# Boundary Objects and Change in Agile Projects

*Type of Submission: Research in Progress*

Anna Zaitsev
Business Information Systems
University of Sydney
Sydney, Australia
Email: anna.zaitsev@sydney.edu.au

Uri Gal
Business Information Systems
University of Sydney
Sydney, Australia
Email: u.gal@econ.usyd.edu.au

Barney Tan
Business Information Systems
University of Sydney
Sydney, Australia
Email: b.tan@econ.usyd.edu.au

## Abstract

*Agile software development projects utilise different boundary objects in volatile and dynamic project environment. This environment subjects the boundary objects to constant change. The behaviour of boundary objects has not been observed in Agile environment and is thus not well understood. Lack of comprehension can lead to misemployment of boundary objects. This Research-in-Progress describes a case study of a project where two organisations are collaborating to create a software product using Agile methods. The study identifies three different types of boundary objects: temporary, evolutionary and mediating boundary objects. The importance of these three object types to the Agile project is discussed.*

**Keywords**

Boundary objects, Agile software development, Change

## INTRODUCTION

The term Agile development methods refers to a collection of dynamic software development methods and best practices based on the principles of the Agile Manifesto (Beck et al. 2001). The Agile Manifesto (2001) outlined four statements and twelve principles which all encourage organisations to utilise collaborative and human centric software development. Since their introduction in the late 1990s, Agile development methods have been gaining prominence as a new standard in software development (Bustard et al. 2013; McHugh et al. 2012). Proponents of Agile acknowledge that the software development environment is chaotic and volatile. They point out that changes in requirements are inevitable and thus, the inherent flexibility of Agile methods should be seen as an advantage rather than something to be feared or requiring management with significant project overheads (Highsmith 2002). Agile methods emphasise stakeholder collaboration and interaction, while proposing different ways of handling communication between multiple parties (Beck 2000). Many of the methods also encourage the use of different development artefacts, such as prototypes or user stories, as a way of enhancing communication and collaboration between the stakeholders (Highsmith 2002). These artefacts can be understood as boundary objects that exist between different the collaborating parties.

Boundary objects need to be adaptable to the requirement of the different parties that use them and robust enough to maintain a common identity across contexts (Star and Griesemer 1986). Boundary objects play a vital role in development projects by enhancing common understanding and collaboration among a diverse set of participants (Barrett and Oborn 2010; Vlaar et al. 2008). Boundary objects and their usage have been studied extensively in domains such as product development (Carlile 2002), as well as engineering and architecture projects (for example Gal et al. 2008; Henderson 1991; Subrahmanian et al. 2003**).** Prior research on Agile development methods (for summary see for example Dingsoyr et al. 2012) has also explored the ways in which they should be applied. However, only two studies explicitly examined boundary objects in the context of Agile projects (Baskerville et al. 2011; Modi et al. 2013). Furthermore, these articles have not examined how boundary objects behave in the dynamic Agile project environment. The Agile projects provide a novel, more dynamic and thus potentially more challenging environment for boundary objects. In this environment the behaviour of boundary objects has not been observed. This can result in misapprehension of boundary objects in Agile project settings.

In Agile development environments, boundary objects are subjected to constant change that can erode their ability to facilitate collaboration (Subrahmanian et al. 2003). This can be an issue because Agile development relies heavily on communication and collaboration. Therefore, a better understanding of how project changes affect boundary objects in development projects is needed to help organisations that undertake Agile projects understand what objects are required and how they should be managed. The purpose of this Research-In-Progress paper is to investigate the boundary objects in the Agile environment and examine how change influences the objects.

This article in organized in the following way. The next section will discuss how change is viewed in Agile software development and in research on boundary objects. This is followed by a description of the pilot case study, the preliminary findings, and our discussion of the findings. The article then concludes with remarks on the potential contributions of our study, as well as future research directions and our limitations.

