AN INTEGRATED MODEL OF INNOVATION DRIVERS FOR SMALLER SOFTWARE FIRMS

High technology innovation is essential for economic development in industrialised societies. Innovation practice in smaller software companies, however, has received little attention. We derive software innovation drivers and outputs from a fragmented literature and analyse their empirical relevance using qualitative data from twenty-five in-depth interviews with software executives in the Silicon Fen. Repeating patterns in the dataset revealed through content analysis show that the most important innovation drivers for smaller software firms are external knowledge, leadership and team process. Specialised innovation tools and techniques are hardly used. We develop a model of software innovation drivers, together with explorative theoretical propositions.

Keywords: software, information system development, creativity, invention, innovation, management, small and medium size enterprises, knowledge leverage

1. INTRODUCTION

Technological innovation has long been associated with entrepreneurship, market power and economic growth, and widely researched by economists, and organisational and management theorists. High technology industries (including the software industry) are particularly dependent on innovation, and provide many high-growth firms. The high technology sector is important for national economies because of its ability to stimulate jobs and growth through high levels of invention and innovation. The result can be new industries with high profits, competitive edge and good salaries (Oakey, 2012). The US National Science Foundation reports that knowledge and technology industries 'have a much higher incidence of innovation than other industries and that 'software firms lead …….., with 69% of companies reporting the introduction of a new product or service.[1]' Thus the performance of software companies has broad economic consequences: "the software sector has effects that spill over beyond its specific niche, particularly as a widening array of economic activity, goods, and services rely to some extent on software-related technologies. Since these technologies promise to command a greater share of economic activity, the size and effectiveness of investment in software-related R&D may determine economic performance and international competitiveness more broadly" (Lippoldt & Stryszowski, 2009). Moreover the importance of the sector is not confined to large companies; high-tech start-ups are a motor for economic growth and a catalyst for technical innovation in societies (Oakey, 2012). However research and development in small high technology small firms remains risky – success in the development of leading edge technology is never guaranteed, and may be both expensive and time-consuming. Small and medium sized software enterprises (SMSEs) operate in difficult competitive conditions as a result of their size in relation to their competitors (Heirman & Clarysse, 2007). They often operate with constrained resources (especially for investment in new projects), specialist skill shortages, and a small customer base over which they have little control. They face entry barriers imposed by larger competitors (Ojala & Tyrväinen, 2006), challenges with internationalization and markets distorted by the availability of free software, and are therefore often confined to niche markets of their own development. One important response to these difficult conditions is the ability to innovate; innovation facilitates the development of novel value for customers, streamlines internal development processes, and opens market spaces that are not yet dominated by larger competitors.

[1] http://www.nsf.gov/statistics/seind14/index.cfm/chapter-6/c6s4.htm

Innovation involves 'the generation, development, and adaptation of novel ideas on the part of the firm' (Trott, 1998), where novelty is accompanied by utility, or value for the firm and its customers. Some researchers link innovation with the creation of new knowledge: 'innovation, which is a key form of organizational knowledge creation, cannot be explained sufficiently in terms of information processing or problem solving. Innovation can be better understood as a process in which the organization creates and defines problems and then actively develops new knowledge to solve them' (Nonaka, 1994). However innovation is normally understood as complex and multi-faceted: 'innovation is not a single action but a total process of interrelated sub processes. It is not just the conception of a new idea, nor the invention of a new device, nor the development of a new market. The process is all these things acting in an integrated fashion" (Trott, 1998).

Innovation in SMSEs requires independent study for two interlinked reasons. The first is that smaller firms may innovate in different ways than large firms. Their innovation advantages tend to be linked to behaviour - entrepreneurial dynamism, flexibility, efficiency, proximity to the market, motivation; whereas the advantages of larger firms are material - economies of scale and scope, financial and technological resources (Love & Roper, 2015). Innovation may be informal, ad hoc and opportunistic, integrated with daily work (in our case software development) – and primarily focused on design. SMEs have a low degree of job specialisation (Wong & Aspinwall, 2004) and do not normally have specialist innovation or research and development departments. Their innovation may involve cooperative and open strategies, and be led by owner-manager-decision makers who are well integrated into the everyday work (Supyuenyong, Islam, & Kulkarni, 2009). It is likely to be financed through bootstrapping (Aaen & Rose, 2011), since smaller firms have greater difficulty raising capital. The second reason why SMSEs are deserving of independent study is that innovation with software may be different from innovation in other sectors, because of the special characteristics of software and its development. Software innovation, according to the OECD, can be defined as

- 'the development of a novel aspect, feature or application of an existing software product or process; or
- introduction of a new software product, service or process or an improvement in the previous generation of the software product or process; and
- entry to an existing market or the creation of a new market.' (Lippoldt & Stryszowski, 2009)

Pikkarainen et al. (2011) argue that software innovation differs from other forms of innovation. Software is intangible, highly malleable, has a low market entry threshold, and often depends on the input of users and experts. Moreover the cost of software is focused in its development; reproduction and distribution costs are negligible. Rose (2010) points out that that the forces of globalisation, standardisation, and industrialisation are forcing software development firms in developed countries to become increasingly reliant on their innovation skills. However, software has particular design characteristics, and software companies operate in particular ways, so it cannot safely be assumed that innovation studies from other industries are directly transferrable - especially not to SMSEs.

Researchers have identified and studied many different facets of software innovation. Early contributions focused on creativity and creativity techniques in systems development (Couger, Higgins, & McIntyre, 1993), innovation leadership (Mclean & Smits, 1993) and creative requirements analysis (Maiden, Manning, Robertson, & Greenwood, 2004). A parallel trend in the organization and management sciences focused on open innovation (Chesbrough, 2003) and open source development (Von Hippel & Von Krogh, 2003). Disruptive innovation has more recently become a focus in IS (Lyytinen & Rose, 2003). Overall, however the literature reflects the complex and multi-faceted nature of the subject; many fragmented contributions from several disciplines, many different related foci, little cross-disciplinary referencing, and thus a lack of cumulative knowledge generation in the area. Moreover, there is little consistent focus on SMSEs – much of the literature focuses on larger companies, some contributions do not distinguish on the basis of company size and only a few researchers (Carlo, Lyytinen, & Rose, 2011; Koc, 2007; Raffa & Zollo, 1994; Romijn & Albaladejo, 2002; Tjornehoj & Mathiassen, 2010; Weterings & Koster, 2007) explicitly target SMSEs. It is currently hard to distinguish what drives innovation in larger software companies from what drives it in SMSEs. Our research questions are

therefore: which organizational levers drive innovation in SMSEs, and how are they related? We primarily consider the work of software developers and their team leaders and managers, the artefacts or products they develop, and the processes they use to develop these artefacts. Our analysis thus spans individuals, teams, and organizations.

The starting point for the study is a literature study identifying the drivers of software innovation (irrespective of size). This provides the initial conceptual framework for semi-structured interviews with experienced software developers in the Silicon Fen. The Silicon Fen is a regional innovation cluster in the East of England centred around Cambridge with a high concentration of small and medium sized software companies. The name Silicon Fen alludes to Silicon Valley in California, and the former wetlands in this area known locally as The Fens. The transcribed interviews were explored through content analysis for structural patterns. Concepts from literature are in this way filtered and refined into an exploratory descriptive theory of software innovation in SMSEs. These methodological considerations are reported in section 3, and the results of the analysis in sections 4 and 5. Section 6 presents the refined concept set as overview and detailed models with a related set of exploratory propositions, and the article ends with a discussion and conclusions.

2. SOFTWARE INNOVATION: OUTPUTS AND DRIVERS

Software innovation outputs

The most common form of software innovation results in the creation of new software functionality used in new *products and services*. 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 software, and embedded software (Rose, 2010). Many forms of software are referred to as services; such as web services or mobile services (Kristensson, Magnusson, & Matthing, 2002). A wide range of software-related activities such as installation, customization, helpdesk, platform management, and consulting can also be referred to as services. In addition, hosting or application service provision represents a combination of software with additional services that permit organizational computing functions to be outsourced to software providers. A modern variant of such an offering is software as a service (SaaS) (Lippoldt & Stryszowski, 2009). *Software process innovation* focuses on the tasks and actions, the shapes and norms, and the formal and informal procedures that lie behind software development. These are expressed in the methods, tools, and techniques that organize the work of a developer, and describe how software is developed (Rose, 2010). Carlo et al. (2011a) define this as innovation in ways to envision, design, and implement software. All significant improvements in design techniques, team organisation, and managerial processes can be classified as process innovations. Product/service innovation and process innovation constitute the two main innovation outputs for this study

Management drivers

An important group of software innovation drivers reside with those taking a leadership role, whether formally as a manager or project-leader, or informally as part of a project group. *Innovation leadership*, monitoring and feedback for project teams, is identified as having an important influence on software innovation. The IS leader may be the champion of innovation (McLean & Smits, 1993). Leaders are responsible for fostering a *work environment* that stimulates creativity and minimizes barriers to creativity, and Florida & Goodnight (2005) characterize such efforts as minimizing hassles and stimulating minds. Leaders are often responsible for *path creation* (Gumusluog & Ilsev, 2009), which guides organizations through changes in base technologies and market and product segments that are beyond the considerations of individual developers. Leaders are also responsible for *portfolio managemen*t (Napier, Mathiassen, & Robey, 2011) *conflict resolution* (Sherif, Zmud, & Browne, 2006) and providing feedback. The most valuable feedback activity is *innovation evaluation* - assessing the work environment, the value of competing ideas during ideation, and new software product concepts (Lobert & Dologite, 1994). Such evaluation may be formal or informal. Informal evaluation of team innovation may be important for the process organizer in deciding to how to manage the project - perhaps taking

the form of observations of team performance (Rose, 2010). Formal evaluation is more dependent on the development and use of specific metrics and targets (Lobert & Dologite, 1994).

Knowledge drivers

A second group of factors understood to be important in software innovation refer to *knowledge leverage* and in particular the role of knowledge external to the development team (Zmud, 1983). Attention has been directed toward the role of *absorptive capacity*: the ability of a firm to identify, assimilate, and exploit external knowledge. Carlo et al. (2011) isolate knowledge depth, diversity and linkages, and routines of sensing and experimenting as important for software innovation. *Market understanding* and *technology trajectory understanding* are important forms of knowledge - taken together they represent the well-known complement of market pull and technology push (Brem & Voigt, 2009). Knowledge of *competitors* and their innovations is also considered (Turner, Mitchell, & Bettis, 2010). A final type of knowledge generally considered a driver for innovation is *user-domain understanding* generated from customers (Lee & Cole, 2003). Knowledge creation and use is understood to be a social process and innovation researchers thus emphasize importance of *community and network* (Franke & Von Hippel, 2003) in progressing knowledge development. In particular, the software industry has witnessed the emergence of a specific form of community-based innovation in the form of the *open source* movement. According to Von Hippel and Von Krogh (2003), open source development is an example of a private-collective model of innovation not previously seen in either private industry or in the collective knowledge creation efforts of universities. Some forms of open source development can also be understood as user-driven software development processes involving open access to intellectual property (such as source code) in a model known as *open innovation* (Chesbrough, 2003). Researchers are also interested in the role of *crowd-sourcing* in software innovation (Leimeister, Huber, Bretschneider, & Krcmar, 2009) involving, for example social networks (Gray, Parise, & Iyer, 2011). Some important forms of knowledge are held by users and recent research has stressed the role of *user involvement* in software innovation (Bogers, Afuah, & Bastian, 2010). Users may include end users, customers, other developers adapting software, and firms buying software products or services. Customers can play an important role in the commercialization of software inventions (Athaide, Meyers, & Wilemon, 1996), helping with *customization*, requirements, and early investment. User integration is thought to play an important role in innovation with agile methods (Gassmann, Sandmeier, & Wecht, 2006) and users often produce creative ideas (Kristensson et al., 2002), especially in respect to service innovation. Moreover users with special skills which enable them to help conceptualise and prototype software systems (*lead users*) are important in *user-driven* innovation (Franke & Von Hippel, 2003) where sticky knowledge makes it difficult for software engineers to understand the use domain.

