Empirical Software Engineering manuscript No.
(will be inserted by the editor)

How Product Owner Teams Scale Agile Methods to Large Distributed Enterprises

Julian M. Bass

Received: date / Accepted: date

Abstract Software development teams in large scale offshore enterprise development programmes are often under intense pressure to deliver high quality software within challenging time contraints. Project failures can attract adverse publicity and damage corporate reputations. Agile methods have been advocated to reduce project risks, improving both productivity and product quality. This article uses practitioner descriptions of agile method tailoring to explore large scale offshore enterprise development programmes with a focus on product owner role tailoring, where the product owner identifies and prioritises customer requirements. In globalised projects, the product owner must reconcile competing business interests, whilst generating and then prioritising large numbers of requirements for numerous development teams. The study comprises eight international companies, based in London, Bangalore and Delhi. Interviews with 46 practitioners were conducted between February 2010 and May 2012. Grounded theory was used to identify that product owners form into teams. The main contribution of this research is to describe the nine product owner team functions identified: groom, prioritiser, release master, technical architect, governor, communicator, traveller, intermediary and risk assessor. These product owner functions arbitrate between conflicting customer requirements, approve release schedules, disseminate architectural design decisions, provide technical governance and propogate information across teams. The functions identified in this research are mapped to a scrum of scrums process, and a taxonomy of the functions shows how focusing on either decision-making or information dissemination in each helps to tailor agile methods to large scale offshore enterprise development programmes.

Keywords agile software development · scrum · large scale offshore enterprise development programmes · product owner · product owner teams · grounded theory.

Julian M. Bass
School of Computing Science and Digital Media, Robert Gordon University, Garthdee Road, Aberdeen, AB10 7GJ, UK
Tel.: +44 1224 262732
E-mail: j.m.bass@rgu.ac.uk

1 Introduction

Problems in large scale offshore enterprise software development programmes can damage the reputation of well-known companies and even challenge the ability of governments to implement policy. For example, Santander UK plc had problems with an IT integration project following the take-over of the smaller Alliance & Leicester bank (Computer Weekly, 2012). Whilst Santander had previously maintained a positive IT track record during earlier take-overs, the adverse publicity that this incident attracted extended beyond technology news outlets (BBC, 2012).

Government decisions to change policies in relation to taxation or social security systems often necessitate large scale software implementations, an ability that has often been questioned. In the UK Parliament, concern has been expressed regarding the government's ability to implement a new social security payment system. In September 2012, Liam Byrne MP, during questions to the Treasury said "the Universal Credit is late and over budget; there is widespread unease surrounding the implementation of the £2 billion [US $ 3.2 billion approx.] scheme's IT system" (Hansard, 2012). In September 2013, a report by the National Audit Office revealed an IT budget increase of 60%, as well as delays in the roll-out of the proposed Universal Credit system. The National Audit Office also reported that the Department for Work and Pensions had written off £34 million [US $ 54.3 million approx] (Glick, 2013). It is therefore apparent that project teams on large scale offshore enterprise development programmes are under considerable pressure to deliver successful outcomes.

There is increasing practitioner interest in tailoring agile software methods to large scale offshore enterprise development programmes (Leffingwell, 2007; Larman and Vodde, 2008; Ambler and Lines, 2012). According to practitioners, the three key agile principles are: (1) achievement of customer satisfaction through early and continuous delivery of valuable software; (2) business and development team members working together frequently throughout the project; and (3) face-to-face conversations as the most efficient way to convey information to, and within, the development team (de Cesare et al, 2010).

These practitioner perceptions are in unanimity with proponents of agile methods, who argue that they improve team morale, resulting in enhanced productivity and improved responsiveness to customer needs, culminating in better software quality (Agile Alliance, 2011). Empirical research suggests that agile methods do indeed improve job satisfaction, productivity and customer satisfaction (Dyba and Dingsoyr, 2009), with other research reporting improved communication between team members, quick releases and the increased flexibility of agile designs (Begel and Nagappan, 2007). Critical success factors for agile projects have been identified as: correct delivery strategy, proper practice of agile methods and high calibre team members (Chow and Cao, 2008). However, there can be challenges with adoption for large and complex projects (Dyba and Dingsoyr, 2009).

This research contributes to the literature on tailoring agile methods for use in large scale offshore enterprise development programmes , including five CMMI maturity level 5 accredited offshore software development vendors. The research question for this study is "how do practitioners describe the tailoring of agile method roles and practices in large scale offshore enterprise software development programmes?" In particular, this article focuses on the research question "how do practitioners describe enhancement and expansion of functions within the product owner role, to meet the needs of large scale offshore enterprise software development programmes?"

A further contribution of this research is to add to the literature on product owner teams. Product owners do not work as individual project stakeholders in the large projects investigated. Rather, they instead work together in informal (or sometimes formal) teams, and the functions of the members of these product owner teams are described in this research.

In answering the research questions, nine product owner functions have been identified, and each of these functions is described in turn. The functions have then been mapped to the scrum agile process. More specifically, the functions are mapped to the scrum of scrums process, which has previously been proposed for extending scrum to large projects (Leffingwell, 2007). In this article, a taxonomy of the functions is developed in which the functions are categorised depending upon the domain (business or technical) and nature of the function (information gathering or information dissemination).

This article will develop an argument with three main steps (1) that the scope of product owner functions can be identified in the context of a broader study on selected software development programmes, (2) that product owner teams emerge from the functions identified, and (3) that the product owner teams require members with different specialist skills. That is, product owner teams often find it difficult to operate with multi-function team members. These three arguments are briefly explored in turn.

At its core, product ownership is about communicating a business need to a development team. The development team know how to design, test and deploy systems, and the product owner knows what system needs to be built. In large scale offshore enterprise software development programmes, articulating and communicating a business need is itself often a time consuming and difficult task. This research shows that the identifying, refining and disseminating of a need is currently performed by people with a wide range of job titles, including: product manager, business analyst, project manager and programme manager. The job functions performed by product owners, proxy product owners and technical product owners have been collated. These job functions relate to identifying, refining and disseminating a need in large scale offshore enterprise software development programmes. Job functions performed by project managers (such as project administration and development environment procurement) and product managers (such as product marketing) that are not connected to communicating with the development team or refining the business need have been excluded. This job function selection was conducted using sources including the Skills Framework for the Information Age (SFIA Foundation, 2014), a taxonomy of IT industry roles and responsibilities.

The product owner team concept emerges from the product owner functions identified, due to two main influences: (1) programme scale, and (2) standardisation of best practice. Firstly, in large scale offshore enterprise software development programmes, the sponsor is usually a senior executive with high-level responsibilities. However, the findings of this research show that the sponsor cannot commit the time necessary to determine needs in sufficient detail or to communicate them to large numbers of development team members. Therefore, intermediaries are identified in order to perform the detailed functions required to articulate and prioritise the need.

Secondly, in the large scale offshore enterprise software development programmes studied, a complex range of technical and policy standards was found to exist. These standards are designed to ensure that development teams comply with established best practice. The standards fall into two main areas. Firstly, good development process practices are used (for example, to ensure compliance with CMMI accreditation requirements); and secondly, that development programmes comply with technical constraints.

This article will argue that the product owner functions reflect the division between the knowledge and skills of the business domain, and of the technical domain. There are, of course, individuals with both business domain and technical skills. However, these individuals are rare and are in high demand in the employment market. Furthermore, in large scale offshore enterprise software development programmes, the level of business domain and technical skills required is high. These are challenging development programmes with stringent deadlines and complex resource needs. This research shows that the individuals with both business domain and technical skills are very rarely able to show executive level leadership in both domains. More commonly, product owner teams are populated by team members with either business or technical domain knowledge. Consequently, it is hard to find product owners capable of tackling both technical and business functions with sufficient credibility at high levels within large organisations.

The paper is structured as follows. An overview of agile software development methods is provided, concentrating first on scrum, which was the most widely used process in the projects investigated, and then functions within the product owner role. Next follows a discussion of the research methods adopted, including the research sites, data collection and data analysis techniques used. The findings describe nine functions that comprise the product owner role in large scale offshore enterprise development programmes: groom, prioritiser, communicator, traveller, intermediary, governor, technical architect, risk assessor and release manager. These findings extend understanding of how, in practice, companies scale agile methods to large scale offshore enterprise development programmes. There is then a discussion of these findings, with implications for practitioners and for process tailoring theory. Finally, conclusions and suggestions for future work are presented.

2 Agile Methods

Agile methods were established during the 1990s, building upon evolutionary development approaches that emerged in the 1980s (Larman and Basili, 2003). There are a range of software development methods that can regarded as being agile, including Dynamic Systems Development Methods (DSDM) (Stapleton, 1997); Feature Driven Development (Coad et al, 1999); Crystal (Cockburn, 2001); Scrum (Schwaber and Beedle, 2001); Extreme Programming (XP) (Beck and Andres, 2004); and Lean Software Development (Poppendieck and Poppendieck, 2003).

XP comprises engineering practices such as test-driven development, continuous integration, refactoring and pair programming (Beck and Andres, 2004). Test-driven development advocates writing unit-tests prior to program code development (Wilkerson et al, 2012). In test-driven development, new tests are written before features are added or changed. The technique tends to be used in association with automated unit-test tools. Continuous integration is where changes are frequently (typically daily) merged into an evolving code base (Cusumano, 2007). Refactoring is used to evolve the underlying architecture and design in order to accommodate new functionality (Mens and Tourwe, 2004). Pair programming, where developers work together in a manner analogous to a pilot/co-pilot configuration, has been studied extensively (Balijepally et al, 2009; Hannay et al, 2010; Lui et al, 2008). Effective XP teams have a shared sense of responsibility, and a relaxed yet rhythmic approach to working; and good quality code is a jointly-owned and valued asset (Sharp and Robinson, 2004).

In contrast with the engineering focus of XP, scrum has tended to focus on the orchestration and management of agile development (Schwaber, 2004). Scrum, most commonly adopted by the company teams investigated in this study, proposes short, focused periods of development called sprints, which typically last between two and four weeks (França et al, 2010).

