Engineering Open Innovation – towards a Framework for Fostering Open Innovation

Krzysztof Wnuk and Per Runeson

Department of Computer Science, Lund University, Sweden
`{Krzysztof.Wnuk,Per.Runeson}@cs.lth.se`,
`http://serg.cs.lth.se`

Abstract. Open innovation is an emerging innovation paradigm that can greatly accelerate technical knowledge innovation in software companies. The increasing importance and density of software in today's products and services puts extensive pressure on excelling the discovery, description and execution of innovation. Despite that, software engineering literature lacks methods, tools and frameworks for full exploitation of technological advantages that open innovation can bring. This paper proposes a software engineering framework, designed to foster open innovation by designing and tailoring appropriate software engineering methods and tools. Furthermore, this paper discusses the methodological and process dimensions and outlines challenge areas that should be reviewed when transitioning to software engineering driven open innovation.

Key words: open innovation, software engineering framework, literature study, methodological and process study

1 Introduction

The development of software products is mainly driven by *innovation* [1]; i.e. the novel utilization of technical knowledge to develop new products and services. For example, Volvo, the truck company, estimates that 90 % of new innovations are in the field of electronics, and 80 % thereof is software[1]. Similarly, most of the innovation and resulting market success at Siemens originates from software [2].

A majority of the innovation within software intensive products is implemented in software and increasingly dependent on a new paradigm called *Open Innovation*(OI), which typically, but not necessarily is implemented using *Open Source Software* (OSS). In recent years, the influence of OI has become significant in the development and evolution of software products and services, e.g. in the Android ecosystem. OI implies that no single firm or other actor is sufficient for developing new products and services; instead several loosely connected organizational actors interplay. The OI context is characterized by: (1) collaborative efforts over single company/person work, (2) loose connections over contractual agreements, (3) demonstrated results over predictions and (4) bottom-up specifi-

[1] http://www.swedsoft.se/Swedsoft_SRA_2010.pdf

cation approaches. These differences in characteristics make software engineering practices significantly challenging.

Software, with its flexibility in multiple aspects, is an excellent enabler for innovation. On the other hand, this flexibility must be managed, not to lose control over the software. Hence, software engineering (SE) in an OI context is a major research challenge since engineering practices for in-house, contract-based development may not be feasible. Therefore, we set out to define a framework to support software engineering for open innovation.

This paper presents result from an exploratory literature study that constitutes the first step of our efforts towards building a framework that *aims to synthesize a scientifically founded software engineering framework for open innovation.* We review existing literature and map existing research as a basis for new research [3]. Our goal is to develop new or adapt existing practices into a framework that meet challenges in the multi-organizational, heterogeneous open innovation context.

This paper is structured as follows: Section 2 outlines definitions and background, Section 3 presents the literature review results. Section 4 outlines the engineering open innovation framework while Section 5 outlines future research directions and concludes the paper.

2 Background and definitions

Open innovation was introduced by Chesbrough [4] as "a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology" [4]. A more recent definition of open innovation by Lichtenthaler [5] focuses on "systematically relying on a firm's dynamic capabilities of internally and externally carrying out the major technology exploitation and acquisition tasks along the innovation process". This increased external stream of knowledge may result in more *disruptive* innovations (i.e. taking large steps towards something new) coexisting with *sustaining* innovations (i.e. continuously improving solutions to create more value [6, 7]). This, in turn, increases the pressure for software engineering methods that can cope with increased interoperability, flexibility and significantly enhanced engineering characteristics.

The OECD defines four main types of innovation in the Oslo manual [8], *product, process, marketing* and *organizational.* The inherent characteristics of software enable novel approaches to all four types of innovation. Product innovation (the software itself) may bring new value to customers at negligible production and distribution costs. Process innovation involves new means of developing software, e.g. OSS communities. Marketing innovations include new business models, e.g. offering services at the price of being exposed to ads or sharing information. Organizational innovation includes new ways to work across different actors, where open innovation is an example. All four types are interconnected and therefore have to be researched in context.