## AGILE PROJECTS AND BOUNDARY OBJECTS

### Change in Agile Project Development

Agile methods consist of collection of techniques that aim to provide an alternative to documentation, management and process- driven information systems development methods (Beck et al. 2001; Highsmith 2002). The most well-known Agile methods include Extreme Programming (Beck 1999), Scrum (Schwaber 2004), Lean development (Poppendieck and Poppendieck 2003) and Kanban (Andersen 2010). These originated from the iterative and incremental software development practices such as Spiral Model (Boehm 1988). The Agile methods came to the attention of the wider software development community when the Agile Manifesto was published by a group of developers in 2001. The manifesto promoted ideas of a human centric, and collaborative development environment. Another important source of influence for the methods came from Japanese product design practices (Beck 1999) that emphasise development speed and team flexibility.

One of the main differences between Agile methodologies and more traditional software development methods (e,g, Waterfall) is the attitude toward change. Agile methods challenge the traditional software development methodologies, which they perceived to be too rigid and change resistant (Larman and Basili 2003; Royce 1970). One of the key messages in the Agile Manifesto is that projects should prefer "responding to change over following a plan" (Beck et al. 2001). Advocates of Agile development were not convinced that a project can be fully documented or planned beforehand. While practitioners of traditional development methods see change as a potential source of risk and suggest an elaborate change-management process to mitigate the perceived adverse effect project changes, (Kerzner 2013), agile proponents recommend that change is "embraced" and incorporated into the project as a source of potential improvement (Beck 1999).

From this perspective, change can encourage creative solutions in development teams because they are not hindered by plans that may possibly be flawed. Such creativity can facilitate a faster creation of systems that customers value (Boehm and Turner 2005). This is obtained by maintaining a short release cycle of the system to ensure faster feedback (Schwaber 2004), which is possible because of the integration of the customer and the development team. After every software version release, the team plans what requirements are done in near future with the sole purpose of assuring that the customers get a working system with the most important requirements and the long-term plans are left deliberately vague.

Extant research on Agile development methodologies confirms that the methods can be beneficial in many areas of software development, including customer collaboration and communication (reviewed by Dybå and Dingsøyr 2008). To the extent that communication tools in distributed projects are discussed (Korkala and Abrahamsson 2007), they are viewed as static artifacts, which enable development (Kelter et al. 2003), rather than as pliable objects that promote information exchange and common understanding across boundaries.

Even though advocates of Agile methods propose using descriptive objects (e.g., prototypes) to facilitate communication across project participants (Highsmith 2002), the full scope of boundary objects and their implications for the participating organisations has generally not been acknowledged (Baskerville et al. 2011; Modi et al. 2013).

### Boundary Objects

Boundary objects reside in the interfaces between organisations or groups of people. They are defined as object that are able to adapt to local needs but also "robust enough to maintain common identity" (Star and Griesemer 1989, p 393). They allow different groups to work together without consensus (Star 2010). Indeed, collaboration among diverse groups or organisations has been at the centre of many studies on boundary objects. Researchers have examined how such tools were used by participants to foster collaboration in different settings such as setting up museum exhibitions (Lee 2007) and using project management tools such as timelines (Yakura 2002). Specific attention has been given to boundary objects in different types of engineering projects. Research has examined boundary objects in product engineering or product design (Carlile 2004; Subrahmanian et al. 2003) with case studies concerning CAD systems (Gal et al. 2008; Henderson 1991; Subrahmanian et al. 2003), machinery design (Karsten et al. 2001) and other computer aided design tools (Boujut and Blanco 2003). Some researchers have also studied boundary objects as communication tools in distributed project environment (Barrett and Oborn 2010; Vlaar et al. 2008).

Boundary objects continually evolve and change (Star 2010). They can be adjusted through group interaction towards the common goals of the groups (Henderson 1991). Changes in flow of information, in technologies and in organisational structures affect boundary objects (Subrahmanian et al. 2003). Because these changes can create mismatches in cross-boundary understanding that can result in loss of organisational performance, they also trigger changes in the boundary objects so that the organisations can avoid the losses.