Software requirements are captured, analysed and prioritised in the form of brief textual, non-technical descriptions called user stories. These user stories are prioritised, before the start of each sprint, by a product owner who represents the strategic needs of the client. Stakeholders, including the development team members and the product owner, then work together to create work estimates for each user story, using a consensus-based scoring technique. The development team members deconstruct each user story into its constituent technical tasks, necessary for implementation at the start of each sprint.

Project team members communicate using a daily coordination meeting, the eponymous 'scrum'. The scrum is traditionally conducted standing up, in a conscious effort to minimise the duration of the meeting. Team members are required to answer three questions: (1) "what have you done since the last meeting?"; (2) "what will you do between now and the next meeting?"; and (3) "what impediments that prevent your progress have you encountered or created for others?"

Scrum teams are self-organising, since team members collaborate in order to develop work estimates, and can select user stories for implementation within the current sprint (Cohn, 2009; Hoda et al, 2010, 2012; Moe et al, 2010; Monteiro et al, 2011). Scrum emphasises incremental software development using a 'feature' team structure (Schwaber and Beedle, 2001). Feature team members tackle the holistic development of end-to-end user story functionality (Coad et al, 1999), which contrasts with traditional approaches that hierarchically distribute team members into specialist groups around architectural components, such as user interface, business logic or persistence layer elements.

Some proponents argue that agile methods must be holistically implemented in their entirety in order to achieve full benefits (Beck and Andres, 2004). However, the findings presented here suggest that this is not always possible, or even desirable, in large scale offshore enterprise development programmes.

2.1 Enterprise Agile

As already discussed, the challenges of scaling agile methods to large offshore enterprise development programmes have received attention from numerous practitioners (Leffingwell, 2007; Larman and Vodde, 2008; Ambler and Lines, 2012). The simultaneous use of agile methods and plan-based methods in large enterprises

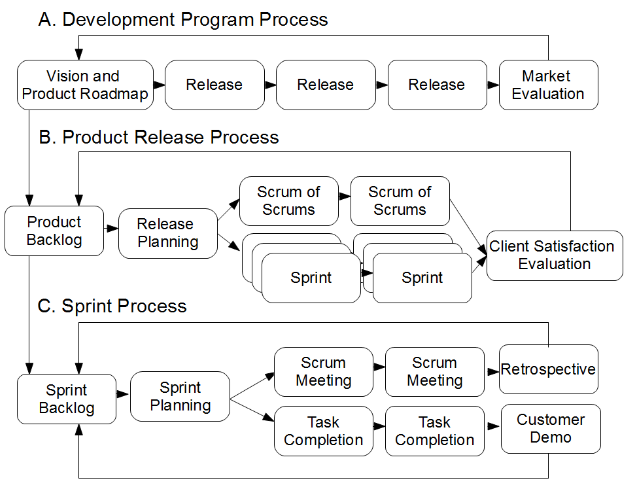

Fig. 1 Overall scrum-of-scrums process

has also received interest from researchers (van Waardenburg and van Vliet, 2013). Large team size, complex business contexts and demanding time constraints can converge to cause a range of threats to productivity in agile projects (Hannay and Benestad, 2010), as the co-existence of plan-based and agile methods increases complexity and impedes the involvement of business stakeholders (van Waardenburg and van Vliet, 2013). Large scale projects therefore require a more disciplined approach to software development (Ambler, 2008). There is also evidence that large scale projects may exacerbate communication problems (Pikkarainen et al, 2008).

A scrum of scrums approach has been advocated to accommodate large team size (Leffingwell, 2007). Several scrum teams are formed, each with a scrum master in the usual way, and each scrum team comprises 7-12 developers. Daily coordination meetings are held within each scrum team, and in addition, the scrum masters attend a coordination meeting across the teams (the scrum of scrums). The scrum of scrums is used to tactically manage and coordinate the progress of iterations through the various scrum teams.

During the scrum of scrums meeting, each scrum master will report: (1) "what my team has done since the last meeting", (2) "what my team will do between now and the next meeting" and (3) "what impediments that prevent progress my team has encountered or created for others". Scrum of scrum meetings with too many participants can lack focus and relevance (Paasivaara et al, 2012b) and communication is improved when scrum of scrum meetings are organised around common interests and participant needs.

The overall software development process is illustrated in Figure 1. The sprint process, (C) in Figure 1, comprises a requirements backlog, sprint planning, retrospective review and customer demonstration of tested code. Within the sprint, tasks identified during planning are completed and communication takes place in scrum meetings.

Sprints are nested within a product release process, (B) in Figure 1, and releases comprise one or more sprints. Large projects tend to have a larger number of sprints within a release. The release comprises a product backlog and release planning. The product backlog is refined and prioritised, based on user surveys and client feedback.

At a more strategic level, development programmes (A) in Figure 1, comprise product releases, while each product release itself is comprised of the sprints shown in Figure 1. A product roadmap is developed as part of overall product marketing and this product roadmap will define the features intended to achieve the desired market penetration within a given market segment.

In the related area of global software development, it has been shown that more time and personnel are required to undertake cross-site work of similar size and complexity when compared with same-site work (Herbsleb and Mockus, 2003); due to increased temporal, geographical and socio-cultural distances. However, the emphasis, in agile methods, on communication and building trust can help ameliorate such challenges (Ramesh et al, 2006). In addition, agile methods improve perceived product quality; although there may be challenges around misunderstood requirements and awkward communication in distributed meetings (Paasivaara et al, 2008). A meta study of research papers suggests that the most researched geographically distributed agile practices are continuous integration; stand-up meetings; pair programming; retrospectives; scrum of scrums; and test-driven development (Jalali and Wohlin, 2010).

To mitigate the challenges of geographical distribution, multiple modes of communication support are available, including telephone, web camera, teleconference, video conference, web conference, net meeting, email, shared mailing lists, instant messaging and short messaging service. A variety of collaboration techniques are available to scrum teams including visits and periods of co-located working, unofficial meetings, training functions and distributed documentation support tools to help alleviate sociocultural distance (Hossain et al, 2009).

The focus of the research presented here is on large scale offshore enterprise development programmes, which includes multiple teams working in onshore and offshore, and occasionally more complex geographically distributed configurations.

2.2 Product Ownership

Extreme programming advocates an on-site customer, a client representative that is available to the team on a full-time basis (Beck and Andres, 2004). The on-site customer role has been extensively explored by Angela Martin (for example, Martin et al, 2009a,b; Martin, 2009). A set of XP customer practices have been identified by investigating 66 practitioner participants involved in 11 projects of varying sizes (Martin et al, 2009a). The concept of an XP customer team is identified; with groups of practices that include collaboration guides, direction setting and skill specialists (Martin et al, 2009b).

Scrum defines three roles in its agile processes: the self-organising team, the scrum master and the product owner (Schwaber and Beedle, 2001). Product ownership plays a central role in the overall software development process (Raithatha, 2007), as the product owner is responsible for communication between the customer and development teams (Hoda et al, 2011). The product owner is responsi-

Table 1 Participating Companies and Project Details

Company	Company Sector	Total Number of Employees	Revenue, 2013 (Gross Income)
A	IT Service Provider	171,000	US $8.8 billion
B	Internet	12,500	US $4.7 billion
C	Software Service Provider	2,500	Not Available
D	Software Service Provider (Offshore Provider to Company E)	7,000	US $550 million (2009)
E	Enterprise CRM	2,500 (Post Takeover)	US $500 million (Post Takeover)
F	Industrial Products	362,000	€75 billion
G	IT Service Provider	131,000	€10 billion
H	IT Service Provider	8,000	US $380 million

ble for developing and maintaining the product backlog, the list of user stories that define the requirements for the project. However, a critique of product owners suggests that they are not always knowledgeable about best practice in requirements engineering (Ktata and Lévesque, 2009).

Two approaches to establishing product owner teams have been identified: area product owner and proxy product owner (Paasivaara et al, 2012a). Area product owners are each responsible for a subset of product features and report to an overall product owner, whilst proxy product owners operate a shared responsibility model. Each of these approaches has been seen to offer both strengths and weaknesses (Paasivaara et al, 2012a).

3 Method

This qualitative study of software engineering practice comprised eight international companies and interviews with 45 practitioners, as shown in Table 1.

3.1 Research Sites

The companies were selected from a population of enterprises engaged in large scale offshore software development programmes . Five of the companies are CMMI accredited at maturity level 5. The companies chosen had head offices in Germany, India and USA. The turnover of the two largest companies is almost €8 billion and over US $1.5 billion. The interviews were conducted in Bangalore, India (January 2010 and April 2011); London, UK (February 2010); and Delhi, India (May 2012), as shown in Table 2.

The companies investigated were involved in either off-shoring (companies B and F), or out-sourcing (companies A, C, D, E, G and H). Project details, including team size and product owner location are shown in Table 3. Three of the projects are geographically distributed (company A, CRM insurance; company G, healthcare; and company F, healthcare). The remaining 15 projects have onshore clients and offshore teams in various configurations. Off-shoring is typically motivated by a desire to access and cultivate worldwide talent pools, and both off-shoring and outsourcing are intended to offer lower cost resource pools, when compared

Table 2 Participating Companies and Interviewees

Company	Interview Locations and Dates	Interviewee Job Titles
A	Bangalore January 2010	Programme Manager Senior Project Manager Team Member
B	Bangalore January 2010 April 2011	Engineering Manager Product Manager
C	Bangalore January 2010	Development Manager
D	Bangalore January 2010	Project Manager Product Owner Scrum Master (3) QA Lead Team Member
E	London February 2010	Programme Manager Project Manager Director of Engineering
F	Bangalore April 2011	Scrum Master
G	Bangalore April 2011	Engagement Manager
H	Delhi May 2012	Chief Technology Officer Corporate Lead Architect General Manager H. R. Delivery/Programme Manager (3) Senior/Project Manager (3) Scrum Master (2) Technical Analyst/Consultant/Specialist (6) Team Member (9) Business Analyst

with in-house onshore resources. For example, Company B is a household name in the Internet business sector, and retains an onshore development capability in California, but has also built up an in-house off-shore development team in India (as well as other territories), in order to reduce costs, while attracting a range of specialist skills. Company F, with broad interests in the industrial products space, has headquarters in Europe but also has research and development centres in India and elsewhere. Work is allocated according to the concentration of expertise into specialist groups within the enterprise (to avoid duplication of competencies throughout the organisation). The IT services companies (companies A, C, D, G and H) are well-known vendors in the world-wide outsourcing sector.