Team process drivers

Software is almost always produced in a team, in which the creative ideas of the team members (often drawn from external knowledge sources) are synthesized into code outputs that form the product offerings of the company. Several factors are important in this team process. *Creative cognition* involves understanding the creative state of mind and creative acts in software development (Couger et al., 1993). Idea generation, or *generative capacity* (Avital & Te'eni, 2009) describes '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 is improved by *ideation,* the evaluation, improvement and realisation of ideas (Brem & Voigt, 2009). Ideation is naturally performed in a team, and *teamwork* is considered an essential feature of innovative projects (Hoegl & Gemuenden, 2001), contributing to team efficiency and the personal satisfaction of team members. *Team composition*, the blend of experience and competences is important for innovation (Cooper, 2000). Tiwana and McLean (2005) highlight *expertise integration,* the capacity to exploit knowledge transfer between team members who possess different skills. A further important aspect is the development of *shared understanding* and relational capital among team members (Koc, 2007). The software team's work is often assumed to benefit from a repertoire of *innovation tools and techniques* as well as situational knowledge of when to apply them. Couger et al. (1993) report on the use of

analytical *creative techniques* (progressive abstraction, interrogatories and force field analysis) and intuitive techniques (associations/images, wishful thinking, and analogy/metaphor) to support creativity in the systems development effort. A related literature explores software systems designed to underpin creative work - *creativity support tools* (Shneiderman, 2007). A specialized form of *user toolkit* (Franke & Von Hippel, 2003) is designed to help end users in the innovation process. Innovation tools and techniques may form part of the team's *development framework,* its processes, underlying assumptions, and work practice norms. Process-oriented software innovation strategies include Aaen's (2008) Essence framework, and *creative requirements analysis* (Maiden et al., 2004). A strand of literature associates *agility* (agile methods) with creativity in development (Highsmith & Cockburn, 2001). Another related stream of research focuses on *experimentation* in the design process (Thomke, 2001; Carlo et al., 2011). In software development contexts this usually involves the use of *prototyping*, particularly where low-cost low-technology strategies are favoured (Martin, 2011). A further aspect of the development framework is the *installed base* (Carlo et al., 2011) with which the team works: those programming languages, application programming interfaces, standards and development environments with which they are familiar. The teams ideas, supported by their process must eventually be expressed as s*oftware design capability*, which is defined as the ability to design a technology *concept* (Leonardi, 2011) - an innovator's vision of what functionality the built technology (the technological artefact) should have, here understood as a novel and useful *feature set* (Roberts, 1988).

Table 1 summarizes innovation outputs and drivers from this discussion.

Concept	Definition	Key references
Output: Product/service innovation	Novel and useful software products and services representing a significant advance or change in direction for a company	Too numerous to list
Output: Process innovation	Step-changes or significant modifications in the processes used to develop software products and services	Too numerous to list
Management Driver: Innovation Leadership	Managing development teams to create innovation	(Boland Jr., Lyytinen, & Yoo, 2007; R. G. Cooper, 2011; Gassmann et al., 2006; Martin, 2011; Mclean & Smits, 1993; Nambisan, Agarwal, & Tanniru, 1999; Napier et al., 2011; Romijn & Albaladejo, 2002; van den Ende & Wijnberg, 2003)
Work Environment	Promoting a creative work environment, minimising creativity barriers	(Cooper, 2011; Florida & Goodnight, 2005; Hocova, Cunha, & Staníček, 2009; Maccrimmon & Wagner, 1994; Napier et al., 2011; Romijn & Albaladejo, 2002)
Path Creation	Creating an overall sense of direction in response to market and technology developments	(Boland et al., 2007; Gumusluog & Ilsev, 2009; Napier et al., 2011; van den Ende & Wijnberg, 2003; Weterings & Boschma, 2009; Weterings & Kotooster, 2007; Yang & Hsiao, 2009)

Portfolio Management	Steering multiple projects in respect to innovation challenges	(Hocova et al., 2009; Napier et al., 2011)
Conflict Resolution	Resolving conflicts between individuals and groups in the pursuit of innovation	(Sherif et al., 2006)
Management Driver: Innovation Evaluation	The ability to reflectively evaluate ideas, techniques and processes for their contribution to innovation	(Briggs & Reinig, 2010; Compeau, Meister, & Higgins, 2007; Higgins, 1996; Koc, 2007; Lamastra, 2009; Lobert & Dologite, 1994; Massetti, 1996; Müller & Ulrich, 2012; Sosa, 2011)
Knowledge Driver: Knowledge Leverage	The use of internal or external knowledge to drive software innovation	(Cooper, 2000; Cooper, 2011; Gassmann et al., 2006; Hanninen, 2007; Heirman & Clarysse, 2007; Hung & Whittington, 2011; Lee & Cole, 2003; Morrison, Roberts, & von Hippel, 2000; van den Ende & Wijnberg, 2003; Weterings & Boschma, 2009; Weterings & Koster, 2007; Zmud, 1983; Yang & Hsiao, 2009)
Absorptive Capacity	The ability of a development team to find, adapt and exploit external knowledge in software innovation	(Adams, Day, & Dougherty, 1998; M. Bogers, Afuah, & Bastian, 2010; Carlo et al., 2011; R. G. Cooper, 2011; Nambisan et al., 1999; Napier et al., 2011; Sosa, 2011; West & Gallagher, 2006)
Market Understanding	The use of information about software markets to promote product innovation	(Adams et al., 1998; Brem & Voigt, 2009; R. G. Cooper, 2011; Hung & Whittington, 2011; Napier et al., 2011; Turner et al., 2010; van den Ende & Wijnberg, 2003; Yang & Hsiao, 2009)
Technology Trajectory Understanding	The use of understandings of the probable direction of future evolution of software and hardware infrastructures, platforms and technologies to guide innovation	(Aerts, Goossenaerts, Hammer, & Wortmann, 2004; Boland Jr. et al., 2007; Brem & Voigt, 2009; R. G. Cooper, 2011; Hanninen, 2007; Hung & Whittington, 2011; Napier et al., 2011; Romijn & Albaladejo, 2002; Yang & Hsiao, 2009)
User Domain Understanding	Using understandings of customers' business domain or specialised internal knowledge to drive innovation	(Gray et al., 2011; Hanninen, 2007; Igira, 2008; Koc, 2007; Lee & Cole, 2003; Martin, 2011; Mich, Berry & Anesi, 2005; Raasch, 2011; Weterings & Boschma, 2009; Yang & Hsiao, 2009)
Competitor Understanding	Monitoring competitors' processes, products and services to inform innovation	(Turner et al., 2010; Cooper, 2011)
Knowledge Driver: Community	Exploiting external	(Boland Jr. et al., 2007; Heirman &

and Network	connections, collaborations and partnerships to promote innovation	Clarysse, 2007; Franke & Von Hippel, 2003; Henkel, 2006; Hung & Whittington, 2011; Lee & Cole, 2003; Leimeister et al., 2009; Morrison et al., 2000; Pisano & Verganti, 2008; Romijn & Albaladejo, 2002; Sosa, 2011; van den Ende & Wijnberg, 2003; West & Gallagher, 2006)
Open Innovation	Using open business models that partially or wholly share intellectual property (for example code) to promote innovation	(Bogers et al., 2010; Henkel, 2006; Lee & Cole, 2003; Leimeister et al., 2009; West & Gallagher, 2006)
Open Source	Exploiting open source code or co-operations to drive innovation	(Bogers et al., 2010; Henkel, 2006; Igira, 2008; Lamastra, 2009; Von Krogh, Spaeth, & Lakhani, 2003)
Crowd Sourcing	Inviting the wide-spread participation of potential users and customers to enhance innovation	(de Jong & von Hippel, 2009; Gray et al, 2011; Leimeister et al., 2009)
Knowledge Driver: User Involvement	Involving users to stimulate innovation	(Athaide et al., 1996; Bogers et al., 2010; Compeau et al., 2007; Franke & Von Hippel, 2003; Kristensson et al., 2002; Leimeister et al., 2009; Martin, 2011; Oliveira & Von Hippel, 2011; Raasch, 2011)
Customisation	Involving users in customisation of standard products and services	(Athaide et al., 1996)
User-Driven/Lead User	Facilitating expert users with specialist competences in directing software innovation	(Franke & Von Hippel, 2003; Kristensson et al., 2002; Morrison et al., 2000; Nambisan et al., 1999; Napier et al., 2011; Oliveira & Von Hippel, 2011)
Team Process Driver: Creative Cognition	The exploitation of individual cognitive creativity for innovation	(Avital & Te'eni, 2009; Cooper, 2000; Couger et al., 1993; Maccrimmon & Wagner, 1994; Massetti, 1996; Santanen, et al., 2004)
Generative Capacity	The ability to generate creative ideas and solutions promoting innovation	(Avital & Te'eni, 2009; Kristensson et al., 2002; Leimeister et al., 2009; Massetti, 1996; Pisano & Verganti, 2008; Romijn & Albaladejo, 2002; Santanen et al., 2004; Shneiderman, 2000; Sosa, 2011)
Ideation Expertise	The ability to refine and exploit creative ideas to	(Brem & Voigt, 2009; Cooper, 2011; Santanen et al., 2004; Shneiderman, 2000)

	promote innovation	
Team Process Driver: Software Design Capability	The ability to design innovative software products and services	(April & Busse, 2007; Carayannis & Coleman, 2005; Quintas, 1994; Sas & Zhang, 2010)
Concept	The ability to develop overall concepts for new products and services	(April & Busse, 2007; Carayannis & Coleman, 2005; Leonardi, 2011; Quintas, 1994)
Feature Set	The ability to create distinct sets of novel and useful software functionality	(Carayannis & Coleman, 2005; Leonardi, 2011; Quintas, 1994: Roberts, 1988)
Team Process Driver: Teamwork	Organising teamwork to promote innovation	(Cooper, 2000; Couger et al., 1993; Hoegl & Proserpio, 2004; van den Ende & Wijnberg, 2003)
Team Composition	Selection of team members to promote innovation	(Aaen, 2008; Cooper, 2000; Hocova et al., 2009; Koc, 2007; Tiwana & McLean, 2005)
Expertise Integration	Facilitating dialogue between experts with different technical and non-technical specialisations	(Heirman & Clarysse, 2007; Leonardi, 2011; Tiwana & McLean, 2005; Von Krogh et al., 2003; Weterings & Koster, 2007)
Shared Understanding	Building and maintaining a team's common purpose in the face of many challenges and direction changes	(Cooper, 2000; Hesmer, Hribernik, Hauge, & Thoben, 2011; Hocova et al., 2009; Koc, 2007; Lu & Wang, 2007; Snow, Fjeldstad, Lettl, & Miles, 2011; Tiwana & McLean, 2005)
Team Process Driver: Innovation Tools & Techniques	Using tools and techniques designed to promote creativity in the development process	(Carayannis & Coleman, 2005; Cooper, 2000)
Creativity Techniques	The use of conceptual tools (such as mind-mapping) to support innovation	(Amoroso & Couger, 1995; Carayannis & Coleman, 2005; Cooper, 2000; Couger et al., 1993; Maccrimmon & Wagner, 1994; Santanen et al., 2004)
Creativity Support Tools	The use of computerised tools designed to facilitate creativity to support innovation	(Avital & Te'eni, 2009; Maccrimmon & Wagner, 1994; Massetti, 1996; Shneiderman, 2000; Shneiderman, 2007)
User Toolkits	The deployment of tools (often computerised) to facilitate user innovation, often with respect to a technology platform	(Franke & Von Hippel, 2003; Müller & Ulrich, 2012; Quintas, 1994; West & Gallagher, 2006)

Team Process Driver: Development Framework	The concepts, methods and techniques used to underpin software team's development effort in respect to innovation	(Aaen, 2008; R. B. Cooper, 2000; Highsmith & Cockburn, 2001; Maiden, Gizikis, & Robertson, 2004; Maiden, Manning, Robertson, & Greenwood, 2004; Quintas, 1994) (April & Busse, 2007)
Agility	Use of agile methods, or adaptations of agile methods as an innovation driver	(Aaen, 2008; April & Busse, 2007; Gassmann et al., 2006; Highsmith & Cockburn, 2001)
Creative Requirements Analysis	Stimulating requirements gathering by use of techniques designed to increase users' an customers' creativity	(Cooper, 2000; Hesmer et al., 2011; Hocova et al., 2009; Maiden et al., 2004b; Mich et al., 2005)
Experimentation/Prototyping	Stimulating creativity by iterative use of experimentation and/or prototyping in the development process	(Carlo et al., 2011; Holmquist, 2004; Martin, 2011; Thomke, 2001)
Installed Base	Exploiting the technical development environment of a software firm to generate innovation	(Aerts et al., 2004; Boland et al., 2007)

Table 1. Drivers and outputs for software innovation

The principal outputs are software process innovation and software product/service innovation, where process innovation is understood also to influence product and service development. Other concepts are understood to facilitate innovation, with a central group of influences closely associated with the team process (development framework, innovation tools and techniques, creative cognition, teamwork and software design capability). Innovation leadership and evaluation primarily influence the team process, whereas knowledge leveraged from users and community and network influence both process and products and services.