Two main issues should be taken into account when changes in objects are concerned: first, to maintain the usability of the object as flexible collaboration medium; and second, to ensure that the object does not hold too much information or crosses too many boundaries. Too much information crammed in one object can happen when organisations create new boundary objects that accommodate new information better. The organisations that try to compensate for the losses of information by including more information into the new boundary objects (Subrahmanian et al. 2003) are in danger of losing the flexibility of the objects.

The loss of flexibility (Star and Griesemer 1989) is also a threat to the object when the object crosses too many boundaries and encompasses too many participants. Objects with too many meanings can lose their value to the involved parties by becoming overloaded with diverse meaning. The parties that were originally using the object may prefer to use workarounds and avoid using the confusing object (Henderson 1991).

Changes within the participating organisation can alter the boundary objects but the adoption of new boundary objects may lead to changes in the organisations that use them. The object can change the identity of the organisation as well as the practices used. The organisations can also try to facilitate the changes in the organisations that surround them by introducing objects in the boundaries between them and others (Gal et al. 2008).

Studies that look at changing boundary objects are scarce and they have been mostly conducted in engineering projects where the boundary objects have been engineering models or prototypes (Gal et al. 2008; Henderson 1991; Subrahmanian et al. 2003). Agile software development methods share some traits with engineering environment in these studies, for example the use of prototypes, but are also different in pace and stakeholder collaboration.

Because of the lack of studies of boundary objects in Agile development settings, the effects of constantly changing environments on the boundary objects is not well understood. The use of boundary objects as a tool that can share knowledge and achieve common understanding with relatively small amount of work in a short period of time, can correct the project from straying to wrong directions (Boehm 1988). Knowing how change affects such important objects can help projects struggling with implementing efficient communication across boundaries.

Consequently organisations undertaking Agile projects clearly need to understand the nature of boundary objects in development projects and how they should be maintained by project developers. To this end, a pilot study was carried out with the purpose to identify boundary objects and their effects in an Agile information systems development project setting.

## RESEARCH METHOD

Agile software development projects are labour intensive undertakings, which require participation from multiple stakeholders often across organisational and geographical boundaries (Cockburn and Highsmith 2001). As the involved stakeholders utilise and create boundary objects, and the understanding of the objects is subjective to each stakeholder (Star 2010), an interpretative and qualitative research approach was selected for our study (Walsham 2006). Preliminary data collection was conducted at two Australian organisations. The organisations were selected based on their customer-vendor relationship providing organisational boundaries and use of Agile practices. By adopting a project as our unit of analysis, it has also provided the premise for examining multiple boundaries between organisations, as well as the plausible interactions between a variety of boundary objects.

## Pilot case

Preliminary, data collection and research method refining pilot case study (Yin 2014) was conducted of an Agile software development project involving two Australian organisations. WebShop (a pseudonym) is a small organisation with a background in software development. It has an Internet based sales business and the main office of the WebShop is located in Sydney. DevTeam (a pseudonym) is a software development consultancy, specialising in Agile projects. The employees of the DevTeam are located in different cities across Australia and North America.

WebShop contracted DevTeam in late 2013 with the goal of building a software system that would replace an existing product currently used by WebShop internally. The system, if successfully implemented, would also be eventually sold to the corporate customers of WebShop as a Software-as-a-Service (SaaS). The management and core team members of WebShop had decided, based on their experience and personal preferences, to utilise Agile methods in the project. They were already familiar with different Agile methods, which led to a tailored combination of methods such as Scrum (Schwaber 2004) and Kanban (Andersen 2010) for this particular project.