Selection of the companies and research study participants was through a snowball sampling technique (for example see Patton, 2002; Miles and Huberman, 1994). Professional contacts provided access to the first study participants. Those participants were then able to provide access to other development teams and companies. Early phases of the study focused on participant breadth, at Companies A, B, C, F and G, gaining access to a range of project teams and stakeholders with different perspectives. Participants ranged from Company C, one of whose defining characteristics is their adherence to agile methods, to agile sceptics, with negative experiences to report, found at Companies E and G. The later phases of the study focused on depth, by targeting participants with a range of stakeholder roles within

Table 3 Project, Team Size, Team Deployment and Project Type

Company	Project	Team Size	Team Deployment	Project Type
A	CRM (Insurance)	325	17 Countries	Bespoke
	CRM (Banking)	50	PO onshore, offshore team	Bespoke
	CRM (Healthcare)	75	PO onshore, offshore team	Bespoke
B	Internet (Calendar)	25	offshore	SPL [a]
	Internet (Mail)	25	offshore	SPL
	Internet (Options)	25	offshore	SPL
C	Transport (Rail Ticketing)	40	3 offshore locations	Bespoke
D/E	Marketing (Campaign Management)	25	PO onshore, offshore team	SPL
	Enterprise CRM (Core)	20	PO onshore, offshore team	SPL
	Enterprise CRM (Banking)	12	PO onshore, offshore team	Bespoke
	Enterprise CRM (Credit Card)	20	PO onshore, onshore team	SPL
	Enterprise CRM (Financial Services)	25	PO onshore, offshore team	Bespoke
F	Healthcare (Instruments)	1000	various locations	SPL
	Industrial Automation	200	PO onshore, offshore team	SPL
G	Media Entertainment	50	PO onshore, offshore team	Bespoke
	Healthcare	180	various locations	SPL
H	Travel (Loyalty)	30	3 teams	SPL
	Travel (Airline Reservation)	25	1 team onshore and 2 teams offshore	Bespoke
	Risk Management and Insurance	30	1 team onshore and 2 teams offshore	SPL

[a] SPL, software product line

the same company or project. Here the interviews at Company H, and Companies D and E provided developer, QA, project management and corporate-level perspectives. This in-depth phase of the study is an implementation of intensity sampling (Patton, 2002, pp. 171). Selecting research participants that provide a wide range of perspectives, using snowball sampling in an early phase and then using intensity sampling in a subsequent phase, is a combination sampling approach, which provides an element of methodological triangulation to the sample. Combination participant sampling provides insights into both the current status of the research problem and the motivation that underlies such practices. The motivations for the use of practices are difficult to obtain using large scale survey methods.

3.2 Data Collection

A range of documentary sources were used to inform the study. Many of the companies investigated produce corporate guidelines on software development processes, and several of these commercially confidential guidelines were studied. These guidelines outline corporate agile practices, roles, policies and recommendations. However, it was not possible to obtain permission to access such guidelines from some of the companies participating in the research, as IT and software services companies regard these guidelines as highly commercially sensitive.

Table 4 Product Owner Team Deployment

Company	Project	Product Owner Location	Development Process
A	CRM (Insurance)	Malaysia	RUP [a] Distributed BA Team
	CRM (Banking)	UK	Scrum
	CRM (Healthcare)	USA	Scrum
B	Internet (Calendar)	India	Scrum
	Internet (Mail)	USA	Scrum
	Internet (Options)	USA	Scrum
C	Transport (Rail Ticketing)	UK	XP Onsite Customer
D/E	Marketing (Campaign Management)	India	Scrum
	Enterprise CRM (Core)	USA	Scrum
	Enterprise CRM (Banking)	New Zealand	Scrum
	Enterprise CRM (Credit Card)	UK	FDD [b] (Capsule Development)
	Enterprise CRM (Financial Services)	UK	RUP
F	Healthcare (Instruments)	USA	Scrum
	Industrial Automation	USA	Scrum
G	Media Entertainment	USA	Scrum
	Healthcare	USA	Scrum
H	Travel (Loyalty)	UK	Scrum
	Travel (Airline Reservation)	UK	Scrum
	Risk Management and Insurance	USA	Scrum

[a] RUP, Rational Unified Process
[b] FDD, feature driven development

Some project documentation has also been investigated, including design and architecture documents. Marketing materials such as publicly available white papers, technical reports, case studies and descriptions of vendor capabilities designed for potential customers, were also reviewed.

Site visits enabled observations of working areas and working practices, and secure development team work environments were also visited. Coordination meetings (stand-up meetings) were observed in real-time for both co-located and distributed scrum teams. The work environment arrangements for distributed scrum coordination meetings using both video- and audio-conferencing technologies were also investigated. A range of informal, sometimes off-site, discussions with executives, project management and development team members were conducted.

Face-to-face recorded interviews were conducted with 46 practitioner interviewees (Hove and Anda, 2005; Seaman, 1999) and these recordings were transcribed. An open-ended semi-structured interview approach was adopted, as open-ended interviews give respondents the opportunity to raise any topics, issues and concerns outside the scope of scripted interview questions. The semi-structured approach adopted in this study contrasts with an interview survey approach, where the same carefully scripted questions are repeated with each participant. Further, probing questions were used to elicit more detail about topics raised by interviewees. This approach generates detailed descriptions from each participant, and enables integration of the varying interviewee perspectives (Weiss, 1994).

An interview guide was used (see Appendix 1), which evolved as the study focused on scaling agile methods to large offshore enterprise software development programmes (see Appendix 2). Interviews were typically conducted on company premises, using small meeting rooms exclusively booked for the purpose.

Research interviews can be viewed from three perspectives, (1) research craft, (2) social production of knowledge and (3) social practice (Kvale and Brinkmann, 2009). When viewed as a craft, interviewing is seen as a set of practical skills and personal judgements to be acquired by the researcher through practice. From this perspective the interview is not rooted in any carefully prescribed methodological process, and interview quality is judged by the depth and value of the knowledge produced.

In contrast, interviewing can be seen as an active conversational process between interviewer and interviewee. This conversational process is contextual, linguistic and narrative. From this perspective, interviewers do not uncover facts buried in the interview, so much as participate in order to generate knowledge.

Finally, interviews can be viewed as social practice, embedded in a historical and social context. The interview is laden with ethical issues and potential social impacts after publication, and there is perceived to be an inherent power asymmetry within the interview situation. Interviews can therefore provide evidence-based findings of practice that shape concepts of human behaviour. This research adopted a social practice conception of interviews. Well executed interviewing can provide “compelling descriptions on the human world” and “provide us with well-founded knowledge” (Kvale and Brinkmann, 2009, pp.47).

3.3 Data Analysis

Initially, both the audio interviews and associated written transcripts were carefully reviewed to ensure consistency. The transcript text was imported into a qualitative data analysis software tool, Nvivo V9 (NVivo, 2013).

The grounded theory analysis began with the identification of concepts within the data. Grounded theory is an analytical process for discovering theory through the analysis of data (Glaser and Strauss, 1999). The new theories arise from the data being analysed and are thus grounded in that data. The interview concepts were coded and then compared within and between interviewees (Gibbs, 2007). These interview concepts were then iteratively grouped and refined into selected categories, as shown in Figure 2. Figure 2 illustrates how concepts were combined to create categories, which were then themselves coded, listed and compared within and between transcripts from different interviewees. This iterative concept categorisation was used to organise the large volume of data into a typology. The categories become saturated as data collection progresses (Glaser, 1992), and the categorisation forms the basis of the grounded theory (Glaser, 1998).

Categories that emerge from the data using the using the terminology of the area under exploration are known as ‘in vivo’ categories (Glaser, 1992). Such categories are also sometimes known as emergent categories, since they emerge from the data without the prior awareness of the researcher (Patton, 2002). These in

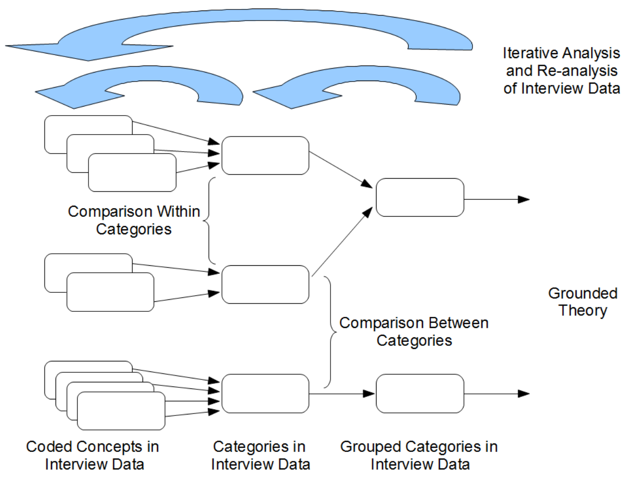
Iterative Analysis and Re-analysis of Interview Data
Comparison Within Categories
Grounded Theory
Comparison Between Categories
Coded Concepts in Interview Data
Categories in Interview Data
Grouped Categories in Interview Data

Fig. 2 Iterative analysis leading to grounded theory.

vivo categories are particularly valued in grounded theory, as they are, by definition, free from any prior assumptions and prejudices held by the researcher.

In grounded theory, researchers make several visits to the field. Data collection is performed iteratively, with analysis conducted between these data collection field visits. Data collection continues until categories become saturated. Subsequent data collection brings diminishing returns in terms of new insights into categories under consideration. Thus, in grounded theory, the researcher iterates back and forth between data collection and analysis. This approach is "close to the common-sense approach which one might use when trying to understand something which is complex and puzzling"(Robson, 2011, pp. 148).

