# Chapter 5
# Conclusion

The previous three chapters presented different forms of software outsourcing to an unknown workforce—opensourcing, innersourcing and crowdsourcing. These sourcing strategies all rely on a workforce that is not fully known in advance, unlike conventional software projects where teams are formed before a project is started. Furthermore, there are varying degrees of 'unknownness.' In opensourcing, it could be very hard to identify the actual people who are working on the software, as developers may operate under an alias. In innersourcing, it is not known in advance who will be working on the software, but it will be much easier to identify who the contributors are, as they will use their corporate identity (their corporate email address, for instance). In crowdsourcing, it will be possible to identify the developers 'after the fact.' That is, once a crowdsourcing contest comes to its end and the 'winner' is selected, his or her identity will be known.

Each form of software sourcing presented in this book is appropriate for some situations, and less so for others. They differ in a number of ways, which is why each chapter drew on a different set of factors or issues that was of particular relevance for the respective strategy. However, in order to compare the three strategies and provide concrete guidance to software managers as to which strategy should be pursued, we developed a comparison framework. This chapter presents this framework; for each comparison criterion, we discuss the implications for each sourcing strategy.

## 5.1 A Comparison Framework

The three alternative strategies for outsourcing that were presented in Chaps. 2 to 4 differ in a number of significant ways. In this section we discuss a number of dimensions along which these alternative forms of outsourcing differ. Table 5.1 below illustrates how these three forms of open-source inspired sourcing differ from outsourcing and from each other.



P.J. Ågerfalk et al., *Software Sourcing in the Age of Open*, SpringerBriefs
in Computer Science, DOI 10.1007/978-3-319-17266-8_5

**Table 5.1** Comparison of outsourcing, opensourcing, innersourcing and crowdsourcing

| | Outsourcing | Opensourcing | Innersourcing | Crowdsourcing |
|---|---|---|---|---|
| Locus of control | • Company<br>• IP protected | • Community<br>• IP open | • Company or Community<br>• IP protected | • Company<br>• IP protected |
| Nature of workforce | • Known<br>• Narrow and deep knowledge | • Unknown, can be difficult to find out<br>• Broad and deep knowledge | • Known<br>• Broad and deep knowledge | • Unknown but known to platform<br>• Broad and deep knowledge |
| Community motivation | • Extrinsic | • Intrinsic and extrinsic | • Extrinsic and intrinsic | • Extrinsic |
| Company motivation | • Resource saving<br>• Overcoming lack of resources | • Innovation<br>• Market growth<br>• Cost sharing<br>• Save resources (commodification) | • Reuse<br>• Resource saving<br>• Innovation | • Resource saving/ overcoming lack of resources<br>• 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 |

### *5.1.1 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. It was clear in the case of opensourcing that the customer could not dominate or control the agenda as this would lead to push-back from the community. The resulting deliverable from opensourcing (i.e., the software produced) 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 refers to the adoption of open source development philosophy by a commercial company. However, in terms of locus of control, innersourcing is pretty much 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 (Stol et al. 2014). 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. 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 such as the FDA. 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 (a different division of the Philips consortium) 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.

In terms of locus of control, crowdsourcing is very much along the lines of outsourcing. The customer company specifies the task to be done by the crowd community. This is decomposed into competitions under which community participants submit proposed solutions. 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.

### *5.1.2 Nature of Workforce*

The nature of the workforce can be characterized by two factors: 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. It is narrow because an outsourcing supplier may focus on a specific domain or certain technologies. However, 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,[1] thus providing their real name and signature. In a sense, the company outsources to a largely unknown workforce. The outsourcing 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 expertise that they would not have access to otherwise.

In contrast to opensourcing, in an inner source context, contributors will be known. User accounts (e.g., for commit access) are typically linked to developers' unique corporate email addresses. 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.