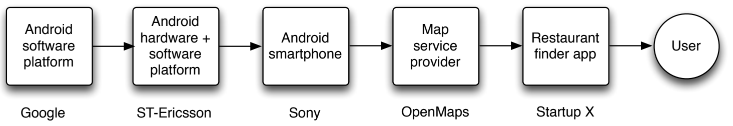

Fig. 1. The Android eco-system as an example of a (partly) open innovation value chain.

An example of a (partly) open innovation value chain is illustrated through the Android eco-system in Figure 1. Google provides the (semi-)open Android software platform to device suppliers, which in turn adapt the platform to their and their customers' specific needs. The Android platform contains million lines of code adapted for hundreds of products, various hardware products, wireless network standards and database systems. Service providers may offer map services tailored to the Android phones (e.g. OpenMaps), which other app providers may utilize to derive specialized map services, e.g. a restaurant finder by a local startup company. This innovation chain is driven by several actors in collaboration and dependencies on one another, with different individual goals, time scales, size of organization, techniques etc. As one actor in the chain evolves its parts, others must follow, but no single actor is in full control.

OSS is a mechanism that may embody the principles of open innovation. OSS is not a new phenomenon on its own, however the novelty in recent years is the widespread adoption in industry, where closed innovation used to be the dominating paradigm, which requires new approaches to software development [9]. A recent literature survey about OSS identified a research gap in the area of OSS and open innovation [10]. Software ecosystems could also be one of the types of open innovation where a network of collaborators constituting an ecosystem is open [11]. Finally, there is a need for processes supporting large-scale development with open innovation; companies typically apply the same processes as used in closed innovation [9], while there certainly are opportunities for more tailored processes to improve the efficiency and effectiveness of the development [12].

3 Literature review

We conducted a literature review using a hybrid approach by combining database search (Compendex and Inspect), and snowball sampling. We selected a mapping study approach because our main goal was to explore the area rather than synthesize the current state of the art [3]. We used the following search queries (searched in titles, abstracts or subjects):

- open innovation AND requirements engineering
- open innovation AND software design
- open innovation AND software development
- open innovation AND software testing
- open innovation AND software
- software engineering AND innovation
- methodology OR method AND open innovation AND software

Table 1. Classification of papers; research type according to Wieringa et al. [14].

Research type	Techniques	Tech & Proc	Processes	Methods
Validation	[15]	[16]		
Evaluation[1]		[17]	[18]	[19]
Solution	[20]	[21] [22]	[23] [24] [25] [26] [27] [28] [29] [30] [31] [11]	
Conceptual[2]	[32]		[33]	
Opinion	[34] [35]	[36] [37]		[38] [39]
Experience		[40] [41] [42]	[43] [44] [45]	

[1] The original classification [14] only covers engineering research, while we here classify also observational empirical studies as 'Evaluation', as they evaluate current practice.
[2] Called 'Philosophical' originally.

– open innovation AND software as a service OR saas
– open innovation AND software AND eco system
– innovation AND software AND eco system

The above queries returned 1480 records that we checked by reading titles and, if in scope, also abstracts. 32 papers were selected for full reading. We categorized these papers into two dimensions; the first dimension categorized the articles according to the topic of *techniques, processes* and *research methods* while the second dimension of research type was created based on systematic mapping guidelines [13, 14]. The classification is summarized in Table 1.

