seems to be at odds with the open source development model which is predicated on the principle of a developer perceiving 'an itch worth scratching,' to use Raymond's (2001) memorable phrase. Also, open source developers have typically been users of the software being developed (Dinh-Trong and Bieman 2004; Gacek and Arief 2004; Mockus and Herbsleb 2002), and the software was often targeted to a horizontal domain (such as office desktop applications or software development tools). In such situations, clear requirements specifications are not necessary as the requirements are widely understood and internalized by the individual developers. Raymond's (2001) characterization of the cathedral v. the bazaar to differentiate open source development from traditional software development caused the perception that open source development was merely about developers following their own agenda developing in parallel in a spirit of optimistic concurrency. However, Raymond's characterization was based on a limited sample of open source projects which did not reflect the heterogeneity of the open source development landscape even at the time. In recent times, open source development has become more formalized. This is evident in the regular project meetings which are now a feature of a number of popular open source products, such as the Apache conferences in the US and Europe, the Zope/Plone and PyPy development sprints, and the GNOME annual project conferences (German 2005) which bring together developers to coordinate and plan development. As open source development is increasingly being purposely 'steered' as customers seek to stimulate open source development in specific vertical domains (such as automotive or telecoms) where a developer may not perceive an itch worth scratching. In these development situations, clear specifications could indeed be important. This increasingly explicit formalization of specifications is already evident in the commercially sponsored projects that are increasingly becoming a feature of the open source landscape.

Interviewees, however, stressed the manner in which requirements specification in opensourcing differed from traditional outsourcing. Companies may have a clear idea of what functionality they would like to see the community adding to the product. However, there has to be consensus as to what functionality will be added. If a company pushes its own agenda too much in seeking to control development, there can be problems. The Celtix project manager within IONA expressed it well:

> A company cannot just go onto the mailing list or the community, and say 'Can you guys build this?' When kicking off the project in the open source community, it's about stating the overall goal and the top-level requirements you are trying to achieve. Then it's driven by consensus. If people perceive you as driving your own agenda, then you will get push-back on having things accepted.

This again emphasizes the delicate equilibrium that must be maintained between acceptable open source community values and customer desire for value creation (Fitzgerald 2006). Interestingly, it was stressed that within the ecosystem formed around this mode of development, it was quite permissible for customers to engage in more traditional outsourcing relationship directly with some developers in the community, outside the strict remit of the opensourcing project.

Also, it was often the case that the initial development community contributions were to provide something that existed within their repertoire. Thus, their contributions were not based on an assessment of the most pressing commercial priority for the company; rather they considered what they could offer based on their existing repertoire and expertise.

2.2.2 Provide Professional Management and Business Expertise

The importance of strong management support has been verified in several studies of ICT adoption (e.g., Agarwal 2000; Chatterjee et al. 2002; Fichman 2004; Gallivan 2001). Project ownership and senior management championship is undoubtedly critical for radical, high-risk initiatives such as open source deployment since it contravenes the traditional model where ongoing support is legally guaranteed by a vendor. Indeed, top management championship is likely to become even more important in the future as open source adoption moves out of the domain of invisible infrastructure systems to more visible, high-profile desktop systems and applications.

Not surprisingly then, the company interviewees identified with a project ownership obligation. One customer interviewee stressed the radical change in mindset represented by opensourcing, suggesting that it represented a strategic initiative which differed from the normal business model where developers could see that their salary was pretty much directly derived from the sales of the commercial product that they developed. In the opensourcing model, their work could appear to be benefiting the open source community, and not leading to an obvious direct remuneration. Thus, top management championship was necessary to convince developers that the strategy made business sense. In fact, the Celtix project could also grow the market for other proprietary IONA products, enabling additional support contract revenue.

In keeping with demonstrating strong ownership and commitment to the project, IONA established a full time position—Open Source Program Director. This person is responsible for engaging with the community, and helps ensure that issues relevant to the open source community receive prompt attention which helps ensure the quality of the software. Also, senior management commitment can help ensure that R&D resources are provided. Interestingly, however, there is a delicate equilibrium to be maintained here also. The Celtix project manager at IONA suggested that if the project is seen as too much an IONA project, the developer community may

have less interest in getting involved. A manager at Philips Healthcare summarized this dilemma:

> This application is deeply used within Philips organization. In several processes we depend on it. And because it's part of our product release, we want to stay in control of it. So on the one hand you want to control the project, and then on the other hand you want to be an open project and provide freedom for independent developers to join. And there's a little bit of conflict between these positions.

