Please cite as: Agerfalk, P, Fitzgerald, B, and Stol, K (2015) Not so Shore Anymore: The New Imperatives when Sourcing in the Age of Open, *Proceedings of the 23rd European Conference on Information Systems (ECIS)* Münster, Germany, May 2015.

NOT SO SHORE ANYMORE: THE NEW IMPERATIVES WHEN SOURCING IN THE AGE OF OPEN

Complete Research

Ågerfalk, Pär J., University of Uppsala, Sweden, par.agerfalk@im.uu.se

Fitzgerald, Brian, Lero—the Irish Software Research Centre, University of Limerick, Ireland, bf@lero.ie

Stol, Klaas-Jan, Lero—the Irish Software Research Centre, University of Limerick, Ireland, klaas-jan.stol@lero.ie

Abstract

Software outsourcing has been the subject of much research in the past 25 years, largely because of potential cost savings envisaged through lower labour costs, 'follow-the-sun' development, access to skilled developers, and proximity to new markets. In recent years, the success of the open source phenomenon has inspired a number of new forms of sourcing that combine the potential of global sourcing with the elusive and much sought-after possibility of increased innovation. Three of these new forms of sourcing are opensourcing, innersourcing and crowdsourcing. Based on a comparative analysis of a number of case studies of these forms of sourcing, we illustrate how they differ in both significant and subtle ways from outsourcing. We conclude that these emerging sourcing approaches call for conceptual development and refocusing. Specifically, to understand software sourcing in the age of open, the important concept is no longer 'shoring,' but rather the degree of 'workforce unknownness' and its implications for the development situation at hand.

Keywords: Outsourcing, open innovation, open source, opensourcing, inner source, innersourcing, crowdsourcing

1 Introduction

Outsourcing of software development has been steadily on the increase according to both US and European reports. However, in many cases, global sourcing of software development (often referred to as distributed development, global software development or global software engineering (GSE)), has not delivered on its promise (e.g., Nakatsu and Iacovou, 2009; Tiwana and Keil, 2009; Ó Conchúir et al., 2009). In fact, a significant body of research has identified economical, technical, organizational, and cultural challenges associated with global software sourcing (Damian and Moitra, 2006; Ågerfalk et al., 2009; Šmite et al., 2010). At the same time, the success of the open source software movement, which seems to overcome many of these challenges (Crowston et al., 2007; Stol et al., 2014), has been an inspiration for a number of specific forms of software sourcing.

The conventional wisdom of software engineering suggests that given the inherent complexity of software, it should be developed using tightly co-ordinated centralized teams, following a rigorous development process. In recent times, the open source phenomenon has attracted considerable attention as an agile, practice-led initiative that appears to address the three core aspects of the so-called 'software crisis,' namely, high cost of development, extended development time-scale, and poor quality of the final software product (Fitzgerald, 2004). In terms of development cost, open source products are usually freely available for public download. From the point of view of development time-scale, the collaborative, parallel efforts of globally-distributed co-developers has allowed many open source products to be developed much more quickly than conventional software through the 'follow-the-sun' elongation of working hours. Finally, in terms of quality, many open source products are recognized for their high standards of reliability, efficiency and robustness, and the open source phenomenon has produced several market leaders in their respective areas — Linux and Apache spring to mind. Indeed, these are known as 'category killers,' so called because their success removes any incentive to develop competing products. The open source model also seems to harness the most scarce resource of all: talented software developers, many of whom exhibit a long-standing commitment to their chosen projects. It is further suggested that the resulting peer review model helps ensure the quality of the software produced (Feller and Fitzgerald, 2002). Also, given that open source developers self-select to work on topics that interest them and that suit their ability, they are likely to be able to produce work of high quality.

Since the primary force driving offshore sourcing appears to be cost savings (Lacity et al., 2010) and the open source model is associated with significant cost savings (Wheeler, 2004), it is natural that companies would seek to exploit the open source development model. However, an additional highly-praised advantage of the open source model is its potential for increased innovation through access to a large skilled developer pool with both broad and deep expertise that offer the capacity to view problems in new ways (Carmel, 1999; 2006; Herbsleb and Grinter, 1999; Carmel and Agarwal, 2001; Ebert and De Neve, 2001; Carmel and Tjia, 2005; Ó Conchúir et al., 2009; Morgan and Finnegan, 2010; Eseryel, 2014). Thus, given the success of the open source model and its potential for game-changing cost savings and innovation, it is not surprising that open source would inspire new forms of sourcing. In previous research, we have conducted detailed case studies of three alternative, nascent forms of open source-inspired software sourcing, namely opensourcing (Ågerfalk and Fitzgerald, 2008), inner-sourcing (Stol et al., 2014) and crowdsourcing (Stol and Fitzgerald, 2014). In this paper, by revisiting and integrating the findings from these earlier studies, we characterize and compare these three forms of open source-inspired sourcing and illustrate how they differ from conventional outsourcing. As far as we are aware, this is the first study that systematically compares these emerging forms of software sourcing and contrasts them with conventional outsourcing. Such an effort should be of interest to managers facing software-sourcing decisions as well as to researchers interested in contemporary software sourcing approaches. In what follows, we discuss three alternative forms of sourcing (Sec. 2), followed by our research approach (Sec. 3). We then present a comparison of these sourcing strategies using an inductively developed framework (Sec. 4). We conclude in Section 5.

2 Sourcing in the Age of Open

2.1 Opensourcing

Carmel and Tjia (2005) have characterized offshore sourcing as 'outsourcing to a global workforce.' Opensourcing, on the other hand, can best be characterized as outsourcing to a *global* but largely *unknown* workforce of open source developers. The term 'opensourcing' has been suggested to refer to the use of the open source development model as a software sourcing strategy (Ågerfalk and Fitzgerald, 2008). Open source software can be defined as software released under the terms of a license that allows the licensee to use, modify and redistribute, either *gratis* or for a fee. Opensourcing thus allows companies to 'subcontract' development activities to an open source community. Since anyone (in principle) can join any open source project, knowing beforehand the location of a particular developer is impossible.

A central aspect of opensourcing is the mutuality and reciprocity between customer and community that are inscribed in the 'copyleft' terms found in many open source licenses (Ågerfalk and Fitzgerald, 2008). These terms decree that software can be used, modified and redistributed provided subsequent modifications are made freely available to others. Also, development is accomplished through the fulfilment of mutual expectations with respect to the activities of coding, debugging, testing and documentation. One of the most significant threats for the open source movement has been suggested to be the 'free rider' phenomenon whereby someone profits from open source without reciprocating, which thus contravenes these values of mutuality and reciprocity (Von Hippel and Von Krogh, 2003).