3.1 Software engineering techniques for open innovation

Five papers discussed or suggested a specific software engineering method or technique for an open innovation context. El-Sharjawy and Schmid proposed and experimentally evaluated an approach for deriving creative triggers from a knowledge map of requirements [15], in a paper of validation type. We identified two opinion papers: Petrenko and Petrenko discussed the challenges of using formal methods to analyze requirements and work with legacy code that can foster what they called Innovation Economy [34] while Grube and Schmid suggested which creativity techniques are appropriate for requirements engineering [35]. A conceptual paper by Kauppinen et al. argued that practitioners do not see requirements engineering as a creative process and suggested focusing on "unarticulated needs" to unlock more innovation from requirements engineering processes and techniques [32]. A solution using social networks to document ideas and thus foster open innovation was proposed by Singer et al. [20].

The remaining 9 papers in the Techniques category were also touching upon the Processes category (see Section 3.2). We found two opinion papers: one focusing on how to avoid innovation lock-in from a pre-planned variability model of a software product line [36] and one focusing on sharing the source code and opening bug-tracking tools with the clients [37]. Next, an experience paper by Copeland suggested new ways of communicating the information about testing [40]. Theodore et al. studied how outsourcing can inject tangible forms of innovation [41] presenting an example of innovative testing methods (unfortunately without detail about the methods) that originated from such a collaboration and improved time-to-test by 90% and reduced cost by 70%.

Five papers, that we categorized in both the Techniques and Processes categories, focused on software development. One experience paper focused on investigating how software startups can use opportunistic and pragmatic reuse to develop innovative products [42]. One evaluation paper focused on how agile development processes can become more open by utilizing outside-in and inside-out process [17] models. Two solution papers proposed giving developers more authority on *when* and *how* to use innovative software development techniques [21] and utilizing prototyping, agile methods, developers using products they develop and sharing knowledge to foster innovation [22]. Finally, one validation paper experimentally concluded that leaving the developers free to define their own development processes is beneficial from the innovation diffusion perspective [16].

3.2 Software engineering processes that foster open innovation

We categorized 14 articles as only concerning the software engineering processes category. Ten papers presented various solutions, among which three supported innovation selection processes by an audition-inspired process for screening, refining and selecting the most promising innovations [23], a knowledge management scheme that supports transition of software innovations (also legacy systems) to enterprise systems [24] and a method based on neuro-fuzzy decision trees for innovation projects selection [25]. The prototyping approach to open innovation was explored in two publications: Eklund and Bosch suggested turning the entire R&D process into an innovation experiment system with direct customer involvement in design decisions [26] while Bullinger et al. proposed an open prototyping solution [27]. Misra et al. advocated using a GQM-based method to derive a measurement framework for software innovation process [28] while Felfernig et al. proposed utilizing artificial intelligence for open innovation in e-government contexts [29]. Jansen [11] focused on measuring the degree of openness of a software organization. Two publications proposed solutions to explore open innovation communities by visualizing people-innovation-networks [30] or modeling service systems in terms of communities of co-innovation [31].

Among experience papers in this category, Hanssen [43] reported lessons learned from opening up a software product line, observing that it improved the ability to catch tacit requirements (related to unarticulated needs mentioned by Kauppinen et al. [32]. Yilmaz discovered, based on a simulation, that decentralized coordination schemes as well as moderate degrees of assertiveness result in a higher incidence of innovation for open source software communities [44]. Carrero [45] discussed how service delivery platforms enable service providers to achieve open innovation.

In a conceptual paper, Lyytinen and Damsgaard observed that the six conjectures of diffusion of innovation need to be revisited for complex and networked

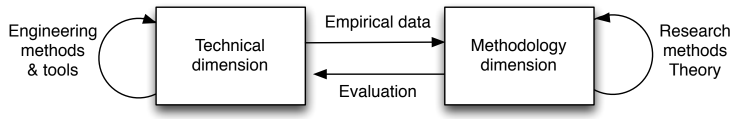

Fig. 2. Overview of framework dimensions and their relations

IT systems and additional issues should be considered [33]. In an evaluation paper, Lane et al. concluded that finding a good balance between art and science, allowing failures as a part of learning process, and trying different approaches are important success factors for software innovative process [18]. To summarize, although ten papers presented solutions ([11] [23] [24] [25] [26] [27] [28] [29] [30] [31]) only one evaluation paper was identified [18].