Several interviewees emphasized the need for the company to market the attractiveness of the project and improve its visibility. This has a twofold purpose. Firstly, the development community will perceive their reputation has been improved through involvement in a high-profile project with a high profile company. As a DVTk community interviewee expressed it: *"working with them [the company] is almost a sort of professional honor, as it were."*

This helps ensure the vibrancy of the project, and can attract further developers to the community, which cannot be taken for granted in a traditional open source project. Few open source projects to date have been deliberately started and nurtured to be successful; rather some extremely successful ones have emerged over time, whereas others have died off. Again, the Celtix project manager offered an interesting insight:

> There are a lot of open source projects which don't go anywhere, even though they have built good code. It also needs to be pushed so that it gets noticed and used by other projects, documented and marketed. This is a big overhead, and commercial companies have structures in place to help achieve that.

Interviewees also suggested that the customer company could provide their expertise in relation to software commercialization and productization in creating a professional open source product and subsequently marketing that product. This would involve a holistic approach and proactive marketing to ensure that all who could usefully consume the product were made aware of it, and could contribute. It was felt that a commercial vendor could usefully complement the open source community by providing this expertise. These efforts grow public awareness of the product, which can then create business-consulting opportunities for members of the development community. A DVTk community interviewee captured the essence of this well:

> What I am looking for is a multiplier. I am providing a piece of my work. I know they [the company] already have a useful chunk of work. But adding my bit to their larger existing work in a cooperative way creates something of greater multiplied use to everybody, including myself.

Given that there is a strong external element in opensourcing, customer interviewees stressed the necessity for clear project milestones and more visibility about product releases. This was contrasted with traditional proprietary development where internal milestones and actual release times are perhaps deliberately kept vague, whereas the need for the customer to be clear about future project plans was important to the community. More frequent product releases was identified as a by-product of the open source approach.

Also, it was suggested that the customer could not insist on a particular project-monitoring regime. Rather, different open source communities may have different norms and approaches in this area, and the customer has to be flexible and prepared to adapt to the particular regime in vogue in the particular open source community.

Related to transparency there was also a community expectation that the company have an open process for accepting community contributions. Interestingly, the company can have unrealistic expectations as to the level and significance of contributions. Indeed, one company manager suggested his experience was that of *"one significant contribution per month from the external community."*

Related to this is the issue of licensing and Intellectual Property (IP). A senior community interviewee with a background in commercial development suggested that it was sensible and pragmatic to have a clear IP policy. They had requested that IONA release the Celtix project under the Lesser General Public License (LGPL) and IONA agreed. The Celtix project manager suggested that IONA were keen to embrace open source and build trust within the community, and the choice of license is a key determinant for developers in deciding whether to participate in a particular open source project, and also for companies to adopt. Thus, the development community tends to have a preference for a restrictive (so-called viral or reciprocal) GPL-style license that safeguards their contributions due to its reciprocal nature. However, companies may tend towards more permissive licenses that afford them greater control over the future of the software. IONA had perceived the need to be even more open to other companies and communities, and hence dual licensed Celtix under the less restrictive Eclipse Public License also.

Related to this issue was the idea raised within Philips Healthcare of creating a single legal entity or foundation to represent the project. This trend towards legal incorporation has been established in several open source projects (O'Mahony 2005).

2.2.3 Help Establish an Open and Trusted Ecosystem

It was seen as important that both company and the community members strive towards the creation of a sustainable ecosystem around the product. A vital factor here is to create an atmosphere of trust. Creating an ecosystem and engendering trust is also facilitated by the fact that the project interaction tended to be "very much techie to techie" as one interviewee put it. The Celtix project management committee is chaired by an IONA Distinguished Engineer who would garner respect from the technical open source development community.

There was broad agreement from the community interviewees on this issue also. However, one community interviewee stressed the importance of meaningful content in the feedback. While promptness was appreciated, this evaporated if the content of the feedback was "empty." However, the "techie to techie" nature of the relationship helped ensure that feedback was meaningful.

Another issue with significant implications for project management practices within the customer company emerged in relation to development staff rotation. Normally, within proprietary software development companies, development staff are rotated after a few months onto other projects. However, because of the strong techie-to-techie interaction between developers in the company and community, there was a strong pressure from the community that developers would remain on the project for a lengthy period. This is in turn matched by the company expectation that community developers show strong loyalty and commitment to the project. The duration of involvement with the project is an aspect that varies across the different forms of outsourcing and is discussed in further detail in Chap. 5.

