Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2016, Article ID 5126069, 25 pages
http://dx.doi.org/10.1155/2016/5126069

Review Article

Innovation Drivers and Outputs for Software Firms: Literature Review and Concept Development

Jeremy Rose[1] and Brent Furneaux[2]

[1] *University of Skövde, P.O. Box 408, 541 28 Skövde, Sweden*
[2] *School of Business and Economics, Maastricht University, P.O. Box 616, 6200 MD Maastricht, Netherlands*

Correspondence should be addressed to Jeremy Rose; jeremy.rose@his.se

Received 1 September 2015; Revised 21 December 2015; Accepted 13 January 2016

Academic Editor: Luigi Lavazza

Software innovation, the ability to produce novel and useful software systems, is an important capability for software development organizations and information system developers alike. However, the software development literature has traditionally focused on automation and efficiency while the innovation literature has given relatively little consideration to the software development context. As a result, there is a gap in our understanding of how software product and process innovation can be managed. Specifically, little attention has been directed toward synthesizing prior learning or providing an integrative perspective on the key concepts and focus of software innovation research. We therefore identify 93 journal articles and conference papers within the domain of software innovation and analyse repeating patterns in this literature using content analysis and causal mapping. We identify drivers and outputs for software innovation and develop an integrated theory-oriented concept map. We then discuss the implications of this map for future research.

1. Introduction

Innovation has become a mantra for organizations in almost every industry and for individuals working in almost every organizational function. Large organizations are appointing innovation managers and leading universities are establishing centres for innovation research. Within the software development industry, the pace of technological change fosters a particular sense of urgency surrounding the need to innovate. This drives innovation toward the heart of the business strategies of software development organizations [1]. Rose [2] argues that the forces of globalisation, standardisation, and industrialisation are pressing software development firms in developed countries to become increasingly reliant on their innovation skills. Globalisation can shift software development activities to countries with lower labour costs while standardisation and industrialisation ensure that the functional scope of existing software can be readily extended through relatively modest changes performed by less skilled labour. The software industry is thus following the path of other engineering industries that have been moving routine tasks to developing countries, leaving developed countries to compete through innovation. At the same time, Pikkarainen et al. [1] argue that software innovation differs from other forms of innovation. Software is intangible and highly malleable, has a low market entry threshold, and often depends on the input of users and experts. They also point out that software development organizations are extremely diverse in nature, having very different innovation needs and styles. Whereas large organizations such as Apple and Google are dependent on innovation to capture market share, small start-ups depend on innovation to carve out profitable market niches [3, 4]. As a result, understanding the unique nature of software innovation and the factors that drive it is of considerable importance to both research and practice.

The research community has been slow to articulate and respond to the need for better understanding of software innovation. Traditional software development research has tended to focus on issues of automation and efficiency [2] while the management and information system (IS) innovation literature has a complementary but distinct focus on organizational innovations that use information technology

(IT) as a driving force or catalyst. Finally, innovation research in other disciplines has largely overlooked the software development industry. Nevertheless, contributions from various perspectives have begun to examine some elements of the story. To give some examples, Avital and Te'eni [5] have developed the concept of generative capacity through their examination of the role of creativity in IS design, Carlo et al. [6] adopt a knowledge-oriented perspective that draws on absorptive capacity literature, Hoegl and Gemuenden [7] expose the role of teamwork, and Aaen [8] contributes a framework for system design. However, these authors do not reference each other such that an integrated view of how absorptive capacity may be translated through effective teamwork (harnessed through a design framework) into improved generative capacity is missing. Rather, these contributions tend to remain isolated such that common themes are not always evident and contributing strands of research have emerged without reference to each other. In this paper we therefore begin the work of synthesizing the literature that examines software innovation.

Innovation describes the creative act and the process of invention that is carried into wider use. It involves the exploitation of new ideas to engender changes in the practices of individuals or groups of individuals such as customers and users [2, 9]. A basic distinction can be made between product innovations as observed in the development of a useful new software application and process innovations such as the introduction of a new software development methodology [10, 11]. A distinction has also been made between radical innovations and incremental innovations where radical innovations produce fundamental changes in the structure of a scientific field or social system while incremental innovations build on existing foundations to clarify current understanding. An analogous distinction has been made between revolutionary and evolutionary innovations. Finally, innovations have been characterized as being either sustaining or disruptive in nature [12]. Sustaining innovations typically help organizations to compete within existing markets and value configurations whereas disruptive innovations lead to the creation of entirely new markets and value networks. Disruptive innovations thus tend to displace rather than complement existing technology. In aggregate, the diversity of views present in the literature serves to highlight the varied nature of innovation and the rather distinct implications that different forms of innovation can have for organizations and their wider environment.

Given the importance of software innovation to organizations and society, we undertake a literature study to synthesize the many detailed observations and assumptions about software innovation, the drivers of such innovation, and the numerous associations between these elements that have been presented in prior work. Research suggests a wide range of influences on software innovation, so it is worth examining two previous attempts at synthesis. Rose [2] draws from the academic literature to organize prior work into the categories of software trajectories and windows, community and network, the innovative software product, development process, personal creativity, teamwork, tools and techniques, and evaluation. In contrast, Pikkarainen et al. [1] offer a more practice-oriented synthesis, proposing eight arts of software innovation that include focusing, idea harvesting, idea valuation, openness, optimizing the impact of critical experts, crafting smart products, innovation stimulation, and innovation incubation. We follow the practice of these contributors and focus on the software team, the software team leaders and managers, the artefacts or products that they develop, and the processes they use to develop these artefacts in an effort to address the following research questions:

(1) What software innovation outputs can be identified in the literature?

(2) What are the salient drivers of innovation that can be identified?

(3) How are salient innovation drivers and innovation outputs related?

Our analysis spans individuals, teams, and the organizational context to better understand what drives software innovation. The ultimate objective of our work is to provide guidance to practitioners seeking to improve software innovation in their organizations and to provide clear direction to researchers seeking further understanding in this important domain. To these ends we offer a causal framework grounded in prior research and a set of specific research hypotheses that integrate and consolidate the previously disparate body of research surrounding software innovation.

Our study is theoretically, rather than empirically, oriented, aiming to identify and consolidate some of the existing strands of work surrounding software innovation. We present our findings as a concept map that incorporates the major concepts of the field together with relevant associations. The following section outlines our literature selection and analysis strategy. We then introduce the major innovation outcomes derived from this effort, salient contributors to these outcomes (drivers), and the underlying theoretical foundations. We conclude by summarizing the results of our analysis in a model of software innovation and by discussing the limitations and implications of our work for future research.

2. Literature Selection and Analysis Strategy

Systematic literature reviews are an important part of the development of a field, offering synthesis and reflection on previous theoretical work and providing solid grounding for the advancement of knowledge [13, 14]. Such reviews generally include a structured approach to identifying source material and the use of a concept matrix or other analytical frameworks to yield "a coherent conceptual structuring of the topic" [15, page 173]. Further to this, systematic literature reviews rely on the use of clear, documented processes to ensure that any insights offered are trustworthy. The output is typically some form of evaluation, synthesis, or forward-looking guidance for researchers in a field [14]. This study shares some characteristics with a class of literature reviews known as systematic mapping studies [16] since these investigate a larger body of literature with the objective of structuring a research area. Given the relatively unstructured

Table 1: Application of guidelines for systematic literature reviews [13] to the current study.

Guidelines for systematic literature reviews	Current study
A review protocol that specifies the research question being addressed and the methods that will be used to perform the review	Research questions given in Section 1 and three-stage method summarized in Figure 1
A defined and documented search strategy that aims to detect as much of the relevant literature as possible	See Figure 1 and the discussion offered in relation to this figure
Explicit inclusion and exclusion criteria	Articles should follow an accepted scientific research method, be published in reputable outlets, and be concerned with both innovation and building of software
Specification of the information to be obtained from each primary study	Innovation outputs, innovation drivers, and relationships between drivers and outputs

nature of current understanding surrounding software innovation, this paper offers a systematic literature review that structures past research to provide a conceptual map of software innovation and specific research propositions [17]. The review adheres to overall guidelines for systematic literature reviews in the software engineering field as established by Kitchenham and Charters [13] and summarized in Table 1.

We deployed a three-stage analysis strategy to search for relevant literature, code and refine major concepts, and map key links among these concepts. In the first stage we located relevant articles by searching the literature using guidelines offered by Webster and Watson [14]. The second stage involved content analysis of the abstracts of articles identified during stage one [18–20]. The purpose of this analysis was to identify key contributions and address research questions (1) and (2) by iteratively refining key concepts for the study (software innovation outputs and drivers). The final stage addressed research question (3) through causal mapping of the identified set of key drivers that contribute to software innovation outputs. Causal mapping is a widely used technique for identifying conceptual links or relationships that are either explicitly demonstrated or implicitly assumed by a text [21, 22]. Our objective in pursuing causal mapping was to offer preliminary insights into the causal structure that surrounds software innovation. By constructing causal maps we aim to provide a clearer understanding of the underlying drivers that contribute to software innovation outputs and offer valuable insights into the relationships between them. Our literature study is thus a line of argument qualitative synthesis [13] in that we are primarily concerned with what we can infer about a larger topic through analysis of individual contributions that study parts of the issue and subsequent synthesis of the results of this analysis. Figure 1 provides an overview of the search, selection, coding, and mapping process.

The initial search of ISI Web of Knowledge was performed using a series of search strings that had been constructed through experimentation [13] to yield the most extensive search results possible without unduly sacrificing the relevance of these results. This led to the use of the following search words and phrases: "software AND innovation," "software AND creativity," "information AND systems AND innovation," and "information AND systems AND creativity" (5179 hits). The search was then refined by focusing on relevant science, engineering, and business databases (4510 hits) as well as a group of 21 journals iteratively identified as being of importance to the topic (941 hits). These journals were identified using the AIS scholars' basket of eight top information system journals, the AIS ranking list of information system journals, and a peer-reviewed list of scientific journals from the Danish research council (see Table 3). An initial vetting of article titles and abstracts was performed to eliminate obvious mismatches and the search was then expanded by forward and backward chaining using Google Scholar. Two researchers independently examined the relevance of the hits according to the inclusion criteria to insure inclusion quality [16]. All article abstracts for identified articles were retrieved and similarly vetted for quality and relevance to the theme. The search ended when Google Scholar searches yielded no new hits. This method resulted in an EndNote® library consisting of 236 references complete with their abstracts (available on request).

In the second stage of inquiry, content analysis was undertaken on the abstracts of the 236 articles identified during stage one. As a technique, content analysis yields "a relatively systematic and comprehensive summary or overview of the dataset as a whole" [23, page 170]. It operates by observing repeating themes and categorizing them using a coding system. Our use of content analysis is similar to the thematic synthesis described by Cruzes and Dyba [24] in that we also identify recurring themes or issues from multiple studies and then interpret and explain these themes or issues. Categories can be elicited in a grounded manner or can originate from an external source such as a theoretical model [23]. We used an integrated approach [24] in which we deductively derived codes from a starting list and inductively added new codes in a grounded manner. In our case, initial codes and subcodes were created from the major chapter and section headings used by an earlier study [2] (see Table 4). This set of codes provided a good basis for our coding given that this earlier study sought to provide its readers with a thematic categorization of software innovation research. Nonetheless, iterative revision and refinement of our initial code set through open coding were an explicit objective of the coding effort. Some codes that were poorly represented amongst the texts were removed and others redefined. The initial set of codes identified in Table 4 was revised and refined to yield the final code set identified in

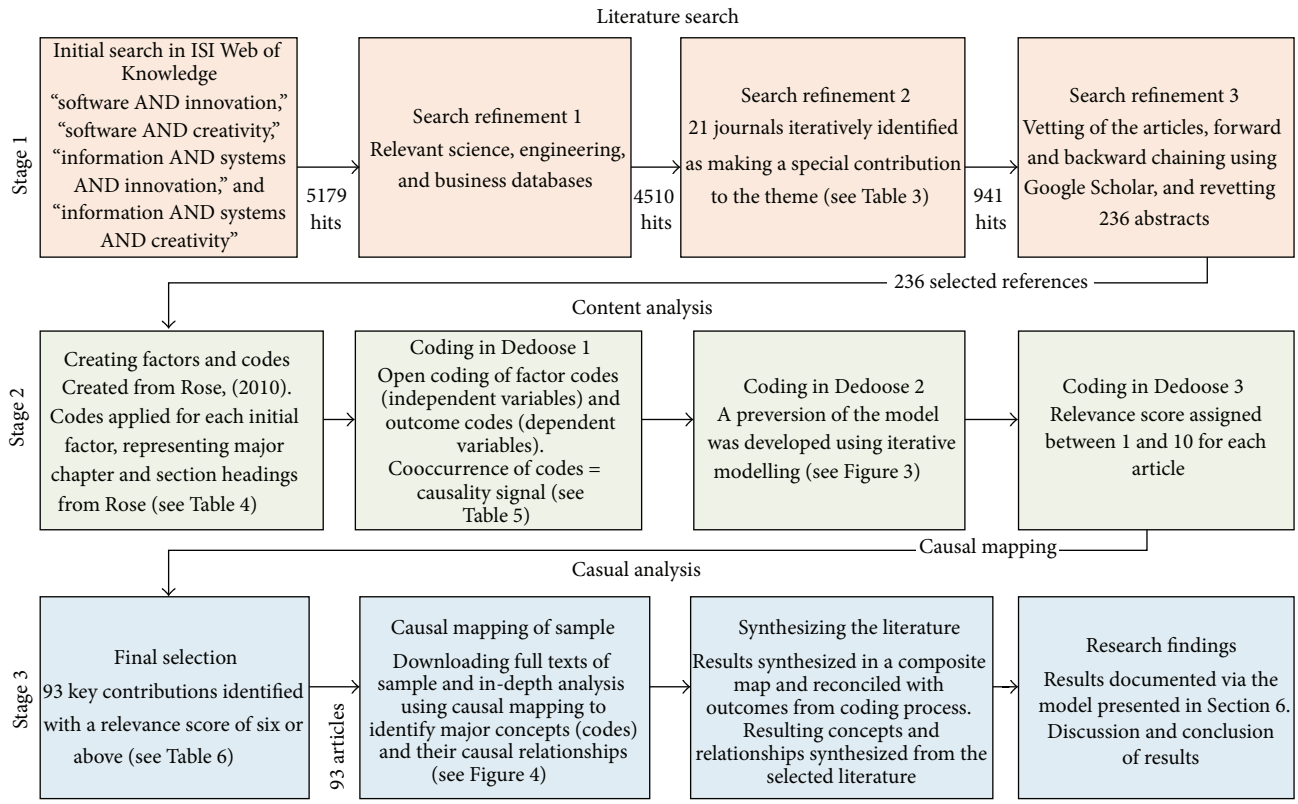

Figure 1: Overall literature selection and analysis strategy.