In crowdsourcing, developers who participate in a contest are unknown to a customer (Stol and Fitzgerald 2014a). While different platforms exist for crowdsourcing, on the platform in the study we presented in Chap. 4, TopCoder, developers go by a nickname, or 'handle.' 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. Also, customer companies may choose to remain anonymous on the TopCoder platform. This creates a two-way level of unknownness as neither customer nor community know who each other is 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.

### *5.1.3 Community Motivation*

The motivation of the developer community varies across the different sourcing strategies. In Chap. 4 above, we drew the distinction between intrinsic and extrinsic motivation. Intrinsic motivation refers to internal motivation that is derived from an individual's pure interest or enjoyment in the task itself. Extrinsic motivation, on

[1] http://en.wikipedia.org/wiki/Contributor_License_Agreement

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 the 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 signaling 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. This is echoed by Nov (2007) who studied incentives to contribute to Wikipedia based on Clary et al.'s (1998) motivational categories in volunteer work. Thus, the reward can be a delayed signaling incentive where successful open source developers could be rewarded eventually by better job prospects. 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) 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. This is in contrast to the earlier finding by Lakhani and Wolf (2005) whose large-scale survey found that only 40 % of open source developers were in organizations.

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. Contributors likewise may derive enjoyment and satisfaction from contributing to a project. On the other hand, external 'rewards' may also arise in inner source when contributors are able to finish their assigned work more quickly. Developers can also overcome their dependence on the maintainers of an inner source project as it allows them to fix defects or make changes themselves, very similar to open source projects.

In crowdsourcing, the community motivations are primarily extrinsic. On the TopCoder platform, various forms of remuneration (first and second prizes, reliability bonus, Digital Run funds) are available to active participants. 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 signaling incentive similar to that proposed by Lerner and Tirole (2002).

### *5.1.4 Company Motivation*

Motivations to adopt a 'traditional' outsourcing strategy include 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). 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 systems). Another example is the DVTk-project discussed in Chap. 2. This project begun as a collaboration between Philips and AGFA because both companies felt that developing a proprietary toolkit would be wasting engineering resources that could be better spent on software that could lead to a competitive advantage. Moreover, the 'innovation happens elsewhere' argument appears to be a strong company incentive to engage in opensourcing since opensourcing allows companies to tap into a global developer community with competencies and experiences that the company may not have inhouse. 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 key reason to adopt inner source is to increase internal reuse of software. 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 et al. (2009) reported that Philips Healthcare was able to reduce the time-to-market by at least 3 months. Partly this faster shorter 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 suggested for software development using TopCoder, for example,[2] although the available evidence would

[2] http://techcrunch.com/2013/09/17/appirio-buys-topcoder-to-add-more-crowdsourcing-and-500k-developers-and-designers-to-its-cloudspokes-network/ (accessed 27 Sep 2014).

not appear to support this estimate (Stol and Fitzgerald 2014a). Also, they 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 thinking and ideas on topics. Indeed, this aspect is heavily promoted by the crowd platform providers.

### *5.1.5 Duration of Engagement*

In traditional 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 in so far as the contractual commitment will be as defined for each engagement.

While traditional 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. 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.

Similar to opensourcing, the duration of engagement of developers in an inner source 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 (Stol and Fitzgerald 2015). 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 be to be short-term as defined by competitions, with ad hoc commitment from the community. Again, this is reflected in the "fleeting relationship" which characterises the crowdsourcing company community interaction. As already mentioned in Chap. 4, competitions of long duration tend not to be attractive to the crowd, and result in fewer and lower quality submissions. The recommendation from TopCoder is to have lots of competitions in parallel. Thus, the duration of engagement is geared much more towards a short-term model.

### *5.1.6 Nature of Participation*

In the 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. Such collaboration between competitors is sometimes referred to as co-opetition.

The nature of participation in inner source 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) 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. 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.

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.