2.3 Community Cues

2.3.1 Clear and Democratic Authority Structure and Process Transparency

In the absence of traditional organizational sanctions, some form of structure is necessary to coordinate development. In many open source projects, this takes the form of 'benevolent dictatorship' as initially suggested by Raymond (2001). Several studies of open source development have detailed the complex authority structure that evolves over time to ensure that all code contributions are vetted and incorporated in a disciplined fashion (Mockus et al. 2002). Moreover, open source development has typically been characterized by the developers proactively taking charge, solving problems independently with minimal customer involvement, and without the need for traditional organizational sanctions to ensure development work is undertaken. Initially, open source developers did not engage in formal requirements analysis with customers (Scacchi 2002), but took direct responsibility for development decisions. Even though the open source development process is becoming more formalized (Fitzgerald 2006), open source developers are still more likely to retain a strong sense of independence.

Community representatives also identified clear and transparent authority structures as important—indeed, one stressed that these are *"not only important, but mandatory."* It was argued that since professional involvement in open source is increasing, the open source development community is expected to show the same level of quality and transparency as could be expected from any professional organization—people need to understand how decisions are made. Indeed, professionalism is permeating opensourcing projects since *"everybody knows there are business reasons why people are there,"* as pointed out by an IONA project manager, and also emphasized by a DVTk community member confessing that *"Getting good karma is great, but it's not my primary objective."* When the opensourcing community involves a large number of 'traditional professionals,' this is particularly emphasized.

The company interviewees also agreed that clear authority structures are important. However, in opensourcing, authority structures are framed by a strong belief in democratic principles:

> It would be good to ensure that that the [democratic] process is working, but I'm not sure that it is possible to see any authority structures other than that. It will always be shared responsibility.—Distinguished Engineer, IONA.

It was furthermore pointed out that such structures are important in two different respects. Firstly, they provide consistency between projects, which means that developers can easily contribute to more than one project. Contributing to several projects is not uncommon in OSS, and with the increasing interest in the so-called 'whole product approach' (Fitzgerald 2006), this is expected to be increasingly important—a point stressed by the Open Source Program Director at IONA. Secondly, they provide for consistent terminology within and across projects which ensures people are *"on the same page, and really focus on innovation."*

As a complement to clear authority structures, the lack of a traditional written outsourcing contract means that an open source community must be clear also about their actual work processes. Interestingly, this mirrors the transparency and close project monitoring that is expected on the part of the company.

2.3.2 Responsible and Innovative Attitude

OSS developers have long been acknowledged as being highly talented (Raymond 2001), and the loyalty of developers in the longevity of their commitment to their development projects has been remarkable (Feller and Fitzgerald 2002). Indeed, the cardinal sin of OSS, that of project forking (whereby a project is divided in two or more streams, each evolving the product in a different direction), is a strong community norm that acts against developer turnover on projects.

Although community and company interviewees alike believed it to be essential that the community takes responsibility and delivers on what has been committed to, this becomes somewhat blurred in the opensourcing context. In the Celtix project, for example, paid IONA employees do a large part of the overall development. Hence, the company has the power to manage part of the development effort more directly than would be possible in a traditional offshore outsourcing context. Although this may change as a project matures and the number of external contributions increases, an IONA project manager explained that there seemed to be a feeling that *"there will always be customer involvement."* However, he also suggested that a lot more community independence was expected in the future and that responsibility follows naturally with independence. A Manager at Philips Healthcare captured this crisply, referring to a potential problem with lack of written contracts:

> I think one of the biggest risks you have in open source development is when someone says—yes I'm going to do that, and he doesn't put any effort into it, nothing is done.

This also supports the company expectation of achieving consensus on the development roadmap, which assumes that the open source community is innovative and contributes constructively to this roadmap. As this will have a positive impact on the project, the community is expected to help achieve a positive impact among the company's customers.