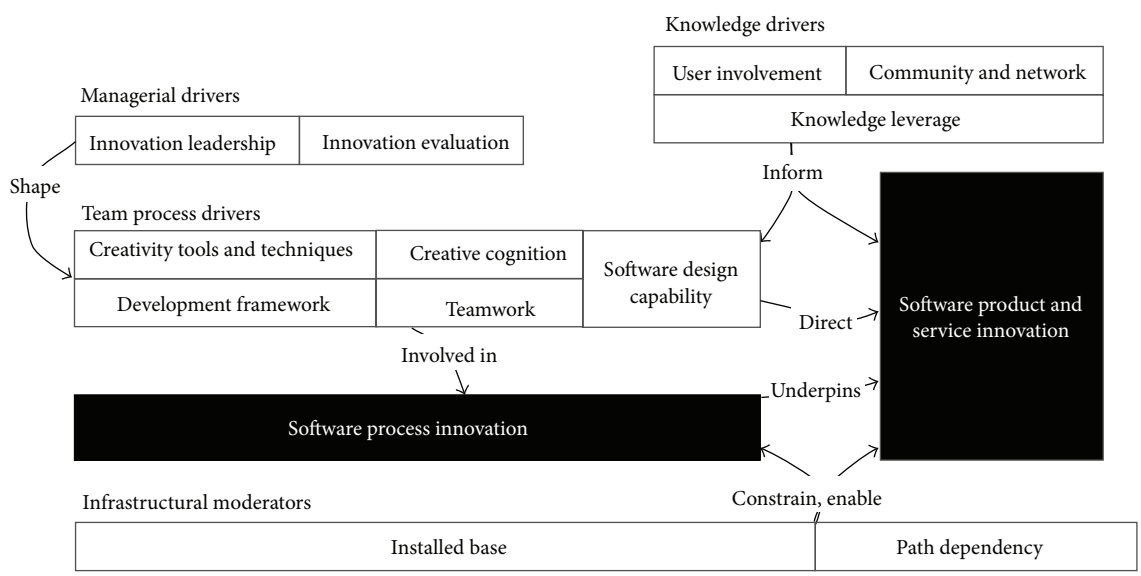

Figure 2: Integrative concept map for software innovation.

Figure 2. The authors and a Ph.D. student collaboratively coded the article abstracts using the online tool Dedoose to facilitate concept development. During the coding process a distinction was made between software innovation outputs and other concepts that might be associated with these innovation outputs (drivers). This permitted us to examine code cooccurrences as an early indication of associations (Table 5) and these associations were then used to guide the construction of initial causal models. Iterative modelling was employed throughout the research effort with Figure 3 providing a causal model developed at this early stage of the inquiry. This model shows elaborated codes and subcodes and identifies provisional linkages among these codes based on their cooccurrence in the text of the articles examined.

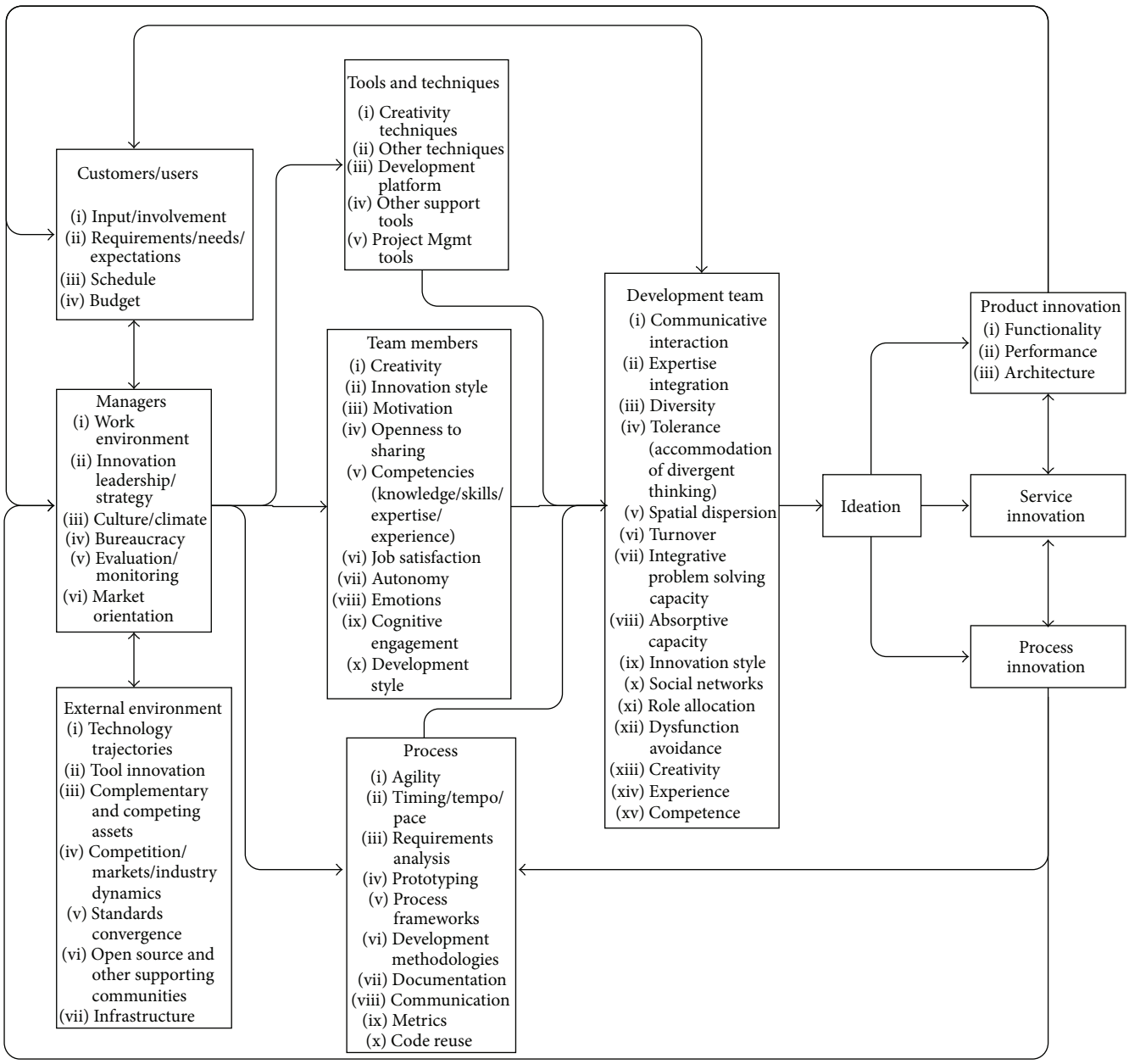

Figure 3: Early version of conceptual model.

While this approach suffers from some obvious limitations, using the cooccurrence data reported in our coding dataset in this way gave us valuable preliminary insights into the causal structure that surrounds software innovation.

In order to facilitate the final third stage of the analysis and refine our initial causal map, all articles were further coded for their relevance to the development of a concept map of software innovation using a scale from one to ten. We scored the articles according to how well they identified patterns of causal relationships between drivers and outputs. We used this score to identify a subset of 93 key contributions that directly addressed our focal interest by selecting those articles with a relevance score of six or above (see Table 6). The full texts of these articles were downloaded and analysed using causal mapping. Causal maps are pictorial representations of "systems of cause-effect relations for the purpose of capturing the structure of human cognition from texts" which can be "examined for patterns, theory building, or hypothesis testing" [21]. Causal associations between drivers and outputs were mapped individually for each article using the coding scheme refined in stage two. We took the opportunity to further refine the concept definitions represented by the coding scheme where examination of the texts provided more information. Two researchers divided the texts, consulting and cross-checking a sample of each other's maps to ensure consistency. For research studies that

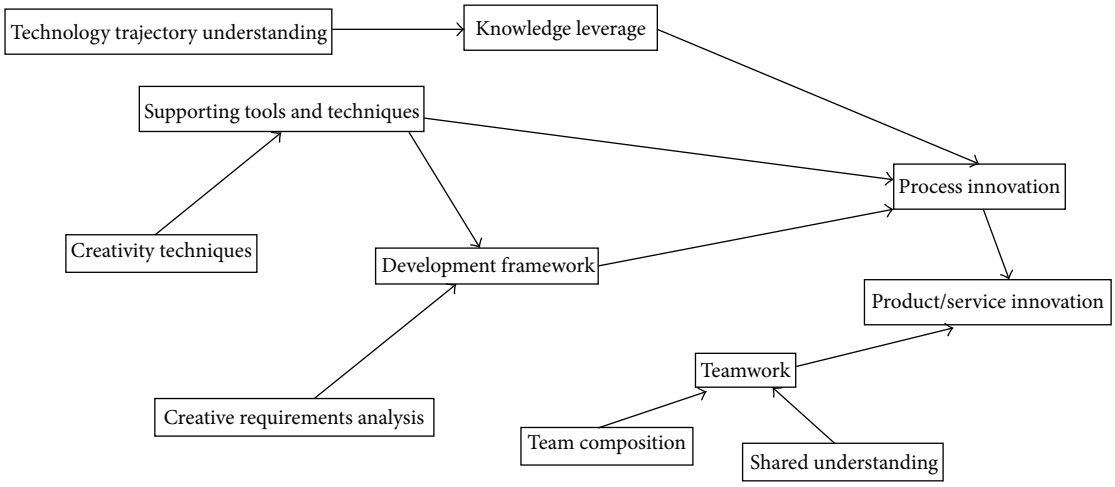

Figure 4: Example causal map based on Cooper [25].