3.3 Software engineering research methodologies

An opinion paper by Bayer and Melone discussed challenges in applying diffusion theory for software engineering technological innovations, outlining seven limitations [38]. Prechelt and Oezbek outlined a research method solution for studying open source software process innovation, suggesting that grounded theory is feasible for deriving mini-theories about process innovation, but not focusing strictly on open innovation [39]. In an evaluation paper, Rossi et al. proposed a quantitative instrument (based on stochastic models) for measuring the assimilation gaps in IT innovation [19]. No solution paper was identified in this category.

4 Engineering the OI–SE Framework

The literature review in Section 3 brings supporting evidence for our research efforts. We identified only three evaluation papers, see Table 1, none strictly focusing on open innovation and mostly reporting exploratory evaluations [17, 18, 19]. Both identified validation papers focused on internally derived innovations [15, 16]. Among 12 identified solution papers, only four are devised for open innovation [29, 30, 31, 11].

We address the open innovation issues from a product and process innovation point of view, in the intersection with marketing and organizational innovation perspectives [8]. Based on the literature review, we propose a framework having two main dimensions, one *technical* and one *methodological*. The dimensions are mutually dependent, as the technical dimension is the empirical basis for the methodological part, and the methodological part is needed for the technical part. Figure 2 gives an overview of the project and its dimensions, which is inspired by Hevner's design science model [46] and Wieringa et al [47].

4.1 Technical dimension

The technical dimension has two interrelated parts, see Figure 2: 1) software engineering techniques, such as requirements engineering, software design, software development and software testing techniques, and 2) software engineering processes as an integrating part of the above mentioned techniques. We believe (providing empirical evidence for our hypotheses is one of the goals of the framework) that using appropriate techniques and analyzing the outcomes of these techniques can foster open innovation.

Requirements: requirements engineering research has evolved from concentrating on the specification problem in the 1990's to pursuing wide and open-ended investigations of the conception and strategic evolution of software in relation to decision-making on enterprise, product/service and project levels. The integrated strategic and tactical decision-making needed in large-scale engineering projects is a key challenge for software engineering for open innovation [48]. However, papers identified during the literature review focus mainly on supporting the discovery of innovation [15, 20, 32, 34].

We build on previous research and focus on release-planning [48, 49], stakeholder analysis, trade-off between effort (cost) and value and the degree of innovation in candidate features needed in evolving systems, to take significant future market shares in open innovation software development [48, 50].

Design: efficient software architectures for open innovation should enable open and seamless integration of externally acquired modules. Among idenfitied papers, Böckle [36] postulated that software product lines are generally hindering innovation and that variability locks in innovation as it focuses on reusing the same code and thus minimizing creative adaptations. Moreover, software product lines are designed with a premise that the same code will be used for a long time, which directly hinders disruptive innovation. Similarly, a software design with high coupling may be hindering open innovation as new modules and sub-systems could not be easily integrated. Thus, there seems to be a need for evolution from traditional SPLs toward software ecosystems [36] which introduced necessary flexibility in an organizational matter.

Development: is pair programming going to result in more innovation that other programming techniques? This is just an example question that should be investigated in the framework. Green suggested [21] that giving developers more authority on when to use the development technique innovations helps to actually use them rather than drop them. Sharing the source code and bug-tracking system [37] or open prototyping [28] also seem to be fostering innovation. However, identified studies focus on development processes rather than techniques [16, 17, 21, 28, 22, 37, 26]. Among identified studies, Jansen [11] presented a model for establishing the degree of openness of a software organization. Further empirical investigations are needed to yield concrete examples of which development techniques foster open innovation.

Testing: Software testing in open innovation has a dual role: 1) to verify functions and characteristics of open components and services, supplied by others, and 2) to verify functions and characteristics of services delivered to stakeholders higher up in the value chain; ultimately end users. Since specifications and contracts are sparse in the open innovation context, they have to be defined otherwise.