3. RESEARCH APPROACH

Having identified a wide range of concepts related to software innovation from the literature, we now proceed to refine the concepts and target them better towards SMSEs through an empirical analysis of their application in SMSEs in a significant regional innovation cluster. Quantitative techniques are not appropriate for integrative studies, since they cannot accommodate many variables with complex patterns of associations. The empirical work is therefore a pre-structured qualitative investigation (Jansen, 2010) where the objective is 'to gather data on attitudes, opinions, impressions and beliefs of human subjects' (Jenkins, 1985). Qualitative surveys aim at determining the 'diversity of some topic of interest within a given population' and establish 'the meaningful variation (relevant dimensions and values) within that population' (Jansen, 2010). The population here consists of experienced software developers working in companies in the Silicon Fen, and the sample is self-selected through interest in the topic. The main topics, dimensions and categories were defined beforehand (Table 1), and explored by means of semi-structured interviews (contextualized through workshops and web-site study).

The empirical setting for the study is the Silicon Fen. The Silicon Fen, sometimes known as Europe's Silicon Valley (Koepp, 2003), is a grouping of high-tech businesses focusing on software, electronic and biotechnology located around Cambridge in Eastern England. They are distributed in an area bounded by Ely, Newmarket, Saffron Walden, Royston and Huntingdon – all roughly a half-hour's drive from Cambridge. The origins of the cluster date back to the establishment of Cambridge Science Park by Trinity College in 1970, and many of the companies have some connections with the university. There were reported to be 1,379 high-tech companies employing 48,099 people in Cambridge and South Cambridge at the last census in 2008 (Doel, 2011). The Cambridge cluster map[2] currently lists 332 information technology (IT) and telecommunications companies, of which a few (e.g. ARM, Aveva and Autonomy Systems Ltd.) are major international companies. Fifty-two of these companies are spin-offs or otherwise closely linked to the university. However only six reported more than 250 employees or revenues of more than €50M; the IT and telecommunications cluster can therefore be described as predominantly made up of small, medium and micro sized enterprises (according to the European Union's definitions[3]), which are the subject of our study.

The Silicon Fen is described as a 'cluster of creativity' by Koepp (2003), making it an excellent area to study innovation; however it is difficult to assess how representative the companies studied are of SMSEs in general. Two characteristics are known to be particular to the silicon clusters: the network effects of having many companies and technical specialists close to each other, and the knowledge effects of having a major university in the area. Nevertheless the companies, executives and senior developers participating in the study represent a varied sample of SMSEs. We contacted over 100 companies chosen at random, and the only obvious characteristic that the sample shares is that of self-selection (they had enough interest and resources to participate). Nineteen companies participated in the study - ranging in size from 3 to 120 people, and producing many different kinds of software, from companies working primarily with a single packaged product to companies developing many different products for clients. This yielded 16 short preparatory interviews with CEO's and 25 in-depth structured interviews (

[2] http://www.camclustermap.com

[3] http://ec.europa.eu/enterprise/policies/sme/facts-figures-analysis/sme-definition/index_en.htm

Appendix 1, Appendix 2) carried out in the first half of 2013. All of the companies had web sites, which were examined, and a total of 33 software developers attended four workshops on closely related themes. Most of the developers also participated in the interviews. Focus groups with feedback were held in two companies. Interviewees were chosen from company employees who both participated in software development, and had some form of managerial responsibility for it (for example as a project manager). The majority (17) were at senior executive level in their companies. Three companies provided more than one interview. The interview protocol (Appendix 3) was piloted and iteratively improved over the first 5 interviews. After initial introductions, and an explanation of the objectives of the research project, the interview was carried out in four parts.

- The interviewee was asked to provide basic factual information about themselves and their company, including, their name, role, company, size of company and the type of software they built
- The two outcomes (product/service innovation, process innovation) were explained and the interviewee was asked to provide a narrative (tell a story) of a successful innovation. This technique is known as priming by psychologists, and is designed to set up a stimulus in the interviewee's mind that will affect the way they later respond to questions. For instance any repetition of the word innovation will trigger, consciously or unconsciously, the narrative they have recently explored.
- The interviewees were given a list of the principal concepts of the study, with short explanations, and asked to prioritise them, in the sense that they should begin with the most important concept associated with their innovation experience and explain the association as they understood it, and continue with the next most important and continue as long as time remained. The interviewer was armed with a more extensive list of concepts to direct supplementary questions.
- In the final part of the interview, the concept list was removed and the interviewee asked to identify other contributing innovation factors that had not yet been identified.

Data analysis was conducted using content analysis (Berelson, 1952; Krippendorff, 2004; Silverman, 2001). As a technique, content analysis yields 'a relatively systematic and comprehensive summary or overview of the dataset as a whole' (Wilkinson, 1997;170). It operates by observing repeating themes and categorizing them using a coding system. Categories can be elicited in a grounded way or can (as in our case) originate from an external source such as a theoretical model (Wilkinson, 1997). Dedoose was used as the coding tool to facilitate on-line interaction. A formal two level coding scheme was developed from the major concepts summarised in Table 1. The coding was piloted and refined. Inter-coder reliability was achieved by using a total of four coders (the three authors and a PhD student). The first coder coded the majority of interviews to ensure consistency, with the three others also coding some complete interviews, and performing various consistency checks (Appendix 4). Open coding was allowed (but sparingly used) to facilitate the development of new concepts. Inter-coder reliability was thus built into the process, but no statistical test was carried out. The coders were careful to pay attention to relationships between innovation drivers and outputs; where these were signalled in the text they were coded with overlapping driver and output codes so that they could easily be traced and analysed later. A complete mapping of relationships by this method was not attempted because of the number of possible relationships: $(39\text{-}1)^{39}$ without the open codings. Code frequency over the dataset was interpreted as a validity signal, and code co-occurrence used as a signal for investigating potential concept associations. The patterns of code associations were tracked manually to unravel complex and mediated relationships. The eventual model was derived by an iterative refinement of concepts and associations according to patterns revealed in the dataset, which also forms the basis for the accompanying propositions.

Figure 1 gives the overall research process.

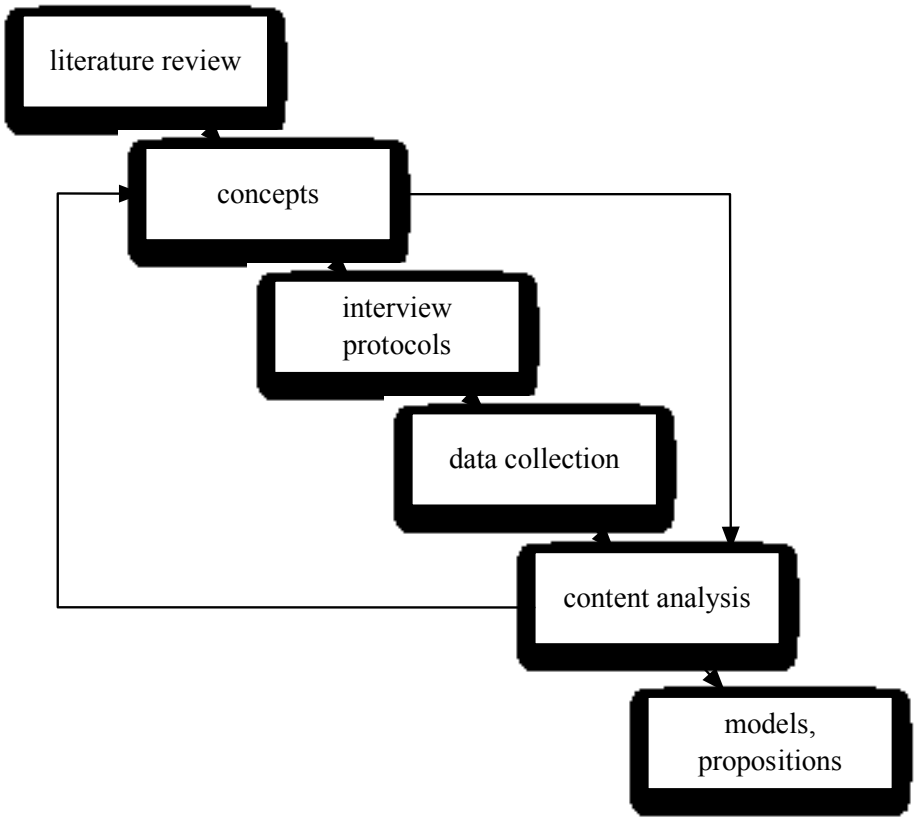


Figure 1. Research process

Two different analyses underpin the development of models and propositions.

Analysis 1 is based on coding frequency, and reports on the relative importance of the innovation drivers as reported by the interviewees. Appendix 5 shows coding frequencies for innovation outputs and drivers. Since interviewees were asked to prioritise the drivers in the context of their own innovations, these can be interpreted as representing their understanding of the relative importance of the drivers. Drivers with less than 20 codings were considered to lack empirical validation and removed from further consideration. Appendix 6 shows the distribution of high level codings, where each plot on the radar chart represents the sum of the high level coding and its second level decomposition. However a detailed ranking of drivers is not part of the purpose of this research; coding frequency is used to show the validity of the codes (they represent a significant part of the interviewees' discourse) and qualitative analysis (through tracking the coded passages of text) is used to investigate how and why they are considered important for innovation. Analysis 1 underpins the selection of concepts for the model presented in section 6.

Analysis 2 investigates the relationships between different drivers and outputs based on code co-occurrence, which signals patterns of associations between concepts (drivers and outputs) in the dataset, and explanations of how and why drivers impact innovation revealed in Analysis 1. The most significant co-occurrences ($n>20$) were tracked back to their contexts in the interviews and the nature of the relationships between drivers and outputs analysed. Appendix 7 summarizes typical explanations for the associations. These explanations are combined with the relational insights from Analysis 1 in the development of a series of propositions explaining the most important relationships between concepts.

4. ANALYSIS 1: INNOVATION DRIVERS FOR SOFTWARE COMPANIES IN THE SILICON FEN

In this section we summarize the major repeating themes of the analysis, illustrating them with quotations from the interviews.

Software innovation outputs: product/service, process

Product/service innovation

The interviewees described a wide variety of firm level innovations, mostly of an incremental rather than a radical nature. Two companies had been through major architectural restructuring of their principal software product, in one case to support a different use of the product, and in the other to make it suitable for use as an open platform (with an application programming interface) for other developers. A third had redeveloped their product range from a single customer, single platform system to a cross-platform system spanning most of the operating systems and database management systems used in their industry segment. Some described new product developments: a smart energy management switching system, a workflow control system, a video driver, an automated testing program, an interface design now widely used in smartphones. One described a programming platform change: one of their products would be migrated to a modern platform, involving updating the feature set to reflect modern technology affordances and coding the entire system from scratch in the new languages. Another interviewee focused on the development of original algorithms to solve customer problems; others focused on their recent projects: an open source collaboration for intelligent houses; a simplified way of organising natural language search for customer service involving new business models for customers; a major overhaul of product platforms and related consultancy practice in response to perceived technology trajectories.

Process innovation

Two companies described the introduction of customised forms of the agile method Scrum, one including major elements of Kanban. A third described the introduction of an ISO 9000 standard, a fourth the introduction of an idea management system with organisational processes to support it[4], a fifth a revision of consultancy support for their core project.

Management drivers: innovation leadership, evaluation

Innovation leadership

The most significant of the leadership activities discussed was *path creation.* SMSEs typically produce many creative ideas and product suggestions; so prioritising those that will be taken further, creating clarity of direction in their execution and seeing them through to completion are essential skills. "I've been able to use the ideas of the really bright guys around me, spot that talent, reinforce it, support it... recognize a good idea and go forward with it..... you have to have the knowledge and foresight of where you might want to get to as well as where you are…… you need utter determination and belief that you are right … there's an awful lot of practical difficulties and you just have to go round them, over them, through them or whatever if you're going to deliver the thing" (i15). Path creation supports shared understanding, and a common direction in teamwork[5]. It is dependent upon excellent knowledge of the environment; as one CEO puts it: "my own perception of where the company needs to go commercially….. we can't be in markets which are dying markets or flat markets...we need to be in buoyant markets where customers have money to spend" (i17)[6]. A further important leadership aspect

[4] see proposition 15
[5] see P6
[6] see P5

was creating a psychologically supportive *work environment* for the team[7], the "drive for execution, spirit, creativity, and fun….happy hacking on Friday afternoon" (i14), a culture of "full empowerment…. if somebody has an idea …. they're allowed to put forward the idea" (i10).

Innovation evaluation

In the SMSEs we investigated there was little formal evaluation of innovation. Instead aspects of product and process evaluation were incorporated into the work of leaders, often owner managers, or small groups of director/managers, working closely with their developers. Leaders controlled scarce resources and an important component of *path creation* was project selection - "sometimes ideas just won't fly…..the real, hard, commercial world, there're lots of reasons why you can just immediately see it isn't going to happen" (i21)[8]. Evaluation often involved resource prioritization and a commercial focus - "we've got these 30 things we need to build …. how many do we actually need before it's sold… we're deferring 20 until version 2… we've got to get 10 of them out there" (i5). These are also components of developing shared understandings and common direction in the team[9].