## 5.2 Concluding Remarks

The outsourcing topic is not new and is one which has been thoroughly researched in the past. However, one of the key implications from our research is that the three alternative forms of open source inspired sourcing that we identify and discuss in this book differ both in significant and in subtle ways from the concept of outsourcing on a range of dimensions as illustrated in Table 5.1. Thus, we need to derive new concepts and rethink others to fully understand and leverage these alternate forms of sourcing.

In the literature, the terms offshoring and outsourcing are often used almost as synonyms. In keeping with Davis et al. (2004), however, it is possible to distinguish between the two:

- Offshoring is about location—when an activity is offshored it is performed in a different location to the main operation (which is then the onshore location).
- Outsourcing, on the other hand, is about governance—when an activity is outsourced it is performed by another organization, as opposed to 'in-house' by the organization itself.

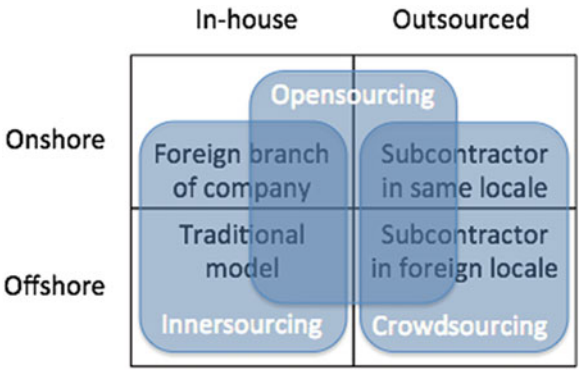


**Fig. 5.1** Shoring and sourcing in the age of open

Consequently, the two concepts are orthogonal—an activity can be performed either offshore or onshore and can be performed in-house or be outsourced. Figure 5.1 shows the distinction and relationship between the two concepts and maps the three sourcing strategies discussed in this book onto the two concepts.

Interestingly, the global software engineering community has devoted considerable effort to distinguishing between different forms of 'shoring' based on location distance between collaborating sites using terms such as 'onshoring,' 'offshoring,' 'nearshoring' and 'farshoring' (Smite and Wohlin 2011). What becomes evident from our studies above is that in the age of open, distinguishing between forms of *shoring* has become a non-issue. Similarly what is in-house and what is subcontracted is no longer obvious. The nature of the unknown workforce means that it is not only irrelevant to talk about developer location, it is by definition *impossible*—any developer can be 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 internally in any given project. Thus, 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.

# References


Agarwal, R (2000) Individual Acceptance of Information Technologies, Framing The Domains of IT Management: Projecting the Future Through the Past, R. W. Zmud (ed.), Cincinnati, OH: Pinnaflex Press, pp. 85-104.

Ågerfalk, PJ (2013) Embracing diversity through mixed methods research, European Journal of Information Systems, Vol. 22, No. 3, pp. 251-256.

Ågerfalk, PJ (2015) Insufficient theoretical contribution: a conclusive rationale for rejection? European 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 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

Asundi, J (2001) Software engineering lessons from open source projects. In Proceedings of the 1st Workshop on Open Source Software Engineering. J Feller, B Fitzgerald, and A van der Hoek (Eds.).

Asundi, J and Jayvant, R (2007) Patch review processes in open source software development communities: A comparative case study. In Proceedings of the 40th Annual Hawaii International Conference on Systems Sciences (HICSS).

Augustin, L, Bressler, D, and Smith, G (2002) Accelerating software development through collaboration. In Proceedings of the 24th International Conference on Software Engineering. pp. 559-563.

Aurum, A, Jeffery, R, Wohlin, C and Handzic, M (2003) Managing Software Engineering Knowledge, Springer.

Baldwin, CY and Clark, KB (2006) The architecture of participation: Does code architecture mitigate free riding in the open source development model? Management Science Vol. 52, No. 7, pp. 1116-1127.

Beecham, S, Baddoo, N, Hall, T, Robinson, H and Sharp, H (2008) Motivation in Software Engineering: A systematic literature review, Information and Software Technology, Vol. 50, No. 9-10.