Test driven development is a method which has proven feasible in a dynamic practice [51]. Its combination of specification and test [52] can be tailored for use in open innovation. Software engineering in open innovation tends to be very iterative, and thus *regression testing* is a key issue. We build on previous findings [53, 54] and adapt test selection and prioritization approaches to open innovation, as well as using it to detect changes in the environment [55].

Efficient processes: The move towards agile processes have substantially changed software engineering practices during the last decade. The efficiency of some practices have been empirically demonstrated [56], and the positive attitudes of engineers "being in control" are witnessed [21, 22, 55]. Still, the efficiency of agile (or other) practices in open innovation is not targeted; our review identified only one evaluation paper [17] and one validation paper [16] in this area. Further, most studies either focus on the small *or* the large context, e.g. [23, 24, 26, 43], but in *OI* we have small contexts within the larger context.

We plan to investigate the issue of stakeholders and stakeholder representatives in an open innovation context [55]. Further, mechanisms for synchronizing different actors in the open innovation value chain are planned to be researched [48], based on observations of different actors in open source projects [57]. We also plan to explore how open source practices can support openness across groups and other organizational borders within closed organizations. Findings should be embodied in practice models for small actors in a large context, enabling growth, as previously done on testing practices for small companies [58].

4.2 Methodological dimension

Studying software engineering for open innovation must be an empirical endeavor since we are addressing complex phenomena in the real world. The question is which type of empirical study is to be conducted.

Prechelt and Oezbek conducted four studies on OSS process innovation, and concluded that using mailing list archives was the most efficient research method, compared to direct participation and polling developers for data [39]. The easy access to electronically searchable information is probably one of the key reasons for the popularity on conducting research in OSS archives, independently of whether purpose of the research is open or proprietary software [59].

The scope of the open innovation framework is wider than OSS projects only. This, combined with general advice on using research method triangulation, lead us to propose combining *archival studies* with *participant-observer* studies [60], to enable insights into the dynamics of the open innovation.

Building synthesized knowledge from the empirical observation needs replication of studies [61], synthesis of findings from several empirical studies [62], as well as work in theory building [63]. However, these needs are general for software engineering, and not specific to the open innovation aspects.

5 Conclusions

Our review of the related literature remains incomplete (the results presented in Section 3 are preliminary). However, we can identify three research areas where both researchers and practitioners can benefit: (1) providing practical guidelines for selecting the most appropriate requirements engineering, software design, software development and software testing techniques for open innovation, (2) researching software engineering processes that support open innovation and (3) finding new research methodologies for conducting software engineering research in open innovation contexts.

In the intersection between different types of innovation (product, process, marketing and organizational [8]), there is a significant potential for software innovations. Especially the intersection between software engineering and open innovation lacks empirical research, as shown in our literature review, which we hope and aim that our research framework will foster.

Future work should focus on investigating if the presented framework could support the intrinsic creativity and unpredictability of innovation. Furthermore, we plan to explore and possible identify activities that can not be clearly categorized into the four traditional software development process steps. As we only searched Compendex and Inspect, more databases should be searched and the literature review should be replicated in a systematic way. Finally, we plan to investigate the possible relationships between the business models and open innovation.

Acknowledgements: The authors would like to thank David Callele for excellent reviews of this paper. This work is founded by the SYNERGIES project, Swedish National Science Foundation, grant 621-2012-5354.