Knowledge drivers: knowledge leverage, community and network, user involvement

Knowledge leverage

Silicon Fen developers identify *knowledge leverage* as the most important contributor to innovation – "knowledge leverage is the key ….you have to know an awful lot of techniques and technologies and extrapolate beyond the known combinations …. to discover new elements" (i15). Knowledge is important for generative capacity[10]: "ideas come from three main sources …. the existing customer base, they're the ones who are driving we want this feature, feature X, Y, Z. ….. market direction …… the adoption of mobile technology and touch devices ….the third area would be where the company needs to go commercially….." (i17). *Absorptive capacity* describes the utilisation mechanism for four different types of knowledge. *User domain* knowledge is considered vital for innovation: "my boss spends a lot of time talking to the users and their managers as well. He gets to know the kind of things that they need to do and what the overall direction is as well…. ... when a shipping line is thinking about buying out another one or when their volumes are likely to increase or decrease" (i10). Some SMSEs have staff who focus on user relationships: "she knows her customers' business often better than many of them do. She works with water companies, and she knows all about their billing cycles. She works with the hotel trade. She gets to learn everything about them… They always feed back great ideas on what the software needs to do …. Some of that is very inspiring. We can't do it all, but we prioritize what we can" (i18)[11]. *Market understandings* help focus technology enthusiasts on business benefits: "don't fall in love with a product before you've done the market research. I've seen two people lose their houses because of that" (i5). However, software specialists often conflate *market* and *technology trajectory* knowledge: "we do some market analysis and we work with our existing customers on what kind of roadmaps they are thinking of and …. what platforms we are going to be using and what architectures we need" (i14). They use these forms of knowledge for idea generation[12]: "we keep an eye on technology roadmaps… we can spot technologies that will be useful to our customers in three or four years time and ….develop crucial concepts for particular technologies….. how we pick particular markets to focus on and what type of prototypes we develop and proposing those to customers," (i4) and predicting future needs: "I'm trying to be there when the curve is going up. I'm not jumping on the bandwagon when we've hit the peak and it's on the way down" (i13); "if you take a particular card or a

[7] see P6
[8] see P5
[9] see P6
[10] see P8
[11] see P2
[12] see P8

particular operating system, you're not picking it for what's the best you're picking it for the one you can use in 25 years' time" (i21). They also track their *competitors*, for instance to focus new product development: "I've done quite a bit of work on competitor analysis …. we're having a really good look internationally……we're ranking them ….. we quite happily nick good ideas from our competitors" (i3) [13] and to keep abreast of technical developments: "we have a business development manager …. watching the big hardware developers like Intel and Apple and generating intelligence for us" (i8). Many specialised niche software companies in the Silicon Fen also keep track of basic science development in their areas (e.g. advanced graphics): "we need to track the developments in the field…keep an eye on the literature……there's one big academic conference called SIGGRAPH and if we can get hold of the papers ….." (i8).

Community and network

Community and network are the source of much innovation-generating knowledge, both in terms of problem solving ("how did we ever write software before we had …web forums? …..it used to take ages to look up the answer to …my software is crashing here" (i12)), and user domain understanding ("part of the network is you're trying to work with accountants, both to understand their needs and that they will guide their clients towards the solutions" (i1))[14]. They stimulate generative capacity: "there's a government-sponsored specification …. and I sit on the technical committee for that…. [which] is bringing forward ideas" (i9). Pre-existing *open source* software can enable many innovations " you can't build this thing from scratch in three months with $50,000, it's not going to happen. You have to go out and find …. a piece of open source software. Then you've got to get a square peg into a round hole…….. it requires innovation. (i22). The community relationships continue to be important even where there is a business imperative - "from a hard-headed business standpoint if we make fixes and we don't contribute them back, we're going to have to make those same fixes over and over again, whereas if you contribute them upstream and they go into the upstream product, then it's less work for us and it'll save us money" (i23). Partnerships sometimes formalized relationships considered important for innovation; for example around open source development: "it's been a very, very good partnership…….their business model is somewhat different from us because a lot of what they do, they do it under the open source framework…… their core system is open source, but they have commercial plug-ins to that open source framework…..and they also provide tools for anyone to provide plug-ins to it ……..it allows us to sell our ideas on a new platform that we wouldn't otherwise be able to do…..if we have an idea for a thing that might impinge on one of their commercial products, we talk to them - is this is going to be competitive to you? We don't want that. They're very nice. We swap ideas" (i13). Various *open innovation* strategies are also prominent, such as open innovation tenders: "several pharmaceutical companies have got the same problem….rather than keeping it closed and innovating themselves …..... they've described the problem….. they've gone out to look for solutions to that problem……. across all the open innovation platforms that is a very common theme" (i22) [15].

User involvement

User involvement in the development process is the most important source of user domain understanding, and sometimes the starting condition for innovation: "somebody has got a problem and there is no solution to that problem….an innovative step is required to construct that solution. (i21). The input may come from end-users or customers who are themselves experienced engineers or scientists - "we start off being treated as suppliers, but it soon ends up that we're effectively colleagues or treated that way" (i23). These relationships can be inspiring: "bouncing ideas off, learning about what they're doing …the whole thing is actually a very innovative planet and it allows you to achieve …impressive results" (i22)[16]. Software innovators often apply a degree of interpretation to what they learn: "listening to the customers is a very valuable source of perception and cognition of what it is

[13] see P8
[14] see P1
[15] see proposition 1
[16] see P3

that they're trying to do…..I discriminate between what they say they want and what they need…… those are usually different…… …… the art of this is to listen to them and work out what those three different animals are….. sometimes you can infer the problem that they're actually trying to solve, which may not be the one that they've told you it is, but you may therefore be able to invent a new solution that solves that problem and a whole class of others in a new way" (i15).

Team process drivers: creative cognition, software design capability, teamwork, innovation tools and techniques, development framework

Creative cognition

Though many of the sources of idea generation can be found in the various kinds of knowledge absorbed, SNSE's still need *generative capacity*, the ability to generate ideas: "when a customer gives us a concept or an idea or we read about a competitor or we see where the market's going we still need to apply that to our use case scenario. There is an element of creativity and cognition required, how will it fit, how will it work, what benefits is it going to bring? … there isn't really an engineering discipline for that. …..it's very, very difficult (i17)[17]. The software developers we met preferred very low-tech tools when working with ideas, feeling that having their hands on a keyboard, or the detailed formality of a programming environment often impeded creativity. Many companies documented ideas and *ideation* (working with ideas) was almost invariably a team process - "it's always been a peer process….. a collaboration…….the ideas you have in that [development] situation are usually some quite small ideas to do with the implementation…..you could have a big idea in that situation that says, "oh wait a second…we're doing this completely wrong…..we ought to scratch this and do it a completely different way" ….that could obviously lead to more discussion before you actually change track…..small, small clever ideas that add up to a good innovative product[18]. The more of those you can do as you're going along, the better the software in the end" (i12)[19]. Much ideation was informal and low key ("If you have a cool idea and you are a techie you have to get one of the business development guys to okay it" (i24)). Some medium-sized companies had more formal processes: "we're big Wiki users…..we've got some conventions that we use within that….we've got cover sheets for projects, but also ideas can just start as people dumping into a page or pulling together links from pre-existing pages and so we've categories that we call futures where we've got pages and pages of ideas that have been built up…. we've got Bugzilla which is bug tracking ….. that also is a repository of a lot of ideas. Between the Wiki and Bugzilla is where people dump stuff. When a release comes to be thought about, there's a top-down seed where the senior management have a strategy or strategic objectives ….then we start trying to find ideas with …prospects in that domain….we try to synthesize those into early wish lists …. then individual winning items might get treed out a bit more" (i16)[20]. Creative cognition applies just as much to process innovation as to product/service innovation; many senior developers are very aware of their development frameworks and have "periods of very intense productivity when you're tinkering with the process …..you're hoping that it'll be an investment that pays back…… you go into continuous process improvement" (i2) [21].

Software design capability

Though generative capacity and ideation deliver ideas, in the case of new product innovation those ideas must be refined through *software design capability* - "an effective software design idea that yields itself to delivering well designed, well maintainable software over a long period" (i8). Such a design idea can be described as a product *concept* - "we're currently working on a product ….which is essentially a sequencing engine for

[17] see P8, P5
[18] see P7, P10
[19] see P7
[20] see P6, P8, P13, P15
[21] see P9

[existing product]. The idea behind the [new product] is to simplify the sequencing of operations in [existing product]. The idea is slightly deeper than that in that it allows you to…..queue up tasks and perform them in a sequence….. there's a decision-making tree within that sequence ……it's not a question of replacing functionality that's already in there, it's simply a way of simplifying the mundane tasks" (i7)[22]. Those concepts may respond to market and technology trajectory understandings: "if we are looking at having high power set-top boxes then the application environment …. is going to be more complex and that is going to take some years ……[we're] trying to evolve a new roadmap for the products where our current partners are able to adapt their architecture to make use of the power we have…." (i14). They may respond to user domain understandings: you sell people what they want but you give them what they need. So that the underlying thing [product} may not have its features exposed, but when they come back to you and say - well, I want to do this - you say - we'll turn it on for you - and it's there." (i6)[23]. Software concepts are often built up of *feature sets*, and another way of incrementally innovating is to add features to existing products: "you can use social media in business and it's probably where everything is going to go ……your LinkedIn profile definitely will get linked in [to our product].... because it then keeps it [personal data] updated" (i1).

Teamwork

Some interviewees described *teamwork* as an vital component of their innovation work: "teamwork is the most important thing with a company our size because most of our costs are the people themselves and most of the results come out of the effort of the people themselves……it is critical that they work together in a team" (i21). Developing and maintaining *shared understanding* was important for a common sense or purpose and direction: "I would say the most important aspect are be very aware of what the whole focus and direction and structure of what the team is doing is, so everybody knows what everybody's doing. The reason …. is that what you're doing will not suddenly rear off at a tangent" (i21)[24]. S*hared understanding* was also important for work coordination: "we would work with a small team that was sufficient that we could have a shared mindset and not have to have too many formal procedures, but do the daily scrum kind of thing - what do we all know to do today - what are you doing - what am I doing?" and rapidly iterate towards a solution" (i15), "focusing more on the communication and long-term understanding of where we are going through our business. Then people know what we are doing and why and the problem then is just getting the buy-in from people that what we are doing is actually the right thing to be doing" (i8). A leadership skill is getting the right people in the team - *team composition*: "I look for people who are technically excellent and enthusiastic about the technology - in other words, they actually find the subject of what they're working on interesting and fun, not just a job" (i21)[25]. A further important aspect is *expertise integration*: "[my partner's] got an amazing knowledge of the specifications of the end product and of the cryptography…. I understand the cryptography and the hardware, and that makes for …overlapping content-specific, but covering the spectrum of our target audience" [i9].

Innovation tools and techniques

Innovation tools and techniques (as developed and recommended in the academic literature) were not much in evidence in the companies we studied, though one had recently purchased innovation portal software for idea management: "we're opening that up to our internal staff to be able to put in ideas …… and they get this type of social media discussion going about ….. these ideas" (i4)[26]. There was scattered use of techniques such as mind mapping and brainstorming. However the many tools and techniques of innovation consultants didn't seem to impress the software engineers, even where they were familiar with them, although they were rather good at adapting their engineering tools (issue trackers, bug trackers, online Scrum tools, and cloud collaboration tools

[22] see P10
[23] see P11
[24] see P6
[25] see P6,
[26] see P15

such as wikis and Google documents) for the purposes of idea documentation and ideation, as documented earlier[27].