Much open source development is done in the absence of any legal employment contract for developers and the norms of how development is conducted are both written and unwritten. Developers are expected to be familiar with the written rules and norms before attempting to contribute (Feller and Fitzgerald, 2002). However, new recruits must also serve their apprenticeship in learning these rules and norms as part of their socialization (Raymond, 2001; Gorman, 2004; Eseryel, 2014).

2.2 Innersourcing

Inner source is defined as the adoption of open source development practices within the confines of an organization (Stol et al., 2011). Whereas well-defined methods, such as the agile Scrum approach, have clearly defined activities (e.g., Scrum meetings), artifacts (e.g., Sprint backlog), and roles (e.g., Scrum Master), this is not so much the case for inner source, although common open source development practices and roles can be identified (Stol et al., 2014). Rather than a well-defined methodology, we consider inner source to be a development philosophy, oriented towards the open collaboration principles of egalitarianism, meritocracy, and self-organization (Riehle et al., 2009). Within inner source, a number of common open source development practices can be observed such as universal access to development artifacts, transparent development, peer-review of contributions, informal communication, and self-selection of tasks by motivated contributors (Stol et al., 2014). Which of these practices are adopted as part of an inner source initiative varies per organization as each implementation of inner source is tailored to the particular context of the adopting organization (Stol and Fitzgerald, 2015). Existing development methods within a company may be augmented with open source practices. However, a key tenet of inner source is the universal access to the development artifacts throughout an organization so that anyone within the organization can potentially participate. In addition to common practices, a number of common roles can be identified (Höst et al, 2014). Inner source projects are often 'grassroots' initiatives, started by individuals, project teams, or departments (Riemens and van Zon, 2006; Gurbani et al., 2006; Melian, 2007). As such, the initiator typically assumes the role of a benevolent dictator (Raymond 2001; Gurbani et al., 2006). As some contributors become experts in parts of the project, they can be promoted to 'trusted lieutenants', and together with the benevolent dictator form a core team (Gurbani et al., 2010). Similar governance structures are commonly found in open source projects (Mockus et al., 2002). Additional roles may emerge in inner source, however; Gurbani et al. (2010), for instance, identified a number of roles in the core team at Alcatel-Lucent, each of which had a specific function in order to tailor the bazaar to a commercial software development context.

2.3 Crowdsourcing

Using contemporary Internet technologies, organizations can tap into a global workforce consisting of anyone with an Internet connection. Customers, or requesters, can advertise chunks of work, or tasks, on a crowdsourcing platform, where individual workers ('suppliers') select those tasks that match their interests and abilities (Hoffman, 2009). Crowdsourcing has been adopted in a wide variety of domains, such as design and sales of T-shirts (Howe, 2008) and pharmaceutical research and development (Lakhani and Panetta, 2007) and there are numerous crowdsourcing platforms through which customers and suppliers can find each other (Doan et al., 2011). One of the best known crowdsourcing platforms is Amazon Mechanical Turk (AMT) (Ipeirotis, 2010), on which chunks of work are referred to as Human Intelligence Tasks (HIT) or micro-tasks. Typical micro-tasks can be characterized as self-contained, simple, repetitive, short, and requiring little time, cognitive effort and specialized skills. Crowdsourcing has worked particularly well for such tasks (Kittur et al., 2011). Examples include tagging images, and translating fragments of text. As a result, remuneration of work is typically in the order of a few cents to a few US dollars. In addition to micro-tasks, there are cases of crowdsourcing also of complex tasks. For instance, InnoCentive deals with problem solving and innovation projects, which may yield payments of thousands of US dollars (Howe, 2008). Software development tasks are more akin to the latter as they are often interdependent, complex, heterogeneous, and can require extended periods of time, significant cognitive effort and diverse sets of expertise.

Similar to the confusion surrounding the term 'crowdsourcing' in general, there is some confusion about what constitutes crowdsourcing in a software development context. In particular, crowdsourcing may be positioned as closely related to other strategies such as outsourcing (Herbsleb and Mockus, 2003) and opensourcing (Ågerfalk and Fitzgerald, 2008). For instance, open source is often cited as the 'Genesis' of crowdsourcing (Howe, 2008, p.8; Kazman and Chen, 2009; LaToza et al., 2013), but others argue that open source is, in fact, not a form of crowdsourcing (Brabham, 2013). Other terms that have been used as synonyms are 'peer production' (Feller et al., 2008) and 'commons-based peer production' (Benkler, 2002; Kazman and Chen, 2009), both referring to the idea that software is developed by a group of peers. While these strategies are similar in some respects, there are significant differences that set crowdsourcing apart (Surowiecki, 2005). We adopt the following definition of crowdsourcing (Stol and Fitzgerald, 2014): *The accomplishment of specified software development tasks on behalf of an organization by a large and typically undefined group of external people with the requisite specialist knowledge through an open call.*

We agree with Brabham (2013) who argued that collaborative initiatives such as Wikipedia are not instances of crowdsourcing, as there is no initiating organization or 'call' for action. A number of potential benefits may arise through the use of crowdsourcing in general, and these would also be applicable in the context of crowdsourcing software development (Stol and Fitzgerald, 2014). These include cost reduction due to lower development costs for developers in certain regions; faster time-to-market due to parallel development on decomposed tasks; high quality due to a broad participation and in-depth knowledge of self-selected participants, and finally the ability to benefit from the creativity of the 'crowd,' (Schlagwein and Bjørn-Andersen, 2014) thus representing an instance of open innovation.

These benefits are similar as those associated with open source as discussed above. Given these benefits, crowdsourcing has the potential to become a common approach to software development (Begel et al., 2012; Kazman and Chen, 2009). The benefit of tapping into the creative capacity of a crowd is captured well in a quote attributed to Sun Microsystems co-founder Bill Joy, *"No matter who you are, most of the smartest people work for someone else"* (Lakhani and Panetta, 2007). As Lakhani and Panetta (2007) point out, completing knowledge-intensive tasks will become increasingly challenging in traditional closed models of proprietary innovation, if most of the knowledge exists outside an organization.