3.3.1 Field Notes and Memo Writing

During data collection, a series of field notes were produced. These informal notes recorded interesting issues arising during the interviews, such as apparent contradictions, areas of possible uncertainty and striking examples of emergent topics. These field notes were extended during the data analysis with descriptions of selected categories. These notes describing categories are examples of memo writing during which categories are identified, refined and sharpened (Glaser, 1998, Chapter 12). The memos were used to keep track of the emerging theory; and they evolved and changed during the analysis as new transcript data was added.

3.3.2 Open Coding

Open coding was conducted on a sentence-by-sentence basis of the interview transcripts. Short descriptive phrases were used as codes, such as "technical product owner", "product grooming" and "user story triage". During the early stages of analysis the codes were hand-written on to hard copies of the transcripts. This approach provided a quick and easy way to identify and collate the initial codes. The codes at this stage were tentative and evolved quickly. Subsequently, as the volume of interview transcript data increased, the coding process was formalised and the data analysis software tool, Nvivo, was employed (NVivo, 2013).

3.3.3 Constant Comparison

Constant comparison was used to refine and sharpen the categories emerging from data in this research. Glaser and Strauss (1999, pp.105) identify four main purposes of constant comparison: comparing incidents that apply to each category, integrating categories and their properties, delimiting the theory, and writing the theory. The codes from each interview were compared with each other at two key levels: firstly, within the same organisation or project team; and secondly, with outside organisations and teams. The codes were honed over time using constant comparison. For example, "requirements gathering" was a coding category early in the analysis, but this was later refined into the two codes "product grooming" and "user story triage". Similarly, the code "technical product owner" was refined into "reference architect" and "governance coordinator" as more detailed transcript data emerged from the intensity sampling stage, as mentioned in Section 3.1.

In summary, data analysis emerged from a process of iteration involving memo writing, open coding and constant comparison. Early topics, identified using line-by-line analysis of the transcript data, were recorded in memos. These topics were subsequently refined and sharpened through constant comparison within and between interview transcripts. As the volume of interview data increased the topics became discrete categories. The categories form the basis of the grounded theory which is described next.

4 Findings

Early proponents of agile methods advocated the use of small, self-organising and co-located teams. However, in enterprise settings, large work volumes, short deadlines and entrenched organisational structures often result in tailored agile approaches. The contribution of this paper is to provide practitioner descriptions of product owner functions. The research identifies the emergence of new functions within product owner teams that allow agile methods to scale-up to large offshore software enterprise development programmes . Nine functions within the product owner role are identified.

First, the overall choice of software development process is considered. In general, plan-based (or waterfall) methods can be contrasted with agile methods. Plan-based methods are described by an experienced senior executive, as follows:

> "[In]. . . the UK, the model was, we'll do all requirements, [then] we'll do technical design and then, we will do development" (Programme Manager, Company E, February 2010).

Using plan-based approaches, each phase in the development process is completed in it's entirety, with all the requirements analysis, followed by all of the design, then the implementation, followed by testing. In contrast,

> "The US [team] had this concept of agile, by having what they call 'capsules.' So what they did was, they developed the requirements and then [implemented the] program and tested a capsule then move to the next one. . . At that time the US [team approach] was to do every thing, in little capsules" (Programme Manager, Company E, February 2010).

Here the term 'capsule' is used to describe the concept of a feature, and each feature is developed in turn. Each feature comprises requirements analysis, design, implementation and testing. In feature driven development methods (for example Coad et al, 1999), each feature is designed to encourage cohesion of closely related features but independence of unrelated features, to avoid undesirable coupling.

However,

> "[using capsules] struck us as a bit dangerous. Because you could get quite a long way down and find something in a capsule, 10 capsules on, that affected an earlier capsule that needed reworking" (Programme Manager, Company E, February 2010).

This highlights the difficulty of ensuring that features are independent. Failure to fully identify inter-dependencies can cause substantial re-work of previously implemented features. So, the plan-based approach was advocated, because it "was a means to avoid rework" (Programme Manager, Company E, February 2010).

Company B also had previous experience of using traditional plan-based methods "the development cycle used to be pretty huge, you know, huge requirement document, huge design phase, three or six [month] development cycle and so on" (Product Manager, Company B, April 2011). However, the time required to conduct the plan-based development process undermined customer satisfaction, as "there was always a huge difference [between] what our customers [were] demanding and what we were giving them" (Product Manager, Company B, April 2011). This is seen as an inevitable part of the development process "I think thats an inherent issue in the waterfall model" (Product Manager, Company B, April 2011). In summary,

> "[in a] typical waterfall model, where you had requirements being collected, analysed and then, you had a design phase. . . and then a development phase started and then a QA phase and then you realise everything is broken and things are not adhering to contracts, so you [the client and software service provider] fight with each other" (Development Manager, Company C, January 2010).

The issues arising from the absence of a product owner are illustrated by drawing on Company E, which conducts (non-agile) customisation projects on client sites, and uses scrum with an out-sourced development partner for its own product development. For example:

> "there are client representatives at [major international bank, UK] they are a little ill-defined in what they are actually doing. But there isn't a stakeholder that comes and makes any real decisions on the project" (Project Manager, Company E, February 2010).

This project manager continues "there's a lot more syndication. . . in coming to a conclusion [about project status]" (Project Manager, Company E, February 2010).

Here this project manager with over 30 years of IT industry experience uses the word syndication as a euphemism for discussion and equivocation. This illustrates that, from a development team perspective, it is difficult to get decisions made about requirements and priorities in the absence of a product owner. At the same bank

> "their governance is actually done on [a] biweekly basis and then, there's another level of governance which is done on a four- to six-week basis... they have a traffic light system of reporting on a project, but they have a very peculiar set of rules about how those traffic lights can go to amber or, very rarely, can go to red, which is very surprising... I would have all the projects at red at the moment, but they're either green or possibly only just about going amber" (Project Manager, Company E, February 2010).

From an enterprise perspective, it is difficult to hold project teams to account in terms of scope, budgets and deadlines. Enterprises tend to expand the number of job titles beyond the roles defined in standard agile literature. For example "there is a team in India of 20 people with a manager. There is a product manager as well as a project manager, and a product owner and there is me overseeing" (Engineering Director, Company E, February 2010). Product owner job functions need to be clarified, due to the lack of standardisation between job titles and product owner functions. Thus, product owner functions may be conducted by staff members holding a variety of job titles.

What a product owner actually does is worth considering. From one perspective, "product [scope], the design, the requirements, the discussion with the business side from the customer, that is all done through product owner" (Engineering Director, Company E, February 2010). The nine functions identified in the research as falling within the product owner role are outlined below.

4.1 Groom: the groom gathers requirements from business clients

Product grooming "is a list of... requirements or features... So it is a product owners responsibility to make sure product backlog should be continuously evolving" (Programme Head, Company H, May 2012). Simply put, "I'd be ready with my sprint backlog. What are the things we are going to do? Maybe a few additional features and some backlog from the previous [sprint] where we have received feedback from the client" (Engagement Manager, Company G, April 2011). Product owners reconcile conflicting priorities thus:

> "there is a product [management organisation in California]. So, different regions like APAC and Europe and North America come with some requirements, this is what they want to get done. And they come to the product manager... who keeps on collecting the requirements, prioritises them. There's a six-monthly road map discussion... where they have a high-level look at the things we are trying to achieve this year. They act as a channel between the development team and the market so, for us, they are the customers." (Engineering Manager, Company B, January 2010).

The product owner needs to interact with customers in order to gather the requirements "our product owner, luckily, has a very rich experience of interaction with customers" (Scrum Master, Company D, January 2010). However, simply compiling a list of requirements is not sufficient, the requirements must also be prioritised according to their value to the business.

4.2 Prioritiser: the prioritiser ensures that requirements bring value to the business

The product owner is also responsible for prioritising requirements in the product or sprint backlogs. For example, "we keep re-triaging [the backlog] because we have too many mandatories" (Engineering Director, Company E, February 2010). Also "if [requirements]. . . are not sequenced properly as for the priority. So whatever we are delivering will not give that priority value to the end user or the business user" (Programme Head, Company H, May 2012). Thus, it is the product owner's responsibility to identify and reconcile the needs of the different parts of the client organisation. Product owners become experienced in assessing and prioritising the needs of different segments of the customer-base, and thus they must have sufficient stature and seniority to perform that conflict resolution function.

In some cases, such as enterprise products simultaneously aimed at a large number of end-user territories, project teams can be more directly involved in the systematic recording of customer requirements: "we call it a requirement template approach. . . so this particular template recognises what is the basic requirement which all [of the client territories] will agree. [Each client territory in turn] will say "okay, this is less priority for me" or there'll be some variation. So against each [requirement template] there is some line item that can be done, okay we want this piece" (Practice Head, Company A, January 2010).

4.3 Release Master: the release master manages and approves release plans

A release master manages and approves release schedules. For example, "a release plan is prepared. It is sent for approval to the product owner. . . once it's approved by the product owner then we stick to that" (Architect, Company D, January 2010). Backlog grooming, requirements prioritisation and release planning are standard scrum practices. The other product owner functions practitioners described are identified below.

4.4 Technical Architect: the technical architect provides architectural coordination to large scale offshore enterprise software development programmes

To design, implement and disseminate a reference architecture, a technical function for the product owner is required. The product owner coordinates technical and architectural policies between the scrum teams. This includes sufficient documentation and illustrative source code to ensure project teams can understand and follow the guidelines. The product owner must disseminate the reference architecture to teams, as the project team may legitimately find anomalies and ambiguities that the product owner should provide guidance on. A significant amount of up-front work should be done by the product owner prior to scrum teams working on increments. These architectural design decisions can often be performed using broad statements of functional requirements. However, non-functional requirements have more bearing on the development of a reference architecture, as this is a technical thought leadership and coordination function. Architectural styles are developed and disseminated by the product owner in compliance with corporate architecture guidelines “so we have a [corporate-wide] council, architecture council” (Architect, Company D, January 2010). The absence of a product owner does not mean that these decisions are not made, on the contrary, the decisions are made by teams in an uncoordinated manner. This results in divergent architectural styles in different parts of the development programme.