Begel, A, Herbsleb, JD and Storey, MA (2012) The Future of Collaborative Software Development. In 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.




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

Bjørnson, FO and Dingsøyr, T (2008) Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used, Information and Software Technology, Vol. 50, No. 11.

Boehm, BW (1981) Software Engineering Economics, Pearson Education.

Boehm, B (2006) A View of 20th and 21st Century Software Engineering. In Proceedings of the International Conference on Software Engineering. Shanghai, China. ACM. pp. 12-29.

Bonabeau, E (2009) Decisions 2.0: The Power of Collective Intelligence, MIT Sloan Management Review, Vol. 50, No. 2, pp. 45-52.

Bonaccorsi, A and Rossi, C (2003) Why open source software can succeed. Research Policy Vol. 32, No. 7, pp. 1243-1258.

Brabham, DC (2008) Crowdsourcing as a Model for Problem Solving: An Introduction and Cases, Convergence, Vol. 14, No. 1.

Brabham, DC (2012) The Myth of Amateur Crowds: A critical discourse analysis of crowdsourcing coverage. Information, Communication & Society Vol. 15 No. 3.

Brabham, DC (2013) Crowdsourcing, MIT Press.

Brandenburger, AM, and Nalebuff, BJ (1996) Co-Opetition: A Revolution Mindset That Combines Competition and Co-operation, New York: Doubleday.

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.

Capiluppi, A, Stol, K, and Boldyreff, C (2012) Commercial Stakeholders in the Evolution of Open Source Software, in: Hammouda et al., Proceedings of the 8th International IFIP WG2.13 Conference on Open Source Systems (OSS), IFIP Advances in Information and Communication Technology (AICT), vol. 378, pp. 178-200.

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.

Chatterjee, D, Grewal, R, and Sambamurthy, V (2002) Shaping Up for E-Commerce: Institutional Enablers of the Organizational Assimilation of Web Technologies, MIS Quarterly Vol. 26, No. 2, pp. 65-89.

Clary, EG, Snyder, M, Ridge, RD, Copeland, J, Stukas, AA, Haugen, J, Miene, P (1998) Understanding and Assessing the Motivations of Volunteers: A Functional Approach, Journal of Personality and Social Psychology, Vol. 74, No. 6, 1516–1530.

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

Dabbish, L, Farzan, R, Kraut, R and Postmes, T (2012) Fresh Faces in the Crowd: Turnover, Identity, and Commitment in Online Groups. In Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW). ACM.

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

Davis, GB, Ein-Dor, P, King, WR and Torkzadeh, R (2004) Information Technology Offshoring: Prospects, Challenges, Educational Requirements, and Curriculum Implications. Proceedings of the 25th International Conference on Information Systems, R. Agarwal, L.J. Kirsch, and J.I. DeGross (eds.), Washington, DC, December, pp. 1027–1038.

Dempsey, B, Weiss, D, Jones, P, and Greenberg, J (2002) Who Is an Open Source Software Developer?, Communications of the ACM Vol. 45, No. 2, pp. 67-72.

Dinh-Trong, T, and Bieman, JM (2004) Open Source Software Development: A Case Study of FreeBSD, in Proceedings of the 10th International Symposium on Software Metrics, IEEE Computer Society.

Dinkelacker, J, Garg, PK, Miller, R and Nelson, D (2002) Progressive open source. In Proceedings of the 24th International Conference on Software Engineering. pp. 177-184.

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

Dow, SP, Kulkarni, A, Klemmer, SR and Hartmann, B (2012) Shepherding the Crowd Yields Better Work. In Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW). ACM.

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

Erdogmus, H (2009) A process that is not. IEEE Software Vol. 26, No. 6, pp. 4-7.

Erenkrantz, JR (2003) Release management within open source projects. In Proceedings of the 3rd Workshop on Open Source Software Engineering. Feller, J, Fitzgerald, B, Hissam, SA and Lakhani, KR (Eds.).