3 Research Approach

The research approach adopted can be described as a qualitative multiple case study (Yin, 2003). Data collection comprised of in-depth interviews with 32 key stakeholders in six case studies across four companies that had adopted these forms of sourcing, and associated open source communities in the case of opensourcing. The companies and respondents were as follows:

- *IONA Technologies*, at the time of the study a NASDAQ-quoted company headquartered in Dublin, Ireland. Incorporated open source by leading a community project to develop Celtix, an open source Java Enterprise Service Bus. The Celtix project was hosted by an established open source community, ObjectWeb, who specialized in developing open source middleware products.
- *Philips*, a global technology organization headquartered in Eindhoven, the Netherlands. One division, *Philips Healthcare* specializes in medical devices and initiated the DVTk project in collaboration with AGFA. After opensourcing the DVTk project, a community emerged around the product with contributions from developers worldwide. The second case study was at *Philips Healthcare* has also established an inner source initiative whereby a number of open source-inspired practices and principles have been incorporated into the development process while remaining compliant with FDA regulations. The third case study at Philips was at *Philips Research*, a different division within Philips, ran an inner source project concerned with an implementation of an internally developed file standard for storing video data.
- *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. Telefonica I+D was the customer "engine," releasing proprietary software and injecting resources into the community.
- *TechPlatform Inc*. (*TPI*, a pseudonym), a multinational offering cloud services and solutions. The company employs several tens of thousands of people, and has offices and partners worldwide.

Table 1. Overview of the six cases and data collection at four organizations.

	Opensourcing	**Innersourcing**	**Crowdsourcing**
Cases	IONA Technologies: ObjectWeb Philips Healthcare: DVTk Telefonica I+D: Morfeo	Philips Healthcare: Inner Source Platform Philips Research: Inner Source project	TechPlatform Inc. (TPI): Migration of field engineer desktop tool to web
Data collection	13 Interviews, incl. Chief Scientist, admin staff, OSS director, 2 project managers at IONA, and Chairman and 2 developers at ObjectWeb	14 Interviews, incl. 2 directors, 4 managers, 2 architects, 2 team leads, 3 key developers, 1 DVTk developer	5 Interviews, incl. a architect, program manager, division manager, software development manager

We adopted qualitative data analysis techniques such as open and axial coding (Straus and Corbin, 1998). During analysis, we first characterized each individual form of sourcing and then inductively derived a set of dimensions along which the sourcing strategies could be meaningfully compared. Our focus was on characteristics that differentiated one form of sourcing from other forms. For instance, in both opensourcing and innersourcing we observed a recurring theme that management cannot tell individuals what to do. However, in crowdsourcing this is a prerequisite, and thus one theme that emerged was that of where the locus of control lies in these forms of outsourcing.

A possible concern with interpretive qualitative research is that it relies on individual interpretations of data and different individuals may interpret the same data differently (Kaplan and Duchon, 1988). To minimize this effect, all coding and memoing were performed as a joint effort by the three authors during a series of online meetings during spring 2014 and two intensive co-located workshops, one in July 2014 and one in September 2014. This time period of more than six months offered ample opportunity to discuss and reflect on the dimensions emerging from the comparative analysis. Furthermore, we adopted the practice of *venting* (Goetz and LeCompte, 1984) whereby emerging interpretations were formally presented and discussed at two research seminars with internationally renowned researchers in the area, one in May 2014 and one in October 2014. The interview recordings, their transcription, and memos written during the analysis established an audit trail, which is another recommended practice in qualitative data analysis (Creswell and Miller, 2000).

4 Comparing Outsourcing and Alternative Forms of Sourcing

Each of the three sourcing strategies discussed in Section 2 presents an alternative to conventional outsourcing. Each approach, however, differs in a number of aspects due to the specific characteristics of these approaches. Following the inductive approach outlined in Section 3, we identified six emergent themes that capture the key differences between these alternative forms of sourcing, and which can also be used to compare them to conventional outsourcing. Table 2 illustrates how these three forms of open-source inspired sourcing differ from outsourcing and from each other on a number of dimensions. These are discussed in the remainder of this section.

Table 2. Comparison of outsourcing, opensourcing, innersourcing and crowdsourcing

	Outsourcing	**Opensourcing**	**Innersourcing**	**Crowdsourcing**
Locus of Control	Company IP protected	Community IP open	Company or community IP protected	Company IP protected
Nature of Workforce	Known Narrow & deep knowledge	Unknown, can be difficult to find out Broad & deep knowledge	Known Broad & deep knowledge	Unknown but known to platform Broad & deep knowledge
Community Motivation	Extrinsic	Intrinsic and extrinsic	Extrinsic and intrinsic	Extrinsic
Company Motivation	Resource saving and overcoming lack of resources	Innovation Market growth Cost sharing (commodification)	Reuse Resource saving Innovation	Resource saving / overcoming lack of resources Innovation
Duration of Engagement	Project-specific, contractual commitment	Prolonged commitment	Prolonged commitment	Ad hoc commitment
Nature of participation	Collaborative	Co-opetive	Collaborative	Competitive, possibly collaborative

4.1 Locus of Control

The locus of control refers to the question of who initiates and retains control in the sourcing relationship. In conventional software outsourcing, the locus of control lies with the customer company who has a certain software development task that is given to a third party to perform. In such an arrangement, one company (the customer) commissions another company (the provider) to perform the work. In this relationship, the customer takes the initiative and retains control by requesting specific and well specified tasks to be carried out by the provider. The resulting deliverable (i.e., the software produced by the provider) becomes the intellectual property of the paying customer.

Although a company engaging in opensourcing may ideally want to steer the long-term direction of a project, the locus of control will often lie primarily with the community, depending on the level to which the opensourcing company stays involved in the project's development. It was clear in our study of opensourcing that the customer should not seek to dominate or control the agenda as this would lead to push-back from the community (Ågerfalk and Fitzgerald, 2008). A project manager within IONA expressed this sentiment well:

> *"A company cannot just go onto the mailing list or the community, and say 'Can you guys build this.' It's [rather] about stating the overall goal and the top-level requirements you are trying to achieve [and] then it's driven by consensus. If people perceive you as driving your own agenda, then you will get pushback on having things accepted."*

The resulting deliverable from opensourcing will typically be released under an open source license and intellectual property will usually be shared openly. Alternatively, a company may opt for a dual licensing model and thus retain some control and flexibility in relation to the IP (as was the case in the Celtix project, for example (Ågerfalk and Fitzgerald, 2008)).

Both innersourcing and opensourcing involve the adoption of open source development philosophy by a commercial company. However, in terms of locus of control, innersourcing can be considered a hybrid of opensourcing and outsourcing. Most cases of innersourcing start out as grassroots initiatives, suggesting that the locus of control lies with the (internal) 'community,' i.e., the developers employed by an organization. Besides the difference in openness and license of the software, innersourcing differs from opensourcing in that developers cannot completely ignore their position as a paid employee of the company or the job requirements of their position within that company (Höst et al., 2014). Different inner source projects that can be observed in practice have different governance models. For instance, the inner source initiative within Philips Healthcare has augmented the traditional governance model, by providing mechanisms and conventions that prescribe how contributions can be made and who is responsible for the maintenance of such contributions. This is necessary given the critical role that the shared asset plays as the platform that underpins the product line of medical devices, which are subject to regulatory authorities including the FDA. Nevertheless, business units within large 'federated' organizations often have a high degree of autonomy. While there is an organizational structure that defines responsibilities and authority, the 'community' of developers within business units retain their autonomy, as one director of technology at Philips Healthcare explained:

> *"We don't have the authority to tell departments, from now on you'll do things like this. It doesn't work that way. It's mainly building a case together with people, [to discuss] what would be sensible, and then you can get things done."*

Other inner source initiatives are more reminiscent of open source projects, whereby the locus of control lies with the 'community.' For example, a different inner source initiative within Philips Research does not have formal leadership but rather a de-facto leadership that lies with the initiators of the project. Other companies may have different models of governance, and the development of a governance taxonomy of inner source projects is one area that we believe needs more research.