2.3.3 Help Establish a Professional and Sustainable Ecosystem

The particular characteristics of open source position it as a good exemplar of the 'whole-product' concept of a market-driven business approach that seeks to deliver a complete solution to the customer in terms of products and services (Moore 1999). In this scenario, developers do the coding while others complete the business model by adding sales and marketing services—necessary activities but ones in which developers may not be interested. The open source 'whole-product' approach is also larger than a single company or software product or service. Indeed, the network benefits of open source arise as a result of the size of the overall community and ecosystem. Thus, an inter-organizational network of interested parties with complementary capabilities can form an ecosystem to offer a professional product and service in an agile, bazaar-friendly manner. Company service requests can be routed to the most appropriate expert partner in the network, perhaps even to the developer who wrote the actual code.

Thus, as mentioned above, both company and community members are expected to aim for a sustainable ecosystem. Consequently, community members are expected to be loyal and committed to continued involvement in the project, which, as mentioned above, is a marked feature of open source projects. This was confirmed by a Telefonica manager who explained that *"a benefit that open source brings is that people are more prepared to commit,"* which also transfers across to customer developers who *"even though they're working on the same technology now commit more personally."*

From a community perspective, it was suggested that the high-quality software associated with the open source model is an indication that the 'human capital management' is working. It was also perceived that the quality of the code is a way to attract more business, which is essential for the ecosystem to develop. As open source is moving away from networks of individuals to networks of companies, if a contributor earns a reputation for producing high-quality code, customers will keep coming back for more. It is also the case that customers sometimes use the open source model to identify the best suppliers, who are then approached directly and contracted in a 'traditional' outsourcing model.

Opensourcing customers expect the open source model to attract "high calibre people" who understand the project domain very well without requiring additional training. The Open Source Program Director of IONA even argued that it attracts a certain personality, with traits not necessarily those traditionally associated with a "top-notch programmer." In her view, people are attracted by the open source model

because they want to "build something better," they want to "get involved," and they want to "be part of a community"—in summary, *"these are the kind of people that I would want on my team, whether I was doing open source or not."*

Interestingly, open source community members do not necessarily see themselves as suppliers commissioned by a customer in the traditional sense. In fact, the open source community as represented by ObjectWeb sees its members not as open source developers but as "ecosystem developers." There seems to be a definite trend towards more organized open source communities, such as that of ObjectWeb. Hence, facilitating effective inter-organizational teams is to a large extent an intrinsic property of the opensourcing model. As indicated above, this is due to the merging of customer and community into one ecosystem: *"I don't consider IONA as a customer. IONA is a member"* was how the ObjectWeb Chairman described the situation. Opensourcing is thus not just about building good working relationships between customer and supplier. It is about "ecosystem development." Hence, although there are business reasons for people to participate, there is a lot more collaboration than in traditional outsourcing:

> In a traditional market you don't call up your competitor and be like, oh, well tell me what your stuff does. But in open source you do.—Open Source Program Director, IONA.

In opensourcing, the software developed is typically not aimed for end-users but is more likely to be tools and infrastructure components. Consequently, the customer and community participants typically share the same level of technical expertise—*"it is mostly developer-to-developer communication."* Therefore, there is no need for formal training. Instead, knowledge transfer is happening continuously "from one research lab to another" within the ecosystem. This was emphasized by the ObjectWeb Chairman who asserted that *"I don't speak about education or anything like that, I speak about exchange between researchers."* The Open Source Program Director at IONA acknowledged this view referring to it as "cross pollination." According to a Project Manager at IONA, knowledge transfer was also facilitated by an early and proactive focus on documentation as part of the work towards achieving consensus on a development roadmap discussed above (a customer obligation).

2.4 Company Versus Community Differences

To validate the interview findings, we also performed a survey among active company and community participants (Ågerfalk and Fitzgerald 2008). In a questionnaire we asked both company and community respondents to rate the importance of fulfillment of each other's expectations. The analysis revealed a number of issues on which company and communities differed. Significantly, community respondents were less positive than company respondents in relation to how open the company was to outside contributions. This suggests that the community were not fully persuaded that companies were living up to the expectation to accepting contributions from the community. Typically, in open source projects, a meritocracy emerges over time as developers 'prove' themselves and are allowed to commit contributions. The

discrepancy between the company and community here suggests that companies are cautious about accepting outside contributions into the code base, but presumably a meritocratic structure could emerge over time, which would see companies overcoming this reluctance.

However, the company respondents also differed from community respondents in that they were less positive about how the community provided a transparent authority structure to allow the company see the decision-making process within the community. This suggests that companies did not always perceive communities as being open in relation to development decisions, and also that the community might be remiss in relation to helping grow the public profile of the project. Given that companies see opensourcing as a means of growing product awareness, this is an obvious point of potential conflict, as companies would expect the community to help in this regard.