There are tensions between performing the architecture design within the self-organising scrum teams or as a separate function. For example “in an ideal waterfall model, we have separate architects, separate designers, separate developers, then the testing teams separate. In a scrum team they’re all working together and there is no differentiation” (Engagement Manager, Company G, April 2011). The problem is that “I have observed that people who are working in the waterfall model who were not taking responsibility [for delivering a quality product on time]” (Engagement Manager, Company G, April 2011). The self-organising teams must also learn about the reference architecture, “this requires every individual to put a lot of effort into understanding the R&D and understanding the architecture. . . this responsibility is shared with the team and not just lying with only the product owner and scrum master” (Scrum Master, Company D, January 2010).

4.5 Governor: the governor ensures project compliance with corporate guidelines and policies

A large corporate e-Commerce website provides the “public face, in 26 languages, across the world, right?” (Lead Architect, Company H, May 2012) for their client. There are significant technical and reputational risks in deploying changes to such a website without a careful review process. That “website needs to have technical governance on how changes are made” (Lead Architect, Company H, May 2012). To achieve such technical governance “you need technical product ownership, especially in large programs right? If you are [working on] something like [MajorAirline.com], right?” (Lead Architect, Company H, May 2012).

The product owner provides a technical governance framework to project teams working on a development programme. The product owner will liaise with governance structures, such as technical oversight committees and corporate architecture groups. The product owner will ensure the selection of common tools and technologies for the project. This function will ensure projects within a development programme share an appropriate technology infrastructure.

The product owner must be aware of corporate governance policies and strategic directions. Any proposals made by the product owner should comply with such governance directives.

4.6 Communicator: the communicator connects onshore and offshore geographical distribution

Geographical distribution is not an ideal attribute for a project team, but it nevertheless remains a feature of the globalized nature of large scale offshore enterprise software development programmes. At a senior level there is a view that geographical distribution itself is not a major problem to overcome. "I think we are now very mature. Initially we thought that our biggest challenge would be the geographical distance, some people here, some people there. So that is definitely a challenge but that is I would say the easiest to fix" (Chief Architect, Company H, May 2012).

Geographical distribution within scrum team membership is discouraged in the companies investigated. Company B avoids geographically distributed scrum teams, the scrum team is co-located "we do our own things here [offshore]" (Engineering Manager, Company B, January 2010). The client is onshore and uses teleconferencing during coordination meetings, "Product managers and UI designers, they dial in from [onshore]" (Engineering Manager, Company B, January 2010).

The compromise of adding a remotely located technical specialist to the co-located team is sometimes helpful "We had one engineer working from [California] join us, she used to directly dial in to our sprint planning meetings, retrospection, everything" (Engineering Manager, Company B, January 2010). This is onerous to the individual when the time zone difference is large (since the remote technical specialist usually has to adopt working patterns to suit the rest of the co-located development team). Sometimes

> "participation in the stand-up is extremely tough because you do it in a room with a speaker phone and five people talking about different things, it becomes very tough for a [remote] person to understand what's going on and contribute we found it was better to have a one-on-one call with her" (Engineering Manager, Company B, January 2010).

Development team working patterns can be adapted to increase the overlap between office hours onshore and offshore. Where the time zone difference is not too great, offshore scrum teams can work into the evening "our normal shift is 11:00[am]-8:00[pm]" (Software Engineer, Company H, May 2012) and from a different project team "we are working 1pm to 10pm" (Project Manager, Company H, May 2012). This is an example where time zone differences between Europe and India are accommodated by shifting work patterns in India to increase the number of overlapping working hours. Offshore staff members can sometimes expect to receive extra payments and free meals when working this type of pattern, and transportation is often provided to staff, whatever shift patterns are being worked. Working after midnight is not popular with respondents, because it disrupts weekend, social, and family life; whereas early evening working can be accommodated without too much disruption.

A more common arrangement is a co-located offshore scrum team working with a remote product owner: "I've got a product owner. . . based in New Zealand" (Engineering Director, Company E, January 2010), reflecting a classic onshore/offshore model. Again, mutual timings for meetings must be found, which is challenging with such a large time zone difference. Video (and less commonly, audio) conferencing is used to conduct daily scrum coordination meetings, which are sometimes conducted as stand-up meetings (using video conferencing technologies).

Communication challenges include the "practical difficulties making sure people can interact; such as booking video conference rooms at both ends, Internet connectivity limitations on video conferencing; tendency to underestimate investment required in travel and connectivity" (Delivery Manager, Company H, May 2012).

4.7 Traveller: the traveller spends time onshore at client sites, gathering first-hand knowledge of a client's needs

Product owner teams have staff members onshore for discussions with clients and off-shore for disseminating information to development teams. The proxy product owner will usually spend time (between one and three months, depending on the scale of the project) on the client site at the start of the project, becoming familiar with any special features of the client's requirements. The traveller is important for supporting development teams: "we have a field person at the customer site who can answer queries... They're the ones who interact with the customer directly" (Architect, Company D, January 2010).

The size of the project determines the breakdown of staff assignment between onshore and offshore "if you are sort of 70 people working from the offshore, [then] there are 5 to 6 guys at the onsite" (Practice Head, Company A, January 2010). This has the advantage that staff are onsite "every day at the same time zone [as the client]" (Practice Head, Company A, January 2010).

Therefore, product owners act as a bridge between the onshore and offshore divisions. They can be based with clients, or they may travel to clients. Alternatively, and at higher cost, entire teams can travel to clients to co-locate with product owners.

Enterprises may also adopt a range of solutions in terms of co-locating teams. "One scrum team is based onshore completely, the other scrum team was sent onshore for a month, had two iterations [onshore], and then came back and started doing iterations [offshore], and then the third team is a distributed team that has been formed. So everybody is getting the exposure of working directly with the [onshore] product owners and the business analysts" (Delivery Manager, Company H, May 2012).

4.8 Intermediary: the intermediary acts as an interface with senior executives, driving large scale offshore enterprise software development programmes, and disseminating domain knowledge to teams

The product owner is supplemented by an intermediary from within the development team, to mitigate domain complexity. The "proxy product owner [is] an extension of the product owner, the product owner's availability or understanding of the off-shoring process being limited" (Delivery Manager, Company H, May 2012). The intermediary will need to have extensive experience of the system business domain. In Company B,

> "we have some kind of shared product ownership, very limited that is done by [local Product Manager]. But it is mostly on requirements. . . we have like four/five different conferences [conference calls] with different stakeholders. So, like product managers and some other folks, engineering folks, and some people who are working on performance, for example" (Engineering Manager, Company B, January 2010).

4.9 Risk Assessor: the risk assessor evaluates technical complexity

Enterprises routinely conduct risk management in order to assess technical complexity and potential shortcomings in the development teams' skills and capabilities. Product owners perform

> "risk management, if something is seen as technically very complex, it will come up as part of the risk [assessment] of that particular project. Then, you will have to see how you mitigate that risk; if it requires support from a centre of excellence within [the company] or stronger [staff technical] profiles to be a part of that project then we will do that." (Delivery Manager, Company H, May 2012).

The risk mitigation might include "assigning people called Technical Analysts, I mean SMEs [subject matter experts] with respect to their technical domain understanding" (Programme Head, Company H, May 2012).

Technical Specialists assigned to the scrum team can provide insight into managing complexity. For example, where multiple programming languages are being used the interfaces between language components require special skills. Similarly interfaces to external systems and credit card payment gateways require access to a technical specialist. Intelligent choices need to be made between sharing a technical specialist between one or more sprint teams. Where the technical complexity is affecting the work of an entire sprint team, then a 100% assignment of a full-time technical specialist makes sense. Where the technical complexity affects some aspect of the work of the sprint team, then access to a part-time technical specialist will usually be sufficient.

5 Discussion

The findings presented here support previous research suggesting that agile methods can be scaled to large offshore enterprise software development programmes (Reifer et al, 2003) and used in globally distributed software development settings (Ramesh et al, 2006). A primary function within the product owner role is to communicate customer needs to software team members, and the negative impact on project outcomes of inadequate customer availability has been highlighted elsewhere (Hoda et al, 2011).

Sutherland et al (2007) explored scrum developers working together while based in the USA, Canada and Russia. That study emphasised tailoring agile method practices rather than the functions performed within roles. One of the companies studied by Sutherland et al (2007) centralised and co-located scrum masters, product owners and architects, while companies A, B, D, E and H in this study each had a distributed model of product ownership.

Product owners are required to undertake new work functions in order to manage large scale offshore enterprise software development programmes. The product ownership team concept emerges, at least in part, from a need to manage multiple offshore development teams and onshore product owners.

Earlier research has focused on XP customer teams (Martin, 2009). Three XP customer functions have been identified that address collaboration within teams: geek interpreter, political advisor and technical liaison (Martin et al, 2009b). A geek interpreter improves communication between technical team members and the business. The political advisor skilfully navigates political dimensions to achieve project success. Technical liaison coordinates with other related projects and technical specialist groups residing within the organisation. The three collaboration functions exist alongside four direction setting roles: negotiator, diplomat, super-secretary and customer coach (Martin et al, 2009b). The negotiator gains agreement from stakeholders on a single vision for the software under development. Diplomats represent a particular stakeholder interest group in project decision-making. The super-secretary provides administrative support to the XP customer team, and the XP customer coach nurtures and supports members of the customer team.

In contrast, this research has focused on practitioner descriptions of functions found within the product owner role. Its contribution is to articulate the functions undertaken by product owners, arguing that product owner teams are required to manage the scale and complexity of product owner functions in large scale offshore enterprise development programmes.