Overall, while opensourcing communities are defined by their ability to self-organize and may resist any attempt by companies to control and dictate the development agenda, the degree of autonomy drops in the case of inner source as the employees, even though they may be in a different division of the organization, certainly do not possess complete autonomy over their work practices, and are less able to self-organize as they still need to consider their responsibilities that come with their position within the organization.

In terms of locus of control, crowdsourcing is very much along the line of outsourcing. The customer company specifies the task to be done by the crowd community, which can result in a significant documentation effort. One architect from TPI illustrated that this process was very different from an internal development approach:

> *"It feels like we've produced a million specification documents, but obviously we haven't. The way we do specifications for TopCoder is entirely different to how we do them internally."*

The work is typically decomposed into competitions under which community participants submit proposed solutions; in our case study, there were more than 50 different competitions. Any IP is owned by the company. Although crowdsourcing is open source inspired, it departs significantly from open source principles. Companies use the crowd to reduce costs or to stimulate innovation through fresh and new ways of considering situations afforded by the crowd. However, the business model requires that the company control the situation. Essentially, the development agenda is dictated by the company, and crowd participants really only has the choice as to participate in a contest or not. However, crowdsourcing participants do have the capacity to self-organize *themselves*, though, typically they do not collaborate in the way that open source developers would.

4.2 Nature of Workforce

The nature of the workforce can be characterized by two aspects: the degree of 'unknownness' and the nature of the knowledge that the workforce has.

In conventional outsourcing, the workforce is necessarily known; that is, an organization will choose a supplier on the basis of their known track record and ability to deliver, and a contract will have been put in place before any of the work is started. The level of expertise by the workforce of a known outsourcing supplier is typically narrow and deep: an outsourcing supplier may focus on a specific domain or certain technologies, and given this specialization, the level of knowledge can be very deep.

In opensourcing, the identities of contributing developers are typically not known, although contributors may be asked to sign a contribution agreement, thus revealing their identity. In a sense, the company outsources to a largely unknown workforce. The model thus assumes that the collective of developers (i.e., the community) will deliver and does not tie compensation and rewards to individual contributions. By tapping into a large pool of developers, possibly spread across the globe, a company may get access to a wide and deep level of expertise that they would not have access to otherwise — to attract "*high calibre people*" as the Open Source Program Director at IONA put it.

In contrast to opensourcing, in an innersourcing context, contributors are known by necessity. User accounts are typically linked to developers' unique corporate email addresses. Members of the internal community will interact on forums such as mailing lists and (IRC) Internet Relay Chat using these corporate identifiers. While developers within the same inner source project may have never met before (not uncommon in large distributed organizations), each member of the inner source project community will have a 'base' position in the company's hierarchy. With respect to the knowledge of the workforce, however, inner source is very similar to open source, in that an inner source project may benefit from a wide variety of contributors from throughout the company with very specific and deep knowledge. Depending on the size of the organization and the particular inner source project, the size of the community may vary from several hundreds of developers as is the case in Philips Healthcare to several tens of developers as was the case within Philips Research's inner source project.

In crowdsourcing, developers who participate in a contest are typically unknown to a customer. While developers need to specify certain information in order to get paid when winning a contest, very little information is public, except for the country in which a developer is based. This is necessary as a customer may choose to exclude submissions from certain countries. Furthermore, one of the problems that our crowdsourcing study revealed was the lack of continuity — the "fleeting relationship" in that developers in the crowd would not tend to wait for further competitions from a particular company but would work on whatever competitions were open (Stol and Fitzgerald 2014). An architect involved in the crowdsourced project at TPI illustrated this as follows:

> *"There is a limited amount of carry-over knowledge. We will get a few contestants that will participate in multiple contests, but they won't build up domain knowledge in the way that an internal person would."*

Also, customer companies may choose to remain anonymous when crowdsourcing in an attempt to protect IP. This creates a two-way level of unknownness as neither customer nor community know each other in some cases.

With respect to the level of knowledge of the workforce, crowdsourcing is very similar to opensourcing and innersourcing, in that the degree of knowledge tends to be broad and deep for similar reasons as in opensourcing and innersourcing. If the available talent-pool is truly global, then there is good reason to expect broad and deep knowledge on the topics under development.

In the case of crowdsourcing and opensourcing, the global reach of both phenomena ensures broad and deep knowledge associated with requisite variety. It is less likely to be as pronounced in the case of innersourcing but certainly in multinational organizations, the variety will increase as the participant pool becomes larger through organization-wide involvement.

4.3 Community Motivation

Community motivation refers to the individual developers' motivations and incentives and can therefore vary among individuals. The motivation of the developer community varies across the different sourcing strategies. A distinction is usually drawn between intrinsic and extrinsic motivation (Ryan and Deci, 2000). Intrinsic motivation refers to motivation derived from an individual's pure interest or enjoyment in the task itself. Extrinsic motivation, on the other hand, arises when an activity is driven by the desire to receive a reward, typically a payment, or to win a competition. Such motivation is generally external to the individual.

In a conventional outsourcing context, the community or supplier motivation is clearly extrinsic. Suppliers perform a task for payment under a contract typically with penalties for late or non-performance.

Lerner and Tirole (2002) argued that the two major motivations for contributing to open source projects are career concerns and ego gratification, which they collectively referred to as the signalling incentive. By contributing to an open source project, developers gain reputation and status within that community, which thus appears to be the main driving force. A Celtix project community member even suggested that working with a company can be "*almost a sort of professional honour*." Thus, the reward can be a delayed compensation where successful open source developers could be rewarded eventually by better job prospects. The intrinsic motivation to participate was further emphasized by a DVTk community interviewee, who stated:

> *"Adding my bit to [the company's] larger existing work in a cooperative way creates something of greater multiplied use to everybody, including myself."*

In opensourcing, payment is sometimes part of the picture and developers may find a more direct link between their 'voluntary' work and potential career advancements than in traditional open source. Interestingly, this appears to be evolving as Riehle et al. (2014) recently reported that more than 50% of open source code contributions occurred during office hours, which suggests a conventional paid workforce, at least to some extent. Earlier studies have suggested about 40% (Lakhani and Wolf, 2005; Jørgensen, 2005).