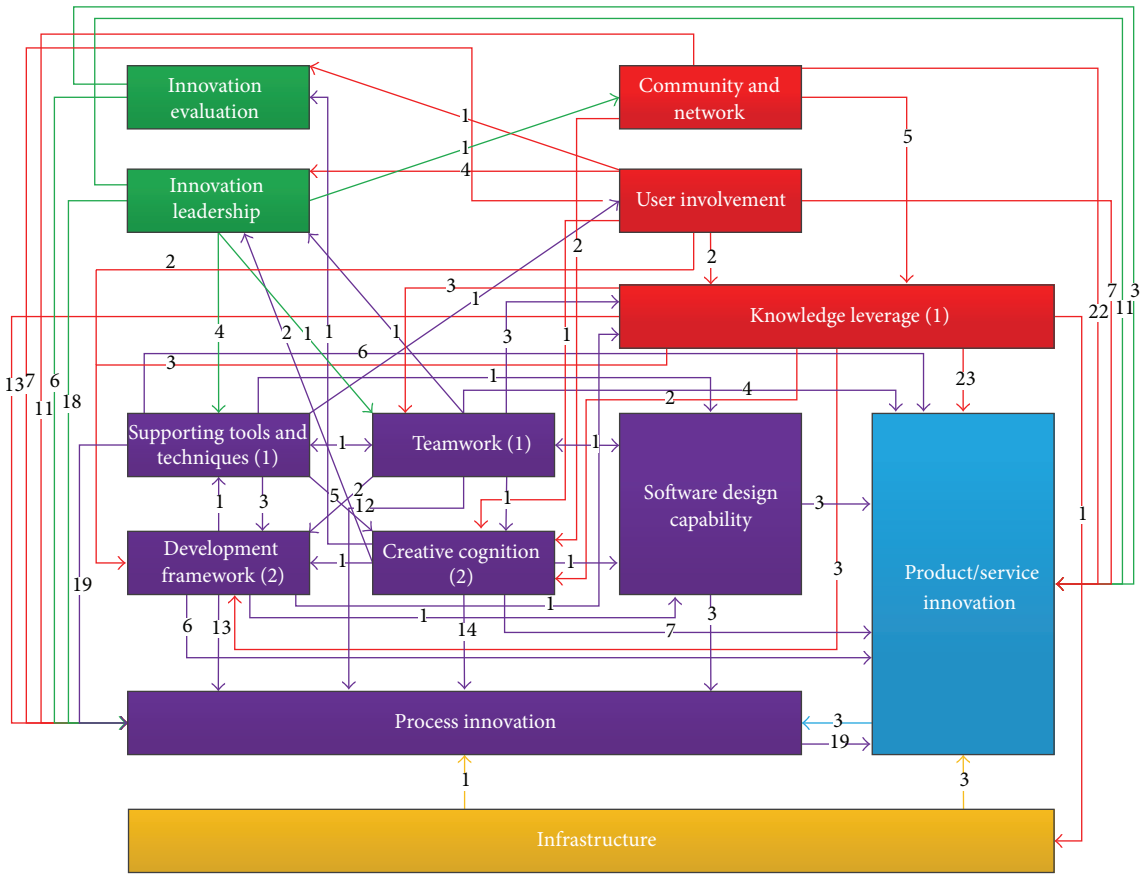

Figure 5: Consolidated causal map, showing number of links mapped.

defined clear dependent and independent variables, causal relationships could be directly extracted. In other cases these were inferred based on the reasoning provided in the articles. Our mapping effort ultimately yielded 93 distinct causal maps that depict approximately four hundred causal linkages (Figure 4 presents an example map for one article). Individual casual maps were synthesized into an association matrix (Table 7) that highlights the key causal relationships suggested by the literature and notes the strength of the literature support for each of these relationships. The consolidated association matrix was then used to construct a high-level composite causal map (Figure 5). The numbers on the causal paths in this map indicate the number of research articles that identified the relationships being depicted and, as such,

provide an indication of the relative importance of the various drivers that we have identified. The composite causal map is thus based on a much deeper level of analysis than the model presented in Figure 3. Since it can be confusing to absorb the large number of causal relationships that are present, they are structured and simplified, though not distorted, for the final iteration of the model presented in Figure 2. This model can be understood as the highest level of abstraction of a synthesis of higher order themes [24]. Specific details of the associations between innovation drivers and outputs as well as the literature sources from which they were derived are presented in Tables 8–10.

The following sections elaborate on the results of our review, analysis, and synthesis efforts beginning with a discussion of the key forms of innovation that have been examined in the software development context.

3. Software Innovation Outputs

We identified two broad categories of innovation outputs discussed in the literature.

3.1. Software Product/Service Innovation. The most common form of software innovation results in novel and useful code and, by inference, the creation of new software functionality. Innovation of this form has led to the creation of an extensive array of software systems including enterprise tools, end-user applications, operating systems, communication protocols, mobile applications, and embedded software [2]. In principle, this form of innovation encompasses all non-physical computing artefacts though this distinction is not entirely unambiguous in the case of embedded software. Further to this, the hardware and software for devices such as mobile phones are often developed in parallel, thereby making the two components quite interdependent. Similarly, it is often difficult to observe the traditional distinction between product and service in the case of software innovation since the term service is widely used to denote different forms of software. For example, Carlo et al. [6] use service as a generic term for all software products apart from base innovations while many other authors use the term service to refer to particular types of software such as web services [26] or mobile services [27].

Beyond use of the term service to describe software product offerings, the term can also be used to describe a wide range of software-related activities such as installation, end-user support, platform management, and consulting. In addition, the term service is frequently used to describe bundled offerings such as Internet hosting or application service provision that represent combinations of software and other services. A modern variant of such offerings is software as a service (SaaS). SaaS providers such as Amazon Web Services offer software tools on a cloud infrastructure, priced based on usage levels, and complemented with related nonsoftware services such as back-up and disaster recovery. The notion of service can also be applied to software that forms part of a wider offering such as a banking service where it is the combined offering that might represent the innovation [28, 29]. Finally, the term service is sometimes used to distinguish between application-oriented software innovations and underlying platform innovations. This can be observed with Apple's iOS developer interface which can be seen as a service innovation that opens the way for their apps market [30, 31]. As a result, we view innovations in software product and service offerings as one key form of software innovation while nonetheless acknowledging that the specific nature of these offerings is subject to considerable variation.

3.2. Software Process Innovation. Software processes encompass the tasks, norms, and formal and informal procedures that support software development efforts. These processes are expressed in the methods, tools, and techniques that organize the work of a developer and describe how software is developed [2]. Carlo et al. [6] view software process innovation as involving innovations in ways to envision, design, and implement software. All significant improvements in design techniques, team organization, and managerial processes can thus be classified as process innovations. However, an interesting case refers to process innovations that are focused on facilitating the development of innovative products. An example is Essence [8], a development framework that has the development of innovative software products as its explicit goal. Given the potential of such frameworks and the complexity of software development efforts, software process innovation can be of considerable importance to the development of innovative software products and services. In fact, our investigation indicates that software process innovation is one of the top three drivers of software product/service indication. We therefore identify software process innovation as the second key form of innovation in the software development context and as an important driver of product/service innovation.

4. Drivers for Software Innovation

Our analysis also led to the identification of eleven innovation drivers that we group into four categories: managerial drivers, knowledge drivers, team process drivers, and infrastructure factors.

4.1. Managerial Drivers. Although our basic unit of analysis is a software producing team, these teams operate in organizational environments that are strongly influenced by managers. Through monitoring, control, and direction setting efforts, managers influence a wide range of important parameters that can have significant implications for software innovation. These parameters include resource allocations, work environments, strategic goals, and specification of the initiatives that are included in a project portfolio.

4.1.1. Innovation Leadership. Innovation leadership has been identified as having a powerful influence on software innovation. We found such leadership to be especially significant in relation to process innovation with innovation leadership being the second most prominent driver of software process

innovation. Nonetheless, innovation leadership was also an important driver of software product and service innovation. Leaders are responsible for fostering a work environment that stimulates creativity, minimizes impediments to creativity, provides needed resources in a timely manner, minimizes bureaucracy and rigidity, and establishes appropriate evaluation and reward systems to foster experimentation [32–35]. Florida and Goodnight [36] characterize such efforts as minimizing hassles and stimulating minds. Leaders can thus serve as important champions of innovation [37]. Leaders are also responsible for path creation through transformational leadership [38–42]. Transformational leaders guide their organizations through changes that are beyond the consideration of individual developers and development teams such as fundamental changes in base technologies and product markets. More routine managerial efforts are also of notable importance to software innovation. These efforts can include goal conflict resolution [43] and efforts to synchronize the introduction of new support technologies and administrative routines in process innovations [44]. Finally, leaders are responsible for managing an organization's portfolio of products [39] to secure and maintain a product portfolio that combines short term profitability with the experimentation and learning needed for innovation and long term success.

4.1.2. Innovation Evaluation. The ability to evaluate creativity and innovation in software engineering work is an important precursor to improvement. Evaluation takes the form of assessing the work environment, assessing the value of competing ideas during ideation, assessing new software product concepts [45, 46], determining the value of process improvements and creativity support systems [47], and determining the value of the software services currently in use [48]. Although evaluation may be formal or informal, informal evaluation of team innovation can be particularly important for process organizers seeking to understand how to best manage a project. Informal evaluations of this type can take the form of simple observations of team performance [2]. In contrast, formal evaluations are fundamentally dependent on management efforts to develop and apply specific metrics and targets [46, 49].

4.2. Knowledge Drivers. A second group of drivers are knowledge-oriented factors that relate to the acquisition and leveraging of knowledge from internal and external stakeholders and the relationships that develop as part of these efforts. Software development organizations can be understood as knowledge-exploitation systems wherein the many forms of technical knowledge necessary to write software are combined with other essential knowledge that is extracted from partners, customers, and users. Innovation thus depends on an ability to leverage existing knowledge and a capacity to acquire new knowledge.

4.2.1. Knowledge Leverage. Prior work suggests that knowledge plays a central role in many aspects of software innovation including creative requirements elicitation [25] and understanding innovation opportunities [50]. We identified knowledge leverage as being the most salient driver of software product and service innovation as well as being a notable driver of software process innovation. Effectively leveraging available knowledge depends upon the development of unique competencies. This had led some researchers to examine how absorptive capacity can support attempts to successfully draw upon knowledge available in the external environment. Absorptive capacity is defined by this work as the ability of an organization to identify, assimilate, and exploit external knowledge. The prevalence of this notion in the literature means that it is one of the more thoroughly developed theoretical concepts in the software innovation literature. For example, Carlo et al. [6] isolate knowledge depth, diversity and linkages, and routines of sensing and experimenting in their efforts to highlight the importance of absorptive capacity to leveraging knowledge in software innovation.

Understanding the evolution and trajectory of customer demand (market understanding) and the trajectory of technology development has also been identified as being of some importance to new product development [30, 34, 39, 51, 52]. Knowledge of competitors and their innovations is also considered to be important in fostering the development of new generations of software systems [53] as is the ability to leverage user-domain understanding generated from customers [28, 54–56]. Taken together, efforts to leverage internal and external knowledge represent the well-known complement of market pull and technology push that has been posited to drive innovation in a wide range of contexts [52].

4.2.2. Community and Network. Community and network was found to be the second most important driver of software product and service innovation. Since knowledge creation and use are understood to be a social process [57], innovation researchers have tended to emphasize the importance of communities and networks to successful innovation [58]. National and regional development projects, science parks, silicon valleys, and industry and university collaborations are often created to encourage innovations and ensure that these innovations spawn further innovations. However, the software industry has witnessed the emergence of a specific form of community-based innovation in the form of the open source movement [59, 60]. The importance of this movement is such that most major software firms now use open source as an integral element of their innovation strategy [50] with platform developers routinely using open strategies for populating their platforms with applications [31].

According to [61], open source development is an example of a private-collective model of innovation that represents a form of innovation not previously seen in either private industry or the collective knowledge creation efforts of universities. Lee and Cole [54] point out that open source brings specialist expertise together in a community that can serve as the primary vehicle for knowledge generation. Some types of open source development might even be characterized as forms of user-driven software development, involving open access to intellectual property such as source

code in a model known as open innovation [50, 54, 60, 62–67]. Supporting these models are developments in social computing technologies that have significantly expanded the nature and scope of the communities and networks that can be drawn upon. The importance of new community forms is evident in research examining the role of social networks in innovation [68], the value of organizational web communities [69], crowd sourced software innovation [66], and collective invention [70].

4.2.3. User Involvement. Though some innovations are driven by system developers and designers, research has increasingly stressed the role of users in software innovation [65]. In particular, customers can play an important role in the commercialization of software inventions [71], assisting with customization, requirements elicitation, and early investment. However, the notion of a user is such that users can encompass a wide range of other stakeholders including end users, developers that adapt an existing system, and organizations that purchase software products and services. For example, many open source developers are users of the software they help to develop such that innovations in this software are often inspired by the application of their knowledge to unmet needs. As a consequence, integrating user knowledge and expertise into the development process is thought to be of central importance to the innovation that takes place via agile methods [55], particularly given that users often produce more creative ideas than developers [27]. Moreover, users with special skills that enable them to conceptualise [5] and prototype software systems are important to user-driven innovation [28, 29, 39, 72, 73]. These lead users can be especially valuable when sticky knowledge makes it difficult for software engineers to understand the use context.

4.3. Team Process Drivers. The next class of drivers relate directly to the software development team and its processes. Software is usually produced in teams that synthesize the creative ideas of team members and external knowledge into code that yields the product/service offerings of software development organizations. As such, team and process characteristics are intimately linked to the nature of the products/services that are ultimately delivered.

4.3.1. Creative Cognition. Creative cognition research aims to understand the creative state of mind and the creative acts of individuals [74, 75], categorize different innovation styles [76], develop creative thinking [77], and foster creative talent in engineers [78]. As a result, it tends to be founded upon theories drawn from the field of psychology. The basic premise is that the creativity of participants in a system development initiative can contribute significantly to the level of innovation generated by this initiative. Avital and Te'eni [5] describe this generative capacity as "the ability to rejuvenate, to produce new configurations and possibilities, to reframe the way we see and understand the world and to challenge the normative status quo in a particular task-driven context." Generative capacity has been reported to contribute to idea generation, evaluation, improvement, and realisation [52, 79, 80]. This process of ideation is understood to be an important collaborative process [81] that is closely linked to preexisting knowledge or expertise [82]. A general review of creativity in information systems research that explores some of the issues discussed here is offered by Müller and Ulrich [83].