The project core team consisted of the Product Owner and Chief Technology Officer from WebShop, as well as two developers, two designers and a technical project manager from DevTeam. The other stakeholders from WebShop included the Project Sponsor, the company board, as well as the internal clients and testers of the system. The Project Sponsor's role was to communicate the project's progress to the company board. The project board provided the necessary financials and approvals for the project schedule and vendor engagement. In line with the principles of Agile methods, the project core teams shared intermediate project results with the internal testers for feedback and quality assurance purposes during several project stages, such as the early planning stage, the beginning of development stage, as well as the mid-development stage.

During the project, the involved stakeholders from WebShop were all located in the same premises. This collocation enabled easy access for the core team members to the testers and vice versa. The stakeholders from DevTeam, on the other hand, were distributed across different locations. They met face-to-face less frequently with each other and with the stakeholders from WebShop. They also relied more heavily on communication and collaboration tools in their daily work.

## Preliminary Data Collection and analysis

A total of ten semi-structured interviews were conducted, of which three were with informants from DevTeam and seven with informants from WebShop (refer to Table 1). The interviews were conducted just before and immediately after the release of the first version of the product, spanning the period from the beginning of April to mid June 2014. The interviews took an average of 45 minutes and were mainly conduced in the premises of WebShop. DevTeam informants were interviewed via Skype or at premises selected by the interviewees. Even though WebShop had other ongoing software development projects, this study was limited only to the project where DevTeam was participating.

Table 1 The Interviews

| The WebShop | The DevTeam |
|---|---|
| **Project Management** | **Technical Development Team** |
| Chief Technology Officer | Technical Project Manager (Scrum Master) |
| Product Owner | User Experience Designer |
| Project Sponsor | Developer |
| **Other Stakeholders** | |
| Four Internal Customers and Testers | |

Interview questions were centred on the processes and tools used in the project. Project goals, potential challenges and stakeholder communication were also recurring topics. The interviewer took notes during the interviews and the interviews were also recorded and later transcribed. Snowball sampling (Stake 2010) was used to choose the interviewees. The snowball sampling technique was used uncover the main project stakeholders and to control the number of the interviewees (Biernacki and Waldorf 1981). The interviews consisted of a standard core of questions that were asked in each interview, as well as a number of questions that were tailored to the role of each informant.

Analysis of the data was performed through a mixture of open and axial coding (Strauss and Corbin 1990). Open coding was used to identify frequently occurring themes such as different artefacts, communication methods, stakeholders or events that unfolded over the duration of the project. Axial coding was then used to identify the relationships between these constructs.

## PRELIMINARY FINDINGS

The scope of the boundary objects studied is limited to the objects shared by the project core team (i.e. DevTeam and WebShop's product owner and CEO) and the WebShop internal testers. Three objects, in particular, were used to transfer information between the two parties: a wireframe prototype, an early version of the system and a spreadsheet used for logging testing results.

The first identified boundary object, the wireframe prototype, was created at the very beginning of the project. It was used as a preliminary communication and clarification tool for the initial phase of the project. The first functional version of the system, the Beta release, replaced the wireframe subsequently as the main shared item and the prototype was discarded. The WebShop testers who shared their testing feedback with the core team via the spreadsheet tested both the prototype and the first release of the system. Each object, and how they evolved during the project, is described in turn.

### The temporary boundary object: The wireframe prototype

WebShop had been collecting a list of initial requirements for the new system already prior to the start of the project. Based on these requirements, DevTeam created a preliminary visual representation of the system. In the software development vernacular, such a representation is usually called a wireframe. Project members, when referring to the prototype, also used this term. The wireframes were used to satisfy the communication needs between the project team and the rest of WebShop stakeholders in a non-ambiguous, visual manner. This simple but powerful visual representation of the desired system was deemed very useful and important to the project success. Similar use of boundary objects as simple visual representations has been described for example by Yakura (2002)

To enhance system usability, the wireframe prototype was subjected to one round of user testing with the internal WebShop testers. The testers included the members of the organisation who engaged with the old version of the system as part of their daily work. They had deep knowledge of what the old system was operation what it was lacking in usability or functions. Based on the testing session feedback, the wireframes were changed by the user experience designer to improve usability. The second version of the wireframes was also more interactive. Users could, for example, click from page to page to see how the different functional parts of the system fitted together. A developer from DevTeam explained:

"*After we came up with the first version, we took it to their users, so we had a first-round interviews, where we listen to what they wanted and then we brought this mock-up back and said 'ok, so this is what we have so far, let me know what you think'. We had a scenario for user testing, and yeah, that's what we did (…) I had few little scenarios that I went through and just gathering people's reactions, feedback, it was really helpful (...) So there was one round of changes. But we could have done more, the rest is probably, there was a lot more tweaks afterwards in visual design*."

The WebShop project management team indicated that they had originally doubted that the investment of time and money in such extensive wireframes was justified. But after seeing the results produced by DevTeam, they were convinced that the wireframes were worth the effort. The interactive features of the wireframes were complimented by everyone who had access to them. One of the project management team members also mentioned that the wireframes provided a clearer picture of what the system would be in the future. It also provided the management with assurance that the project was heading in the right direction. Boundary objects used as shared representation and as a way to stimulate mutual understanding has been discussed also for example by Boujut and Blanco (2003)

The wireframe prototype was never intended to be used for long. They were only meant to capture the visual look and feel of the system that would be subsequently enhanced by applying an additional layer of visual design. For the site developers in particular, the wireframes provided information on the interface of the system. Eventually the development progressed to the point where the wireframes were surpassed by other formats of presenting requirements information. The developers confirmed that a requirements management tool gradually became a more reliable source of requirements because it was constantly updated, whereas the wireframes were only updated once, after usability testing. The diminishing importance of wireframes follows similar patterns as neglecting of management boundary objects in research by Sapsed and Salter (2004) .

Later in the project lifecycle, the wireframes were still used as a static reference tool for WebShop, but they were no longer utilised as a boundary object between the testers and the project team. The first release of the system had replaced the wireframes as the main project communication object.

### The evolutionary boundary object: The Beta release

The wireframes were followed by a first version of an actual functional system: the Beta release. This system version was shared across organisation boundaries in line with Agile guidelines that prescribe multiple small releases (e.g. Beck 1999). The Beta release was the first version of the system that was subjected to the internal testing by the WebShop testers, which were composed of users familiar with the older system. Some of testers had already participated in the first round of user testing of the wireframes prototype. One of the testers described how the first release differed from the wireframes:

"*I think, obviously at the stage where I was looking at it wasn't intended to be ready to go on, I think it was them getting me a be the first tester of what they thought was a good system. I think it was just getting someone's input throughout the process that they were going thought and um, making sure that they were on the right track so not to get to the end and then realise that 'we should not have done this' and go back. Aah, so when I reviewed the current <software system>, or <new software system> since changed since it was first released to us, based on some of the feedback. When it was first released it was more user friendly, it was further developed form the initial stages*."

The development of the system did not cease after the Beta release. It continued during testing and new functionalities were constantly added for the testers to review. The goal after the Beta release was to produce a Minimum Viable Product, a version with all the crucial features, which could be potentially used as the basis for a commercial product. WebShop had planned to extend the testing of the system with the MVP version with external testers sourced from their pool of clients. This is to gain a different perspective on the requirements and further changes to make. After the MVP version, more steps were incrementally added to the project schedule. This resulted in more changes and the further evolution of the system.

During internal testing, the Product Owner was the main communicator between testers and DevTeam. To ensure that all feedback and new requirements were captured and clearly communicated between the stakeholders, a testing spreadsheet was used.