Motivation of developers in an innersourcing context can be either extrinsic or intrinsic. Typically, setting up an inner source initiative is not done at the request of a manager or supervisor, but rather these are often set up by visionary individuals, or 'champions,' who seek to improve internal collaborations.

Inner source contributors likewise may derive enjoyment and satisfaction from contributing to a project. One initiator of the inner source project at Philips Reseaerch commented:

> *"at some point [the work] is appreciated by colleagues, and at that point you also do it to help our your colleagues, solving their problem, and that results in satisfaction because [the software] is used by others."*

On the other hand, external 'rewards' may also arise in inner source when contributors are able to finish their assigned work more quickly—innersourcing offers them empowerment. Developers can overcome their dependence on the maintainers (the core team) of an inner source project as it allows them to fix defects or make changes themselves (in a local copy), very similar to open source projects.

In crowdsourcing, the community motivations are primarily extrinsic. On the TopCoder crowdsourcing platform, various forms of remuneration (first and second prizes, reliability bonus, Digital Run funds) are available to active participants (Stol and Fitzgerald, 2014). Furthermore, the extrinsic motivation becomes even clearer given that many registered contestants will withdraw from competitions if they perceive that they have no chance of winning a prize. Also, some developers seek an official TopCoder rating and use that on their CVs to indicate independent validation of their technical ability. It thus forms a career signalling incentive similar to that proposed by Lerner and Tirole (2002).

4.4 Company Motivation

Company motivation to adopt a conventional outsourcing approach includes reduced development costs, reduced time-to-market as a result of 'follow-the-Sun' software development across multiple time-zones, cross-site modularization of development work, access to a larger and better skilled developer pool, innovation and shared best practices, and a closer proximity to customers (Ågerfalk and Fitzgerald, 2006; Ó Conchúir et al., 2009).

A distinguishing motivation for opensourcing is that of *commodification* (van der Linden et al., 2009; Whelan et al., 2014). Increasingly, large parts of software systems are becoming 'commodities' — non-differentiating components that, although needed for a system to function properly, do not add any unique business value to a product. Classic examples are operating systems, database management systems and network protocol stacks (e.g., TCP/IP). No software company will, for example, implement their own database management system (unless, of course, their core product is a database system). Moreover, the 'innovation happens elsewhere' (Goldman and Gabriel, 2005) argument appears to be a strong company incentive to engage in opensourcing since it allows companies to tap into a global developer community with competencies and experiences that the company may not have in-house. According to the Open Source Program Director at IONA, opensourcing provides access to *"the kind of people that I would want on my team, whether I was doing open source or not."* Since open source developers are often also users, engaging with the community can also be a way of reaching out to, and even creating, new markets.

While innersourcing refers to the application of the open source development philosophy within an organization's boundaries, the motivations to adopt inner source differ significantly from those which are relevant to the adoption of opensourcing. Firstly, a common reason to adopt inner source is to increase internal reuse of software (Vitharana et al., 2010). By making available various internally developed software components to all departments, projects or business units, others can reuse these components as they see fit. Inner source can also help in reducing the time-to-market; Van der Linden (2009) reported that Philips Healthcare was able to reduce the time-to-market by at least three months. Partly, this faster time-to-market will be a result of software reuse, but also due to the flexibility that the inner source model allows. Product divisions are empowered to make 'local' changes to the inner source product so as to allow them to overcome certain limitations (or to fix specific bugs) shortly before a product release, without escalating it to the 'core team' which may not have time to address these issues immediately.

Besides these motivations, 'open innovation' is another reason why a company may want to adopt inner source (Morgan et al., 2011). Similar to opensourcing, inner source projects can potentially attract a larger pool of developers (albeit within company boundaries) than found in conventional projects — especially in large, global organizations that employ thousands of people.

The motivation for companies who participate in crowdsourcing is certainly based on resource saving issues. Companies may be persuaded by the savings promised by crowdsourcing platforms — a 62% saving has been suggested for software development using TopCoder, for example, although the available evidence would not appear to support this estimate (Stol and Fitzgerald, 2014). Furthermore, a company may not have in-house expertise in a particular topic or technology and seek to source that from the crowd. Also, the desire for innovation is certainly a factor as companies seek to get fresh ideas on topics. Indeed, this aspect is heavily promoted by crowdsourcing platform providers.

4.5 Duration of Engagement

In conventional outsourcing the engagement tends to be project-specific as governed by a contract between both parties. Although a company may have a long-term relationship with a particular supplier, it will be episodic insofar as the contractual commitment will be as defined for each engagement.

While conventional outsourcing is primarily about commissioning software development to a third-party, opensourcing is rather about engaging in long-term collaborative activities that create and reinforce a sustainable ecosystem of individuals and organizations (Ågerfalk and Fitzgerald, 2008). Recently, Von Krogh et al. (2012) emphasized that although extrinsic motivation is important to sustainable community participation, long-term engagement and contribution to the community are even more critical. This was echoed by a respondent who suggested that the interaction among company and community developers in the IONA project was *"very much techie to techie,"* which created a strong pressure to remain on the project.

Similar to opensourcing, the duration of engagement of developers in an innersourcing project tends to be long-term as developers will have a long-term interest in the software product. However, actual activity in terms of contributions, fixes, etc. can vary from daily activity to a very sporadic pattern, depending on the type of software as well as its level of maturity. For example, developers of the inner source project within Philips Research only worked on the project in 'bursts of activity' as defects or new requirements were identified. Activity in this project would only last for a short time, after which weeks or months could pass before the next contributions. In contrast, other inner source projects can be in a state of perpetual development, similar to many large successful open source projects.

In crowdsourcing, the engagement between the company and community tends to be short-term as defined by competitions, with ad hoc commitment from the community. This is reflected in the "fleeting relationship" mentioned above, which characterizes the crowdsourcing company community interaction (Stol and Fitzgerald, 2014). Competitions of long duration tend not to be attractive to the crowd, and result in fewer and lower quality submissions. The recommendation on the TopCoder platform is to have lots of competitions in parallel. Thus, the duration of engagement is geared much more towards a short-term model. However, although if the duration of engagement is relatively short in crowdsourcing, frequent engagement can emerge if a small group of participants participate in numerous consecutive contests.

4.6 Nature of Participation

In an outsourcing context, the participation between customer and supplier is clearly collaborative. Suppliers are carefully chosen on the basis of their ability to perform a particular task. The company will decompose the work in such a way that the supplier will supply complementary offerings in a collaborative manner.