Erenkrantz, JR and Taylor, RN (2003) Supporting distributed and decentralized projects: Drawing lessons from the open source community. In Proceedings of the 1st Workshop on Open Source in an Industrial Context. Marc Sihling (Ed.).

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

Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (Eds) (2005) Perspectives on Free and Open Source Software, MIT Press, Cambridge, MA.

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.

Fichman, RG (2004) Going Beyond the Dominant Paradigm for IT Innovation Research: Emerging Concepts and Methods, Journal of the Association for Information Systems Vol. 5, No. 8, pp. 314-355.

Fitzgerald, B (2006) The Transformation of Open Source Software, MIS Quarterly Vol. 30, No. 3, pp. 587-598.

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

Fitzgerald, B, Stol, K, O'Sullivan, R and O'Brien, D (2013) Scaling Agile Methods to Regulated Environments: An Industry Case Study. In Proceedings of the 35th International Conference on Software Engineering. San Francisco, CA, USA. IEEE.

Fogel, K (2005) Producing Open Source Software: How to Run a Successful Free Software Project. O'Reilly Media.

Gacek, C and Arief, B (2004) The many meanings of open source. IEEE Software Vol. 21, No. 1, pp. 34-40.

Gallivan, M (2001) Organizational Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework, The DATA BASE for Advances in Information Systems Vol. 32, No. 3, pp. 51-85.

Garvin, D (1993) Building a Learning Organization. Harvard Business Review Vol. 71, No. 4, pp. 78-91.

Gaughan, G, Fitzgerald, B, and Shaikh, M (2009) An examination of the use of open source software processes as a global software development solution for commercial software engineering. In Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 20-27.

German, DM (2005) Software Engineering Practices in the GNOME Project. In Perspectives on Free and Open Source Software, Feller, J., Fitzgerald, B, Hissam, SA, Lakhani, KR (Eds.) MIT Press

Ghosh, RA (2005) Understanding Free Software Developers: Findings from the FLOSS Study, in Perspectives on Free and Open Source Software, Feller, J., Fitzgerald, B, Hissam, SA, Lakhani, KR (Eds.) MIT Press

Goldman, R, and Gabriel, RP (2005) Innovation Happens Elsewhere: Open Source as Business Strategy, San Francisco: Morgan Kauffman Publishers.

Gonzalez, R, Gasco, J and Llopis, J (2006) Information Systems Outsourcing: A literature analysis, Information & Management Vol. 43, No. 7, pp. 821-834.

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

Greengard, S (2011) Following the Crowd, Communications of the ACM Vol. 54, No. 2, pp. 20-22.

Gurbani, VK, Garvert, A and Herbsleb, JD (2006) A case study of a corporate open source development model. In 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.

Halloran, TJ and Scherlis, WL (2002) High quality and open source software practices. In Proceedings of the 2nd Workshop on Open Source Software Engineering. Feller, J, Fitzgerald, B, Hecker, F, Hissam, SA, Lakhani, K and van der Hoek, A (Eds.).

Herbsleb, JD, and Grinter, RE (1999) Splitting the Organization and Integrating the Code: Conway's Law Revisited, in Proceedings of the 21st International Conference on Software Engineering, Los Angeles, California.

Herbsleb, JD and Mockus, A (2003) An Empirical Study of Speed and Communication in Globally Distributed Software Development, IEEE Transactions on Software Engineering, Vol. 29, No. 6.

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

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

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

Inkpen, AC (1996) Creating Knowledge through Collaboration, California Management Review Vol. 39, No. 1, pp. 123-140.

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

Ipeirotis, PG and Paritosh, PK (2011) Managing Crowdsourced Human Computation. In Proceedings WWW.

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

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.

Kinnaird, P, Dabbish, L, Kiesler, S and Faste, H (2013) Co-Worker Transparency in a Microtask Marketplace. In Proceedings Computer Supported Coordination Work.