### The mediating boundary object: The testing spreadsheet

The wireframes and Beta release both had multiple functions. They were both used for communication and collaboration across organisational boundaries. The wireframes also served as visual development and design aid, while the Beta release was already a commercially viable product in itself. The last boundary object, the testing spreadsheet, differs from the other two because it was used only for the purpose of communicating a specific set of information (i.e. testing feedback). The spreadsheet was shared between the internal testers and the project core team, and contained results of testing and feedback. The Product Owner managed the spreadsheet and translated the information into a set of requirements or issues. These requirements were then shared with the developers via a requirements management tool. One of WebShop's internal testers explained:

"*That [testing spreadsheet] document was us providing [information to the Product Owner], it was more of a one way sort of feed of information. Um, anytime I needed sort of clarification on something (…). <Instant messaging tool> which is like an instant messaging program so we would just use that. Umm, or email or something if I thought that was needed. So once or twice I send him some strange odds via email and he just responded saying that that's all great, can you also add to the [testing spreadsheet] document so. Everything was recorded in that [testing spreadsheet] doc*."

The testing spreadsheet served to simplify communications between the testers and the core project team. With all the information consolidated within a single repository, the Product Owner had easier access to the test results and could share them with the rest of the core team more efficiently. As a part of the Agile development principle of small releases (Beck 1999), new versions of the system were constantly released for testing. The Product Owner communicated the major changes and project schedule to the internal testers. All interviewed testers were satisfied with extent of the information they were given about the project schedule, main milestones and the testing process.

As a boundary object the testing spreadsheet had a limited lifespan. It was used for boundary spanning communications only during the testing phase. After testing was completed, the spreadsheet was archived for reference like the wireframes. The changes made to the spreadsheet were highly granular as every feedback item resulted in a change.

The source of the object change was also different compared to the other boundary objects. The project core team changed both wireframes and the Beta release but spreadsheet was modified by the internal testers. The main difference between the two is experience in software development. The core team was well versed in software development and required less guidance and external control when conducting changes. On the contrary, the internal testers required ease of access and guidelines for submitting their changes. By providing the simple feedback forum and directions for submission of changes, the Product Owner ensures that the internal testers were able to perform their assigned tasks. The combination of the testing spreadsheet and the Beta release ensures that communication was flowing both ways across the boundary, from the core team to testers and back.

## DISCUSSION

In our pilot study, we uncovered three types boundary objects that may be crucial in Agile software development projects. Moreover, each type of boundary objects may serve a different purpose, and react differently to change.

The first type of boundary objects is temporary boundary object. These objects are created to serve a specific purpose for a limited period of time. The temporary boundary objects success depends on its ability to clarify and share understanding across boundaries. After this objective is accomplished, the object is intentionally discarded. Change is important to the object only if it makes the object better at conveying information across the boundary. Temporary boundary object can be a powerful tool in Agile project for it’s ability to contain information that aids key principles of Agile methods: the stakeholder collaboration and interactions (Beck et al. 2001).

The second type of boundary objects is an evolutionary boundary object. The evolutionary boundary objects undergo several iterations in their life span. These objects are most threatened by the change erosion (Subrahmanian et al. 2003) because they should sustain the changes and maintain their original identity to retain their functionality. Devoted use of Agile practices that encourage constant customer co-operation, communication and feedback (Beck 2000) should alert the organisations if the object is failing to accommodate changes.

The third type of boundary objects is mediating boundary object. These objects are used to distribute information across boundaries and to ensure stakeholder participations. The difference between temporary and mediating boundary object is that several stakeholders with different experience levels change mediating boundary objects. This makes them more vulnerable to overloading which can reduce their meaningfulness (Henderson 1991). Using mediating objects in Agile projects is a technique to enhance the communication between stakeholders.

Next section discusses future research, limitations and contributions of these research findings.

## CONCLUDING REMARKS

With our preliminary findings, our study has contributed to a better understanding of the types of boundary objects that should be taken into consideration when planning and maintaining Agile software development projects, and how they evolve over time. It also contributes to a richer understanding of change in the Agile project setting by summarising the literature of change and linking it to concrete examples.