As noted above, the 'free rider' phenomenon has been identified as a threat to opensourcing. On a similar note, the ethics of crowdsourcing has been questioned due to its taking advantage of the creativity of the user community for commercial gains (Bruns, 2007). Successful opensourcing, however, is characterized by reciprocity and symbiosis (cf. Dahlander and Magnusson, 2005). In fact, in addition to individuals, the ecosystem that emerges in opensourcing is typically constituted by several commercial organizations that would normally compete but instead choose to collaborate on a particular project, thus suggesting as specific form of open innovation (Remneland Wikhamn and Wikhamn, 2013). Such collaboration between competitors is sometimes referred to as co-opetition. *"I don't consider IONA as a customer. IONA is a member,"* was how the situation was described by the Chairman of ObjectWeb. The Open Source Program Director at IONA confirmed this view when pointing out that:

> *"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."*

The nature of participation in innersourcing projects is similar to opensourcing; participants in inner source projects are working collaboratively to improve the software. Again as in opensourcing, this collaboration may be implicit (everybody working towards a better product, possibly on different parts of the software) or explicit (two or more developers working and discussing the implementation of a feature). They may work on a specific feature or module either on their own, or in collaboration with others while communicating through email or IRC.

In crowdsourcing, the nature of participation is clearly competitive, though collaboration models may vary across different crowdsourcing platforms. Crowd participants work on competitions in isolation without sharing or collaborating on solutions, and the best entry is adjudged to be the winner of the competition. One implication of the competitive nature of participation is that potential participants may decide not to partake after all, once they find out that certain other, very successful and skillful, members of the community are also participating, as they are expected to 'win' the contest anyway. In such a case, any efforts spent in such a competition are expected to be in vain.

Interestingly, Brooks (1995) observed that software should be considered as public property and viewable to all. This is consistent with the open source model which has had enormous success due to the opportunities for learning that developers are afforded by being able to see the code of other developers. The nature of competition in crowdsourcing ensures that such sharing does not take place, and this is inevitably a sub-optimal situation. Ironically, one of the problems with crowdsourcing is *information under-load* as contestants have no organizational knowledge or 'organizational memory,' to help interpret the requirements as specified. Furthermore, communication is typically through a narrow chat forum that does not facilitate any flow of rich information. Similar problems occur in the case of opensourcing although the lack of the competitive element in opensourcing is likely to encourage knowledge sharing through facilities such as wikis and mailing lists. Innersourcing participants are likely to possess more organizational knowledge to help create information redundancy.

5 Conclusion

The key contribution of this work is the articulation of how three alternative forms of open-source inspired sourcing (opensourcing, innersourcing and crowdsourcing) differ in significant and subtle ways from conventional outsourcing on a range of dimensions (see Table 2). Although companies are increasingly getting involved in opensourcing, little is known about these alternative models of engagement. Likewise, research on inner source is scarce and a taxonomy of different inner source mechanisms and models is still lacking. Crowdsourcing has attracted considerable interest in recent years, but models of interaction have not been systematically compared beyond suggesting that some models are competitive whereas others can be co-opetive.

By revisiting our previous research, this paper has characterized and compared three approaches to software sourcing that are inspired by the OSS development model. The analysis led to a framework with six dimensions along which the three approaches can be compared (see Table 2), which is discussed in Section 4. Based on this analysis, five imperatives for sourcing in the age of open stand out. These imperatives represent conditions for which companies need to strategize, or at least become cognizant of when adopting these emerging forms of sourcing.

First, the control of a project may, to some extent shift from a company to a developer community. Such a shift is true for IP also: a company may need to disclose some of its IP to an open community. We refer to this 'letting go' of control imperative as *governance sharedness*, an issue also identified by Shaikh and Cornford (2010).

Second, sharedness is accentuated by the fact that we are dealing with a potentially unknown workforce. This *unknownness* imperative means that developer location, which has been the focus of much global software engineering research (Šmite and Wohlin, 2011), is becoming less relevant since development can happen at any shore at any time (perhaps *anyshoring* would be an appropriate term). Furthermore, while innersourcing is, per definition, in-house and crowdsourcing is a type of external subcontracting, opensourcing can happen both internally and externally in any given project. Indeed, there is increasing participation by companies in open source projects, and many large corporations, such as HP, Samsung, and Wipro, now have an 'open source community expert' role.

Third, the success of open source inspired sourcing approaches appears strongly linked to intrinsic motivation. The importance of this *intrinsicness* imperative varies across different sourcing approaches and is likely to be critical when reward is non-monetary, as can be the case in opensourcing.

Fourth, an important incentive for organizations to engage with these emerging forms of sourcing is arguably a perceived potential for innovation, both in terms of innovating the software development process (e.g. saving resources and decreasing time to market) and of innovating the software products and services (e.g. gaining access to new markets). Thus, the innovativeness imperative is in many ways seen as the *raison d'être* and as such becomes key to understanding the potential benefits of open-source inspired sourcing.

Fifth, a basic tenet of the age of open is the new modes of collaborations that form around the *co-opetitiveness* imperative whereby competing actors find mutual benefits in co-operating in certain activities.

Thus, to understand software sourcing in the Age of Open, the important concept is no longer 'shoring,' but rather these five imperatives (governance sharedness, unknownness, intrinsicness, innovativeness and co-opetitiveness) and their implications for the development situation at hand.

In this paper we adopted a primarily descriptive approach and have not aspired to theoretical contribution beyond conceptualizing the three sourcing strategies and five imperatives. However, in keeping with Ågerfalk (2014), we argue that our findings have important theoretical implications yet to be engaged. For instance, sharedness has implications for our understanding of commodification (Van der Linden et al., 2009) and global software engineering (Šmite and Wohlin, 2011), as well as for international business in general (Santos et al., 2004). Similarly, unknownness could be further explored in relation to business intelligence, forecasting and risk management (Enkel et al., 2005). The issues surrounding intrinsicness can probably help shed light on the more general psychological questions of motivation (Von Krogh et al., 2012) and also our understanding of open source as gift culture (Bergquist and Ljungberg, 2001) and commons-based peer production (Benkler, 2002; Feller et al., 2008; Kazman and Chen, 2009). Moreover, the importance of innovativeness and co-opetitiveness places this research squarely in the open innovation discourse with contemporary issues of peer production and democratization vis-à-vis business-model driven open innovation in relation to open source services networks (Feller et al., 2008; Chesbrough, 2012).

Acknowledgements. We thank the anonymous reviewers for constructive feedback. This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 to Lero—the Irish Software Research Centre, Enterprise Ireland grant IR/2013/0021 to ITEA2-SCALARE (scalare.org) and the Irish Research Council New Foundations programme.