We also measured respondent opinion as to the success of opensourcing. However, there were no statistically significant differences between the company and community respondents. See Ågerfalk and Fitzgerald (2008) for details.

2.5 Summary of Opensourcing Cues

Altogether, the company and community cues identified above form a reciprocal set of cues that highlight important factors to consider when engaging in an opensourcing relationship. These are summarized in Table 2.1.

Table 2.1 Summary of company and community cues

Company Cues	Community Cues
Do not seek to dominate and control process	**Clear and democratic authority structure and process transparency**
• Not too forceful and dominant in pushing own agenda	• Provide a transparent authority structure to allow customer see the decision making process within the community
• Accept a general roadmap (vision) of future functionality rather than seeking a precise requirements specification	• Behave as a professional team
Provide professional management and business expertise	**Responsible and innovative attitude**
• Preserve continuity by keeping developers on projects for a longer period than the norm in proprietary software development	• Take responsibility and deliver on what is committed to
• Provide a business opportunity for the community to use the product	• Be creative and innovative in suggesting new functionality and directions for the project
• Provide professional expertise in relation to marketing and productizing the software	• Help achieve a positive impact among customers
• Provide R&D resources to further develop the project	
• Provide senior management commitment to the project	

(continued)

Table 2.1 (continued)

Company Cues	Community Cues
Help establish an open and trusted ecosystem	**Help establish a professional and sustainable ecosystem**
• Behave as a responsible member of the opensourcing ecosystem • Open to outside contributions • Transparent in plans for the future of the project • Seek to create trust in the relationship with the community • Engage in community-sustaining activities	• Offer high quality people who understand the project domain very well without requiring additional training • Exhibit loyalty and continued involvement in the project

2.6 Implications for Practice

Given that the opensourcing phenomenon has only recently emerged, success is not guaranteed. Our study findings can thus function as a checklist of salient issues for customers and communities engaging in opensourcing. The symmetrical and complementary nature of the company–community relationship illustrates how obligations and expectations need to be operationalized differently by the customer and community, but managed jointly, to ensure achievement of the mutual goal of a particular opensourcing initiative. Here we identify the potentially problematic issues that companies need to be aware of, and we also identify some new practices, or significant changes to traditional practices that are required.

2.6.1 Complementariness of Expectations: Product Lifecycle

While the company must be prepared to compromise at all stages and not seek to dominate and control the opensourcing process, the community must provide a transparent and democratic authority structure with shared responsibility. Furthermore, while the company must also be tactful and seek to embrace the open source values of openness and democracy, the community must be able to show the same level of management capability as would be expected from a traditional outsourcing partner. It is important that the company avoids pushing too hard its own agenda and accepts that requirements evolve throughout the project with active input from the community. To reciprocate, the community must make sure that its development process is communicated and accepted by the company. Furthermore, the company has to provide complementary expertise in relation to product commercialization and marketing. In turn, the community is expected to help improve public perception and awareness of the product.

When viewed holistically, a product lifecycle is evident. Firstly, the community is expected to take responsibility and deliver on what is committed to. In turn, the company can provide R&D resources to complement and further develop innovative ideas, and also provide its professional expertise in relation to marketing and productization of software, and in improving the visibility of the product. Furthermore, the credibility provided by the company in marketing a product can create a business opportunity for the product which community developers can then leverage.

Furthermore, the company must seek to reach consensus on the project monitoring system that will be instituted, on trying to show leadership and ownership of the project but not so strongly as to deter the development community. Overall, the company must achieve that delicate equilibrium between value-creation in terms of a successful business model for itself while not transgressing the open source community values. This includes embracing open source values of openness to be trustworthy in the eyes of the community. On the other hand, the community must adopt a professional attitude and approach in order to be taken seriously in the corporate world.

2.6.2 Changes to Standard Development Management Practices

We already referred to the need for management support on the company side to ensure buy-in to a risky initiative where it appears as if the 'crown jewels'—the company's software—is being given away for free. Standard operating practices that tend to apply in the customer company may need to change. For example, more clarity is required in relation to software release milestones so the community can plan their own business opportunities. Also, more frequent product releases are likely rather than artificially separating product release functionality on the basis of the revenue that can be generated which is often the case with proprietary software product releases. Furthermore, the policy in companies of rotating developers onto different projects after a period of several months may not be sustainable as company developers become associated with the project in the open source community. The community develops a trusted relationship with a company developer and do not wish to see that relationship disturbed through the company moving developers on to other projects.