4.3.2. Software Design Capability. Design capability encapsulates developer capacity to integrate customer understanding, market understanding, and technological advances into novel and useful product features [84]. The relationship between design and creativity is perhaps most fully explored in the field of human-computer interaction (HCI) [85] though the IS literature also contains numerous references to the role of design in innovation. Notably, Martin [56] and Sas and Zhang [86] suggest that design is of central importance to the innovation effort. Similarly, Leonardi [87] discusses the impact of designer background on technology outcomes, suggesting the importance of "an innovator's vision of what functionality the built technology (the technological artefact) should have". Avital and Te'eni [5] go beyond the functionality represented in a simple feature set to develop the idea of generative fit as a design characteristic of certain innovative systems. In aggregate, these observations serve to highlight the broad based importance of a development team's design capability to innovation outcomes.

4.3.3. Teamwork. Effective teamwork is considered an essential feature of innovative projects [7, 88], contributing to team efficiency and the personal satisfaction of team members. The blend of experiences and competencies found in the composition of a team are also of fundamental importance to innovation [25]. As a result, numerous perspectives have been adopted to explore the implications of teamwork for software innovation. Cooper [25] focuses on how appropriate group norms, task clarity, and member diversity can stimulate cross-fertilisation of ideas while van den Ende and Wijnberg [40] consider the implications of internal and external autonomy. Other elements of teamwork that have been explored include dysfunction avoidance [89], suitable role allocation [8], appropriate communicative interactions [89, 90], the accommodation of divergent thinking, and team learning [91]. Finally, Tiwana and McLean [92] highlight the importance of expertise integration [3, 41, 87] which they see as the capacity to exploit knowledge transfer between team members who possess different skills. An important facilitator of such integration as well as effective team work in general is the development of shared understanding and relational capital among team members [69, 93–96].

4.3.4. Innovation Tools and Techniques. Software innovation is often assumed to benefit from access to a repertoire of suitable creativity techniques and support tools as well as situational knowledge of when to apply them [97]. Our work suggests that these tools and techniques are especially important drivers of software process innovation. In this vein, Couger et al. [98] report on the use of analytical techniques

(progressive abstraction, interrogatories, and force field analysis) as well as intuitive techniques (associations/images, wishful thinking, and analogy/metaphor) to support creativity in the systems development effort [83]. Related literature explores creativity support systems [97, 99–101] which can be defined as software systems designed to underpin creative work. Given the relevance of such systems to any creative endeavour, they are of potential importance to innovation in the software development context. Other tools that have been experimented with include decision support systems [9], collaborative work systems [102], and programming toolkits [103]. Finally, specialized forms of toolkits have also been developed to assist end users in the innovation process [40, 50, 73].

4.3.5. Development Framework. Development frameworks provide the foundation for an organization's approach to software development by offering support for processes, underlying assumptions, and work practice norms. One stream of research associates agile methods with creativity in development [8, 55, 104, 105]. Another related stream of research focuses on experimentation in the design process [6, 106]. In software development contexts this usually involves the use of prototyping, particularly where low-cost, low-technology strategies are favoured [56, 107]. Another important aspect of the development framework is the installed technological base employed by the development team. Programming languages, application programming interfaces, standards, and development environments can all have significant implications for software innovation.

An often-overlooked element of the development framework is the extent to which consideration for innovation is integral to the development process. Although there are many innovation-oriented processes in other fields, innovation has not been a core focus of software development process designers. One exception is Aaen's [8] Essence framework which offers four views (product, process, paradigm, and project) and three roles (challenger, responder, and child) as guiding principles for innovative software development. Other process strategies include creative requirements analysis such as the RESCUE method [108–110]. The RESCUE method integrates requirements elicitation techniques within a creativity framework to facilitate workshops with users at an early stage in the software development effort. Such methods highlight an important question concerning the extent to which software development processes can be designed for innovation.

4.4. Infrastructure Moderators. In addition to the managerial, knowledge, and team process drivers of software innovation evident in the literature, we also identified elements of infrastructure as having important moderating influences on software innovation. Infrastructure determines the fundamental, unavoidable technological and social conditions in which innovative efforts are situated. These can include broadband availability and speed, microprocessor power, customer experience, and user computer literacy. Since infrastructure develops slowly in relation to the production of software, it often has significant implications for the innovations that can be successful within a given window of time. In particular, two key aspects or elements of infrastructure have been reported to both enable and constrain innovation.

4.4.1. Installed Base. The first salient element of infrastructure is the technological ecosystem in which software development is undertaken. At a societal level the existing technological infrastructure represents the technological ecosystem [51] into which an innovation must fit [111]. The influence of this ecosystem is such that it becomes essential that innovators understand technology trajectories [112, 113] and convergence [114]. Since the installed base evolves independent of a developer group, the timing of innovations can become crucial [2]. If an innovation is too early, supporting infrastructure such as communication bandwidth and processing power may not be widely available. If it is too late then the innovation has likely become evident to many competitors.

4.4.2. Path Dependency. The second salient element of infrastructure relates to the technological capabilities that are available to developers [6]. Infrastructures such as computing architectures, operating systems, telecommunication platforms, middleware, and programming languages all enable and constrain software innovation in specific ways. Innovations typically grow out of existing software systems which are, themselves, part of the knowledge infrastructure [34]. As a result, technological capabilities engender a form of path dependency [51] that can have significant implications for the innovations that are achievable and may require major leadership interventions to change. Complicating efforts to identify and understand key path dependencies is the fact that infrastructures and architectures can evolve in parallel. In this vein, Boland et al. [51] describe waves of innovations while [30] highlights interrelations between business, platform, and application architectures.

5. An Integrative Concept Map for Software Innovation

Building on the preceding discussion of software innovation outcomes and drivers, we consolidate key innovation drivers into an integrative model of software innovation (Figure 2).

Figure 2 provides a high-level map of software innovation that highlights the principal associations among salient innovation drivers as well as between these drivers and the two key forms of software innovation. The model makes it clear that software innovation can be influenced by a wide range of factors that arise at multiple levels of analysis. Innovation can, for example, be driven by customer or user demands, by the actions of organizational managers, and by actions and characteristics of members of the software development team. However, given that it is the software development team that directly creates software products, the impact of many drivers can be characterized as being largely mediated through elements of the development team. Please note that there is weaker support in the causal mapping analysis for

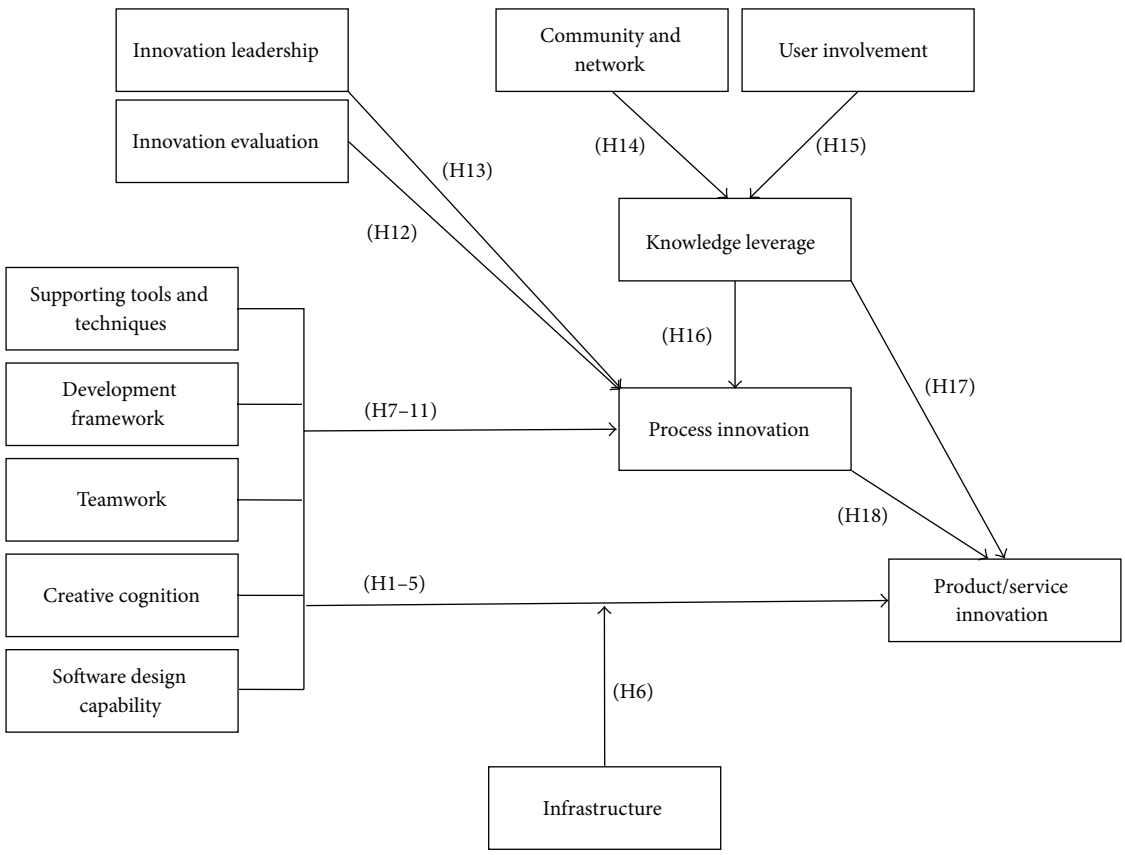

Figure 6: Sketch for high-level factor model of software innovation with hypotheses. (H1) Innovation tools and techniques are positively associated with product/service innovation. (H2) Development framework is positively associated with product/service innovation. (H3) Teamwork is positively associated with product/service innovation. (H4) Creative cognition is positively associated with product/service innovation. (H5) Software design capability is positively associated with product/service innovation. (H6) Infrastructure moderates (H1–5). (H7) Innovation tools and techniques are positively associated with process innovation. (H8) Development framework is positively associated with process innovation. (H9) Teamwork is positively associated with process innovation. (H10) Creative cognition is positively associated with process innovation. (H11) Software design capability is positively associated with process innovation. (H12) Innovation evaluation is positively associated with process innovation. (H13) Innovation leadership positively associated with process innovation. (H14) Community and network is positively associated with knowledge leverage. (H15) User involvement is positively associated with knowledge leverage. (H16) Knowledge leverage is positively associated with process innovation. (H17) Knowledge leverage is positively associated with product/service innovation. (H18) Process innovation is positively associated with product/service innovation.

the individual infrastructural moderating effects than for the other relationships. They are included for completeness and consistency.

At a broad level of abstraction, the model presented in Figure 2 postulates that team process drivers lie at the heart of software innovation. These processes are influenced by managerial activities of leadership and evaluation that are undertaken by people inside the team (project leaders or opinion formers) as well as by outsiders. Team processes are also influenced by knowledge leverage itself driven by relationships with customers and outside communities. Team process drivers, knowledge leverage, and process innovation ultimately influence product/service innovation. Process innovation and product/service innovation are both enabled and constrained by infrastructure factors present in the wider social context.

In order to facilitate future research, Table 2 provides working definitions for the key outputs of software innovation (labelled (O1) and (O2) in the table), the salient drivers of such innovation (labelled (Dx.y)), and the key moderating factors (labelled (Mx.y)) evident in the literature. In addition to defining these concepts, Table 2 also provides definitions for the various elements that are linked to them and a synopsis of their salient casual links or associations. A detailed presentation of the supporting literature for these definitions and associations can be found in Tables 8–10.

Researchers and practitioners can use the integrated, consolidated insights provided in Table 2 and Figure 2 to direct research efforts and to steer practical initiatives in directions that offer the most promising opportunities to improve the innovation capacity of software development organizations. We elaborate on our findings and the opportunities that they present in the discussion in the next section.

Table 2: Concept definitions and associations.