References

Ågerfalk, PJ (2014) *Insufficient theoretical contribution: A conclusive rationale for rejection?* Eurpean Journal of Information Systems, Vol. 23, No. 6, pp. 593–599.

Ågerfalk, PJ and Fitzgerald, B (2006) *Flexible and Distributed Software Processes: Old Petunias in New Bowls?* Communications of the ACM Vol. 49, No. 10, pp. 26-34

Ågerfalk, PJ, Fitzgerald, B and Slaughter, SA (2009). *Flexible and Distributed Information Systems Development: State of the Art and Research Challenges.* Information Systems Research, Vol. 20, No. 3, pp. 317–328.

Ågerfalk, PJ and Fitzgerald, B (2008) *Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy*, MIS Quarterly, Vol 32, No. 3, pp. 385-410

Begel, A, Herbsleb, JD and Storey, MA (2012) The Future of Collaborative Software Development. *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*. ACM

Benkler, Y (2002) *Coase's Penguin, or, Linux and the Nature of the Firm*, The Yale Law Journal, Vol. 112, No. 3, pp. 369-446.

Brabham, DC (2013) *Crowdsourcing*, MIT Press.

Brooks, FP (1995) *The Mythical Man-Month*. Addison Wesley Longman, Inc.

Bruns, A (2007) Produsage: Towards a Broader Framework for User-Led Content Creation, *Proceedings of the 6th ACM SIGCHI Conference on Creativity and Cognition*, Washington, DC.

Carmel, E (1999) *Global Teams: Collaborating Across Borders and Time Zones*, Upper Saddle River, NJ: Prentice-Hall.

Carmel, E (2006) *Building Your Information Systems from the Other Side of the World: How Infosys Manages Time Zone Differences*, MISQ Executive Vol. 5, No. 1, pp. 43-53.

Carmel, E, and Agarwal, R (2001) *Tactical Approaches for Alleviating Distance in Global Software Development*, IEEE Software Vol. 18, No. 2, pp. 22-29.

Carmel, E, and Tjia, P (2005) *Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce*, Cambridge, NY: Cambridge University Press.

Chesbrough, H (2012) *Open innovation: Where we've been and where we're going*, Research-Technology Management, Vol. 55, No. 4, pp. 20–27.

Creswell, JW and Miler, DL (2000) *Determining Validity in Qualitative Inquiry*, Theory into Practice, Vol. 39, No. 3, pp. 124–130.

Crowston, K, Li, Q, Wei, K, Eseryel, UY and Howison, J (2007) *Self-organization of teams for free/libre open source software development*, Information and Software Technology, Vol. 49, No. 6, pp. 564-575.

Dahlander, L and Magnusson, MG (2005) *Relationships between Open Source Software Companies and Communities: Observations from Nordic Firms*, Research Policy Vol. 34, pp. 481-493.

Damian, D and Moitra, D (2006) Global software development: how far have we come? IEEE Software. Vol. 23, No. 5, pp. 17–19.

Doan, A, Ramakrishnan, R and Halevy, AY (2011) *Crowdsourcing systems on the World-Wide Web*, Communications of the ACM, Vol. 54, No. 4.

Ebert, C, and De Neve, P (2001) *Surviving Global Software Development*, IEEE Software Vol. 18, No. 2, pp. 62-69.

Enkel, E, Kausch, C and Gassmann O (2005) *Managing the risk of customer integration*, European Management Journal, Vol. 23, No. 2, pp. 203–213.

Eseryel, UY (2014) *IT-Enabled Knowledge Creation for Open Innovation*, Journal of the Association for Information Systems, Vol. 15, No. 11, pp. 805–834.

Feller, J and Fitzgerald B (2002) *Understanding Open Source Software Development*. Pearson Education Ltd.

Feller, J, Finnegan, P, Fitzgerald, B and Hayes, J (2008) *From Peer Production to Productization: A Study of Socially Enabled Business Exchanges in Open Source Service Networks*, Information Systems Research, Vol. 19, No. 4.

Feller J, Finnegan, P, Hayes, J and O'Reilly, P (2010) Leveraging 'The Crowd': An Exploration of How Solver Brokerages Enhance Knowledge Mobility, *Proceedings of the 18th European Conference on Information Systems*.

Fitzgerald, B (2004) A Critical Look at Open Source, IEEE Computer Vol. 37, No. 7, pp. 92-94.

Fitzgerald, B (2011) *Open source software: Lessons from and for software engineering*. IEEE Computer Vol. 44, No. 10, pp. 25–30.

Goetz, J and LeCompte, MD (1984) *Ethnography and Qualitative Design in Educational Research*. Academic Press.

Goldman, R and Gabriel, RP (2005) *Innovation Happens Elsewhere: Open Source as Business Strategy*, Morgan Kaufmann.

Gorman, M (2004) *Understanding The Linux Virtual Memory Manager*, Technical Report, University of Limerick, Ireland.

Gurbani, VK, Garvert, A and Herbsleb, JD (2006) A case study of a corporate open source development model. *Proceedings of the 28th International Conference on Software Engineering*. pp. 472-481.

Gurbani, VK, Garvert, A and Herbsleb, JD (2010) *Managing a corporate open source software asset*. Communications of the ACM Vol. 53, No. 2, pp. 155–159.

Hoffmann, L (2009) *Crowd Control*, Communications of the ACM, Vol. 52, No. 3.

Höst, M, Stol, K and Orucevic-Alagic, A (2014) Inner Source Project Management, In: Ruhe, G and Wohlin, C (Eds.) *Software Project Management in a Changing World*, Springer.

Howe, J (2008) *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*, Crown Business.

Ipeirotis, PG (2010) *Analyzing the Amazon Mechanical Turk marketplace*, XRDS, Vol. 17, No. 2, pp. 16-21.

Jørgensen, N (2005) Incremental and Decentralized Integration in FreeBSD. in *Perspectives on Free and Open Source Software*, J. Feller B. Fitzgerald, S. Hissam, and K. Lakhani (eds.), Cambridge, MA: MIT Press

Kaplan, B and Duchon, D (1988) *Combining Qualitative and Quantitative Methods in IS Research: A Case Study*, MIS Quarterly Vol. 12, No. 4, pp. 571-587.

Kazman, R and Chen, HM (2009) *The Metropolis Model: A new Logic for Development of crowdsourced systems*, Communications of the ACM, Vol. 52, No. 7.

Kittur, A, Smus, B, Khamkar, S and Kraut, RE (2011) CrowdForge: Crowdsourcing Complex Work. *Proceedings of the ACM Symposium on User Interface Software and Technology*.