References

1. Quinn, J.B., Baruch, J.J., Zien, K.A.: Software-based innovation. Sloan Management Review **37**(4) (1996) 11–24
2. Achatz, R.: Product line engineering at siemens – challenges and success factors: A report on industrial experiences in product line engineering. In: Software Product Line Conference (SPLC), 2011 15th International. (aug. 2011) 10–11
3. Kitchenham, B.A., Budgen, D., Brereton, O.P.: Using mapping studies as the basis for further research - a participant-observer case study. Information & Software Technology **53**(6) (2011) 638–651
4. Chesbrough, H.: Open Innovation: The new imperative for creating and profiting from technology. Boston: Harvard Business School Press (2003)
5. Lichtenthaler, U.: Open innovation in practice: An analysis of strategic approaches to technology transactions. IEEE Trans. on Eng. Mgmt **55**(1) (2008) 148–157
6. Khurum, M., Gorschek, T., Wilson, M.: The software value map an exhaustive collection of value aspects for the development of software intensive products. Journal of Software: Evolution and Process (2012) n/a–n/a

7. Khurum, M., Aslam, K., Gorschek, T.: A method for early requirements triage and selection utilizing product strategies. In: Proc. of the 4th Asia-Pacific Software Engineering Conf. APSEC '07, IEEE Computer Society (2007) 97–104
8. : Oslo Manual – Guidelines for collecting and interpreting innovation data. 3rd edn. OECD and Eurostat (2005)
9. Höst, M., Orucevic-Alagic, A., Runeson, P.: Usage of open source in commercial software product development - findings from a focus group meeting. In: 12th Int. PROFES Conference. Volume 6759 of LNBIP., Springer (2011) 143–155
10. Höst, M., Orucevic-Alagic, A.: A systematic review of research on open source software in commercial software product development. Information and Software Technology **53**(6) (2011) 616–624
11. Jansen, S., Brinkkemper, S., Souer, J., Luinenburg, L.: Shades of gray: Opening up a software producing organization with the open software enterprise model. Journal of Systems and Software **85**(7) (2012) 1495–1510
12. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and mozilla. ACM Trans. Softw. Eng. Methodol. **11**(3) (2002) 309–346
13. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proc. 12th Int. Conf. Evaluation and Assessment in Soft. Eng. EASE'08, UK, British Computer Society (2008) 68–77
14. Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirement Engineering **11** (2006) 102–107
15. El-Sharkawy, S., Schmid, K.: A heuristic approach for supporting product innovation in requirements engineering: a controlled experiment. In: Proc. of the 17th REFSQ Conference, Heidelberg, Springer-Verlag (2011) 78–93
16. Tortorella, M., Visaggio, G.: Empirical investigation of innovation diffusion in a software process. International Journal of Software Engineering and Knowledge Engineering **09**(05) (1999) 595–621
17. Conboy, K., Morgan, L.: Beyond the customer: Opening the agile systems development process. Inf. and Soft. Techn. **53**(5) (2011) 535–542
18. Lane, J.A., Boehm, B., Bolas, M., Madni, A., Turner, R.: Critical success factors for rapid, innovative solutions. In: Proc. of the Int. Conf. on New modeling concepts for today's software processes. ICSP'10, Heidelberg, Springer-Verlag (2010) 52–61
19. Rossi, B., Russo, B., Succi, G.: Path dependent stochastic models to detect planned and actual technology use: A case study of openoffice. Inf. and Soft. Techn. **53**(11) (2011) 1209–1226
20. Singer, L., Seyff, N., Fricker, S.A.: Online social networks as a catalyst for software and it innovation. In: Proc. of the 4th Int. workshop on Social soft. eng. SSE '11, New York, USA, ACM (2011) 1–5
21. Green, G., Hevner, A.: The successful diffusion of innovations: guidance for software development organizations. IEEE Soft. **17**(6) (nov-dec 2000) 96–103
22. Moe, N., Barney, S., Aurum, A., Khurum, M., Wohlin, C., Barney, H., Gorschek, T., Winata, M.: Fostering and sustaining innovation in a fast growing agile company. In: Product-Focused Software Process Improvement. Volume 7343 of LNCS. Springer Berlin Heidelberg (2012) 160–174
23. Gorschek, T., Fricker, S., Palm, K.: A lightweight innovation process for software-intensive product development. IEEE Soft. **27**(1) (jan-feb 2010) 37–45
24. Corbin, R.D., Dunbar, C.B., Zhu, Q.: A three-tier knowledge management scheme for software engineering support and innovation. Journal of Systems and Software **80**(9) (2007) 1494–1505