Concept	Definition	Principal associations
(O1) Product/service innovation	Novel and useful software products and services representing a significant advance or change in direction for a company	Understood as an output influenced by various drivers
(O2) Process innovation	Step-changes or significant modifications in the processes used to develop software products and services	Understood as an output influenced by various drivers and often contributes to product/service innovation
(D1) Innovation leadership	Managing development teams to create innovation	Drives both process innovation and, through heavy influence on team process, product/service innovation
(D1.1) Work environment	Promoting a creative work environment and minimizing creativity barriers	Promotes creative cognition and teamwork
(D1.2) Path creation	Creating an overall sense of direction in response to market and technology developments	Drives product/service innovation through team process
(D1.3) Portfolio management	Steering multiple projects in relation to innovation challenges	Can be a form of innovation evaluation
(D1.4) Conflict resolution	Resolving conflicts between individuals and groups in the pursuit of innovation	Enables creative cognition and teamwork
(D2) Innovation evaluation	The ability to reflectively evaluate ideas, techniques, and processes for their contribution to innovation	Drives product/service innovation moderated by team process and also relates to evaluation of ideas
(D3) Knowledge leverage	The use of internal or external knowledge to drive software innovation	Drives product/service innovation and, to a lesser extent, process innovation
(D3.1) Absorptive capacity	The ability of a development team to find, adapt, and exploit external knowledge in software innovation	Basis for leverage of other knowledge drivers
(D3.2) Market understanding	The use of information about software markets to promote product innovation	Drives product/service innovation
(D3.3) Technology trajectory understanding	The use of understandings of the probable direction of software and hardware infrastructures, platforms, and technologies to guide innovation	Drives product/service innovation
(D3.4) User domain understanding	Using understandings of customers' business domain or specialized internal knowledge to drive innovation	Drives product/service innovation, related to user involvement
(D3.5) Competitor understanding	Monitoring competitors' processes, products, and services to inform innovation	Drives product/service innovation
(D4) Community and network	Exploiting external connections, collaborations, and partnerships to promote innovation	Strong connection with the various forms of knowledge leverage
(D4.1) Open innovation	Using open business models that partially or wholly share intellectual property (e.g., code) to promote innovation	Strong connection with the various forms of knowledge leverage
(D4.2) Open source	Exploiting open source code or cooperation to drive innovation	Has implications for both product/services and process
(D4.3) Crowd sourcing	Inviting the widespread participation of potential users and customers to enhance innovation	Often includes a process innovation in the incorporation of knowledge or ideas into the development framework
(D5) User involvement	Involving users to stimulate innovation	Drives both product/service innovation and to a lesser extent process innovation, through knowledge leverage, particularly user domain knowledge
(D5.1) Customization	Involving users in customization of standard products and services	Involves development framework adjustments

Table 2: Continued.

Concept	Definition	Principal associations
(D5.2) User-driven/lead user	Facilitating expert users with specialist competences in directing software innovation	Related to user toolkits, often the vehicle for user innovation
(D6) Creative cognition	The exploitation of individual cognitive creativity for innovation	Involves process innovations and drives product/service innovation
(D6.1) Generative capacity	The ability to generate creative ideas and solutions promoting innovation	Involves process innovations and drives product/service innovation
(D6.2) Ideation expertise	The ability to refine and exploit creative ideas to promote innovation	Involves process innovations and drives product/service innovation, related to evaluation
(D7) Software design capability	The ability to design innovative software products and services	Often a form of knowledge leverage moderated by teamwork
(D7.1) Concept	The ability to develop overall concepts for new products and services	Often the product of absorptive capacity
(D7.2) Feature set	The ability to create distinct sets of novel and useful software functionality	As above
(D8) Teamwork	Organizing teamwork to promote innovation	Drives process innovations and contributes to product/service innovation
(D8.1) Team composition	Selection of team members to promote innovation	Affects teamwork and generative capacity
(D8.2) Expertise integration	Facilitating dialogue between experts with different technical and nontechnical specializations	Relationship with knowledge leverage
(D8.3) Shared understanding	Building and maintaining a team's common purpose in the face of many challenges and direction changes	Also an innovation leadership task
(D9) Innovation tools & techniques	Using tools and techniques designed to promote creativity in the development process	Strongly associated with process innovation
(D9.1) Creativity techniques	The use of conceptual tools (such as mind-mapping) to support innovation	Strong relation to development framework, which promotes creative cognition
(D9.2) Creativity support tools	The use of computerised tools designed to support creativity to support innovation	Strong relation to development framework, which promotes creative cognition
(D9.3) User toolkits	The deployment of tools (often computerised) to facilitate user innovation, often in respect to a technology platform	Relates to user-driven innovation
(D10) Development framework	The concepts, methods, and techniques used to underpin software team's development effort in respect to innovation	Associated with process improvement
(D10.1) Agility	Use of agile methods, or adaptations of agile methods as an innovation driver	Associated with process improvement
(D10.2) Creative requirements analysis	Stimulating requirements gathering by use of techniques designed to increase users' and customers' creativity	Associated with process improvement
(D10.3) Experimentation/prototyping	Stimulating creativity by iterative use of experimentation and/or prototyping in the development process	As above, often concerned with exploitation of external knowledge
(M1) Infrastructure	Recognizing social and technical infrastructural preconditions for innovation	Moderates both product/service innovation and process innovation
(M1.1) Installed base	Exploiting the technical development environment of a software firm to generate innovation	Moderates (enables and constrains) the development effort and can be upgraded to drive process innovation

Table 2: Continued.

Concept	Definition	Principal associations
(M1.2) Path dependency	Recognizing innovation directions partially determined by developments in infrastructures	Relates to installed base

Table 3: Target journals.

Journal
European Journal of Information Systems
Harvard Business Review
IEEE Transactions on Engineering Management
Information and Software Technology
Information Management
Information Systems Journal
Information Systems Research
International Journal of Technology Management
Journal of Information Technology
Journal of Management Information Systems
Journal of Product Innovation Management
Journal of Strategic Information Systems
Journal of Systems and Software
Journal of the Association for Information Systems
Management Science
MIS Quarterly
MIT Sloan Management Review
Organization Science
Organization Studies
R&D Management
Research Policy

6. Discussion

In this paper we set out to provide definitional focus and conceptual organization to an emerging but hitherto fragmented research area. Given the importance of innovation in the organizational, managerial, and economic literatures and the many advances in the computing science literature, it is not surprising that there has been considerable work undertaken in relation to innovation and software organizations. What is somewhat more surprising is that this research area has been slow to emerge as a prominent endeavour for information systems researchers. As a discipline, we have been slow to escape the historical legacy of software development research focused on the automating of work processes and efficiency. However, these are arguably no longer the principal drivers of the software industry. At any rate, a successful software company must incorporate innovation into its armoury of capabilities. In addressing the question of "what drives software innovation?" we synthesized a fragmented body of work displaying many internal inconsistencies. As a basis for this effort, we identified a core literature for software innovation and analysed it using content analysis and causal mapping to highlight salient trends and patterns. We defined basic concepts for the research area, established important associations among these concepts, and formulated a conceptual map of software innovation based on these linkages.

We can envisage several uses of our conceptual map by researchers. Firstly the map can be used by researchers to locate their work in relation to this emerging field and it can be used to search for related work among the many references categorized. It can also be extended or amended to give a more up-to-date picture of research concerns as the field evolves. Further to this, the conceptual map that we provide can serve as a basis for the development of various types of descriptive and normative theory through the derivation of further concepts and the articulation and empirical testing of the relationships between them. In fact, significant opportunities exist in relation to research aimed at empirically testing our research hypotheses and research that seeks to explore our casual map in more depth.

Although our concept map is not formulated as a variance or path model that explicitly depicts research hypotheses, it does synthesize many researchers' working hypotheses (couched in many forms) about what drives software innovation. As such, it can be used as a basis for developing specific hypotheses related to software innovation (see Figure 6 for an initial sketch). These hypotheses can then be empirically tested either alone or in combinations. In contrast, qualitative researchers might wish to undertake empirical work aimed at refining our causal map or using this map to elucidate causal patterns that apply to particular circumstances. This latter line of inquiry would serve to further unpack our current understanding given the potential for the elements of our causal map to exhibit varying degrees of salience depending on circumstances. It might, for example, be the case that some drivers are more important for software development organizations than for other types of organizations and this can be empirically tested. Finally, there are some notable opportunities to empirically explore the role of infrastructure as a moderator of innovation and to undertake efforts aimed at identifying other salient moderators of the context that surrounds software innovation.

As our observations suggest, the development of an integrative model often provokes more questions than it readily answers. Interested researchers must be prepared to return to the original contributions that we have summarized where many discrepancies in concept terminology and numerous conflicting assumptions are evident. Of notable concern is the lack of consistency displayed in articulating the nature and scope of innovation being discussed. The product/service distinction is not clearly maintained in the literature and other constructs such as installed base, infrastructure, and platform have been characterized as both sources and consequences of

Table 4: Initial codes derived from Rose [2].

Codes	Subcodes
Environmental factors	Markets, infrastructure, technology trajectories, convergence, and innovation timing
Network and community factors	Open innovation models and open source movement
Product design factors	Customer/user input, market orientation, new technology incorporation, and novel and useful product features
Process factors	Development style, agility, requirements analysis, process frameworks/methods, prototyping, and user-driven innovation
Psychological factors	Creative state, innovation styles, developing creative thinking, and fostering creative talent in engineers
Team performance factors	Dysfunction avoidance, role allocation, communicative interactions, accommodation of divergent thinking, team learning, absorptive capacity, and expertise integration
Managerial factors	Work environment, creativity barriers, and innovation leadership
Tools and techniques factors	Creativity techniques and support tools
Evaluation factors	Evaluation

innovation depending on the context and level of granularity of the study. Knowledge leverage factors such as absorptive capacity are often defined at the level of the firm though it would seem that for the purposes of software development they might be better conceptualized at the team level. In this vein, it is not clear whether the preponderance of knowledge-oriented studies is as a result of the behavioural bias of the information systems discipline or if it simply reflects a basic characteristic of software innovation. In any case, the more recent emergence of design science as a major contributor to the discipline should focus researcher attention on more process-oriented factors such as software design capability and design frameworks. Understanding in this area could certainly be improved. For example, there would be value in exploring creativity techniques that are more software focused rather than being exclusively directed at the ideation stage, in efforts to better understand the relationship between agility and innovation and in the exploration of the value of user-driven design as a method or strategy for software development. The management of software innovation is also surprisingly neglected in the information systems literature and could be better developed. Moreover, the role of communities and partnerships in the ecosystems of software innovator organizations lacks comprehensive investigation.

6.1. Implications for Practice. As an integrative work, our efforts have numerous implications for practice. In broad terms we have consolidated current understanding surrounding software innovation into a conceptual map that can be used by practicing managers to evaluate their development environments. These evaluations could be used to understand the extent to which organizations are providing environments that foster innovation, to identify areas of focus and neglect within organizations or specific development groups, and to recommend particular innovation approaches from the literature that we reference. Further to this latter point, we highlight the most important drivers of innovation so that organizations seeking to improve their innovative capacity know where to focus their resources. For example, the particular salience of knowledge leverage suggests that this is an important starting point in efforts to improve innovation capacity. In addition to steering managers in particular directions, our results enumerate specific actions that should be pursued in order to foster key drivers of software innovation. Our work highlights, for example, that developing a sound understanding of the user domain, the product market, and the trajectory of technological development are all important to organizations that wish to foster innovation through knowledge leverage.

Analytical inquiry could be undertaken to compare an organization against innovative organizations using the parameters suggested by our causal map or to benchmark innovative organizations against larger groups of organizations. A useful working hypothesis may be that innovative organizations use specific configurations of innovation drivers rather than attempting to create environments that encompass the entire landscape of the map.

From a process perspective, we have reaffirmed the importance of sound software development processes to software product innovation. Our work suggests that these processes should not be considered incidental as they can, in fact, become a significant source of competitive advantage owing to their path dependency, their relatively tacit nature, and the fact that they are far less transparent to outside observers than software product feature sets. Organizations seeking to take advantage of this opportunity to achieve competitive advantage can draw on our work to identify specific actions that should be focal points of their efforts to develop software development processes that foster the development of innovative software products. In particular, we highlight the importance of nurturing an environment that contributes to teamwork and the need to provide appropriate tools as well as a development framework that encourages innovation. Further to this, we identify specific action items to foster these

Table 5: Concept and innovation code cooccurrence in article abstracts.

Concepts	Product/service innovation	Process innovation
Environmental	16	11
Markets	13	5
Infrastructure		
Technology trajectories	3	2
Convergence		
Network and community	11	7
Open innovation models	3	2
Open source movement	4	2
Product design	6	2
Customer/user input	16	5
Market orientation	4	4
New technology incorporation	3	4
Novel and useful product features	2	1
Process	6	6
Development style	1	1
Agility	2	8
Requirements analysis	1	8
Process frameworks/methods	1	6
Prototyping		1
Innovation timing	11	8
User-driven innovation	2	1
Psychological	8	4
Creative state	1	
Innovation styles		1
Developing creative thinking	4	9
Fostering creative talent in engineers	1	6
Team performance	10	8
Dysfunction avoidance		
Ideation	4	3
Role allocation		
Communicative interactions		2
Accommodation of divergent thinking		
Team learning	3	5
Absorptive capacity	4	6
Expertise integration		2
Managerial	16	21
Work environment	10	
Creativity barriers		1
Innovation leadership	1	3
Tool and technique		1
Creativity techniques	4	7
Support tools	16	13
Evaluation	8	3
Totals	272	203

drivers of software process innovation. Organizations that, for instance, find their teamwork in need of improvement can draw on our work to recognize that managing team composition, integrating suitable expertise, and fostering a shared understanding are the most important contributors to the type of teamwork that drives innovation.