Lacity, M, Khan, S, Yan, A, and Willcocks, L (2010) *A Review of the IT Outsourcing Empirical Literature and Future Research Directions*, Journal of Information Technology, Vol. 25, No. 4, pp. 395-433.

Lakhani, KR, and Wolf, RG (2005) Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. in *Perspectives on Free and Open Source Software*, J. Feller B. Fitzgerald, S. Hissam, and K. Lakhani (eds.), Cambridge, MA: MIT Press

Lakhani, KR and Panetta, JA (2007) *The Principles of Distributed Innovation*, Innovations: Technology, Governance, Globalization, Vol. 2, No. 3.

LaToza, TD, Towne, WB, van der Hoek, A and Herbsleb, JD (2013) Crowd Development. In *Proceedings of the 6th CHASE Workshop*. San Francisco, CA, USA. IEEE.

Lerner, J, and Tirole, J (2002) *Some Simple Economics of Open Source*, The Journal of Industrial Economics Vol. 50, No. 2, pp. 197-234.

Melian, C (2007) *Progressive open source*. Ph.D. Dissertation. Stockholm School of Economics, Sweden.

Mockus, A, Fielding, RT, and Herbsleb, JD (2002) *Two Case Studies of Open Source Software Development: Apache and Mozilla*, ACM Transactions on Software Engineering and Methodology Vol. 11, No. 3, pp. 309-346.

Morgan, L and Finnegan, P (2010) *Open innovation in secondary software firms: an exploration of managers' perceptions of open source software*, ACM SIGMIS Database, Vol. 41, No. 1, pp. 76-95.

Morgan, L, Feller, J, and Finnegan, P (2011) Exploring inner source as a form of intraorganisational open innovation. *Proceedings of the European Conference on Information Systems*, Helsinki, Finland.

Nakatsu, RT and Iacovou, CL (2009) *A Comparative Study of Important Risk Factors Involved in Offshore and Domestic Outsourcing of Software Development Projects: A Two-Panel Delphi Study*, Information & Management Vol. 46, No. 1, pp 57-68.

Ó Conchúir, E, Ågerfalk, PJ, Holmström Olsson, H, and Fitzgerald, B (2009) *Global software development: never mind the problems — where are the benefits?* Communications of the ACM, Vol 52, No 8.

Raymond, ES (2001) *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media.

Remneland Wikhamn, B and Wikhamn, W (2013) *Structuring of the Open Innovation Field.* Journal of Technology Management & Innovation, Vol. 8, No. 3, pp. 173–185.

Riehle, D, Ellenberger, J, Menahem, T, Mikhailovski, B, Natchetoi, Y, Naveh, B and Odenwald, T (2009) *Open collaboration within corporations using software forges*. IEEE Software Vol. 26, No. 2, pp. 52-58.

Riehle, D, Riemer, P, Kolassa, C, and Schmidt, M (2014) Paid vs. Volunteer Work in Open Source. In *Proceedings of the 47th Hawaii International Conference on System Science*. pp. 3286-3295.

Riemens, B and van Zon, K (2006) *PFSPD short history*. http://pfspd.sourceforge.net/history.html.

Ryan, RM and Deci, EL (2000) *Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions*, Contemporary Educational Psychology, Vol. 25, pp. 54-67.

Santos, J, Yves D, and Williamson P (2004) *Is Your Innovation Process Global?* MIT Sloan Management Review, Vol. 45, No. 4, pp. 31–37.

Schlagwein, D and Bjørn-Andersen, N (2014) *Organizational Learning with Crowdsourcing: The Revelatory Case of LEGO*, Journal of the Association for Information Systems, Vol. 15, No. 11, pp. 754-778.

Shaikh, M and Cornford, T (2010) 'Letting Go of Control' to Embrace Open Source: Implications for Company and Community. *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, Koloa, Kauai, Hawaii.

Šmite, D and Wohlin, C (2011) *A whisper of evidence in global software software engineering*, IEEE Software Vol. 28, No. 4, pp. 15-18.

Šmite, D, Wohlin, C. Gorschek, T and Feldt, R (2010) *Empirical evidence in global software engineering: a systematic review*. Empirical Software Engineering, Vol. 15, No. 1, pp. 91–118.

Stol, K, Babar, MA, Avgeriou, P and Fitzgerald, B (2011) *A comparative study of challenges in integrating open source software and inner source software.* Information and Software Technology, Vol. 53, No. 12, pp. 1319–1336.

Stol, K, Avgeriou, P, Babar, M, Lucas, Y and Fitzgerald, B (2014) *Key Factors for Adopting Inner Source*, ACM Transactions on Software Engineering Methodology (TOSEM), Vol. 23, No. 2

Stol, K and Fitzgerald, B (2014) Two's Company, Three's a Crowd: A Case Study of Crowdsourcing Software Development, *Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, India, May 2014, pp. 187-198.

Stol, K and Fitzgerald, B (2015) *Inner Source—Adopting Open Source Development Practices within Organizations: A Tutorial*, IEEE Software, Vol. 32.

Strauss, A and Corbin, JM (1998) *Basics of Qualitative Research: Techniques and Procedures for Devleoping Grounded Theory*. Sage Publications, Thousand Oaks, CA, USA.

Surowiecki, J (2005) *The Wisdom of Crowds: Why the Many Are Smarter Than the Few*, Abacus.

Tiwana, A and Keil, M (2009) *Control in Internal and Outsourced Software Projects*, Journal of Management Information Systems Vol. 26, No. 3, pp 9-44.

Van der Linden, F, Lundell, B and Marttiin, P (2009) *Commodification of industrial software: A case for open source*. IEEE Software, Vol. 26, No. 4, pp. 77-83.

Vitharana, P, King, J and Chapman, HS (2010) *Impact of internal open source development on reuse: participatory reuse in action*, J Manage Inf Syst, Vol. 27, No. 2, pp. 277-304

Von Hippel, E, and Von Krogh, G (2003) *Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science*, Organization Science Vol. 14, No. 2, pp. 209-223.

Von Krogh, G, Haefliger, S, Spaeth, S, and Wallin, MW (2012) *Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development*. MIS Quarterly, Vol. 36, No. 2, pp. 649–676.

Wheeler, D (2004) *Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!*, available online at http://www.dwheeler.com/oss_fs_why.html.

Whelan, E, Conboy, K, Crowston, K, Morgan, L and Rossi, M (2014) *Editorial: The Role of Information Systems in Enabling Open Innovation.* Journal of the Association for Information Systems, Vol. 15, No. 11, pp. xx–xxx.

Yin, RK (2003) *Case Study Research: Design and Methods* (3rd Ed.) Sage Publications, Thousand Oaks, CA, USA.