Kittur, A (2010) Crowdsourcing, Collaboration and Creativity, XRDS, Vol. 17, No. 2.

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

Kittur, A, Nickerson, JV, Bernstein, MS, Gerber, EM, Shaw, A, Zimmerman, J, Lease, M and Horton, JJ (2013) The Future of Crowd Work. In Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW). ACM.

Koh, C, Ang, S, and Straub, DW (2004) IT Outsourcing Success: A Psychological Contract Perspective, Information Systems Research Vol. 15, No. 4, pp. 356-373.

Kraut, RE and Streeter, LA (1995) Coordination in Software Development, Communications of the ACM, Vol. 38, No. 3.

Kulkarni, A, Can, M and Hartmann, B (2012) Collaboratively Crowdsourcing Workflows with Turkomatic. In Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW). ACM.

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

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, Garvin, DA and Lonstein, E (2010) TopCoder (A): Developing Software through Crowdsourcing, Harvard Business School 610-032.

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.

Leonard-Barton, D (1995) Wellsprings of Knowledge: Building and Sustaining the Sources of Innovation, Boston: Harvard Business School Press.

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

Lings B, Lundell B, Ågerfalk P J, Fitzgerald B (2007) A reference model for successful Distributed Development of Software Systems. Proceedings of the 2nd International Conference on Global Software Engineering (ICGSE 2007), Munich, Germany, 27–30 August 2007.

MacCormack, A, Rusnak, J and Baldwin, CY (2006) Exploring the structure of complex software designs: An empirical study of open source and proprietary code. Management Science Vol. 52, No. 7, pp. 1015-1030.

Malone, TW and Crowston, K (1994) The Interdisciplinary Study of Coordination, ACM Computing Surveys, Vol 26, No. 1.

McConnell, SC (1999) Open-source methodology: Ready for prime time? IEEE Software Vol. 16, No. 4, pp. 6-8.

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

Merilinna, J and Matinlassi, M (2006) State of the art and practice of open source component integration. In Proceedings of the 32nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) pp. 170-177.

Michlmayr, M and Fitzgerald, B (2012) Time-based release management in free and open source (FOSS) projects. International Journal of Open Source Software and Processes Vol. 4, No. 1, pp. 1-19.

Michlmayr, M, Fitzgerald, B, Stol, K (2015) Why and How Should Open Source Projects Adopt Time-Based Releases? IEEE Software, Vol. 32, No. 2.

Millar, C, Choi, CJ, Russell, ET, and Kim, JB (2005) Open Source Communities: An Integrally Informed Approach, Journal of Organizational Change Management Vol. 18, No. 3, pp. 259-268.

Mockus, A and Herbsleb, JD (2002) Why Not Improve Coordination in Distributed Software Development by Stealing Good Ideas from Open Source?, in Meeting Challenges and Surviving Success: The Second Workshop on Open Source Software Engineering, pp. 19-25.

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, 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.

Moore, G (1999) Crossing the Chasm, Harper, NY

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.

Neus, A and Scherf, P (2005) Opening minds: cultural change with the introduction of open source collaboration methods. IBM Systems Journal Vol. 44, No. 2, pp. 215-225.

Nonaka, I (1991) The Knowledge-Creating Company, Harvard Business Review (69:6), pp. 96-104.

Nov, O (2007) What motivates wikipedians? Communications of the ACM, 50(11), 60-64.

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.

O'Mahony, S (2005) Non-Profit Foundations and Their Role in Community-Firm Software Collaboration, in Perspectives on Free and Open Source Software, J. Feller B. Fitzgerald, S. Hissam, and K. Lakhani (eds.), Cambridge, MA: MIT Press, pp. 393-414.

O'Reilly, T (1999) Lessons from open source software development. Communications of the ACM Vol. 42, No. 4, pp. 33-37

Parnas, DL (1972) On the criteria to be used in decomposing systems into modules. Communications of the ACM Vol. 15, No. 12, pp. 1053-1058.