Development framework

The *development framework* refers to the set of conceptual and software tools, techniques and processes used in the innovative development context. The most important of these, according to the participants, are *experimentation* carried out through various forms of *prototyping*: "a lot of the stuff we've been doing is…. showing people onto the website and watching how they use it and talking to them about why they're shopping ….. the other half …. is doing paper prototyping and wire frame prototyping and seeing how they react" (i2)[28]; "sketches are quite important….it can be low tech; it can be a simplified demonstration of the algorithm …….. play with the algorithm see what it can achieve. ……. it can just be a set of sample screens …… and let people play with [them] to see whether they interact the way they want to" (i5)[29]. In software development this kind of experimentation is an advanced form of ideation through design experiments – the tools that are used to design the product are also used to experiment with competing designs, and elicit design feedback[30]. The companies in the study either worked explicitly with an agile method, or worked informally in a process that resembled it to some degree (an exception is that they sometimes made reasonably detailed specifications where the customer expected it). Incremental/iterative development with a degree of experimentation is supported by *agility:* "one of the guys working for us had been using agile development in the videogames industry. He was able to get everyone on board for a process change here…… people bought into the idea to different degrees. We managed to hammer out of that a working system based around the ideas of Scrum and agile development using user stories and sprint iterations……. customized for us ….. and now no one can believe that we ever did any other way" (i8)[31]. Agile methods often support ways of developing shared understanding, especially about the design concepts which: "using Scrum obviously makes people stand up every day in front of the whiteboard and talk about what they are doing, there is a level of knowledge transfer" (i14)[32]. An important innovation support is the *installed base* – here referring to the development technical platform. It has already been observed that developers use these tools for helping store and improve ideas. Installed base helps generative capacity by lifting the level of programming abstraction: "I don't know if you've used Visual Studio …… five years ago, we were just writing C++ code in text editors. … you constantly having to track what file, where you go to find what. … you don't have to think about that anymore…...your mind is free to think about the object map of your code rather than how it's stored in files. … people who code think - I'm quite capable of managing all that in my head, at the same time as coming up with innovative ideas - if your mind is free from that, then there's the whole portion of your brain that is free to think" (i12)[33]. Modern tools also facilitate experimentation: "one of the supporting pillars is your source code control system….we've moved away from a centralized one to a distributed one ……say we're looking at an individual feature, maybe speculative, maybe experimental or maybe just destabilizing and we don't want it published back to the central place too soon ….. one spinoff benefits is that effectively you're always working in a branch…… a developer can break up the work into ten chunks and they're all unpublished or private until they say I'm done ….. then you're not forcing the developer to publish intermediate pieces of work……. a really commonplace tool changes the way you work" (i16)[34].

[27] see P9, P13, P15
[28] see P4
[29] see P13
[30] see P14
[31] see P9, P13
[32] see P12, P14
[33] see P9
[34] see P13, P14

5. ANALYSIS 2: ASSOCIATIONS BETWEEN INNOVATION DRIVERS AND OUTPUTS IN THE SILICON FEN

Since drivers without sufficient empirical grounding are removed from the study, those that remain are, by definition, associated with one or both of the two innovation outputs: product/service innovation and process innovation. The purpose of this section is to unpack the more significant associations, which are later captured as theoretical propositions in section 6.

Product/service innovation associations

Since companies' future revenues are dependent on new (or updated) products and services, product/service innovation serves as the end-goal for many different chains of reasoning (some of which are illustrated below). Here is a typical example: "So the idea is that, working with a greater range of clients, that we'll start to generate more ideas, which will feed back into software products, which we can then commercialize to generate more sales, to create a virtuous circle, which will help this company grow" (i11). In the concept language of this study, *user involvement* leads to *generative capacity,* which leads to *product/service innovation*; when institutionalised this is a minor *process innovation* (the virtuous circle).

Process innovation associations

The study observes a recursive relationship between process innovation and product/service innovation. Since process innovation in itself generates no new revenues in the software industry, it's often undertaken to support the capacity to innovate with products and services. However process innovation may itself be provoked by new product development requirements. "We brought out our new generation of high definition boxes……that was an opportunity for us to get rid of a lot of legacy code that is hard to maintain, and it was an opportunity to review the infrastructure that we had and that we used to manage our software development …… …we adopted agile development …. so we have been using Scrum for some time….we had a historic source control system and build system that were a little inefficient….. so one of the things that we did as a part of our new product development was to transition new development over to Git [source control software]. …….with that infrastructure, that thing speeds up the process and the quality, so we get something out to market faster" (i14). Here *product/service innovation* sparks a change to the *development framework* (*agility, installed base*), which constitutes a *process innovation*, which has as its primary purpose speeding up *product/service innovation.*

Innovation leadership and innovation evaluation associations

Although leaders have a hand in ideation, the SMSE leaders interviewed in this study seldom saw themselves as the source of inspiration, but as the moderator of creativity in their teams. Therefore the most significant association for *leadership* is *teamwork*. Leaders are responsible for path creation for their teams: "a lot of what we do is decided with conversations……..getting the right people in the room to thrash out what we are going to do and having just enough presentation material and input to planning process that the people who are going to focus more on that area are going to be able to pick that up and run with it" (i14). They are also responsible for maintaining a creative work environment, for significant evaluative actions that primarily affect their teams (such as project selection and process improvement decisions) and for generating finance for innovative projects.

Knowledge leverage associations

Three important associations for knowledge leverage are observed in the empirical material. Firstly, knowledge leverage forms the background for *innovation leadership*, particularly path creation and evaluation. The managers were good boundary spanners, and adept at absorbing a wider variety of external information. An understanding of technology trajectories was particularly important for making product innovation choices. Secondly knowledge leverage was a prerequisite for *creative cognition*, both for *generative capacity* (often in a combinatorial style) and for *ideation* (developing the ideas): "it's based on years of mathematical, scientific knowledge and experience…. and sometimes it is the application of an idea from one field into another one"

(i15). Lastly knowledge leverage (particularly *market understandings* and *technology trajectories*) aids *software design capability*, both in terms of identifying new product *concepts* and *features*: "what's innovative about this software is more that it's in a different space…it's a space that's quite new…. emerging…the world of transfer switches has been mechanical …..something that uses software to make intelligent decisions is a big change." (i7).

Community and network associations
Community and network provides an important source for knowledge which can be trusted, so its principal association is with *knowledge leverage*: "we have an idea as to where we want to go……. but then we want to find out what everybody else is doing, where everybody else gone to, let's learn from one another……people are surprisingly open….. sharing" (i25)

User involvement associations
User involvement is associated (obviously) with better understanding of *user domains*, but also with better *creative cognition*, in that conversations with users spark many ideas, and with productive *experimentation* and *prototyping*, since users prove highly relevant feedback. In combination these also produce improved *software design capability.*

Creative cognition associations
Creative cognition (producing and working with ideas) lies at the root of all innovation. In the study, *generative capacity* and *ideation* showed a recursive association - they work together to strengthen creative cognition. *Generative capacity* was associated with *process innovation* and strongly associated with *product/service innovation*. Software engineers think in terms of software designs: "we started, just trying to think, probably about three or four of us initially ….. coming up with a frame, an architecture and an overall design that would meet the requirements we'd been given, but would also meet all the other requirements that we knew would exist at some point from our experience" (i21), so we record a primary association between creative cognition and *software design capability*. However creative cognition is also important for improving the *development framework*: "we build tools for our own staff to use ….our staff in London use customized tools we build here" (i2) and thus for *process innovation*.

Software design capability associations
Software design capability showed, as might be expected, a strong empirical relationship with *product/service innovation*: "the building blocks of what we do are algorithms that …. are mainly released under open source licenses……what we do is take these components and put those together, in new ways to build workflows that support particular data processes and tasks" (i22)

Teamwork associations
The most developed teamwork association was in supporting *creative cognition*: "each month we have an activity called the den, in which a number of people will evaluate the ideas …..if somebody's put in an idea….people will then iterate around that idea and it could change direction or get optimized in a particular way" (i4).

Development framework and innovation tools and techniques associations
A more complex set of associations surround the development framework. The configuration of the development framework affects *creative cognition*, for example *experimentation* and *prototyping* support idea development, especially if user involvement is possible, and tools from the *installed base* are often used to record ideas – this is the closest that SMSEs normally come to dedicated innovation tools and techniques. *Agility* supports *teamwork,* which in turn supports *creative cognition*. A well-configured development framework is essential to *software design capability*, and most *software process innovations* are made up of modifications to the development framework.

6. SOFTWARE INNOVATION IN SMSEs: THEORY DEVELOPMENT

Overview concepts and primary associations

Figure 2 gives a simplified overview of significant innovation drivers and their relationship to outputs, which is derived from the literature study, and supported by the interview findings.

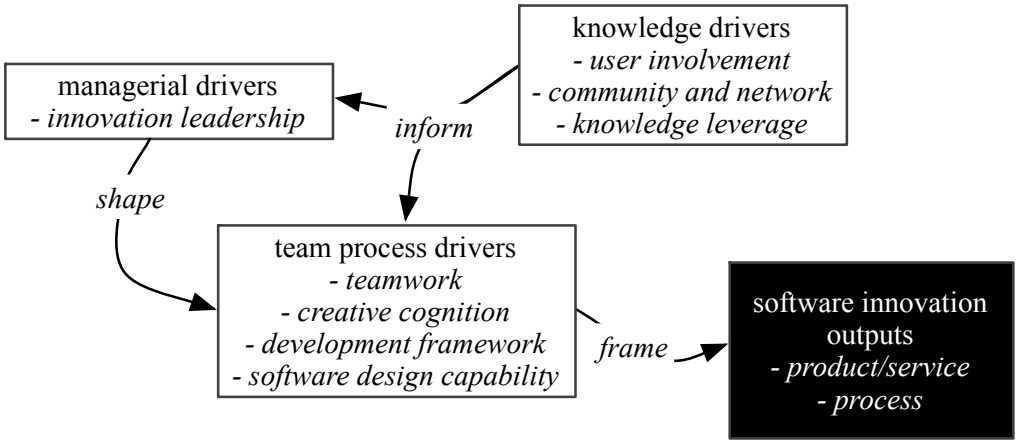


Figure 2. Software innovation in SMSEs: drivers and outputs

The significant knowledge drivers for Silicon Fen SMSEs are user involvement, community and network and knowledge leverage. Users and community are important sources of knowledge, and knowledge leverage represents the process of exploiting knowledge for innovation. Knowledge informs both management and team process. The primary management driver is innovation leadership, which is important in shaping the team process. Significant team process drivers are teamwork, creative cognition, the development framework, and software design capability. The team process is important in the framing both of innovative software products and services, and innovations in its own software development process. The primary associations between drivers and outputs can be describes as follows:

- Knowledge drivers inform (frame innovation sense making, provide evidential support for decision making and creative inspiration) innovation management and the team process.
- Managerial drivers shape (provide overall guidance and direction for) the innovative team process
- Team process drivers frame (provide the work environment and cognitive framing for) software innovation outputs.

Detailed model and exploratory theoretical propositions

In this section the overview model in Figure 2 is decomposed using the results of the qualitative analysis. Figure 3 shows all the significant concepts in their study with exploratory relationships drawn from the two analyses. Concept and coding frequency analysis (analysis 1) determines the selection of concepts for the model. Coding co-occurrence analysis (analysis 2) underpins the building of relational propositions.

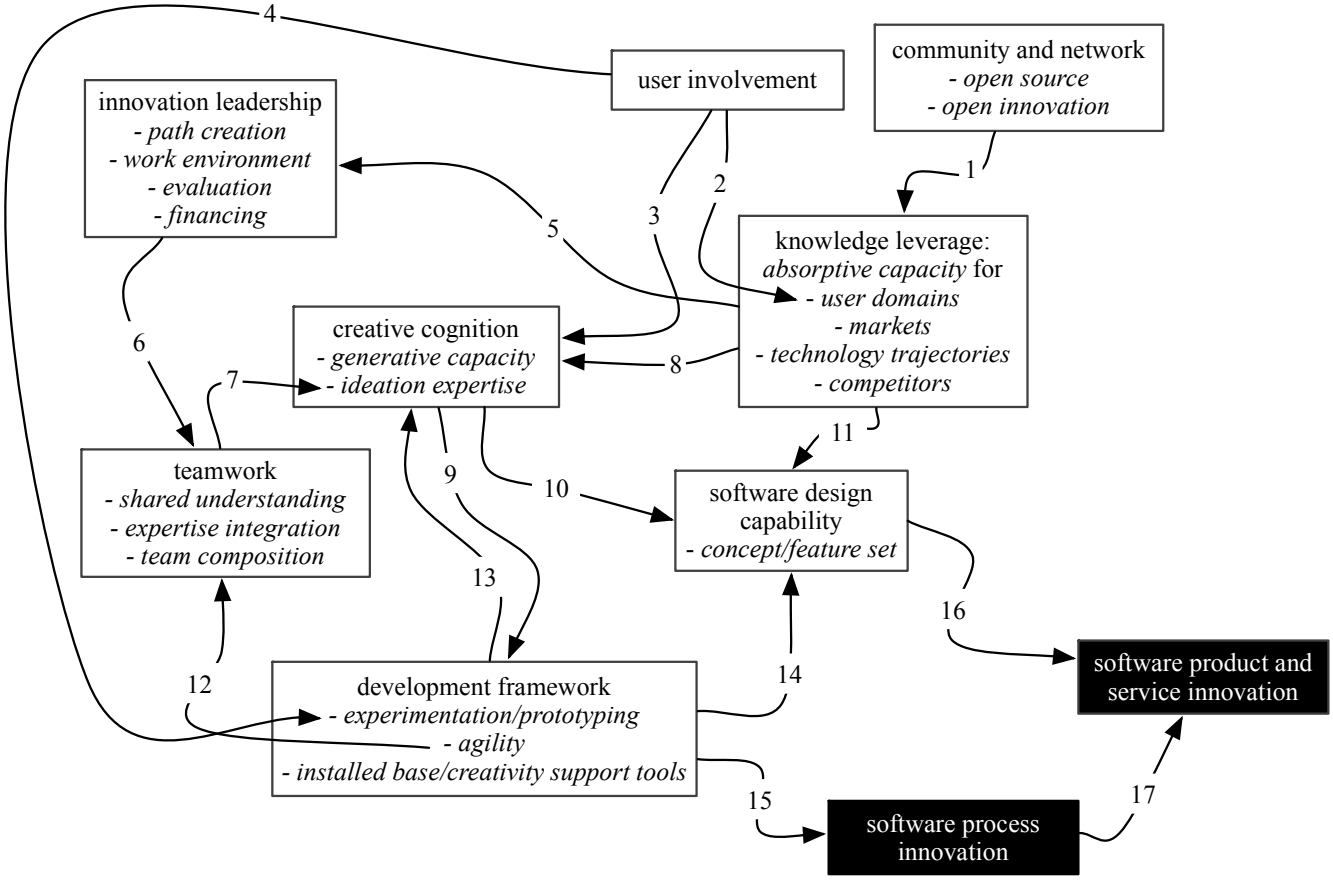


Figure 3. Detailed model of software innovation in SMSEs indicating relational propositions

Table 2 recasts derived relationships between concepts as an exploratory proposition set for the model, with explanations of the mechanisms underpinning the propositions and showing their analytical derivation.