2.6.3 Global Recruitment: From Unknown to Known

The study also revealed that outsourcing to the open source community provides a significant opportunity for companies to headhunt top developers, whereby community members become also employees of the customer—hence moving from

outsourcing to a largely unknown open source workforce towards recruitment of talented developers from the open source community. In this study we see a move from 'unknown unknowns' as neither the customer nor the community are known to each other, to a scenario of 'known knowns' as each gets to understand each other's position and build complementary skills. Essentially, the opensourcing model consists of a company (as opensourcing customer) and a community of individual developers and other companies with whom the customer interacts. This community provides potential for development cost savings, recruitment opportunities, and the capability of increasing innovation. Interestingly, while the community is expected to behave professionally and provide transparent authority structures (thus reducing the 'unknown factor'), innovation potential is primarily to be expected from the 'unknown' part of the workforce. Hence, there are forces in the ecosystem pulling in opposite directions: while cost-savings and innovation are facilitated by a large 'unknown' workforce, trust-building and recruitment of community developers by the customer will tend to erode the unknown aspect.

This progression from unknown to known is also possible for innersourcing, and, but perhaps to a lesser extent, crowdsourcing. In the case of the latter, the competitive element and the fact that developers in the crowd tend not to build a relationship with a single customer (see Chap. 4) militates against this happening in a crowdsourcing context to some extent.

2.6.4 Open Source: From Replication to Innovation

Interestingly, open source is often assumed to be about replication rather than innovation. However, creativity and innovation is stimulated by multi-disciplinary teams operating outside conventional organization structures (Nonaka 1991; Garvin 1993; Leonard-Barton 1995; Inkpen 1996; Goldman and Gabriel 2005). Certain characteristics have been identified, including *autonomy* (which forms the basis for self-organizing and widens the possibility that individuals will motivate themselves to form new knowledge); *creative chaos* (whereby individuals do not have to follow organizational 'rules,' but are challenged to investigate alternatives and re-think assumptions); *information redundancy* (where individuals have information that goes beyond their immediate needs for a particular task); and *requisite variety* (whereby the individuals involved have the skill diversity to match the complexity and variety of the environment they face). Interestingly, these characteristics appear to be present in open source communities and can thus be expected to play an important part in the success of opensourcing as a software sourcing strategy. Studies of open source participation have suggested that about 40–50 % of open source developers are employed within professional organizations, suggesting an 'open' community of about 50–60 % (Lakhani and Wolf 2005; Jørgensen 2005; Riehle et al. 2014). This latter cohort may provide the creative and innovative spark as open source loses its image of being merely about imitation of proprietary products, and innovation becomes the defining feature. In this study, the characteristics

of the community facilitate innovation as community developers operating outside traditional organizational constraints can identify new functionality and develop the software in creative ways—innovation does indeed seem to happen elsewhere (Goldman and Gabriel 2005).

2.6.5 Open Source: From Individual to Company

The study reveals an ongoing shift from open source as community of individual developers to open source as community of commercial organizations, primarily small and medium-sized enterprises, operating as a symbiotic ecosystem in a spirit of co-opetition. Overall the goal seems to be to create such a sustainable open and trusted ecosystem where customers and community participants operate as equals with neither party dominating. Thus, in contrast to traditional outsourcing, open-sourcing is not primarily about commissioning software development to a third-party, but rather about engaging in long-term collaborative activities leading towards a sustainable ecosystem. Since many of the collaborators in this ecosystem are likely to be the company's competitors, the collaboration is necessarily done in a spirit of co-opetition (Brandenburger and Nalebuff 1996). Both company and community members have a shared responsibility to actively contribute to the development and sustainability of the ecosystem. One way to ensure that this responsibility is honoured by the company is to do like IONA and appoint an Open Source Program Director. Other well known companies, including Google, HP and Microsoft, have done this as well.

http://www.springer.com/978-3-319-17265-1

Software Sourcing in the Age of Open
Leveraging the Unknown Workforce
Ågerfalk, P.J.; Fitzgerald, B.; Stol, K.-J.
2015, XII, 77 p. 4 illus., 1 illus. in color., Softcover
ISBN: 978-3-319-17265-1