The findings presented here also confirm other research that advocates focusing scrum of scrum coordination meetings around specific product or feature areas, rather than across entire development programmes (Paasivaara et al, 2012b). However, this research develops such ideas further, describing nine functions that comprise the product owner role in large scale offshore enterprise software development programmes: groom, prioritiser, communicator, traveller, intermediary, governor, technical architect, risk assessor and release manager.

The groom, prioritiser and release manager functions comply with the product owner role defined in early scrum literature (Schwaber and Beedle, 2001; Schwaber, 2004). The groom cultivates requirements by liaising with clients and customers, using their domain knowledge and experience of previous similar projects. The groom corresponds closely to the negotiator XP customer acrtivity identified by Martin et al (2009b). The prioritiser ensures that high value requirements are identified and implemented. In this study, more than one product owner talked about 'triaging' requirements. This triaging did not appear to be reflected in Martin et al (2009b). This could suggest that product owners are taking greater responsibililty for deciding prioritisation than XP customers, who tend to focus on negotiation. The release manager ensures quality and scope targets are achieved prior to code release to users. In summary, the groom, prioritiser and release manager functions identified here closely correspond with aspects of the core product owner role from the scrum literature.

Technical architect, governor, communicator, traveller, intermediary and risk assessor functions are not described as product owner functions in early scrum literature. Architectural coordination on large projects is provided by technical architects. The technical architect combines elements of the geek interpreter and technical liaison XP customer function identified by Martin et al (2009b). This study did not find evidence of the political advisor, diplomat, super-secretary and customer coach XP customer activites identified by Martin et al (2009b). It was found instead that the super-secretary function is performed by project managers and not product owners in the companies investigated.

Governors ensure project compliance with corporate guidelines and policies. This corporate governance is partly a requirement for maintaining CMMI maturity level 5 compliance, demonstrating consistent quality standard and procedures across development programmes. Communicators bridge onshore and offshore geographical distribution; while travellers spend time onshore at client sites in order to gather first-hand knowledge of a client's needs. Communicators and travellers are conveying on-shore client needs to off-shore development teams. Intermediaries act as an interface to senior executives driving large scale offshore enterprise software development programmes, while risk assessors evaluate technical complexity.

5.1 Implications for Product Owners

The product owner functions can be used to mitigate risks of overlooking activities required on large scale offshore enterprise software development programmes. The groom, prioritiser and release master are core product owner functions and are mandatory on all projects. The risk assessor, technical architect, governor and intermediary are mandatory to support large scale development programmes. While, communicator and traveller are mandatory to manage geographical distribution. Programme managers, product owners and teams should be alert for missing functions, because these are likely to pose risks for successful programme completion.

This research raises three main issues for agile practitioners (such as product owners) in large scale offshore enterprise software development programmes. First, product sponsors in some teams make explicit the process of gathering a product owner team around themselves. Second, product owner teams comprise both management and technical representatives. Third, some product owners use teams to induct and mentor new product owners. This research suggests, there is a higher degree of maturity in organisations where product owner teams are explicitly established. In less mature organisations, the product owner team functions can be observed but are somewhat unconscious acts of stakeholders and even the participants themselves. Less mature organisations miss opportunities to develop product owner skills through induction processes.

The breadth of product owner functions identified in this research extend beyond the scope and skill set of a single individual. This breadth of skills illustrates the need for explicit product owner team formation during the early stages of large scale offshore enterprise software development programmes, as the explicit formation of a team with the full range of business and technical skills is required.

The product owner functions can now be mapped to a model of the scrum of scrums process. The product owner team is comprised of both management-facing (often business or application domain) specialists and technical-facing specialists. Figure 3 shows the business facing product owner functions and associated information flows. Here groom, prioritiser, communicator, traveller and intermediary functions are primarily focused on communication between onshore and offshore

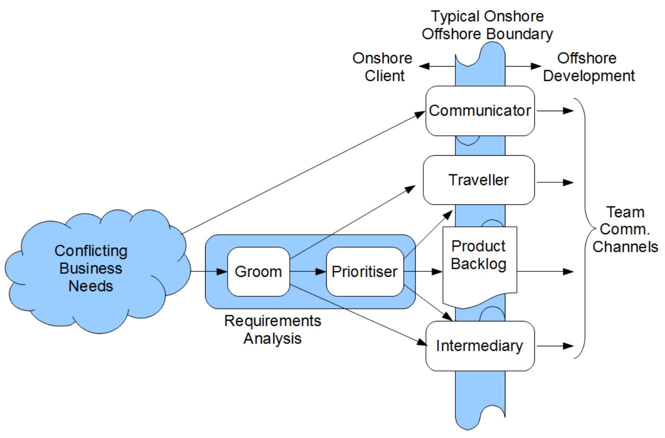

Fig. 3 Client-side product owner functions and information flows.

actors and product backlog production. The main focus of these activities is to gather and prioritise requirements in the face of conflicting business needs.

In contrast, Figure 4 shows the development team facing product owner functions. The Figure reflects the scrum of scrum context by showing multiple parallel sprints. The parallel sprints comprise multiple sprint backlogs, sprints, customer demonstrations and code releases. The governor, technical architect, risk assessor and release master functions are focused on these software production practices. An emphasis here is on meeting the needs of CMMI Level 5 accreditation in areas such as record keeping and process governance, while retaining the agility and responsiveness offered by the scrum method. The product owner team can also be used to induct new staff members into positions of responsibility on large scale offshore enterprise software development programmes. For example, new technical architects can be given support roles on larger development programmes, while taking a leadership role on a smaller development programme. This provides a well-defined career development route, that also reduces risk.

5.2 Implications for Process Tailoring

Large scale offshore enterprise software development programmes are instructive for process theorists, since they are at the boundaries of applicability for agile methods. This research identifies nine product owner team functions, which affect three main aspects of the software development process; namely: requirements, technical oversight and on-shore/off-shore communications. Requirements gathering and feature delivery is undertaken during the groom, prioritiser and release master functions. Technical oversight is undertaken during the governor, technical architect and risk assessor functions, whilst on-shore/off-shore communications are performed during the communicator, traveller and intermediary functions.

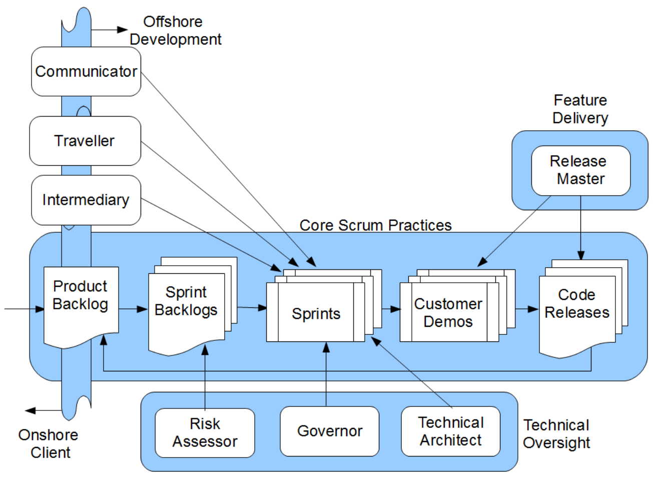

Fig. 4 Production-side product owner functions and information flows.

Table 5 A taxonomy of product owner team functions

	Information Gathering (& Decision-Making)	**Information Dissemination**
Business	Groom Prioritiser	Communicator Traveller Intermediary
Technical	Release Master Risk Assessor	Technical Architect Governor

Scrum helpfully provides physical expressions for the requirements gathering and feature delivery functions, in the form of the product backlog, sprint backlogs and software releases. However, no physical expressions are defined for technical oversight or on-shore/off-shore communications. Industry best practices are available in these areas, such as reference architectures or project wikis, but they are not currently part of the scrum method.

Scrum also provides ceremonies for the requirements gathering and feature delivery functions in the form of sprint kickoffs and customer demonstrations. Again, alas, scrum does not provide ceremonies for technical oversight or on-shore/off-shore communications.

A taxonomy of the product owner team functions can be derived from the results of this research, and is shown in Table 5. The taxonomy emerges from the analysis of the product owner functions themselves and from their mapping to the scrum of scrums process presented in Figures 3 and 4. Table 5 shows that groom, prioritiser, release master and risk assessor functions focus on information gathering leading to decision-making. In contrast, the functions communicator, traveller, intermediary, technical architect and governor focus on information dissemination.

6 Limitations

Three tests for evaluating the quality of descriptive empirical social research have been identified: construct validity, external validity and reliability (Yin, 2009). Construct validity can be ensured by using multiple sources of evidence. This has been achieved through conducting studies at eight companies. As mentioned in Section 3.1, a combination of snowball and intensity sampling was used. The snowball sampling provided a cross-section of numerous projects, while the intensity sampling at Companies D, E and H enabled detailed investigations with corporate executives, project portfolio managers, project managers as well as various development team member roles. This intensity sample offers sources of evidence and occasionally different perspectives on the software development processes used.

External validity is achieved by conducting the studies across eight companies and through the wide range of project stakeholder respondents. However, the findings and conclusions presented here cannot be generalised to small and medium sized companies, as smaller companies work under profoundly different commercial pressures with different quality assurance responsibilities.

Reliability is developed by conducting data collection, coding and analysis until categories are saturated. Saturation is said to have occurred when no new information seems to emerge from the data. Data collection and analysis should continue by studying first one group and then other groups until categories become settled and stable. The reliability of categories in this research have been established from the perspective of different actors within project teams and their surrounding organisational contexts. Further, the categories have been explored by investigating actors from different organisations. It is argued saturation has been achieved by using these techniques.

7 Conclusions

This paper has used practitioner descriptions of agile method tailoring to contribute to the literature on large-scale enterprise software development. Specifically, tailoring of the product owner role has been investigated. The existence of product owner teams has also been observed.