	Proposition	Mechanisms
P1	Interactions with community and network improve knowledge leverage	Open source communities provide code which facilitates software design capability, open innovation stimulates creative cognition and innovative business models
P2	User involvement improves user domain understanding	Knowledge transfer
P3	User involvement improves creative cognition	Users produce good ideas of their own (generative capacity) and improve the quality of ideation
P4	User involvement improves the quality of experimentation/prototyping	Users provide commercially-oriented feedback that is otherwise difficult for developers to reproduce
P5	Knowledge leverage improves innovation leadership	Provides the foundation for path creation and evaluation

P6	Innovation leadership improves teamwork	Path creation and evaluation feed into the shared understandings and expertise integration of the team, contribute to its work environment and is the primary inspiration for team composition
P7	Teamwork improves ideation expertise	Incorporates complementary expertise and develops shared understandings
P8	Knowledge leverage increases the quality of creative cognition	Provides the inspiration for idea generation and the evaluative context for ideation
P9	Creative cognition improves the development framework	Provides effective ideas for orienting the development framework towards the promotion of innovation
P10	Creative cognition increases software design capability	Improving generative capacity and ideation provide more innovative product designs
P11	Knowledge leverage promotes software design capability	Provides the evidential and experiential base for product design decisions
P12	Agility supports teamwork	Supports shared understanding and expertise integration
P13	The development framework can be organised to support creative cognition	Experimentation/prototyping supports ideation expertise, agility encourages iteration for prototyping (as well as teamwork and interaction with users), the installed base can be used as creativity tools to support ideation
P14	A development framework including experimentation/prototyping, agility and installed base used to support ideation increases innovative software design capability	These features combine to stimulate generative capacity and productive ideation in software design
P15	Targeted improvements in the development framework improve the capacity of the software process to support innovation	Agility, experimentation/prototyping (including low-tech prototyping) and installed base supporting ideation and experimentation constitute process innovations which promote innovation
P16	Innovation-directed capability in software design supports software product and service innovation	The ability to generate novel and useful software concepts and feature sets drives product/service innovation
P17	Targeted software process innovations increase software product and service innovation	Innovation-directed process improvements (such as, but not limited to, those mentioned in p14) can improve product and service innovation.

Table 2. Integrated proposition set for software innovation in SMES

7. DISCUSSION

The empirical evidence supports the importance of knowledge leverage for innovation in SMES's, and offers qualified support for the importance of community and network and user involvement, especially as knowledge

sources. Qualified support is also offered for the role of creative cognition, software design capability, teamwork and the development framework. Innovation tools and techniques are not much used in our sample of SMSEs (though we met one company that had recently installed a proprietary idea management system) and neither was formal innovation evaluation. Some codes (customization, portfolio management, creativity techniques, user-driven/lead user, user toolkits, conflict resolution, crowd sourcing, innovation tools & techniques, creative requirements analysis) were little used and consequently dropped from the models. The study shows minimal adoption of creativity techniques proposed by Couger et al. (1993) by SMSEs, perhaps because of a lack of fit with engineering cultures; in any case our engineers preferred low-tech prototyping for design ideation. There was no use of the workshop-based creative requirements gathering proposed by Maiden et al. (2004) and others, and no real use of crowd-sourcing. The only kinds of user toolkits (Franke & Von Hippel, 2003) in use were application programming interfaces (APIs) to open platforms. One new code emerged as significant through open coding: finance (finding economic resources to support innovative projects or changes).

The study supports the work of Carlo et al. (2011) in highlighting the role of knowledge leverage through absorptive capacity, extending it to incremental innovation in SMSEs and categorising important knowledge areas. These support the importance of market and technology knowledge (Brem & Voigt, 2009) for ideation, and add some complimentary knowledge sources, particular user domains. The study reaffirms the contribution of users and customers to software innovation (they provide both knowledge and ideas (Marcel Bogers et al., 2010)) and suggests some mechanisms (absorptive capacity, ideation, prototyping) for how that contribution is transformed into software features and products. There is evidence of the emergence of varied open innovation strategies (Chesbrough, 2003) amongst SMSEs, though the majority of companies retain closed strategies and intellectual property protection for the code they write themselves. Companies have relationships with open source communities in several ways (Lee & Cole, 2003) and this promotes innovation (Von Hippel & Von Krogh, 2003); however the mechanisms amongst SMSEs are more to do with innovation speed (through reusing existing code) and alternative (open) business models than they are to do with community learning. The findings are in agreement with Zmud (1983) that external information is important for innovation, and that internal teamwork affects its utilisation, however we offer several mechanisms (experimentation, prototyping, ideation) which explain how this happens. We focus on some different aspects of teamwork than Hoegl and Gemuenden (2001), including expertise integration (Tiwana & McLean, 2005). The study supports the relationship between teamwork and creative cognition theorised by Cooper (2000), adding an ideation perspective as the principal operational mechanism. In common with Koc (2007) we focus on idea generation, and human resource (in SMES's this is organised through team composition) and cross-functional integration (in SMES's usually taking the more limited form of expertise integration within the team). We adapt the concept of generative capacity (Avital & Te'eni, 2009a) for use in software development (which is useful as an antidote to a prevailing belief that system requirements come perfectly formed from customers and users), by suggesting many moderating factors and mechanisms. We could also have used their idea of generative fit to investigate how well developers' installed base promotes their creativity. The findings offer some empirical evidence of the role of agile methods in innovation, as claimed by Highsmith and Cockburn (2001) and suggest some mechanisms for its operation (supporting teamwork, shared understanding, underpinning experimentation through iteration). As Carlo et al. (2011) argue, experimentation, primarily in the form of low-tech prototyping appears to be a key element of innovation ideation. Finally, like Gumusluog and Ilsev (2009) we propose a role for innovation leadership in SMSEs; however more in terms of path creation and providing a creative work environment than in redefining process.

The fragmented literature on software innovation that forms the starting point for this study explicitly accommodates neither the innovation characteristics of small and medium enterprises (such as behavioural advantages, informality, openness, etc.) nor the special characteristics of software (malleability, low market entry costs, negligible reproduction costs). These were the motivating reasons for the integration study. In the

light of the study's findings we may therefore identify the distinctive features of software innovation in SMSEs as follows:

- They are more likely to focus on incremental innovation achieved by adapting, refining and combining their own (often technical) software ideas with external ideas in a niche market
- They are thus dependent on much external knowledge possessed by customers, users, and their technical communities – but they have informal processes and strategies for acquiring this knowledge
- They tend to encourage openness (sometimes through the open source movement) and various forms of cooperation both to improve the acquisition of relevant knowledge and to bootstrap the innovation process by providing extra resources (for example software libraries for open source projects)
- They rarely have specialist skills or processes for innovation but integrate innovation into the daily work of software development in a seamless way
- The management of innovation is accomplished in a hands-on, informal way – managers are often deeply involved in software development as designers, developers and software architects and guide their teams through mutual engagement rather than deliberate strategies, hierarchical authority or formal evaluation
- The team process exploits the malleability of software – it is usually flexible, iterative in nature (though without necessarily formally adopting an agile method), and exploits the power of prototyping for developing software. It is usually focused on design rather than extensive analysis.
- Ideation is often conducted away from the programming interface - in notes, sketches, conversations and low tech prototypes
- SMSEs rarely engage with the innovation industry (whether innovation consultancy or academic theories), but adapt their own engineering tools to support their innovation

8. CONCLUSIONS

SMSEs are significant contributors to national economies, and dependent on their capacity to innovate to survive and grow. The innovation literature is extensive, and several researchers have investigated aspects of innovation processes in the context of software development, leaving a fragmented literature in need of integration. We focused on the smaller software firm, synthesized a set of drivers and outputs for software innovation from literature that deals directly with software companies, and investigated it empirically through a study located in the Silicon Fen. We found that the most significant innovation drivers were knowledge, innovation management and the team process. Few of the innovation industry's practices or the tools and techniques recommended in the academic literature had found their way into practice in smaller companies. However companies are adept at leveraging knowledge from their surroundings, adapting the engineering tools that they work with to support iterative ideation and experimentation in teams, and transferring those ideas into code and accompanying business models. These abilities are complemented by a focus on improving their development process – also a form of innovation. Patterns revealed in the dataset underpin overview and detailed models of SMSE innovation, and the accompanying set of exploratory propositions. The principal contributions of the article, therefore, are to provide an integrated account of organisational drivers for software innovation in smaller companies, and a set of propositions with both theoretical and empirical grounding. These contributions can serve to focus future research initiatives in an evolving research area. Further contributions add to developing understandings in the supporting literature in the ways described in the discussion section.

Some significant limitations of this exploratory study should also be noted. The internal validity of the study may be limited by the sample size and selection method, and more extensive data collection is required. The study reflects the perceptions of appropriate research subjects with dual strategic and development roles, and a broader selection of respondents might influence its outcomes. The study's external validity and generalizability may be affected by particular characteristics of high tech clusters such as the Silicon Fen. This means that the importance of the knowledge and networking factors in SMSEs' innovation noted in this study may not be

generalizable to companies working outside the high tech cluster environments. Although the study exposes generalities in the way Silicon Fen SMSEs innovate, individual companies obviously differ considerably; an interesting research question for future research is whether different types of companies (for instance pure consultancy companies, and own-product companies) deploy different combinations of drivers to marshal innovation. The exploratory nature of the study also limits its application by practitioners; however a good descriptive theory of how SMSEs organise innovation (to which this study contributes) is clearly a precursor for normative theories, and a pre-requisite for sound prescriptive advice for developers and managers.

ACKNOWLEDGEMENTS

This research work is supported by the Danish Research Council: grant no. 12-133180. Many thanks to the interviewees and the participating companies.

REFERENCES

Aaen, I. (2008). Essence: facilitating software innovation. *European Journal of Information Systems*, *17*(5), 543–553.

Aaen, I., & Rose, J. (2011). A Software Entrepreneurship Course Between Two Paradigms. In *15th Annual Interdisciplinary Entrepreneurship Conference*. St. Gallen and Zurich.

Adams, M. E., Day, G. S., & Dougherty, D. (1998). Enhancing new product development performance: An organizational learning perspective. *Journal of Product Innovation Management*, *15*(5), 403–422.

Aerts, A. T. M., Goossenaerts, J. B. M., Hammer, D. K., & Wortmann, J. C. (2004). Architectures in context: on the evolution of business, application software, and ICT platform architectures. *Information & Management*, *41*(6), 781–794.

Amoroso, D. L., & Couger, J. D. (1995). Developing Information Systems with Creativity Techniques: An Exploratory Study. In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences* (pp. 720–728).

April, A., & Busse, D. K. (2007). Fast-tracking Product Innovation. In *CHI '07 Extended Abstracts on Human Factors in Computing* (pp. 1703–1708).

Athaide, G. A., Meyers, P. W., & Wilemon, D. L. (1996). Seller-buyer interactions during the commercialization of technological process innovations. *Journal of Product Innovation Management*, *13*(5), 406–421.

Avital, M., & Te'eni, D. (2009). From generative fit to generative capacity: exploring an emerging dimension of information systems design and task performance. *Information Systems Journal*, *19*(4), 345–367.

Berelson, B. (1952). *Content analysis in communicative research*. New York: Free Press.

Bogers, M., Afuah, a., & Bastian, B. (2010). Users as Innovators: A Review, Critique, and Future Research Directions. *Journal of Management*, *36*(4), 857–875.

Boland Jr., R. J., Lyytinen, K., & Yoo, Y. (2007). Wakes of innovation in project networks: the case of digital 3-D representations in architecture, engineering, and construction. *Organization Science*, *18*(4), 631–647.