The preliminary findings were gathered from a pilot case study. The purpose of the pilot study was to gather data that could be used to test the research approach and refine the research questions (Yin 2014). Our future research direction will include gathering more data from other case organisations with different goals and boundaries between parties for cross-case analysis. We will also revisit some of the key stakeholders in the pilot study after the release of the system to external testers, extending the scope of our study for a more complete view the nature and evolution of boundary objects across the project life cycle. The relationships of the boundary objects with other objects will also be further examined. Inclusion of case organisations with similar backgrounds but different project goals and boundary objects would also complement our existing data for the purpose of triangulation (Eisenhardt 1989) as well as improving validity and reliability of our results by providing data for multiple interpretations (Klein and Myers 1999). The study is now limited to one case project and one set of Agile software development methods. The data collection method was also limited to interviews. Future data collection could include for example study of secondary data.

The understanding of how the nature and evolution of boundary objects a successful agile development project could shed light on why stakeholders in some Agile projects are less successful than others in communicating their requirements across boundaries. Our study contributes to research on Agile methods by providing the foundation for a new theoretical lens for the study of change and objects of Agile software development. To the literature on boundary objects, our research provides a glimpse of how they operate in a new domain, that is, the volatile Agile project environment which has not been extensively studied to date.

## REFERENCES

Andersen, D. J., 2010. *Kanban*. Blue Hole Press.

Barrett, M. and Oborn E., 2010. “Boundary object use in cross-cultural software development teams,” *Human Relations*, (63:8), pp 1199–1221.

Baskerville, R., Pries-Heje, J., and Madsen, S., 2011. “Post-agility: What follows a decade of agility?,” *Information and Software Technology*, (53:5), pp 543-555.

Beck, K., 1999. “Embracing change with extreme programming,” *Computer*, (32:10), pp 70–77.

Beck, K., 2000. *Extreme programming explained: embrace change.* Addison-Wesley Professional

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001). “Manifesto for Agile Software Development” *Manifesto for Agile Software Development.*

Biernacki, P., and Waldorf D., 1981. "Snowball sampling: Problems and techniques of chain referral sampling," *Sociological methods & research* (10:2) pp 141-163.

Boehm, B., 1988. “A Spiral Model of Software Development and Enhancement,” *Computer*, (21:5), pp 61–72.

Boehm, B., and Turner, R., 2005. “Management challenges to implementing agile processes in traditional development organizations,” *Software, ieee* (5:22), pp 30-39.

Boujut, J., and Blanco E., 2003. “Intermediary objects as a means to foster co- operation in engineering design,” *Computer Supported Cooperative Work (CSCW)*, (12:2), pp 205–219.

Bustard, D., Wilkie G., and Greer D., 2013, “Towards optimal software engineering: learning from agile practice,” *Innovations Syst Softw Eng*, 9, pp 191–200.

Carlile, P. R., 2002. “A pragmatic view of knowledge and boundaries: Boundary objects in new product development,” *Organization science*, (13:4), pp 442–455.

Carlile, P. R., 2004. “Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries,” *Organization science*, (15:5), pp 555–568.

Cockburn, A. and Highsmith J., 2001. “Agile Software Development: The People Factor,” *Computer*, (34:11), pp 131-133.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B., 2012. “A decade of agile methodologies: Towards explaining agile software development,” *Journal of Systems and Software*, (85:6), pp1213-1221.

Dybå, T., and Dingsoyr, T., 2008. “Empirical studies of agile software development: A systematic review,” *Information and Software Technology*, 50, pp 833–859.

Eisenhardt, K. M., 1989. "Building theories from case study research," *Academy of management review* (14:4), pp 532-550.

Gal, U., Lyytinen, K., and Yoo, Y., 2008. “The dynamics of IT boundary objects, information infrastructures, and organisational identities: the introduction of 3D modelling technologies into the architecture, engineering, and construction industry,” *European Journal of Information Systems*, (17:3), pp 290–304.