25. Hongxia, J., Jianna, Z., Xiaoxuan, C.: The application of neuro-fuzzy decision tree in optimal selection of technological innovation projects. In: Eighth ACIS International SNPD Conference. Volume 3. (aug 2007) 438 –443
26. Eklund, U., Bosch, J.: Architecture for large-scale innovation experiment systems. In: Working IEEE Conf. on Soft. Architecture. (2012) 244–248
27. Bullinger, A.C., Hoffmann, H., Leimeister, J.M.: The next step - open prototyping, Helsinki, Finland (2011)
28. Misra, S.C., Kumar, V., Kumar, U.: Goal-driven measurement framework for software innovation processes. In Arabnia, H.R., Reza, H., eds.: Proc. of the Int. Conf. on Soft. Eng. Research and Practice, CSREA Press (2005) 710–716
29. Felfernig, A., Russ, C., Wundara, M.: Toolkits supporting open innovation in e-government. In: Proc. of the Sixth Int. Conf. on Enterprise Information Systems, Porto, Portugal (2004) 296–302
30. Friess, M., Groh, G., Reinhardt, M.: Supporting open innovation communities by an interactive network visualization. In: Proceedings of the IADIS International Conferences, New York, NY, USA (2010) 23–8
31. Janner, T., Schroth, C., Schmid, B.: Modelling service systems for collaborative innovation in the enterprise software industry - the st. gallen media reference model applied. In: IEEE Int. Conf. on Services Computing. Volume 2. (2008) 145–152
32. Kauppinen, M., Savolainen, J., Mannisto, T.: Requirements engineering as a driver for innovations. In: 15th IEEE Int. Req. Eng. Conference. (2007) 15–20
33. Lyytinen, K., Damsgaard, J.: What's wrong with the diffusion of innovation theory. In: Fourth Working Conf. on Diffusing Software Products and Process Innovations, The Netherlands, Kluwer, B.V. (2001) 173–190
34. Petrenko, A.K., Petrenko, O.L.: Formal methods and innovation economy: Facing new challenges. In: Proc. of the 6th IEEE Int. Conf. on Software Engineering and Formal Methods. SEFM '08, Washington, DC, USA, IEEE CS (2008) 367–371
35. Grube, P., Schmid, K.: Selecting creativity techniques for innovative requirements engineering. In: 3rd Int. Workshop on Multimedia and Enjoyable Requirements Engineering. (sept. 2008) 32–36
36. Bockle, G.: Innovation management for product line engineering organizations. In Obbink, H., Pohl, K., eds.: Software Product Lines. Volume 3714 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 124–134
37. Riepula, M.: Sharing source code with clients: A hybrid business and development model. Software, IEEE **28**(4) (july-aug. 2011) 36 –41
38. Bayer, J., Melone, N.: A critique of diffusion theory as a managerial framework for understanding adoption of software engineering innovations. Journal of Systems and Software **9**(2) (1989) 161–166
39. Prechelt, L., Oezbek, C.: The search for a research method for studying oss process innovation. Empirical Softw. Engg. **16**(4) (August 2011) 514–537
40. Copeland, P.: Google's innovation factory: Testing, culture, and infrastructure. In: Third Int. Conf. on Soft. Testing, Verification and Validation. (april 2010) 11–14
41. Forbath, T., Brooks, P., Dass, A.: Beyond cost reduction: Using collaboration to increase innovation in global software development projects. In: IEEE Int. Conf. on Global Soft. Eng (ICGSE). (aug. 2008) 205–209
42. Jansen, S., Brinkkemper, S., Hunink, I., Demir, C.: Pragmatic and opportunistic reuse in innovative start-up companies. IEEE Soft. **25**(6) (nov.-dec. 2008) 42–49
43. Hanssen, G.K.: Opening up software product line engineering. In: Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering. PLEASE '10, New York, NY, USA, ACM (2010) 1–7