Brem, A., & Voigt, K.-I. (2009). Integration of market pull and technology push in the corporate front end and innovation management-Insights from the German software industry. *Technovation*, *29*(5), 351–367.

Briggs, R. O., & Reinig, B. A. (2010). Bounded Ideation Theory. *Journal of Management Information Systems*, *27*(1), 123–144.

Carayannis, E., & Coleman, J. (2005). Creative system design methodologies: the case of complex technical systems. *Technovation*, *25*(8), 831–840.

Carlo, J., Lyytinen, K., & Rose, G. (2011). A knowledge-based model of radical innovation in small software firms. *MIS Quarterly*, *36*(3), 865–895.

Chesbrough, H. W. (2003). *Open innovation: The new imperative for creating and profiting from technology*. Boston, MA.: Harvard Business School Publishing.

Compeau, D. R., Meister, D. B., & Higgins, C. A. (2007). From prediction to explanation: Reconceptualizing and extending the Perceived Characteristics of Innovating. *Journal of the Association for Information Systems*, *8*(8), 409–439.

Cooper, R. B. (2000). Information technology development creativity: A case study of attempted radical change. *MIS Quarterly*, *24*(2), 245–276.

Cooper, R. G. (2011). Perspective: The Innovation Dilemma: How to Innovate When the Market Is Mature. *Journal of Product Innovation Management*, *28*, 2–27.

Couger, J., Higgins, L., & McIntyre, S. C. (1993). (Un)structured creativity in information systems organizations. *MIS Quarterly*, *17*(4), 375–397.

De Jong, J. P. J., & von Hippel, E. (2009). Transfers of user process innovations to process equipment producers: A study of Dutch high-tech firms. *Research Policy*, *38*(7), 1181–1191.

Doel, C. (2011). *Cambridge Cluster at 50; The Cambridge economy, retrospect and prospect*. Cambridge. doi:http://www.stjohns.co.uk/wp-content/uploads/2011/04/Cambridge-cluster-report-FINAL-210311.pdf

Florida, R., & Goodnight, J. (2005). Managing for creativity. *Harvard Business Review*, *83*(7), 124–131.

Franke, N., & Von Hippel, E. (2003). Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software. *Research Policy*, *32*(7), 1199–1215.

Gassmann, O., Sandmeier, P., & Wecht, C. H. (2006). Extreme customer innovation in the front-end: learning from a new software paradigm. *International Journal of Technology Management*, *33*(1), 46–66.

Gray, P. H., Parise, S., & Iyer, B. (2011). Innovation Impacts of Using Social Bookmarking Systems. *MIS Quarterly*, *35*(3), 629–643.

Gumusluog, L., & Ilsev, A. (2009). Transformational Leadership and Organizational Innovation: The Roles of Internal and External Support for Innovation. *Journal of Product Innovation Management*, *26*(3), 264–277.

Hanninen, S. (2007). The "perfect technology syndrome": sources, consequences and solutions. *International Journal of Technology Management*, *39*(1-2), 20–32.

Heirman, A., & Clarysse, B. (2007). Which tangible and intangible assets matter for innovation speed in start-ups? *Journal of Product Innovation Management*, *24*(4), 303–315.

Henkel, J. (2006). Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy*, *35*(7), 953–969

Hesmer, A., Hribernik, K. A., Hauge, J. M. B., & Thoben, K. D. (2011). Supporting the ideation processes by a collaborative online based toolset. *International Journal of Technology Management*, *55*(3-4), 218–225.

Higgins, L. F. (1996). A Comparison of Scales for Assessing Personal Creativity in IS. In *Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences* (pp. 13–19). Hawaii: IEEE.

Highsmith, J., & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer*, *34*(9), 120–127.

Hocova, P., E Cunha, J. F., & Staníček, Z. (2009). Design and management of an innovative software enterprise: A case study of a spin-off from University. In D. F. Kocaoglu, T. R. Anderson, T. U. Daim, A. Jetter, & C. M. Weber (Eds.), *PICMET 2009. Portland International Conference on Management of Engineering & Technology* (pp. 2704–2713).

Hoegl, M., & Gemuenden, H. (2001). Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization Science*, *12*(4), 435–449.

Hoegl, M., & Proserpio, L. (2004). Team member proximity and teamwork in innovative projects. *Research Policy*, *33*(8), 1153–1165.

Holmquist, L. (2004). User-driven innovation in the future applications lab. In *CHI EA '04 CHI '04 Extended Abstracts on Human Factors in Computing Systems* (pp. 1091–1092).

Hung, S.-C., & Whittington, R. (2011). Agency in national innovation systems: Institutional entrepreneurship and the professionalization of Taiwanese IT. *Research Policy*, *40*(4), 526–538.

Igira, F. T. (2008). The situatedness of work practices and organizational culture: implications for information systems innovation uptake. *Journal of Information Technology*, *23*(2), 79–88.

Jansen, H. (2010). The Logic of Qualitative Survey Research and Its Position in the Field of Social Research Methods. *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, *11*(2).

Jenkins, A. M. (1985). Research Methodologies and MIS Research. In E. Mumford (Ed.), *Research Methods in Information Systems*. Amsterdam, Holland: Elsevier Science Publishers B.V.

Koc, T. (2007). Organizational determinants of innovation capacity in software companies. *Computers & Industrial Engineering*, *53*(3), 373–385.

Koepp, R. (2003). *Clusters of creativity: enduring lessons on innovation and entrepreneurship from Silicon Valley and Europe's Silicon Fen*. Chichester: John Wiley & Sons.

Krippendorff, K. H. (2004). *Content Analysis: An Introduction to Its Methodology*. Thousand Oaks, CA: Sage Publications Ltd.

Kristensson, P., Magnusson, P. R., & Matthing, J. (2002). Users as a Hidden Resource for Creativity: Findings from an Experimental Study on User Involvement. *Creativity and Innovation Management*, *11*(1), 55–61.

Lamastra, C. R. (2009). Software innovativeness. A comparison between proprietary and Free/Open Source solutions offered by Italian SMEs. *R&D Management*, *39*(2), 153–169.

Lee, G. K., & Cole, R. E. (2003). From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development. *Organization Science*, *14*(6), 633–649.

Leimeister, J. M., Huber, M., Bretschneider, U., & Krcmar, H. (2009). Leveraging Crowdsourcing: Activation-Supporting Components for IT-Based Ideas Competition. *Journal of Management Information Systems*, *26*(1), 197–224.

Leonardi, P. M. (2011). Innovation Blindness: Culture, Frames, and Cross-Boundary Problem Construction in the Development of New Technology Concepts. *Organization Science*, *22*(2), 347–369.

Lippoldt, D. & Stryszowskli, P. (2009). *Innovation in the Software Sector*. OECD.

Lobert, B., & Dologite, D. G. (1994). Measuring creativity of information system ideas: an exploratory investigation. In *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences* (pp. 392–402).

Love, J. H., & Roper, S. (2015). SME innovation, exporting and growth: A review of existing evidence. *International Small Business Journal*, *33*(1), 28–48.

Lu, I.-Y., & Wang, C.-H. (2007). Technology innovation and knowledge management in the high-tech industry. *International Journal of Technology Management*, *39*(1-2), 3–19.

Lyytinen, K., & Rose, G. (2003). The disruptive nature of information technology innovations: the case of internet computing in systems development organizations. *MIS Quarterly*, *27*(4), 557–596.

Maccrimmon, K. R., & Wagner, C. (1994). Stimulating ideas through creativity software. *Management Science*, *40*(11), 1514–1532.

Maiden, N., Gizikis, A., & Robertson, S. (2004). Provoking creativity: Imagine what your requirements could be like. *Ieee Software*, *21*(5), 68–75.

Maiden, N., Manning, S., Robertson, S., & Greenwood, J. (2004). Integrating creativity workshops into structured requirements processes. In *DIS '04 Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques* (pp. 113–122).

Martin, R. L. (2011). The Innovation Catalysts. *Harvard Business Review*, *89*(6), 82–87.

Massetti, B. (1996). An empirical examination of the value of creativity support systems on idea generation. *MIS Quarterly*, *20*(1), 83–97.

McLean, E. R., & Smits, S. J. (1993). The I/S leader as `innovator'. In *Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences* (pp. 352–358).

Mich, L., Berry, D. M., & Anesi, C., (2005). Applying a pragmatics-based creativity-fostering technique to requirements elicitation. *Requirements Engineering*, *10*(4), 262–275.

Morrison, P. D., Roberts, J. H., & von Hippel, E. (2000). Determinants of User Innovation and Innovation Sharing in a Local Market. *Management Science*, *46*(12), 1513–1527.

Müller, S. D., & Ulrich, F. (2012). Creativity and Information Systems in a Hypercompetitive Environment : A Literature Review. *Communications of the Association for Information Systems*, 32, 175-201.

Nambisan, S., Agarwal, R., & Tanniru, M. (1999). Organizational mechanisms for enhancing user innovation in information technology. *MIS Quarterly*, *23*(3), 365–395.
Napier, N. P., Mathiassen, L., & Robey, D. (2011). Building contextual ambidexterity in a software company to improve firm-level coordination. *European Journal of Information Systems*, *20*(6), 674–690.
Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization Science*, *5*, 14–37.
Oakey, R. (2012). *High-Technology Entrepreneurship*. London: Routledge.
Ojala, A., & Tyrväinen, P. (2006). Business models and market entry mode choice of small software firms. *Journal of International Entrepreneurship*, *4*(2), 69–81.
Oliveira, P., & Von Hippel, E. (2011). Users as service innovators: The case of banking services. *Research Policy*, *40*(6), 806–818.
Pikkarainen, M., Codenie, W., Boucart, N., & Alvaro, J. A. H. (eds). (2011). *The Art of Software Innovation: Eight Practice Areas to Inspire your Business*. Berlin, Heidelberg: Springer.
Pisano, G. P., & Verganti, R. (2008). Which Kind of Collaboration Is Right for You? *Harvard Business Review*, *86*(12), 76–8.
Quintas, P. (1994). A product-process model of innovation in software-development. *Journal of Information Technology*, *9*(1), 3–17.
Raasch, C. (2011). The sticks and carrots of integrating users into product development. *International Journal of Technology Management*, *56*(1), 21–39.
Raffa, M., & Zollo, G. (1994). SOURCES OF INNOVATION AND PROFESSIONALS IN SMALL INNOVATIVE FIRMS. *International Journal of Technology Management*, *9*(3-4), 481–496.
Roberts, E. B. (1988). Managing invention and innovation. *Research Technology Management*, *31*(1), 11–27.
Romijn, H., & Albaladejo, M. (2002). Determinants of innovation capability in small electronics and software firms in southeast England. *Research Policy*, *31*(7), 1053–1067.
Rose, J. (2010). *Software Innovation - Eight work-style heuristics for creative system developers*. Aalborg University, Aalborg: Software Innovation.
Santanen, E. L., Briggs, R. O., & De Vreede, G. J. (2004). Causal relationships in creative problem solving: Comparing facilitation interventions for ideation. *Journal of Management Information Systems*, *20*(4), 167–197.
Sas, C., & Zhang, C. (2010). Investigating emotions in creative design. In *DESIRE '10: Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design* (pp. 138–149).
Sherif, K., Zmud, R. W., & Browne, G. J. (2006). Managing peer-to-peer conflicts in disruptive information technology innovations: The case of software reuse. *MIS Quarterly*, *30*(2), 339–356.
Shneiderman, B. (2007). Creativity support tools: Accelerating discovery and innovation. *Communications of the ACM*, *50*(12), 20–32.
Shneiderman, B. (2000). Creating Creativity: User Interfaces for Supporting Innovation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *7*(1), 114–138.
Silverman, D. (2001). *Interpreting qualitative data*. London: SAGE Publications Ltd.
Snow, C. C., Fjeldstad, O. D., Lettl, C., & Miles, R. E. (2011). Organizing Continuous Product Development and Commercialization: The Collaborative Community of Firms Model. *Journal of Product Innovation Management*, *28*(1), 3–16.
Sosa, M. E. (2011). Where Do Creative Interactions Come From? The Role of Tie Content and Social Networks. *Organization Science*, *22*(1), 1–21.
Supyuenyong, V., Islam, N., & Kulkarni, U. (2009). Influence of SME characteristics on knowledge management processes. *Journal of Enterprise Information Management*, *22*(1/2), 63–80.
Thomke, S. (2001). Enlightened experimentation - The new imperative for innovation. *Harvard Business Review*, *79*(2), 66–75.
Tiwana, A., & McLean, E. R. (2005). Expertise integration and creativity in information systems development. *Journal of Management Information Systems*, *22*(1), 13–43.