Henderson, K., 1991. “Flexible sketches and inflexible databases: Visual communication, conscription devices, and boundary objects in design engineering,” *Science, technology & human values*, (16:4), pp 448–473.

Highsmith, J. A., 2002. *Agile software development ecosystems*, Addison-Wesley Longman Publishing Co., Inc.

Highsmith, J. and Cockburn, A., 2001. “Agile software development: The business of innovation;” *Computer*, (34:9), pp 120–127.

Karsten, H., Lyytinen, K., Hurskainen, M., and Koskelainen, T., 2001. “Crossing boundaries and conscripting participation: representing and integrating knowledge in a paper machinery project,” *European Journal of Information Systems*, (10:2), pp 89–98.

Kerzner, H. R., 2013. *Project management: a systems approach to planning, scheduling, and controlling*, John Wiley & Sons.

Klein, H. K, and Myers, M. D., 1999. "A set of principles for conducting and evaluating interpretive field studies in information systems," *MIS quarterl*y, pp 67–93.

Korkala, M., And Abrahamsson, P., 2007, "Communication in distributed agile development: A case study," *In Software Engineering and Advanced Applications,* IEEE *33rd EUROMICRO Conference on,* August, pp 203-210.

Larman, C., and Basili, V.R., 2003. "Iterative and incremental developments, a brief history," *Computer*, (36:6), pp 47–56.

Lee, C. P. 2007. "Boundary negotiating artifacts: Unbinding the routine of boundary objects and embracing chaos in collaborative work,*"* *Computer Supported Cooperative Work (CSCW)*, (16:3), pp 307–339.

McHugh, O., Conboy, K., and Lang, M., 2012. "Agile Practices: The Impact on Trust in Software Project Teams," *IEEE Software*, (29:3), pp 71–76.

Modi, S., Abbott P. and Counsell S., 2013. "Negotiating common ground in distributed agile development: A case study perspective", *In: proceedings of 2013 IEEE 8th International Conference of Global Software Engineering (ICGSE),* pp 80-89

Poppendieck, M., and Poppendieck, T., 2003. *Lean software development: an agile toolkit,* Addison-Wesley Professional.

Royce, W. W., 1970. "Managing the development of large software systems," *In: proceedings of IEEE WESCON, vol. 26*. Los Angeles.

Sapsed, J. and Salter A., 2004. "Postcards from the edge: local communities, global programs and boundary objects," *Organization Studies*, (25:9), pp 1515–1534.

Schwaber K., 2004. "*Agile project management with Scrum*." O'Reilly Media, Inc.

Star, S. L., 2010. "This is not a boundary object: Reflections on the origin of a concept," *Science, Technology & Human Values*, (35:5), pp 601–617.

Star, S. L., and Griesemer, J. R., 1989. "Institutional ecology,translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39," *Social studies of science*, (19:3), pp 387–420.

Strauss, A. and Corbin, J. M., 1990. *Basics of qualitative research: Grounded theory procedures and techniques*, Sage Publications, Inc.

Subrahmanian, E., Monarch I., Konda S., Granger H., Milliken R. and Westerberg, A., 2003. "Boundary objects and prototypes at the interfaces of engineering design," *Computer Supported Cooperative Work (CSCW),* (12:2), pp 185–203.

Walsham, G., 2006. "Doing interpretive research," *European journal of information systems*, (15:3), pp 320–330.

Vlaar, P. W., van Fenema, P. C. and Tiwari, V. 2008. "Cocreating understanding and value in distributed work: How members of onsite and offshore vendor teams give, make, demand, and break sense," *MIS quarterly*, (32:2), pp 227-255.

Yakura, E. K., 2002. "Charting time: Timelines as temporal boundary objects," *Academy of Management Journal*, (45:5), pp 956–970.

Yin, R. K., 2014. *Case study research: Design and methods,* Sage publications.

## ACKNOWLEDGEMENTS

We would like to acknowledge all the great people from both case study organisations who volunteered their time to further this study.

## COPYRIGHT