Nine product ownership functions that are used to scale agile methods to large projects have been described: groom, prioritiser, release master, technical architect, governor, communicator, traveller, intermediary and risk assessor. The groom gathers requirements from business clients; while the prioritiser ensures that requirements bring value to the business. The release master manages and approves release plans; and technical architects provide architectural coordination on large projects. This architectural coordination is achieved by using reference architectures to guide and support self-organising scrum teams. Governors are required to ensure project compliance with corporate guidelines and policies. The communicator bridges onshore and offshore geographical distribution; while travellers spend time onshore at client sites, gathering first-hand knowledge of a client's needs. The intermediary acts as an interface to senior executives driving large scale development programmes, disseminating domain knowledge to teams, and the risk assessor evaluates technical complexity.

The product owner functions identified in this research have been mapped to the scrum-of-scrums process. This mapping reveals two function classes: client-side and production-side. The client-side functions are: groom, prioritiser, communicator, traveller and intermediary. The production-side functions are risk assessor, governor, technical architect and release master. The client-side functions primarily concern requirements capture and the dissemination of business needs to development teams; while production-side functions, in contrast, primarily focus on technical oversight and the delivery of features.

A taxonomy of the functions has been produced. The taxonomy distinguishes between business and technical functions, identifying tasks that primarily involve information gathering; to enable product owner decision-making and information dissemination functions. The information gathering functions are groom, prioritiser, release master and risk assessor. The information dissemination functions are communicator, traveller, intermediary, technical architect and governor.

These product owner functions provide a layer of governance to agile methods in the CMMI maturity level 5 accredited companies investigated. Furthermore, these functions can be performed by members of a product owner team. Thus, product owners can devolve selected functions to technical and remotely located colleagues. The product owner team becomes a key tool in tailoring agile methods for large scale distributed projects.

Future work will explore scrum master functions in large-scale enterprise development projects. Scrum masters also form teams, such as when a scrum-of-scrums project configuration is used. This creates an additonal range of responsibilities and concerns for scrum masters.

Acknowledgements The anonymous reviewers are thanked for their detailed comments, which have substantially improved the clarity of the article. I am also grateful to all the companies and interviewees who were generous enough to contribute their time and resources to participate in this research. Thanks also go to the current and former students of the Executive MBA at the Indian Institute of Management, Bangalore, who helped to identify target companies. The research benefited in part from travel funding from the UK Deputy High Commission, Bangalore, Science and Innovation Network; the Institute for Innovation, Design & Sustainability (IDEAS) at Robert Gordon University, UK; and accommodation and sustenance from Company H during the data collection visit to Delhi, India.

Appendix 1

Interview Guide for Off-Shore Agile Software Projects, January 2010

Background Notes

I want to ask you about your experience of geographically distributed agile software development projects. The research involves interviews with people doing a range of different roles and from companies with different development models.

I want to learn more about your views on agile processes. I am particularly interested to know what factors are affected by geographical location and separation. The purpose here is to try to understand the factors that affect project outcomes, successful or otherwise, so that we can try to learn for the future.

Interview Guide

I want to ask you the following questions and tape record your answers. I will keep your responses completely confidential and nothing will be shared with any client companies. I do plan to publish interview extracts but I will make names and companies anonymous.

Can I switch on the recorder?

Your Current Project(s)

How many projects are you working on currently?
What is (was) the title of your current (or most recent) project?
What is the project management structure?
How is the project organised geographically?
Is the project maintenance/evolution or a new build project?
Is the project COTS or bespoke?
How many people are in the project team?

Agile Practices

What are the special features of agile projects (compared with other development processes)?
What agile project development practices are being used in your project?
What agile practices are not used?

Requirements

How are requirements decided and prioritised?
How do user stories evolve over time?
How do user stories move up or down the backlog?
How do you interact with customer representatives (XP)? Are they onsite?

Product Owner/Customer (POC)

How do you represent the product development team within the client organisation?
How do you represent the client organisation within the product development team?
Which remotely located stakeholder groups are most supportive or challenging?
Which remotely located stakeholder groups do you interact with most/least frequently?
What would help make life easier for you?

Releases and Testing

How does unit tested code become a release?
How is user acceptance testing managed?
How are bugs reported back, prioritised and fixed?

Social Media/Cloud

What forms of social media or electronic communication are used in the projects?
What forms of cloud computing services are used in the projects?

Learning

How does learning take place within the team?
How does learning take place for you personally?

Any other comments

Now, I want to check if there is anything else you would like to add?
Do you have any further comments in relation to geographically distributed agile development projects?

About Your Organisation

Now I want to ask some questions about you and your organisation. These details will be kept confidential.

What is the name of your organisation?
What industry sector is your organisation in?
What industry sector are your development projects in?
How many people are there in your organisation?

About You?

Your name?
Your age?
What is your role (product owner, developer, architect or scrum master)?
What experience or formal qualifications do you have?
How much experience do you have or when did you qualify?
How long have you been working in your current organisation?

Appendix 2

Interview Guide, Agile Method Tailoring [Company H], May 2012

Background Notes

I want to ask you about your experience of agile software development projects. The research involves interviews with people holding a range of different roles, and from companies with different development models.

The purpose here is to try to understand how agile methods are used in [Company H], so that we can try to learn in the future. I want to ask you the following questions and tape record your answers. I do plan to publish interview extracts but I will make names, projects and clients anonymous. Can I switch on the recorder?

Agile Processes

What agile methods and practices are you using?
Would you describe agile methods as being successful for you? In what ways?
What challenges have you encountered with agile methods?

Scaling to Enterprise Projects

Describe any software tools or technologies you use to support agile methods?
Have you adapted agile methods because of the geographical distribution of the team?
Have you adapted agile methods because the client organisation was geographically distributed?
Have you adapted agile methods because of a particularly large team?
Have you used agile methods in a context with demanding regulatory compliance? What adaptations did you make?
Have you used agile methods in a particularly complex domain context? What adaptations did you make?
Have you used agile methods on a particularly technically complex project? What adaptations did you make?
Have you used agile methods with an especially complex range of stakeholder relationships? What adaptations did you make?
Have you adapted agile methods for use on a strategically important enterprise architecture programme?

Future Perspectives

What future trends do you forsee in your use of agile methods?
If there was one thing you could change about the way agile methods are used at [Company H] what would it be?
What advice would you give to improve agile productivity?

What advice would you give to improve agile product quality?
What advice would you give to improve transitioning to offshore agile?

About Your Project(s)

Now I want to ask some questions about you and your project. These details will be kept confidential.

What project are you working on currently? How many projects?
How is the project team structured (for management purposes)?
How is the project team organised geographically?
What is the project domain? What is the project purpose?
How large is the project in terms of team size? In terms of value?
When did the project start?
How much longer will the project run for?

Any other comments

Now, I want to check if there is anything else you would like to say?
Do you have any further comments in relation to agile methods?

About You?

Your name?
Your age?
What is your role (product owner, developer, architect or scrum master)?
How long have you been in your current role?
How long have you been working in your current organisation?
How long have you been working in the software industry?
What formal qualifications do you have, if any?

References

Agile Alliance (2011) Agile alliance. http://www.agilealliance.org/, [accessed 25-09-2011]

Ambler S (2008) Agile software development at scale. In: Meyer B, Nawrocki J, Walter B (eds) Balancing agility and formalism in software engineering, lecture notes in computer science, vol 5082, Springer Berlin Heidelberg, pp 1–12, DOI 10.1007/978-3-540-85279-7_1, URL http://dx.doi.org/10.1007/978-3-540-85279-7_1

Ambler SW, Lines M (2012) Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise. IBM Press, Boston, MA, USA

Balijepally V, Mahapatra R, Nerur S, Price KH (2009) Are two heads better than one for software development? The productivity paradox of pair programming. MIS Quarterly 33(1):91–118, URL http://misq.org/are-two-heads-better-than-one-for-software-development-the-productivity-paradox-of-pair-programming.html

BBC (2012) BBC news - Bank merger account glitches. http://news.bbc.co.uk/1/hi/programmes/moneybox/8946199.stm, [accessed 15-09-2013]

Beck K, Andres C (2004) Extreme programming explained, 2nd edn. Addison Wesley, Boston, MA, USA

Begel A, Nagappan N (2007) Usage and perceptions of agile software development in an industrial context: An exploratory study. In: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, Washington, DC, USA, ESEM '07, pp 255–264, DOI 10.1109/ESEM.2007.85, URL http://dx.doi.org/10.1109/ESEM.2007.85

de Cesare S, Lycett M, Macredie RD, Patel C, Paul R (2010) Examining perceptions of agility in software development practice. Commun ACM 53(6):126130, DOI 10.1145/1743546.1743580, URL http://doi.acm.org/10.1145/1743546.1743580

Chow T, Cao DB (2008) A survey study of critical success factors in agile software projects. Journal of Systems and Software 81(6):961 – 971, DOI http://dx.doi.org/10.1016/j.jss.2007.08.020, URL http://www.sciencedirect.com/science/article/pii/S0164121207002208

Coad P, LeFebvre E, Luca JD (1999) Java modeling in color. Prentice Hall, Englewood Cliffs, NJ, USA

Cockburn A (2001) Agile software development. Addison Wesley, Reading, MA, USA

Cohn M (2009) Succeeding with agile: Software development using scrum, 1st edn. Addison-Wesley Professional, Upper Saddle River, NJ, USA

Computer Weekly (2012) Santander migration glitch affects Alliance & Leicester customers. http://www.computerweekly.com/news/1280093541/Santander-migration-glitch-affects-Alliance-Leicester-customers, [accessed 15-09-2013]

Cusumano MA (2007) Extreme programming compared with Microsoft-style iterative development. Commun ACM 50(10):15–18, DOI 10.1145/1290958.1290979, URL http://doi.acm.org/10.1145/1290958.1290979

Dyba T, Dingsoyr T (2009) What do we know about agile software development? IEEE Software 26(5):6–9, DOI http://doi.ieeecomputersociety.org/10.1109/MS.2009.145

França A, da Silva F, de Sousa Mariz L (2010) An empirical study on the relationship between the use of agile practices and the success of scrum projects. In: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, New York, NY, USA, ESEM '10, pp 37:1–37:4, DOI 10.1145/1852786.1852835, URL http://doi.acm.org/10.1145/1852786.1852835