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

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.

Rigby, PC, German, DM and Storey, MA (2008) Open source software peer review practices: A case study of the Apache Server. In Proceedings of the 30th International Conference on Software Engineering. ACM, pp. 541-550.

Rigby, PC, Cleary, B, Painchaud, F, Storey, MA and German, DM (2012) Contemporary peer review in action: Lessons from open source development. IEEE Software Vol. 29, No. 6. pp. 56-61.

Robbins, J (2005) Adopting open source software engineering (OSSE) practices by adopting OSSE tools. In Perspectives on Free and Open Source Software, Feller, J, Fitzgerald, B, Hissam, SA, and Lakhani, KR (Eds.), MIT Press, pp. 245-264.

Robles, G, Scheider, H, Tretkowski, I, and Weber, N (2001) Who is doing it? A research on libre software developers. *Research Paper, TU Berlin, August*.

Royce, WW (1987) Managing the development of large software systems. In Proceedings of the 9th International Conference on Software Engineering. Originally published in Proceedings of WESCON'70. pp. 328-338.

Scacchi, W (2002) Understanding the Requirements for Developing Open Source Software Systems, IEE Proceedings—Software Vol. 149, No. 1, pp. 24-39.

Scacchi, W (2004) Free and open source development practices in the game community. IEEE Software Vol. 21, No. 1, pp. 59-66.

Schenk, E and Guittard, C (2011) Towards a Characterization of Crowdsourcing Practices, Journal of Innovation Economics, Vol. 1, No. 7.

Senyard, A and Michlmayr, M (2004) How to have a successful free software project. In Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC).

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

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

Stol, K and Fitzgerald, B (2014b) Researching Crowdsourcing Software Development: Perspectives and Concerns. In Proceedings of the First International Workshop on Crowdsourcing in Software Engineering (CSI-SE) co-located with ICSE'14, Hyderabad, India

Stol, K and Fitzgerald B (2014c) Research Protocol for a Case Study of Crowdsourcing Software Development. Technical Report. University of Limerick.

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

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

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

Tajedin, H and Nevo, D (2013) Determinants of success in crowdsourcing software development. In Proceedings of SIGMIS Computer and People Research. Cincinnati, OH, USA.

Tiwana, A and Keil, M (2009) Control in Internal and Outsourced Software Projects, Journal of Management Information Systems (26:3), pp 9-44.

Torkar, R, Minoves, P and Garrigos, J (2011) Adopting free/libre/open source software practices, techniques and methods for industrial use. Journal of the Association of Information Systems Vol. 12, No. 1, pp. 88-122.

Torvalds, L (1999) The Linux edge. In Open Sources: Voices from the Open Source Revolution, Chris DiBona, Sam Ockman, and Mark Stone (Eds.), O'Reilly Media.

Torvalds, L (2000) Linux Kernel mailing list. https://lkml.org/lkml/2000/8/25/132.

Van der Linden, F, Schmid, K and Rommes, E (2007) Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer.

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.

Venkatesh, V, Brown, S A, and Bala H (2013) Bridging the qualitative–quantitative divide: guidelines for conducting mixed methods research in information systems. MIS Quarterly 36(1), 21-54.

Vitharana, P, King, J and Chapman, HS (2010) Impact of internal open source development on reuse: Participatory reuse in action. Journal of Management Information Systems 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, M. W. (2012) Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. MIS Quarterly, Vol. 36, No. 2, pp. 649-676.

Vukovic, M (2009) Crowdsourcing for Enterprises. SERVICES.

Wesselius, J (2008) The bazaar inside the cathedral: Business models for internal markets. IEEE Software, Vol. 25, No. 3, pp. 60-66.

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.

Zhao, Y and Zhu, Q (2012) Evaluation on crowdsourcing research: Current status and future direction, Information Systems Frontiers, April.