Tjornehoj, G., & Mathiassen, L. (2010). Improvisation during process-technology adoption: a longitudinal study of a software firm. *Journal of Information Technology*, *25*(1), 20–34.
Trott, P. (1998). *Innovation management and new product development*. Harlow: Pearson.
Turner, S. F., Mitchell, W., & Bettis, R. A. (2010). Responding to Rivals and Complements: How Market Concentration Shapes Generational Product Innovation Strategy. *Organization Science*, *21*(4), 854–872. doi:10.1287/orsc.1090.0486
Van den Ende, J., & Wijnberg, N. (2003). The organization of innovation and market dynamics: Managing increasing returns in software firms. *IEEE Transactions on Engineering Management*, *50*(3), 374–382.
Von Hippel, E., & Von Krogh, G. (2003). Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science*, *14*(2), 209–224.
Von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, *32*(7), 1217–1241.
West, J., & Gallagher, S. (2006). Challenges of open innovation: the paradox of firm investment in open-source software. *R & D Management*, *36*(3), 319–331.
Weterings, A., & Boschma, R. (2009). Does spatial proximity to customers matter for innovative performance? Evidence from the Dutch software sector. *Research Policy*, *38*(5), 746–755.
Weterings, A., & Koster, S. (2007). Inheriting knowledge and sustaining relationships: What stimulates the innovative performance of small software firms in the Netherlands? *Research Policy*, *36*(3), 320–335.
Wilkinson, S. (1997). Focus group research. In D. Silverman (Ed.), *Qualitative research: Theory, method and practice*. London: Sage Publications Ltd.
Wong, K. Y., & Aspinwall, E. (2004). Characterizing knowledge management in the small business environment. *Journal of Knowledge Management*, *8*(3), 44–61.
Yang, H.-L., & Hsiao, S.-L. (2009). Mechanisms of developing innovative IT-enabled services: A case study of Taiwanese healthcare service. *Technovation*, *29*(5), 327–337.
Zmud, R. (1983). The effectiveness of external information channels in facilitating innovation within software development groups. *MIS Quarterly*, *7*(2), 43–59.

Appendix 1. List of participating companies and interviewees

Amino Communications	http://www.aminocom.com/	Paul Fellows, Gareth Crocker
ARK CLS Ltd	http://www.arkcls.com/	Adrian Bennetton, Tony Benn, John Watson
ArtVPS Ltd	http://www.artvps.com/	Matthew Bentham
Cambridge Cognition	http://www.camcog.com	Ricky Dolphin
Digital Locksmiths Ltd.	http://www.digitallocksmiths.com/index.html	Sean Kelly
Eagle Genomics Ltd.	http://www.eaglegenomics.com/	William Spooner, Glenn Proctor
LeoTel Software Systems Limited	http://www.leotel-software.co.uk	James Bridson, Justine Jackson, Jonathan Reichert
Linguamatics Ltd	http://www.linguamatics.com/index.html	Jason Trenouth
McMillan Technology	www.mcmillantech.co.uk	John McMillan
Metail Ltd	http://www.metail.com/	Jim Downing
NationSoft	http://nationsoft.co.uk/	Ivar Jenssen
OpenDCU	http://opendcu.org	Kim Spence-Jones
PARIS Transport Management Solutions	http://www.paris-tms.com/home.htm	Darren Shaw
Plextek Ltd	http://www.plextek.com/	Jon Lewis
Product Technology Partners Ltd	http://www.ptpart.co.uk/	Kevin Snelling
Sentec Ltd	http://www.sentec.co.uk/	Katie Smith
Speedwell	http://www.speedwell.co.uk/	David Yeneralski
Synthetix Ltd	http://www.synthetix.com/index.php	Peter McKean
TriSys Business Software	http://www.trisys.co.uk/	Garry Lowther

Appendix 2. List of in-depth interviews

No.	Date	Role
1	20/03/2013	Chief Executive Officer
2	21/03/2013	Chief Technical Officer
3	26/03/2013	Technical Director
4	27/03/2013	Chief Innovation Officer
5	02/04/2013	Senior Consultant
6	04/04/2013	Programme Director
7	05/04/2013	Director
8	08/04/2013	Operations Director
9	17/04/2013	Chief Executive Officer
10	19/04/2013	Systems Manager
11	22/04/2013	senior developer
12	22/04/2013	senior developer
13	22/04/2013	Managing Director
14	30/04/2013	Head of Product Development
15	30/04/2013	Chief Technical Officer
16	07/05/2013	Chief Technical Officer
17	22/05/2013	Chief Executive Officer
18	23/05/2013	Managing Director
19	31/05/2013	senior developer
20	31/05/2013	senior developer
21	31/05/2013	Director and Technical Lead
22	14/06/2013	Chief Technical Officer
23	14/06/2013	senior developer
24	17/06/2013	Head of Software
25	02/07/2013	Chief Technical Officer

Appendix 3. Interview protocol

Interview protocol

Introduce self

Explain:

- Purpose and organization of project, expectations and benefits, eventual outcomes
- Interview will be recorded and later transcribed, material may be used in research articles but will not be attributed to individuals or companies without your permission.
- Semi-structured interview around a pre-determined list of topics, room for interpretation and divergence, not an administered questionnaire, length

Ask:

interviewee's name, contact details and role, general details about the software operation: what they build and how

Explain:

types of software innovation: process, product/service

Ask:

Can you offer some examples of innovation that you've been involved with? Tell us a story, develop a narrative.

Question areas (as presented to interviewees)**:**

- **Knowledge Leverage**: gaining and exploiting knowledge about technologies, markets competitors, and users and integrating and deploying that knowledge in development projects
- **Development Framework:** the governing frameworks for ways of working with innovative development projects for example with methods, agility, creative requirements gathering or prototyping
- **Supporting Tools & Techniques:** aimed at underpinning innovation and creativity – creativity techniques, software tools, user toolkits
- **Creative cognition**: the psychology of individual creativity in software design and how it is enhanced – idea generation and selection
- **Teamwork:** the structure and performance of an innovative team – how it is supported and developed
- **Community and Network:** links with outsides collaborators and partners, open innovation, working with open source and crowd sourcing
- **Innovation Leadership**: creating and sustain and innovation climate, choosing directions and focus and managing portfolios of projects, conflict resolution
- **Software Design Capability**: developing concepts and feature sets for new products
- **User Involvement:** involvement of users in design and development, customization, user-driven innovation
- **Infrastructure/Installed Base:** influence of already installed software and hardware, existing ecosystem
- **Innovation Evaluation:** assessing innovation and creativity

Explore other important innovation factors not already discussed.

Appendix 4. Coding steps for reliability

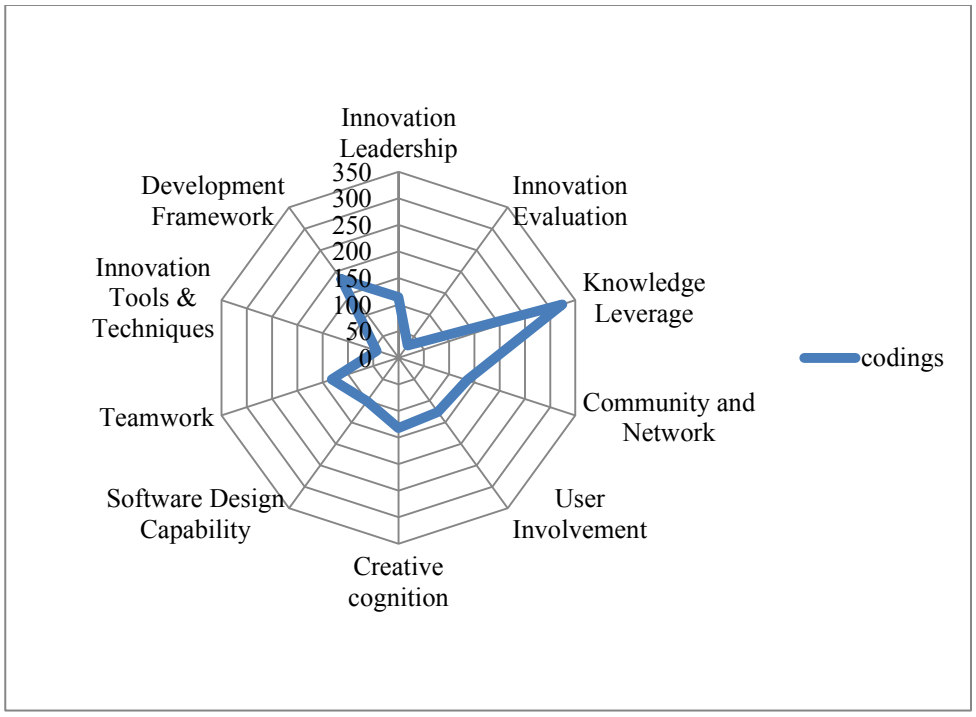
Table 3: Coding steps			
Step	**Activity**	**Actor(s)**	**Outcomes**
Coding scheme development	Concepts from the theoretical model articulated as Dedoose codes and sub-codes	Coders 1,2,3,	Coding scheme
Pilot coding	Three interviews from the same company coded from the initial codes and sub-codes	Coders 1,2,3, working synchronously and discussing evolving scheme over Skype	Coding scheme revised
Evaluation of the pilot study	The pilot study was evaluated to ensure coding reliability	Coders 1, 2	Refined coding scheme and interview coding
Coding of the remaining dataset	Remaining interviews coded using the coding scheme	Coders 1,2,4	All interviews coded
Evaluation of the coded dataset	Complete coding evaluated	Coders 2,4	Reliably coded dataset

Appendix 5. Coding frequency (>20 codings) organized in order of coding frequency

User Involvement	101
User Domain Understanding	78
Community and Network	75
Market Understanding	72
Technology Trajectory Understanding	66
Installed base	64
Experimentation/Prototyping	60
Generative Capacity	58
Ideation Expertise	44
Feature Set	44
Absorptive Capacity	42
Shared Understanding	41
Knowledge Leverage	40
finance	37
Innovation Leadership	36
Open Source	36
Expertise Integration	36
Concept	33
Path Creation	32
Teamwork	32
Development Framework	32
Creative cognition	30
Innovation Evaluation	29
Work Environment	28
Agility	27
Competitor Understanding	26
Creativity Support Tools	25
Software Design Capability	24
Team Composition	23
Open Innovation	22

italics = code introduced during open coding

Appendix 6. Distribution of codings for top-level codes



Appendix 7. Significant code co-occurrences (n>20) signalling possible associations between concepts

code	related code	no. of co-occurrences	explanation	related propositions	evidence sources
Product/Service Innovation	User Domain Understanding	44	User Domain Understanding improves software design capability (14 co-occurrences), leads to P/S innovation	P11, P16	i24, i22, i21, i17, i18, i11, i16, i5, i8, i2, i11
Product/Service Innovation	User Involvement	34	User involvement promotes user domain understanding, stimulates creative cognition (23 co-occurrences), increases learning through prototyping (18 co-occurrences), thus leading to improved software design capability (11 co-occurrences), and leads to P/S innovation	P2, P3, P4, P16	i24, i11, i23, i22, i21, i17, i18, i5, i9, i7, i1, i3, i10, i11, i12
Product/Service Innovation	Market Understanding	32	Market understanding improves software design capability (9 co-occurrences) leads to P/S innovation	P11, P16	i24, i17, i18, i16, i5, i7, i4, i13, i11
Product/Service Innovation	Concept	30	Software design capability (concept/feature) improves P/S innovation	P16	i24, i22, i21, i18, i5, i14, i9, i7, i1, i3, i4, i11, i8, i12, i13
Product/Service Innovation	Feature Set	28	Software design capability (concept/feature) improves P/S innovation	P16	i24, i22, i21, i18, i5, i14, i9, i7, i1, i3, i4, i11, i8, i12, i13
Product/Service Innovation	Technology Trajectory Understanding	28	Technology Trajectory Understanding, mediated, improves software design capability (11 co-occurrences), leads to P/S innovation	P11, P16	i24, i22, i21, i17, i16, i1, i3, i4, i11, i13
Process Innovation	Product/Service Innovation	24	Process innovation is often a driver for product and service innovation, occasionally necessary as a result of them	P17	i11, i6, i23, i22, i16, i14, i9, i3, i2, i4, i11
Product/Service Innovation	Experimentation/Prototyping	30	Experimentation/Prototyping improves ideation expertise (11 co-occurrences) and software design capability, leading to P/S innovation	P13, P16	i24, i22, i20, i17, i18, i16, i5, i1, i3, i2, i4

Product/Service Innovation	Generative Capacity	23	Generative capacity improves software design capability and thus P/S innovation	P10, P16	i11, i21, i16, i15, i5, i14, i6, i10, i11, i13,
Generative Capacity	Ideation Expertise	22	Generative capacity is supported by the ideation expertise through teamwork (17 co-occurrences) in a recursive process which improves the quality of creative cognition	P7	i21, i16, i5, i8, i3, i10, i4, i12