Gibbs G (2007) Analyzing qualitative data. Sage Publications Ltd, London, UK

Glaser BG (1992) Basics of grounded theory analysis: Emergence vs. forcing. Sociology Press, Mill Valley, CA, USA

Glaser BG (1998) Doing grounded theory: Issues and discussions. Sociology Press, Mill Valley, CA, USA

Glaser BG, Strauss AL (1999) Discovery of grounded theory: Strategies for qualitative research. Aldine Transaction, Piscataway, NJ, USA

Glick B (2013) DWP writes off millions of pounds on universal credit IT, damning NAO report reveals. http://www.computerweekly.com/news/2240204715/DWP-writes-off-millions-of-pounds-on-Universal-Credit-IT-damning-NAO-report-reveals, [accessed 02-11-2013]

Hannay JE, Benestad HC (2010) Perceived productivity threats in large agile development projects. In: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, New York, NY, USA, ESEM '10, pp 15:1–15:10, DOI 10.1145/1852786.1852806, URL http://doi.acm.org/10.1145/1852786.1852806

Hannay JE, Arisholm E, Engvik H, Sjoberg DIK (2010) Effects of personality on pair programming. IEEE Transactions on Software Engineering 36(1):61–80, DOI http://doi.ieeecomputersociety.org/10.1109/TSE.2009.41

Hansard (2012) House of Commons Hansard debates for 11 Sep 2012 (pt 0001). http://www.publications.parliament.uk/pa/cm201213/cmhansrd/cm120911/debtext/120911-0001.htm, [accessed 15-09-2013]

Herbsleb JD, Mockus A (2003) An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering 29(6):481–494, DOI http://doi.ieeecomputersociety.org/10.1109/TSE.2003.1205177

Hoda R, Noble J, Marshall S (2010) Organizing self-organizing teams. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ACM, New York, NY, USA, ICSE '10, pp 285–294, DOI 10.1145/1806799.1806843, URL http://doi.acm.org/10.1145/1806799.1806843

Hoda R, Noble J, Marshall S (2011) The impact of inadequate customer involvement on self-organizing agile teams. Information and Software Technology 53(5):521–534

Hoda R, Noble J, Marshall S (2012) Developing a grounded theory to explain the practices of self-organizing agile teams. Empirical Software Engineering 17(6):609–639, URL http://link.springer.com/article/10.1007\%2Fs10664-011-9161-0

Hossain E, Babar M, Paik Hy (2009) Using scrum in global software development: A systematic literature review. In: Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on, pp 175–184, DOI 10.1109/ICGSE.2009.25

Hove S, Anda B (2005) Experiences from conducting semi-structured interviews in empirical software engineering research. In: Software Metrics, 2005. 11th IEEE International Symposium, pp 10 pp.–23, DOI 10.1109/METRICS.2005.24

Jalali S, Wohlin C (2010) Agile practices in global software engineering - a systematic map. In: IEEE 5th International Conference on Global Software Engineering (ICGSE), IEEE, pp 45 –54, DOI 10.1109/ICGSE.2010.14

Ktata O, Lévesque G (2009) Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. In: Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering, ACM, New York, NY, USA, C3S2E '09, pp 59–66, DOI 10.1145/1557626.1557636, URL http://doi.acm.org/10.1145/1557626.1557636

Kvale S, Brinkmann S (2009) Interviews, learning the craft of qualitative research interviewing, 2nd edn. Sage Publications, Inc, Thousand Oaks, CA, USA

Larman C, Basili V (2003) Iterative and incremental development: A brief history. Computer, IEEE 36(6):47–56, DOI http://doi.ieeecomputersociety.org/10.1109/MC.2003.1204375

Larman C, Vodde B (2008) Scaling lean and agile development: Thinking and organizational tools for large-scale scrum: Successful large, multisite and offshore products with large-scale scrum. Addison Wesley, Upper Saddle River, NJ, USA

Leffingwell D (2007) Scaling software agility: Best practices for large enterprises. Addison Wesley, Boston, MA, USA

Lui KM, Chan KCC, Nosek J (2008) The effect of pairs in program design tasks. IEEE Transactions on Software Engineering 34(2):197–211, DOI http://doi.ieeecomputersociety.org/10.1109/TSE.2007.70755

Martin A (2009) The role of the customer in agile projects. PhD thesis, Victoria University of Wellington, New Zealand

Martin A, Biddle R, Noble J (2009a) Xp customer practices: A grounded theory. In: Agile Conference, 2009. AGILE '09., pp 33–40, DOI 10.1109/AGILE.2009.68

Martin A, Biddle R, Noble J (2009b) The xp customer team: A grounded theory. In: Agile Conference, 2009. AGILE '09., pp 57–64, DOI 10.1109/AGILE.2009.70

Mens T, Tourwe T (2004) A survey of software refactoring. Software Engineering, IEEE Transactions on 30(2):126–139, DOI 10.1109/TSE.2004.1265817

Miles MB, Huberman AM (1994) Qualitative data analysis: An expanded sourcebook, 2nd edn. Sage Publications, Inc, Thousand Oaks, CA, USA

Moe NB, Dingsøyr T, Dybå T (2010) A teamwork model for understanding an agile team: A case study of a scrum project. Inf Softw Technol 52(5):480–491, DOI 10.1016/j.infsof.2009.11.004, URL http://dx.doi.org/10.1016/j.infsof.2009.11.004

Monteiro CV, da Silva FQ, dos Santos IR, Farias F, Cardozo ES, do A Leitão AR, Neto DN, Pernambuco Filho MJ (2011) A qualitative study of the determinants of self-managing team effectiveness in a scrum team. In: Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering, ACM, New York, NY, USA, CHASE '11, pp 16–23, DOI 10.1145/1984642.1984646, URL http://doi.acm.org/10.1145/1984642.1984646

NVivo (2013) NVivo 9 help. http://help-nv9-en.qsrinternational.com/nv9_help.htm, [accessed 10-09-2013]

Paasivaara M, Durasiewicz S, Lassenius C (2008) Using scrum in a globally distributed project: A case study. Software Process: Improvement and Practice 13(6):527–544, DOI 10.1002/spip.402, URL http://dx.doi.org/10.1002/spip.402

Paasivaara M, Heikkilä VT, Lassenius C (2012a) Experiences in scaling the product owner role in large-scale globally distributed scrum. In: 2012 IEEE 7th International Conference on Global Software Engineering, IEEE Computer Society, Los Alamitos, CA, USA, pp 174–178, DOI http://doi.ieeecomputersociety.org/10.1109/ICGSE.2012.41

Paasivaara M, Lassenius C, Heikkilä VT (2012b) Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work? In: Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, ACM, New York, NY, USA, ESEM '12, pp 235–238, DOI 10.1145/2372251.2372294, URL http://doi.acm.org/10.1145/2372251.2372294

Patton MQ (2002) Qualitative research & evaluation methods, 3rd edn. Sage Publications, Inc, Thousand Oaks, CA, USA

Pikkarainen M, Haikara J, Salo O, Abrahamsson P, Still J (2008) The impact of agile practices on communication in software development. Empirical Software Engineering 13(3):303–337, DOI 10.1007/s10664-008-9065-9, URL http://dx.doi.org/10.1007/s10664-008-9065-9

Poppendieck M, Poppendieck T (2003) Lean software development: An agile toolkit. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA

Raithatha D (2007) Making the whole product agile: A product owners perspective. In: Proceedings of the 8th international conference on Agile processes in software engineering and extreme programming, Springer-Verlag, Berlin, Heidelberg, XP'07, pp 184–187, URL http://dl.acm.org/citation.cfm?id=1768961.1769003

Ramesh B, Cao L, Mohan K, Xu P (2006) Can distributed software development be agile? Commun ACM 49(10):4146, DOI 10.1145/1164394.1164418, URL http://doi.acm.org/10.1145/1164394.1164418

Reifer D, Maurer F, Erdogmus H (2003) Scaling agile methods. Software, IEEE 20(4):12–14, DOI 10.1109/MS.2003.1207448

Robson C (2011) Real world research, 3rd edn. John Wiley and Sons Ltd., Chichester, UK

Schwaber K (2004) Agile project management with scrum. Microsoft Press, Redmond, WA, USA

Schwaber K, Beedle M (2001) Agile software development with scrum. Prentice Hall, Upper Saddle River, NJ, USA

Seaman CB (1999) Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering 25(4):557–572, DOI http://doi.ieeecomputersociety.org/10.1109/32.799955

SFIA Foundation (2014) SFIA - skills framework for the information age. http://www.sfia-online.org/, [accessed 15-04-2014]

Sharp H, Robinson H (2004) An ethnographic study of XP practice. Empirical Software Engineering 9(4):353–375, DOI 10.1023/B:EMSE.0000039884.79385.54, URL http://dx.doi.org/10.1023/B\%3AEMSE.0000039884.79385.54

Stapleton J (1997) DSDM: Dynamic systems development method. Addison Wesley, Harlow, England

Sutherland J, Viktorov A, Blount J, Puntikov N (2007) Distributed scrum: Agile project management with outsourced development teams. In: System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on, p 274a, DOI 10.1109/HICSS.2007.180

van Waardenburg G, van Vliet H (2013) When agile meets the enterprise. Information and Software Technology 55(12):2154 – 2171, DOI http://dx.doi.org/10.1016/j.infsof.2013.07.012, URL http://www.sciencedirect.com/science/article/pii/S0950584913001584

Weiss RS (1994) Learning from strangers: The art and method of qualitative interview studies. Free Press, New York, NY, USA

Wilkerson JW, Nunamaker JF, Mercer R (2012) Comparing the defect reduction benefits of code inspection and test-driven development. IEEE Transactions on Software Engineering 38(3):547–560, DOI http://doi.ieeecomputersociety.org/10.1109/TSE.2011.46

Yin RK (2009) Case study research: Design and methods, 4th edn. Sage Publications, Inc, Thousand Oaks, CA, USA