44. Yilmaz, L.: An agent simulation study on conflict, community climate and innovation in open source communities. IJOSSP **1**(4) (2009) 1–25
45. Carrero, M.: Innovation for the web 2.0 era. Computer **42**(11) (2009) 96–8
46. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly **28**(1) (2004) 75–105
47. Wieringa, R., Daneva, M., Condori-Fernández, N.: The structure of design theories, and an analysis of their use in software engineering experiments. In: Proc. 5th Int. Symp. on Empirical Software Engineering and Measurement, IEEE (2011) 295–304
48. Wnuk, K., Pfahl, D., Callele, D., Karlsson, E.A.: How can open source software development help requirements management gain the potential of open innovation: an exploratory study. In: Proc. of the ESEM 2012 Symposium, New York, USA, ACM (2012) 271–280
49. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., och Dag, J.N.: An industrial survey of requirements interdependencies in software product release plannin. In: 5th IEEE Int. Symposium on Req. Eng., Toronto, Canada (2001) 84–93
50. Wnuk, K., Callele, D., Regnell, B.: Guiding requirements scoping using roi: Towards agility, openness and waste reduction. In: 18th IEEE International Requirements Engineering Conference, Sydney, Australia (2010) 409–410
51. Williams, L., Maximilien, E., Vouk, M.: Test-driven development as a defect-reduction practice. In: 14th Int. Symp. on Soft. Reliability Eng. (2003) 34–45
52. Regnell, B., Runeson, P.: Combining scenario-based requirements with static verification and dynamic testing. In: 4th Int. Working Conference Requirements Engineering: Foundation for Software Quality. (1998) 195–206
53. Engström, E., Runeson, P.: A qualitative survey of regression testing practices. In: 11th PROFES Conference. Volume 6156 of LNCS., Springer (2010) 3–16
54. Engström, E., Runeson, P., Skoglund, M.: A systematic review on regression test selection techniques. Information and Software Technology **52**(1) (2010) 14–30
55. Karlström, D., Runeson, P.: Integrating agile software development into stage-gate managed product development. Emp. Soft. Eng. **11**(2) (2006) 203–225
56. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review. Information and Software Technology **50**(9-10) (2008) 833–859
57. Orucevic-Alagic, A., Höst, M.: A case study on the transformation from proprietary to open source software. In: IFIP Advances in Information and Communication Technology. Volume 319., Springer (2010) 367–372
58. Karlström, D., Runeson, P., Nordén, S.: A minimal test practice framework for emerging software organizations. Soft. Testing, Verification and Reliability **15**(3) (2005) 145–166
59. Robinson, B., Francis, P.: Improving industrial adoption of software engineering research: a comparison of open and closed source software. In: Proceedings of the ESEM Conference, 2010, Bolzano, Italy, ACM (2010)
60. Runeson, P., Höst, M., Rainer, A.W., Regnell, B.: Case Study Research in Software Engineering. Guidelines and Examples. Wiley (2012)
61. Schmidt, S.: Shall we really do it again? the powerful concept of replication is neglected in the social sciences. Review of General Psych. **13**(2) (2009) 90–100
62. Cruzes, D.S., Dybå, T., Runeson, P., Höst, M.: Case studies synthesis: Brief experience and challenges for the future. In: Proceedings of the 2011 ESEM symposium, Banff, Canada (2011)
63. Sjøberg, D.I.K., Dybå, T., Anda, B., Hannay, J.E.: Building theories in software engineering. In: Guide to Advanced Empirical Soft. Eng. Springer-Verlag (2008)