6.2. Limitations. As with any work, the work that we have reported upon is not without some limitations. Among these is the potential for bias introduced as a consequence of our reliance on ISI Web of Knowledge and Google Scholar to identify research articles. While the inclusion of additional data sources in our search efforts might have identified additional work, comparisons of Web of Knowledge, Scopus, and Google have failed to report consistent bias or gaps in these sources [115]. Furthermore, the results of our study are not based on individual contributions but on patterns that can be observed across many contributions. We therefore believe that it is unlikely that excluding consideration for Scopus or other such sources had a significant impact on the insights that we offer.

A second limitation relates to potential threats to the validity of our work that arise as a consequence of the process that we used to code our data. Coding efforts are always somewhat vulnerable to various forms of researcher bias. The salience of particular remarks might, for example, be overlooked during the coding process or data might be misunderstood and, as a result, coded incorrectly. However, the results that we present are based on independent coding and cross-checking by both authors and by an independent researcher. Thus, while we acknowledge that coding is not an exact science and is, therefore, vulnerable to coding errors, we are confident that our results do not exhibit undue bias, particularly in relation to our most salient observations. These observations are based on recurring evidence that was coded and cross-checked by multiple researchers. This makes it quite unlikely that significant bias is present in our results.

7. Conclusion

Software producing organizations and ecosystems are now commonplace and a majority of technology start-ups incorporate some form of software into their product offerings. As such, innovation in the software development arena is of fundamental and growing importance to the success of many organizations as well as to our economic and social wellbeing. Organizations are continuously pressed to find new and better solutions to old problems and to respond in a timely manner to new opportunities and challenges as they arise. These efforts routinely draw on the potential of software to deliver innovative solutions that provide powerful, economical capabilities that would not otherwise be possible. The information systems discipline is well placed to study such organizations and how they innovate with the goal of providing the guidance necessary to ensure the continued vitality of software innovation. In particular, organizations would benefit immensely from research efforts aimed at exploring the drivers that we discuss and how these might

Table 6: Final set of articles summarized by publication outlet and year of publication.

Journals	Year of publication										References
	1980–1983	1984–1987	1988–1991	1992–1995	1996–1999	2000–2003	2004–2007	2008–2011	2012–2015	Total	
Communications of the ACM							1			1	[97]
Communications of the AIS									1	1	[83]
Computer						1				1	[104]
Computers & Industrial Engineering							1			1	[95]
Creativity and Innovation Management						1				1	[27]
European Journal of Information Systems					1			2		3	[8, 39, 44]
Expert Systems with Applications								1		1	[85]
Harvard Business Review						1	1	2		4	[36, 56, 81, 106]
IEEE Software							1			1	[116]
IEEE Transactions On Engineering Management						1				1	[40]
Information & Management							1			1	[30]
Information and Software Technology									1	1	{[117] #1592}
Information Systems Journal								2		2	[5, 118]
International Journal of Human-Computer Studies							1			1	[99]
International Journal of Technology Management							2	3		5	[55, 64, 93, 96, 119]
Journal of Information Technology				1				1		2	[11, 120]
Journal of Management								1		1	[65]
Journal of Management Information Systems							2	2		4	[66, 79, 80, 92]
Journal of Product Innovation Management					2		1	3		6	[3, 35, 38, 69, 71, 121]
Journal of the Association for Information Systems							1	1		2	[122, 123]
Lecture Notes in Computer Science							1			1	[124]
Management Science					1	1			1	3	[72, 125] {[126] #1588}
MIS Quarterly	1			1	2	1	1	1		8	[25, 28, 43, 68, 98, 101, 127]
Organization Science						2	2	3		7	[7, 51, 53, 54, 82, 87, 128]
R&D Management							1	1		2	[50, 60]
Research Policy						3	2	4		10	[29, 34, 59, 63, 67, 73, 88, 129, 130]
Requirements Engineering							1			1	[131]
Technology Analysis & Strategic Management									1	1	{[132] #1605}
Technovation							2	2		4	[42, 52, 133, 134]
Total	1			2	6	11	23	29	4	73	
CHI EA							2			2	[105, 107]
DESIRE								2		2	[86, 135]
DIS							1			1	[110]
HICSS			2	4	1					7	[37, 45, 46, 76, 136–138]
ICSE							2			2	[108, 139]
IFIP WG 8.2								1		1	[140]
PICMET								1		1	[94]
RE							1			1	[141]
REFSQ							1			1	[142]
TOCHI						1				1	[100]
TOOLS						1				1	[143]
Total			2	4	1	2	7	4		20	

Table 7: Consolidated causal map matrix.

	Influenced												
Influencing	Product/service innovation	Process innovation	Innovation leadership	Innovation evaluation	Knowledge leverage	Community and network	User involvement	Creative cognition	Software design capability	Teamwork	Supporting tools and techniques	Development framework	Infrastructure
Product/service innovation		3											
Process innovation	19												
Innovation leadership	11	18				1				1	4		
Innovation evaluation	4	6											
Knowledge leverage	23	13	4	1				2		3		3	1
Community and network	22	11			5			2					
User involvement	7	7			2			1				2	
Creative cognition	7	14	2	1					1			1	
Software design capability	3	3								1			
Teamwork	4	12	1		3			1	1		1	2	
Supporting tools and techniques	6	19					1	5	1	1		3	
Development framework	6	13			1				1		1		
Infrastructure	3	1											

Table 8: Associations between drivers and product/service innovation.

Software innovation drivers	Software innovation output	Sources
Innovation leadership →		[28, 34, 35, 39, 40, 51, 55, 56]
(i) Work environment →		[33–35]
(ii) Path creation →		[39–42, 51, 129]
(iii) Portfolio management →		[39]
(iv) Conflict resolution		
Innovation evaluation →		[82, 101, 122]
Knowledge leverage →		[3, 25, 35, 40–42, 54, 55, 72, 129, 130, 144]
(i) Absorptive capacity →		[6, 28, 35, 39, 50, 65, 82, 121]
(ii) Market understanding →		[35, 39, 40, 42, 52, 53, 121, 130]
(iii) Technology trajectory understanding →		[30, 34, 35, 39, 42, 51, 52, 130, 144]
(iv) User domain understanding →		[42, 56, 119, 120, 129, 144]
(v) Competitor understanding →		[35, 53]
Community and network →		[3, 34, 40, 50, 51, 54, 66, 67, 72, 81, 82, 130]
(i) Open innovation →		[50, 54, 65–67]
(ii) Open source →		[65, 67, 120]
(iii) Crowd sourcing →		[66]
User involvement →		[27, 29, 56, 66, 71, 73, 119, 122]
(i) Customization →		[71]
(ii) User-driven/lead user →		[27–29, 39, 72, 73]
Creative cognition →	Product/service innovation	[5, 25, 33, 79, 98, 101]
(i) Generative capacity →		[5, 27, 34, 66, 79, 81, 82, 100, 101]
(ii) Ideation expertise →		[35, 52, 79, 100]
Software design capability →		[105, 133]
(i) Concept →		[87, 105, 133]
(ii) Feature set →		[87, 133]
Teamwork →		[25, 40, 88, 98]
(i) Team composition →		[25]
(ii) Expertise integration →		[3, 41, 87]
(iii) Shared understanding		[25]
Innovation tools & techniques		[133]
(i) Creativity techniques →		[25, 33, 46, 79, 98, 133]
(ii) Creativity support tools →		[5, 33, 97, 100, 101]
(iii) User toolkits		[50, 73]
Development framework		[105]
(i) Agility		[105]
(ii) Creative requirements analysis →		[25, 71, 105]
(iii) Experimentation/prototyping →		[6, 56, 106]
Infrastructure/installed base →		[30, 51]
(i) Path dependency →		[51]

→ is a driver for/is associated with.

Table 9: Associations between drivers and process innovation.

Software innovation drivers	Software innovation output	Sources
Innovation leadership →		[36, 37, 39, 43, 44, 51, 55, 56, 83, 95, 137]
(i) Work environment →		[36, 37, 83, 93, 95, 135, 138]
(ii) Path creation →		[39, 51, 95]
(iii) Portfolio management →		[39, 94]
(iv) Conflict resolution →		[43]
Innovation evaluation →		[60, 80, 82, 83, 95, 101, 136]
(i) Knowledge leverage →		[25, 43, 64, 68, 69, 92, 95, 96, 107, 135]
(ii) Absorptive capacity →		[6, 37, 82, 127]
(iii) Market understanding →		[36, 39, 52]
(iv) Technology trajectory understanding →		[25, 37, 39, 51, 52, 83, 135]
(v) User domain understanding →		[56, 68, 95, 131]
(vi) Competitor understanding		
Community and network →		[51, 63, 68, 69, 82, 107, 123]
(i) Open innovation →		[60, 63, 64]
(ii) Open source →		[59, 60]
(iii) Crowd sourcing →		[63]
User involvement →		[55, 56, 73, 105, 110, 116]
(i) Customization		
(ii) User-driven/lead user →	Process innovation	[73]
Creative cognition →		[80, 83, 86, 101, 125, 136, 138]
(i) Generative capacity →		[37, 82, 92, 95, 101, 125]
(ii) Ideation expertise →		[52, 80, 86]
Software design capability →		[11, 86]
(i) Concept →		[11]
(ii) Feature set →		[11]
Teamwork →		[69, 83, 88, 92, 95]
(i) Team composition →		[8, 92, 94, 95]
(ii) Expertise integration		[59, 92]
(iii) Shared understanding →		[69, 92–96]
Innovation tools & techniques →		[25]
(i) Creativity techniques →		[8, 25, 83, 110, 131, 138]
(ii) Creativity support tools →		[83, 93, 94, 99, 101, 125]
(iii) User toolkits →		[11, 73, 83]
Development framework →		[8, 11, 25, 104, 110, 116]
(i) Agility →		[8, 55, 104]
(ii) Creative requirements analysis →		[25, 93, 94, 110, 131]
(iii) Experimentation/prototyping →		[6, 56, 107]
Infrastructure/installed base		
(i) Path dependency →		[51]

Table 10: Associations between process innovation and product/service innovation.

	Literature sources
Process innovation → Product/service innovation	[8, 11, 25, 30, 55, 56, 60, 64, 69, 73, 80, 97, 99, 104, 105, 107, 110, 116]
Product/service innovation → Process innovation	[42, 55, 56, 73]

be leveraged in various ways to yield innovative, effective software solutions while practicing managers can benefit immediately through application of the insights we provide.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research work is supported by the Danish Research Council, Grant no. 12-133180, and the Judge School of Business, Cambridge University.

References

[1] M. Pikkarainen, W. Codenie, N. Boucart, and J. A. Heredia Alvaro, *The Art of Software Innovation: Eight Practice Areas to Inspire Your Business*, Springer, Berlin, Germany, 2011.

[2] J. Rose, *Software Innovation—Eight Work-Style Heuristics for Creative System Developers*, Software Innovation, Aalborg University, Aalborg, Denmark, 2010.

[3] A. Heirman and B. Clarysse, "Which tangible and intangible assets matter for innovation speed in start-Ups?" *Journal of Product Innovation Management*, vol. 24, no. 4, pp. 303–315, 2007.

[4] A. Onetti, A. Zucchella, M. V. Jones, and P. P. McDougall-Covin, "Internationalization, innovation and entrepreneurship: business models for new technology-based firms," *Journal of Management & Governance*, vol. 16, no. 3, pp. 337–368, 2012.

[5] M. Avital and D. Te'eni, "From generative fit to generative capacity: exploring an emerging dimension of information systems design and task performance," *Information Systems Journal*, vol. 19, no. 4, pp. 345–367, 2009.

[6] J. Carlo, K. Lyytinen, and G. Rose, "A knowledge-based model of radical innovation in small software firms," *MIS Quarterly*, vol. 36, no. 3, pp. 865–895, 2011.

[7] M. Hoegl and H. G. Gemuenden, "Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence," *Organization Science*, vol. 12, no. 4, pp. 435–449, 2001.

[8] I. Aaen, "Essence: facilitating software innovation," *European Journal of Information Systems*, vol. 17, no. 5, pp. 543–553, 2008.

[9] J. J. Elam and M. Mead, "Designing for creativity: considerations for DSS development," *Information & Management*, vol. 13, no. 5, pp. 215–222, 1987.

[10] J. Fagerberg, "Innovation: a guide to the literature," in *The Oxford Handbook of Innovation*, J. Fagerberg, C. Mowery, and R. Nelson, Eds., pp. 1–27, Oxford University Press, Oxford, UK, 2005.

[11] P. Quintas, "A product-process model of innovation in software development," *Journal of Information Technology*, vol. 9, no. 1, pp. 3–17, 1994.

[12] C. Christensen and M. Overdorf, "Meeting the challenge of disruptive change," *Harvard Business Review*, vol. 78, no. 2, pp. 66–76, 2000.

[13] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3," 2007, https://www.cs.auckland.ac.nz/~norsaremah/2007%20Guidelines%20for%20performing%20SLR%20in%20SE%20v2.3.pdf.

[14] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: writing a literature review," *MIS Quarterly*, vol. 26, no. 2, pp. 13–23, 2002.

[15] D. J. Bem, "Writing a review article for psychological bulletin," *Psychological Bulletin*, vol. 118, no. 2, pp. 172–177, 1995.

[16] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: an update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[17] J. D. Novak, *Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations*, Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1988.

[18] D. Silverman, *Interpreting Qualitative Data*, SAGE, London, UK, 2001.

[19] B. Berelson, *Content Analysis in Communicative Research*, Free Press, New York, NY, USA, 1952.

[20] K. H. Krippendorff, *Content Analysis: An Introduction to Its Methodology*, Sage, Thousand Oaks, Calif, USA, 2004.

[21] V. K. Narayanan and D. J. Armstrong, *Causal Mapping for Research in Information Technology*, IGI Global, 2005.

[22] M. Laukkanen, "Comparative causal mapping and CMAP3 software in qualitative studies," *Forum Qualitative Sozialforschung*, vol. 13, no. 2, article 13, 2012, http://www.qualitative-research.net/index.php/fqs/article/view/1846/3371.

[23] S. Wilkinson, "Focus group research," in *Qualitative Research: Theory, Method and Practice*, D. Silverman, Ed., SAGE Publications, London, UK, 1997.

[24] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM '11)*, vol. 7491, pp. 275–284, IEEE, Banff, Canada, September 2011.

[25] R. B. Cooper, "Information technology development creativity: a case study of attempted radical change," *MIS Quarterly*, vol. 24, no. 2, pp. 245–276, 2000.

[26] H. Xu, S. K. Sharma, and R. Hackney, "Web services innovation research: towards a dual-core model," *International Journal of Information Management*, vol. 25, no. 4, pp. 321–334, 2005.

[27] P. Kristensson, P. R. Magnusson, and J. Matthing, "Users as a hidden resource for creativity: findings from an experimental study on user involvement," *Creativity and Innovation Management*, vol. 11, no. 1, pp. 55–61, 2002.

[28] S. Nambisan, R. Agarwal, and M. Tanniru, "Organizational mechanisms for enhancing user innovation in information technology," *MIS Quarterly*, vol. 23, no. 3, pp. 365–395, 1999.

[29] P. Oliveira and E. Von Hippel, "Users as service innovators: the case of banking services," *Research Policy*, vol. 40, no. 6, pp. 806–818, 2011.

[30] A. T. M. Aerts, J. B. M. Goossenaerts, D. K. Hammer, and J. C. Wortmann, "Architectures in context: on the evolution of business, application software, and ICT platform architectures," *Information & Management*, vol. 41, no. 6, pp. 781–794, 2004.

[31] K. Boudreau, "Open platform strategies and innovation: granting access vs. devolving control," *Management Science*, vol. 56, no. 10, pp. 1849–1872, 2010.

[32] J. D. Couger, "Press: measurement of the climate for creativity in IS organizations," *Creativity and Innovation Management*, vol. 5, no. 4, pp. 273–279, 1996.

[33] K. R. Maccrimmon and C. Wagner, "Stimulating ideas through creativity software," *Management Science*, vol. 40, no. 11, pp. 1514–1532, 1994.

[34] H. Romijn and M. Albaladejo, "Determinants of innovation capability in small electronics and software firms in southeast England," *Research Policy*, vol. 31, no. 7, pp. 1053–1067, 2002.

[35] R. G. Cooper, "Perspective: the innovation dilemma: how to innovate when the market is mature," *Journal of Product Innovation Management*, vol. 28, no. 1, pp. 2–27, 2011.

[36] R. Florida and J. Goodnight, "Managing for creativity," *Harvard Business Review*, vol. 83, no. 7-8, pp. 124–193, 2005.

[37] E. R. Mclean and S. J. Smits, "The I/S leader as 'innovator'," in *Proceedings of the 26th Hawaii International Conference on System Sciences*, vol. 4, pp. 352–358, IEEE, Wailea, Hawaii, USA, January 1993.

[38] L. Gumusluoğlu and A. Ilsev, "Transformational leadership and organizational innovation: the roles of internal and external support for innovation," *Journal of Product Innovation Management*, vol. 26, no. 3, pp. 264–277, 2009.

[39] N. P. Napier, L. Mathiassen, and D. Robey, "Building contextual ambidexterity in a software company to improve firm-level coordination," *European Journal of Information Systems*, vol. 20, no. 6, pp. 674–690, 2011.

[40] J. van den Ende and N. Wijnberg, "The organization of innovation and market dynamics: managing increasing returns in software firms," *IEEE Transactions on Engineering Management*, vol. 50, no. 3, pp. 374–382, 2003.

[41] A. Weterings and S. Koster, "Inheriting knowledge and sustaining relationships: what stimulates the innovative performance of small software firms in the Netherlands?" *Research Policy*, vol. 36, no. 3, pp. 320–335, 2007.

[42] H.-L. Yang and S.-L. Hsiao, "Mechanisms of developing innovative IT-enabled services: a case study of Taiwanese healthcare service," *Technovation*, vol. 29, no. 5, pp. 327–337, 2009.

[43] K. Sherif, R. W. Zmud, and G. J. Browne, "Managing peer-to-peer conflicts in disruptive information technology innovations: the case of software reuse," *MIS Quarterly*, vol. 30, no. 2, pp. 339–356, 2006.

[44] T. Ravichandran, "Software reusability as synchronous innovation: a test of four theoretical models," *European Journal of Information Systems*, vol. 8, no. 3, pp. 183–199, 1999.

[45] B. M. Lobert and D. G. Dologite, "Measuring creativity of information system ideas: an exploratory investigation," in *Proceedings of the 27th Hawaii International Conference on System Sciences*, pp. 392–402, Wailea, Hawaii, USA, January 1994.

[46] D. L. Amoroso and J. D. Couger, "Developing information systems with creativity techniques: an exploratory study," in *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, vol. 4, pp. 720–728, Wailea, Hawaii, USA, January 1995.

[47] Z. Chen, "Toward a better understanding of idea processors," *Information and Software Technology*, vol. 40, no. 10, pp. 541–553, 1998.

[48] A. Katsirikou and E. Sefertzi, "Innovation in the every day life of libraries," *Technovation*, vol. 20, no. 12, pp. 705–709, 2000.

[49] J. J. Shah, S. M. Smith, and N. Vargas-Hernandez, "Metrics for measuring ideation effectiveness," *Design Studies*, vol. 24, no. 2, pp. 111–134, 2003.

[50] J. West and S. Gallagher, "Challenges of open innovation: the paradox of firm investment in open-source software," *R & D Management*, vol. 36, no. 3, pp. 319–331, 2006.

[51] R. J. Boland Jr., K. Lyytinen, and Y. Yoo, "Wakes of innovation in project networks: the case of digital 3-D representations in architecture, engineering, and construction," *Organization Science*, vol. 18, no. 4, pp. 631–647, 2007.

[52] A. Brem and K.-I. Voigt, "Integration of market pull and technology push in the corporate front end and innovation management—insights from the German software industry," *Technovation*, vol. 29, no. 5, pp. 351–367, 2009.

[53] S. F. Turner, W. Mitchell, and R. A. Bettis, "Responding to rivals and complements: how market concentration shapes generational product innovation strategy," *Organization Science*, vol. 21, no. 4, pp. 854–872, 2010.

[54] G. K. Lee and R. E. Cole, "From a firm-based to a community-based model of knowledge creation: the case of the Linux kernel development," *Organization Science*, vol. 14, no. 6, pp. 633–649, 2003.

[55] O. Gassmann, P. Sandmeier, and C. H. Wecht, "Extreme customer innovation in the front-end: learning from a new software paradigm," *International Journal of Technology Management*, vol. 33, no. 1, pp. 46–66, 2006.

[56] R. L. Martin, "The innovation catalysts," *Harvard Business Review*, vol. 89, no. 6, pp. 82–87, 2011.

[57] I. Nonaka, "The knowledge-creating company," *Harvard Business Review*, vol. 69, pp. 96–104, 1991.

[58] I. Tuomi, *Networks of Innovation: Change and Meaning in the Age of the Internet*, Oxford University Press, Oxford, UK, 2003.

[59] G. Von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research Policy*, vol. 32, no. 7, pp. 1217–1241, 2003.

[60] C. R. Lamastra, "Software innovativeness. A comparison between proprietary and Free/Open Source solutions offered by Italian SMEs," *R & D Management*, vol. 39, no. 2, pp. 153–169, 2009.

[61] E. Von Hippel and G. Von Krogh, "Open source software and the 'private-collective' innovation model: issues for organization science," *Organization Science*, vol. 14, no. 2, pp. 209–225, 2003.

[62] H. W. Chesbrough, *Open Innovation: The New Imperative for Creating and Profiting from Technology*, Harvard Business School Publishing, Boston, Mass, USA, 2003.

[63] J. P. J. de Jong and E. von Hippel, "Transfers of user process innovations to process equipment producers: a study of Dutch high-tech firms," *Research Policy*, vol. 38, no. 7, pp. 1181–1191, 2009.

[64] S. Spaeth, M. Stuermer, and G. von Krogh, "Enabling knowledge creation through outsiders: towards a push model of open innovation," *International Journal of Technology Management*, vol. 52, no. 3-4, pp. 411–431, 2010.

[65] M. Bogers, A. Afuah, and B. Bastian, "Users as innovators: a review, critique, and future research directions," *Journal of Management*, vol. 36, no. 4, pp. 857–875, 2010.

[66] J. M. Leimeister, M. Huber, U. Bretschneider, and H. Krcmar, "Leveraging crowdsourcing: activation-supporting components for IT-based ideas competition," *Journal of Management Information Systems*, vol. 26, no. 1, pp. 197–224, 2009.

[67] J. Henkel, "Selective revealing in open innovation processes: the case of embedded Linux," *Research Policy*, vol. 35, no. 7, pp. 953–969, 2006.

[68] P. H. Gray, P. Salvatore, and B. Iyer, "Innovation impacts of using social bookmarking systems," *MIS Quarterly*, vol. 35, no. 3, pp. 629–643, 2011.

[69] C. C. Snow, Ø. D. Fjeldstad, C. Lettl, and R. E. Miles, "Organizing continuous product development and commercialization:

the collaborative community of firms model," *Journal of Product Innovation Management*, vol. 28, no. 1, pp. 3–16, 2011.

[70] M. Osterloh and S. Rota, "Open source software development-Just another case of collective invention?" *Research Policy*, vol. 36, no. 2, pp. 157–171, 2007.

[71] G. A. Athaide, P. W. Meyers, and D. L. Wilemon, "Seller-buyer interactions during the commercialization of technological process innovations," *Journal of Product Innovation Management*, vol. 13, no. 5, pp. 406–421, 1996.

[72] P. D. Morrison, J. H. Roberts, and E. von Hippel, "Determinants of user innovation and innovation sharing in a local market," *Management Science*, vol. 46, no. 12, pp. 1513–1527, 2000.

[73] N. Franke and E. Von Hippel, "Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software," *Research Policy*, vol. 32, no. 7, pp. 1199–1215, 2003.

[74] G. Wallas, *The Art of Thought*, Jonathan Cape, London, UK, 1926.

[75] M. Csikszentmihalyi, *Creativity: Flow and the Psychology of Discovery and Invention*, Harper Perennial, New York, NY, USA, 1997.

[76] W. Miller, J. Couger, and L. F. Higgins, "Comparing innovation styles profile of IS personnel to other occupations," in *Proceedings of the 26th Hawaii International Conference on System Sciences*, vol. 4, pp. 378–386, IEEE, Wailea, Hawaii, USA, January 1993.

[77] E. De Bono, *The Use of Lateral Thinking: A Textbook of Creativity*, Penguin, 1971.

[78] J. D. Couger, "Creativity/innovation in information systems organizations," in *Proceedings of the 30th Hawaii International Conference on System Sciences*, vol. 3, pp. 349–350, IEEE, Wailea, Hawaii, USA, January 1997.

[79] E. L. Santanen, R. O. Briggs, and G.-J. De Vreede, "Causal relationships in creative problem solving: comparing facilitation interventions for ideation," *Journal of Management Information Systems*, vol. 20, no. 4, pp. 167–197, 2004.

[80] R. O. Briggs and B. A. Reinig, "Bounded ideation theory," *Journal of Management Information Systems*, vol. 27, no. 1, pp. 123–144, 2010.

[81] G. P. Pisano and R. Verganti, "Which kind of collaboration is right for you?" *Harvard Business Review*, vol. 86, no. 12, p. 76, 2008.

[82] M. E. Sosa, "Where do creative interactions come from? The role of tie content and social networks," *Organization Science*, vol. 22, no. 1, pp. 1–21, 2011.

[83] S. D. Müller and F. Ulrich, "Creativity and information systems in a hypercompetitive environment: a literature review," *Communications of the Association for Information Systems*, vol. 32, pp. 175–200, 2013.

[84] E. B. Roberts, "Managing invention and innovation," *Research Technology Management*, vol. 50, no. 1, pp. 35–54, 2007.

[85] X. Liu, Y. Li, P. Pan, and W. Li, "Research on computer-aided creative design platform based on creativity model," *Expert Systems with Applications*, vol. 38, no. 8, pp. 9973–9990, 2011.

[86] C. Sas and C. Zhang, "Investigating emotions in creative design," in *Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design (DESIRE '10)*, pp. 138–149, Aarhus, Denmark, August 2010.

[87] P. M. Leonardi, "Innovation blindness: culture, frames, and cross-boundary problem construction in the development of new technology concepts," *Organization Science*, vol. 22, no. 2, pp. 347–369, 2011.

[88] M. Hoegl and L. Proserpio, "Team member proximity and teamwork in innovative projects," *Research Policy*, vol. 33, no. 8, pp. 1153–1165, 2004.

[89] E. W. Duggan and C. S. Thachenkary, "Integrating nominal group technique and joint application development for improved systems requirements determination," *Information & Management*, vol. 41, no. 4, pp. 399–411, 2004.

[90] E. W. Duggan, "Generating systems requirements with facilitated group techniques," *Human-Computer Interaction*, vol. 18, no. 4, pp. 373–394, 2003.

[91] K. Lyytinen and G. M. Rose, "Information system development agility as organizational learning," *European Journal of Information Systems*, vol. 15, no. 2, pp. 183–199, 2006.

[92] A. Tiwana and E. R. McLean, "Expertise integration and creativity in information systems development," *Journal of Management Information Systems*, vol. 22, no. 1, pp. 13–43, 2005.

[93] A. Hesmer, K. A. Hribernik, J. M. Baalsrud Hauge, and K.-D. Thoben, "Supporting the ideation processes by a collaborative online based toolset," *International Journal of Technology Management*, vol. 55, no. 3-4, pp. 218–225, 2011.

[94] P. Hocova, J. F. E Cunha, and Z. Staníček, "Design and management of an innovative software enterprise: a case study of a spin-off from university," in *Proceedings of the Portland International Conference on Management of Engineering & Technology (PICMET '09)*, D. F. Kocaoglu, T. R. Anderson, and T. U. Daim, Eds., pp. 2788–2797, IEEE, Portland, Ore, USA, August 2009.

[95] T. Koc, "Organizational determinants of innovation capacity in software companies," *Computers & Industrial Engineering*, vol. 53, no. 3, pp. 373–385, 2007.

[96] I.-Y. Lu and C.-H. Wang, "Technology innovation and knowledge management in the high-tech industry," *International Journal of Technology Management*, vol. 39, no. 1-2, pp. 3–19, 2007.

[97] B. Shneiderman, "Creativity support tools: accelerating discovery and innovation," *Communications of the ACM*, vol. 50, no. 12, pp. 20–32, 2007.

[98] J. Couger, L. Higgins, and S. C. McIntyre, "(Un)structured creativity in information systems organizations," *MIS Quarterly*, vol. 17, no. 4, pp. 375–397, 1993.

[99] T. T. Hewett, "Informing the design of computer-based environments to support creativity," *International Journal of Human Computer Studies*, vol. 63, no. 4-5, pp. 383–409, 2005.

[100] B. E. N. Shneiderman, "Creating creativity: user interfaces for supporting innovation," *ACM Transactions on Computer-Human Interaction*, vol. 7, no. 1, pp. 114–138, 2000.

[101] B. Massetti, "An empirical examination of the value of creativity support systems on idea generation," *MIS Quarterly*, vol. 20, no. 1, pp. 83–97, 1996.

[102] J. W. Fellers and R. P. Bostrom, "Application of group support systems to promote creativity in information systems organizations," in *Proceedings of the 26th Hawaii International Conference on System Sciences*, vol. 4, pp. 332–341, IEEE, Wailea, Hawaii, USA, January 1993.

[103] S. Greenberg, "Toolkits and interface creativity," *Multimedia Tools and Applications*, vol. 32, no. 2, pp. 139–159, 2007.

[104] J. Highsmith and A. Cockburn, "Agile software development: the business of innovation," *Computer*, vol. 34, no. 9, pp. 120–127, 2001.

[105] D. K. Busse, "Fast-tracking product innovation," in *Proceedings of the 25th Extended Abstracts on Human Factors in Computing*

Systems (CHI '07), pp. 1703–1708, ACM, San Jose, Calif, USA, May 2007.

[106] S. Thomke, "Enlightened experimentation. The new imperative for innovation," *Harvard Business Review*, vol. 79, no. 2, pp. 66–75, 2001.

[107] L. E. Holmquist, "User-driven innovation in the future applications lab," in *Proceedings of the Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*, pp. 1091–1092, ACM, Vienna, Austria, April 2004.

[108] N. Maiden, C. Ncube, and S. Robertson, "Can requirements be creative? Experiences with an enhanced air space management system," in *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*, pp. 632–641, IEEE, Minneapolis, Minn, USA, May 2007.

[109] A. Dearden and S. Howard, "Capturing user requirements and priorities for innovative interactive systems," in *Proceedings of the Australasian Computer Human Interaction Conference (OzCHI '98)*, pp. 160–167, IEEE, Adelaide, Australia, December 1998.

[110] N. Maiden, S. Manning, S. Robertson, and J. Greenwood, "Integrating creativity workshops into structured requirements processes," in *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '04)*, pp. 113–122, ACM, Cambridge, Mass, USA, August 2004.

[111] H. A. Koski, "The installed base effect: some empirical evidence from the microcomputer market," *Economics of Innovation and New Technology*, vol. 8, no. 4, pp. 273–310, 1999.

[112] G. H. Walker, N. A. Stanton, and M. S. Young, "Where is computing driving cars? A technology trajectory of vehicle design," *International Journal of Human Computer Interaction*, vol. 13, no. 2, pp. 203–229, 2001.

[113] P. T. Helo, "Technology trajectories in mobile telecommunications: analysis of structure and speed of development," *International Journal of Mobile Communications*, vol. 1, no. 3, pp. 233–246, 2003.

[114] F. Hacklin, V. Raurich, and C. Marxt, "How incremental innovation becomes disruptive: the case of technology convergence," in *Proceedings of the IEEE International Engineering Management Conference: Innovation and Entrepreneurship for Sustainable Development (IEMC '04)*, vol. 1, pp. 32–36, Singapore, October 2004.

[115] N. Bakkalbasi, K. Bauer, J. Glover, and L. Wang, "Three options for citation tracking: Google Scholar, Scopus and Web of Science," *Biomedical Digital Libraries*, vol. 3, article 7, 2006.

[116] N. Maiden, A. Gizikis, and S. Robertson, "Provoking creativity: imagine what your requirements could be like," *IEEE Software*, vol. 21, no. 5, pp. 68–75, 2004.

[117] M. Khurum, S. Fricker, and T. Gorschek, "The contextual nature of innovation—an empirical investigation of three software intensive products," *Information and Software Technology*, vol. 57, no. 1, pp. 595–613, 2015.

[118] J. L. Carlo, K. Lyytinen, and G. M. Rose, "Internet computing as a disruptive information technology innovation: the role of strong order effects," *Information Systems Journal*, vol. 21, no. 1, pp. 91–122, 2011.

[119] C. Raasch, "The sticks and carrots of integrating users into product development," *International Journal of Technology Management*, vol. 56, no. 1, pp. 21–39, 2011.

[120] F. T. Igira, "The situatedness of work practices and organizational culture: implications for information systems innovation uptake," *Journal of Information Technology*, vol. 23, no. 2, pp. 79–88, 2008.

[121] M. E. Adams, G. S. Day, and D. Dougherty, "Enhancing new product development performance: an organizational learning perspective," *Journal of Product Innovation Management*, vol. 15, no. 5, pp. 403–422, 1998.

[122] D. R. Compeau, D. R. Meister, and C. A. Higgins, "From prediction to explanation: reconceptualizing and extending the perceived characteristics of innovating," *Journal of the Association of Information Systems*, vol. 8, no. 8, pp. 409–439, 2007.

[123] M. Sojer and J. Henkel, "Code reuse in open source software development: quantitative evidence, drivers, and impediments," *Journal of the Association of Information Systems*, vol. 11, no. 12, pp. 868–901, 2010.

[124] C. L. de la Barra and B. Crawford, "Fostering creativity thinking in agile software development," in *HCI and Usability for Medicine and Health Care*, vol. 4799 of *Lecture Notes in Computer Science*, pp. 415–426, Springer, Berlin, Germany, 2007.

[125] G. M. Marakas and J. J. Elam, "Creativity enhancement in problem solving: through software or process?" *Management Science*, vol. 43, no. 8, pp. 1136–1146, 1997.

[126] Y. Huang, P. V. Singh, and K. Srinivasan, "Crowdsourcing new product ideas under consumer learning," *Management Science*, vol. 60, no. 9, pp. 2138–2159, 2014.

[127] R. Zmud, "The effectiveness of external information channels in facilitating innovation within software development groups," *MIS Quarterly*, vol. 7, no. 2, pp. 43–59, 1983.

[128] K. J. Boudreau, "Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation," *Organization Science*, vol. 23, no. 5, pp. 1409–1427, 2012.

[129] A. Weterings and R. Boschma, "Does spatial proximity to customers matter for innovative performance? Evidence from the Dutch software sector," *Research Policy*, vol. 38, no. 5, pp. 746–755, 2009.

[130] S.-C. Hung and R. Whittington, "Agency in national innovation systems: institutional entrepreneurship and the professionalization of Taiwanese IT," *Research Policy*, vol. 40, no. 4, pp. 526–538, 2011.

[131] L. Mich, D. M. Berry, and C. Anesi, "Applying a pragmatics-based creativity-fostering technique to requirements elicitation," *Requirements Engineering*, vol. 10, no. 4, pp. 262–275, 2005.

[132] M. Sarma and T. Matheus, "'Hybrid' open source software virtual communities of practice—a conceptual framework," *Technology Analysis & Strategic Management*, vol. 27, no. 5, pp. 569–585, 2015.

[133] E. Carayannis and J. Coleman, "Creative system design methodologies: the case of complex technical systems," *Technovation*, vol. 25, no. 8, pp. 831–840, 2005.

[134] C. Koch, "Innovation networking between stability and political dynamics," *Technovation*, vol. 24, no. 9, pp. 729–739, 2004.

[135] M. Qin, "Determinants of information system innovation behavior in enterprises: an empirical investigation," in *Proceedings of the International Conference on Management of e-Commerce and e-Government (ICMECG '09)*, pp. 223–227, IEEE, Nanchang, China, September 2009.

[136] L. F. Higgins, "A comparison of scales for assessing personal creativity in IS," in *Proceedings of 29th Hawaii International Conference on System Sciences*, vol. 4, pp. 13–19, IEEE, Wailea, Hawaii, USA, January 1996.

[137] J. L. Sampler and D. F. Galletta, "Individual and organizational changes necessary for the application of creativity techniques in the development of information systems," in *Proceedings of the 24th Annual Hawaii International Conference on System Sciences*, vol. 4, pp. 404–411, IEEE, Kauai, Hawaii, USA, January 1991.

[138] T. A. Snow and J. D. Couger, "Creativity improvement intervention in a system development work unit," in *Proceedings of the 24th Annual Hawaii International Conference on System Sciences*, vol. 4, pp. 412–418, IEEE, Kauai, Hawaii, USA, January 1991.

[139] N. Maiden, S. Robertson, and J. Robertson, "Creative requirements: invention and its role in requirements engineering," in *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*, pp. 1073–1074, ACM, Shanghai, China, May 2006.

[140] K. Conboy, X. Wang, and B. Fitzgerald, "Creativity in agile systems development: a literature review," in *Information Systems—Creativity and Innovation in Small and Medium-Sized Enterprises: IFIP WG 8.2 International Conference, CreativeSME 2009, Guimarães, Portugal, June 21–24, 2009. Proceedings*, G. Dhillon, B. C. Stahl, and R. Baskerville, Eds., vol. 301 of *IFIP Advances in Information and Communication Technology*, pp. 122–134, Springer, Berlin, Germany, 2009.

[141] N. Maiden and S. Robertson, "Integrating creativity into requirements processes: experiences with an air traffic management system," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE '05)*, pp. 105–114, IEEE, September 2005.

[142] L. Mich, C. Anesi, and D. M. Berry, "Requirements engineering and creativity: an innovative approach based on a model of the pragmatics of communication," in *Proceedings of the 10th Anniversary International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '04)*, pp. 1–15, Riga, Latvia, June 2004.

[143] P. McBreen, "Creativity in software development," in *Proceedings of the Tools 39: Technology of Object-Oriented Languages and Systems, Software Technology for the Age of the Internet*, Q. Y. Li, R. Riehle, G. Pour, and B. Meyer, Eds., p. 390, Santa Barbara, Calif, USA, August 2001.

[144] S. Hanninen, "The 'perfect technology syndrome': sources, consequences and solutions," *International Journal of Technology Management*, vol. 39, no. 1-2, pp. 20–32, 2007.